Capturing domain knowledge from very small enterprises, during agile development - a Case Study

PAPER WITHIN Software Product Engineering

AUTHOR: Ajay Kumar Rajamundry & Onete Bogdan Gabriel

TUTOR: Robert Ian Day

JÖNKÖPING September 2018
This exam work has been carried out at the School of Engineering in Jönköping in the subject area Software Product Engineering. The work is a part of the Master of Software Product Engineering.
The authors take full responsibility for opinions, conclusions and findings presented.

Examiner: Ulf Seigeroth

Supervisor: Robert Ian Day

Scope: 30 credits (second cycle)

Date: 6th May 2019

<table>
<thead>
<tr>
<th>Postadress:</th>
<th>Besöksadress:</th>
<th>Telefon:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box 1026</td>
<td>Gjuterigatan 5</td>
<td>036-10 10 00 (vx)</td>
</tr>
</tbody>
</table>

551 11 Jönköping
Abstract

Very Small Enterprises (VSEs) are currently the backbone of all industries, since they account for roughly 80% of the world’s economy. In some countries, they account for over 95% of all companies and enterprises. As such, VSEs are a fundamental element in any given industry. Because of this, understanding how VSEs function and how, as a software developer, work and think is crucial to facilitate the software development process. A software unaware VSE (SUVSE) is a company which has little tangency with understanding software in general. The problem lies in how to properly and efficiently communicate with SUVSE, due to their limitations, as well as the general lack of knowledge regarding software development.

This report will cover the topic of challenges faced when working with a very small company for gathering domain knowledge (DK) and facilitating the communication and exchange of meaningful information, in the context of developing a software product, using agile methodologies. In the report, some of the more popular agile methodologies will be covered and chosen, in addition to existing ISO standards for VSEs.

The purpose of the research is to answer the following questions:

What topics are software unaware very small enterprise (SUVSE) aware of, in the software development process?

Which aspects of the software development process must be known by a software unaware domain expert of a very small enterprise (VSE), so that a software development team can capture the required domain knowledge (DK)?

How is the software unaware very small enterprises (SUVSEs) understanding of the software development process affected by their involvement in an agile development methodology?

To fulfill the purpose of the study, a case study was performed using 3 techniques to gather data: questionnaire, observation and interview. The participants were employees and CEOs from a few VSEs, contacted either via phone or face-to-face meetings.

The study showed the problems faced when working with SUVSE, what steps should be performed to facilitate communication with domain experts for gathering DK, and how the VSE can benefit from the knowledge and experience gained because of actively participating in the software development process. It concludes with practices on how to improve communication between software developers and VSE stakeholders to optimize the exchange of meaningful data and knowledge in the context of VSEs, and recommends future work based on the findings.

Keywords
Domain knowledge, domain expert, VSE, agile methodology
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEO</td>
<td>Chief Executive Officer</td>
</tr>
<tr>
<td>DE</td>
<td>Domain Experts</td>
</tr>
<tr>
<td>DK</td>
<td>Domain Knowledge</td>
</tr>
<tr>
<td>DSDM</td>
<td>Dynamic systems development method</td>
</tr>
<tr>
<td>FDD</td>
<td>Feature Driven Development</td>
</tr>
<tr>
<td>IR</td>
<td>Information Retrieval</td>
</tr>
<tr>
<td>JMCS</td>
<td>Jens Mårten Company Sweden</td>
</tr>
<tr>
<td>MUSCOW</td>
<td>M Ust S hould C Ould W ould</td>
</tr>
<tr>
<td>NIO</td>
<td>Nordisk Industri Optimering</td>
</tr>
<tr>
<td>RUP</td>
<td>Rational Unified Process</td>
</tr>
<tr>
<td>SAVSE</td>
<td>Software Aware Very Small Enterprises</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life Cycle</td>
</tr>
<tr>
<td>SDM</td>
<td>Software Development Methodology</td>
</tr>
<tr>
<td>SME</td>
<td>Small Medium Enterprises</td>
</tr>
<tr>
<td>SUVSE</td>
<td>Software Unaware Very Small Enterprise</td>
</tr>
<tr>
<td>SUVSE</td>
<td>Software Unaware Very Small Enterprises</td>
</tr>
<tr>
<td>TDD</td>
<td>Test Driven Development</td>
</tr>
<tr>
<td>UP</td>
<td>User Proxy</td>
</tr>
<tr>
<td>VSE</td>
<td>Very Small Enterprises</td>
</tr>
<tr>
<td>XP</td>
<td>Extreme Programming</td>
</tr>
</tbody>
</table>
## Contents

1 Introduction......................................................................................... 1

1.1 BACKGROUND................................................................................. 1

1.2 PURPOSE AND RESEARCH QUESTIONS ........................................... 2

1.3 CONTEXT ......................................................................................... 3

1.4 DELIMITATIONS .............................................................................. 6

1.5 OUTLINE ......................................................................................... 7

2 Theoretical framework ........................................................................ 8

2.1 RELATED STUDIES ........................................................................ 8

2.2 DOMAIN EXPERT (DE) ................................................................ 10

2.2.1 Challenges faced with DEs .......................................................... 10

2.2.2 Domain expert vs subject matter expert ...................................... 10

2.2.3 Domain expert vs stakeholders ................................................... 11

2.3 DOMAIN EXPERT INVOLVEMENT ................................................ 12

2.3.1 Roles and responsibilities ........................................................... 12

2.3.2 Vision of Knowledge gathering .................................................. 12

2.3.3 Information Retrieval ................................................................. 12

2.3.4 Feedback .................................................................................... 13

2.3.5 Acceptance tests ....................................................................... 13

2.3.6 Test Requirements and Test Cases ............................................ 13

2.3.7 Software Product Architecture .................................................. 13

2.4 DOMAIN KNOWLEDGE ................................................................ 14

2.5 VERY SMALL ENTERPRISE ............................................................ 14

2.5.1 Software unaware VSE ............................................................... 15

2.5.2 Software aware VSE ................................................................ 15

2.6 SOFTWARE DEVELOPMENT LIFE CYCLE ..................................... 15

2.6.1 Chosing an approach ................................................................. 15

2.6.2 Agile Methodologies ................................................................. 18

2.6.3 Choosing methodologies .......................................................... 19

2.6.4 Extreme Programming (XP) ....................................................... 20

2.6.5 Scrum ....................................................................................... 21

2.7 SOFTWARE DEVELOPMENT PROCESS IN SU VSES .................. 23
2.7.1 VSE limitations................................................................. 23
2.7.2 Challenges of Software Development Process in software unaware VSEs: 23
2.8 ISOSTANDARDS - ISO 29110.................................................. 24
2.8.1 Definition................................................................. 24
2.8.2 Problems with adopting ISO 29110.............................. 25

3 Method and implementation ............................................. 27
3.1 OPTIONS OF RESEARCH METHODOLOGIES .................. 27
3.2 CHOICE OF THE CASE STUDY APPROACH.................... 27
3.3 CASE STUDY FOCUS.................................................. 28
3.4 CASE STUDY SCOPE .................................................. 28
3.4.1 Context of VSE.................................................. 28
3.5 CASE STUDY STRUCTURE........................................... 28
3.5.1 Methodology triangulation...................................... 29
3.5.2 Data collection.................................................. 30
3.5.3 Data analysis method........................................... 32
3.5.4 Software Development Life Cycle............................... 33
3.6 VALIDITY AND RELIABILITY ....................................... 33

4 Findings and analysis ....................................................... 35
4.1 TARGET GROUP .......................................................... 35
4.2 QUESTIONNAIRE .................................................... 35
4.2.1 Overall Results of software unaware VSE and analysis of Questionnaire: 38
4.3 OBSERVATIONS.................................................... 42
4.3.1 Overall Results of software unaware VSE and analysis of Observations: 42
4.4 INTERVIEWS .......................................................... 43
4.4.1 Overall Results of software unaware VSE and analysis of Interviews: 46

5 Discussion and conclusions ............................................... 50
5.1 DISCUSSION OF METHOD ........................................... 50
5.2 DISCUSSION OF FINDINGS ........................................... 51
5.2.1 Questionnaires.................................................. 51
5.2.2 Observations.................................................. 51
5.2.3 Interviews.................................................. 52
5.2.4 RQ1 .......................................................... 52
5.2.5 RQ2.................................................. 52
Contents

SAVSE response to Question 11 ................................................................. 70
SAVSE response to Question 12 ................................................................. 71
SAVSE response to Question 13 ................................................................. 72
SAVSE response to Question 14 ................................................................. 73
SAVSE response to Question 15 ................................................................. 74
SAVSE response to Question 16 ................................................................. 75
SAVSE response to Question 17 ................................................................. 76
SAVSE response to Question 18 ................................................................. 77
SAVSE response to Question 19 ................................................................. 78
SAVSE response to Question 20 ................................................................. 79
SAVSE response to Question 21 ................................................................. 80
SAVSE response to Question 22 ................................................................. 81
SAVSE response to Question 23 ................................................................. 82
SAVSE response to Question 24 ................................................................. 83
SAVSE response to Question 25 ................................................................. 84
SAVSE response to Question 26 ................................................................. 85
SAVSE response to Question 27 ................................................................. 86
SAVSE response to Question 28 ................................................................. 87
SAVSE response to Question 29 ................................................................. 88
SAVSE response to Question 30 ................................................................. 89

Table of Figures - Response from SAVSE to Questionnaire

SAVSE response to Question 1 ................................................................. 60
SAVSE response to Question 2 ................................................................. 61
SAVSE response to Question 3 ................................................................. 62
SAVSE response to Question 4 ................................................................. 63
SAVSE response to Question 5 ................................................................. 64
SAVSE response to Question 6 ................................................................. 65
SAVSE response to Question 7 ................................................................. 66
SAVSE response to Question 8 ................................................................. 67
SAVSE response to Question 9 ................................................................. 68
SAVSE response to Question 10 ............................................................... 69
SAVSE response to Question 11 ............................................................... 70
SAVSE response to Question 12 ............................................................... 71
SAVSE response to Question 13 ............................................................... 72
SAVSE response to Question 14 ............................................................... 73
SAVSE response to Question 15 ............................................................... 74
SAVSE response to Question 16 ............................................................... 75
SAVSE response to Question 17 ............................................................... 76
SAVSE response to Question 18 ............................................................... 77
SAVSE response to Question 19 ............................................................... 78
SAVSE response to Question 20 ............................................................... 79
SAVSE response to Question 21 ............................................................... 80
SAVSE response to Question 22 ............................................................... 81
SAVSE response to Question 23 ............................................................... 82
SAVSE response to Question 24 ............................................................... 83
SAVSE response to Question 25 ............................................................... 84
SAVSE response to Question 26 ............................................................... 85
SAVSE response to Question 27 ............................................................... 86
SAVSE response to Question 28 ............................................................... 87
SAVSE response to Question 29 ............................................................... 88
SAVSE response to Question 30 ............................................................... 89
# Introduction

This research was conducted during the last semester of a 2-year education on advanced level in product development: Software Product Engineering Program. The purpose behind this research is to explore the barrier of capturing domain expertise input from a software unaware Very Small Enterprise (VSE), as a software development team. Due to limited knowledge fields and personnel, many VSEs who desire a software tool tend to not have any member knowledgeable in software development processes, views and terminologies, which creates a communication gap between software developers and them. To tackle this problem, a case study will be conducted to understand the current situation. This chapter will cover the subject of the research, the purpose behind it, the research questions, the research objectives and finally, the delimitations of this research.

## 1.1 Background

As software developers, it is necessary to properly and effectively communicate with other stakeholders of an enterprise, to capture all the knowledge needed to facilitate the development of software products. Irrespective of the field in which the development process occurs, software developers must embed the knowledge of the domains they are working with. As said by Lepouras [27] one of the main challenges software developers must face is acquiring that Domain Knowledge (DK) which can be obtained from the domain experts found among the stakeholders of an enterprise.

There are, however, differences between large scale and small-scale enterprises, one of which is the communication between developers and stakeholders. Sánchez-gordón [25] say these differences in turn affects the process of knowledge sharing and how the developers must change their approach, to facilitate it, and it is particularly challenging when working with VSEs. Another important factor which influences knowledge sharing is the level of depth and awareness of software processes and terminologies among VSEs as per Laporte, Claude Y [31].

In general, VSEs, depending on their fields of work and previous experiences, we (authors of this research) divide VSE as either software aware (SA), or software unaware (SU) only in terms of their experience/working with in software development. Software aware VSE (SUVSE) can be described as a company which has worked with software development teams, has been in contact with the software industry, and has a degree of understanding of software development methodologies, terminologies, and processes (cross ref 2.5.2). Where Software Unaware VSE (SUVSE) is a company which has no understanding of software development in general (cross ref 2.5.1).

According to Laporte [1] the problem with SUVSE when developing a software tool is that they will either have very few people for the developers to communicate with properly and efficiently, or none. This communication gap will affect the relationship between developers and clients and will slow down the process.

In the context of SUVSEs, when looking at stakeholders, Laporte, Claude Y [31] says due to their(client) lack of understanding of certain terminologies or concepts, they misunderstand what is meant and what should be done. For instance, clients consider that because they can make changes all the time as they want, they should also do it. The clients, in this case, fail to understand the importance of counterbalancing the customer involvement. Hoda [10] mentions they (clients) don’t understand that changes are not easy to make, or even possible at times, given the constraints the development team must work with.
Introduction

The main issue is in the challenge of communicating with SUVSEs which lack knowledge or expertise in software development, to secure the domain expertise needed as a foundation for the design, framework, and development of a software product. Furthermore, because of the continuous feedback needed to properly work in an agile development, the impact and changes on the SUVSEs must also be considered.

According to Laporte, Claude Y [26] normally, VSEs are comprised of staff focused on certain fields of expertise and lack the staff and skill set to cover other fields of knowledge. However, little research has been done on how to approach a VSE which has no experience in software development process. Researchers Paucar, Luis García, Laporte, Claude Y, Arteaga, Jaylli, Bruggmann, Marco gives detailed explanation of implementing ISO/IEC 29110 standards in project management and software development which improves software development practices and management in agile process for a VSE [24] and mentions some VSE might fail achieving it for various reasons. An author Fajardo, J in a blog gives a set of guidelines for to maximize the productivity of subject matter expert [14] but lack validity it the guidelines really work in reality since it’s only a blog.

It is crucial for development teams to understand how they are supposed to approach VSEs to avoid delays, misunderstandings and confusions from the very beginning. Lastly, it is important to understand the difference of mindset between a developer’s point of view, and that of the client, since each one has a different view of the project and have different approaches to it, thus reaching a common point between these views is also vital.

1.2 Purpose and research questions

Sharon [29] mentions in software development, a link between the developer and the client must be established and maintained from the very beginning of the process and both sides must stay in touch often and must be able to understand each other to facilitate the process. However, this is the first barrier which must be overcome, since quite often, the client will not have the experience with dealing with developers or with the software development process in general. In this case, it is up to the development team to fill in the knowledge gap.

The purpose behind the research began after working with both SA and SU VSEs. What was noticed is that, compared to SA VSEs, there were more difficulties in exchanging knowledge and bridging the gap between developer and stakeholder, in SUs. The process of knowledge gathering took longer and a lot of time was lost due to misunderstandings or explanations. From this observation, an idea was formed:

“What exactly is the major difference between working with SA and SU VSEs, regarding knowledge sharing? Is it the understanding of software development processes and terminologies?”

The objective of the research is to study the interaction between software development team and software unaware domain experts in VSEs during software development, to better understand the barrier of knowledge that exists in terms of domain language, development process. In addition, it is important to understand how the DE’s knowledge will change because of the involvement and presence in the software development process.
Current studies Fajardo [14] Paucar [24] Laporte [26] Lepouras [27] give reasons or advantages of involving the client and explaining roles/ responsibilities of every person involved in development process for a successful project but none of them provide guidelines, references, or recommendations for communication with clients who do not have any software background (cross ref 1.1) (cross ref 2.1).

Thus, the findings of the research would be useful and affect communication with client area in software development process.

Based on this, the research can be formulated in the following questions:

RQ 01: What topics are software unaware very small enterprise (SUVSE) aware of, in the software development process?

RQ 02: Which aspects of the software development process must be known by a software unaware domain expert of a very small enterprise (VSE), so that a software development team can capture the required domain knowledge (DK)?

RQ 03: How is the software unaware very small enterprises (SUVSEs) understanding of the software development process affected by their involvement in an agile development methodology?

Thus, the findings of the research would be useful and affect communication with client area in software development process.

1.3 Context

As shown in Fig. 1, based on this idea, a literature survey was conducted to better understand the problem and find possible solutions. Unfortunately, the results we found were unsatisfactory, since they were discussing VSEs in general, without a focus on the SU ones, and without adequate ways of solving the issue raised. As such, the reason for the research was determined.
Introduction

Looking at the background and the problem domain, one theory could be constructed: in the context of the software development process, a DE needs to be software aware, to a certain extent. The sampling base of enterprises will be based on this assumption. It also serves as a model, based on experience and the literature reviews.

The context of the thesis work will be around a small number of VSEs, including NIO and Jens-Mårten Company Sweden (JMCS). These companies lack a proper understanding of the software development process at the beginning. What will be done is to assess the knowledge the VSEs have regarding the software development process and what product they want at the end of it, to gradually identify the lacks in understanding between the developers and VSEs from the beginning. Then, as the production stage progresses, help them fill these gaps to make communication easier, and to avoid misunderstandings. These steps are done by having the clients directly involved in the process, in an Agile method of software development. Finally, the body of knowledge attained by the VSEs at the end of the production must be assessed, to better understand how this will aid the VSEs in future projects.

The research was conducted in collaboration with few SA and SU VSEs, with a focus on understanding the SU VSEs. In the context of this paper, a SUVSE (Nordisk Industri Optimering - NIO) was chosen to work with, on a software prototype which the VSE requested. The VSE in question is a consultant, a sub-contractor which was outsourced some work from a large-scale manufacturer (Volvo, Saab etc.), in the same field of industry in expertise with FMEA (Failure Model and effects analysis) domain.

Since the consultancy work being performed is done manually, using PDFs, word documents and excels, the process is slow and tedious. As such, the VSE requested a software prototype which can automate and digitalize the work for VSEs, thus greatly reducing the amount of work needed, while also reducing the possibility of human error. Looking at the challenges and problems faced by similar VSEs, due to a lack of personnel and resources and estimating no profit, the prototype will have a large impact digitizing work thus following consistence standards across all VSEs. Fig. 2 gives a diagrammatic representation of overview of the context of the research.

Figure 1 Background - Approach of research

![Diagram](image-url)
Once the request was made for such a prototype, a small development team engaged the VSE to begin the development process. The initial approach chosen was the general one, assuming the stakeholders, and more importantly, the Domain Experts, had some understanding of software development processes and terminologies. Shortly after, during the initial meetings, it was noticed how the process of knowledge sharing was slow and ineffective, due to the lack of understanding between the development team and the stakeholders of the VSE. It is important that the Domain Experts and developers can properly communicate and exchange knowledge.

In the product requirement context, the chosen VSE knows a significant amount of the whole DK needed for the prototype. Other VSEs could also contribute with the missing parts, as each VSE focuses on a different aspect of the same domain. Below, Fig. 3 shows the visuals of DK areas NIO over FMEA domain and overlap of other VSE to maximize the content of DK in specific area.
According to Lepouras [27] Domain Experts (DE) have specialized, in-depth knowledge of a business field which enhances a software development team’s understanding of a given process. DE plays a critical role in providing domain expertise, defining business needs, test requirements, and software functionality during software development. The expert is also an important stakeholder for a software development team, that has recognized expertise and authority for a given business process. By our (authors of this research) experience we could sum up and explain DE as a trainer, end user, or knowledge sharer, and can contribute to designing the architecture and acceptance tests for a software system.

1.4 Delimitations

The first delimitation will be number of people involved. The first delimitation will be number of people involved. A total of 4 SUVSE companies and 1 SAVSE company, from which 18 members from SUVSEs and 5 from SAVSE will be participating. A target group of domain experts from VSE’s will be taken for the research and due to availability, the number of people being interviewed will be limited. To support the research’s accuracy, the main target group of the domain experts will be of a software unaware background. In addition, only a small number of VSEs will be involved in the study.

The second delimitation will be the size of the project to be considered. Taking time and lack of personnel as factor, and implementation not to be the priority, a concentrated research on a small 6-week project will be carried out. An increase in the scope of the project would affect the results, as the research team is undermanned and cannot process too much data.

To solve the first delimitation, as many VSEs looking for a software project must be approached. NIO, mainly, will be asked for support in getting more contacts, though
other options are possible. For the second delimitation, the scope of the project will be restricted, and only a concept demonstrator for NIOs product will be aimed at for implementation.

1.5 Outline

Chapter 2 will give an overview of the theoretical framework. The following topics will be introduced and discussed: VSE, software unaware specific VSE, domain experts, Software development process, Agile methodologies (Extreme Programming), Activities of DE in software development process, Important Aspects DE should be aware of.

Chapter 3 will give an explanation and describe what methods have been chosen and what they involve, giving references to the various methods for answering the research questions and achieving the purpose. An overview of how the case study was performed will be included.

Chapter 4 will report the findings and the results of the research work, and their subsequent analysis.

Chapter 5 will cover discussions and conclusion and possible future work.
2 Theoretical framework

In software development, there are many factors to be considered before and during it, to facilitate the optimum production of a software product. It is not enough that a client has an idea for a product and, after communicating it to a development team, they get to work. Without the proper knowledge, data, and the comprehension of the actual desire of the client, the product can end in failure even before it is completed. To avoid this scenario, the roles of each contributing stakeholder must be understood, and their effectiveness improved.

This is achieved by having all sides understanding each other and keeping in touch constantly throughout development. To do so, the developers must understand the constraints and mentality of the company. There are differences between what approaches to use between a VSE and a larger company, both in terms on experience, knowledge and personnel.

To better understand how to approach VSEs regarding software development and obtaining DK, one must first investigate what research is already done about this topic. Having related cases and studies performed could well serve as a reference. Other than that, understanding the nature of VSEs is also crucial. VSEs are a different entity than a larger, more robust enterprise. It is precisely due to its limitations caused by its reduced size and access to resources and manpower that defines VSEs. What’s more, SUVSE could prove even more troublesome due to a lack of base knowledge and experience in software industry and software development. Once the higher level is understood and analyzed, the next challenge comes from the domain expert and the DK itself.

Domain experts are key stakeholders in the software development process, since they possess vital knowledge needed to properly and correctly develop a high-quality tool designed for that specific domain. Approaching domain experts and gaining their knowledge is both crucial, and at times troublesome, due to the rather unorganized and complex fashion of the DK presented. It is also a problem that domain experts in a VSE are both few and usually busy, so the meetings and their times must be carefully managed.

Their lack of knowledge regarding software development terminology, processes and notations also slows things down. Feedback must be done often to ensure that the functionalities of the software perform correctly and adequately. Test cases must be written to check for bugs or inconsistencies. But for all these steps to be performed efficiently, the software development process itself must be suited and tailored for VSEs and their limitations, as well as for the nature and scale of the product desired.

2.1 Related Studies

There are guidelines provided on how to improve effectiveness of stakeholders in the software development process only with respect to software inspection by Fajardo [14] but fails to address the issue or gap between stakeholders and software development team. To be specific, what is aimed to be improved is the communication between clients and development teams and the understanding of the software development process by the client. What was covered was about assigning activities and responsibilities, while also organizing procedures and meetings. Regarding the involvement of domain experts, a study by Lepouras, George [27] explains the key roles the DE has while also showcasing the importance of the DE and some of the problems
Method and implementation

faced when working with them, but its drawback is that it is not performed on VSEs. However, there is a lack of material regarding domain experts in VSEs. Studies conducted by Paucar [24] and Laporte, Claude Y [26] that have proven the effectiveness of applying standards in VSEs and successfully delivering the intended projects have not gone into detail as to how they acquired the DK needed from experts.

Based on studies by Fajardo [14], Paucar [24] and Laporte, Claude Y [26] tasks that must be covered by the stakeholders include:

- Review test cases for integration testing.
- Help validate user requirements for payroll application.
- Conduct code walkthrough for accounts payable interface to legacy system.
- Review requirements traceability matrix and ensure that they have coverage.
- Help refine and determine feasibility, correctness, and completeness of end-user's requirement.
- Provide input for the design and construction of test cases and business scenarios.
- Help answer questions associated with the design of the application, its features, and its capabilities.
- Validate executed test results.

In addition to tasks, meetings between stakeholders should be standardized and adhere to a certain process, to avoid confusion and delay. Guidelines and schedules should be established beforehand, and everyone involved should know what to do during these meetings. Procedures must also be applied to peer reviewing, to refine test requirements and test cases, and ensure that all questions are answered.

Next step is about the lessons learned from either mistakes or mishaps during implementation, and how such knowledge should be documented for later reference and use. This also ensures that inputs from various stakeholders are not ignored. Finally, there is also the task of choosing domain experts based on their expertise and knowledge, and which of them should be assigned as contacts for software development teams. Factors that influence the selection of a DE by the client, from his side, by Fajardo, J [14] include:

- Domain Expertise
- Business Process Expertise
- Methodology Expertise
- Availability
- Authority

Given the lack of research and knowledge in this area, our (authors of this research) research could further add more value, with the existing studies used as reference. Based on these references, an approach can be identified and applied to validate or even improve current practices based on the references found.

The focus of this research is to identify key aspects and problems regarding VSEs, and the challenges faced by developers in order to approach such companies, adapt to them, and obtain the DK needed while also encouraging the DEs to be more active.
2.2 Domain Expert (DE)

According to Lepouras [27] Domain Experts (DE) is an individual who possesses a deep understanding of DK in a certain field, through years of working, studying, researching, and experience. A DE can be observed as a vital source of information in any field of any enterprise, and especially in VSEs, since that person will be a key figure for the company’s performance. Fischer [15] mentions in VSEs, a DE will typically be assigned the role of a person of reference, and even as a supervisor in the CEOs place, given our experience. Depending on the nature of the software tool required, a DE can also act as an end-user, and provide requirements, acceptance tests, meaningful feedback, and even aid in the architecture design and models of the software tool developed as we have noticed happen before. Another aspect to consider would be the need for DK in multiple areas. For example, if a robot is developed, it would require multiple fields of knowledge: physics, mechanics, electronics, machine learning and embedded software. This means that some experts involved in the project might not necessarily have the knowledge required in all fields used. Henceforth, multiple DE could be required to cover all fields of knowledge needed.

2.2.1 Challenges faced with DEs

There are a few constraints associated with DE’s (from a software unaware background) compared to software developers, based on our (authors of this research) knowledge and experience. DEs possess a different mindset and view of the world, due to their experience and educational background. Moreover, they are under distinct domain and work limitations.

According to Fischer [15] DEs normally show a lack of curiosity in concepts such as software development life cycles, diligent testing procedures, etc., and are neither interested nor willing to spare considerable time learning about these concepts as they find them a short-term need and are already set in terms of knowledge base. Because of this, non-technical DEs have difficulty with software development notations, terminologies and procedures. However, this could also lead to most DEs remaining passive mediators of knowledge, rather than being more actively involved in the process.

Another constraint is that DEs with a large set of DKs may be a disadvantage, by Laporte, C [1] due to their knowledge confining them to a field of knowledge and their view, in which the solution and image of the software tool does not reside. Overcoming it would require a different mindset, based on a different set of knowledge, from a software development perspective. In other words, DK may act as a mental barrier, encouraging a fixed view in creative problem-solving attempts.

Final concern is their general lack of time. Begier [12] says they (clients) are always busy performing work in their field of knowledge from our (authors of this research) observations. This will cause a problem. Since DEs are vital for providing domain expertise, they are vital for the development of a software tool. However, they will not always be available.

2.2.2 Domain expert vs subject matter expert

It is often found that the term DE and subject matter expert are used interchangeably, in many fields. In the software development and design fields, the term DE is more often used. There is, however, a subtle difference between the two terms. A subject matter expert can be described as a person who is focused on a particular category or
topic, whereas a DE will be a person who focused on a whole discipline or sphere of knowledge.[14][27] For this research, the term domain expert (DE) will be used.

2.2.3 Domain expert vs stakeholders

From our (authors of this research) knowledge and experience, we can say that the DEs are crucial stakeholders in the development of software products, as they act as a prime source of knowledge. There are of course differences between various stakeholders, although it is entirely possible for an individual to fulfill multiple roles, and thus act as multiple stakeholders. It is important to note however the distinctions between stakeholder roles and how having multiple roles can influence one’s role.

2.2.3.1 DE vs User

If the DE is also one of the intended users, it will help simplify the process, since the DE, as a user, will know what to expect and what to demand. However, that is not always the case, since most users will have a lesser understanding of the domain as a whole than the DE.

In this case, the DE can be viewed as a User Proxy (UP). As described by Mike Cohn [32] User Proxies are needed when the actual users intended cannot be acquired to test the product, and thus require representatives. A stakeholder can adopt this role in addition to their initial one, such as DEs or marketing for instance. While UPs do fill the gap, they will have different views of the product based on their initial role and knowledge. A UP with an IT background will have a different approach from a UP who is a DE.

Another element to consider is where in the hierarchy is the UP situated. A manager will have a different flow and view of the project than a DE or a sales person. In addition, a DE’s approach will be for the software product to cover as much of the DK specifics as possible. Mike Cohn [32] says this could be a problem if it falls out of the scope of the project, and thus demand more functionalities than initially planned or even needed, which in turn will increase the amount of time and resource needed for the product to be developed.

Lastly, while DEs are an asset, their usefulness as UPs greatly depends upon whether they are former or current users of the software type being developed. Mike Cohn [32] mentions the reason for this lies in the changes and differences between the systems the DEs are used to, know, and trust, and the system being developed. Their view and approach will be affected since the differences between the old and the new system could make their decisions questionable. If possible, their approach should be compared to that of user of similar type systems.

2.2.3.2 DE vs Client

If the DE is also a Client, the development process can be greatly improved. As said by Lepouras [27] if the Client is already a DE, they will know what they want from the product, and will have a better image of the product in their mind. As a DE, they will be a valuable source of knowledge, without necessarily having a DE intermediary between the developer and client. This also helps with the acceptance tests needed, since the client will know better what the results should be. The only problem will be that the client might not always be available, so over relying on them could pose a problem which we (authors) experienced before.
2.2.3.3 DE vs Customer

According to Mike Cohn [32] Customers are essentially the individuals responsible with the buying decision and are not necessarily the users of the software product. It is important to know and consider the desires of the customer since it is, they, and not the users, who will purchase the product, unless the customer and user are the same.

If the DE is a Customer as well, they will be more demanding of the quality of the product and its functionalities as said by Lepouras [27], so meeting their desires might be more challenging, which in turn may make the development process longer and costlier.

2.3 Domain Expert involvement

In agile development, Begier [12] says the product vision might be mutually developed by a collaboration of software team and domain experts. A vital step is acquiring DEs due to their knowledge. Through them, the development team can understand how the product is supposed to work, as it is normal in agile. Begier [12] also mentions the required feedback is not constrained to working out the specification of requirements, but it should also cover the whole software life cycle.

2.3.1 Roles and responsibilities

One of the first steps in an agile development is establishing roles and responsibilities between the development team and the client and stakeholders as said by F. Tripp [16]. Based on the software development life cycle, the development team will have a spokesperson or coordinator, while we (authors of this research) have seen the client will have a CEO/Owner in many cases was appointing a “product champion”, a person from his side that can offer more detail and vision about the product needed, and that will be available for feedback and on-site meetings. It is vital that such roles are established early, to understand who is responsible for what.

2.3.2 Vision of Knowledge gathering

Initially, the larger scope of the vision will be provided by the CEO, or the “product champion”, for the development team to understand what it is they will be building, and who is the targeted user, given our (authors of this research) expectations. As the vision gets narrowed down, the role will fall to the “product champion”, who can provide additional detail, or point to the person most appropriate at the time.

2.3.3 Information Retrieval

Based on our (authors of this research) experiences and contact with stakeholders in general, the Information Retrieval (IR) takes place once the following questions are understood:

1. what the system being developed should solve?
2. what functionalities and inputs will be needed for it to work properly?
3. what the system should not do?
4. what are the system limits and expected amount of data to handle?

This is done in several meetings between a team of developer and client representatives. From our (authors of this research) knowledge, this will typically include the spokesperson of the development team, along with one or two other members, who could have additional questions regarding the system or act as scribes,
and a few domain experts, end users, and the “product champion”, possibly even the CEO if available.

It might be necessary to first establish a terminology which can be used by both sides, so that all stakeholders involved can understand each other, given our (authors of this research) experience with VSEs. Generating models and user stories will prove challenging at first, since the client needs to be educated on what these are, why they are needed and if they truly reflect what the client needs. Information Retrieval is not necessarily over at the beginning of software development, as changes can always occur at any step during the production, as we have experienced before.

In this phase as described by Lepouras [27], DEs have the responsibility equivalent to:
- defining the data required by the service developed
- writing the rules for verifying user input for acceptance tests
- defining the backend processes needed for the software logic
- writing all possible outcomes for unit testing
- defining and knowing what data must be collected for the software architecture

2.3.4 Feedback

As part of the agile manifesto, continuous and meaningful communication and feedback between developer and client is vital. Fowler [18] tells it is necessary that the stakeholders are actively involved in the software development process, to ensure that what is developed meets the current expectations and requirements, and what should be changed or improved as time goes on. Periodic and often meetings are thus necessary, and it is important that by now the communication gap between developer and client is crossed in terms of comprehending each other.

2.3.5 Acceptance tests

From our (authors of this research) understanding, once the first modules of the tool are about to be finished, it is up to the client’s side to start creating a series of acceptance tests, to check whether the product meets the expectations or not. An issue arises here, as many clients will not understand what acceptance tests are, why they are important so early, and how they should be formed. Attention from the development team is necessary at this step, to ensure the clients understands what he wants and what the product should be capable of. Since the client has DEs who will also act as end users, the task will fall to them, especially to the “product champion” allocated.

2.3.6 Test Requirements and Test Cases

There are instances where stakeholders, especially DEs, could help in writing out test requirements and even test cases which will be used to validate the code developed as per our (authors of this research) understanding. Much like acceptance tests, the problem will be how to formulate them, and figuring out all the necessary requirements.

2.3.7 Software Product Architecture

DEs can also participate in writing the software architecture in the early phases. Fischer [15] explains due to their (DE's) knowledge in the domain and their understanding of how the software tool should look like, they can also help model out its structure. To achieve this however, the development team will need to explain what
the architecture model looks like, what it should and should not do, and how it should be modularized.

2.4 Domain Knowledge

According to Lepouras [27] Domain knowledge, in the context of this report, refers to the valid and specialized knowledge possessed by a domain expert, with regards to a certain field. Fischer [15] explains this knowledge is then used to build the domain model, necessary for the software tool. When discussing software development, it is important to understand how vital this base of knowledge is regarding the tool developed, what value it possesses and how it impacts the software development process. Fischer [15] mentions Domain expertise will also impact the functionality as well quality of the tool developed. However, that depends on how efficiently the DK is obtained and understood by the development team.

From what we (authors of this research) have experienced, DK usually must be learned from software users in the domain (as domain specialists/experts) and from company documents, and literature on the subject. It includes artefacts such as user workflows, data pipelines, business regulations and policies, configurations, limitations and constraints and is crucial in the development of any software application. An expert’s DK, after being understood by the development teams, is then transformed into software and active data.

Given our (authors of this research) understanding and point of view, the problems with DK lie both the knowledge, and the expert. Experts tend to frequently present the DK informally and ill-structured. This is because compressing all the knowledge needed by a software development team, which has little to no base knowledge in that domain is certainly not easy. Lack of time and responsibilities of DE shows impact in the way information is exchanged. As for the DK itself, it varies in both depth and complexity, depending on both which parts are needed for developing a software tool, and the field of the knowledge itself, from our (authors of this research) experience.

2.5 Very Small Enterprise

According to Laporte [1] Very Small Enterprises, or VSEs, are defined as companies with less than 25 employees in total and are typically focused on certain fields of expertise or production industries. By our (authors of this research) experience we could add another characteristic as their lack of departments, since limited personnel pretty much forces them to either have all their members focused on the field the company has chosen, and nothing else, or some members must fulfill multiple roles, to a certain limited extent. Laporte [1] mentions VSEs are the backbone of industries, since much of the work of larger companies is eventually outsourced to VSEs, for them to focus on other tasks and fields. In fact, roughly 80% of the world economy is based on VSEs from Laporte [1], and as such, the importance of understanding how to improve communications with them in the software development process is critical. Also, with our (authors of this research) experience, we consider due to VSE’s limited resource and manpower, software development tools are critical for the functioning and efficiency of VSEs.

For this research, VSEs will be divided into two categories: SUVSE and SAVSE. Though the focus is on the SUVSE, SAVSE are also considered, to highlight the difference, and importance, of having prior knowledge regarding software development, and how that helps the exchange of knowledge between stakeholders and developers. To reflect that
Method and implementation

distinction, the questionnaires and interviews will be separated for each type, and the results will then be compared.

2.5.1 Software unaware VSE

From our (authors of this research) perspective, a software unaware VSE is a company which has little understanding of software in general. Since such companies are usually occupied with their fields of work, they don't usually concern themselves with the software industry or the software development processes in general. Such companies make up the bulk of VSEs in general, since there are many other fields of industry outside of software. Though most VSEs are dependent of software to function properly, that does not mean that the individuals comprising it understand how software is made, what the process is, and even how coding works. These prove to be the most difficult to work with, given our experience, as the stakeholders need to be taught about software development processes, methodologies, terminologies, help them assign roles and create acceptance tests, gain the DK needed, etc. This lack of technical knowledge and experience can also discourage such VSEs from adopting the ISO 29110 standards.

2.5.2 Software aware VSE

From our (authors of this research) perspective, a SAVSE is a company which has worked with software development teams, has been in contact with the software industry, and has a degree of understanding of software development methodologies, terminologies, and processes. The SAVSEs are thus familiar, at least to an extent, with software development processes in general. Such companies are easier to approach, since they already have a base knowledge on software development and have gained some much-needed experience and knowledge on the topic. However, even such companies will show some degree of difficulty to work with, due to the lack of resources and personnel that is typical of VSEs. Normally, such companies will have an intermediary with a good understanding of software processes and methodologies, to present to the development team immediately. SAVSE will also be more willing to apply the ISO 29110 standards, due to its experience, as studies show. [24] [26]

2.6 Software Development Life Cycle

From Blog Itinfo.am [13] the software development life cycle (SDLC) is well described a framework or plan for defining tasks which must be performed at each step of a software development process. It is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to design, develop, maintain and replace specific software. In other words, SDLC is methodology designed to improve both the quality of software, and the overall development process. From Blog Itinfo.am [13] defines the software development life cycle can also called as the software development process.

2.6.1 Chosing an approach

When researching what software development approach to use for our (authors of this research) research, several options were presented such as Agile, Waterfall, V-Model and Spiral. To understand which one was the most adequate, we analyzed each of them.

According to Dybå[2] agile software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers or end-users. agile software development advocates adaptive planning, evolutionary development,
early delivery, and continual improvement, and it encourages rapid and flexible response to change.

According to Dybå [2] the waterfall model is a relatively linear sequential design approach for certain areas of engineering design. Dybå [2] also strongly argues in software development, it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction ("downwards" like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, deployment and maintenance.

Dybå [2] describes the waterfall development model originated in the manufacturing and construction industries, where the highly structured physical environments meant that design changes became prohibitively expensive much sooner in the development process. Dybå [2] says in the early stages adopted for software development, there were no recognized alternatives for knowledge-based creative work.

According to Itinfo.am [13] in software development, the V-model represents a development process that may be considered an extension of the waterfall model and is an example of the more general V-model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. Dybå [2] says the V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represent time or project completeness and level of abstraction, respectively.
According to Boehm [9] spiral model is a risk-driven software development process model. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, or waterfall. Boehm [9] mentions spiral model combines some key aspect of the waterfall model and rapid prototyping methodologies, to combine advantages of top-down and bottom-up concepts. Spiral model can also be seen providing emphasis in a key area many felt had been neglected by other methodologies: deliberate iterative risk analysis, particularly suited to large-scale complex systems.
Of these software development processes, the one chosen for the thesis approach was Agile. The reasons for this are as follows:

Waterfall and V-model are older processes which don’t highlight the importance of communication throughout development. In addition, their rigidity doesn’t allow them to adapt to changes or unexpected situations in general, and their general one-directional flow doesn’t allow for preemptive measures before development.

As for spiral, though it adapts elements from process models to fit the requirements of the project, it is best suited for large-scale systems, and large-scale teams. Given the scope of the thesis and of the software project within it, it was deemed redundant.

On the other hand, the Agile approach allows for greater flexibility and it fits the smaller scope of the project as well. Another key feature is its priority on communication with the client/end-user. Lastly, the number of methodologies that fit under Agile from which we can choose, to best fit the needs of the software development process.

2.6.2 Agile Methodologies

According to Dybä [2] Agile software development describes an approach to software development based on close collaboration between organized, cross-functional teams and the different stakeholders involved, for requirements and solutions to be created and gradually evolved. It supports adaptive planning, evolutionary development, short iterations for early deliveries and continuous improvements. It emphasizes the need to be flexible and open to changes, since that is often the case in real life scenarios.

Eichhorn [38] mentions agile development essentially has four core values.

1. Individuals and interactions over processes and tools:
   From Eichhorn [38] any project consists of people, processes and tools naturally and have equal importance in building a software. Each person may use different tool and can contribute to any process. However, regardless the software succeeds or fails it’s the whole team shares the credit and not the individual.

2. Working software over extensive documentation:
   From Eichhorn [38] documentation is no exception in software development. Customers are not interested in reading long unrelated lines of text. Thus, consider what is valued to customer and document them.

3. Customer collaboration over contract negotiation:
   From Eichhorn [38] the closer you work with customer the better you build the product. So, instead of having a contract negotiation have collaboration with customer.

4. Responding to change over following a plan:
   From Eichhorn [38] it is very normal to not have strict plan to build dream project, and it is always better to change plan and aim for the quality product instead sticking to old plan knowing it will fail.

In this study, though all points are relevant and contribute to obtaining the vital domain knowledge and information needed for the software development process, the highlight will be on the interactions between individuals and customer collaboration.
It is important to note how the other core values are also supporting factors, since they impact what will be discussed and how with the customers.

Thanks to the values promoted by agile development, several methodologies have surfaced. Each of these methodologies is different in the types of projects they specialize, the terminologies used, the artefacts they are focused on, what approach they focus on, and how interactions between the stakeholders take place.

Dybå [2] mentions the most popular of these at the moment are:

1. Extreme Programming (XP)
2. Scrum
3. Rational Unified Process (RUP)
4. Dynamic System Development Method (DSDM)
5. Feature Driven Development (FDD)

2.6.3 Choosing methodologies

Considering the limitations at hand, both from the company, and us, as well as the desired tool to be produced at the end of the research, it was determined that the XP and SCRUM approaches are most suited for the task at hand. The reasons for this are as follows.

Firstly, the tool itself is not a large scale, complex project, and any methodology focused on that would not be necessary. Secondly, the output of the development will be a concept demonstrator or prototype at best, meaning the code should be simple. Thirdly, the company itself lacks for the most part any in-depth knowledge of software development, and as such, the meetings held must also take that into account. Next, the company itself does not strictly adhere to ISO standards, and in return won’t demand too much documentation, so any methodology focused on documentation will be redundant. In addition, the purpose of the thesis has to do with the interaction between developer and customer, so getting the company specialists, and representatives involved in the project is vital.

One other thing to consider is that the client is not aware of the software development process or terminology, so everything must be kept simple and clear. Any methodology that has a complex structure will be more confusing for the clients as well. Lastly, the small size of the development team is also a factor, seeing as how, for this research, there are only 2 developers available. Methodologies that are focused on large scale projects or a multitude of teams will not be beneficial.

Based on the body of knowledge and our (authors of this research) experience possessed with Agile methodologies, both XP and SCRUM are favorable approaches, due to familiarity and ease of use. As there are only two developers available, they must fulfil multiple roles at same time if required.

Since XP and SCRUM were chosen as the agile methodologies to be used, these two will be further explained.
2.6.4 Extreme Programming (XP)

According to Wood [3] XP is a methodology for creating software applied within a very unstable environment, because it allows flexibility within the modelling process. The main goal of XP is to lower the cost of change in software requirements. Wood [3] describes with traditional system development methodologies, such as the Waterfall methodology, the requirements for the system are determined and rigidly set at the beginning of the development project. This results in higher costs to change the requirements at a later stage in the project.

Itinfo.am mentions XP attempts to reduce the cost of changes in requirements by having multiple short development cycles, rather than a few, long cycles. Wood [3] says changes are a natural, inescapable and desirable aspect of software-development projects, and should be planned for, instead of attempting to define a stable set of requirements and as part of the agile methodology, highlights interaction between the developers and the customers, having the customer on site and always available. In simple words it focuses on keeping code simple, and reiterating it when necessary, and often testing it in order to reduce failures before presentation. One other key aspect of XP programming is communication; the need to have the customer be in touch throughout the entire process and relying on constant feedback.

According to Sfetsos [4] the XP methodology is based around a set of practices:

1. Shared understanding :
   1. Simple design
   2. System metaphor (story than anyone involved can tell about how the system works)
   3. Collective code ownership (everyone is responsible for all the code)
   4. Code standards and conventions

2. Fine scale :
   1. Test Driven Development (TDD)
   2. Planning Game (planning process, once per iteration)
   3. Whole Team (customer presence in the development is mandatory)
   4. Pair Programming

3. Continuous process :
   1. Continuous Integration
   2. Design improvement (focus on simple, generic architecture)
   3. Small Release

When discussing XP, it is important to note the existence of 3 similar, but distinct, practices of this methodology, namely XP1 (traditional), XP2 (evolutionary), and XP corollary (including modern practices, and advices, mainly found on the internet). The later can be considered an addition to the former two practices, providing helpful additions or changes, based on the nature of the software tool being developed, the scale of the process and the nature of the relationship between the client and the developer. Lutz Prechel [21] mentions in some cases, the corollary XP is a combination of the two practices and can be used individually. Each practice, though similar, has distinctions, and that includes the processes or steps performed to gather DK.

2.6.4.1 Domain knowledge gathering - XP1

Given our (authors of this research) knowledge, when applying XP1, the practices relevant are those involving the customer’s involvement in the process, namely the planning game and the on-site customer. In the planning game, the project will be divided into several releases, which each release focusing on completing certain functionalities. To know what these functionalities will be the DK must be obtained.
Although the entirety of the DK needed cannot be obtained before the development commences, the bulk of will be obtained over the course of many meetings.

As for the on-site customer process, this is required to fill out the more refined details and knowledge needed when developing, which can’t always be covered in meetings. We believe domain experts will be needed on site whenever developers require more in-depth understanding of the knowledge needed to complete their tasks.

2.6.4.2 Domain knowledge gathering - XP2

Using our (authors of this research) experience, we can say that, when applying XP2, the practices relevant are those involving sit together, whole team, and user stories. Much like XP1, having the customer (or DE appointed) present in the meetings and with the team is a necessity. The whole team needs to be formed of members focused and specialized on their tasks, and this includes domain experts required to fill the gaps in the body of knowledge required. Furthermore, having them present in the team means they are available and close by when needed, and it simplifies communication with them. In addition, it encourages them to be more active in the process, and since the knowledge needed can be provided on the spot in a matter of minutes, rather than organize another meeting, means the process can also be sped up. For the bulk of the DK needed however, domain experts provide this via User Stories, at the beginning of the project. When demanding functionalities, we believe they must also explain what exactly will be needed and expected from them.

2.6.4.3 Domain knowledge gathering - XP corollary

Regarding the corollary XP, the practices used can vary. Seeing as how at times XP can become a hybrid of XP1 and XP2, we believe corollary XP can use XP practices regarding DK gathering. However, at times, corollary XP can also be an addition to the former two, since developers, over the internet, can come up and propose additional practices which can improve the process. The addition relevant to DK gathering is real customer involvement, which is a form of on-site customer and planning game in one single process, given our (authors of this research) experience.

2.6.5 Scrum

According to Mundra [5] Scrum is an agile method for project management whose goal is to dramatically improve productivity in teams previously paralyzed by heavier, process driven methodologies.

From the blog itinfo.am [13] typically, Scrum is characterized by:

1. A living backlog of prioritized work to be completed
2. Completion of said backlog priorities in a series of short iterations (or sprints)
3. Brief daily meetings (called scrums), where progress is explained, upcoming work is revealed, and obstacles are presented
4. Brief weekly backlog session where the backlog items for the next sprint are defined
5. Brief retrospective where all team members reflect on the past sprint

According to blog mountaingoatsoftware [23] another key feature of Scrum is that it is facilitated by a designated scrum master, whose primary job is to remove impediments to the ability of the team to deliver the sprint goal. The scrum master is not the leader of the team, since the team is self-organizing, but acts as a productivity buffer between the team and any destabilizing influences. Additionally, the scrum master is the one most in contact with the customer, and is also the one who organizes all meetings, or sessions.
Blog itinfo.am [13] describes that scrum enables the creation of self-organizing teams by encouraging verbal communication across all team members and across all disciplines that are involved in the project. A key principle of scrum is its recognition that fundamentally empirical challenges cannot be addressed successfully in a traditional “process control” manner. As such, scrum adopts an empirical approach - accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team’s ability to respond in an agile manner to emerging challenges.

2.6.5.1 Domain knowledge gathering – Scrum

According to blog mountaingoatsoftware [23] when applying Scrum, it is important to understand the way it is structured, and how the methodology works. The first thing to point out is the role assignment. In Scrum, there are 3 main roles: Scrum master, responsible for focusing the team on the sprint they are working on and improve the process, the Scrum development team, which is an umbrella term describing the group of individuals of various skills and specializations tasked with a sprint, and the Product Owner, who represents the business, customer or user, and is responsible for guiding the team towards developing the right product. For DK gathering, we will investigate the Scrum team and the Product Owner.

From the Scrum Team’s perspective, they are a group of individuals with various skills and bodies of knowledge, who work together as a whole to complete a sprint. Though individuals may possess various titles, those are meaningless in Scrum. According to blog mountaingoatsoftware [23] the structure of the Scrum team works like this: everyone is tasked and will work mostly (or entirely) on the tasks suited for their experience, knowledge and skill. It is however possible for an individual to cover more than one role. Though it is impossible to find a know-it-all individual, many of them strive to be as all-knowing as possible.

Based on our (authors of this research) experiences, one individual that is part of the team will be the domain expert. The DE will be involved in each sprint, with the main (and possibly solely) task of providing the DK needed to complete the current sprint. If the DE is an expert in multiple fields relevant to the project, then he can provide knowledge for those fields when required. The DE is also important when writing the backlogs, as DK will be needed when planning out the functionalities needed in the future. We believe he could be a key person in meetings, aiding the Product Owner when needed.

From the Product Owner’s perspective, he could make use of DEs whenever needed during the sprint planning meetings. It is during these meetings that the DE can be heavily involved if not part of the Scrum team. One other possibility is that the Product Owner himself can be the DE, in which case the sprint meetings can proceed faster. This can be a burden for the Product Owner however, as he must cover an additional role.
2.7 Software development process in SU VSEs

2.7.1 VSE limitations

According to Laporte [17] one of the first constraints regarding VSEs is that they are economically vulnerable, as they are driven by cash-flow and are dependent on project profits, so they need to perform the projects within a typically tight budget. Their tendency to have low budgets will have several notable impacts, such as: lack of funds to perform corrective maintenance; few resources allocated for staff training; little or no budget to perform quality assurance activities, software reuse processes, risk response, and limited budget to perform Process Improvement or obtain certification and assessment to name a few.

Based on our (authors of this research) observations due to their (VSE’s) restricted size and lack of variety of specializations and suffer from a lack of resources and focus outside their field of knowledge. Though they are the backbone for many large companies, organizations and corporations, we believe that this lack of variety of specialization or knowledge can impact the higher chain.

In addition, the lack of both resources and manpower also means that VSEs cannot follow many standards set up by ISO. To do so would ruin them, as they cannot allocate more energy into tasks deemed unnecessary and expensive than they have. Though this does not have a great impact on the VSE itself, the higher chain of command enterprises, such as SMEs, and even higher, will. Paucar mentions [24] these companies are forced to follow these standards and have the resource to do so. But the lack of documentation for instance from VSEs can cause misunderstandings and discrepancies. Also, based on the nature of VSEs, some do not even have to follow standards as strictly as others as said by Laporte [17].

2.7.2 Challenges of Software Development Process in software unaware VSEs

Laporte [17] describes VSE products to typically have a single customer, who is in charge of both the management and maintenance of the system and its operations. For most of the time the customer, do not define high quantitative and quality requirements and the customer satisfaction is dependent completion of specific requirements, which are open to change during development. Sánchez-gordón mentions [25] the relationship between all project members involved in the development process in VSEs is strongly human oriented, due to the more personal involvement of higher-ups of the VSE, and not just with employees. This shows that communication is vital.

Sánchez-gordón [25] describes the first instance this will become apparent, and set the tone of the first meetings, will be during the requirements and domain expertise gathering. To describe in brief the “language barrier” will force the development team to consider how much software development process information they can expose a SUVSE in an agile development methodology, for them to acquire the necessary information, while also assuring the VSE understands what it is being asked without confusing it.

The next instance of a challenge emerges during model building mentioned by Sánchez-gordón [25]. To describe in brief a software tool is generally an intricate set of methods bounded together to achieve several functionalities and features demanded by the client. A good practice is to break down the tool into a few modules, each with its own purpose and dependencies. The problem lies in how the client can comprehend
what these models are, how are they relevant to them, and what changes or feedback can they propose based on their understanding of them. Models must be made detailed enough for the developers to understand how the code should be structured and what each module should look like, while keeping them simple enough for the stakeholders to understand what’s happening in them, to propose meaningful feedback when needed and to avoid misunderstanding.

According to Laporte [17] another challenge is in the client’s understanding of what the tool is supposed to do, or not which means most clients in SUVSE will have a simplistic understanding of what the software tool they want should do or look like, since they do not, and cannot, view the tool from a software perspective. Because of this, they will likely initially be focused on what the tool should do but will miss out on what the tool should also not do as well. As a solution Laporte [17] tells it is important for the development team to understand this and question the VSE on matters such as what the input format or limit should be, or what exceptions, borderline and extreme cases will there be, since these must be included in the documentation and models beforehand, so that the development team can begin developing with the understanding of what the system limits are.

From Laporte [31] one other concern could be the lack of experts as DEs are in very small numbers, in some cases VSEs only having one DE relevant for the software tool developed. Due to their small numbers, combined with their general lack of time, obtaining DK from DEs can prove problematic. It is vital to manage their time, for them to provide valuable domain expertise and meaningful feedback for the software development team, throughout the software development process.

Additionally, VSEs have a typical reluctance to follow standards. Paucar [24] clearly explains this is because most standards are designed by, and for, large companies. VSEs on the other hand do not have the manpower or resource to properly follow and apply them. As such, as part of an agile approach, the developers must adapt to the client’s needs and situations, and work around them. Paucar [24] describes the lack of technical knowledge and experience in software development can also discourage VSEs from adopting the ISO 29110 standards, thus forcing developers to work around it and try to adapt to the client’s needs.

Lastly, there will be a difference of views on what is considered vital or top priority between developers and clients as mentioned by Laporte [1]. The clients will focus their attention on the user interface, the input system and outputs produced, and what the user thinks of the high-level aspect of the tool. The developers will investigate deeper aspects, such as what technology would best be suited for the client’s needs, what modules, libraries and functionalities will be needed to process the expected input and generate the desired output, how the code should be structured and optimized, and so on. It is important that the SUVSE understand what will be needed to create their desired tool, and what priorities the development team will have, when pushing forth their set of priorities as well. This will set the expectations of the VSEs to more realistic levels.

2.8 ISO Standards - ISO 29110

2.8.1 Definition
According to publicly available standards [19] International Standardization Organization (ISO), is responsible with creating guidelines, documentations and procedures which can be consistently used to ensure that the production process, materials, products and services generated are suitable for use and are optimized.
Method and implementation

Since ISO standards apply to all fields and domains, VSEs also have an ISO dedicated for it, to improve the processes and products of them. This paper will investigate the existing ISO standards and check to see how valid they are.

According to the ISO documentation [19], based on previous studies and surveys performed, it is evident that most international standards do not properly address the requirements of VSEs. Both the implementation and conformance of most standards, regarding VSEs, is difficult or even impossible in some cases. This is due to the limitations of VSEs, which hinder their possibility of complying with standards applied to larger enterprises of the same fields. Therefore, VSEs have little to no ways of being recognized as entities capable of producing quality products, services, or systems. It has been found that VSEs have a hard time to relate to the standards required by their business, due to the effort required to apply them in practice.

To address at least some of the difficulties faced, a set of guidelines has been developed based on the characteristics of VSEs. These are mainly based on subsets of standards, activities, tasks and processes, referred by the ISO standards as profiles. These profiles are meant to define a subset of ISO standards relevant to the VSEs context. Examples of these include processes, task and activities for software (ISO 12207), for systems (ISO 15288), and information products for both software and systems (ISO 15289).

Paucar [24] argues if the VSEs are to be recognized, they should implement a profile, and be audited against, the ISO 29110 specifications. The ISO 29110 series of standards can be applied at any phase of both software and system development, within any lifecycle, including agile. This ISO is intended to be used by VSEs which have little experience or expertise in adapting the existing ISOs for larger entities. Those that do have the expertise needed are encouraged to use those standards over the ISO 29110.

For more details on the structure of ISO 29110, check Appendix 2.

There is evidence to support the usage and the success of ISO 29110, when used by IT-based (or software aware) VSEs. However, the question then arises: why aren’t more VSEs applying a standard designed for them? According to Laporte [26] studies show that, in 2015, 18% of VSEs follow the ISO standard and are certified, while 74% where pointing out the importance of being certified and recognized. Yet, despite having the desire to be certified, and having an available standard, most do not follow it.

2.8.2 Problems with adopting ISO 29110

The reason why this is the case has less to do with the nature and core of the ISO standard, and more with the nature of VSEs and how they perceive standards in general. Although the ISO 29110 documentation claims that it is structured for VSEs, considering their particularities and weaknesses, that may sometimes appear to be at surface level. Essentially, what the ISO 29110 has done was to take the standards, regulations and procedures used by similar ISOs for large-scale companies and edit and customize them to fit smaller enterprises as said openly in publicly available standards [19].

However, this might feel more like a stop-gap than a proper solution, and why VSEs might perceive it as such and why they do not comply to ISO standards. At their core, the “borrowed” procedures, rules, regulations and documentations are designed for large enterprises. As pointed out before, VSEs have limitations specific to them, which lead to an unwillingness to comply to strict regulations. Paucar [24] argues as such,
procedures must be written from the very core to fit those restrictions, and not just changed from existing ones not fit for VSEs.

In addition, it must be considered that such a standard must not be as demanding and add a degree of freedom in order to both suit and coerce VSEs into adopting and applying them. Laporte [28] mentions one other factor is, given the vast variety of VSEs, with their multitude of fields and specializations, the standard should be flexible. As for the requirements for each field of work, a subset of standards specific for each field should be made, based on existing standards, but more suitable for VSEs. Since VSEs do the bulk of production for much larger companies, it should be up to them to field the gap in procedure requirements which the VSEs alone are unable to fulfil entirely.

From the VSEs perspective however, they tend to use process models in a sort of textbook fashion, choosing to either drop elements or change them according to their wants and needs. On the other hand, software engineering practices are chosen opportunistically as well. This has to do with the agile methodology focusing on prioritizing customer satisfaction, through early and continuous releases.

There is additional evidence to support the claim that the majority of VSEs are not adopting standards because they generally perceive them as being created by large companies and organizations and are thus suited for them instead of the VSEs. More studies such as Paucar [24] Sánchez-gordón [25] Laporte [26] Laporte [28] show that VSEs have a negative perception of process model standards due to bureaucracy, costs and documentation. Finally, it has been reported that VSEs find it difficult to relate their business needs to international standards and to justify the application of them in their operations, due to perceived costs and stress on resource and manpower.

In short, though there is evidence to support the efficiency of ISO 29110, the standard has yet to be adopted at a large scale due to the negative image VSEs have towards standards.
3 Method and implementation

3.1 Options of research methodologies

In modern times, several research methodologies have been developed, tested, and validated, offering researchers a variety of approaches for observing, solving, researching and improving aspects of any and every field. Based on the focus of the research, its nature, and the field it is being performed in, each research methodology fits for certain needs. It also depends on whether the research is qualitative or quantitative by nature. In addition, one must look at the type of research performed: basic research, applied research, problem-oriented research, or problem solving. According to McBurney [33] Cohen [36] Bell [37], some of the most popular approaches are:

- Case Study
- Action Research
- Design Research
- Experiment
- Literature Review

The main research method used to answer the research question in this thesis will be Case Study, supported by Literature Review.

3.2 Choice of the Case Study Approach

To understand why the Case Study methodology was chosen, the background of the thesis theme must be understood. The purpose of this research is to investigate and analyze the interactions between software development teams and DEs in VSEs for the purpose of information gathering, and how that SDLC can affect the VSEs knowledge base during development. Due to the nature of the research, which is optimizing communication and exchange of information between software developers and the other stakeholders, it has been concluded that it is qualitative oriented, as it is focused on collecting, analyzing and interpreting data, by observing what people do and say, which fits the thesis. Semi-structured interviews will be one important strategy in the research.

Then, the type of research must be determined. In this case, the research is problem oriented, as it seeks to know the exact nature of problem that is required to be solved. By problem, according to Bell [37] it refers to the dilemma of how software development teams should interact with DEs in VSEs for DK gathering. It does not fit with basic research, as that is focused on generating or enhancing knowledge, nor with applied research which focuses on solving practical problems on a large scale, or with problem solving research, which is performed by individual companies for their specific problems.

Design research was not adequate, as said by Cohen [36] that would involve developing a tool and measuring the impact of it, action research focuses on a loop of providing a solution and receiving feedback, experiment would require a hypothesis and variables to test out to see if it holds true or not, and for literature review there is not currently much to investigate. Due to these constraints, and the nature of the thesis research, it has been determined that a Case Study approach would best fit, as explained by Cohen [36] the idea is not as much on giving a solution as much as it is on focusing on
exploring the process of domain expertise gathering in a software development process, and possibly coming up with a set of references to be used in future work with VSEs.

3.3 Case Study Focus

The focus of it will be on the process of domain expertise and requirement gathering between a software development team and a SUVSE client with little to no knowledge in the software development process. These must be obtained through a series of meetings and interviews, backed up by several feedbacks to validate the knowledge obtained. In addition, the changes in the VSE itself must be monitored, to see how it is affected by the process and how they knowledge and the relationship between them and the developers change over time. The case study will be performed by a small team of developers and several SUVSE, over a period of 6 weeks, during the design and development of a software process tool for the VSEs. The interactions, limitations of communication and changes to the client will then be documented, analyzed, and a conclusion drawn. Based on the conclusion, several references can be deduced in order to aid software development with VSEs.

3.4 Case Study Scope

There are a few factors to look regarding the scale of the Case Study. To begin with, the sample size used for it. Approximately 5 VSEs will be used, each with several representatives. Next, the time duration. Due to the constraints under which this thesis was written, a long-term case study was not possible. However, a snapshot of approximately 6 weeks will be used as the time scale, to grasp the general look of the software development process. Third and final, the nature of the case study. Though a multi case-study scenario would have been more appropriate, due to the lack of more VSEs, especially software aware ones, combined with the lack of time needed to properly manage it, a case study has been adopted, using a main sample for the study itself, and a smaller separate sample for validation.

3.4.1 Context of VSE

The VSE in the context of this research is one located in Skövde, called Nordisk Industri Optimering, or NIO. NIO is a VSE focused on automotive industry consultancy, on three branches: Maintenance, Tooling, and Quality. This company is comprised of 17 individuals at the time of the thesis, of which two CEOs and two domain experts. The company has tasked the software development team with the creation of a tool used to greatly reduce the manual work and cumbersome nature of their consultancy tasks. Through this company, the development team will be put into contact with other VSEs NIO is working with, to increase the scope of the research and to properly validate the findings. In addition, to ensure the validity of the results, a second set of VSEs will be contacted. These ones will be software aware, however.

3.5 Case Study Structure

The Case Study will be performed with the use of semi-structured interviews, questionnaires, and on-site meeting observations between the development team and several VSE representatives, throughout the development process, to determine the
difficulties faced by software developers when working with inexperienced VSEs, as well as point out what should be done to correct them.

To begin with, the VSEs capability of communicating with the software development team must be assessed, in order to determine if the domain knowledge and information obtained would indeed be valuable, understandable and correct. One must look at how little most VSEs know about software development, and what objectives the developers must accomplish before going after requirements, and specifications for the software project itself. For this purpose, a questionnaire was set up for the two types of VSEs: the software aware, and the software unaware ones. The questionnaire is the same, but it was shared separately for the two VSE types, in order to compare results, and observe the differences between VSEs with experience in software development versus those lacking in it, to know the base knowledge of what the SUVSEs possess, before the study.

There was an initial meeting, for the developers and stakeholders to introduce each other, and discuss the initial requirements of the project. One additional purpose for these meetings was to form a general idea of the VSE’s understanding. We had some difficulties obtaining DK, due to the terminologies used, and the fact that there was no official documentation organized at all, to properly explain everything. Also, we have noticed how the VSE stakeholders had trouble understanding the developers when talking about technical details at times. After the first meeting was set, a structured questionnaire was sent.

Once the questionnaire was sent, three more meetings took place to discuss the software project, the application desired, the people we will be working with, requirements and so on, as the next step. The meetings went out just like a normal project with an average company, to see where it would lead to naturally. During those meetings, it was noticed how troublesome it was to properly acquire the DK needed for the project and later, explaining the stakeholders what the software architecture should look like, and how to integrate it with their plans, due to a lack of basic knowledge and terminology from the stakeholders.

A good portion of the meetings were also to explain software concepts, and to translate the view from a developer’s point of view in such a way that the stakeholders would comprehend, and then be able to add value to them in turn. Things such as UML diagrams, user stories or software architecture were mostly alien to the VSE, prior to the meetings. This slowed things down. However, as soon as the stakeholders were properly informed of what all this was, and more importantly, why it was relevant, the meetings proceeded much smoother afterwards. Due to the VSE’s lack of time, and personnel, all meetings required that most employees of the VSE be present, in order to make the meetings more efficient. This did however mean that the meetings were few and far apart.

From the SAVSE, whenever the CEO was not available, the project manager, Henrik Engqvist, would help.

3.5.1 Methodology triangulation

According to Bekhet [11] Methodology Triangulation is used to gather same information at different instance of time, in this research we use methodology triangulation to gather the same data of SUVSE knowledge of the software development process, at different instances (questionnaire, observations, interviews). The primary reason for considering more than one method is to understand and confirm the same situations with different point of views and scenarios. The focus will
be on gathering information from different methods and to conduct research of the same phenomenon to get a broader understanding. When using this methodology, one can get data cross referenced and validity checked in early stages. It is also possible to make conclusions based on where the different methods have a common denominator in the findings. According to Bekhet [11] Methodological triangulation is used to make sure that the analysis and clarification of the findings are improved. Additionally, the researcher’s insight is broadened when multiple sources are used, and it will strengthen the validity of the results.

3.5.2 Data collection

According to McBurney [33] Data collection is the process of gathering and measuring information on targeted variables in an established systematic fashion, to evaluate outcomes and achieve meaningful answers, while methods vary by discipline, the emphasis is always to ensure accuracy and validity. Our (authors of this research) goal for data collection is to capture quality evidence that allows analysis to lead to convincing and credible answers to the questions that have been posed.

3.5.2.1 Questionnaire

A structured questionnaire with closed questions will be conducted at the beginning of the case study, which consists standardized questions and all the participants are given the same set of questions. An overview of the VSEs knowledge of software development and understanding will be required, and as such, the “product champions” and CEOs, will be given a small questionnaire regarding the software development process in an agile methodology, what they thought of it, what they learned, and if it will help with future software tool developments.

Since there will be two types of VSEs questioned: those familiar with software development and those who are not, the questionnaire results will be divided for these two groups, to properly assess the software development topics which the VSEs are aware of, how that impacts the software development process, and the interaction between developers and stakeholders of the VSEs. These questionnaires can be very economical and well suited for the environment as domain experts will be busy most of the time, by help of questionnaire form they can answer the questions whenever they have free time.

Questionnaire will be made with help of Google Forms where we provide questions with multiple choice answers, where answers would be made according to the type of questions. For Example, If the question contains did you involve in anything? then the answers would be Yes, No or I don’t Know as such. This can be used as a reference to show the reliability and validity of the data (e.g., count the number of ‘yes’ or ‘no’ answers).

3.5.2.2 Observation

According to blog Study.com [22] Naturalistic observation involves observing the subject or the topic in their own environment, without changing the daily routine to capture real time observations as even a slight change in the environment would change the end results.

The data collected by naturalistic observation will be of instantaneous sampling which means data from observations will be recorded, using observer narrative methods. This
includes the observer taking notes during the meetings/sessions and extracting relevant information from the notes. No pre-structured or aimed data set is made before the observation. Everything happening before or after is ignored. Also, some factors which affect the communication, such as the language barrier, or the derail from the main topic due to the stakeholders by either explaining additional information or just having a casual talk or coming up with new ideas and so on were ignored as well.

In this research, naturalistic observations were conducted during the face-to-face meetings with the stakeholders. The process took place as such, one of us played the role of the developer, asking questions and teaching the stakeholders concepts and terminologies, while the other acted as the researcher and scribe during the meeting, taking note of the parts where the flow of the meeting was either interrupted or slowed down, due to misunderstandings, lack of knowledge, or problems grasping some concepts.

Throughout the case study, the relationship between developer and client will be monitored for changes during the software development process, through observations. The difference in how much information can the client be exposed to; how much can they be involved in the development process and how much do the meetings improve as time progresses will be noted.

3.5.2.3 Interviews

At the end of 6 weeks of interaction with VSE understanding them and helping them learn unknown topics, a semi-structured interview will be conducted. They will be initially conducted, with the CEOs of each VSE requesting them to give an overall answer based on whole team participated for the questionnaire, since the aim of the research is not focusing on one person instead on the whole VSE. Semi-structured interviews will be conducted throughout the case study in all phases of software development, in order to obtain the necessary requirements, expectations, feedback, and domain expertise from the VSEs. McLeod [35] describes Semi-structured interviews are mostly participant answer driven based interviews and can be prolonged or shortened based on his interest and are more versatile. It would help understanding the reason for their answer and would be more relevant to take them into consideration and make a conclusion for the case study.

The Interview will be an open type interview, with a focus on being qualitative and record people's emotions rather than obtain quantitative results. The theme of the interview was about the overall experience in working with software development, as well as what they think of the new insight obtained. A total of 6 questions were made, with an additional 7th if there was anything else the stakeholders wanted to add regarding the experience as a whole or they had any particular feedback to give. The interviews were conducted over a period of 2 days and lasted an average of 20 min. In the paper, only the relevant parts were included. A total of 4 SUVSE and 1 SAVSE companies were interviewed over the phone, usually including the CEO and 1-2 other stakeholders.
Method and implementation

3.5.3 Data analysis method

3.5.3.1 Questionnaire

Google Forms will be used as tool for analysis of questionnaire for the research. The participants provide answers with help of google forms for questionnaire, which further helps in analyzing the data.

Google forms offers a detailed analysis for each question by sorting the responses based on answers and provides the organizer of questionnaire with a chart plotting individual responses with total response in percentages (%) for a question to further make use of the information.

The results will be provided in coming chapters (cross ref 4.2), then after responses of each question are changed from percentages to scores from 1-10 where each unit in the score is equal to 10%. For example, 34% is equal to 3.4. This step is further explained in next chapter (cross ref 4.2.1)

Then questions form same topic are grouped together to measure a topic as whole and plot graph with respect to scores and topics which would give an understanding of SUVSE and SAVSE understanding over topics. This step is further explained in next chapter (cross ref 4.2.1)

3.5.3.2 Observations

These observations will be during our meetings and only the ones relevant to the scope of the thesis were kept. The relevance of the observation will be based on the background of the topic.

There are areas such as involvement in software development, terminology, communication, questionnaire topics, emotions, opinions should be considered as relevance since they are directly related to answering research questions.

Considering other areas would divert or over kill the data and could be identified for future study.

These observations will be then converted to percentages with respect to total number of people involved in the meeting and the number of people exhibit the same output to get quantitative data to present.

3.5.3.3 Interview

Each question will have answers form 4 SUVSE and 1 SAVSE. As explained by Maguire [39] Thematic analysis can be used to extract information from the interview answers to find a pattern to the question.

All the five answers will be checked thoroughly to find out keyword’s participant is trying to tell.

Each key word is highlighted from the answer and written down.

Combining the keywords from 5 participating VSE’s we can find a similar or opposite pattern for a question which participants agree or have an opposite view.

These patterns can be considered as a over all answer for the question which will further answer the research questions.
Method and implementation

3.5.4 Software Development Life Cycle

The SDLC used for this case study will be XP mixed with Scrum, with terminologies borrowed from Scrum and DSDM, for the software development process. The reason XP and Scrum was chosen as the agile methodology for this thesis is due to the limitations of both the development team and of the company, in addition to the lack of time.

For the demonstrator, FDD will not be needed since there will not be many features to implement, According to Boehm [9] Spiral is needed for larger, more complex projects, According to Duncan [8] RUP requires extensive documentation, for which there is not enough manpower for need for initially, and According to Rising [6] there are not enough members for a Scrum approach alone.

The reasons why XP and Scrum are the most adequate SDLC for the research are directly related to their principles. Test Driven Development requires client intervention in forming test cases and requirements. Due to the goal of creating a demonstrator, a simple design is required, with a simple process to make it. The thesis’s theme of analyzing the process of information gathering and customer involvement in the process is another reason why XP is also well suited. The need for small increments and frequent releases, along with continuous integration are also compatible with XP. Finally, since the work is done for a professional client in a professional environment, code standards must be put into place.

However, according to blog Itinfo.am [13] the Scrum organization of stakeholders involved, along with the weekly meetings and sprints, is better suited for the documentation and analysis aspect of the thesis. In addition, in Scrum, a stakeholder could hold multiple roles, as it allows more flexibility, which fits the constraints and limitations the thesis case study is subjected to. Lastly, Scrum terminology will also be borrowed, to better describe the structure of the stakeholders.

The reason why Scrum and DSDM terminology is being used has to do with the client. Using terminologies that are easy to understand for the client is important, as to avoid confusion. In addition, the development team of the thesis is also familiar with the glossary used.

3.6 Validity and reliability

For the validity of the results of the case study, we will rely on a combination of interviews, questionnaires and observations, and a “before and after” approach. In addition, the thesis will be using references on case studies as a potential benchmark.

3.6.1 Questionnaire

As described by Cohen [36] Questionnaires enhance the validity of the research due to them being more reliable, due to them being anonymous, which in turn encourages more honesty. They are also more economical, in terms of time, and they offer the possibility of being emailed.

3.6.2 Observations

From Cherry [34] Naturalistic observation involves looking at a subject’s behavior as soon as it occurs in a natural setting, without any intervention on behalf of the researcher and allows the researcher to observe events which cannot be reproduced in labs, which helps support the validity of the research, as the event is not manipulated.
3.6.3 Interviews

McLeod [35] describes interviews as a qualitative research method, where the validity is impacted by the interview style. A semi-structured interview is one with fixed questions, while also adding open questions based on each scenario. Thus, the interviews will have a degree of flexibility as well, where participants could be asked to follow up questions, such as “how”, “which” or “in what way”, thus adding depth to answers by allowing the participants to answer in their own way.

3.6.4 Scale

To ensure the scientific reliability of the thesis research, 5 VSEs (1 software aware and 4 software unaware), and their subsequent stakeholders will be involved. The stakeholders in question are a combination of appointed “product champions”, CEOs, and end users.

A few SUVSE will be considered as the main sample for the case study. The idea is that such VSEs represent the majority and are the focus of this paper. However, to properly check the validity of the results, in addition to having a small but reliable sample size of VSE representatives from several VSEs related in the same domain, at least 2 SAVSE will be used for cross referencing and validating. Such a VSE has had prior experience with software development, and thus has a base knowledge to work with.
4  Findings and analysis

4.1 Target group

A target group of 18 members will be participation from 4 SUVSE’s and 1 SAVSE.

4.2 Questionnaire

A structured questionnaire was made by looking over concepts of how DK must be gathered using agile methodologies mentioned in theoretical framework (cross ref DK in agile). The questionnaire was used to assess the base of knowledge of VSEs regarding the software development process, methodologies and terminologies.

From previous experience working with software projects and development of authors, a base list of suggested topics is been made which authors consider to be adequate to expose knowledge of VSE’s over software development:

1. Basic questions must be asked to know whether a person was exposed to any kind of software development process and if yes then record their experience.
2. Research is intended to carry out an agile methodology thus basics of agile development questions are to be asked to understand where companies are standing before research.
3. Instead of directly introducing agile workflow it would be good to know if companies know anything about other software development methodologies and do they understand the benefits of using different software development methodologies?
4. Software architecture is heart of development and response from companies as of how they give importance to architecture related questions might be vital.
5. When company approaches developers for a product or business in the first phase, requirements and priorities are must. Understanding does companies have basic knowledge over priority tables and requirements related questions such as User Stories, RFP will be beneficial.
6. During the development period critical times where developers and domain experts must meet and talk regarding progress basic questions regarding backlogs, terminologies and sprint meetings gives a good amount of understanding over companies and their lack of knowledge.
7. And finally, few topics such as delivery, roles and responsibilities like who is responsible for UI finalization can be made.

Based on the list above, a questionnaire is formed, and all the questions are made only on a basic level, since going deeper might take more time and cross track research. Several topics have been eliminated or removed such as Business Model Canvas, Budgets, Risk Management, Retrospective, looking at priority and relevance for research. Considering time and personnel as factors, authors have chosen 11 topics considered to be adequate for the questionnaire, in order to better understand the VSEs.

A total of 30 questions were asked.

The following table contains the questions asked for the participants grouped accordingly to the topics:
### Table 1 Questionnaire

<table>
<thead>
<tr>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic</strong></td>
</tr>
<tr>
<td>Were you involved in any software development process before?</td>
</tr>
<tr>
<td>How good was your communication with Development Teams?</td>
</tr>
<tr>
<td><strong>Agile</strong></td>
</tr>
<tr>
<td>Do you know of Agile software development?</td>
</tr>
<tr>
<td>How frequent you think client and development team should meet to show progress?</td>
</tr>
<tr>
<td>Do you know the principle of &quot;fail fast, fail often&quot;, and why it is relevant to the software development process?</td>
</tr>
<tr>
<td><strong>Software Development Methodologies</strong></td>
</tr>
<tr>
<td>Do you think a Development Team should tell the Client briefly in advance about software methodologies?</td>
</tr>
<tr>
<td>What do you know of software development methodologies?</td>
</tr>
<tr>
<td>Do you consider that having a good understanding of the basics of what software development methodologies are is useful for the software development process?</td>
</tr>
<tr>
<td><strong>MUSCOW Priority Tables</strong></td>
</tr>
<tr>
<td>Do you know what the MUSCOW table is, and why it is important to the software development process?</td>
</tr>
<tr>
<td>If you know what the MUSCOW table is, who do you think should be involved in writing it?</td>
</tr>
<tr>
<td><strong>Requirements</strong></td>
</tr>
<tr>
<td>Do you know about Request for proposal?</td>
</tr>
<tr>
<td>Do you know what User Stories are, and why they are relevant to the software development process?</td>
</tr>
<tr>
<td>If you know what User Stories are, who do you think should write them?</td>
</tr>
<tr>
<td>If you know what User Stories are, how much time do you think should be spent on writing them in the beginning? Consider a small-to-medium sized project</td>
</tr>
<tr>
<td>If you know what User Stories are, how often do you think they should be updated, whenever changes occur or more functionalities are added?</td>
</tr>
<tr>
<td><strong>Backlogs</strong></td>
</tr>
<tr>
<td>Do you know the term &quot;backlogs&quot;?</td>
</tr>
<tr>
<td>Who do you think should keep track of backlogs?</td>
</tr>
</tbody>
</table>
**Discussion and conclusions**

<table>
<thead>
<tr>
<th><strong>Sprint Meetings</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you know the term called Sprint meetings?</td>
</tr>
<tr>
<td>Do you think Sprint meetings are useful?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Terminology</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Every field has its domain terminology. If you were developing a software of your field (say Automobile), do you think you should use major terminology from same field (Automobile) or daily usage words from your company?</td>
</tr>
<tr>
<td>Do you think you should take inspiration regards style/design /terminology/concepts from competitors or from the same field of expertise?</td>
</tr>
<tr>
<td>Would you compromise if the software you asked for shows up different than expected, because of terminology error between you and development team?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Software Architecture</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you know what UML diagrams are, and what they are for, regarding the software development process?</td>
</tr>
<tr>
<td>Do you know the term software architecture?</td>
</tr>
<tr>
<td>Do you think you should know about software architecture?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Delivery</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you think it would be useful if the Development Team gives you documented details of the software after finishing the project?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Myths</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>How do you think software development teams or companies will react when told to add more functionalities at a later stage in the development process, beyond the initial plan?</td>
</tr>
<tr>
<td>Who do you think has control over the software being built and take final decisions at crucial situations?</td>
</tr>
<tr>
<td>Who do you think should decide the aspect of the User Interface?</td>
</tr>
</tbody>
</table>

Detailed results for the questionnaire are displayed in Appendix 3
Discussion and conclusions

4.2.1 Overall Results of software unaware VSE and analysis of Questionnaire:

Results from questionnaire can be found in Appendix 3.

There were two identical sets of questions, one for SUVSE, and the other for SAVSE. Google forms was used to make the questionnaire, since it was the easiest and best way to gather the results, without disturbing the VSE domain experts. It also enables the analysis and filtering of the data faster, compared to manual analysis by using google response views.

Response for a question is automatically processed and pictorially represented as a pie chart mapped with number of people answering same option. For Example, for below question Do you think sprint meetings are useful? 55.6% of people answered Yes (Which means they know sprint meetings importance, and is an expected right answer from authors), 33.3 % answered I Don’t know (Which means they do not know sprint meetings importance, and is an wrong answer from authors point) and 5.5% answered May be (Which means they have no idea if sprint meetings are useful or not, and is an wrong answer from authors point).

![Pie chart example response for questionnaire](image)

Figure 7 Example response for questionnaire

In analysis a max knowledge of 100% is been set on each topic which authors consider the right answers.

All the results for individual topic were summed up and were given a score from 1-10 dividing them with number of questions and total points they scored in perceiving the correct knowledge (Refer to Appendix 3).

For example, in the sprint topic, there are 2 questions. All the participants scores are added up from 2 questions which gave the right answer (6.1+5.5) and divided them with the number of questions (2) gave the result (5.8) and plotted on right answer column and obviously wrong answer column is equal to difference of Max knowledge and right answer column.
Below Graphs shows the analysis of questionnaire from SUVSE’s and SAVSE

Overall there if we could category topics on the nearest scores there would be 3 categories. Lowest less than 4, Middle between 4 and 5.5 and Highest above 5.

We as authors expected SUVSE’s would score highest in basic and delivery topics and topics Delivery scored highest with 7.7 score followed by Spring Meetings with 6.1 whereas Backlogs 5.7.

Irrespective of the fields, SUVSE’s have similar pattern/expectations of work or understanding in Delivery, Spring meetings and backlogs topics which helped them to score more points.

Topics requirements and terminology scored 4.3 and 4.2 respectively standing in between highest and lowest scored topics.

We as authors expected SUVSE’s would score Lowest in Myths and Terminologies and topics Muscow scored Lowest with 8.5 score followed by Software methodologies with 8.4.

In the similar way not having terminologies or similar ways of workflow in their fields led SUVSE’s to score lowest in Muscow and Software methodologies.
Discussion and conclusions

Below Graphs shows the analysis of questionnaire from SUVSE’s and SAVSE

Overall there if we could category topics on the nearest scores there would be only 2 categories. Lowest less than 8, and Highest above 8.

We as authors expected SAVSE’s would score highest in basic and Agile topics and topics sprint meetings, software architecture, delivery and Myths with a max knowledge of 10 out of 10.

Working with software team before and learning new topics or terminologies from them (development teams) were found to be the key reasons for the performance in the highest score in questionnaire.

We as authors expected SUVSE’s would score Lowest in software methodologies but to surprise topics basics and Muscow scored very Lowest with a score of 6 followed by Software methodologies and requirements with 7 and 6.8.

SAVSE’s from the beginning are expected to perform well on the basic questions on the chosen topics, as expected the results were satisfying through there is a room for improvement SAVSE’s clearly shows experience improves understanding over topics in software development in exchanging domain expertise.
Comparing both Software unaware and SAVSE Results:

![Graph comparing right answers](image)

**Figure 10** Comparing SUVSE and SAVSE questionnaire right answers

We as authors expected SUVSE’s and SAVSE’s would score similar in Basics and Delivery since they are basic questions and delivery topic require involvement of client in our case a VSE.

As expected, delivery topic from SUVSE and SAVSE are almost close whereas basics topic scores of both are not even close.

![Graph comparing wrong answers](image)

**Figure 11** Comparing SUVSE and SAVSE questionnaire wrong answers

We as authors expected SUVSE’s and SAVSE’s would score similar score in software methodologies and Muscov topics since they have unique terminologies and usage in software development and are complex topics for any VSE to have understanding.

There is a huge score gap between SUVSE’s and SAVSE’s answers for questionnaire in almost every topic thus, giving a clear idea how beneficial it is for a VSE in the involvement of software development.

Overall, we could conclude VSE’s learn new topics, understand the importance, and knows the values and their involvement while working with software development teams.
4.3 Observations

Overall Observations (Note – “they” in all the observation refers to SUVSE):

1. On average, roughly 75% of the SUVSE participants feel or say that they were somehow involved in software development projects (at least one project) or have been called up to work for a few days from tied up software aware companies when a software project showed up.
2. Despite saying they have experience with terminology, only 10% claim to have had no issues communicating, while the rest often used to get confuse or couldn’t remember.
3. Roughly 60% didn’t know about certain terms, such as Agile development, fail fast fail often, request for proposal, User Stories, MUSCOW tables. 50% did however know about UML diagrams, but 20% were not entirely sure why they were needed.
4. 75% are too active when a time comes to work over a topic (for example user stories). They would google up and understand in 2 minutes and speak with a half-based knowledge.
5. Around 30% said or considered that having a basic understanding of software development methodologies was not useful for the software development process, even after explaining them different kinds of projects are to be done in different ways. They feel they cannot understand them at this age, and they would be better off not understanding them.
6. Approximately 50% are scared to give their opinion over terminology or UX and consider that the UI should be decided only by the development team, which is not a good practice.
7. Around 25% believe that meetings should take place once a month.
8. Around 75% believe that when a software project requires unplanned changes, it is not a major problem, but instead only more time and money is all the developers will request or need. Only 50% considered the need of rescheduling plans.
9. Though 60% know about backlogs, but only 10% are consider backlogs are necessary for the development process.
10. 90% do know about sprint meetings, but from our observations, they will approach them as weekly meeting which take a long time, and gradually devolve into a fun, enjoyable gathering, rather than an organized planning session.
11. 20% believe that the terminologies that should be used are those within the company, over the ones used by the field of expertise they are working.
12. Over 50% are not entirely sure what software architecture is and just as many believe that knowledge of it is not necessary in the software development process.

4.3.1 Overall Results of software unaware VSE and analysis of Observations:

To sum up the results, roughly 65% didn’t have a solid knowledge of the software development process or considered that some practices were not necessary.

Observations were made while working with SUVSE throughout the research period. These observations were done during our meetings with NIO and were made during the development phases. Only the ones relevant to the scope of the thesis were kept.
Discussion and conclusions

4.4 Interviews

There was only one agenda for Interviews

“How was the overall experience working in a software development process?”

Concluding it, Table 2 was formed containing the questions for the interviews with regards to the agenda.

Table 2 Open Interview questions

<table>
<thead>
<tr>
<th>Open Interview</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>05</td>
</tr>
<tr>
<td>06</td>
</tr>
<tr>
<td>07</td>
</tr>
</tbody>
</table>

SUVSE response for the interview:

Table 3 SUVSE company 01 interview result

<table>
<thead>
<tr>
<th>SUVSE Company 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>05</td>
</tr>
<tr>
<td>06</td>
</tr>
<tr>
<td>07</td>
</tr>
</tbody>
</table>
### Table 4 SUVSE company 02 interview result

<table>
<thead>
<tr>
<th></th>
<th>S U V S E  C o m p a n y  0 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Good, got new topics to know and now we know more things which we can confidently speak with next software team.</td>
</tr>
<tr>
<td>02</td>
<td>We expected you to be busy and make things fast and fast. But now we understand you also work like us only but more professional way.</td>
</tr>
<tr>
<td>03</td>
<td>Few topics which you speak, we don’t understand and maybe they are useful, but it is always good to have to software terminology and topics aware person in our company whom we can refer you to.</td>
</tr>
<tr>
<td>04</td>
<td>Yes, many topics are good to know</td>
</tr>
<tr>
<td>05</td>
<td>Yes, they are logical and useful too</td>
</tr>
<tr>
<td>06</td>
<td>If there is no person who has software knowledge, they it is hard for us to talk according to you. Because you need to know more thing which will affect or what is correct and what is wrong.</td>
</tr>
<tr>
<td>07</td>
<td>It is always good to know new knowledge. It was a useful time being with you people.</td>
</tr>
</tbody>
</table>

### Table 5 SUVSE company 03 interview result

<table>
<thead>
<tr>
<th></th>
<th>S U V S E  C o m p a n y  0 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Since the project you have taken was on small scale, if we could have more time we could have more interaction. We feel working with you students nice because you listen and explain more, and we always have a comfort that you don’t think low about us. But if the time you spent was more we could have better working.</td>
</tr>
<tr>
<td>02</td>
<td>We thought we must give the document consisting of all our details and requirements and then you will start coding and show progress. But the way you have understood the problem talking with us</td>
</tr>
<tr>
<td>03</td>
<td>Definitely, terminology difference is there but way of expressing ideas is the same and your knowledge over technology is also beneficial in a way to say us whether it is possible or not.</td>
</tr>
<tr>
<td>04</td>
<td>Yes, there were many topics which are good to learn while working with software team</td>
</tr>
<tr>
<td>05</td>
<td>Yes, they were logical, as a person is new to the field he has to learn the basics and they were understandable.</td>
</tr>
<tr>
<td>06</td>
<td>Not so bad. Every time when two non-mixture of fields want to work together, it will be chaos. Now when we even have gained knowledge how to work with software team we think it would be easier next time.</td>
</tr>
<tr>
<td>07</td>
<td>Every software unaware VSE or a client has to be taken with extra care and software team must have ethics not to cheat or loo client instead look over problem as their own. Then it will be a healthy environment.</td>
</tr>
</tbody>
</table>
Discussion and conclusions

Table 6 SAVSE company 04 interview result

<table>
<thead>
<tr>
<th>SAVSE Company 04</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>05</td>
</tr>
<tr>
<td>06</td>
</tr>
<tr>
<td>07</td>
</tr>
</tbody>
</table>

SAVSE response for the interview:

Table 7 SAVSE company 01 interview result

<table>
<thead>
<tr>
<th>SAVSE Company 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>05</td>
</tr>
<tr>
<td>06</td>
</tr>
<tr>
<td>07</td>
</tr>
</tbody>
</table>
4.4.1 Overall Results of software unaware VSE and analysis of Interviews:

For Q1 How did you feel working or contributing for software a development team?

Exciting, knowledgeable. Our first software experience, it would be memorable.

Good, got new topics to know and now we know more things which we can confidently speak with next software team.

Since the project you have taken was on small scale, if we could have more time we could have more interaction. We feel working with you students nice because you listen and explain more, and we always have a comfort that you don’t think low about us. But if the time you spent was more, we could have better working.

It was good.

It is always good to work with a variety of field people in the company taking their advantages and improve the company.

**Keywords:** Excited, Improved knowledge, New Topics, Less time, comfortable, Improvement, Progressive, No issues.

End Statement: Everyone had a positive answer, claiming the experience to be exciting and useful. One company wanted to have more meetings to iron things out and gain more understanding.

For Q2 What are your thoughts on software development teams before and after?

We do not have previous experience, so it was all special and more excited to work with.

We expected you to be busy and make things fast and fast. But now we understand you also work like us only but more professional way.

We thought we must give the document consisting of all our details and requirements and then you will start coding and show progress. But the way you have understood the problem talking with us.

We were not expecting to work so early but day after day we now successful finished the work.

Few topics were new but even though you were student, you followed and know most the topics which professional companies follow like sprint meetings, backlogs, user stories, UML diagrams and way of capturing information.

**Keywords:** Expectations, Myths, Improved knowledge, New Topics, comfortable, Improvement, Progressive.

End Statement: Everyone was surprised to see that software development teams required frequent meetings to make sure the product would be delivered accordingly.
Discussion and conclusions

For Q3 Did you find the communication with a software development team somehow different with respect to your employees?

We initially were a little scared listening to terminology which we do not know anything of, what you people speak, but the way you have started slowly and explained us little by little helped us more understand over working culture.

Few topics which you speak, we don’t understand and maybe they are useful, but it is always good to have to software terminology and topics aware person in our company whom we can refer you to.

Definitely, terminology difference is there but way of expressing ideas is the same and your knowledge over technology is also beneficial in a way to say us whether it is possible or not.

We did not have many interactions, as of what we had it went all fine.

Not at all.

Keywords: Terminology gap, Myths, beneficial after explanations.

End Statement: 3 out of 4 interviewed were initially worried by the terminology gap that would occur by working with software developers. However, given our (authors of this research) approach, taking it slow and explaining all they need to know, they were pleased with the outcome.

For Q4 Do you think you have gained knowledge over how to work with a software development team?

Yes of course

Yes, many topics are good to know.

Yes, there were many topics which are good to learn while working with software team

Yes, it was informative and would like to participate if needed again to improve more knowledge

Not in a very big extent but a few terminologies.

Keywords: YES= They have Gained knowledge, beneficial.

End Statement: All SUVSE’s claimed to have gained a lot of knowledge from the experience. Since SAVSE already had exposure to terminologies, they were less surprised with basic terminology.
Discussion and conclusions

For Q5 Did the topics discussed in the meetings feel logical?

Some of them are useful, some of them are too advance for us.

Yes, they are logical and useful too.

Yes, they were logical, as a person is new to the field, he has to learn the basics and they were understandable.

Yes, we felt it.

Keywords: YES= They Felt meetings are logical, learnt new topics, beneficial. Some topics are too advanced.

End Statement: All claimed the questions and explanations were logical and useful.

For Q6 Did you find it hard to work with a software development team to exchange the right amount of domain knowledge?

The space you have given us to express our thoughts, help to contact easily. We feel there should be more time to spend only on understanding what is needed to be built and their dependencies is key to success.

If there is no person who has software knowledge, then it is hard for us to talk according to you. Because you need to know more thing which will affect or what is correct and what is wrong.

Not so bad. Every time when two non-mixture of fields want to work together, it will be chaos. Now when we even have gained knowledge how to work with software team, we think it would be easier next time.

Not do much you were too active to understand what we are saying.

Experience taught us how to work and what to expect from them. Now we got used to work with software team.

Keywords: Not bad (=not easy), space matters (give them time to express), active (help filling their thoughts).

End Statement: The results were mixed. 3 companies overall were able to pass on their domain knowledge, while the other 1 claim to exchange domain knowledge not so easy.
Discussion and conclusions

For Q7 Anything you would like to add?

Happy to work with you people, good luck.

It is always good to know new knowledge, it was a useful time being with you people.

Every software unaware VSE or a client has to be taken with extra care and software team must have ethics not to cheat or loo client instead look over problem as their own. Then it will be a healthy environment.

Informative, good to work.

No.

**Keywords:** Informative, Happy, Software teams should show ethics

End Statement: All were happy with the experience overall, while one company additionally claimed that there needs to be an ethic when software developers work with VSEs.

Looking at the results, it can be concluded that the SUVSEs were pleased with the experience and were glad to gain a better understanding of the software development process. Some expressed a desire to have additional meetings and reflected on what could be improved or done.
5 Discussion and conclusions

5.1 Discussion of method

The initial approach for use of questionnaires, observations, interviews turned out to be the correct approach to get the results. However, there was confusion in the beginning regarding what kind of data sampling should be recorded between event sampling, which aims at pre-decided data to be recorded, or instantaneous recording of data respectively. It was decided to opt for instantaneous data recording as it would cover more feelings, and emotions than event sampling which would help in validating the data.

The main aim of the research was to understand the challenge of communicating with VSEs which lack knowledge or expertise in software development, to secure the domain expertise needed as a foundation for the design, architecture, and development of a software product. The gathered data is enough to form preliminary answers for the research questions made in the beginning of the research.

Observations were the hardest to record and to conclude as there was a lack of personnel, with one of the two authors acting as a developer having a talk and the other author was acting as a researcher and noting down the data. Sometimes we miss out the researcher role and were acting both as developers and concentrating on project more than observing the DE difficulties over exchange of information. If the research was to be improved, the observation part would be set with aims and pre-focused topics, instead of doing it casually. The lesson learnt was not to go out of the assigned role, to not have a developer mindset only, and to be more flexible and considerate of other roles.

The decision to conduct an open interview with a casual talk instead of a formal or a preset of questions yielded extra statements which would not be possible with closed interview. To the end an overall answer was taken, as the aim was focusing more of a VSE than an individual person, but the authors (researchers) are eager to know if - the same interview was taken with each and individual participant will give a new set of results or not?
Discussion and conclusions

5.2 Discussion of findings

This section presents a discussion based on the results and findings of Chapter 4 – Findings and analysis. Each data acquiring method’s findings will be presented separately. The discussion will be used in the conclusion in chapter 5.3.

5.2.1 Questionnaires

The general consensus is that SUVSEs are lacking in both width and depth of software development terminology and process, especially compared to SAVSEs. This lack of knowledge not only impacts the effectiveness of communication between SUVSE stakeholders and software development teams, but it also affects their perception of the software development process altogether. [chapter 4.3.1, fig 6 & 7]

The SUVSE stakeholder’s knowledge in software architecture, MUSCOW, software methodologies, myths in particular were found lacking, with a scoring grade of 3/10 or less [chapter 4.3.1], which are especially important aspects to know when exchanging DK, planning the software development process, and understanding the stakeholder’s role in it. [chapter 4.3.1, fig 4]

On the other hand, the stakeholders did have some understanding in aspects such as delivery, sprint meetings, requirements and backlogs, compared to SAVSEs, with a scoring value average of around 5/10 and in the delivery case 8/10, although these areas too need more improvement on the SUVSE’s behalf. This performance can partially be justified due to their fields of work having similar concepts to software development in these areas. [chapter 4.3.1, fig 4]

5.2.2 Observations

Observations were made while working with SUVSEs through the research. Note – “they” and “them” in all observations refers to the SUVSE DEs.

The number of participants present at each meeting varied in each case, but a general set of observation could be drawn from the average behavior of the group throughout these meetings.

The observations did not show a significant deviation (at most ± 5%) from the questionnaire results and findings, in terms of proportions. Since the results from the observations are roughly similar to those of the questionnaires, we could look into the discussion of the questionnaire findings as well for comparison.

Most of them, roughly 75%, claimed to have participated in some way in software development projects, but only in terms of providing small amounts of feedback or DK. However, most of them had trouble communicating with the software developers, with roughly 90% on average either not understanding what was asked of them, or requiring further explanation on the developer’s part, for them to be able to understand the software requirements and provide the necessary DK asked. Roughly 60% were also found lacking in terminology knowledge from various software development process aspects. [chapter 4.3]

Due to all this, most meetings were significantly slowed down regarding progress in acquiring DK and software requirements, as a good portion of the time was dedicated in educating them in these aspects.
5.2.3 Interviews

In chapter 4.4, Software unaware company 03, and Software unaware company 04 mention the time of interaction was too short. It would be good time taken to working together with the SUVSE would be given a deeper understanding. Considering the time for the thesis the interaction with the VSE was made faster and shorter.

Companies 1,2 from Table 3,4 respectively give a statement that they have improved their knowledge which is a success. Just like how we form presentation depending of audience in the same way SUVSE must consider the clients in first place and have good communication since many of them fear development process like in table 3,4,5 companies 1,2,3 respectively

Table 3 company 2 gives a clear example for question 2, they don’t even have a basic knowledge over how a software is built, thus must be given special care and teach them basics first. Every company feels they have improved their knowledge before and after.

Table 5 company 3 explains there is a difference in terminologies for same ideas which is a critical point to be considered and a strong point mentioning weather what they think is possible or not, all the DE have enough DK but bringing DK to reality is a challenge where a proper communication will be key for success.

Table 3,4,5,6 and companies 1,2,3,4 agree the topics are good to know topics and are logical to understand, giving a sign of acceptance towards learning new knowledge related to another field.

5.2.4 RQ1

What topics are software unaware very small enterprise (SUVSE) aware of, in the software development process?

There are different topics to capture DK in agile methodologies, considering the agile methods being used in research which are XP and scrum, DE who must involve or should contribute are been chosen and been asked a questionnaire to get a basic understanding which topic SUVSE are familiar with. There is wide spread of answers which are heading towards right directions. Considering questionnaire response reference to chapter 4.3 tables number 4,5,6,7 show the topics which SUVSE’s are aware of in software development process.

5.2.5 RQ2

Which aspects of the software development process must be known by a software unaware domain expert of a very small enterprise (VSE), so that a software development team can capture the required domain knowledge (DK)?

Many participants did well in the questionnaire despite not having any background in software. Considering the questionnaire, observations and interview, many disagree to accept topics as mandatory for basic knowledge, such as software methodologies and software architecture. Looking at chapter 4.5 company 01 felt a few topics were too advance for them to learn. To be stable and to not scare companies, software teams should be grounded and not speak more advanced terminology in the beginning itself as said by company 01 in chapter 4.5

To summarize, from the questionnaire, observations and interviews, the following table provides topics to be prioritized which a SUVSE should know.

Table 8 Prioritized list of topics a SUVSE should focus on learning basics.
Discussion and conclusions

<table>
<thead>
<tr>
<th>Topics</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics, Agile, Software methodologies, Software Architecture, MUSCOW</td>
<td>High</td>
</tr>
<tr>
<td>Requirements, Terminology errors, Backlogs, Sprint Meetings, Myths</td>
<td>Medium</td>
</tr>
<tr>
<td>Delivery</td>
<td>Low</td>
</tr>
</tbody>
</table>

5.2.6 RQ3

*How is the software unaware very small enterprises (SUVSEs) understanding of the software development process affected by their involvement in an agile development methodology?*

The observations show noticeable obstacles were in the way of communicating, which changed from being scared to talking freely about the domain. They gradually turned to open little by little after every day and finding out the new issues/ideas one after another as mentioned by companies 01 and company 03 in chapter 4.5.

All the companies in the interviews, chapter 4.5 refer to learning new things and feeling excited at some point and few companies are so strong to face the new software team with full confidence that they know a good amount of terminology and topics on *how a software is being developed* and *what should their contribution towards it be*.

In the interview companies 01, 02, and 03 mention an overall statement that VSEs are scared because they feel software teams are too advanced and need a special care as well patience to give them time to explain, along with teaching them things which they do not know about and keeping ethics, such as not cheating the customer and treating customer problem as their (software team) own, while trying to implement elements which VSE are not aware of.

5.3 Conclusions

5.3.1 Summary

Based on the findings obtained, the following conclusions can be drawn:

VSEs will always present a challenge for software development teams due to their nature: their lack of time, resource and personnel will impact the flow and efficiency of the software development process. Their negative view of standards due to the VSEs faults, as well as a lack of coverage for the ISO standard will also make it more difficult, due to the fact that at times, developers will have to adapt to this. Constant and continuous communication with the VSE and its key stakeholders is paramount to ensure a proper flow of information, DK exchange and meaningful feedback. VSEs should also be encouraged more to utilize standards designed for them, namely the current ISO 29110. Should they still be in doubt regarding standardization, developers must be ready to adapt to their needs and fulfill the request in an agile manner, by delivering fast and continuously.

Domain experts are vital to any software project, due to the DK they possess. Communication and knowledge exchange with them are crucial for the fast and efficient development of the required software product or service. Throughout the
development process, the DEs must either be constantly available on-site, or at the very least be in constant touch with the development team.

Since many VSEs and DEs don’t possess the proper technical knowledge in software development processes, they must first be taught the notations, procedures, processes and terminologies, to ensure a bridge through which both DE and developers can exchange DK. This will also encourage them to be more active in the software development process and adopt a more meaningful role than just a passive mediator of knowledge. In other words, to enhance their role, a developed environment that assists them in sharing their knowledge and having knowledge shared in return must be put into place.

As for what should be taught to the VSEs and DEs, a recommendation would be everything that affects the exchange of information between both sides, understanding of the software architecture and services required, and how to prioritize the work. Elements such as Software development process, User Stories, Software Architecture, UML Diagrams, MUSCOW Table, Acceptance Tests, Backlogs, Requirements, as well as any terminologies used by the software development process chosen should be considered.

The development team must take the VSE and DE limitation factors into account when planning the development process, and the communication problems must be addressed and resolved immediately, at the start of the project.

Once a solid platform of communication is set, where all key stakeholder can share information, data, DK and feedback in a manner that is understandable to all the involved participants, the entire development process will be greatly improved and will reduce the chances of delays, misinterpretations and errors, while also speeding up the process, thus saving more time. In addition, at the end of the project the VSE will have gained a valuable body of knowledge in the sphere of software development, which will be of great help for future projects, and can thus now be considered a SAVSE.

5.3.2 Future Work

The scale of the case study does not offer an adequate or accurate representation of VSEs, but it does offer a general understanding of what the current situation roughly is and what is to be expected. There could be other factors that might impact the communication with VSEs and DEs, such as cultural differences from one VSE to the other, different standard and non-standard practices and preferences which vary from case to case, or scenarios where multiple VSEs are asked to cooperate on a common project, to name a few.

Given the reduced scale, with a small sample of VSEs and a small number of overall participants, as well as the lack of research of domain knowledge gathering and domain experts in the context of VSEs, especially software unaware ones, more work could be performed to have a better understanding of the particular challenges faced by development teams when working with VSEs, and how those challenges could be better overcome.

Looking back to the agile methodologies used in the research, which were limited by time and size, it is recommended to perform additional research using different agile methodologies which were not chosen, to capture domain knowledge in different scenarios and exploring alternative approaches.
Lastly, a conjecture could be made that, in the context of the software development process, a domain expert will be more valuable and contribute more if they are software aware and have a basic, but solid understanding of the development process and some of the terminologies. This conjecture is based on the results which compare SA and SU VSEs, which highlight an important difference between the two.
6 References

23. Scrum, Available at: http://www.mountaingoatsoftware.com/agile/scrum
25. Sánchez-gordón, Mary-luz; O'connor, Rory V (Sep 2016) 'Understanding the gap between software process practices and actual practice in very small companies', Software Quality Journal, 24(3), pp. 549-570 [Online]. Available at: https://search.proquest.com/docview/1800618015/fulltextPDF/632DCCDD96054A46PQ
27. Lepouras, George; Vassilakis, Costas; Halatsis, Constantin; Georgiadis, Panagiotsi (Sep 2007) ' Domain expert user development: the smartgov approach', Communications of the ACM, 50(9), pp. 79-83 [Online]. Available at: https://dl.acm.org/citation.cfm?doid=1284621.1284623
29. Sharon, Ryana; Rory, V.O'Connor (Sep 2013) 'Acquiring and sharing tacit knowledge in software development teams: An empirical study', Information and Software Technology, 55(6), pp. 1614-1624.
32. Mike Cohn (February 2009) User Stories Applied for Agile software development, 13th edn., RR Donnelley Crawfordsville: Addison-Wesley. (Cohn, February 2009)
7 Search terms

Agile, 4, 6, 17, 31, 35, 41, 44
Case Study, 24, 25, 26
DK, 46
Domain Expert, 9, 11
Domain Knowledge, 12
Interviews, 25, 28, 36
ISO, i, 13, 14, 17, 21, 22, 23, 24, 29, 42, 44, 45, 47
Observations, 26, 35, 40
Questionnaire, 27, 30, 31
Scrum, 17, 19, 20, 28, 29, 44, 45
Software aware, 14
Very Small Enterprise, 1, 13
XP, ii, 17, 18, 19, 28, 41, 44
8 Appendices

8.1 Appendix 1 (VSEs & CEOs)

Nordisk Industri Optimering - Martin Eriksson
CC Pack AB - Andreas Carlsson
Jonicom AB - Daniel Blixt
Gnutti Sweden AB - Mike Borger
Jens-Marten Company Sweden - Marten Albinsson

8.2 Appendix 2 (ISO)

ISO 29110 is comprised of a number of specifications:

ISO/IEC TR 29110-1 defines the terms common to the series. It introduces processes, lifecycles, and standardization concepts, in addition to the catalogue of ISO 29110 profiles. It also includes the characteristics and requirements of VSEs, and explains the rationale behind specific profiles, documents, standards and guides.

ISO/IEC TR 29110-2 defines the concepts for systems and software engineering profiles for VSEs. It establishes the logic behind the definition and application of profiles. For standardized profiles, it specifies all common elements between them, while for domain-specific profiles it provides a general guidance based on the standardized profiles.

ISO/IEC TR 29110-3 defines assessment guidelines, compliance requirements and certification schemes for process capability assessment (ISO 33xxx), conformity assessment (ISO 17xxx) and self-assessment for process improvement. It is addressed to stakeholders directly involved in the assessment process, who need guidance to ensure the requirements for performing an audit have been met.

ISO/IEC TR 29110-4 provides the specifications for all profiles in one profile group, based on subsets of appropriate standards

ISO/IEC TR 29110-5 provides a management and engineering guide for each profile

ISO/IEC TR 29110-6 provides management and engineering guides not tied to a specific group

Table 9 ISO 29110 assessment

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Terms &amp; Processes</td>
<td>Adequate</td>
</tr>
<tr>
<td>3</td>
<td>Guidelines, Requirements &amp; Specifications</td>
<td>Adequate, but potentially misleading</td>
</tr>
</tbody>
</table>
Appendices

8.3 Appendix 3 (questionnaire results)

**Question 1**

Were you involved in any software development process before?

- Yes: 77.8%
- No: 22.2%

18 responses

**SUVSE response to Question 1**

Were you involved in any software development process before?

100%

5 responses

**SAVSE response to Question 1**
Appendices

Question 2

How good was your communication with Development Teams?
18 responses

- 72.2%: We did not understand each other terminology and always had misunderstandings
- 11.1%: Sometimes good, sometimes misunderstandings
- 11.1%: We were well communicating without any problem
- 2%: I did not have any interaction with Development Team
- 2%: I was not involved in any software development process

SUVSE response to Question 2

---

How Well Was Your Communication With Development Team?
5 responses

- 80%: We did not understand each other terminology and always had misunderstandings
- 20%: Sometimes good, sometimes misunderstandings
- 2%: We were well communicating without any problem
- 2%: I did not have any interaction with Development Team
- 2%: I was not involved in any software development process

SAVSE response to Question 2
Question 3

Do you know about Request for proposal?
18 responses

SUVSE response to Question 3

Do you know about Request for proposal?
5 responses

SAVSE response to Question 3
Appendices

**Question 4**

How frequent you think client and development team should meet to show progress?

18 responses

**SUVSE response to Question 4**

How frequent you think client and development team should meet to show progress?

5 responses

**SAVSE response to Question 4**
Appendices

Question 5

Do you know of Agile software development?
18 responses

SUVSE response to Question 5

Do you know of Agile software development?
5 responses

SAVSE response to Question 5
Question 6

What do you know of software development methodologies?

18 responses

- 50% Nothing
- 33.3% Only a basic understanding
- 16.7% Well aware and informed of them

SUVSE response to Question 6

What do you know of software development methodologies?

5 responses

- 60% Nothing
- 40% Only a basic understanding
- Well aware and informed of them

SAVSE response to Question 6
Appendices

**Question 7**

Do you consider that having a good understanding of the basics of what software development methodologies ...r the software development process?

- 18 responses

![Pie Chart](image1)

- No, not really: 50%
- Maybe: 16.7%
- Yes, definitely: 33.3%

**SUVSE response to Question 7**

Do you consider that having a good understanding of the basics of what software development methodologies ...r the software development process?

- 5 responses

![Pie Chart](image2)

- No, not really: 80%
- Maybe: 20%

**SAVSE response to Question 7**
Question 8

Do you know what UML diagrams are, and what they are for, regarding the software development process?

18 responses

SUVSE response to Question 8

Do you know what UML diagrams are, and what they are for, regarding the software development process?

5 responses

SAVSE response to Question 8
Appendices

Question 9

Who do you think should decide the aspect of the User Interface?
18 responses

SUVSE response to Question 9

Who do you think should decide the aspect of the User Interface?
5 responses

SAVSE response to Question 9
Appendices

**Question 10**

Do you know the principle of "fail fast, fail often", and why it is relevant to the software development process?
18 responses

SUVSE response to Question 10

---

Do you know the principle of "fail fast, fail often", and why it is relevant to the software development process?
5 responses

SAVSE response to Question 10
**Question 11**

Do you know what the MUSCOW table is, and why it is important to the software development process?

18 responses

- 88.9%: No, I don't know what it is
- 11.1%: Yes, I have heard of it, but don't know its relevance precisely
- Yes, I do

SUVSE response to Question 11

Do you know what the MUSCOW table is, and why it is important to the software development process?

5 responses

- 40%: No, I don't know what it is
- 40%: Yes, I have heard of it, but don't know its relevance precisely
- 20%: Yes, I do

SAVSE response to Question 11
Appendices

Question 12

If you know what the MUSCOW table is, who do you think should be involved in writing it?

18 responses

SUVSE response to Question 12

If you know what the MUSCOW table is, who do you think should be involved in writing it?

5 responses

SAVSE response to Question 12
Appendices

Question 13

Would you compromise if the software you asked for shows up different than expected, because of terminology... between you and development team?

18 responses

SUVSE response to Question 13

Would you compromise if the software you asked for shows up different than expected, because of terminology... between you and development team?

5 responses

SAVSE response to Question 13
Question 14

Do you know what User Stories are, and why they are relevant to the software development process?
18 responses

SUVSE response to Question 14

Do you know what User Stories are, and why they are relevant to the software development process?
5 responses

SAVSE response to Question 14
Appendices

**Question 15**

*If you know what User Stories are, who do you think should write them?*

18 responses

- Developers: 44.4%
- Clients: 22.2%
- Both developers and clients: 27.8%
- I don’t really know

**SUVSE response to Question 15**

*If you know what User Stories are, who do you think should write them?*

5 responses

- Developers: 60%
- Clients: 40%
- Both developers and clients
- I don’t really know

**SAVSE response to Question 15**
Appendices

Question 16

If you know what User Stories are, how much time do you think should be spent on writing them in the beginning...sider a small-to-medium sized project

16 responses

SUVSE response to Question 16

If you know what User Stories are, how much time do you think should be spent on writing them in the beginning...sider a small-to-medium sized project

5 responses

SAVSE response to Question 16
**Appendices**

**Question 17**

If you know what User Stories are, how often do you think they should be updated, whenever changes occur or more functionalities are added?

18 responses

![Pie chart showing response distribution]

**SUVSE response to Question 17**

If you know what User Stories are, how often do you think they should be updated, whenever changes occur or more functionalities are added?

5 responses

![Pie chart showing response distribution]

**SAVSE response to Question 17**
Appendices

Question 18

How do you think software development teams or companies will react when told to add more functionalities a...ent process, beyond the initial plan?

18 responses

SUVSE response to Question 18

How do you think software development teams or companies will react when told to add more functionalities a...ent process, beyond the initial plan?

5 responses

SAVSE response to Question 18
Appendices

Question 19

Who do you think has control over the software being built and take final decisions at crucial situations?
18 responses

![Pie Chart](image1)

SUVSE response to Question 19

Who do you think has control over the software being built and take final decisions at crucial situations?
5 responses

![Pie Chart](image2)

SAVSE response to Question 19
Appendices

Question 20

Do you think a Development Team should tell the Client briefly in advance about software methodologies?
18 responses

SUVSE response to Question 20

Do you think a Development Team should tell the Client briefly in advance about software methodologies?
5 responses

SAVSE response to Question 20
Appendices

**Question 21**

Do you know the term "backlogs"?
18 responses

---

**SUVSE response to Question 21**

---

Do you know the term "backlogs"?
5 responses

---

**SAVSE response to Question 21**
Appendices

Question 22

Who do you think should keep track of backlogs?
18 responses

SUVSE response to Question 22

Who do you think should keep track of backlogs?
5 responses

SAVSE response to Question 22
Appendices

Question 23

Do you think having backlogs is good or with no backlogs is good?
18 responses

SUVSE response to Question 23

Do you think having backlogs is good or with no backlogs is good?
5 responses

SAVSE response to Question 23
Question 24

Do you know the term called Sprint meetings?
18 responses

SUVSE response to Question 24

Did you hear term called Sprint meetings?
5 responses

SAVSE response to Question 24
Question 25

Do you think Sprint meetings are useful?
18 responses

SUVSE response to Question 25

Do you think Sprint meetings are useful?
5 responses

SAVSE response to Question 25
**Appendices**

**Question 26**

Every field has its terminology. If you were developing a software of your field (say Automobile), do you think you...daily usage words from your company?

18 responses

SUVSE response to Question 26

Every field has its terminology. If you were developing a software of your field (say Automobile). Do you think you...daily usage words from your company?

5 responses

SAVSE response to Question 26
Question 27

Do you think you should take inspiration regards style/design /terminology/concepts from competitors or from the same field of expertise?

18 responses

SUVSE response to Question 27

Do you think you should take inspiration regards style/design /terminology/concepts from competitors or from the same field of expertise?

5 responses

SAVSE response to Question 27
Appendices

Question 28

Do you know the term software architecture?
18 responses

- 44.4%: Yes, I know what it means
- 33.3%: No, I don’t know anything about it
- 22.2%: I heard about it, but I don’t know more about it

SUVSE response to Question 28

Do you know the term software architecture?
5 responses

- 100%: Yes, I know what it means

SAVSE response to Question 28
Question 29

Do you think you should know about software architecture?
18 responses

SUVSE response to Question 29

Do you think you should know about software architecture?
5 responses

SAVSE response to Question 29
Appendices

**Question 30**

Do you think it would be useful if the Development Team gives you documented details of the software after finishing the project?

18 responses

- Yes: 77.8%
- No: 16.7%
- Maybe: 5.5%

SUVSE response to Question 30

Do you think it would be useful if the Development Team gives you documented details of the software after finishing the project?

5 responses

- Yes: 100%

SAVSE response to Question 30