Incorporating Pragmatic Principles in Daily Software Development

Master Thesis

Group:
cs-22-sd-10-02

Supervisor:
Ivan Aaen

February 10, 2023

Department of Computer Science
Aalborg University
http://cs.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Incorporating Pragmatic Principles in Daily Software Development

Theme:
Master Thesis

Project Period:
Autumn 2022

Group:
cs-22-sd-10-02

Participants:
Theresa Walker Junker

Supervisor:
Ivan Aaen

Pages:
56

Date of Completion:
February 10, 2023

Number of Copies:
1

Abstract:

This report documents an attempt to include the values of Essence throughout project development to aid a software development process. To accomplish this, I follow the design theory for visual inquiry tools and characteristics of daily stand-up meetings in order to create a process that promotes joint inquiry and keeps a project on track.
Essence is a methodology in the pragmatic paradigm. This methodology does not include a means of reviewing work in the daily development of software projects, and therefore I propose a framework a process that covers how a team can be coordinated during their sprints in order to ensure project stability and efficiency. This coordination occurs by team members helping each other identify and resolve problems early in development in order to create software that is relevant and useful to its end users.
This is achieved by creating two types of meeting, one to raise problems, and the second to handle them. These two meetings are designed so that they align with the Essence methodology. By encouraging the handling of problems in daily development in a directed, constructive manner, I aim to help a team understand their project and its course better and overall aid in the creation of software products.

Contents

1 Introduction **1**

1.1 Motivation . . . 1
1.2 Overview . . . 2

2 Problem Analysis **3**

2.1 Daily Standup Meetings . . . 3
2.2 Visual Inquiry Tools . . . 6
2.3 Essence . . . 7
2.3.1 Core Concerns and Values . . . 8
2.3.2 The Configuration Table . . . 10
2.4 Problem Statement . . . 12
2.4.1 Method . . . 12
2.4.2 Evaluating a DSM as a VIT . . . 13

3 Daily Stand-Up in Essence **14**

3.1 A Stand-Up Meeting in Essence . . . 14
3.2 An Essence Project DSM - Case Example . . . 15
3.3 Initial Prototype . . . 17
3.3.1 Evaluation of the VIT for a DSM . . . 18
3.3.2 Findings . . . 20

4 The Stand-Up Meeting **21**

4.1 Inclusion of Essence Methodology . . . 21
4.2 Contents of a Meeting . . . 21
4.2.1 Categorising Problems . . . 22
4.2.2 Action Points . . . 23
4.3 Process for a DSM . . . 24
4.4 DSM as a VIT . . . 25
4.4.1 Evaluation as a VIT . . . 26
4.4.2 Evaluation as a DSM . . . 27
4.5 Findings . . . 28

5 Focused Meetings **29**

5.1 A Focus on Problems . . . 29
5.1.1 Question-Based Discussion . . . 30
5.1.2 Methods of Evaluation . . . 30
5.1.3 Sprint Pivot or Sprint Persevere . . . 31
5.1.4 The Process for a Problem-Focused DSM . . . 31

5.2 A Focus on Solutions . . . 34
5.2.1 Evaluation of a Solution . . . 34
5.2.2 The Process for a Solution-Focused DSM . . . 36
5.3 Problem and Solution-Focus as a VIT . . . 38
5.4 Findings for Problem and Solution-Focused Meetings . . . 41
5.4.1 The Problem and Solution-Focus Pairing . . . 41

6 Activities during a Sprint **43**

6.1 A Sprint in an Essence Project . . . 43
6.1.1 The Overall Process . . . 45
6.1.2 The Lung Clinic Case . . . 45
6.1.3 Findings . . . 47

7 Discussion **50**

7.1 Formalisation of Problems . . . 50
7.2 Pragmatism in Essence . . . 51
7.2.1 An Accompaniment to a Review . . . 53
7.3 Limitations . . . 53

8 Conclusion **55**

8.1 Future Work . . . 56

Bibliography **57**

Chapter 1

Introduction

Essence was created in order to shift the focus for development away from software production and more towards software innovation [1, p.49]. A part of development is the day-to-day work that leads to the production of a software product. Essence is centred around how to ensure that a team makes a relevant product that creates value for their customer, where value is a solution to the problem they are experiencing. What it does not consist of, however, are practices that ensure that these values are incorporated and iterated through the daily progress and development of a project. This is therefore the focus of the rest of the report.

1.1 Motivation

Development teams typically conduct meetings within sprints to help keep them on track and know how much they are progressing. A common way of doing this is by conducting a daily stand-up meeting. The purpose of such a meeting is establishing teamwork and collaboration between the members of the team [2]. This purpose aligns with the definition of the term inquiry, which is to make an uncertain situation certain, where we reflect on our thought processes in order to understand and solve a problem [3]. When this occurs as a team effort, we call it joint inquiry [4]. It requires that the entire team discuss the problem and come to an agreement on how it should be solved. When working on software development projects, it is generally the case that there are multiple people involved, as the problems for such projects can be complex and generally require the experience and expertise of different people.

Essence is a methodology that helps software teams organise the complexity of solving a problem with regard to creating a software product. However, when problems arise within a development cycle, Essence does not have a means of intercepting them, nor a means of treating them when they arise in the daily work of a development team. Many decisions are made before development begins that seem relevant and meaningful at the time, but when put into practice can host many problems that were unforeseen.

As daily stand-up meetings are generally used as a tool for making members of a team aware of progress during sprints, it could also be used as a way of identifying and handling problems during the development of a software product. In this project, I explore the possibility of creating a tool that allows software development teams to identify and handle problems as they occur within a sprint. In order to explore this, I pose the following initial questions:

1. What processes exist for encouraging joint inquiry within a development team?

2. Does Essence support a means for encouraging inquiry on a daily basis?

1.2 Overview

Chapter 2 is an analysis of the problem domain for this project. I present two articles on daily stand-up meetings, which outlines how development teams actually practice this type of meeting and what they are for [5][2]. I also discuss and present the Essence methodology and what it entails with regard to reviewing a software development project and its direction. The last part of the analysis is in regard to the design theory for visual inquiry tools and how incorporating this type of tool into a daily meeting could be beneficial for a software team. The reason for looking at these three aspects is to be able to see how I can promote joint inquiry in Essence during daily development.

Chapter 3 covers the project's initial proposal (proof of concept) for designing a meeting that can improve the levels of communication within a software development team. Essence does not currently have a meeting that is tailored to it, and it is in this chapter, that I present a first prototype for what a meeting could entail whilst respecting the design theory for VITs and the DSM guidelines.

Chapter 4 is building upon the initial prototype and the findings from chapter 3 and is a more robust and pragmatic proposal for how a daily stand-up meeting in Essence could be. Here I present the overall goal of the meeting, which is to identify problems and decide how the team will deal with them.

Chapter 5 is an introduction of a meeting type not quite fit as a daily standup meeting, but are used in conjunction with a daily meeting. This type of meeting focuses on either problems or solutions that are raised at a daily stand-up meeting. It is in this meeting that problems are explored in detail with regard to the rest of the project as well as dealing with the problem itself.

Chapter 6 is where I describe the connection between the daily stand-up and the focused meetings, defined in chapters 4 and 5 respectively. It also discusses how the project's proposal of how they could be used within a sprint that uses the Essence methodology.

Chapter 7 is a discussion converging the findings discovered through designing the prototypes for the two types of meetings and how this can affect the review process and the daily development of software.

Chapter 8 is the conclusion to my project, where I present my findings and how the ideas I present in this report could be utilised and further refined to create a positive way of discussing and dealing with problems that occur during sprints.

Chapter 2

Problem Analysis

To investigate whether a daily-stand up meeting is an appropriate vessel for identifying and handling problems, I analyse related work on daily-stand up meetings in practice. In addition to this, I also explain and analyse aspects of the Essence methodology as well as an introduction to visual inquiry tools and how they are used to promote inquiry.

2.1 Daily Standup Meetings

The paper "Daily Stand-Up Meetings Start Breaking the Rules", reflects on software teams that incorporate agile practices in their workplace [2]. The authors behind this paper specialise in agile practices and have experience with this domain that spans over a decade. Stray et al. state that conducting daily stand-up meetings, so that the whole team benefits, can be challenging. Their study includes four different companies, interviews with 60 project members with varying degrees of seniority in 15 different teams. The study also includes observations on 102 daily stand-up meetings. Through the interviews and observations of meetings, they conclude that it is uncommon for a software team to benefit fully from a daily stand-up, with the traditional three questions of: What did I do yesterday?, What will I do today? and Are there any issues?. Instead, they discovered that software teams found that elaborating on problem issues and discussing solutions was the most positive aspect of their daily meetings.

Daily stand-ups typically have a facilitator of the meetings, someone who controls the flow of conversation. Stray et al. observe that these meetings often became status reports of actual discussions and that not all information is relevant due to the diversity of a team. They discovered that facilitators who were good at getting the team to discuss within a group were more involved in the day-to-day work and therefore did not need a status update. However, if a facilitator has a personal interest in a specific task, it is often the case that these members are given more attention, which negatively affects other team members. Stray et al. feel that sharing leadership and changing roles within a team could benefit meetings so that communication happens across the team rather than one-on-one conversations with no input from other team members.

Another consideration they make as to what benefits a daily meeting, is the time of day that it is held. Many team members expressed a dislike of having their day be interrupted due to having to detach themselves from what they are working on to have a meeting with the rest of the team, as it takes time to get the same workflow when the meeting ends. The conclusion they make for time of day, based on their interviews and observations, was that starting before team members usually would have lunch led to a more positive experience, as there were no abrupt interruptions.

Changing the time of the meetings was greeted with positive feedback from the teams.

To conclude their study, they include a set of initial recommendations for daily stand-up usage. These recommendations are omitting status reports, circulating a facilitator role, finding the least disruptive time, and finding a frequency for meetings that offers value.

In another article about the usage of daily stand-up meetings, which I also refer to as a DSM, Stray et al. present a way to increase satisfaction when conducting daily stand-up meetings. The article "The Daily Stand-Up Meeting: A Grounded Theory Study" is based on data from 79 observations of meetings of eight teams divided between three companies across all roles [5]. Their work resulted in a set of guidelines to help teams organise and improve future meetings. They present a list of characteristics these types of meetings have, showing what an optimal stand-up meeting should include based on their research. This is depicted in Figure 2.1 [5].

Characteristic	Regular Stand-Up Meeting (based on empirical data)
Purpose	Obtain a shared understanding of the current activities of other team members
Potential benefits	Improve communication, knowledge sharing and team orientation. Identify, avoid and solve problems
Potential pitfalls	Reporting status that is not relevant to all team members. Wasting time because team members are disengaged. Interruption of workflow
DSM questions	The team members explain the following: 1. What will I do today to help our team accomplish the iteration goal? 2. What problems do I know of that may prevent progress?
Format	Meeting participants stand up in a semicircle in front of a physical or electronic board
Turn-taking	Round-robin
Frequency	Must be held regularly, but the frequency may differ depending on iteration phase, need and level of informal communication
Time of day	Find the least disruptive time for the team
Duration	As brief as possible (think "huddle" instead of "meeting") but maximum 15 minutes

Figure 2.1: Guidelines for generic standup meeting [5]

To begin my analysis of the characteristics, I start by looking at the purpose of a daily stand-up meeting, according to Stray et al.

Purpose The stated purpose is to ensure that the development team gains a shared understanding of the current state of the project and what their team members are doing. In other words, ensure that the team communicates what is being done, and how they are going to do it without being exclusively a status meeting.

Potential benefits The benefits that they discover are that daily stand-ups improve communication, not just in the meeting, but also in after meeting conversations. The improved communication helps with sharing information and discussion of problems that were consequently able to be solved or be avoided in the first place. This is a great benefit for a team, as finding problems sooner rather than later helps a team lessen waste of resources and furthers productivity.

Potential pitfalls A problem that was perceived in their study, was the fact that a daily stand-up meeting can often be reduced to a status meeting, which lowered morale of participants. The reason for it lowering morale is due to some team members feeling uncomfortable about the

amount of progress they had made and other members feeling that the meeting is a waste of time because they are not learning anything new about the project.

DSM questions The three questions of the Scrum and XP guidelines were used by many teams, however it was the second and third that were most useful, as the first question led to status reporting, which was negatively looked upon. The first question was also very time-consuming, meaning that not only was the status reporting negative, it also took up valuable time in what should be a brief meeting. Stray et al. state that if a status is necessary it can be given with other means than at a daily stand-up.

Format It was discovered that when meetings were held standing, the duration was shorter than when sitting down, even though both meetings did produce decisions of similar quality. They also discovered that it had a positive impact on the meetings when the team use a board or other means of visualising what is being discussed.

Turn-taking In their observations, they saw that when a facilitator chose the order in which members could speak, it quickly became a status report, or it became a discussion between the facilitator and the member whose turn it was to speak. Therefore, Stray et al. recommend that development teams use an approach that rotates the facilitator role between team members.

Frequency Even though the name of the type of meeting is ”daily stand-up”, they discover that it is not crucial for a meeting to be held every day. However, they also mention that in some phases of the current sprint, it could be beneficial to conduct a meeting more often. This finding is the reason for the suggestion of a change in name for the daily stand-up to instead be called a ”regular stand-up meeting”.

Time of day Not every team found it beneficial to have a meeting first thing in the morning, and as every member is working on different parts of the project, they are interrupted in their workflow. This is why they recommend conducting the meeting before lunch in order to not disrupt workflow and also allow for any discussion that are not completed at the meeting can also be discussed over lunch, if the members feel it is necessary. However, their overall recommendation here is that a development team should find the time that suits them and leads to the least disruption.

Duration Stray et al. argue that 15 minutes is the maximum duration for a daily stand-up. It is also important to end the meeting on time, as when the time limit is reached, the participants do not pay as much attention.

These characteristics are good practice for a DSM based on many interviews and observations, and therefore they are what Stray et al. recommend software teams follow when conducting such meetings. Based on this analysis of daily stand-ups, and what they entail, I discover that the theory of daily stand-up meetings and daily stand-up meetings in practice are not always similar. The constructs of their theory that give a positive output are discussing and solving problems and team information sharing. Other positive aspects were, using a board or other visualisation tool to help communication between team members, as well as identifying and solving problems in a daily stand-up meeting, since participants perceived the time used here as useful. The observed stand-up meetings were more focused on solving problems and delegating tasks than what the actual guidelines suggest. Even though, the theory for a daily Scrum or daily XP, do not explicitly mention visualising tasks, many teams do this anyway in order to organise the points that they want to accentuate and remember during discussions. This is why I look into how aspects of a DSM can be visualised via use of a visual inquiry tool.

2.2 Visual Inquiry Tools

The way in which teams use inquiry in order to complete their projects differs from team to team, however, there are certain guidelines and theories that have been established that can aid project work in order to solve problems. Inquiry is not limited to being a part of a daily stand-up meeting, and in fact multiple tools that promote inquiry have their basis in visual queues. Bobek et al. argue that "visual explanations map thought more directly than words and provide checks for completeness and coherence as well as a platform for inference" [6]. This argument is interesting with regard to investigating how the inquiry that is present at meetings like a daily-stand up in Scrum can be visualised in order to enhance the development team's understanding of what is being discussed and determined.

In software development, visualising tasks or content of sprints is fairly common. In Stray et al. they mention that the teams during meetings find it beneficial for them to work off of visual cues instead of pure discussion [5]. The paper "A Design Theory for Visual Inquiry Tools" by Avdiji et al. proposes a design theory for how teams can cope with managerial problems using a visual aid. Daily stand-up meetings are used to address problems being experienced by teams, and therefore I see it being relevant to look at how VITs can help organise and structure a DSM. A visual inquiry tool is not limited to being used in a specific scenario, but they can be used whenever there is a need for communicating with members of a team in order to gain a shared understanding and have a common language for discussions [4].

Avdiji et al. analyse three successful VIT projects in order to create their design theory. One of the projects they use is the business model canvas, seen in Figure 2.2. This VIT is built on nine building blocks, where each block reflects an aspect of a business idea. Their aim is to create a shared understanding of the business model in a simple and intuitively understandable way [7]. Each building block has an accompanying description and examples of what they should include. From the way in which this VIT is built and the two other examples in their paper, they construe that there are three main principles that are prevalent, of which each one is composed of three subprinciples for designing. I use the word component in order to describe what the principle should cover. An overview of the principles is shown in Table 2.1, which I relay in more detail now.

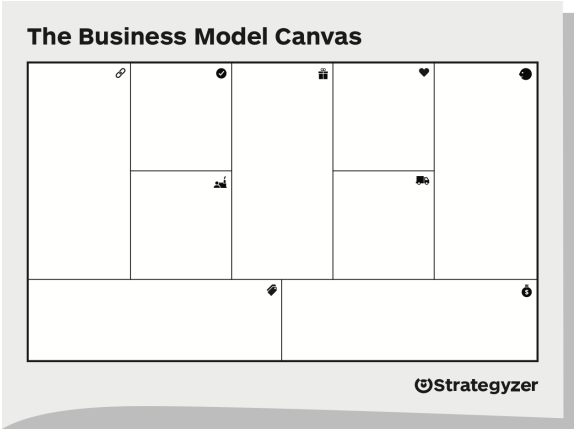

Figure 2.2: A Visual Inquiry Tool - The Business Model Canvas [8]

A conceptual model is a main principle. It is supposed to be able to adequately set the scope of the purpose for inquiry, in other words frame what it is we want to achieve. Each aspect should be relevant without having any overlap. The first principle within this is *frame*, where the overall model should establish all relevant components which are used to solve the problem. The second

Conceptual Model	Shared Visualisation	Directions for Use
Frame	Functionality	Ideation
Rigour and relevance	Arrangement	Prototyping
Parsimony	Facilitation	Presentation

Table 2.1: Main principles with their subprinciples from the design theory [4]

principle is *rigour and relevance*, where the model has a sound reason for being included and whether it actually reflects what the target group of the VIT work with. The third principle for the conceptual model is *parsimony*, where it is important to try to have the fewest components possible while still being relevant. They also state that if there are certain subcomponents that are important, additional tools can be used [4].

Shared Visualisation is a main principle supposed to address the need for assisting proper communication between team members. The aim is to be able to represent the task visually, by structuring the tasks in such a way that is intuitive and removes complexities that require a deeper understanding of the task. The first principle is *functionality*, where the components of the model are represented as empty problem spaces to promote the exchanging of ideas. The second principle is *arrangement*, where the components of the model should be arranged based on the relationships they have. The third principle here is facilitation, where it is stated that a common language is important to use. This can be in the form of any type of visual communication.

The final main principle is Directions for Use, where it is important to include techniques that allow joint inquiry and shared communication. It should help stimulate discussions and ideas, as well as allowing the presentation of ideas. The first principle is ideation, where the VIT should encourage the users to exchange ideas for the problem at hand. The second principle is *prototyping*, which should aid the team in creating and choosing different options regarding the task they are facing. The final principle here is *presentation*, which suggests that a means of being able to discuss the solution for the task using markers should be present.

This design theory helps aid joint inquiry and helps create common understandings among members of a team. I see similarities in what the daily stand-up meetings guidelines say are good for an overall meeting environment and what a VIT can give us. These similarities are:

- Sharing information amongst team members.
- Visualising the problem in some form.
- Problem-solving is the main benefit of the task.

Using the theory for a visual inquiry tool, helps us fulfil these criteria, as that is the ultimate goal of VITs, namely having a shared visualisation for managing and solving problems experienced within a group.

The next section is an analysis of Essence and the various aspects of the methodology that help a team discuss complex problems. This is not only an explanation of what activities the methodology includes, but also the values that it is built on.

2.3 Essence

In terms of reflective practices, Essence employs several. First, we have the different roles that are supposed to allow for seeing different perspectives. Second, the values of Essence state that it is

fundamental to using the Essence methodology, and thirdly, there are concrete aspects to Essence that help a development team reflect on their actions.

The image, shown in Figure 2.3, depicts the Essence methodology from the beginning of a project to the end.

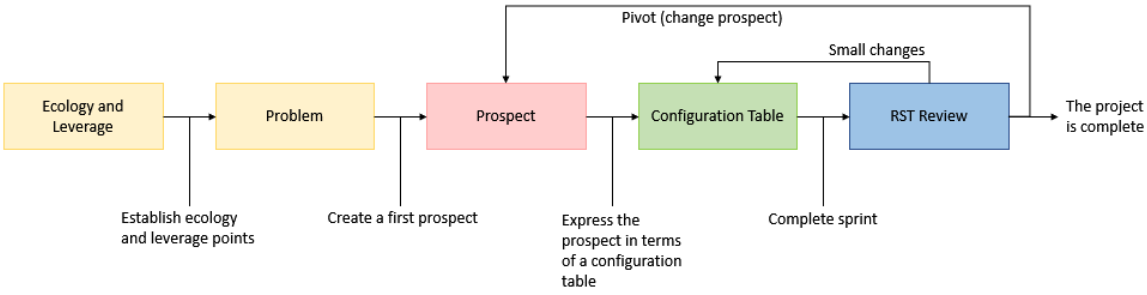

Figure 2.3: The Essence methodology

Ecology and Leverage Before a problem can be defined, a team must explore their options with regard to different ecology objects, these being services, artefacts, repositories, and people. In other words, external aspects a system would interact with. A team must also explore their options with regard to leverage points that relate to technologies, artefacts, repositories or people that can help reduce tasks for the developers and improve the system.

Problem The overall problem is defined by what options of ecology objects and leverage points the development team sees as most viable. This is because they have helped us identify our current understanding of the challenge and therefore are able to define an initial problem.

Prospect Prospects are discovered by setting up dichotomies in order to gain a new perspective and think in other terms. The team then ultimately choose one of these to continue with.

Configuration table The Configuration Table is a means of keeping track of all aspects of the project. When a team looks at the configuration table they should have a clear image of the state of the project, what they are developing, why this specific design makes sense, and how they can develop the design.

RST Review The review is for reflection on the team's latest sprint, where each level of the table is discussed, ultimately leading to an update of the configuration table. This decision is based on whether what they are working on is still relevant, and whether the team wish to persevere with their current trajectory or change course and pivot, changing the overall focus of the project.

Based on the Essence methodology, the only aspects that need to be visited again in one project, are the configuration table, and the RST Review. It could be argued that prospects can be revisited if a pivot is seen as necessary when reflecting on the course of the project, but otherwise, it is the final two that are most prevalent in the lifetime of a project. The configuration table and the RST Review is where a development team focus their reflection practices, whatever they may be.

2.3.1 Core Concerns and Values

Essence's values are based on an analysis of the values of the agile manifesto combined with principles from the pragmatic philosophy [1].

Value	View	Role
Reflection	Situation	Child
Transaction	Contribution	Responder
Reasoning	Solution	Challenger
Appreciation	Valuation	Anchor

Figure 2.4: Each corresponding value, core concern, and role

Aaen allocates the four values to the four Essence core concerns, as depicted in Figure 2.5 and Figure 2.4. The values are "*expressions of principles or standards of behavior; our foundation for deciding what matters most*" [1, p. xx]. They are present in the Essence methodology in order to influence how we act, how we reflect, and how we evaluate. Each value has been developed so that they are tied to one of the four core concerns and are used to complement each other and encapsulate an entire project. If one value leads to new ideas, this can affect another value and therefore affects more than one part of the project.

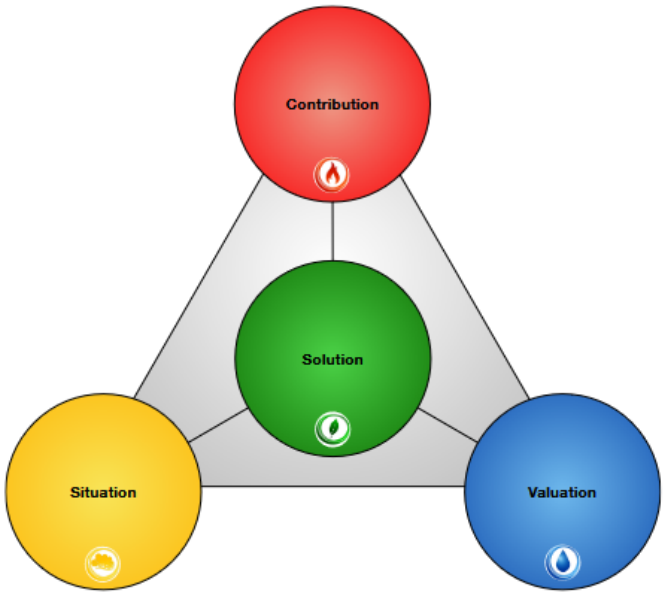

Figure 2.5: Core concerns in Essence

As each value has a corresponding core concern and role, it gives us a solid foundation, where we can incorporate them in an initial design for a stand-up meeting. The following values are where it is important to see how these differ from traditional agile values, and how, if applied to a daily activity, would be upheld.

Reflection This term is used to depict that we value reflection in a project instead of only looking at requirements. When working on a project, we gain a deeper understanding than what is stated in a requirements list. As innovation involves collaboration and learning along the

way, it may require that team members look at alternatives to go beyond what a customer expects. It is important that the team ensures that what they are creating is well understood and well-defined in order to give a customer a good product. Reflection helps us validate that the elements for the project represent a sound understanding of the problem to be solved [9, p.118].

Transaction It is used to show that we do not just have a focus on a solution. It is making sure that a team questions whether the internal objects, such as architectural components or algorithms could be utilised better, and by asking questions, our internal objects could have more potential than originally envisioned. It is meant to reject the conventional struggle for finishing work and points to viewing any step as offering a possibility for better solutions. By transaction, it refers to interactions between the inquirer and our means. These are the elements that are part of the problem at hand and what we have to be able to solve it. We want to validate that the technologies are well-chosen and that the ways we build our product using these are well-designed [9, p.118].

Reasoning It is used as a value to denote the fact that this is more important than having an explicit assignment. It is important to discuss our end-in-view with our team members due to the fact that having a list of requirements is difficult in seeing where change can occur, whereas having an overall goal, where a team can discuss why this is a solution, and where there are limitations instead of a list of assignments. To allow for innovation we must focus on defining the scope and goals of the project as it unfolds, as once assignments are given, no more innovation happens. A requirements based plan should be replaced with a reasoned project vision [9, p.118].

Appreciation This is for helping a team make sure that they are getting the best they can out of the other three values, where a team can create a list of criteria for acceptance of certain solutions and determining whether what they have created in a good contribution. It is supposed to reject the conventional focus on structuring work according to universal practices and instead focus on specifics of the moment that aid in evaluation and maturation of ideas and visions. The process must provide support for observation and validation and help promote adapting to changes [9, p.118].

If it is the case that these values are accurately represented within the four core concerns, then a review of the entire configuration table makes sure that a project encapsulates the values of Essence.

A keystone in Essence is joint inquiry and working as a team in order to effectively create a software product. And part of this requires the team to reflect on what they have created. Donald Schön states that there are two separate means of reflection [10]. Reflection-in-action as established by Donald Schön is supposed to address reflection while an event is occurring. Its counterpart, however, reflection-on-action is supposed to address reflection after events have occurred, and we are looking back at what has happened, in order to improve future practice. Some examples of this are various agile practices, like the daily stand-up or sprint retrospectives.

2.3.2 The Configuration Table

The configuration table consists of four columns, one for each value, that each depict a core concern for a project [1]. The first column represents the situation, meant to represent the value *reflection*. The second column represents the solution for the project. These elements are supposed to reflect the value *transaction*. The third column is for a means of describing the project in terms of our contribution; why it is a solution to a problem. This represents the value *reasoning*. The final

	Situation	Contribution	Solution	Valuation
Rationale	(P) Problem	(L) Leverage	(R) Resolution Prospect Warrant Backing	(CR) Criteria for rationale
Strategy	(O) Outer Environment	(I) Inner Environment	(Q) Qualifier Reservation Rebuttal	(CI) Criteria for strategy
Tactics	(M) Manifestations	(C) Capabilities	(V) Value	(CV) Criteria for tactics

Figure 2.6: Configuration Table with corresponding level and core concerns

column in the table is called valuation, which is used to determine the relevance and the arguments for why these are solutions to our overall problem. This relates to the value *appreciation*.

The categories presented in Table 2.2, are the ways in which a project can be defined. They each relate to a core concern in the configuration table as seen in Figure 2.6. Once each of these categories has been reflected on and completed, it becomes clearer for a software team, the direction their project is taking, and what tasks have to be completed in order to realise their goal.

Category	Description
Problem	What the project wants to solve
Outer Environment	The way the system interfaces with the problem domain
Manifestations	Scenarios that represent how the problem can be solved
Leverage	Technologies that we can use to solve the problem
Inner Environment	Components that the team build to solve the problem
Capabilities	Features that help support the scenarios
Prospect	A possible solution to the problem
Warrant	The reason why this problem should be solved
Backing	The reason why this is a good solution
Reservation	A limitation in the solution
Rebuttal	The reason the solution is acceptable
Value	The contribution the solution gives to the problem

Table 2.2: Descriptions of the categories in the CT [9, p. 109]

These aspects of Essence are a visual representation of the project, that allow a software team an overview of what they are currently working towards. Any changes made to the configuration of a project are also documented within the configuration table, making the direction of a project, and what is being worked on, transparent.

2.4 Problem Statement

The goal of this project is to explore opportunities for an activity like a daily stand-up meeting for Essence. This chapter reflects the initial questions from chapter 1, which I discuss below.

1. What processes exist for encouraging joint inquiry within a development team?
2. Does Essence support a means for encouraging inquiry on a daily basis?

In order to answer these I looked at current methods of encouraging teams to communicate on a daily basis. Essence also has the RST Review, which is similar to Scrum's sprint review. Looking at how visual inquiry tools can also be used in order to encourage joint inquiry gives a different perspective of how teams can communicate regarding problems, it does not necessarily require a meeting, like a DSM, but it can give the task at hand a structure that makes it easier and more manageable to discuss.

Additionally, I have discovered that there is currently no explicit means of promoting inquiry on a daily basis other than the team are working towards the end of their sprint, so that they can complete an RST Review. Due to the lack of a structure during development cycles that takes Essence into consideration, it is relevant to explore how daily stand-up meetings, visual inquiry tools and Essence could be combined in order to help software development teams discover problems within a sprint and keep the project on track.

The Essence methodology is firmly placed in the pragmatic paradigm, and already consists of one main reflective practice between iterations. It has been shown that more frequent, informal meetings help to improve productivity in a development team [2]. This improvement in productivity can be attributed to team members having a better understanding of the direction of the project on a daily basis, as well as addressing problems as they crop up within an iteration. Based on the findings from the initial questions, I have the following problem statement:

> How can joint inquiry become a useful part of the daily activities in Essence?

2.4.1 Method

In this report, I present a means of encouraging inquiry using Essence and its values during development between sprints. The model is to combine the various theories from the problem analysis and incorporate aspects of Essence into a Daily Stand-Up Meeting. I use the design theory for visual inquiry tools (section 2.2) because of the similarity of what a DSM aims to achieve (section 2.1), and what a VIT provides, such as a shared visualisation and defining the problem at hand. In order to see what my contribution could be, I begin by exploring the aspects of Essence that lend itself to being reflected upon between iterations, namely how the configuration table could be handled between sprints and how the RST Review helps a team reflect on their choices.

For this project, I create prototypes using the design theory for Visual Inquiry Tools, and discuss what the different prototypes would mean for a daily stand-up meeting in Essence and what the advantages and disadvantages these would bring to a development team.

The creation of the prototypes follows the design theory for a VIT, but the ideas behind the prototypes themselves are grounded in Essence and the criteria for a Daily Stand-Up. These prototypes lead to the creation of a proof of concept of a process to be used with the Essence methodology that helps promote joint inquiry during sprints.

2.4.2 Evaluating a DSM as a VIT

The design theory for visual inquiry tools provides a means of sharing information between team members and allows for visualising complex problems and its goal is to solve these problems, as described in section 2.2. These are the motivation for evaluating a daily stand-up meeting as a visual inquiry tool throughout this project.

Recall, as described in Section 2.2 that for each of the VIT design principles there are three sub-principles, and the questions that I use to evaluate are listed below. These questions, shown in Figure 2.7, are used as a means for evaluating not only a VIT, but also the entire process of the daily stand-up meeting that I develop.

Frame Are all elements needed to address the problem identified?

Rigour and relevance Do the elements adequately reflect the reality of the task?

Parsimony Are the elements parsimonious?

Functionality Do the elements allow for the facilitation of ideation and prototyping?

Arrangement Are the elements' relationships adequately abstract?

Facilitation Is there a common language in the form of images or visual arrangements?

Ideation Do the directions specify techniques for joint inquiry?

Prototyping Do the directions support alternative hypotheses?

Presentation Are the directions able to help present and critique the solution?

Figure 2.7: Questions for DSM evaluation as a VIT

In the following chapter, I use these questions to evaluate a prototype for a DSM to be used with Essence. This prototype is an initial proposal for what a meeting in Essence could entail, and by using these questions, I can see where there are limitations and how it can be improved in order to fulfil the design principles of a visual inquiry tool, the characteristics of a DSM and have relevance to the Essence methodology.

Chapter 3

Daily Stand-Up in Essence

The purpose of this chapter is to establish a process for Essence that can help promote joint inquiry between sprints. I do this by creating a prototype for a visual inquiry tool that reflects the characteristics of a DSM as by Stray et al. Essence does not currently have a process exclusive to Essence for promoting joint inquiry during sprints. This can be a problem due to the fact that without any indicators of potential challenges within a sprint, any work that is being completed could have the potential to be wasted if problems being faced are not questioned or analysed.

Therefore, I begin by looking at how DSMs can be used in Essence and what it looks like for a development team.

3.1 A Stand-Up Meeting in Essence

Currently, Essence does not have its own form of meeting between sprints. Instead, Essence suggests to its users that because Essence is compatible with other forms of methodologies, whichever work form works well for the development team, they should use that. As one of the most popular forms of daily meetings in software teams is Scrum, I assume that this is what would be used in conjunction with Essence.

As there is no study that discusses the use of Scrum with Essence, I have to make some assumptions as to how it could be used.

Currently, an iteration in Essence can be related to an iteration in Scrum. One sprint in a development project is illustrated in Figure 3.1. The red lines depict an agile process for software development, where reviews are performed between sprints. The green lines are where Essence means to extend it, to focus on what makes the software development meaningful [1].

There is precedence for what happens when a sprint ends, where we look at the rationale, strategy and tactics. However, this does not take daily occurrences into consideration. Figure 3.1 depicts that a sprint review and sprint planning are performed in close connection with the configuration table in Essence, where we look at the three levels, rationale, strategy and tactics. Essence already has an RST Review that mirrors a sprint review and allows for a reflection of the entire project. These three levels together describe the entire project and would be difficult to fit into the smaller scale of one 15-minute meeting per day, as a daily scrum meeting suggests as well as the DSM guidelines.

For the 24-hour iteration depicted in Figure 3.1, I have to reflect on what would have the greatest impact for a team to discuss between the larger RST Reviews. To begin, I look at the options that

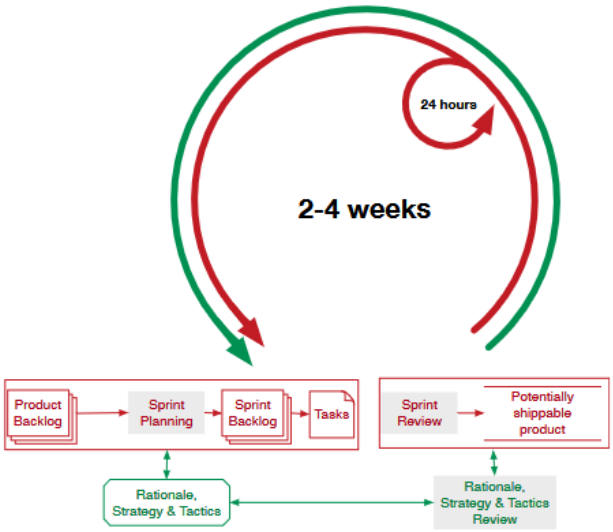

Figure 3.1: Incremental software innovation [1]

are available with regard to the Essence methodology.

The purpose of the Daily Scrum is ”to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work” [11]. If we imagine this as being used in Essence, the Product Backlog is similar to parts of the contents of this iteration's configuration table. However, the majority of actionable items in a configuration table are related to the core concern, contribution. So a backlog would consist of items that are found in leverage, inner environment or capabilities, as these are the items that are developed on during an iteration.

”To inspect progress”, as mentioned by the Scrum guide, implies that we take stock of our current situation [11]. This can be conducted in several ways, depending on how progress is measured. In order to inspect progress, one way would be by telling the team how far we are with this certain feature, and then explaining what the next steps are for completion and how we are planning on doing this. A problem I see here is that, it is not necessarily crucial for everyone on the team to know this information. And as a meeting should only last 15 minutes, we need a way of getting vital information across to a team, as not to disturb workflow more than necessary.

The outcome of a Daily Scrum should be an updated Sprint Backlog based on discussions. Each member discusses their progress in an item from the configuration table They mention what is on their agenda today and whether there are problems facing them and based on the member's status, it should be possible to evaluate whether the plan for this iteration is realistic or whether resources should be reallocated in order to complete the iteration successfully.

Even though parallels can be drawn between using Essence and using Scrum, discovering how Scrum can be best used in conjunction with Essence is not within the scope of this project, and I instead look at how a daily stand-up meeting can be tailored to Essence.

3.2 An Essence Project DSM - Case Example

As an example of a problem to solve, I use a fictitious case to illustrate the concepts I describe. The case is based on a semester project conducted by me, where I created an application that could measure the respiratory rate of a patient by using the inbuilt microphone in a smartphone.

A lung clinic wants a mobile application for their asthma patients to help them monitor their symptoms effectively at home. The software development team has discussed the various options available to them. What they discovered to be the most optimal solution for the lung clinic's needs and what they can deliver is then documented in their configuration table, ready for the first iteration of development. The project they are working on is based on a prospect that strives to address the problem that asthma attacks can be avoided if symptoms are monitored. The configuration table that describes it is shown in Figure 3.2.

For such daily meetings during sprints, I assume that the team are mainly working on their contribution to the project, similar to having backlog items, which is what the focus of a daily stand-up meeting would be.

Situation	Contribution	Solution	Valuation
Problem Asthma attacks are occurrences that sometimes can be avoided if symptoms had been better monitored	**Leverage** A smartphone and its inbuilt microphone	**Resolution** AsthmaAid - anticipate oncoming asthma attacks by monitoring symptoms, where monitoring symptoms can lead to awareness of oncoming asthma attacks, we can start with few features and build on it	**Criteria for rationale** Is this prospect a meaningful solution to the problem?
Outer Environment A system for asthma patients who use the system via a mobile application.	**Inner Environment** Computation module Mobile application client Digital symptom diary module Database	**Qualifier** The solution won't prevent an attack, but it will warn a patient if one is about to occur.	**Criteria for strategy** Does this architecture allow scalability? Do we utilise the opportunities elements present us with?
Manifestations A user can input symptoms Can record breathing Can see pollen/weather warnings Receives automatic warnings based on user input	**Capabilities** User profile Symptom database Connection to weather forecasts Computation of respiratory rate	**Value** Feedback on the shown asthma symptoms Potential warnings of an attack	**Criteria for tactics** Are potential warnings effective in reducing effects of attack? Does feedback help a user understand why they have an attack?

Figure 3.2: Configuration Table for an iteration

Due to the fact that in daily stand-up meetings most of the participants are developers, it seems most likely that any problems they are facing have a base in what they are working on - something in the contribution column, and therefore it is these problems that are brought to the meeting.Recall the table in Figure 2.1. To give the meeting form, I use the characteristics from Stray et al.'s paper as a checklist to make sure I address the points a DSM should encompass.

Purpose If we maintain that the purpose is to obtain a shared understanding of current activities, it would mean that every part of the configuration table that is currently being worked on would have to be mentioned. The current activities would be related to the Contribution column. This means that the purpose here is to discover how the team is getting on with creating a user profile, a database for symptoms, connection to weather forecasts and computation of respiratory rates.

Potential benefits Benefits include an improved communication. This means that everyone in the team is aware of what in the configuration table is being worked on, on the day of the meeting, and also knows who is having difficulties and might need extra resources allocated to their part of the project. Two developers are working on the mobile application client, and by talking to the team they are then made aware of a problem in the computation of the

respiratory rate. This awareness lets them know that it is okay that during tests they receive the wrong output, as the problem is already known. This is not, however, isolated for use of a configuration table, as it could also occur in a scrum team not utilising Essence.

Potential pitfalls We want to try to avoid creating a status meeting, where everything is reviewed and discussed, and instead focus on things that can have relevance to all team members. There are four capabilities for this prospect, with some requiring more resources than others. It would be a problem for a team if there was no adequate time to talk about the larger items, due to the fact that everyone had to have their turn saying what they are doing on this day, even if it is relevant to the team. The developer working on the user profile has a lot of work to do that day in order to complete their task before the iteration is complete, and therefore relay a lot of information that is irrelevant to the rest of the team.

DSM questions The guidelines suggest using the two questions, "What will I do?" and "What problems do I know of?" Each member of the team is supposed to answer these questions. For this case I assume that the team are aware of their assignments independently of the CT and who is working on which capability, but what they would not be aware of is the fact that when looking at the computation of respiratory rate, it was more complex than initially imagined. The first question, as answered by the developer working on the user profile has a lot to say about what he will do that day, but he has not experienced any problems.

Format Each participant is supposed to stand in front of a visualisation of their tasks, with room for communicating what needs to be done and when. An example could be that the user profile was finished earlier than imagined, and therefore it can be communicated as being complete until it is to be reviewed, therefore freeing a resource to be able to help elsewhere. Using the board shows the team that one capability is complete, showing progress to the team.

Turn-taking It is suggested that a round-robin approach is best, so that each member of the team gets their turn in mentioning their work. This would mean that there is no need for the Anchor of the team to act as facilitator, but instead be a part of discussions, if relevant to their work.

Frequency As per the guidelines, the meetings should be held regularly. However, not every sprint would be equal. It could be that this sprint has four capabilities, with varying degrees of complexity, but another sprint could have four capabilities that require more attention than in this iteration. Having to discuss when meetings are necessary is also a part of reflecting on the project's status.

Time of day The least disruptive time for the team. For the team it makes sense that they conduct the meeting at a set time so to make sure that everyone is present due to different arrival times.

Duration A maximum of 15 minutes does not allow for in depth discussion of greater problems, so it is necessary to be concise and perhaps have a plan for how larger items can be picked up after the meeting, if need be. If there are many problems being faced on this day, one 15-minute meeting where everyone's tasks are supposed to be discussed is not enough.

3.3 Initial Prototype

To make the daily stand-up meeting utilisable, the contents of a meeting should be clear. This is why I take the guidelines for a DSM and categorise them in to two parts. The process consists

of an informing step, and a discussion step, illustrated in Figure 3.3. What each step includes, is described below.

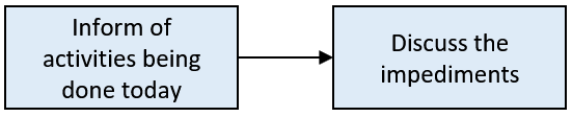

Figure 3.3: Process for a DSM

Inform When conducting a meeting, every participant should keep in mind that it should be brief. Therefore, the information given here should be simple and not go into too much detail as it is important, given the DSM guidelines, that every participant should get their turn to speak.

Discuss Turn taking I assume would be about discussions as well as when informing. The purpose is to discuss and solve problems, so a discussion would be based on the answers to the second question of what problems are being faced. The discussion should then lead to something actionable that can be worked on within the current sprint.

The issues that I see with this type of daily meeting, is that it is up to interpretation. Since I want to create a daily stand-up meeting that is tailored to the Essence methodology to promote joint inquiry, some alterations are necessary. The guidelines state that a benefit of the meeting is to share knowledge and identify, avoid and solve problems. However, the way in which this is done is up to the team. I see this as being problematic in that there are only 15 minutes allocated to the meeting, and therefore there should be a structured procedure that is easy to follow.

3.3.1 Evaluation of the VIT for a DSM

The simple VIT prototype shown in Figure 3.4 is supposed to reflect what the DSM guidelines explicitly state should occur. I evaluate this VIT, and therefore the daily stand-up meeting process, using the design theory for visual inquiry tools as a frame, to see where changes can be made. This initial prototype is based purely on the purpose of the DSM according to Stray et al., including the two DSM questions they feel give most value [5].

In order to see where changes should be made first, I use the list of questions from Figure 2.7 to determine the presence of the design principles in a prototype, as first mentioned in section 2.4.2 with regard to a DSM in Essence. After this discussion, I present my findings and begin looking at how the DSM can be adapted in order to better reflect Essence's values and methodology.

The elements that are referred to in the questions are, the list of participants, the second column and the third column containing the DSM questions.

- **Are all elements needed to address the problem identified?**
  There are no means of discussing solutions to the problems that are brought up at the meeting within the confines of the VIT, any discussion about the content of the VIT happens using the VIT as a starting point. The VIT solely enables inquiry into whether members are aware of problems and does not handle the addressing of actual found problems. It furthermore does not assist the team into identifying problems but relies solely on the team member's experience into identifying problems.
- **Do the elements adequately reflect the reality of the task?**
  The components reflect that each participant has a turn, and that they have two questions

	What will I do today?	What problems do I know of?
Responder1		
Responder2		
Responder3		
Respnder4		

Figure 3.4: Proposal for a VIT based on DSM guidelines

they have to respond to. However, what is not framed in the VIT is the fact that the overall purpose is to discuss problems and their solutions.

- **Are the elements parsimonious?**
  This prototype is very simple in its structure and contents, this does however lead to the actual goal of the meeting to not be shown within the confines of the VIT, which is a potential problem when discussing the issues that are brought up at meetings. The only aspect is noting the problems down. The team would have to be able to picks problems up themselves and correctly prioritise and address them with no further aid from the VIT.

- **Do the elements allow for the facilitation of ideation and prototyping?**
  There are empty cells for documenting answers, but no way in which a response can be guided. The empty problem spaces allow developers to describe their responses to the questions. However, for a shared visualisation, a method in which these responses can actually be further inquired is necessary.

- **Are the elements' relationships adequately abstract?**
  There are not that many components in this prototype. However, it is clear that each participant has their own space in which they can state their answers to the two DSM questions. This was created in order to visualise the DSM criteria that suggest a round-robin approach works best for software development teams.

- **Is there a common language in the form of images or visual arrangements?**
  There are no images in this prototype, the visual arrangement however makes it obvious that each participant should have some response to each question.

- **Do the directions specify techniques for joint inquiry?**
  There are no specific techniques other than presenting what the participants feel should be discussed. There are no ways in which a discussion can be focused but is instead dependent on the software team having a good sense of what problem makes most sense to discuss.

- **Do the directions support alternative hypotheses?**
  The fact that multiple problems can be presented within this VIT allows participants to see

what their team members are facing at this time. The VIT does not, however, allow for choosing which problem a team feels is most important but instead displays them with no means of focusing in.

- **Are the directions able to help present and critique the solution?**
  Similarly to alternative hypotheses there are no explicit directions for presenting or critiquing the information presented in the VIT, and is instead up for interpretation by the software development team.

For the conceptual model, I see that there are several things missing for the purpose of a meeting to be fulfilled. The most explicit aspect of the DSM are the questions, which are what I include in this first prototype. However, in this evaluation I discovered that a means of showing what the purpose and ultimate goal of the meeting, i.e. to address the problems rather than just state them, is missing.

The DSM should be expanded upon to include a means of displaying the conclusion to these meetings, otherwise it just shows the status of each participant. In general, there are no elements that facilitate addressing the problems that are brought forth in a meeting, nor is there a way in which a participant can share their ideas for solutions visually, as the criteria for DSM suggest is good practice.

There are no specific instructions for how the meeting should be conducted, apart from the guidelines. There is no support within the meeting for how a team end up at the conclusions that the DSM intends them to. At the moment it is a very free approach to addressing problems faced by the participants, which could be beneficial, but when restricted to 15 minutes and to try and avoid redundancy, it makes sense that the meeting should be more directed. This also helps a team self-manage, instead of participants fixating on one aspect that might not be the most vital for the current sprint.

3.3.2 Findings

This prototype does not fulfil any of the design principles satisfactorily, and therefore requires further investigation as to how this task can be adequately represented in a VIT. There is a lot that is left up to interpretation, which is problematic for a meeting that should not take up too much of the workday and should remove redundancies that take time away from more important issues. If a DSM activity is able to utilise a VIT, there are several things that need to be included. The VIT should show directions of use more explicitly, so that the participants know where they stand in the meeting process, there should be a means of deciding the severity of problems, there should also be a means of discussing potential solutions as a team. These aspects of a VIT are to be incorporated in the next iteration of creating a VIT for a DSM.

Considering time restraints, it could be difficult to include every single thing the DSM states are good, and we are instead obliged to see where things can be merged in order to still fulfil the purpose as well as getting the benefits of what the meeting should give us.

A shift in focus away from what exactly people are doing today would enable to focus more on joint inquiry through raising and identifying problems From the RST reviews, it would be apparent that people know what others are working on, and this update is not necessary every single day, but instead should most likely focus on what problems are being faced, if any.

It also becomes apparent when using the VIT, that not all problems are equal in size, and it is necessary to take into consideration the scope of the problem and its solution when deciding on what should be done about them. The next iteration of this prototype is to discuss how this can be included in a DSM and whether it is the correct forum for addressing problems in such detail.

Chapter 4

The Stand-Up Meeting

Previously I have called the type of meeting that is described in this chapter a daily stand-up meeting. However, due to the fact that it is not a requirement that these meetings are held every day, I alter the name of the meeting to be ”the stand-up meeting”.

After evaluating the initial prototype in section 3.3.2, I discover the aspects of the meeting that are most vital for a software development team. These parts are vital because the overall purpose according to the DSM guidelines is to be able to identify and solve problems, as well as increasing a shared understanding. These changes are incorporated in this section. Once these changes are included, I look at how the DSM can be altered in order to be usable as part of the Essence methodology. This next iteration of a VIT should include a means of looking at problems, and the solutions that correspond to them whilst including Essence methodology.

Even though the process in the initial prototype, as described in section 3.3, covers the purpose of what a meeting should do, it does not help participants reach what the DSM guidelines call potential benefits. These benefits being addressing problems to be able to both avoid and solve them.

4.1 Inclusion of Essence Methodology

To begin, the DSM guidelines are created in order to be as generic as possible, so any development team can use them. I, however, believe it to be advantageous to keep the methodology consistent in order to remove confusion with regard to terminology. As previously mentioned, using the Essence methodology results in a configuration table that is used to document the current sprint and what it aims to solve. When we look at the configuration table, we see that there are four columns, one for each concern of a project (see section 2.3 for an example). As in chapter 3, I limit the scope to being the contribution column, due to the majority of participants being developers. The reason for this relation is that if there are fundamental issues related to items in the contribution column that enables risks for the iteration or project, the related problems need to be addressed and handled properly and in a timely manner.

4.2 Contents of a Meeting

From the initial prototype, it is clear that it is not enough to just document what will be done and what problems the participants are facing. This is why I analyse the purpose of a meeting further

and how they can be incorporated in a way that encourages inquiry.

Currently, the prototype presents a way for participants to share their problems but not how they are to be addressed. Even though, the generic guidelines for a DSM state that one of the two questions should be "What will I do today?", I feel that this becomes redundant in a meeting that also addresses the participants' impediments and should not be the main focus of the meeting. This is because any impediment that is being faced needs a solution in order to move forward, and therefore takes precedence over a more status like activity. Instead, a meeting that focuses on the problems that are being faced gives more value in that, when people are faced with a problem, it is difficult to not think in terms of solutions. In order to further a DSM for Essence I see several things that have to be incorporated.

1. A specific set of instructions
2. A means of discussing problems and their scope
3. A means of deciding what to do with the information presented

To elaborate on what these points entail, I present the deliberations in the following chapters, resulting in the final process for this type of meeting in a development team.

4.2.1 Categorising Problems

In the "Grounded Theory Study" for daily-stand up meetings [5], one of the propositions was that discussing and solving problems had a positive impact on the team's attitude towards the daily meeting. This is due to the fact that working on problems is perceived as a useful activity. However, a problem we face when discussing the challenges being met in a software development project is that they differ in how complex and how extensive they can be. This can be difficult when trying to address a problem and its potential solutions, as we do not know whether the size of the problem nor its potential solution is going to fall within the scope that the development team has for their current iteration.

Something we know from Essence, is that the cells in the configuration table are not stand-alone items, they are connected to each other. This could mean that if a problem is discovered in one part of the configuration table, it could also be present in another. However, just because the cells are connected, it does not mean that the perceived problem is. This makes it difficult to know how a problem should be managed, therefore I present a way of categorising a problem using the connections in a configuration table. The information in Table 4.1, is based on the layout

Category	Definition
Small problem	1 category
Medium problem	>1 category in the same core concern
Large problem	>1 category in the same level

Table 4.1: A suggestion for categorising problems

of a configuration table. I assume that if a problem is only based in one category, for example, capability, it is a small problem that does not affect other aspects of the CT. A medium problem would be if more than one category is affected within the same core concern, as these have a close relation, an example of this being a problem that is found in capabilities affects an object in our inner environment. I define a large problem to be one that affects more than one category within the same level. This is due to the fact that each level depicts either why it is a solution, what

will the solution do, or how is it a solution. I see this as showing a fundamental flaw if there is a problem that reaches past one category in the same level.

Categorising the problem could allow us to know whether what we are discussing is actually in scope for our sprint. This should help avoid wasting time on issues that are too complex and actually require a more in-depth look into what the cause of this is, and if a whole review of the configuration table is needed instead.

Even though this solution for categorising a problem lets us know how much of the project it actually affects, it does not tell us how time-consuming or complex it could be. The categorisation of a problem could lead to a development team knowing whether the action point should be created for it or whether something greater has to occur.

The problems that fall into the ”large category” are perhaps too great to be able to be adequately handled within this sprint, but it is still up to the team to decide whether the problem is big enough that the sprint should be stopped earlier than expected, and perform an RST Review to get the project back on track. However, if there is a problem that affects the entire structure of a project, or in terms of Essence, each core concern, the team should stop the sprint and perform a review before schedule.

The RST Review requires that a development team looks at each cell in the configuration table and determines whether the points in them still have relevance. This would be too great a task for a meeting that should be brief, and therefore a categorisation allowing for quick decision-making seems advantageous.

4.2.2 Action Points

An action point is a task for a specific person or group to complete after a meeting or discussion [12]. I propose using action points for a DSM in order to give the development team a focus when convening for a meeting. The categorisation of action points is based on what sort of problems a team finds. The Daily Stand-up Meeting, according to the Scrum Guide, should promote quick decision-making [5]. This is where action points can be beneficial.

When looking at action points with regard to problems, an action point needs a description of what it entails, but having an overall category in order to quickly decide whether it should be looked into further or not could have a positive impact on the quick-decision making in DSMs. Therefore, I propose a categorisation, as to not use time discussing a description or timeframe if the action point is not needed at this point in the iteration. The action points categorisations for problems are listed in Table 4.2.

Category	Definition
Discard	The problem is already solved
Pursue	We pursue the problem, and determine the task, owner and scope
Queue	The problem needs to be addressed, but is not relevant in this iteration

Table 4.2: How action points can be labelled for dealing with problems

These three categories state what should happen to the problem that is currently being investigated. The first category is, discard. The reason for including this category, is that even if a problem is brought up at a meeting, there might be a solution that other team members are already aware of and can assist the participant who is experiencing the problem after the meeting. The second category, pursue, is for the problems that are relevant to the current sprint and need to be addressed immediately. This category would then trigger the development team to focus on this problem and

look into how it could be solved. The third category is to queue the problem. The reason for placing a problem in a queue, is that the team can see that the problem has relevance and should be investigated further, but they do not have the time to be able to explore it in greater details within this iteration.

I see the inclusion of action points being important due to the fact that it gives the meeting a focused purpose, instead of only being discussion with no constructive output.

4.3 Process for a DSM

Based on the deliberations above, I can now describe what each component consists of in a DSM for Essence.

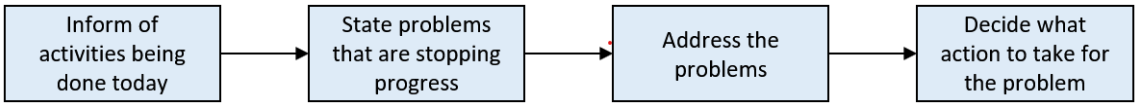

Figure 4.1: Components for the process

Inform of activities being done today When looking at what each participant of the meeting is going to be doing today, I do not see it as advantageous to the VIT, even if it is a part of the DSM. The reason for this, is that it does not promote joint inquiry on the same level as focusing efforts into looking at problems. However, if there are participants who feel that it is relevant for other members of the team to know what they are doing, this should not be ignored, and therefore it is placed as the first part of a meeting, in order to get it out of the way before inquiry for problems takes place. However, this part of the process should still be very brief and not be the focus.

State problems that are stopping progress It is important for the DSM to look into the problems that are being experienced by the software development teams. The most important problems to look at are the ones that stop progress for the current iteration or those that affect multiple CT dimensions. This is perhaps not evident for a team member before discussing it as a group. In order to see if the problem is limiting the project, or if there is a trend to the problems they should be discussed as a group. This could help a team see whether there is an overarching problem or if they are isolated problems.

Address the problems Once the problems have been discussed by the team, it is necessary to see if they have the means of addressing them within their current scope. Therefore, a quick analysis of the problem, namely, how large it is, can give a good indication of whether it is possible to address this problem within the sprint, or whether it is a symptom of a larger problem in the project. This helps us not have redundancies in our problem-solving as we can quickly see how many aspects of the project are actually affected by the newly discovered problem.

Decide what action to take for the problems When the problems have been addressed, a more detailed plan of what is going to happen within the next few days in the sprint can be investigated. The plan consists of knowing whether the problem is to be dealt with now, whether it can be discarded or whether the problem should be queued as it is not urgent in terms of creating most value with the time that we have in this sprint.

To illustrate this, I use the lung clinic case to provide an example of how the process could be in practice. This is described in Figure 4.2.

> The development team are ready for the DSM. Each member is present. The meeting starts, and the participants take turns informing of what activity is being done today, and if they are aware of any impediments that stop them from progressing on their task.
> The meeting starts with Devin, who states that he is working on the computation of a respiratory rate based on audio input, but during development he noticed that the current solution is based on looking at the sound frequency of input and since not all microphones record using the same frequency, this is going to be a problem. The team categorises this problem as a medium problem as it affects more than one category in the core concern. The team are in agreement with Devin that this is a problem, as not all users are going to have the same type of audio equipment to record their breathing. They agree that it should be investigated further in order to discover how the project is affected by not fixing this problem and what can be done to mitigate it. The team assign the relevant developers to look into how the problem can be solved, and estimate when they should look into it by. Therefore, the problem is marked as ”Pursue”.
> After this, the other members state what they are going to do today, and if they also are aware of impediments that hinder progress. Dolores mentions that she is having difficulty getting the ID of a user. During discussion of this problem, another developer mentions how this can be mitigated, and Dolores agrees with this assessment, and can apply it to her work after the meeting, leading to the problem being marked with ”Discard”.

Figure 4.2: Example of a DSM process for the lung clinic case

4.4 DSM as a VIT

I propose the prototype in Figure 4.3, based on the findings in this chapter for what should be included in a DSM. Since the last prototype, I have removed having a list of all participants on the same VIT, as it is not important to the DSM who has experienced the problem, as it is the action point, and who has the responsibility for the action point that takes precedence. The aspect that remains is the empty problem space for answering one of the DSM questions. This prototype addresses the components from the process on stating problems that are currently being experienced, and what overall action the team choose to take. In Figure 4.3, I show an example of what the VIT could look like for a DSM. Each problem that is mentioned at the meeting is written down, and for each problem that is brought forward, the team gives it an action in the form of discard, pursue, or queue. What each of these types of actions entail is to provoke constructive thoughts about how to cope with the problem in a way that aids quick decision-making.

Based on the considerations in the previous section, I analyse the DSM prototype as a VIT by using my evaluation method based on the design theory. The considerations that I make with regard to the design theory are used to create a visual inquiry tool that reflects each design principle. The proposal is illustrated in Figure 4.3. This VIT prototype aims to reflect the purpose and goal of a DSM with regard to Stray et al.'s guidelines, but also by trying to keep the number of elements present in the tool itself to a minimum. The purpose is to establish problems, and the overall goal is to know what is going to be done on the day the meeting is held and how problems can be solved. In the next section, I evaluate how the proposal fares as a VIT, as well as what this means for the DSM activity with regard to following the DSM guidelines.

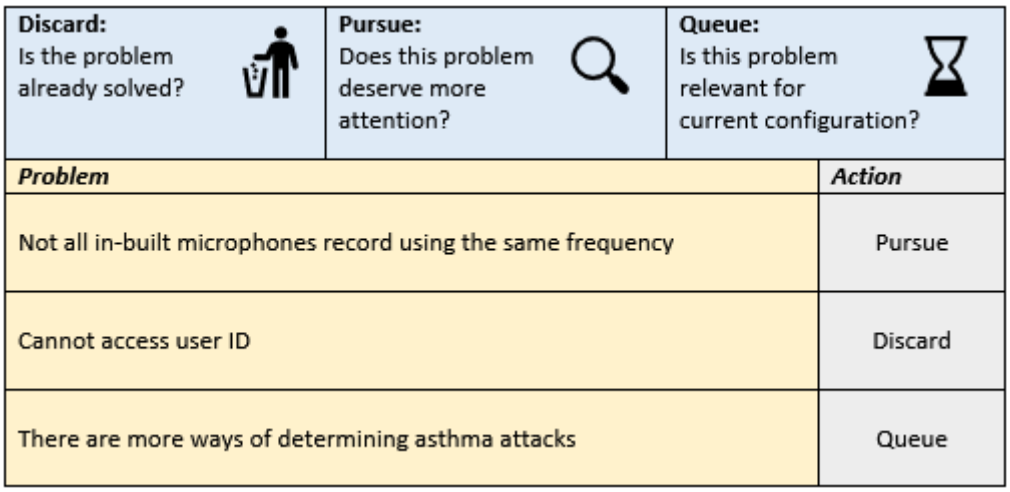

Figure 4.3: Three problems found at a DSM with the action to be taken

4.4.1 Evaluation as a VIT

In the following sections, I evaluate this prototype based on my knowledge of Essence, the DSM guidelines, and the design theory. To frame this, I use the questions from Figure 2.7 from Section 2.4.2. The elements that I refer to are the three ways of categorising what to do with a problem, the column for stating the problem and the column where the problem is given a course of action.

- **Are all elements needed to address the problem identified?**
  The overall construction of the VIT should identify all relevant components. For this activity, the focus of the DSM is the problem at hand, and how we plan to solve it. I see the two questions for DSM as being similar, and therefore when framing the activity, it is not necessary to include both questions as separate problem spaces, and instead focus on that which is actionable, the problem.

- **Do the elements adequately reflect the reality of the task?**
  Removing having both DSM questions improves the rigour and relevance of a VIT, due to the overall goal of a meeting being to identify and solve problems. Including action points also allows for quick-decision making with regard to the problems being experienced, and that this meeting is not for finding solutions, but for identifying the problem and its plan for after the meeting.

- **Are the elements parsimonious?**
  This principle is for removing an information overload for users, this is also one of the reasons why it makes sense to remove having a VIT that includes all participants' reactions to both DSM questions, as all it gives us is a status of what was said, and not what is to be done. Therefore, a VIT should focus on avoiding information overload and instead try and merge these questions and remove empty problem spaces for each participant. There are more components present now than in the previous iteration, despite having removed several aspects, such as a separate problem space for each participant. Even though it is important for the DSM guidelines, it is not necessarily important to have it depicted in the VIT, and instead focus on the purpose and the benefits that it can give us - identifying the problem and looking at how to solve it.

- **Do the elements allow for the facilitation of ideation and prototyping?**
  There are now specific parts of the VIT that allow for documenting the problem. However, it is not explicit in asking the participants and leading them through a process as much as it could be in order to complete the activity quickly. This is why the three choices for action are made explicit, so that the team can decide what is relevant for the problem at hand.

- **Are the elements' relationships adequately abstract?**
  The arrangement of the elements should be according to their relevance in our overall frame. This is why the elements should consist of a means stating what the problem is, if we can solve this problem at this point in time, and how it could be solved. The way in which the VIT is formed allows a team to see what their options are, as well as listing the action for each problem.

- **Is there a common language in the form of images or visual arrangements?**
  As I assume that the developers who use this tool are aware and use Essence in their software development projects, that they are aware of the meaning behind the colours. This way the team knows what they are related to in terms of core concerns for the project. I could have attempted to facilitate a stronger connection to Essence by keeping colours that are so heavily used throughout the methodology and applying them to the appropriate tasks. However, this could be done in a second iteration of this VIT to enhance its usability.

- **Do the directions specify techniques for joint inquiry?**
  Catering for the possibilities for ideation within a software team is imperative when discussing which problems are most vital and how they should be solved. Having empty problem spaces for allowing teams to write down their ideas, and filter them out as they discuss, would be helpful for this. In the prototype I include questions to help a team focus discussion on what they should do with the problems brought up at the meeting. These questions are meant to start a discussion as to what the relevance is and what should be accomplished.

- **Do the directions support alternative hypotheses?**
  There are many ways of addressing problems, and this is something that a VIT should also reflect. However, to create focus and quick decision-making, there should be a choice. This prototype makes a software team choose one action per problem. The benefit of this is that the meeting has a focus and a purpose. It is also made aware to the team that there are several options for how a problem can be handled i.e. choosing one of the three categories, discard, pursue or queue.

- **Are the directions able to help present and critique the solution?**
  The directions support decision-making by having to state what category the problem falls under i.e. whether we discard, pursue or queue. They help a team quickly look at their problems and decide whether the problem belongs to one particular category instead of another. It is not necessarily a way of critiquing the solution, but they are directions that help a team visualise how to handle and present the problems that are being experienced.

4.4.2 Evaluation as a DSM

Using the configuration table for a concrete case gives a greater understanding of what the options for a daily-stand up meeting are. The overall purpose of a DSM is to obtain a shared understanding of what the team members are currently working on. This characteristic of a DSM is fulfilled for this prototype, due to the fact that the meeting requires that each participant informs of the activity being worked on that day, as well as stating problems that are preventing progress. Making sure that we attain this focus also makes it possible for us to promote the potential benefits that a

meeting has. The benefits that we emphasise are open communication being key for the meeting, and identifying problems also being a main focus.

The way in which the potential pitfalls of a DSM are avoided is by removing redundancy for the participants. This happens in that the decisions made for each problem are not in-depth discussions, but instead a way the team can deal with them with the relevant team members after the meeting is complete.

The two overall questions, although not explicitly asked as stated in the guidelines, are included with the two components of the process ”inform of what activity is being done today” and ”state problems that are stopping progress”. Including the essence of what the two DSM questions are also helps us achieve the purpose a DSM has. However, the brief/inform question is not explicitly guided in the VIT as it is expected to be picked up by the team and not required to be documented in the VIT.

The duration of the meeting should be brief. This is possible as long as discussions do not go off-topic. This is where the focus of a meeting being on the problem, and not how to solve the problem is beneficial. This way, no decisions are taken lightly because the team feels they are running out of time, and instead the problems that are being experienced can be explored in detail at a later time.

4.5 Findings

In this chapter, I uncovered how a DSM can be used in relation to working on a project using Essence. Using the configuration table for a concrete case gives a greater understanding of what the options for a daily-stand up meeting are. The process I described allows for a development team to identify problems and decide what the next course of action is within the sprint.

This meeting is most likely not applicable for a team consisting completely of new developers, as they presumably would have a greater need for a follow-up to ensure progress and help to seeing what the problem means for the project. This proposal for a stand-up meeting helps me discover a means of promoting joint inquiry during sprints. Identifying problems at the meeting helps the development team discover problems quickly, which could be a benefit for the team as due to the stand-up meeting having a problem focus, they are more likely to look out for them during development. This not only promotes inquiry on a personal level, but as soon as they begin discussing their problems on a team level, it becomes joint inquiry.

Using the Visual Inquiry Tool design theory in order to frame the meeting gives the meeting focus and allows for a quick overview of all the problems that are presented at a meeting and what action should be taken. This is helpful for when working on the problems after the meeting due to the fact that they are documented, so the development team knows what problems are currently being faced and if we are going to pursue them within this iteration.

The presentation of problems at a stand-up meeting encourages inquiry on a daily basis, but once the problems are identified, how a team should pursue them is not within scope due to time constraints of the meeting.

This proposal assumes that when the stand-up meeting is complete, the participants go back to their work with the knowledge of what problems exist and whether they are relevant for their current configuration. It does not cover how to assist a development team with exploring how a problem actually affects the project and its core concerns with regard to Essence, or what types of solutions would be most beneficial. These findings are discussed in the next chapter, where I focus on how a team can discuss the problems that are identified in the stand-up meeting, and what the different types of problems can have of significance for the entire configuration.

Chapter 5

Focused Meetings

This chapter looks at how a VIT can consider both discussions of problems and their action points, and solutions and their action points. To begin, I discuss the similarities and the differences that these two focuses have. It seems intuitive that when you look at problems, you also have to look at the solutions that accompany it. I see this, however, as creating a daily standup meeting that could become quite complex if we both explore the problem and the solutions within the same meeting. I therefore propose that a meeting could have two focuses depending on what the software development team feel that they would get most out of. Are the members sitting on a problem that is difficult to deduce, or do they have several solutions that they would like to explore before trying to implement it. I see the overall process for these two types of meeting including three

Figure 5.1: The general process for a DSM for both problem and solution focus

components, illustrated in Figure 5.1. What each of these components includes more specifically is investigated in the following sections. We begin with the meeting that focuses on problems.

5.1 A Focus on Problems

The reason for looking at problems is to discover them early on, and identify how extensive they are. Smaller problems are not reliant on a meeting that involves the whole development team, but larger problems could be. Previously when discussing problems, the aim was to identify them and find a solution. This is how the DSM guidelines state how problems are looked at [5]. I do not see this as being satisfactory in order to fully understand the problem and result in effective meetings that help the overall project. In the last chapter, I devised a means of categorising problems, so that a development team would know the extent of the problem they are facing and what action should be taken next. For the problems that need further investigation, I present a meeting that focuses on exploring a problem in terms of the relevant configuration table.

5.1.1 Question-Based Discussion

In section 4.2.1, I established three separate categories that are dependent on how much of the configuration table is affected. I still see this as being a sufficient way of seeing how much of the project is affected by a problem, however it seems too simple when discussing the magnitude of problems. The means in which the extent of the problem is discovered, it should be more focused in order to give a team a more direct way of assessing the extent of the problem. One way in which I see this being achieved, is by invoking a response from the team by means of answering questions. An example of this could be by having one question per core-concern in order to prompt a response in a focused manner.

This is why I design questions for each core concern to provoke discussion. For the core concern, *situation*, the question could be "Does this problem affect our overall problem and the context of it? If yes, how?". For *contribution* the question could be "Does the problem affect what we are designing and implementing? If yes, how?". The third question for *solution* is "Does the problem affect why our contribution is a solution?". The final question for *valuation* is "Does the problem change what criteria we feel effects our solution?"

Depending on how much of the configuration table is impacted by the investigation into the problem, there could be several actions that have different consequences. For example, if all three columns are affected, it might be necessary to consider ending the sprint and performing an RST Review instead.

5.1.2 Methods of Evaluation

When evaluating a problem, there should be certain criteria. The problems I assume that would be brought into such a meeting, are problems that do not take one google search to be able to solve, and would have been discarded at the stand-up meeting, but instead those problems which require a more in depth look as to why the problem is occurring. Establishing this is used to identify problems that are not known and assess the impact of the project and therefore be able to assess what the next step should be.

The method of evaluating problems should include the CT, and it should be determined where the problem is, and how it impacts the rest of the configuration table. As the majority of participants would be developers, I see that it is most likely that the problems that are discovered belong in the contribution column. This is also mentioned in section 3.2.

In Essence, the connections between the cells in the CT are most visibly connected depending on their column or row. Either connected via a core concern or a level, Rationale, Strategy or Tactics. This can be difficult to visualise when experiencing a problem and being able to see which parts are directly affected, instead of being overwhelmed by the entire CT. As an example of how the contribution's column's cells can have an effect on the other parts of the configuration table, I refer to the illustration in Figure 5.2.

In order to evaluate whether a problem is extensive or not, I propose that the development team look at their configuration table and see how many of the cells are actually affected. The connections between cells in the configuration table help a team see how the project is linked. However, certain aspects of the product are going to have more value than others with regard to what it provides for the product. It is problematic if the aspects of a product that give the product value are in jeopardy and the problem is not easily solvable. Therefore, it should be investigated.

An example of this in relation to the Lung Clinic Case could be that the library that was used in order to calculate the respiratory rate from an audio file is no longer accessible. The fact that the library is gone, means that other parts of the project can no longer be worked on, as the project is

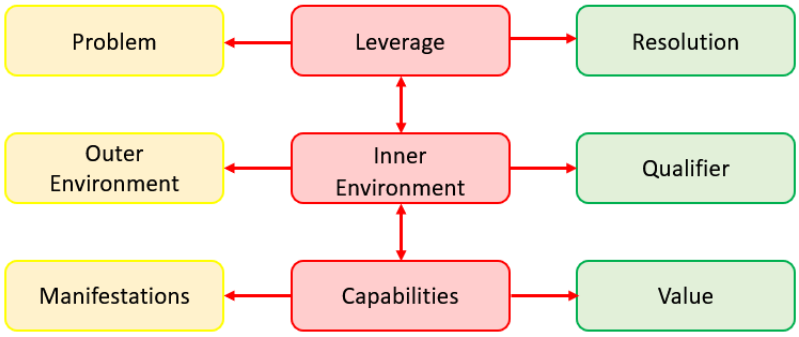

Figure 5.2: Connected categories from the contribution core concern

reliant on this one thing. In the category *capabilities*, the team have "computation of respiratory rate" noted. This capability, affects the fact that warnings cannot be given, a user won't receive feedback, and the computation module that was reliant on the library is now redundant. This problem affects several cells in the configuration table, so it is an important problem to look at now and determine whether it can be accomplished within this sprint, or whether the current sprint needs a complete overhaul in order to create value.

5.1.3 Sprint Pivot or Sprint Persevere

Previously I proposed a means of categorising action points depending on what the team decide to do with the problem. The team could either choose to discard, pursue or queue the problem. This categorisation is very intuitive when discussing the problem as it quickly becomes apparent if a problem should be pursued or not. The evaluation of a problem and how we should deal with the problem are closely related. Once we have determined what the scope of the problem is, it is possible to talk of what action should be taken. The action point, is not necessarily the only solution to a single problem, as there can be multiple solutions which solve the problem to varying degrees. Due to the fact that there can be multiple action points, it is necessary that the team look at them in response to the magnitude of the problem.

I suggest that there are two paths, either the team can choose to pivot or to persevere. These terms are already in use in Essence for the RST Review, that indicate whether a team want to continue with their current prospect or change [1]. Therefore, for this type of pivot or persevere, they are called sprint pivot or sprint persevere. In Essence a significant change in the configuration table is considered to be a pivot. Because a pivot is a major change in the direction of the project, the reasoning behind the decision should be thorough and measured. A sprint pivot is not as drastic as a complete change in the course of a project, but it would be a pivot in the sense that this iteration cannot be completed. A sprint pivot would occur when we discover that the iteration we are in cannot be completed effectively now this problem has been discovered. The reason for this can have two reasons, either the scope of the problem is too great, and we need to assess the whole project, or the solution for this problem has a lot of potential and should be investigated further.

5.1.4 The Process for a Problem-Focused DSM

Based on the considerations made in this chapter, the new process should reflect and include the relevant methods and means of evaluating problems. I have tried to formalise them, which has resulted in the process described below. An example of how I imagine this process could occur is

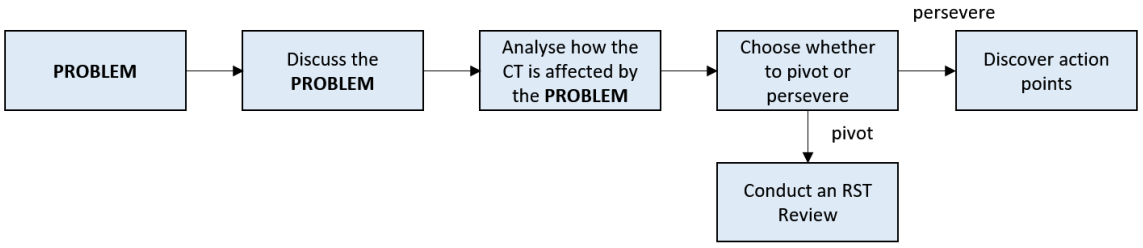

Figure 5.3: Process for a problem-focused DSM

described in Figure 5.5 and the overall process is depicted in Figure 5.3.

Present a problem For this component, it is important that a participant is already sitting with the problem and does not feel that he or she is capable of making the decision themselves as to whether something should be done about it and what that something could be. For the lung clinic case, the problem is "not all in-built microphones record using the same frequency". This problem is realised when testing the computation on several models of mobile phones. For some mobile phones that record with the same frequency, the computation works, but for others that operate differently, the computation was not accurate at all. This problem is presented to the rest of the team in order to discuss it.

Discuss the problem Once the problem is presented to the team, it is critical that each member that has something of relevance to inject, has the opportunity. The discussion is whether or not, the problem should be explored further, or whether there is a simple solution that would fix it instantaneously that the participant who experienced the problem oversaw. However, if the discussions lead to the team agreeing that it should be looked into, they need to see how far the problem actually goes. An example of a discussion in the Lung Clinic case is seen in Figure 5.4. Each core concern of the configuration table is discussed via a focused question. This gives an overview of whether this problem is extensive without going into detail immediately.

Analyse how the CT is affected by the problem An evaluation of the problem is used in order to see the extent to which it affects the entire project. The team have now discussed the problem in broad terms, but now they need to explicitly look at what aspects are going to be radically changed by this problem. The evaluation should consist of a means of determining whether it is possible to address within this iteration. This requires that the team evaluate how much of the configuration table is affected. From the previous discussion, the team discovered that not only are they computing the wrong respiratory rate based on auditory input, but this results in the warnings of oncoming asthma attacks also being wrong. The team are in agreement that this limits the value of their product, but it does not make the rest of the work in this iteration useless, as the affected parts are not fundamentally changed by this problem, they just need a solution.

Choose whether to sprint pivot or to sprint persevere Once the team has diagnosed the scope of the problem, they have to decide whether to stay on course, or whether they need to call for an early stop of the iteration, and conduct an RST Review. If they decide that it is possible to address within this iteration without affecting the progress of other work, a means of action should be discussed and set in action. The team decide to sprint persevere for their current iteration, as they cannot see that this is a change to their overall CT or that the problem is so large that they cannot find a resolution within their scope.

The Problem: Not all in-built microphones record using the same frequency			
Does this affect our overall problem to solve? If the microphone inputs read differently, it is not possible to give a user accurate warnings of a potential asthma attack.	***Does this affect what we are designing and implementing?*** If the output of the audio input is incorrect, we cannot compute the correct respiratory rate.	***Does this affect why our contribution is a solution?*** Not being able to compute the correct respiratory rate removes value from our product.	***Does this change what criteria affects our solution?*** We cannot see if potential warnings occur accurately enough to help reduce the effects of an attack.

Figure 5.4: A discussion of how the problem affects the Lung Clinic Case

Discover action points These are a reflection of the discussions of the problem. Once the team has decided to sprint persevere, they can look at how exactly the problem should be solved. For this meeting, they then assign the relevant participants, how long they have, and an estimate of what a solution could be. The action points that are discovered by the Lung Clinic case team, are that they should look into how mobile phones record sound, and find a way to make the frequencies similar. The team agree that it is okay if this takes until the end of the iteration as it is part of what they wanted to achieve from the beginning of the sprint. The developers who were tasked with the computation of respiratory rate are asked to look into it.

Conduct an RST Review If the scope of the problem is deemed too much, and the team choose to sprint pivot instead of sprint persevere, they have decided to stop the iteration before the original end date. The sprint pivot does not have to mean that it is a change of course for the project, but the problem was deemed so great that it made the work already being worked on for the current iteration redundant, and instead they can start a new iteration with this new-found problem in mind. This was not the case for the lung clinic team in this example, but if they had decided that the problem was too great and would remove all value from their product, they would have decided to change course and end the sprint early in order to accommodate this.

For the problem from the meeting, Devin and Dolores along with two other developers conduct a focused meeting. They discuss the problem in order to see how it could be solved. The problem they are faced with is that the computation of respiratory rate is reliant on the same frequency being used. This is a problem due to the fact that different devices record at different frequencies.
They begin by defining the problem as "not all in-built microphones record using the same frequency". In order to see how large of a problem this actually is with regard to what they wanted to build for this iteration, Dolores starts the discussion by asking the questions related to each core concern. The first being related to the situation: "Does this affect our overall problem to solve?" They agree that if different microphones record different it is not possible for us to provide an accurate warning of asthma attacks. The second question is related to the transaction: "Does this affect what we are designing and implementing?" They agree that it does affect the current iteration in that the correct respiratory rate is not output, which in turn also affects the third core concern by removing what we deem as our value for this iteration.
By answering these questions they are made aware that it is a problem that affects their entire iteration, but it does not show a fundamental flaw in the iteration, and it should be able to be solved within the iteration, and therefore decide to sprint persevere and find a solution.
In order to do this, they finish the meeting with action points for who will investigate the different aspects they have discussed in the problem-focused meeting. The action points that are discovered are looking into the different frequencies of mobile phones, and looking at making the computation dynamic depending on the device that is used. As these solutions could be costly, the team decide to analyse the solutions mentioned as well in order to choose the solution that benefits the project the most.

Figure 5.5: Example of a Problem-Focused Meeting using the Lung Clinic Case

5.2 A Focus on Solutions

A solution-focused meeting requires that a problem has already been identified, and we want to discover solutions and what they can contribute to our overall solution. The two key reasons I see for having a solution as a focus are determining the size of a solution, and determining whether there are any other benefits of the solution.

The potential solutions for the entire project were already completed in the prioritising ecology and leverage points phase in the pre-project. But when we are looking at solutions to problems within the overall problem to solve, we still need some general criteria so that the solution to our problem does not grow larger than intended or veer off course between sprints, something has to keep us grounded and focused. This is why I propose a method of evaluating proposed solutions.

5.2.1 Evaluation of a Solution

The evaluation of a solution is important because of several factors. These could be the amount of time the team has, the number of developers that can help, and also the benefits that it creates. Another factor for developing a solution-focused meeting is that even if it seems like a viable solution, the solution could be too complex for the current sprint without giving the development team any other benefit apart from solving one problem. This is problematic for a development

team if they discover at some point that this part of their product is not relevant any more, then the work that was made for solving a problem related to that part of the product is wasted. To mitigate this issue, I propose the following evaluation methods.

LCRT Comparative Analysis

An example of how evaluations of solutions could be conducted is by using a Leverage Cost Risk Time (LCRT) analysis [9]. Not only is LCRT an Essence assessment technique, but it is also helpful for a team to make decisions based on what the team gains with relation to how much it will cost them. The reason for including this as an evaluative method for choosing a solution is that it is a technique that is specifically developed for looking at what it can give a software development team [1]. There are two different ways the analysis can be carried out, either qualitative or quantitative and either as a comparative tool or a single-idea tool. I give an example of what this could look like for a comparative, quantitative evaluation in Table 5.1. For this example it is clear to a development team that SolutionOne gives the most benefit with regard to the resources it uses.

This can seem time-consuming to conduct, but if there are few proposed solutions and the focus is on conducting an LCRT analysis, the process should result in a clear answer as to whether the development team should go ahead with the problem. The LCRT analysis tool should result in intuitive, quantitative evaluation based on discussions in a group.

Solutions	Assessment						Sum	Total
SolutionOne	Leverage						9	3
	Cost	1	Risk	1	Time	4	6	
SolutionTwo	Leverage						5	1
	Cost	1	Risk	1	Time	2	4	
SolutionThree	Leverage						9	2
	Cost	2	Risk	2	Time	3	7	
SolutionFour	Leverage						8	1
	Cost	3	Risk	3	Time	1	7	

Table 5.1: Evaluation of Potential Solutions

Valuation Criteria

As another means of evaluation, we could use the criteria from the Valuation column in the Configuration Table. In the Essence methodology, Valuation is described as being "used to offer a range of ways to develop ideas, invent alternative lines" and to "assess and evaluate in order to provide a sound basis for decision-making" [1, p.120]. In other words it helps us get closer to a meaningful end to the sprint. During an RST Review, a team fills in the cells in the CT as to reflect their current sprint. In this CT, the final column includes the criteria that we wish to adhere to. This evaluation is dependent on the team's efforts at the previous RST Review, so if the criteria listed here are not sufficient, using this as an evaluative tool, won't be either.

Effectuation Principles

Another means I see of being able to evaluate whether a solution is a good investment for the development team, is by using the principles of effectuation. The reason for this is that it helps us focus on our means, rather than an ultimate goal. A solution arises because a problem has been experienced. This does not mean that this solution is the optimal one for the project.

I propose that we use Sarasvathy's principles of effectuation in question form to focus the discussions of a solution towards helping a team stay within their means. The effectuation model involves a one-to-many mapping, where the focus is on what we can control, and to discover what we can create with these means [13]. The focus is on choosing the most desirable effects that we can produce with the means available to us. This seems important when choosing solutions. An example of what these questions could be, based on Sarasvathy's principles of effectuation, are listed below [14]:

- What value does this solution create?
- What opportunities does this lead to?
- Are there already solutions that do this?

Another aspect of effectuation to consider is near-decomposability. This requires that a solution allows us to change its implementation without having to change the interface. This is of course not always possible, but it is important to keep in mind when creating one's solution to be able to avoid waste. Sarasvathy states that "The key to understanding near-decomposability is that in this architecture, what constitutes a good design for a component is nearly independent of the designs of other components" [13] so even if the solution is not that great for this particular problem, it is not wasted and can be used somewhere else.

5.2.2 The Process for a Solution-Focused DSM

Sometimes a solution can seem very obvious when faced with a problem. However, it is not always the case that there is just one solution to a problem, there can be multiple. How does a developer decide which is the correct direction to go in and what sort of criteria should we use?

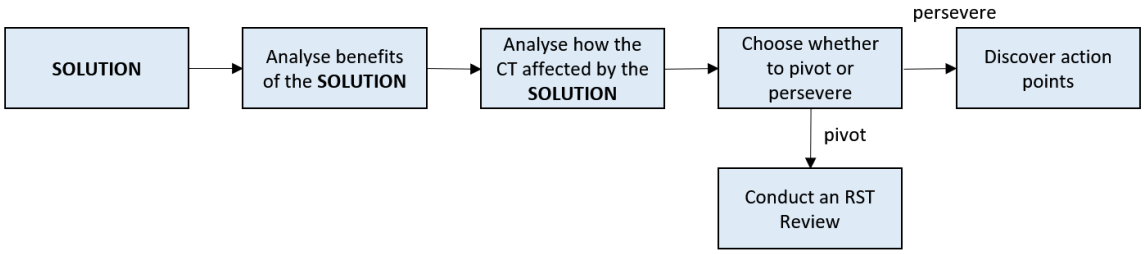

Figure 5.6: Process for a solution-focused DSM

The solution chosen should give us leverage, so it is not tailored to one specific project part, but could have potential in other aspects of the project, if in case what we are doing loses relevance.

This is why I present the process in Figure 5.6. What each component in this process consists of is relayed below.

Present possible solution This component implies that the development team are already beginning to look at solutions to a problem they have experienced within this iteration and whilst trying to overcome their problem, feel that it is necessary to present a solution to the team before they implement it. It must also be made clear that it is not every problem and solution that needs to be brought forward at a meeting, but if a developer thinks that the solution they have in mind could be larger than the scope, there must be a good reason for implementing it and discussing how this can be done. The solution to discuss for the lung clinic case is that we want to determine the frequency at what a device records with in order to make them similar.

Analyse the benefits of the solution This component is for the other members of the team to see how they can leverage this solution and whether they see it as a good use of their means. If the consensus is that it could be a good idea to look into, they can evaluate it. In order to discuss how good a solution could be for their product, I see it making sense to use Sarasvathy's principles of effectuation in order to lower the risks of implementing this solution and keeping in mind whether their solution also could be used for other aspects of the project or other projects. For evaluation, I see it as being necessary to have some questions or other evaluation tool that can be used in order to critically evaluate whether this solution is better than anything else they can come up for the problem that is being faced. An example of this could be a LCRT analysis, or questions that stimulate focused discussion for the criteria of the team. The team discovers through asking questions related to effectuation that they are not

Solution: Determine what frequency the device records with and level it, so they are all similar		
What value does this solution create? It allows us to accurately compute respiratory rate no matter the device being used, this in turn lets us accurately produce warnings of an oncoming asthma attack.	***What opportunities does this lead to?*** Being able to determine the frequency of any mobile phone that uses the application means we do not need to update a list over the frequencies that they use whenever a new phone. The frequency levelling could also be use for other recording devices, it could also help processing noise reduction	***Are there already solutions that do this?*** Mobile phone companies already list the specifications of the phone's recording equipment, is there a library that gets updated with this, that we could use?

Figure 5.7: Example of using questions for the solution processing

aware of other solutions that currently do this, but there could be a list of all specifications instead of having to look them up individually, but that a solution would be better if it can be determined at the time of recording, instead of having it hardcoded into the project. The LCRT analysis can also ask the team questions about their product with how it would impact their project. The example for the qualitative LCRT analysis is seen in 5.8.

Figure 5.8: Example of using a qualitative LCRT analysis

Analyse how the CT is affected by the solution The reason for analysing the reach of a solution, is to see if there are aspects that are no longer needed or if it enhances other aspects that were not yet seen. Another means of analysing this could be to use Valuation column in the Configuration Table, to see if the solution is in line with these criteria. The analysis could be similar to that of the problem-focused meeting, as it focuses discussion onto the separate core concerns instead of the more instinctual reactions that could arise when asked if there are other parts of the project that could benefit or be limited by the solution. The team can see that the way in which the CT is affected is only positive in that it helps us keep our value and helps uphold the manifestations if we implement this solution. This analysis helps the team choose whether to pivot or persevere.

Choose whether to sprint pivot or sprint persevere Once evaluation has occurred and the team are in agreement as to what has to be done within this iteration, they need to decide whether they can continue with this iteration or whether they should end it early. If they choose to end it early, it would be in order to incorporate a solution that affects a lot of the configuration table, and therefore what is being worked on. If they choose to sprint persevere, it is because the solution is not so extensive that it makes the rest of the work in this iteration

The solution they want to discuss is "determine the frequency of what a device records with". They are not sure if this is the best solution, and therefore begin the analysis using the questions created to provoke thoughts on effectuation. They go through each question and finally, can have a discussion on whether it is still the correct solution to go with. They agree that the solution creates a lot of value for the project, they have also identified that identifying the frequency can also be used in other parts of their product, for example, being able to help with processing the reduction of noise in an audio input. They are not aware of any solutions that currently do this that they are able to utilise, but Devin suggests that they can test whether the frequency at which a device records at is correct by looking at the specifications of the different devices.
The solution they have agreed upon does not require a sprint pivot, and instead they persevere and discussion of the solution is complete. The meeting resulted in action points that they now formalise in order to be able to implement the solution efficiently. Once the action points are delegated the team can begin implementation of the solution and continue their work to complete the sprint.

Figure 5.9: Example of a Solution-Focused Meeting using the Lung Clinic Case

redundant and fits in well with the project in its current direction. For this example, the team chooses to persevere as it is in line with their current iteration, and it would not result in an iteration going beyond its scope.

Discover action points Action points are decided once a team has chosen to persevere. The action point should state who is responsible, when it should be done by and how it will be included. The team choose to persevere with implementing the solution, and therefore they need to discover how this should happen. The team decide the appropriate people to look into implementing the solution and in tandem with them, ask how long they think it will take in order to plan their iteration with the new solution in mind.

Conduct an RST Review The RST Review is conducted when the team decides to stop the sprint early. If the team choose to sprint pivot, it is because they conclude that the solution is not within scope, but it would give value to the project in such a great way, that the sprint should be stopped in order to give the solution more focus.

5.3 Problem and Solution-Focus as a VIT

Both types of meetings have a similar approach. The main difference between the problem-focused and the solution-focused is the contents of the components, even though the overall goal is the same; we present an idea, we discuss it, we evaluate it, and then we discover action points.

The problem-focused VIT shown in Figure 5.10 concentrates on the parts of the process that are vital to completing the meeting. Namely addressing the problem and its scope, as well as how the team should manage it. The solution-focused VIT shown in Figure 5.11 is similar in that it focuses on discussing the proposed solution's effect on the overall prospect, and the action points that describe how it should be managed.

Due to the fact that the two proposals are similar, I evaluate them as one VIT, only mentioning the differences if they are applicable to the question. In order to evaluate what could be missing from my process with regard to promoting joint inquiry, I again use the evaluation method for a

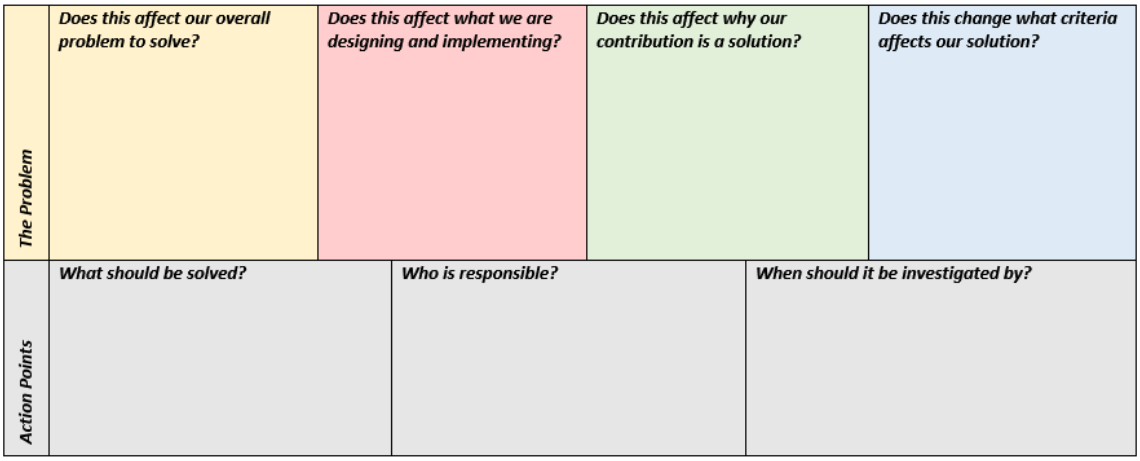

Figure 5.10: Problem-focused VIT

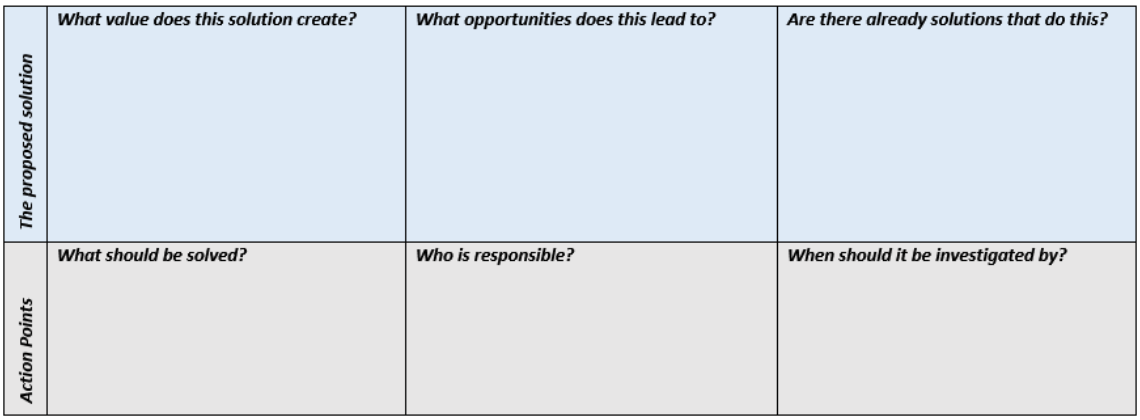

Figure 5.11: Solution-focused VIT

VIT, first shown in Figure 2.7 in Section 2.4.2, on my prototype for the two types of meeting. The findings from this evaluation aid in formulating what should be changed in order to be a successful VIT, but also help discover how the meeting itself could be conducted.

The elements I discuss in my evaluation below, are the problem spaces for discussing the problem or solution, and the empty problem spaces for discussing the action points of the problem or solution.

- **Are all elements needed to address the problem identified?**
  In the VIT I identify the elements that I feel are most important, discussing the problem at hand and how the team plan on managing it. Some aspects are missing from the VIT, for example, how exactly the configuration table is affected, but in the problem-focused VIT the CT core concerns are mentioned implicitly via the use of colour. This is not true of the solution-focused VIT, which relies solely on team discussion and does not have a focus on the CT.
- **Do the elements adequately reflect the reality of the task?**
  The task at hand is to discover whether the team wants to persevere with the iteration or not. The VIT assumes that a team has chosen to persevere, and therefore does not visualise

pivoting to an RST Review, but focuses on the meeting at hand, and what the team should do when solving the task.

- **Are the elements parsimonious?**
  The components are parsimonious in that they do not have an overlap and have focus in two separate areas, but are still connected via one being the discussion of the task i.e. either the problem or solution, and then addressing the action points.

- **Do the elements allow for the facilitation of ideation and prototyping?**
  The components each have an empty problem space that although being focused allow for free discussion between team members and to ideate how their task can be handled within this iteration.

- **Are the elements' relationships adequately abstract?**
  The relationship between the task at hand and the managing of the task is separated in that each has their own row in the VIT, the first row being for discussion of the task, and the second being a discussion of how we can manage the findings from the discussion.

- **Is there a common language in the form of images or visual arrangements?**
  As previously mentioned, the division of task and action points are split so that they each have their own row. The problem-focused version of the VIT also maintains the colours of the configuration table in order to visualise what aspect of a project they relate to.

- **Do the directions specify techniques for joint inquiry?**
  Within the VIT there are directions that explicitly tell the participants of a meeting what they should be discussing without being restrictive. The questions are there to used as a stepping stone and as a means to start discussion with a focus.

- **Do the directions support alternative hypotheses?**
  In that the VIT has questions that can be used as a discussion starter, there is no final answer that has to be presented, therefore a team is able to let their discussions wander, but they can always come back to the question that initiated their discussion if they feel that they are off track. The questions are meant to let a team explore different options that could be useful for the task they have to complete.

- **Are the directions able to help present and critique the solution?**
  The directions in the form of questions are useful in that they make the development team reflect on their decisions, before finally deciding how they will handle the task. This reflection is based on group discussion where they can present their ideas and hear what the other participants think.

The conceptual model follows the overall process that is presented in section 5. Where it differs, is the methods in which problems and solutions are evaluated with regard to how much of the configuration table they infiltrate. This is a conscious decision, due to the fact that the VIT should not be overly complex, and should focus on that which is most important for the task at hand.

The shared visualisation is simple in that it should just aid in leading discussion and having a means to gather the team's thoughts on problems, solutions and action points. Even though the visualisation is simple, it groups the team's thoughts in a way that is easy to understand at a glance.

Using questions to promote joint inquiry gives discussions focus, however, it does then force a team in a certain direction. The problem with this, is that when developing the questions, one must be certain that they provide value for the team, otherwise it could be difficult to have productive discussions.

5.4 Findings for Problem and Solution-Focused Meetings

Choosing to have the meeting as separate entities allows me to look closer as to what the differences between handling a problem and handling a solution are. I see each of these types of meetings as having value for reflecting on problems and solutions to keep them within scope of the team. However, this is perhaps too concentrated to be classified as a Daily Stand-Up Meeting, and if a team were to employ the processes discussed in this section, a different type of meeting would have to be held in order to communicate effectively.

The problem that I see with this model is that it is difficult to imagine one meeting without the other. To illustrate this, we imagine that we are dealing with a problem. Once we have investigated this problem, we could be made aware of a potential solution. The solution is then investigated, and once that is investigated, we can run into another problem.

In chapter 4, I suggest three types of categories for action points for the problem being experienced, either to discard, pursue, or queue. However, I now see the definitions of the "pursue" category as being slightly different. Previously, it was to "determine the scope, the task, and the owner" of the problem, but after the addition of focused meetings, I now see the category as the team choosing to conduct a problem or solution-focused meeting depending on which type has most relevance for the problem. If the team does not choose to pursue the problem being presented at the DSM, then the sprint continues until the next allocated DSM.

5.4.1 The Problem and Solution-Focus Pairing

The reason for splitting the meeting into two, was to be able to give time for immersion of one topic at a time. However, since the decision of having this type of meeting move away from being a DSM in the traditional sense, perhaps they can be combined in some form to complement each other. These meetings do not have to occur sequentially, they can be conducted independently of one another. However, when impediments are discussed, in order to move forward, there has to be a solution. If the solution that is presented needs a more focused discussion, it would make sense to use an evaluation method that allows the team to decide what solution makes most sense before implementation to avoid waste and increase value.

The process in Figure 5.12 shows that assessing the problem, and assessing the solution have similar steps. It connects the two different meetings by depicting that when we find a solution, this in of itself can lead to new problems, and when we discover what action needs to be taken for the problem, we have discovered a solution.

This is the reason for combining the two focuses to being one meeting that can have a different starting point, depending on what has most prevalence. This means that it would be one type of meeting, a focus meeting, but with two types of entry points.

The authors of "Identifying Viable "Need-Solution Pairs": Problem Solving Without Problem Formulation", von Hippel and von Krogh, suggest that "in informal problem-solving, a need and a solution are often discovered together" [15]. The reasoning being that a solution can be raised, even though we were not aware of there being a problem. The solution can give us leverage that we did not previously know existed, and it prevents a team from having to formulate a problem, just because they have discovered a type of solution that could have great value for a project. The problem is not formally identified, but it is still raised due to the introduction of a new solution.

To summarise, I have discovered that the type of meeting proposed in this chapter is not suited for a daily stand-up meeting as it requires a lot of discussion and deliberation, and that when used by multiple members in a team who all have their own opinion, most likely will exceed the criteria of being "brief". This is why I propose that this meeting can be used in conjunction with a DSM,

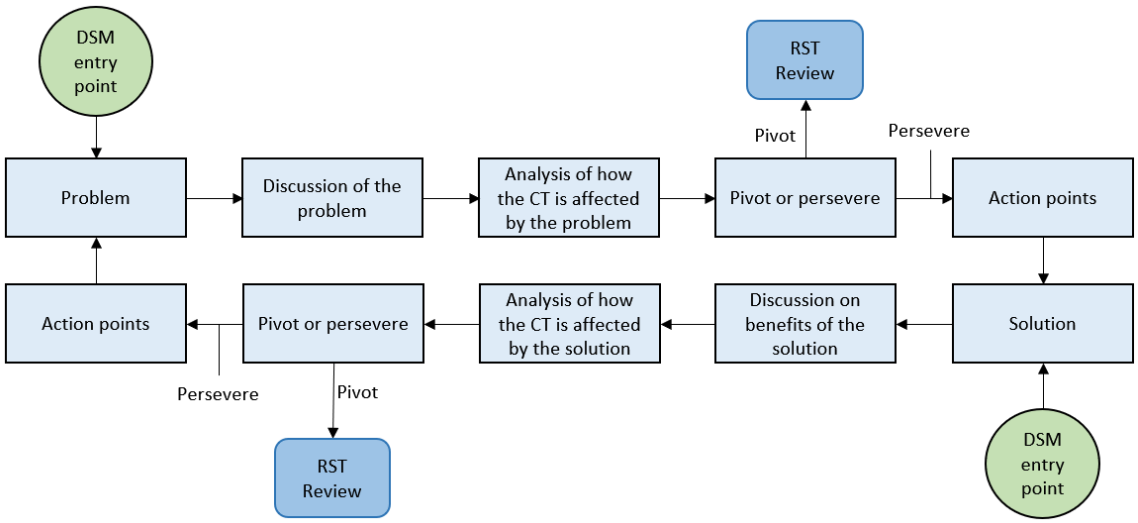

Figure 5.12: The problem-focus and solution-focus relationship

instead of as a DSM. The purpose of this meeting is to gain a deeper understanding of the state of a project, whether it is from the perspective of a problem or a solution and analyse whether what is being worked on is still relevant.

The next chapter is a look at how I can combine the stand-up meeting with the meeting suggested in this chapter by addressing the relationship between a DSM and a problem or solution focused meeting and the relationship between a problem and solution focused meeting.

Chapter 6

Activities during a Sprint

During the course of the project, I address the stand-up meeting, and the focused meeting with either a problem-focus or a solution-focus. The daily stand-up meeting is investigated to see whether it helps promote joint inquiry during sprints. However, after seeing the need in a DSM to address problems and their solutions, it is apparent that one DSM that follows the guidelines as described in section 2.1, is not enough. In this chapter, I address how the different meetings can be utilised by a software development team using Essence, and how they can be used in combination in order to promote joint inquiry and problem-solving within a sprint. The aspects of the meetings to be changed in order to better accommodate each other are addressed and implemented in the following sections. This chapter shows the final prototype for promoting joint inquiry within a sprint. It includes a description of what the prototype tries to accomplish, discussion of the prototype, as well as a proof of concept example of usage. Finally, I look at aspects of the process that are not explicitly addressed, as well as the advantages and disadvantages that they come with.

6.1 A Sprint in an Essence Project

The diagram in Figure 6.1 shows my proposal for what a sprint could look like in Essence where the proposed meetings are incorporated to try and ensure joint inquiry.

A diagram for what the steps include is depicted in Figure 6.2. Before this proposal, Essence's process consisted of creating a configuration table to state what the overall project addresses. It is then up to the team, what their sprint should include from here. Once the sprint is complete, they then perform an RST Review. I have since added two types of meeting that should occur between these two steps. In other words, what happens between the sprint start, and the sprint end. The scope of this project is to see how inquiry can be promoted during a sprint, which is why I focus on the stand-up meeting, and problem and solution handling.

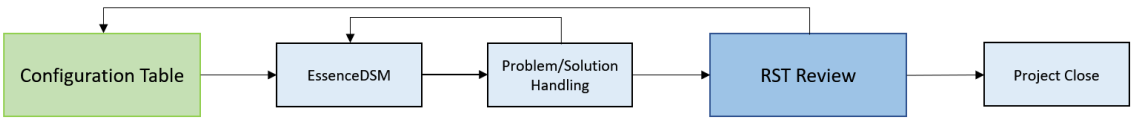

Figure 6.1: Product development cycle for project using Essence

Stand-Up Meeting The goal of the DSM is to make the team aware of impediments that hinder

progress for the development team and to attempt to solve them. If we also want to follow what is considered good practice for a DSM according to Stray et al., it is difficult to do this in one meeting. This is why the stand-up meeting for Essence that I propose needs alterations with regard to its overall goal.

Since the addition of focused meetings, the stand-up meeting should still highlight issues that are being experienced, but instead of attempting to discover solutions, we discover who is going to look into the problem. The reason for this, is that the complexity of a problem is not known before we investigate it. Therefore, in order to not rush a decision regarding a problem or draw what should be a short meeting out, we delegate this to a meeting specifically tailored to dealing with it.

This way the stand-up meeting maintains its goal of allowing the team to address problems, but it removes the pressure of having to solve them within a meeting, and instead, the people who have relevance for the impediment can investigate it thoroughly. This also helps address the issue that Stray et al. raise, where participants do not like having the meetings because they feel that a lot of the information is not relevant to them [5].

For this meeting, the components are to inform of the activity being done today, state problems that hinder our progress, address the problems to decide what action needs to be taken. These components are based on the guidelines for a DSM. With regard to promoting joint inquiry during sprints, it is necessary to have a meeting that provokes participants to share problems, so that they can be addressed by the team. The VIT that was created for this meeting, focuses on the second DSM question rather than the first of "What will I do today?". This is positive because it has previously been established that this sort of contents leads to demotivation during stand-up meetings. The findings in the previous prototype lead to the realisation that actual problem-solving within a traditional DSM is too substantial and would not follow the guidelines for a successful DSM. This is why the prototype for problem-focused meetings focuses on exploring the problems being faced in greater depth and allows for immersion into how the problem affects the team's work and overall project.

Problem/Solution Handling This is a meeting that can have two focuses depending on the task. For a focus on a problem the parts of the process are to present the problem, discuss why it is a problem, evaluate whether it is extensive with regard to the configuration table and then decide what action needs to be done. The VIT created for this meeting focuses heavily on asking questions about how the problem affects the overall project in terms of the configuration table, as well as asking a team to discover action points with regard to how the problem should be addressed.

For the solution-focus, we are to present a solution, analyse why it is a solution, evaluate its viability and then to decide what needs to be done in order to implement it. The VIT for this meeting focuses on asking questions related to effectuation in order to ensure that any solution being presented provides value in some form to the software development team. The VIT does not ask questions about or represent aspects of the configuration table.

The result of a problem or solution meeting can be a directed trigger to end an iteration earlier than anticipated. This happens when the team determine that whatever problem or solution is greater than the contents of the iteration, and needs a more extensive review, in the form of an RST Review. The other direction a problem/solution handling meeting can take, is to persevere with the iteration and solve the problems as they are discovered and can be solved within the means of the iteration. It could be argued that this meeting also could allow us to change the configuration table, however, I maintain that changes to the configuration table should only occur in connection to a greater RST Review that looks at each core concern of the project in depth. In order to keep the complexity of an iteration down, I do not think that the configuration table should be subject to change unless a full review occurs. The

reasons for this are that, firstly, it would be time-consuming, and secondly, it would result in a sprint that suddenly has a new focus. Another reason as to why it would not be necessary, is because problems that allow for us to persevere would most likely not change the contents of the configuration table.

6.1.1 The Overall Process

The process that I establish for use in development projects using the Essence methodology is depicted in Figure 6.2. The diagram shows the two types of meetings and how they are utilised with regard to the rest of the Essence methodology. The configuration table is green due to its relevance to the core concern solution, which is represented by the colour green in the Essence methodology. This is also applicable for the RST review, which is represented in blue in Essence.

Each meeting type is encapsulated in order to establish that they occur separately from one another, where it is either a problem or a solution that is the connecting factor. In order to simplify the diagram, I have generalised the points of the meeting for either a meeting that handles problems or solutions, as they generally include the same points of discussion, but with a different focus.

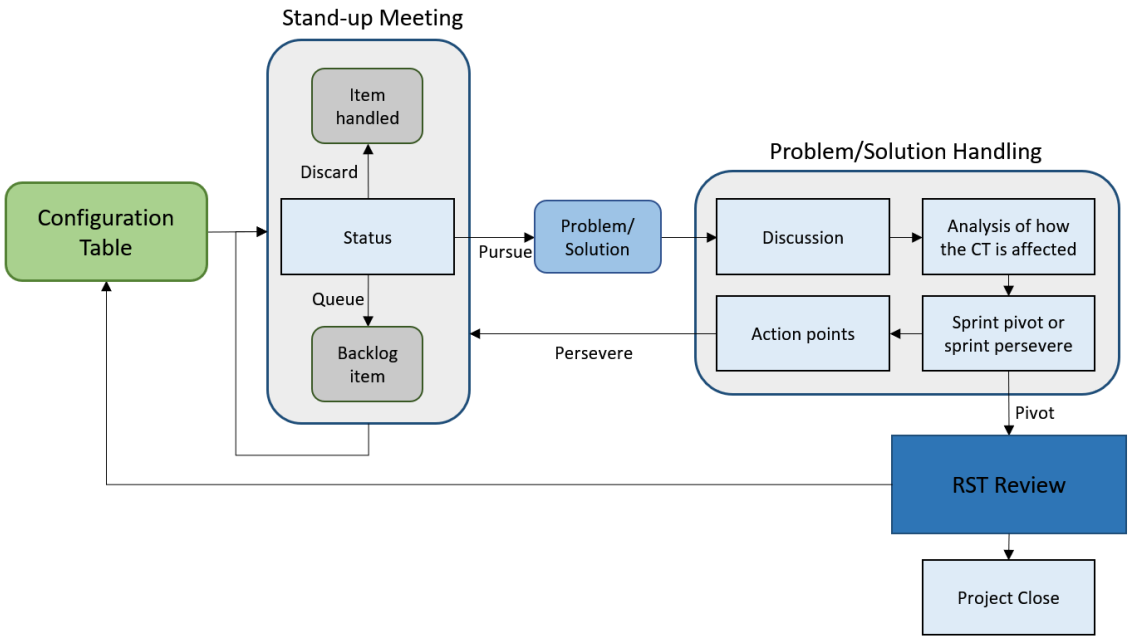

Figure 6.2: The process with both meetings

6.1.2 The Lung Clinic Case

The examples in this section are based on following the process shown in Figure 6.2.

A Discard Case During a stand-up meeting, Devin, shares what he is doing today, and expresses that he is having difficulty accessing user IDs in order to complete the diary based warnings. Dolores hears this problem and knows how shr can help, so they mention this to the team, and they agree that the two developers can look into handling the problem after the meeting.

The problem is still documented as having been risen at the stand-up meeting, but is noted in the VIT as being "Discarded". How the two developers handle a problem of this type outside the meeting is not within the scope of this project.

A Queue Case It is Dolores' turn to share what she is working on. During development, Dolores has realised that there are other ways that the team could determine asthma attacks other than those already present in the configuration table. The team find this interesting as it could give a better warning to a user if there were more ways in which the team can measure and estimate asthma symptoms. They decide, however, that it is not within the scope of this iteration, and that it should be marked as "Queued" in the VIT in order to investigate this problem during an RST Review for a future sprint as a backlog item. Due to the fact that all items that are raised at the stand-up meeting are documented, I imagine that these items could be picked up as a part of the RST Review and determine if they have relevance for the upcoming sprint. The RST Review is not, however, within the scope of the project, so I do not investigate the potential of this further.

A Pursue Case In order to depict how a sprint would be conducted, I include an example, and argue for why the process has the sequence it does. To begin, a development team have finished updating their configuration table for the lung clinic problem, and are ready to start a new sprint. Every developer within the team is present at the daily stand-up meeting. The meeting begins by one Daniel stating what he is going to be working on today and whether he has a problem that he wants to raise. Daniel shares that he is working on the computation of respiratory rate, but that in order to provide something that is of value, he has to discover a way to make the computation correct no matter the frequency that it is being recorded at. When this is mentioned, no one on the team has an immediate idea as to how this can be done, and they agree that it is a major flaw of their product, and that the problem is to be pursued. Once every participant has had their say, the team can look at the VIT (Figure 6.3a), where they have a list of problems experienced that day, and then quickly delegate who is responsible for looking at the problems marked as pursue.

Some point after the stand-up meeting, Daniel and the other relevant participants conduct their meeting for problem/solution handling. In this case it is problem handling where they would like to begin. They are going to be discussing the issue of different frequencies of sound affecting how they compute a respiratory rate. To begin, they ask each question based on the configuration table to see how extensive the problem is. They discover that it is an isolated problem, in that each core concern of the configuration is affected. However, it does not harm the other members of the team who are working on separate components, such as user profiles or the creation of a symptoms' database. This means that even though this computation is a great part of creating value for the team, it is something that is solvable and does not make the other parts of the configuration redundant. This is why they are certain the team can persevere and not need a full pivot and an overhaul of the configuration table.

A solution that was mentioned as being viable for this problem, was to determine the frequency at which a device records at, so that the module can take the frequency level into consideration when computing the respiratory rate. Another solution that they consider is by changing how the respiratory rate is computed completely. In order to determine which solution could be most viable, they perform a LCRT analysis of both to be able to compare in terms of what gives the team most value with regard to costs. The first solution, where they determine the frequency and compute based on that, gives the most leverage with regard to the costs due to the fact, that it does not require them to change the entire model, and they know that the computation of respiratory rate is correct for the audio inputs within the correct frequency.

In order to see how this solution can also be leveraged with regard to other aspects, they complete the VIT for a solution, where they answer the three questions, as seen in Figure

6.3c, and complete the action points. The VIT, as well as helping a team argue for why a solution should be implemented and whether it is the best means possible, it also serves as documentation as to why this decision was made, and what possible leverage it could give the team further into the project, or potential other projects if the solution is broad enough.

This is an example of one day within a sprint. The number of problem/solution meetings is dependent on what the team determines to be the most suitable given their current configuration. It could be imagined that some iterations could be more challenging than others and therefore there are many problems that can be risen at stand-up meetings.

Having multiple problems per day, every day that require pursuing would make conducting a problem/solution meeting heavy and time-consuming, however I do not see this as a fault of the tool. If so many problems are arising, it could be a sign for the project that this iteration is not viable, and an RST Review should be called. This means that they can update their configuration table in such a way that makes it possible for them to complete a sprint in a satisfactory manner.

6.1.3 Findings

If we follow the sequence of events in Figure 6.2, instead of addressing them separately, I discuss them as one process, as there are aspects of all three meetings that need to be addressed in order to work cohesively.

The stand-up meeting and problem/solution handling meetings are for software development teams that wish to have a process in the daily development that reflects innovation and project relevance. This is why the focus is on how to explore problems and how to leverage potential solutions within the confines of the project. However, due to the nature of the process there are certain aspects that are up to the individual development team. In this section I mention several aspects that I leave to the individual team as to how they best see fit to manage.

Sprint pivot or sprint persevere The choice of whether to pivot or persevere is based on whether a problem or solution is great enough that it requires an overhaul of the configuration table. Due to the fact that I imagine a problem/solution handling meeting does not include all members of a team, I suggest that the pivot is seen as an indication. This indication can then lead to thee participants of the meeting making the team aware that this problem is greater than initially anticipated, and that the sprint should end earlier than planned.

Seeing the choice of whether to pivot or persevere as a guideline rather than a rule could also be beneficial. It is a big decision to prematurely end a sprint that the team has spent time on planning. Therefore, I imagine that if a problem leads to a team most likely having to pivot, they might be hesitant to actually want to put it into practice and see if they have other options. A decision on whether to pivot should include the entire team, but the decision should be based on a well-reasoned foundation and not whether it sounds difficult.

Time Management The discussion of how a problem or a solution should be investigated does not include a detailed description of how long each task should take. This is again for the individual software team to decide. How a team wish to manage their time and determine how long a task will take, is dependent on a team's composition and experience, and therefore not something the process I propose takes into account.

The action points that I include in my proposal also include a "when should it be investigated by?". This question is based on the assumption that an action point should have a description of a task, an owner and a date. Letting a subset of a group choose a tentative date could be problematic, if they are inexperienced and not aware of how long certain tasks can take,

Discard: Is the problem already solved?	**Pursue:** Does this problem deserve more attention?	**Queue:** Is this problem relevant for current configuration?

Problem	***Action***
Not all in-built microphones record using the same frequency	Pursue
Cannot access user ID	Discard
There are more ways of determining asthma attacks	Queue

(a) An example of the DSM VIT proposal

The Problem	***Does this affect our overall problem to solve?*** If the microphone inputs read differently, it is not possible to give a user accurate warnings of a potential asthma attack.	***Does this affect what we are designing and implementing?*** If the output of the audio input is incorrect, we cannot compute the correct respiratory rate.	***Does this affect why our contribution is a solution?*** Not being able to compute the correct respiratory rate removes value from our product.	***Does this change what criteria affects our solution?*** We cannot see if potential warnings occur accurately enough to help reduce the effects of an attack.
Action Points	***What should be solved?*** The frequency being recorded at, should not affect the overall respiratory rate.	***Who is responsible?*** Developers 1, 2, and 3.	***When should it be investigated by?*** Until the end of the sprint, where if it is not solved, it can be taken up for review.	

(b) An example of a problem-focused problem/solution handling meeting as a VIT

The proposed solution	***What value does this solution create?*** It allows us to accurately compute respiratory rate no matter the device being used, this in turn lets us accurately produce warnings of an oncoming asthma attack.	***What opportunities does this lead to?*** Being able to determine the frequency of any mobile phone that uses the application means we do not need to update a list over the frequencies that they use whenever a new phone. The frequency levelling could also be used for other recording devices, it could also help processing noise reduction	***Are there already solutions that do this?*** Mobile phone companies already list the specifications of the phone's recording equipment, is there a library that gets updated with this, that we could use?
Action Points	***What should be solved?*** The frequency being recorded at should not affect the overall respiratory rate.	***Who is responsible?*** Developers 1, 2, and 3.	***When should it be investigated by?*** Until the end of the sprint, where if it is not solved, it can be taken up for review.

(c) An example of a solution-focused problem/solution handling meeting as a VIT

but since they are followed up on at stand-up meetings, I do not see this as a fundamental problem. It should more be seen as a guideline. The fact that it is also written down in the VIT, can aid a more senior developer who is responsible for the team in seeing how the project is progressing.

Number of Tasks If everyone has a problem that needs to be pursued, a sprint would become heavy in problem handling meetings. It is up to the team to decide how to make sure that they are all addressed and how they wish to manage this. If a team consists of ten people, and they all raise a problem, that would be ten tasks that would have to be completed. So it does not scale well. However, if there are ten problems being raised each day in a sprint, the team might have to revise what a sprint should include at this point in time.

Another aspect that could lead to an increased number of tasks are that problems can be related. Some problems are a part of a greater problem. I imagine that this could be discovered by seeing the affected items in a configuration table when investigating a problem. If many of the same items are being affected, it could be that there is a greater problem to be resolved, so that these smaller problems are not as prevalent during the rest of the sprint.

Participants Who is present at the focused meetings should be up to the team and what they feel most comfortable with. The guidelines for a DSM suggest conducting a meeting using a round-robin approach, where everyone gets a say. This can become time-consuming the more participants that are present. Stray et al. state that a team can circulate a facilitator role, as this helps the communication within a team, as they are more likely to communicate with one another than just reporting to the facilitator [2].

Frequency Stray et al. state that a daily stand-up meeting should be held regularly, but the frequency is dependent on the phase a team are in, as well as the need and level of informal communication between a team [5]. The frequency that is stated is in the name "daily stand-up", but what is necessary for some teams is not certain, nor is it certain that a problem/solution handling meeting is necessary everyday.

The aim of my contribution to Essence was to promote joint inquiry during sprints, not that it should occur every day. Due to the fact that teams are aware that these meetings are held, could also lead to increased inquiry on a daily basis, as the members of the team are looking out for potentials.

Timing of Meetings The sequence of events in a sprint states that the problem/solution handling happens after a stand-up meeting. However, when the problem/solution meeting is held is up to the development team and what suits their working environment best. Stray et al. mention that before the team have a lunch break is most beneficial for a team in order to not find the meetings disruptive. The time of day a meeting is held is not within the scope of the project, but it is still relevant in the sense that a team should feel that it is beneficial to conduct such meetings, and resentment due to feelings of disruption of the workday would be counterintuitive.

Chapter 7

Discussion

I have formed a process that is used to promote inquiry during sprints. This project is centred around providing a means of problem solving during the development of software products. This process consists of two separate meetings. One, to raise and draw attention to a problem, and the other, to explore its scope in full. Due to the fact that many software development processes utilise a type of daily stand-up meeting (DSM), this was also the form I felt would be most natural for a team to implement in their work process.

A study on what a good DSM consists of showed that discussing problems was seen as one of the more beneficial parts of a DSM [5], which I have also discussed in chapter 2. In order to incorporate the positive aspects of a DSM, I present a meeting that focuses on what is currently being looked at, rather than what has already been done. The output of such a meeting is a list of problems that can either be resolved immediately, queued to be investigated in a different sprint, or be explored further in a separate meeting dedicated to that problem.

The meeting dedicated to handling problems has two starting points based on input from the stand-up meeting. A team can either start by exploring a problem, or by exploring a potential solution. Each starting point has its own process that allows a team to discuss an item with regard to the rest of their project in order to determine its relevance and reach.

In this chapter, I discuss formalising problems during software development for software teams, and finally how I utilise the values in Essence in order to be able to formalise problem solving.

7.1 Formalisation of Problems

The purpose of this project was to promote joint inquiry during sprints. The way in which I attempt this, is by utilising a popular means of daily review in the form of a stand-up meeting. The daily stand-up meeting is where team members can raise problems and discuss their progress. Once problems are identified, inquiry can begin and the scope of the problems can be explored.

During the initial phase of my project, I attempted to create one meeting that could handle both identifying problems and exploring them in order to promote joint inquiry during sprints. However, once I explored what this would actually require of a software development team, it was apparent that to be able to fulfil the guidelines for daily stand-up meeting's good practice, certain aspects of what I deem important when exploring problems would make the DSM cumbersome instead of helpful. This is ultimately why raising and exploring problems have two separate activities. Because of this, the visual inquiry tool (VIT) that was created also needed to have two separate aims. Under

the principle of parsimony, Avdiji et al. state that "designers can merge some components into higher-order components. If subcomponents are deemed important, they can be used to develop additional tools [4]." This was also another reason for separating the two tasks, and instead we have a VIT for a DSM to support what problems would be tackled in the next VIT, which is for handling problems and exploring their scope.

The purpose of the stand-up meeting is in line with what Stray et al. state an efficient DSM should be, where we obtain a shared understanding of the current activities happening in the sprint. I have attempted to follow what the potential benefits of such a meeting could be by having a means of determining whether a problem has relevance now or not.

The study of daily stand-up meetings in practice showed that many team members were dissatisfied with their meetings due to the fact that not all information relayed at such meetings was relevant to them [5]. By splitting the meeting into two, where one is for raising problems and the other for handling them, the entire team does not spend time on looking at how to solve them, but more on whether it is relevant to look at now. This means that the problems that are raised at a DSM can then be handled by the participants for whom the problem has relevance. In order for this to work, it requires that each participant who wishes to present a problem that they are experiencing, formalises it, in such a way that makes it easy to determine whether it should be acted on in this sprint or not. This helps keep the duration of the meeting as brief as possible, as is stated in Stray et al.'s guidelines. However, if participants have difficulty stating the problems they are facing, problems with great risk can go unseen.

The format of these meetings is supposedly best when the participants have a physical board to stand in front of. I have attempted to do this via creating a VIT that gives the meetings a structure in which they can formalise their problems either in terms of category in the DSM, or in terms of the problem's scope in a problem/solution handling meeting. The other characteristics, such as turn-taking and time of day that are listed in their guidelines are not something that I take into consideration in my project, and are up to the individual software development teams. The reason why I do not look into it, is because these characteristics are not something that affects the contents of a meeting, but are related to what suits the individual needs of a team best.

7.2 Pragmatism in Essence

In the Essence methodology, the main aspect of reflection in development are the RST reviews. RST Reviews are supposed to ensure that teams are on the right path. The RST Review is a planned event that occurs at the conclusion of a sprint, where the team reviews the work that has been done and whether their assumptions about the project are still relevant given the environment. The contents of the sprint are based on the decisions that were made during the RST Review. However, as perceptions can change during a sprint, and problems arise as development occurs, it does not seem enough to only have the RST Review as a form of reflection, there should also be a means of keeping a project on track during a sprint. A sprint is not a defined length of time and changes from team to team and therefore an RST Review can be quite far away from where a team is in their sprint. On the other hand if a sprint is one week long, it might not be necessary for a team to require both forms of discussion of a project and can settle for one method or the other.

Problems are identified during development and are raised at the meetings. The team can then begin to explore what it means for the overall project. When subject to inquiry in this way, it can become apparent that the problem is more far-reaching than first anticipated. If a team experiences that problems are not resolved when solutions are discovered, but grow in size, it could indicate that there is something fundamentally wrong with the sprint. Should this be the case, the sprint can be prematurely ended so that a complete RST review can occur instead of at a fixed point in time

which can reduce time waste. I discovered during development that problems are not singular and can lead to discovering new problems. This is due to the fact that our understanding of a problem increases when subject to inquiry and gives us new insights. If problems keep being discovered despite attempts to mitigate them, it could mean that a sprint is not stable, and should also result in ending the sprint and conducting an RST Review.

The process that I present as a means of promoting joint inquiry during sprints is supposed to complement Essence and its values. The four values in Essence encourage the use of inquiry by looking at a problem and exploring its scope, considering opportunities, determining a goal, and finally constantly evaluating the relevance of a solution. The process helps mitigate that requirements are not fully known when projects start, nothing is set in stone, and therefore development can be adapted depending on the changes in our environment.

In chapter 2, I listed and described the four values of Essence. The ways in which I incorporate these values in this project are as follows.

Reflection This value is for continuously looking at the relevance of work by having a deeper understanding of the problem and its context [1]. This project allows team members to raise and identify problems that occur during a sprint and explore the scope. This is done by having members of a software development team raise problems at a stand-up meeting, and allows the rest of the team to decide whether the problem should be explored within this sprint or not. When the problem is handled, we also require that we reflect the core concern *situation* by making sure that what is created has relevance to the situation and problem.

Transaction This value is to understand that one is constantly in transaction with the environment as the project develops. When problems are explored, a team is helped to find and leverage unused potential. This is encouraged by members of the software team having to explore the scope of solutions and discussing what else a specific solution could be used for. Once a problem or solution has been raised at a stand-up meeting, the team conducts a meeting to handle it. By using effectuation and asking a team questions relating to its principles, inquiry is promoted via asking a team to consider what opportunities certain solutions can offer and whether there are aspects we have not yet seen with regard to the problem.

Reasoning This value is to provide a foundation for a project that states the vision of what it wants to achieve. A development team is working towards a goal that is subject to change. This value is reflected in this project by having to discuss problems as they occur and takes a changing environment into consideration, as the experiences of the developers also lead to a change of the project's understanding, which in turn can change our overall vision for the project. The process I present states that a development team should conduct a stand-up meeting with the frequency that suits their needs and that the problems raised at these meetings can then be explored in a follow-up meeting. Due to the fact that these meetings occur during a sprint, and therefore occur alongside development, the real-time issues being faced reflect the environment. There is a focus on finding problems that can create change in the direction of a project, a focus on discussing problems as a team and focusing on creating relevant solutions with regard to the current environment.

Appreciation This value is used to determine whether we are heading towards solving a problem, using sound reasoning. I believe that my contribution does this in that a software team is pushed to constantly evaluate the current solution by exploring problems that arise during sprints through raising problems at a stand-up meeting and then exploring them in detail at a problem handling meeting. Due to the fact that the software development teams are encouraged to evaluate their current work, they are therefore also evaluating the viability of a project and through the framework of meetings proposed has a structured way of ending a sprint if it is not viable.

The author of Essence states that it should be ”light and useful” and ”not demand too much from the team”. This is why the process I have created should also be light in order to reflect this aspect of Essence. It is also stated that ”the core to Essence is that teams continuously arrive at new insights, refine and prune ideas and evaluate their potential.” [1, p.xvii] The process that I establish does this by encouraging teams to identify problems during their daily work and assess what these problems could mean for development. This process might not be the right fit for every team and, as such, they can tailor the overarching principles to suit their needs. It requires that a team is capable of planning and organising their work in order to utilise the process fully.

The process I present in this report focuses on the present and the impediments that a team is currently facing and not the end goal. I am not sure whether it would make sense to keep the overall goal in mind when discussing current impediments, as the overall goal can also be subject to change as our understanding evolves.

An aspect of Essence that I do not address is the use of roles and what this means during development. The Essence methodology has four roles. The role that I consider in my project is the responders, which are the developers. Another role that could have made sense to incorporate to a degree would be the challenger, this is the team's customer who has knowledge of the problem domain. Currently, the focus of this process is very internal, and only considers what the developers feel give most value. I do see an opportunity for teams to be able to include a challenger if they see fit in the process I establish, however, the challenger still has their place at the RST Review where the entire project is discussed and not current problems with regard to development. It could also be argued that a challenger does not need to be present for such items, and if they are needed for their domain expertise, having a formal meeting to address that specific problem could still occur. They are just not needed at a stand-up meeting for raising the problem. Otherwise, I assume that any issue a challenger has would come to light in an RST Review.

7.2.1 An Accompaniment to a Review

Due to the fact that this process asks a development team to continuously look at their progress critically, it could make a review activity, such as the RST Review seem less daunting. This would be positive for many software teams as it can make the RST Review process quicker as the members are aware of changes that can have been made along the way, but also are aware of how problems have been managed. Overall, the process gives a greater overview of a project and its intricacies, making each software developer more aware of the work being created and the effort it takes to implement and deploy it.

7.3 Limitations

The design theory for visual inquiry tools has been used in order to give my process a structure and help ensure that inquiry was part of the focus. The focus of this project was not, however, to create a visual inquiry tool. The goal was to promote joint inquiry during development. This is why the VIT has been a way to facilitate the ideas.

Fulfilling the design principles from the design theory for a VIT, does not necessarily make the tool one that is good at what it is meant to do. This is why even though technically the principles are addressed, there is still room for the problem-handling tool to be refined and explored further.

For a process that is supposed to be light, it can seem excessive to use the design theory, however, following the principles helped to focus on inquiry and assessing what the results of this inquiry mean for a project. But before developing a fully fledged VIT for this process, I would first need to be certain that this process would provide value for software development teams. The VITs

presented in this report are prototypes of how it could occur.

The lack of evaluation makes it difficult to conclude whether this kind of process would be beneficial for those who utilise Essence. The entire project is built on my own experiences with software development as well as my own knowledge of using Essence. In order to assess whether this process is something software teams would find useful, it would require formal evaluation, where I would expect teams to try out the process during several sprints to see what feedback it could give. Due to the fact that Essence is taught at Aalborg University through mini-projects, it could be interesting to see how my contribution is received in semester projects and whether it is deemed useful here.

Despite these limitations, the project is a contribution that is a proposal for a method to increase joint inquiry during sprints based on established theories and methods.

Chapter 8

Conclusion

This project began as looking for a means of promoting joint inquiry during sprints in Essence. I then used the following questions in order to analyse the domain:

1. What processes exist for encouraging joint inquiry within a development team?
2. Does Essence support a means for encouraging inquiry on a daily basis?

I discovered that Essence does not currently have a way of encouraging inquiry on a daily basis, but there are processes available that could be utilised by Essence in order to do this. These were a form of daily stand-up meeting that involves all members of a development team and by using the design theory for visual inquiry tools allowed me to create a model that could facilitate the contents of such meetings.

The analysis of the domain led me to the problem statement:

> How can joint inquiry become a useful part of the daily activities in Essence?

Due to the fact that daily meetings are typically the way in which team members interact with each other to understand how the project is progressing, I decided to explore what opportunities for joint inquiry there could be here. This is the reason for the focus at the beginning of the project being a DSM and what that would look like using Essence.

Through the project, I determined that a DSM alone would not be enough in order to address problems in the scope that I deem as being adequate when trying to find potential solutions and determining the size of the problems. These meetings are connected via either a problem or solution that a team wants to investigate further. The overall process starts with having a configuration table with backlog items, where a team then has a means of keeping their project in check, where at the end of the sprint they perform their RST review.

It is uncertain how such a process that promotes joint inquiry within sprints would be received by actual software development teams, given that such processes that already exist like the DSM, are already subject to much critique. This is not focused on in this project. However, the studies of DSM showed that many participants wanted problem solving to be on the agenda, which is something I have attempted to take into consideration.

Looking at how DSMs are received by software development teams and applying the design theory for VITs has helped me create a process that should encourage joint inquiry within sprints and even on a daily basis when necessary. This in itself should lead to teams feeling that they are more

aware of the development that is happening around them, as well as making a project feel more stable, and that the team is in control of the direction of the project. However, due to the lack of evaluation in this project it is not possible to say how the process would be utilised in actual software development projects, and what the teams here deem as useful or beneficial and what is a hindrance.

The process that I present in this report shows a way a development team can communicate problems in their daily work to the rest of the team in a directed, constructive way that results in a deeper understanding of the problem and therefore an understanding of the stability of their project and more robust software products as the outcome.

8.1 Future Work

This project is one look at how software development teams can promote joint inquiry in the daily development of their product. Given the fact that a process to help developers should feel easy to use, a balance between strict governance and freedom to work undisturbed should be obtained. This is why the first part of future work would be to conduct evaluations of the current process.

Evaluation In future iterations of the development of this process, I imagine that other means of looking at problems could be beneficial for teams finding the process that works best for them. Therefore, any future work would require evaluation with multiple teams in different projects to get a wide span of potential cases and what they find useful with regard to promoting inquiry during development.

The RST Review Currently, there is no greater connection between a daily meeting and the RST Review other than the fact that it is used to review progress and discover potentials. Given the fact that the VIT for both meetings are a way of documenting problems that are discovered, these could also have use in the RST Review as a type of backlog of items to go through as a part of the RST review process. Creating this connection could make it easier to perform RST reviews, so there would be lesser redundancy if aspects of the configuration table are already picked up during development.

Backlog Items To give the daily meetings more clarity of what specific items there are, it could make sense to highlight these from the configuration table and explicitly state what they are. This way the connection from item to be handled and the problem that is being experienced is transparent. This list of items could also be beneficial to the final review of the sprint, the RST Review.

Bibliography

[1] Ivan Aaen. *Essence - Problem Based Digital Innovation*. 2020.

[2] Viktoria Stray, Nils Brede Moe, and Dag I.K. Sjoberg. "Daily Stand-Up Meetings: Start Breaking the Rules". In: *IEEE Software* 37.3 (2020), pp. 70–77. DOI: 10.1109/MS.2018.2875988.

[3] John Dewey. *Logic : The Theory of Inquiry*. Hentry Holt and Company, inc, 1938.

[4] Hazbi Avdiji et al. "A Design Theory for Visual Inquiry Tools". In: *Journal of the Association for Information Systems* 21 (May 2020), pp. 695–734. DOI: 10.17705/1jais.00617.

[5] Viktoria Stray, Dag I.K. Sjøberg, and Tore Dybå. "The daily stand-up meeting: A grounded theory study". In: *Journal of Systems and Software* 114 (2016), pp. 101–124. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2016.01.004. URL: https://www.sciencedirect.com/science/article/pii/S0164121216000066.

[6] Eliza Bobek and Barbara Tversky. "Creating visual explanations improves learning". In: *Cognitive Research: Principles and Implications* 1 (Dec. 2016). DOI: 10.1186/s41235-016-0031-6.

[7] Alexander Osterwalder and Yves Pigneur. *The Business Model Generation*. John Wiley Sons, Inc., 2010. ISBN: 978-0470-087641-1.

[8] *Strategyzer*. URL: https://www.strategyzer.com/canvas (visited on 12/19/2022).

[9] Ivan Aaen. *Essence - Problem Based Digital Innovation*. 2023.

[10] Donald A Schön. *The reflective practitioner: How professionals think in action*. Routledge, 2017.

[11] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. URL: https://scrumguides.org/scrum-guide.html#daily-scrum (visited on 11/21/2022).

[12] Cambridge Dictionary. *action point*. URL: https://dictionary.cambridge.org/dictionary/english/action-point (visited on 12/06/2022).

[13] Saras Sarasvathy. "Entrepreneurship as a Science of the Artificial". In: *Journal of Economic Psychology* 24 (Apr. 2003), pp. 203–220. DOI: 10.1016/S0167-4870(02)00203-9.

[14] Saras D. Sarasvathy. "Causation and Effectuation: Toward a Theoretical Shift from Economic Inevitability to Entrepreneurial Contingency". In: *The Academy of Management Review* 26.2 (2001), pp. 243–263. ISSN: 03637425.

[15] E. Hippel and G. Krogh. "Identifying Viable "Need-Solution Pairs": Problem Solving Without Problem Formulation". In: 27 (Jan. 2016), pp. 207–221. DOI: 10.1287/orsc.2015.1023.