Exploring and Theorizing Velocity Flux in Agile Development

Supervisor: Jonas Sjöström

Master Student: Wenfei Dong

wenfei_dong@hotmail.com

Master Thesis in Information Systems

Department of Informatics and Media

Uppsala University

Sweden

08-06-2015
ABSTRACT

We mainly study development velocity in agile teams in this dissertation. The concept of development velocity relates to the classical problem of time estimation in software development and software development planning. Building on previous literature as well as a case study, we explore and theorize the factors that cause ‘velocity flux’, i.e. fluctuations in development velocity through studying the relationship between development velocity and the rate of incoming customer feature requests. The aim of this study is to contribute to a better understanding of what causes velocity flux in agile development, and discusses the implications of the findings for research and practical implications for agile planning. As a result, we propose nine factors that cause velocity flux, and provide some strategies to overcome them in order to make a more effective sprint planning in agile teams.

Keywords: agile development team, development velocity, incoming customer requests, sprint planning, and productivity
ACKNOWLEDGEMENTS

First of all, I am very grateful for the constant and helpful feedback and guidance from my supervisor, Jonas Sjöström. During the process of completing this dissertation, he always gives valuable guidance and encouragement. I would say the completion of this work is indispensable with the help from my supervisor.

My grateful thanks are also extending to the agile development team in U-CARE for their kind assistance and support of this study.

I would also like to thank all my teachers at Uppsala University. Special gratitude is given to Steve Mckeever, who provided guidance in thesis writing. Grateful thanks are also given to Anneli Edman, who devoted herself to education and student care.

Last but not least, I really appreciate the financial support from the Swedish Institute and I am grateful to my family and friends who are always encourage and understand me.
# TABLE OF CONTENTS

1. **INTRODUCTION** .......................................................................................................................... V  
   1.1 BACKGROUND ............................................................................................................................. 3  
      1.1.1 Development velocity ............................................................................................................. 3  
      1.1.2 Customers in agile development ............................................................................................ 4  
   1.2 RESEARCH QUESTIONS ................................................................................................................. 6  
   1.3 INTERESTED AUDIENCE .............................................................................................................. 6  
   1.4 DELIMITATION .............................................................................................................................. 6  
   1.5 DISSERTATION OUTLINE ............................................................................................................. 7  

2. **RESEARCH DESIGN** ......................................................................................................................... 8  
   2.1 RESEARCH DESIGN OVERVIEW ................................................................................................. 8  
   2.2 LITERATURE REVIEW ................................................................................................................... 9  
   2.3 CASE STUDY ............................................................................................................................... 10  
      2.3.1 Conducting case studies: data source ...................................................................................... 11  
   2.4 DATA ANALYSIS .......................................................................................................................... 13  
      2.4.1 Analysis of quantitative data ................................................................................................ 13  
      2.4.2 Analysis of qualitative data ................................................................................................... 14  
   2.5 CHAPTER SUMMARIZATION ........................................................................................................ 17  

3. **LITERATURE REVIEW** ..................................................................................................................... 18  
   3.1 FLUCTUATION OF VELOCITY ..................................................................................................... 18  
   3.2 RELATIONSHIP BETWEEN CUSTOMER AND DEVELOPMENT VELOCITY .............................. 20  
      3.2.1 How feature requests affect the development team and development velocity ................... 20  
      3.2.2 How velocity flux affects how stakeholders request new features .................................... 23  
   3.3 MOTIVATION FROM LITERATURE REVIEW .............................................................................. 23  

4. **CASE STUDY** .................................................................................................................................. 24  
   4.1 THE FOCUS OF THE CASE STUDY ............................................................................................... 24  
   4.2 BACKGROUND OF THE CASE STUDY ......................................................................................... 24  
   4.3 DATA COLLECTION ..................................................................................................................... 25  
      4.3.1 Database ............................................................................................................................... 25
List of Figures

Figure 2.1 Research Design .............................................................................................................. 9
Figure 2.2 Data Abstraction Process (Source: Hoda, Noble & Marshall, 2010) .................. 15
Figure 3.1 Amount of Time Spent on Tasks (Source: Cohn, 2010) ........................................ 22
Figure 4.1 Line Chart about TDV and TRV ............................................................................... 28
Figure 4.2 Six Cs Coding Family ................................................................................................. 33
Figure 4.3 Factors with Indirect Impact on Velocity ................................................................. 34
Figure 4.4 Suggestions Based on the Indirect Causes............................................................... 39
Figure 4.5 Suggestions Based on the Direct Causes of Velocity Flux ..................................... 41

List of Tables

Table 2.1 Coding Families (Source: Glaser, 1978) ................................................................. 16
Table 4.1 Original Dataset ............................................................................................................ 26
Table 4.2 New Dataset ................................................................................................................ 26
Table 4.3 Constructs in Use to Analyze Correlations ................................................................. 26
Table 4.4 Correlation Results ..................................................................................................... 29
Table 5.1 Findings from Case Study Compared to Literature Review Result ...................... 46
1. INTRODUCTION

Agile Project Management is one of the revolutionary methods introduced to software development management to provide flexible software development practices and principles (Abrahamsson, 2002). Agile development caters to the growing needs of productivity in the software industry. Nowadays, agile software development methods have become more and more important in the field of software development. Ericksson et al. (2005, p.89) defined agility as “strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like.”

The process of agile development is based on iterative and incremental development, where requirements and solutions are evolved through collaboration between customers and development teams (Schwaber & Beedle 2002). Agile methods promote adaptive planning, and encourage rapid and flexible response to change. There are many different software development methods considered as agile development methods since different methods embrace different characteristics and are suitable to different cases (Sillitti & Succi, 2005). These agile software development methods are: Extreme Programming (XP), Scrum, Adaptive Software Development (ASD), Dynamic System Development Method (DSDM), Feature Drive Development and so on.

Agile principles suggest a focus on people and interactions (Conboy, 2009), and its implications for development, e.g. team management, and time and resource management (Erickson, Lyytinen & Sias, 2005). Agile development - as a contrast to plan-driven methods – is characterized by the prevalence of developer-customer interaction, and the fundamental idea that change should be embraced in the development process (Zaretskiy & Serbin, 2009). Various agile practices, e.g. ‘on-site collaboration’ and the planning of sprints, were introduced to improve the interaction between stakeholders, and ‘embrace change’ under controlled forms (Abrantes & Travassos, 2011).
According to Leffingwell (2011), most agile teams use the concept of development velocity to estimate productivity. It is described as an effective tool for management teams and customers to estimate schedule and cost. “By calculating development velocity, it is easier to know where the project is and predict where the team will be next ”(Leffingwell, 2011, p.193). Ideally, an agile team has a stable velocity that increases over time, to facilitate reliable sprint planning (Schwaber & Beedle 2002). Stable development velocity is one of the most important process indicators for developers in agile software development methods. If development velocity does not change too much in a sprint, then this means that the working team and the process of development are well aligned (Omanovic & Buza, 2013). In this paper, we are driven by a phenomenon observed within an agile project: development velocity is not always keeping stable or increasing over time. For simplicity, we introduce the term ‘velocity flux’ to refer to development velocity fluctuation. Previous research on the topic provides some explanations for velocity flux, such as unexpected interruptions, high work in progress, team dynamics and rework, and hidden complexity. While these four explanations listed above appear plausible, our literature review suggests that more research is needed to understand the dynamics between customers and developers, and its implications for productivity in an agile team.

To summarize, the purpose of this study is to further explore and theorize factors that cause velocity flux through investigating its relationship with the pace of incoming customer feature requests. We aim to contribute to agile software development methods, especially for sprint planning in this study.

In this chapter, the background of this study will be first provided in order to state the research domain and present relevant concepts. Then the research questions, interested audience and delimitation of this study will be introduced. In the Section 1.5, the outline of this dissertation will be presented.
1.1 Background

This study mainly investigates the causes of velocity flux and its relationship with the pace of incoming customer feature requests as stated in above. Explanations about why we are interested in these concepts – development velocity and customers- are presented in the following sub-sections.

1.1.1 Development velocity

In agile development, development velocity measures how many features an agile team delivers in a Sprint (Cohn, 2014). However, there is another interpretation of development velocity, which is related to complexity points: “a team’s velocity is a representation of how many complexity points that a team can deliver within a Sprint” (Moreira, 2013, p.192). In this dissertation, we use the first definition of development velocity and will not take complexity points into consideration. There are two reasons for our choice:

1) Complexity points are subjective estimates or guesses of developers on the time or effort that is needed to complete a task. The measurement thus becomes quite rough due to the lack of unified criteria to make complexity estimations – every developer has his/her own criteria to make the guess (Schwaber & Beedle 2002).

2) Our datasets, which will be used in the case study for quantitative analysis, would reduce significantly if we wanted to analyze complexity points, since such data only exists for approximately half the time period we are currently analyzing.

According to Leffingwell (2011), most agile teams use the concept of development velocity to estimate productivity. It is described as an effective tool for management teams and customers to estimate schedule and cost. According to Leffingwell, 2011, there are three reasons to support the importance of using velocity to make estimation in agile software development:
1) **Determining cost**: cost serves as the base of doing business. By estimating the time and effort of completing a task based on team development velocity, customers could estimate the cost and make a plan for their investment.

2) **Establishing prioritization**: tasks will be prioritized according to team velocity during sprint planning meetings. Developers could determine which unit of tasks fits into the current sprint best by taking their previous development velocity into consideration.

3) **Scheduling and commitment**: estimation of task completion time based on previous development velocity could help development teams to make commitments to customers about their deliverables in next sprint during sprint planning meetings. The commitments made by developers can affect the planning and business objectives of customers.

The discussion above indicates that keeping a stable velocity is important in agile teams and the investigation based on development velocity will provide useful insights and help make a more effective sprint planning. In the sub-Section 1.1.3, we will present why we are interested in studying the rate of incoming customer feature requests within the scope of agile development.

**1.1.2 Customers in agile development**

In traditional software development projects, customer involvement is limited to providing the requests in the beginning and feedback in the end, without any immediate communication between customers and the development team (Sillitti et al., 2010). However, customers are considered to be an important role in agile software development. According to Sillitti and Succi (2005, p.316), “the term ‘customer’ identifies a set of stakeholders that belongs to the organization that is paying for the development of a software product”. In agile development, these stakeholders reduced in to one on-site customer, who works with the development team during the whole development phase. Customer activities in agile are explained as follows in order to show their contribution in
agile development:

- **Build a customer team.** Before a project starts, a customer team should be built. The configurations of the on-site customer group need to be decided and a customer representative - product owner - should be selected (Schwaber & Beedle, 2002).

- **Create a well development environment.** Creating a good working environment for the development team and on-site customer to work independently or together is necessary (Martin, Biddle & Noble, 2010).

- **Give feedback and requests the project plan.** Customers should be able to review the project plan before / during sprint planning meetings and give feedback at the appropriate sprint review meeting or during the appropriate sprint (Sillitti et al., 2010).

- **Manage feature requests.** Customers send out new requirements at the beginning of each sprint, and confirm priority levels of them (Wang, Wu & Zhao, 2008). These feature requests will be put into product backlogs and will be completed according to the priority level.

- **Participate in testing.** Customers should provide test cases based on the real software operation environment to development teams, join integration tests and operate currently released software in order to find bugs in the software (Wang, Wu & Zhao, 2008, p.3).

- **Trace the whole development process.** It is better for customers to know the progress of current project and difficulties in every development process in order to identify changes and trace the progress of the changes (Martin, Biddle & Noble, 2010; Wang, Wu & Zhao, 2008).

The description of customer activities in agile in the above paragraphs implies that some customer activities, such as managing feature requests and participating in the testing, require customers to interact with the development team. We narrowed the research field
down and put emphasis on studying the pace of incoming customer feature requests and its association with development velocity in this study.

1.2 Research Questions

Research questions make theoretical assumptions more explicit (Cornford & Smithson, 2006). The research questions of this study are:

- Is there any relationship between development velocity and the pace of incoming customer feature requests?
- What are the causes of the relationship between development velocity and the pace of incoming customer feature requests?
- How to contribute to the current agile software development methods based on this study, especially for sprint planning?

Directed by these research questions, nine factors that directly and indirectly cause velocity flux are proposed, and eight strategies that can be used to overcome velocity flux in an agile team are provided. Our findings stress the need for an encompassing view on velocity. Besides, we stress the need to include a planning for learning and quality assurance – and to also report on such activities in sprint reviews.

1.3 Interested Audience

The audience for the topic in this dissertation could be:

- Organizations/teams working or plan to work with agile software development methods;
- Organizations/teams trying to stabilize or improve development velocity;
- Organizations/teams aiming to make a more effective sprint planning.

1.4 Delimitation

As mentioned before, agile software development methods is a quite broad topic. In this dissertation, we will narrow it down and focus on velocity flux and its relationship with
the pace of incoming customer feature requests to contribute to sprint planning in agile development.

1.5 Dissertation Outline

Chapter 1, INTRODUCTION, provides the problem domain, definition of relevant concepts and motivation of conducting this study. The research questions, interested audience and delimitation of this study are also presented.

In Chapter 2, RESEARCH DESIGN, we will give a comprehensive description about the design of our research, and the data collection and analysis methods that will be adopted in this dissertation.

Then, in Chapter 3, we will use LITERATURE REVIEW as a data collection method to find out what has been investigated regarding development velocity flux, the pace of incoming customer feature requests and their relationship in previous studies.

In Chapter 4, CASE STUDY, quantitative analysis will be applied to investigate the association between development velocity and the pace of incoming customer feature requests based on the collected data. Moreover, we will conduct interviews with different roles that work in an agile team, which will be highly helpful in order to interpret quantitative analysis results and generate a new theory.

In the last chapter, the Chapter 5, CONCLUDING DISCUSSION, we will answer our research questions and present contributions, (1) added knowledge about factors that cause velocity flux, and (2) another layer of abstraction to theorize causes of velocity flux (indirect vs. direct factors). Limitations of this study and future work for further study will also be presented.
2. RESEARCH DESIGN

After the research questions and goals have been settled in Chapter 1, it is important to plan and organize the research design. As suggested by Toledo (2012), the research design should be clear, possible to fulfill and follow logic. In this chapter, the design of this study, including the data collection and analysis methods that will be used in this study, will be described.

2.1 Research Design Overview

The research setting for our work is an inter-disciplinary research program in an e-Health context, namely U-CARE. The project is carried out by practitioners (e.g. software developers) in collaboration with academics from psychology, medicine, information systems, caring sciences and economics. The overall goal of the project is to study treatment efficacy and health economic aspects of online psychosocial care for patients with somatic disease. One aspect of fulfilling the overall project goal is to develop software to support online psychosocial care. The software has evolved continually since early 2011. In average, three full-time developers have been working in the development team. At present there are three full-time developers, supported by two PhD students who work part-time with software development. In total, 10 developers have contributed to the development. The software is mid-sized, consisting of four subsystems, has around 50K LOC and ~100 database tables. Our empirical study is based on the agile software development process in the project.
Figure 2.1 Research Design

Figure 2.1 states that the research is divided into four stages, and they are: i) Project Planning, ii) Data Collection, iii) Data Analysis, and iv) Conclusion. The first stage is project planning, in which we defined the research questions and goals (in Chapter 1) after reviewed previous results (literature review). The second stage is data collection. In this stage, we will collect data from previous work, the U-CARE database and interviews (in a case study). Then we will adopt both quantitative and qualitative methods to analyse the collected data in stage 3. In the end, we will discuss the results and the contribution of this study.

In the following sections of this chapter, we will give an explicit explanation about why and how we will conduct literature review and a case study in this dissertation.

2.2 Literature Review

As demonstrated in Figure 2.1, literature review helps in both project planning and data collection stage. Literature review provides inspirations to define the research topic and goals before conducting the research and this is what we have done in Chapter 1. Besides, it presents evidence to support the new knowledge that we try to investigate in the research (Oates, 2005). Therefore, literature review as a data collection method in this dissertation,
aims to review previous publications to address the questions listed as follows:

(i) What are the possible reasons that cause velocity flux in agile development?
(ii) How will the pace of incoming customer feature requests and development velocity influence one another?

Resonating with (i) and (ii), the keywords used to support literature search are: agile development, development velocity and customer feature requests. Searches will be conducted using Google Scholar and Uppsala University Library (a Meta search tool that forwards queries into most major databases). Books, articles, journals and conferences will be included. The content of the articles identified in the search process will be analysed keeping in focus the aim of the literature review.

2.3 Case Study

Case study, as one of the useful research strategies, has been used in many fields, such as sociology, political science, social work and community planning (Yin, 2003). According to Oates (2005, p. 141), “a case study focuses on one instance of the ‘thing’ that is to be investigated” and this case will be studied in depth by using a variety of data generation methods, like interview.

A logic plan serves as a bridge to link study research questions and conclusions. Therefore, a detailed plan for the case study is necessary. According to Yin, 2003, there are five important components for the design of a case study:

(1) Study questions: clarify precisely the nature of the study is an important start. In this case study, the study questions are same as the dissertation’s research questions.
(2) Study propositions: direct the attention to the ‘thing’ that needs to be examined within the scope of study. We aim to discover the factors that cause velocity flux and these factors are the ‘things’ that will be examined in this case study.
(3) Unit of analysis: defines what the ‘case’ is. The ‘case’ in this dissertation is a community that adopting agile software development methods in their project, namely U-CARE.
(4) Logic linking data to propositions: decides which kind of data should be collected in order to match study propositions. As in our case, data will be collected from the U-CARE database, and interviews with the U-CARE agile team members.

(5) Criteria for interpreting the findings: In this case study, quantitative analysis will be conducted. Grounded Theory will be applied to interpret the data collected from interviews.

2.3.1 Conducting case studies: data source

According to Yin, 2003, there are six important sources of evidence can be used in a case study: documentation, archival records, interviews, direct observation, participant-observation, and physical artefacts. In our case study, we choose the evidence of archival records and interviews within U-CARE due to the advantages as follows, according to Yin (2003):

1) Archival records (Database):
   - Stable - can be reviewed repeated
   - Exact - contains exact names, references and details of an event
   - Broad coverage - long span of time, many event, and many settings
   - Precise and quantitative

2) Interview:
   - Targeted - focuses directly on case study topic
   - Insightful - provides perceived causal inferences

Source of Archival records (Database-product backlog)

In this case study, the archival record (database) that is used to record the U-CARE development progress, works as the foundation to conduct extensive retrieval and quantitative analysis. Professional statistical analysis software, such as SPSS, will be used to facilitate the analysis process. One thing that draws attention according to Yin (2003) is that most of the archival records are created for specific reasons and audience, but not for the case study investigator. Therefore, interpreting or categorizing the original database
might be necessary.

Source of Interview

According to Robson (2002) researchers usually choose interviews as a data collection method in psychology and sociology fields. Since this study aims to uncover the causes of velocity flux, interviews are a suitable data collection method to support in-depth qualitative investigation. Before conducting interviews in this study, there are some issues that need to be settled down:

1) **How many interviewees should be invited?**
   In order to collect sufficient but not redundant information, we plan to invite the three full time developers and the product owner in the U-CARE agile team to our interviews.

2) **Who are they and how to contact them?**
   Developers could provide ideas regarding the research questions from their previous working experience, and the product owner as the representative of customers in U-CARE, could present valuable information from the perspective of management and customers. My supervisor, Jonas Sjöström, who used to be the technical management in the U-CARE agile team, will provide the contact information of the interviewees.

3) **Where should we conduct interviews?**
   All the interviewees can decide the places for their interviews. They can select a place where they feel comfortable and close to their working place.

According to the book “*Real world research*” from Robson (2002), there are three different interview types based on different interview structures. The extreme one is called “*fully structured interview*”, in which all the interview questions are set beforehand and the responses are recorded on a standardized schedule (Robson, 2002). The medium one is called “*semi-structured interview*”, in which some questions will be set before the interview, but it is possible to modify the order of the questions, change the way to
represent the questions, give explanations, delete or add some other questions based on the situation of a particular interviewee. The third type is called “unstructured interview”, in which the interviewer could only have a general interest but no specific questions prepared. In this dissertation, we apply semi-structured interview in order to give more space for the interviewer and interviewees to express themselves without digression.

2.4 Data Analysis

According to Oates (2005), well-established mathematical and statistical procedures can facilitate quantitative data analysis to uncover the hidden patterns and themes within the data. Qualitative analysis is useful to study one problem in depth (Oates, 2005). Therefore, in order to analyse the collected data thoroughly, we will adopt both quantitative and qualitative analysis methods in this dissertation.

2.4.1 Analysis of quantitative data

The basic idea of the quantitative analysis in this study is to discover the relationship between velocity flux and the pace of incoming customer requests based on the collected quantitative data. The quantitative analysis process in this dissertation mainly follows the steps listed as below:

1) Sort the collected data according to certain criteria;
2) Generate useful figures and tables to support analysis;
3) Statistical analysis will help verify whether the relationship that is under investigating does /does not exist. In our case study, a correlation test will be conducted.

The statistical analysis in this study will be supported by professional statistical software - SPSS. SPSS is one of the world’s earliest statistical software driven by graphics menu interface (Green & Salkind, 2010). The data interface of SPSS is relatively common since SPSS adopts the way similar to Excel spread sheet to input and management data. The basic function of SPSS includes statistical analysis, data management, chart analysis, output management and the like (Norusis, 2007).
2.4.2 Analysis of qualitative data

Grounded Theory (GT) will be applied to the qualitative analysis in this study. In 1967, Barney Glaser and Anselm Strauss created GT as a way to extend the ideas of research questions and produce a new theory on the results (Böhm, 2004). According to Oates (2005, p.274): “Grounded theory is a particular approach to qualitative research where the intention is to do field research and then analyse the data to see what theory emerges, so that the theory is grounded in the field data”. Böhm (2004) also stated that GT is suitable for the purpose of formulating a valid theory and producing an explanation and a description of the social phenomena that is under investigation, which is catering to our research purpose.

Glaser (1978, 1992) and Strauss (1987; Strauss and Corbin, 1990) provided us detailed analysis strategies of GT, even though there is some different practical details subsequently appeared. However, the core set of analytic strategies for the formulation of a theory remains as common ground to all accounts of the method (Pidgeon & Henwood, 1997). In this dissertation, we will apply the strategies provided by Glaser, which put theoretical coding as the last stage of the analysis instead of following the coding paradigm (open coding, axial coding and selective coding) as suggested by Strauss and Corbin (2014). We listed the stages that are used for grounded theory in this study:

1) Open coding: break down and create labels (codes) for chunks of data.
2) Constant comparison method: After open coding, codes will be constantly summarized and compared in order to develop a higher level of abstraction, called concepts. These concepts will be used as building blocks for the creation of a theory. Same method will be repeated to produce an even higher level of abstraction, called category (Hoda, Noble & Marshall, 2010). Categories will be compared further together, until no new categories can be discovered (Kan & Parry, 2004). The process of data abstraction is shown in Figure 2.2.
3) Writing of Memos: writing theoretical memo based on the codes, concepts and categories to explore emerging theories and connections to the existing theory (Hoda, Noble & Marshall, 2010);

4) Sorting: a constant process of writing and revision of the theory to more theoretical levels in order to uncover the relationships between codes, concepts and categories.

5) Generating a theory (theoretical coding): conceptualizing the relationship between categories and integrating them into a theory.

<table>
<thead>
<tr>
<th>Coding families</th>
<th>Concepts</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Six Cs</td>
<td>Causes, contexts, contingencies, consequences, conditions</td>
<td>… of pain suffering</td>
</tr>
<tr>
<td>Process</td>
<td>Stages, phases, phasings, transitions, passages, careers, chains, sequences</td>
<td>Career of a patient with chronic pain</td>
</tr>
<tr>
<td>The Degree Family</td>
<td>Extent, level, intensity, range, amount, continuum, statistical average, standard deviation</td>
<td>Extent of pain suffering</td>
</tr>
<tr>
<td>Type Family</td>
<td>Types, classes, genres, prototypes, styles, kinds</td>
<td>Kinds of pain – sharp, piercing, throbbing, shooting, sting, gnawing, burning</td>
</tr>
<tr>
<td>The Strategy Family</td>
<td>Strategies, tactics, techniques, mechanisms, management</td>
<td>Coping with pain</td>
</tr>
<tr>
<td>Interactive Family</td>
<td>Interaction, mutual effects, interdependence, reciprocity, symmetries, rituals</td>
<td>Interaction of pain experience and coping</td>
</tr>
<tr>
<td>Identity-Self Family</td>
<td>Identity, self-image, self-concept, self-evaluation, social worth, transformations of self</td>
<td>Self-concepts of pain patients</td>
</tr>
<tr>
<td>Cutting-Point Family</td>
<td>Boundary, critical juncture, cutting point, turning point, tolerance levels, point of no return</td>
<td>Start of chronification in the medical career of pain patient</td>
</tr>
<tr>
<td>Cultural Family</td>
<td>Social norms, social values, social beliefs</td>
<td>Social norms about tolerating pain, ‘feeling rules’</td>
</tr>
<tr>
<td>Consensus Family</td>
<td>Contracts, agreements, definitions of the situation, uniformity, conformity, conflict</td>
<td>Compliance</td>
</tr>
</tbody>
</table>

Table 2.1 Coding Families (Source: Glaser, 1978)
Glaser (1978) provided different theoretical framing concepts – coding families, to facilitate theoretical coding stage, as presented in Table 2.1.

We choose “Six Cs coding family” to direct our theoretical coding. This is because:

**First, Six Cs coding family can help researchers to clarify the relationship between codes, concepts and categories, which caters to the research purpose of this study.**

According to Kan and Parry (2004), a set of questions will be asked in advance to facilitate theory generating as listed:

- Is it a cause or a consequence of some other category (Cause, and Consequences)?
- What are the intervening conditions for the consequences (Condition)?
- In what context will this category emerge (Context)?
- Is this category having a bearing on another category (Contingency)?
- Is there any correlation between this category and other categories (Covariance) (Strauss & Corbin, 1990)?

By considering these questions and trying to make answers, the outline of the theory will be much clearer.

**Second, Six Cs coding family is best suited with describing of a phenomenon** (in the column of ‘Example’ from Table 2.1, “pain suffering”, is a phenomenon that is under study by adopting the Six Cs coding family). In this dissertation, the phenomenon that we try to investigate is “velocity flux”. The Six Cs model is the first of 10 coding families to consider while coding. The remaining 9 coding schemes as we can see from Table 2.1 mainly describe:

1. Process
2. Degree
3. Types
4. Strategies
5. Interactions
6. Self-identity
7. Cutting point
8. Cultural (social categories)
9. Consensus

All the coding families could be used in various stages of analysis and these coding techniques are flexible and may overlap with each other (Kan & Parry, 2004). The Six Cs model provides a useful framework for us to remain sensitive to uncover the relationships in the data (Kan & Parry, 2004). However, other coding families, such as “Type Family” and “The Degree Family”, are laying emphasis on developing theories in other directions, but do not aim to describe a phenomenon and investigate potential relationships, which are the goals of this dissertation.

2.5 Chapter Summarization

In this chapter, we introduced the research design of this study. This chapter works as a foundation to direct the data collection and analysis process. In Chapter 3, we will present the results after reviewing previous literatures regarding our topic. The process and results of the case study will be provided in Chapter 4.
3. LITERATURE REVIEW

As introduced in Chapter 2, literature review is one of the data collection methods that are used in this study. This chapter will present the results after reviewing previous publications on our topic and proceed as follows: (1) a reflection about the factors that cause velocity flux based on previous publications (2) existing research on the relationship between changes in development velocity and the pace of incoming customer feature requests.

As brief, in this chapter, we will find answers regarding the questions:

1) What are the possible reasons that cause velocity flux in agile development?
2) How will the pace of incoming customer feature requests and development velocity influence one another?

3.1 Fluctuation of Velocity

Stable development velocity is one of the most important process indicators for developers to achieve in agile software development. If development velocity does not change too much in a sprint, then this means the working team and the process of the development are well aligned (Omanovic & Buza, 2013).

In the introduction chapter, we mentioned four explanations for velocity flux: unexpected interruptions, high work in progress, rework, and hidden complexity. These four explanations are based on the conceptualization by Albero Pomar (2014). We did, however, merge two of Albero Pomar’s explanations (commitments are not fulfilled and correlation of team availability and commitment fulfilment) into the concept of unexpected interruptions. The rationale for our re-packaging of Albero Pomar’s concepts is that all the explanations may lead to unfulfilled commitments. We will give an explicit explanation about these four factors as follows:
1) **Unexpected interruptions**

Developers in an agile team should make commitments on what they will deliver in the next sprint, which means team members can consider their availability and task complexities to decide what they will complete in a sprint. However, in Albero Pomar’s investigation, there are only 3 out of 20 iterations that meet such commitments. One important reason used to explain this phenomenon is unexpected interruptions, such as unscheduled meetings, health issues, or technology issues. Such interruptions will consume time that is intended for development.

2) **High work in progress**

By following Definition of Done (a clear and concise list of requirements that a software increment must adhere to for the team to call a task complete), tasks that not meet the requirements cannot be defined as done. In this situation, some tasks that are too big to be finished in one sprint can lead to high work in progress. High work in progress may be perceived as low productivity when looked at the sprint level. High work in progress may also cause developers to frequently shift their focus between assignments, which may even lower their productivity and development velocity.

3) **Team Dynamics and rework**

When the delivered software does not fulfil customer expectations, the task is not to be considered as done (Davis, 2013). Rework may be caused by many reasons that are related to the dynamics in a development team, e.g. an unclear definition of done, inadequate communication between developers and product owners, or politics that pressure the team to take on too many assignments (Albero Pomar et al., 2014). The development team will have to redo the task in this situation, causing a decrease in velocity.

4) **Hidden complexity**

The complexity degree of software development projects can be divided into four categories - simple, complicated, complex and chaos - according to the level of technology that developer mastered and the complexity of customer requests.
Sometimes, commitments are made but missed due to complex requests reported and unfamiliar technology adopted (Albero Pomar et al., 2014). Hidden complexity in feature requests from customers – as perceived by the development team – may lead to difficulties for developers to complete the feature in time (Albero Pomar et al., 2014; Omanovic and Buza, 2013).

In above, we have listed the causes of development velocity flux discovered by Albero Pomar et al., 2014, in their paper “Understanding sprint velocity fluctuations for improved project plans with Scrum: a case study”. This paper is most related to our research topic after searching in the library database of Uppsala University and Google Scholar. Some other literatures also mentioned development velocity, but they did not put emphasis on its fluctuation or they provided similar ideas as stated in Albero Pomar’s paper. For example, in the paper “Importance of Stable Velocity in Agile Maintenance” by Omanovic and Buza in 2013, they motioned an idea: “if developers were not understand customer requests properly, development velocity would decrease”, which is a similar factor as “hidden complexity” that is proposed in Albero Pomar’s paper. Therefore, same opinions will not be represented in this chapter and we use Albero Pomar’s view as a representative of the factors that cause development velocity flux in previous studies.

3.2 Relationship Between Customer and Development Velocity

While our interest includes the relationship between new feature requests and development velocity, the literature review includes two sides of this relationship. First, research that examines this relationship, i.e. how the rate of incoming customer feature requests affects the development team – and consequentially development velocity. Second, the other way around, research that scrutinizes how changes in development velocity affect stakeholders request new features.

3.2.1 How feature requests affect the development team and development velocity

Leffingwell (2011) stated that velocity can be used as a reliable predictor for future, but there is a need for careful reflection on how to adopt the concept in agile management.
Agile teams typically aim at continuously increasing velocity while improving quality at the same time. The need to preserve or increase quality means that increased velocity is not necessarily beneficial for customers (Leffingwell 2011). However, if customers try to put more requests to push the development team to increase development velocity, the development team will react in one of three ways according to Leffingwell, 2011:

1) **Continuously improve team’s true productivity and agility in all aspects**

The increase of development velocity must be accompanied with continuous improvement of true productivity and agility in all aspects (e.g. technique, understanding of agile process). Customers and management teams need to support development teams in this situation, for example, by providing resources to facilitate implementation of new agile practices or adoption of better technologies.

According to Cohn (2010, p. 193), “I did eventually learn that teams cannot be pushed infinitely hard and that beyond a certain point, working more hours in a week will move the team backward rather than forward”. This is means if customers only push the development team to work overtime without appropriate support, high velocity will not last long.

2) **Cut back on quality and building technical debt for future**

The increase of velocity must not lead to a cutback on quality and building technical debt for a future period. When a development team realizes they cannot fulfill stakeholders expectation with their current productivity, while they cannot increase their efficiency with current agile practices or technology, there is a risk that they sacrifice product quality to increase the perceived productivity. Although it may appear as if productivity has been improved, the actual productivity decreased and so does the velocity. Troubles regarding quality and functionality are postponed to the future. According to Omanovic and Buza (2013, p. 6), “When sprint backlog is overloaded, then you have a choice to delay the delivery - break time boxing, not deliver all functionalities”. Otherwise, all the troubles regarding quality and functionality are just left for the future.
3) **Multitasking**

In 1992, Clark and Wheelwright investigated the influence of multitasking on development velocity. They pointed out that the total amount of development time goes up when a person only has two tasks to work on. However, if one person took more than two tasks at same time, the total working time on these tasks would be decreased. As we can see from Figure 3.1, when a person has three tasks to work on, the total amount of time working on tasks is lower than only one task to work on (Clark & Wheelwright, 1992). The reason why the working time on two tasks has increased might be because the second task could fill the time when the developer is blocked by waiting for a call, an email, or an approval of the design and so on. However, situations become different when developers take more than two tasks. According to Cohn (2010,p.193), “one of the reasons that multitasking is so horrible is the task switching cost involved. The more tasks or projects we are involved in, the more likely we are to be interrupted while working on them”. Under this circumstance, the quality or development velocity will be influenced due to the decreased working time on tasks.

![Figure 3.1 Amount of Time Spent on Tasks (Source: Cohn, 2010)](image)

Based on the above study, we found that the development team has three possible reactions when customers increased the amount of feature requests. Support from customers and the coordination from product owners are the key factors to achieve a stable or increased velocity in a development team.
3.2.2 How velocity flux affects how stakeholders request new features

In this section, we aim to investigate the mutual influence between changes of incoming customer feature requests rate and development velocity. In the above sub-Section, we discussed how the improved requests affect development velocity. In this sub-Section, we are trying to investigate the potential effects to customers when velocity changed. Development velocity affects new features request of stockholders in the following way:

1) Make Adjustment
   The adjustment of cost, time, quality and functionality should have to be done more than once during a sprint if the trends are not going as expected (Schwaber & Beedle, 2002). Based on this, if velocity were not developed as expected, customers requirements on quality and quantity and the investment for a sprint should be changed accordingly.

2) Lower expectation when slow progress
   Customers expectation is connected with the productivity of the development team. According to Schwaber and Beedle (2002, p.38), “What really impresses the customer, though, is that the team has gone for months without producing any functionality and the customer has given up.” Thereby, when team velocity is low, customers should lower their expectation on the development team and fewer requests will be reported.

3.3 Motivation from Literature Review

Not many useful literatures that directly connected to our research questions have been found from the databases that we used for searching literatures, Google Scholar and Uppsala University Library. It is useful and meaningful if we could fill the gaps through our particular study perspective - the relationship between the rate of incoming feature requests and development velocity - to investigate the factors that cause development velocity flux as suggested by the existing literatures. In Chapter 4, CASE STUDY, we will present how we will collect and analyze data regarding our topic within an empirical case study.
4. CASE STUDY

We have discussed previous findings regarding our topic in last chapter. Since our literature review results (in Chapter 3) suggests that more research is needed in order to understand the dynamics between customers and developers, we will conduct an empirical study to further explore and theorize the causes of velocity flux and its relationship with the pace of incoming customer feature requests. By following the research design in Chapter 2, this chapter is a case study report to bring the process, results and findings of the study. Our work is conducted in accordance with Yin’s (2003) idea on how to present a case study, including the following aspects:

1) A description of the focus (aim) of the case study;
2) A description of the background, context or settings of the case;
3) A description that how the data will be collected;
4) A description of the data analysis (both quantitative and qualitative);
5) A description of the conclusions and implications based on the finds.

4.1 The Focus of the Case Study

The focus of the case study corresponds to the purpose of the paper, which is to further explore and theorize factors that cause velocity flux through investigating its relationship with with the pace of incoming customer feature requests. In addition, eight strategies will be provided to overcome velocity flux in agile teams. In short, we aim to contribute to agile software development methods, especially for sprint planning.

4.2 Background of the Case Study

In this dissertation, U-CARE is the case under study. U-CARE is a multi-disciplinary research community as introduced in Chapter 2, which involves academics from psychology, medicine, information systems, caring sciences, economics and health practitioners, at Uppsala University, Uppsala, Sweden (Sjöström et al., 2014). The aim of this project is to establish an internet-based treatment of depression and anxiety for patients
with somatic disease (Sjöström et al., 2014). The development team in U-CARE follows Scrum (one of most popular agile development methods) as their daily agile development method.

There are two reasons for us to apply U-CARE as the case to study in this dissertation:

1) This is a project held by Uppsala University. As a Master student at the same university, an easier access to the database (product backlog) is provided and it is easier to contact the researchers working in the project in this situation.

2) The development team in U-CARE has adopted agile software development methods for more than two years. Therefore, the researchers (agile team members) are familiar with agile methods, which means the data collected from U-CARE is reliable.

4.3 Data Collection

As stated in sub-Section 2.3.1, the source of evidence used in this dissertation is archival record (database) and interview. We will describe how data will be collected in the sub-Section 4.3.1 and 4.3.2.

4.3.1 Database

The quantitative analysis is based on 46 two-week periods (cases), corresponding to 92 weeks of product backlog data. In total, there were 509 new feature requests and 480 completed requests during the time period. We exported the date, the number of fixed bugs, reported bugs, reported issue (‘issue’ means the total number of reported bugs and features), fixed issue, issue Delta (the number of reported issues minus fixed issues), and bugs Delta (the number of reported bugs minus fixed bugs) from U-CARE product backlog into an Excel file, as we can see in Table 4.1. Then the dataset will be further processed for the ease of analysis in the sub-Section of Data sorting.
Data sorting

Based on the original dataset exported from U-Care database (Table 4.1), we aggregated the data into two-week time chunks that we refer to as case(s) from now on as presented in Table 4.2. For each case we calculated the value for the constructs shown in Table 4.3. In this study, we are interested in velocity in TWO aspects: i) development velocity, i.e. the amount of fixed features in a certain time period ii) “velocity” that customer report features, i.e. the pace of incoming feature (including bugs) requests in a certain period.

<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
<th>Day</th>
<th>IssueDelta</th>
<th>BugDelta</th>
<th>Fixed Issues</th>
<th>Fixed Bug</th>
<th>Reported Issues</th>
<th>Reported Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>2</td>
<td>27</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>2013</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2013</td>
<td>3</td>
<td>3</td>
<td>-4</td>
<td>-1</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2013</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>2013</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1 Original Dataset

<table>
<thead>
<tr>
<th>CaseNum</th>
<th>CRV</th>
<th>PRV</th>
<th>TRV</th>
<th>TDV</th>
<th>CDV</th>
<th>PDV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26</td>
<td></td>
<td>26</td>
<td>23</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>26</td>
<td>48</td>
<td>39</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>22</td>
<td>66</td>
<td>58</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>18</td>
<td>102</td>
<td>81</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>36</td>
<td>111</td>
<td>107</td>
<td>26</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>9</td>
<td>117</td>
<td>110</td>
<td>3</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 4.2 New Dataset

<table>
<thead>
<tr>
<th>Construct</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Development Velocity (CDV)</td>
<td>Completed issues (features) within the current case</td>
</tr>
<tr>
<td>Previous Development Velocity (PDV)</td>
<td>Completed issues (features) in the previous case</td>
</tr>
<tr>
<td>Current Request Velocity (CRV)</td>
<td>New feature requests within the current case</td>
</tr>
<tr>
<td>Previous Request Velocity (PRV)</td>
<td>New feature requests in the previous case</td>
</tr>
<tr>
<td>Total Request Velocity (TRV)</td>
<td>Total amount of requested features in the backlog</td>
</tr>
<tr>
<td>Total Development Velocity (TDV)</td>
<td>Total amount of completed features</td>
</tr>
</tbody>
</table>

Table 4.3 Constructs in Use to Analyze Correlations
4.3.2 Interview

We interviewed all the active developers in the development team (three persons) as well as the product owner in order to gather reliable and diverse response. These four respondents will be referred to as R1 – R4 (randomly ordered) in the analysis to respect for their confidentiality. Since the interview is designed as semi-structured, eight questions were prepared beforehand (in the appendix), while the order of the interview questions could be changed or additional questions can be asked according to the answer of interviewees. Most of the interview questions were designed based on the research questions of this dissertation with rationale from previous publications. Other questions were asked in order to help us to understand the context and the agile process in U-CARE. During the phase of interviewing, every interview lasts around 40 minutes. The interviews will be transcribed, followed by grounded theory – a qualitative analysis method.

4.4 Database Analysis

In this section we will present how we will conduct quantitative analysis on the new dataset with SPSS.

Figure 4.1 shows the growth of the total amount of feature requests in the backlog (TRV) and the total number of completed features (TDV). There appeared to be a correlation between the pace of incoming feature requests (including bug reports) and development velocity, since TDV and TRV grow at a similar rate and direction as we observed in Figure 4.1. Under this assumption, we focus on the correlation between the changes of TDV and TRV in every case in next sub-section.
Figure 4.1 Line Chart about TDV and TRV

4.4.1 Correlations

From the analysis of the line chart in above, we infer that there should be an association between the changes of TDV and TRV. However, we are more interested in looking for the correlation between CDV and PRV, or between PDV and CRV. By doing this, we can demonstrate whether the change of one variable (TDV or TRV) in one case would influence the other variable change accordingly (i.e. in the next case) or not.

Correlation coefficient (denoted r) is a measure of the direction and strength of the linear relationship between two quantitative variables (Moore, McCabe and Craig, 2009). Pearson correlation test is conducted to investigate the following two assumptions:

1) There is an association between previous development velocity (PDV) and current request velocity (CRV);
2) There is an association between previous request velocity (PRV) and current development velocity (CDV).
We do not make any claims that the findings of the quantitative study are generalizable in any way. They should be seen as a way to characterize the correlations in the current empirical setting. Following an interpretive stance, we consider our quantitative analysis to show tendencies of the past, rather than predictions of the future (Walsham, 1995).

In this dissertation, we use SPSS to give us the correlation test results directly as shown in the following table:

![Table 4.4 Correlation Results](image)

The results of Pearson correlation tests between PDV and CRV, and between PRV and CDV are shown in Table 4.4. The correlation coefficient ($r$) has a scope from -1 to 1 and indicates the strength of the relationship by how close it is to -1 or 1 (Moore, McCabe and Craig 2009). If $|r|<0.3$, it indicates that there is a weak correlation between the variables. If $0.3 < |r|<0.7$, it suggests that there is a moderate correlation between the variables. And if $0.7 < |r| < 1$, it represents there is a strong correlation between the variables.

Another factor, which deserves our attention, is the significance of $r$, as shown in Table 4.4 as “Sig.”. When the value of Sig. is low (usually compared to a significance level of 0.05), it indicates the correlation between variables is significant; otherwise, even if $r$ is high, it does not mean there is an association between the variables.
From Table 4.4, we get:

1) The $r$ between PDV and CRV is 0.228 with Sig. (significance of $r$) is 0.132 (>0.05, the correlation is not significant). Therefore, it can be concluded that there is no correlation between CRV and PDV.

2) The $r$ between PRV and CDV is 0.586 and its Sig. (significance of $r$) is 0.000 (<0.05, the correlation is significant). Thereby, there is a moderate and significant correlation between PRV and CDV.

In conclusion, there is a correlation between PRV and CDV, which means the rate of incoming requests in previous case will affect development velocity in current case. Besides, the reason why TDV and TRV share a similar trend of growth is because there is a correlation between CDV and PRV, according to the analysis in above. In next sub-Section, we will verify the relationship of causation between CDV and PRV.

4.4.2 Causation

According to Moore, McCabe and Craig (2009, p.154), “even when a strong association is present, the conclusion that this association is due to a causal link between the variables is often hard to find.” Moore et al. suggest the use of controlled experiments to examine causality. In this study, it is not feasible to perform a controlled experiment since it is a longitudinal study that needs continuous observation, and it is tough to control other factors that may affect the experiment results.

Instead, we reason about the potential causality using Moore, McCabe and Craig’s (2009) five criteria for establishing causation between CDV and PRV as shown below:

1) The association is significant. The association between PRV and CDV is significant.
2) The association is consistent. Analyzed data spans over a two-year period.
3) Higher does are associated with stronger responses. As we observed from the line chart and correlation test, increased PRV was continuously followed by an increase in CDV.
4) *The alleged cause precedes the effect in time.* Our analysis is based on comparing requirements from the previous two-week period with completed issues in the current period.

5) *The alleged cause is plausible.* Our interpretation is that it is reasonable that an increase in feature requests from customers will *cause* increased development velocity.

Supported by these criteria, we conclude that the change of the pace of incoming customer requests is one cause for velocity flux. In next section, we will present the qualitative analysis based on the interviews conducted in the agile team of U-CARE.

### 4.5 The Adoption of GT

The purpose of conducting interview in this phase is to collect opinions from the U-CARE team regarding the relationship between the pace of incoming customer requests and development velocity to investigate what causes development velocity flux. Suggestions are provided in the end to overcome velocity flux in the development team of U-CARE.

Since we decided to use GT as the means to analysis interview transcripts (in Chapter 2), open coding is first applied after the key points in each interview transcript are summarized. We break down and create label *(codes)* for chunks of data by using a phrase to summarize the key points of similar meaning. Thereafter, GT’s constant comparison method is applied as described in sub-Section 2.4.2. The codes collected from each interview will be compared with the codes from the same interview and other interviews, to generate *concepts* in GT (Hoda, Noble & Marshall, 2010). The abstraction will continue based on the concepts obtained in previous steps, to produce a higher level of abstraction, which is called a *category* in GT.

By following GT’s constant comparison method, the concepts we get from the interview transcripts are *hidden complexity, unreported development efforts, periods of minor issues and bug fixing, ill-defined feature requests, high work in progress* and these concepts lead to emerging of the category *(change of the pace of incoming customer*
feature requests’. The other category is called ‘velocity flux’, which is the consequence and has association to the category ‘change of the pace of incoming customer feature requests’.

The next step of GT is to get ideas about the codes and outline the emergent theory, which could be used to represent the relationships by writing and sorting memo and theorizing. The last step is to generate a theory, namely theoretical coding. After studying the theoretical coding families in the sub-Section 2.4.2, we decided to use “Six Cs coding family” as the framework to represent the theory we generated. According to “Six Cs coding family”, we will describe the following factors in the theory, which is also shown in Figure 4.2:

(1) Contexts: the ambiance (the agile team in U-CARE);
(2) Conditions: factors that are prerequisites for the category ‘change of the pace of incoming customer feature requests’;
(3) Causes: reasons that cause the rate of incoming customer feature requests change in a certain period of time;
(4) Consequences: outcomes or results regarding the change of the rate of incoming customer feature requests in U-CARE;
(5) Contingencies: moderating factors of the consequences;
(6) Covariance: correlations between categories (velocity flux as the factors that cause the change of pace of incoming customer feature requests change).

The category ‘change of the pace of incoming customer feature requests’ is at the centre of the diagram and the other factors are described in the rectangles in relation to central category. In the following sections we will give an explicit explanation about the theory we built, as shown in Figure 4.2.
4.5.1 Context

The context under investigation is the development team that adopts agile methods in U-CARE. The agile development team follows the process of Scrum, such as daily stand-up meeting, integrative development, sprint planning, and sprint review meeting and so on.

4.5.2 Condition

The development velocity in U-CARE agile development team is not stable, and the change of the pace of incoming customer feature requests is a reason that cause velocity flux. Besides, customers are a new role that works directly with the development team in U-CARE under the context of agile development.
4.5.3 Causes and Covariance

The analysis of interview transcripts resulted in a number of concepts that can be used to explain velocity flux. Besides, there is a correlation (covariance) between development velocity and the change of the pace of incoming customer feature requests in U-CARE since velocity flux as the changes of factors that cause the pace of incoming customer requests incoming rate change.

These concepts, described below, have been categorized into two overarching categories: (i) Factors with an indirect impact on velocity (factors 1 – 5), shown in Figure 4.3, and (ii) factors with a direct impact on velocity (factors 6 – 9). Since our correlation analysis has shown plausible causality between PRV and CDV, we assume that changes in the rate of customer feature requests affect the development velocity indirectly.

![Diagram](image)

**Figure 4.3 Factors with Indirect Impact on Velocity**

1) **Hidden complexity**

“It is hard to know if the case is too small or too big by customers.” – Respondent 1
Requested features may be too big for developers to complete in a sprint. This may cause a form of hidden complexity. Tasks that are started but have not been finished within a sprint will not be immediately visible in velocity assessment – this was also reported by Albero Pomar et al. (2014). In this case, the incoming feature requests will decrease because the dependent tasks will not be visible as feature requests in the backlog. Then the development velocity will decrease as well since tasks not completed.

2) Unreported Development Efforts

“When people report bugs, we will immediately open a task and address it. Whereas, when there is not so many bugs reported, we still put same amount of effort into developing, and work on the task, for example, improve the inner system, that are not immediately visible for the users.” “I will not open a lot of tasks to solve a problem. Because I already know how to do it, so I just do it.” - Respondent 3

In U-CARE, the reported tasks come from two sources: (a) customer reported requests, (b) developer created tasks. In this case, while developing, developers could create their own tasks when the incoming requests from customers are completed before the due date. When developer created tasks increased, customers reported features will decrease accordingly. However, these tasks are normally not reported as complex tasks since developers know what should be fixed. In this situation, developers spend their effort to solve tasks that are not presented in the product backlog. Such unreported work causes velocity flux.

3) Periods of Minor Issues and Bug Fixing

“For customers, everything is important. When customers use the system, they will report the issues that they are not satisfied with based on personal experience, but not from the technique perspective.” – Respondent 2

Customers usually do not have solid education background in the field of software development technology and management. From the experience of developers, the
problems reported by customers are dominated by minor problems. This is especially apparent in periods when customers focus their efforts on beta testing. Minor problems are usually easier solved than developer created tasks. Therefore, in times when minor problems are prevalent, the velocity will increase since the tasks need less time and effort to be completed.

4) **Ill-defined Feature Requests**

“That bug could only occur with certain conditions. If customer can specify the problem as much as possible, it will make our jobs easier. Because we could get the point of where the problem is and this could save our time to address the problems.” - Respondent 2

For some reported tasks, the problems and the context in which they occur are often not clearly described. Ill-defined problem descriptions triggers a need for developers to spend more time to understand the problem (Pikkarainen et.al 2008), and may possibly lead to rework if the developers make assumptions instead of investigating the problem in collaboration with the customer. Under this circumstance, the incoming requests probably decrease and subsequently development velocity will decrease as well.

5) **High Work-in-progress**

“Currently we focus on fixing bugs and not so many new features are requested by customers” – Respond 4

The feature of software is the basic requests of the product. They serve as the base on which can be used to direct the development of a working system. The one interviewee stated that usually, developing a new feature is a more complicated and time-consuming job than bug fixing, resonating with Leffingwell (2011). Customers usually request new features when they become aware that the developers have completed – or nearly completed – a previously requested feature. It indicates that when a new feature is under development, the incoming requests will decrease – thus,
this ‘modus’ of development is quite different from periods of minor issues and bug fixing.

In addition to the causes that have indirect impact on velocity that connected to the change of the pace of incoming customer requests, we get four factors that directly cause velocity flux in U-CARE, and they are:

6) **External interruptions**

   “*Some of the meetings only need one developer to go. However, if all the three developers go for the meeting, it is a waste of time*”. – Respond 3

   Interruptions in such as meetings and health issues consume the developers’ time for actual development. The team availability – i.e. effective developer working time – decreases due to external interruptions (Dearden, Rizvi and Gupta 2010). Logically, development velocity will decrease.

7) **Team Dynamics and Rework**

   “*I prefer a little bit more to be pushed and I do not think that will have impact on quality.*” - Respondent 1

   “*It is very individual how the quality in contrast to the workload.*”- Respondent 4

   Some developers think that adequate pressure team dynamics can stimulate their productivity. If the product owner could keep a good balance between customer demands and requests on developers’ productivity, the development velocity will increase without compromise of quality (de-Ste-Croix & Easton, 2008). However, if the external pressure and dynamics exceeds the maximal level that the individual developer can tolerate, it will in turn have a negative effect on product quality and development velocity. Rework is needed if the quality of the delivered features are not acceptable which will cause decrease of velocity.
8) Learning Needs

“It is a big system under developing. If all developers need to do everything, then they need to learn something new for each part of the system or technology and might be have a hard time to get into the problems.” – Respondent 4

It is necessary for developers to develop new skills when facing new problems, since not all the problems can be solved instantly based on experience that they gained from previous works. Meanwhile, it will take considerable time to learn new things, thereby slowing down the progress of the project and the development velocity of the development team. If a developer works with system parts s/he is familiar in one sprint, and parts s/he is less familiar with the next sprint, there will be a perceived velocity decrease.

9) Periods of Intensive Testing

“Every time we finish a task, we also write a task for it. I am responsible for system development and making all the tests pass currently.” – Respondent 3

The programmers in the current case take responsibility for both software development and testing, while the work done for software testing is not taken account into development velocity. Similar situations have been reported by Wang, Wu and Zhao (2008). When programmers spent too much effort in testing, the development velocity will be cut back inevitably, since the work on testing is not always taken into account in velocity measures. If tests are given different focus in different sprints, there will appear to be a flux in velocity. In the current case, there have been several periods where feature development was halted to make room for refactoring and testing of the existing code.

4.5.4 Contingencies

“Scrum is a learning process”, said by one of our interviewees. Through constant learning, the agile practices will be enhanced and the agility will be improved. These improvements
will in turn facilitate the development team to improve their productivity and development velocity when they take same amount of effort into the development.

4.5.5 Suggestions

Velocity is an important metric that can help management and Product Owner to estimate the progress of the project as described in sub-Section 1.1.1. Besides, keeping a stable development velocity that can be increased over time is important in agile development teams. In this sub-Section, we will provide eight strategies to overcome velocity flux to U-CARE agile development team.

By studying and analysing the indirect causes of velocity flux and the correlation (covariance) between development velocity and customer feature requests incoming rate, we give five suggestions to overcome velocity flux in U-CARE as shown in Figure 4.4:

![Figure 4.4 Suggestions Based on the Indirect Causes](image-url)
(1) **Break tasks down**

Big and complex tasks should be divided into two or more sub-tasks (Accardi-Petersen 2011). Hidden complexity is one of the causes for the change of the pace of incoming customer requests, which will affect development velocity as shown in Figure 4.6. If complex tasks could be divided into smaller parts in a smart and reasonable way, the development velocity will keep stable when developers take the same amount of effort into a sprint and the situation of high work in progress will not happen.

(2) **Long-term plan should be available to the development team and customers**

A clear long-term goal is very important for both the development team and customers (Cartaxo et al. 2013). It can encourage developers to build their own tasks in the direction of the long-term goal, and it also helps customers to know when they need to add new features to the product backlog or whether it is a good time to focus on improving the current feature. With the help of a clear long-term plan, customers and the development team will make a collective effort on a common goal. Hence, development velocity will not be affected by developers spending time and effort on invalid tasks.

(3) **Keep track of developers created tasks**

We suggest developers to keep an accurate record on their jobs since they will omit reporting some sub-tasks that they think unimportant. Some professional Scrum management systems could be adopted in order to provide both customers and developers an easy and convenient problem-reporting environment (Hoda, Noble & Marshall, 2010).

(4) **Re-calibration**

The knowledge of customers in the field of software development and management will expand and correspondingly they will have a more specific expectation for the product with time passing by. Besides, the productivity and development velocity of the
development team will be enhanced due to constant learning and agile practice improvement (Leffingwell, 2011). Therefore, making adjustment about resources, techniques and development commitments after a certain period of time based on the improvement is necessary (Martin, Biddle & Noble, 2010).

(5) Enhance the communication between developers and customers

The communication between developers and customers is very important in agile software development. Effective communication can enhance the knowledge of customers about techniques and development management (Pikkarainen et.al 2008). For developers, it is critical to get feedback and ideas of customers about the completed tasks and new features in the future (Wang, Wu and Zhao 2008). The communication will help to establish mutual understanding for both developers and customers. With a better understanding between developers and customers, the development velocity will be increased.

In the following sections, we will describe three improvements that could be made in U-CARE regarding how to keep a stable or increased development velocity based on the direct causes of velocity flux. The model is shown in the Figure 4.5.

Figure 4.5 Suggestions Based on the Direct Causes of Velocity Flux
(6) **Developers should be experts in part of the system**

When developers have a specific focus in the system, the average learning time of the development team will be shorten since the familiarity with the system and customer requests is high. By learning one part of the system in depth instead of learning the whole system in broad, less effort and time will be taken up. Therefore, being an expert in a certain part of the system will help to ensure a stable and even increasing development velocity.

(7) **Full time testers**

It will be helpful if the role of tester and developer could be separated, which is not currently applied in U-CARE. Development velocity will increase if developers can focus on software development without responsible for testing. A full time tester will also help to control the quality of products (Abrahamsson et.al 2002). With higher product reliability, the possibility that bugs remain in the delivered tasks will be much less and the incoming customer requests regarding bugs will decrease as a consequence.

(8) **Control external influences**

External influence is one of the main causes for velocity flux. The control of external influences, for instance, reducing the redundant meeting and removing technical impediments, is very important for keeping a stable or increasing development velocity. It is recommended that, the management or product owner should think over whether it is necessary for developers to attend the planned meetings and the Scrum Master should provide technical help to developers before a new sprint starts (Dearden, Rizvi and Gupta 2010). With a good working environment, developers could focus on their work and development velocity will increase as a consequence.
4.6 Summarization of the Case Study

In this chapter, we presented how we collected data through exporting data from U-CARE product backlog and interviews, and analysed the collected data by quantitative analysis (Line Chart, Correlation Test, Criteria for causation) and qualitative analysis (Grounded Theory-Six Cs coding family) in a case study.

We discovered nine factors that can cause velocity flux indirectly or directly. We also provided some strategies to overcome velocity flux. In the next chapter, answers will be given to the research questions and a concluding discussion on the contribution that we made to agile development will be presented.
5. CONCLUDING DISCUSSION

In this dissertation, we have argued that there is a need for a more thorough understanding of the causes of velocity flux in agile development. We have investigated an agile software development process in an interdisciplinary research setting, and shown that there is a moderate and significant correlation – possibly a causal relation – between the pace of incoming customer feature requests (PRV) and the development velocity in the subsequent two-week period (CDV). Further, we conducted a qualitative study based on interviews with the development team and the product owner to investigate why there is a flux in development velocity in the project. Nine factors that cause velocity flux have been found and eight strategies are provided to overcome development velocity flux as well.

In this chapter, we will first present the answers to the research questions. The contributions we made for agile software development and sprint planning will be discussed in the third research question. In addition, the future work that could be done for further development will be described in the Section 5.2.

5.1 Answers of Research Questions

Is there any relationship between development velocity and the pace of incoming customer feature requests?

We investigated the relationship using both quantitative and qualitative analysis method. With quantitative analysis method, we found that there is a moderate and significant correlation between the pace of incoming customer feature requests (PRV) and development velocity in the subsequent two-week period (CDV). We also demonstrated that the change of the pace of incoming feature requests (PRV) could cause development velocity flux in the subsequent two-week period (CDV) according to some criteria that is used to verify causation, as suggested by Moore, McCabe and Craig (2009).
Then we conducted interviews with the agile team in U-CARE with respect to our research questions and quantitative analysis results. We analysed the interview transcripts by adopting GT. The results revealed the development velocity changes as the factors that influence the pace of customers report features change (the correlation has been verified in quantitative analysis). However, some other causes, such as external interruptions and time for testing, that are not connected with the pace of incoming customer feature requests, will also cause development velocity flux. In other words, we found the change of the pace of incoming feature requests is a significant but NOT the only reason causes the change of development velocity.

**What are the causes of the relationship between development velocity and customer feature requests incoming pace?**

By interpreting interview transcripts, we found there are five factors that affect the pace of requests reported, and the development velocity will change as these factors change. Detailed analysis is in sub-Section 4.5.3. These factors are:

1) High work in progress  
2) Hidden Complexity  
3) Periods of Minor Issues  
4) Ill-defined Feature Requests  
5) Unreported development efforts

**How to use this relationship to make contribute to the current agile development method, especially for sprint planning?**

By studying the previous literature in Chapter 2, we found some factors could affect development velocity in an agile team and they are: (1) Unexpected interruptions; (2) high Work in Progress (WIP); (3) rework and (4) hidden complexity. Through studying the U-CARE case, we confirmed some of the factors that could influence development velocity we found in previous work, as stated in Table 5.1: 

45
<table>
<thead>
<tr>
<th>Category</th>
<th>Concepts</th>
<th>Case Study</th>
<th>Literature Review</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Factors with an INDIRECT impact on velocity</strong></td>
<td>High work in progress</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Hidden Complexity</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Periods of Minor Issues</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ill-defined Feature Requests</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unreported development efforts</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Factors with a DIRECT impact on velocity</strong></td>
<td>Unexpected interruptions</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Team Dynamics and Rework</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Periods of Intensive Testing</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Learning Needs</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1 Findings from Case Study Compared to Literature Review Results

Given the correlation between PRV and CDV, and through the qualitative data analysis, we have identified nine factors that may cause development velocity flux (Table 5.1) either indirectly or directly. Four of these factors have been discussed previously in the literature. Our contribution to research thus lies in (1) added knowledge about factors that cause velocity flux, and (2) another layer of abstraction to conceptualize causes of velocity flux (indirect v. direct factors). In sub-Section 4.5.5, we listed eight strategies to overcome development velocity flux. The causes that affect development velocity in this study could be the general issues that exist in other agile development teams. Therefore, management and Product Owner can apply some of the suggestions given in this dissertation during sprint planning and development to facilitate their work.

To sum up, our findings stress the need for an encompassing view on velocity. If customers assess developer productivity based on a view of productivity too focused on completing issues, there is a risk that developers do not prioritize learning, long-term thinking and testing – which are all necessary to avoid technical debt and increase long-term productivity. By conceptualizing these factors, we provide a tool for managers, scrum masters and product owners in software development projects to reflect systematically about the risks of focusing too much on short-term velocity. In addition, they may serve as a foundation for retrospectives, supporting development teams in
improving their work process. We thereby stress the need to include a planning for learning and quality assurance – and to also report on such activities in sprint reviews.

5.2 Future Work

Agile software development methodologies are a broad topic and there are many scopes worth to study in order to improve its management and planning, but we only focused on the perspective of development velocity under the case study of U-CARE.

For further development of this study, longer period of data could be collected in an agile team’s product backlog and more professional statistical analysis methods can be adopted in order to get a more convincing quantitative analysis. Furthermore, other scopes that is related to sprint planning and management, such as technology innovation and agile practice improvement, are not studied in depth in this dissertation but they are also worthy of investigation in later stage.
REFERENCES


*Evolution and Process*, 26(9), 776-783.


## APPENDIX

<table>
<thead>
<tr>
<th></th>
<th>Interview questions</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What is your role in U-CARE project and what are you responsible for?</td>
<td>Important to know in which respect will the opinion come from.</td>
</tr>
<tr>
<td>2</td>
<td>What do you think the responsibility of the whole development team in the U-CARE project?</td>
<td>Moreira (2013): Determine size for stories, collaborate with product owner and testers, writing code and executing unit test. They need to become active participants in understanding product requirements, talk to customers and users and spend more time interacting with their coworkers (Cohn, 2010).</td>
</tr>
<tr>
<td>3</td>
<td>What are the possible reasons that cause the variance of development velocity in different sprint? - Is cannot fulfill commitment a reason? - How will customer normally react when they realize this change? - Do you have any suggestions for customer and development team when they face this problem based on current U-CARE practice?</td>
<td>According to Albero Pomar, et al., 2014: <strong>Underestimate</strong> the task complexity point, external <strong>interruptions</strong>, high work in progress and <strong>rework, financial crisis</strong> are the reasons cause the fluctuation of velocity. (The number of developer should be kept stable) According to Cohn (2010), the <strong>overtime working</strong> will increase velocity at first sprint, but will lower than average in the following.</td>
</tr>
<tr>
<td>4</td>
<td>In my quantitative analysis, I found there is only a weak association between the changes of sprint velocity in last sprint and the amount of requirements in this sprint. - Do you believe this finding? Why or why not?</td>
<td>Linked to the quantitative analysis results. (Potential factors: requirements <strong>decided by the needs, features and software requirements</strong> (Leffingwell, 2011))</td>
</tr>
<tr>
<td>5</td>
<td>What do you think the responsibility of customer in the U-CARE project? - What do them normally responsible for?</td>
<td>Dingsøyr, Dybå and Moe, (2010): customer could provide the project with collaboration guides, skills specialists and direction setting.</td>
</tr>
<tr>
<td>6</td>
<td>Customer could put issues (tasks) into product backlog. Do you think the amount of reported issues is not always same for a certain period (one week)?</td>
<td>Samir (2013), customer behavior change (plan or budget or technology) will result in increased number of changed requirements, increased size and complexity of changed requirements and customers</td>
</tr>
</tbody>
</table>
- What cause this change?
- How will the development team respond accordingly in U-CARE? Will developer working over time?
- Will the product quality be affected (bugs, less functionally)?
- Do you have suggestions on customer (give incentives, or improve technology) and development team based on current U-CARE practice?

Leffingwell, 2011, when requirements amount increased, there are three possible responses: (1) continuously improve true productivity and agility in all aspects. (2) Cut back on quality, technical debt for a future period (3) simply increase the size of estimates.

In the quantitative analysis, we found that there is a significant correlation between changes of total requirements amount in last period and the sprint velocity in this period.

- Do you believe this finding? Why or why not?

Linked to the quantitative analysis results.
(Potential factors: budget, customer satisfaction, the cycle of more requirements cause more rework, and more rework cause more requirements)

Do you think the current sprint planning should be improved? How? (From the perspective of the relationship between requirements and velocity)

“When sprint backlog is overloaded, then you have a choice to delay the delivery- break time boxing, not deliver all functionalities.” (Omanovic and Buza 2013). So we need to control the size of sprint backlog in sprint plan meeting. And more?