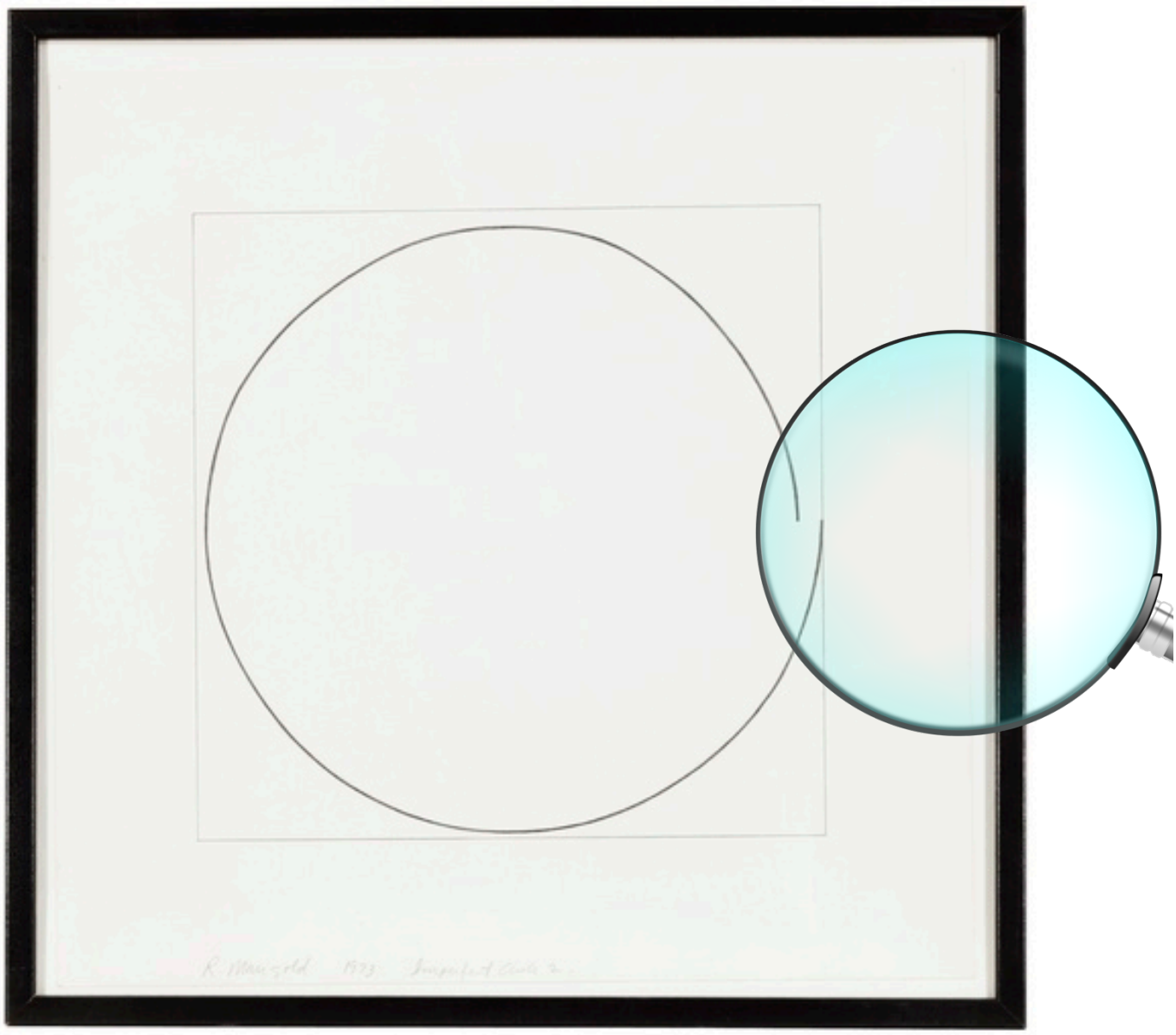
CLOSING THE OPEN LOOP

A method for aligning software with customer value



Date: 28/11/2017

Final Thesis

Student: Lody Koop

First supervisor: Fabiano Dalpiaz

Second supervisor: Sjaak Brinkkemper

ABSTRACT

Aligning a software product with the end user's wishes is an important factor that determines a software producing company's success. Shortcomings of current methods that attempt to align user value with software products by using research and development as an experiment system triggers the need for a method that truly aligns the software product with the wishes of the end user/customer. In this thesis, the "Learning as an Organization to Optimize the Product" (LOOP) method is proposed. The design science framework is applied to create the LOOP method, which is subsequently evaluated by means of a case study. One iteration of the method is executed in this case study. The evaluation of this case study indicates that the LOOP method is effective in providing the information needed to align a software product with the end user's/customer's values on a short-term period, and most likely, also actuates learning to make better product decisions on the long-term. The evaluation on feasibility concerning the LOOP method indicates that the method is scalable but is possibly not suited for companies with limited resources or a small user base. A longitudinal study to validate the LOOP method's intended effects over multiple iterations on the long term and in different contexts is suggested as a future study.

WORD OF GRATITUDE

During the past months, I have often fantasized about this moment. And now I sit here, at this very moment, writing my word of gratitude.

Firstly, I want to show my gratitude to the people who have supported and helped me in the period of this thesis. I thank Koen Swart and Symen Wahle for being so resourceful, enthusiastic and cooperative during my case study. I thank the gentlemen from YNC for helping me initiate my thesis project, and for always being available to help (preferably if you could provide me with your critical opinions). I thank my second supervisor Sjaak Brinkkemper for his enthusiasm and willingness to help me with my study and career. I want to deeply thank my first supervisor Fabiano Dalpiaz for always being willing to help, answering my mails so fast, providing me with feedback when I last minute sent work, giving me guidance when I thought I hit a dead end, and mostly for (unconsciously) conveying his relaxedness to me when I felt this typical thesis despair.

Secondly, since I conclude my master study with this thesis, I want to show my gratitude to the people who made my master study an amazing experience. I thank the lecturers Sergio Espana Cubillo and Sietse Overbeek for the pleasant lectures and support they provided during my study. I thank my friend Garm Lucassen, for providing his wisdoms when I asked him all these study and non-study related questions outside office hours. I deeply thank my allies Roland Wondolleck and Jan Kramer for acing all these assignments. Especially Roland, who, without self-interest, supported me by peer-reviewing this thesis.

Thirdly, since I also conclude my role as a scholar with this thesis. I want to show my gratitude to some very special people who have supported me during this epoch. Joyce, little buddy, I thank you for all the (unasked) advice, food, fun, and mental support you gave me. My special thanks to Nick, Tom, Rens, Sjors, Bram and Mo who have made life fun outside my life as a scholar.

Finally, my deepest and sincerest gratitude goes to my guardian angels and parents Carla and Herman, who unconditionally supported and loved me, and with times, forcefully pushed me back in line.

TABLE OF CONTENTS

Abstract 2

Word of gratitude 3

Table of Contents 4

1 Introduction 8

2 Research approach 9

2.1 Problem statement 9

2.1.1 Beyond the current R&D methods 11

2.2 Method objectives 12

2.2.1 Feasible 12

2.2.2 Effective 12

2.2.3 Beneficial 13

2.3 Research question and subquestions 14

2.4 Outline of and justification for the chosen research method 15

2.5 Literature research protocol 17

3 Theoretical foundation 18

3.1 Methods to form R&D as an experimenting system 18

3.1.1 The RIGHT model 18

3.1.2 The HYPEX model 18

3.1.3 The QCD model 20

3.1.4 EVAP 21

3.1.5 DVOCE 22

3.2 Methods comparison 23

3.3 A high-level overview of the primal solution method 24

3.4 Map customer value to requirements 26

3.5 Feature oriented development 30

3.5.1 FODA 30

3.5.2 FORM 30

3.5.3 FeatuRSEB 31

3.5.4 FArM 31

3.6 Verification and Validation 32

4 Final method iteration 34

4.1 LOOP Phase 1: Feature selection 35

4.1.1 Indicate suitable features for the LOOP method 37
4.1.2 Gather stakeholder assumptions on features 39
4.1.3 Perform market analysis 39
4.1.4 Divide users in different segments *(Optionally)* 39
4.1.5 Develop a main feature tree model for the product *(optionally)* 40
4.1.6 Add a goal to every optional feature 41
4.1.7 Add a hypothesis (per customer segment) to every feature 41
4.1.8 Decide on which product configurations are going to be (partially) implemented and tested 45
LOOP 47
4.2 Phase 2: implementation of features and data collection mechanisms 47
4.2.1 Ask questions about hypotheses and goals 49
4.2.2 State metrics per feature that can answer the questions 49
4.2.3 Select data collection mechanisms for answering the questions 51
4.2.4 State the initial situation before the feature is exposed to the user 52
4.2.5 State the desired post-exposure situation of the feature 53
4.2.6 Implement features with their data collection mechanisms 53
4.3 LOOP Phase 3: Verification and Validation 55
5 LOOP Case study 58
5.1 Company 58
5.2 Application LOOP Phase 1 58
5.2.1 Indicate suitable features for the method 58
5.2.2 Gather stakeholder opinions on features 59
5.2.3 Divide users in different segments 59
5.2.4 Develop the feature tree 60
5.2.5 Add a feature goal to every feature 61
5.2.6 Add hypotheses 62
5.2.7 Decide on which product configurations are going to be (partially) implemented 62
5.3 Application LOOP Phase 2 63
5.3.1 Ask questions about the hypotheses and goals 63
5.3.2 Metrics 65
5.3.3 Select additional data collection mechanisms for verification and validation 66
5.3.4 State the initial situation before the feature is exposed to the user 69
5.3.5 State the desired post-exposure situation 69
5.3.6 Implement features with their data collection mechanisms 69
5.4 Application LOOP phase 3 71
5.4.1 Conclude whether the goal is met in terms of functionality (verification) 71
5.4.2 Compare realized situation with initial situation and intended situation in terms of user impact 71

5.4.3 Conclude whether the hypotheses are likely to be valid (validation) 72
5.4.4 Document findings and decide on the feature's future lifecycle 73
5.4.5 Learn from wrong assumptions and reasoning 73
6 Case study Evaluation 75
6.1 Observations and evaluation 75
6.2 Interviews 76
6.2.1 Questions 76
6.3 interview evaluation 77
6.3.1 Interview 1 80
6.3.2 Interview 2 84
6.3.3 interpretation 90
7 Conclusion 94
7.1 Discussion on Research questions 94
7.2 Threats to validity and limitations 96
7.2.1 Internal validity 96
7.2.2 External validity 97
7.2.3 Reliability 98
7.3 Future work 98
8 Bibliography 99
Appendix A: First method iteration 102
Background information 102
Prioritize features 104
Model Feature Goal 104
Set Hypotheses 104
Question 105
Define Implementation and Set Metrics. 106
Choose Data Collection Technique 107
Implement MVF 107
Collect Data 107
Validate Feature Implementation 107
Validate Hypotheses 108
Classify Feature 108

1 INTRODUCTION

We live in an exciting new era where on-demand software products are offered as a service to the customer. Since software producing companies stay committed to their delivered software products, such companies never were so close to potentially knowing their customers and users, and with this, closing the open loop between product decisions and customer feedback (Olsson & Bosch, 2014b).

Different approaches have been proposed to close this open loop by using "research and development" as an experiment system (Fagerholm, Guinea, Mäenpää, & Münch, 2017; Olsson & Bosch, 2014b; Olsson & Bosch, 2015; Fabijan, Olsson, & Bosch, 2015; Ekström & Þorvaldsson, 2016) This thesis contends the existence of two major shortcomings in the methods from the literature. Firstly, these methods do not truly align the software product with what the user values and helps in understanding this relationship. The current methods are mainly focused on testing a high number of features in a short time, and collecting large amounts of data. In these methods, there is the risk to blindly following patterns discovered in the obtained data. The discovery of these patterns in data, relating feature behavior with the user environment is valuable, but does not aid the company in understanding the user and customer better. We theorize that the reasons between these patterns should be sought and analyzed. Secondly, the methods are proposed on a high level of abstraction and are therefore difficult to implement.

The aim of this thesis is to define an actionable method that helps in maximizing the customer and user value delivered by chosen software features while using R&D as an experiment system. Our method aims to close the open loop between the customer/ end-user of a software product and the software company. To do this, the method should be actionable. Also, when executing the method, the company should learn to understand the value impact of features on the customer/end-user.

The main research question studied within this thesis arises from this stated objective, and is defined as: *"What is an actionable method to maximize the customer and user value delivered by the chosen software features, by using R&D as an experiment system?"* The main objectives for this method are that it should be *feasible*, *effective* and *beneficial*. Feasible and beneficial in order to make sure that it is actionable. Whereby beneficial refers to the positive ratio between actuated benefits from executing the method and incidental costs. Effective in the sense that its intended effects truly occur and therefore help in maximizing customer and user value by choosing the right set of features for a software product.

In this thesis, we follow the design science research paradigm as proposed by Wieringa (2014) to create an artifact in the form of a method that fulfils the aforementioned objectives. The method is validated by means of a case study at a software producing company. The data is attained by observations and semi-structured interviews.

The research approach, problem statement, objectives and research questions are further elaborated in Chapter 2. The third chapter provides a review of the literature containing the analysis of similar methods. The final solution method is presented in Chapter 4. The application of the method in a real-life context by means of a case study is described in Chapter 5. The evaluation of the case study is elaborated in the sixth chapter. Finally, Chapter 7 presents the conclusion of the thesis by answering the stated research questions.

2 RESEARCH APPROACH

This chapter begins with stating the problems found in current literature. The subsequent section elaborates upon the objectives that the solution to stated problems should meet. Section 2.3 defines the research questions that arose from the solution's objectives. In Section 2.4 and 2.5 the research approaches used for answering this thesis project's research questions are described.

2.1 PROBLEM STATEMENT

In order for a software producing company to be successful it needs to adhere to its customers wishes. If the company does not adhere to customer demands, the KPIs of the company will be negatively affected. Therefore, correctly prioritizing features that translate customer desires into functionalities for new releases is one of the key activities to stay successful as a software company. Many companies however, prioritize features based on the opinion and experience of product-, project- and senior managers. The risk of applying this approach could translate into a product that starts deviating from the customer's wishes and in the end, will not be bought anymore.

Olsson and Bosch (2014a) argue that software companies evolve their development process over time. The evolution of this process is called: "Stairway to Heaven". The Stairway to Heaven consists of different stages (Figure 1). The initial stage, on which most software companies start is called "Traditional Development". The final stage as argued by Olsson and Bosch is called: R&D as an experiment system. The software company proceeds in stages when the software development method of the company evolves. The phases of the Stairway to Heaven will be elaborated in the subsequent sections.

The Traditional Development stage is characterized by rigorous upfront elicitation of requirements and planning of development activities. Development methods that hold these characteristics are often referred to as "stage wise" or "waterfall" models (Boehm, 1988). These methods are suitable in some software projects where all requirements are clear and solid. However, in most projects, the drawback of these types of methods is that requirements can change during the long development phases and therefore the developed product shifts from the customers' wishes. Furthermore, traditional development tends to extensively emphasize documentation of development specifications. These documents are often cryptic by nature and lead to large quantities of unusable code.

A software company evolves to the second stage when it implements agile practices within the R&D department. The agile manifesto was formed in order to address the drawbacks from traditional development activities (Fowler & Highsmith, 2001). The agile manifesto holds four pillars:

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

The benefits of adhering to these principles is that a software company receives more frequent customer feedback concerning requirements. Furthermore, less resources are wasted on extensive documentation and development of requirements which are not adding any customer value.

The third stage is called "continuous integration". According to Humble and Farley (2010) continuous delivery is a practice whereby the development team members frequently integrate their code,

resulting in multiple integrations per day. Both the development team and the test- and verification team work according to agile practices in this stage.

After continuous integration follows the "continuous deployment stage". The code which is continuously integrated and tested is also continuously deployed at the customer side. This is contrary to the release of several large builds, holding significant amounts of new functionalities., Since the company receives continuous customer feedback on new functionalities, the implementation of this stage further solidates the agile benefits. Therefore, changes can rapidly be made. This stage is not suitable for every product type; the product needs to be cloud based.

As mentioned before, the final step in the Stairway to Heaven model is the "R&D as an innovation system" stage. During this phase, the software company anticipates on instant customer data. The purpose of this phase is to expose customers to a piece of functionality, while at the same time recording their usage to see whether these specific sets of functionalities add value. The ultimate benefit of this continuous experimentation is that the prioritization of new requirements is not solely opinion-driven anymore, but also based on actual user data.

Due to the wide increase of internet speed, the adoption of cloud based solutions becomes more common. With this, the possibility for software companies to evolve to the "ultimate stage" of the Stairway to Heaven model is rising. In furtherance of the adoption of this stage several models have been proposed. The most relevant models will be explained in Section 3.1.

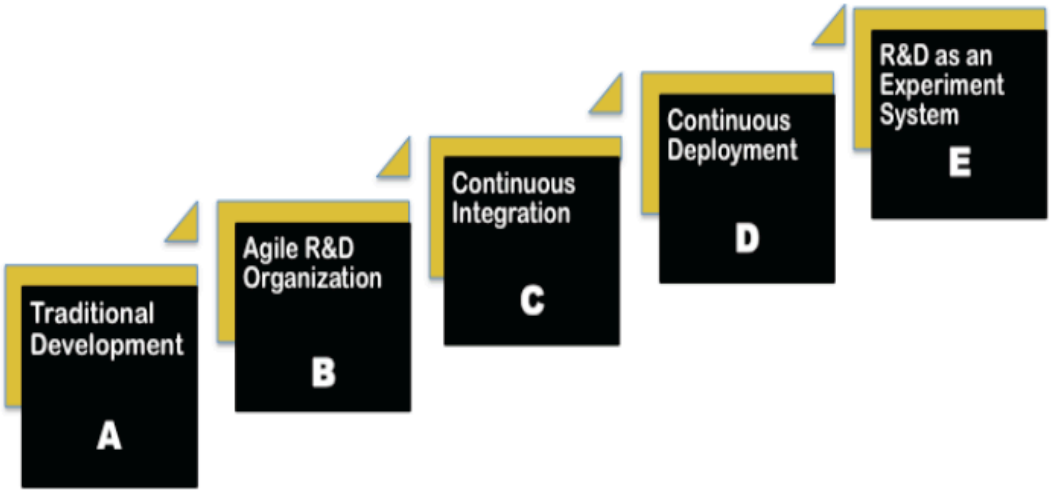


Figure 1. Stairway to heaven model (Olsson & Bosch, 2014a, p. 2).

2.1.1 BEYOND THE CURRENT R&D METHODS

Several methods are proposed using R&D as an experiment system (see Section 4.2). Although all stated methods are conceptually promising, they lack one, or both, of the following aspects:

All methods, except the DOVCE model, stay very high level and give no specific set of activities and techniques for application in industry. Although a certain level of abstraction is desired from a scientific viewpoint, the industry needs to have handles to realize the implementation of the method. Even when looking at the relative low-level DOVCE model questions such as *"Why and how do you estimate 10% of a feature's total functionality in order to implement a MVF"* arise.

Secondly, there is a conceptual flaw in all of these methods, as they do not fulfill their ultimate goal of measuring customer value by means of experimentation. In other words, the models pretend that features will be prioritized by looking at the value for the customer and therefore keep the software product attractive and lucrative. For example, Olsson and Bosch state: "*with insufficient mechanisms to continuously confirm customer value, there is the risk of lack of alignment of product and customer needs during the roadmapping of new software functionality*" (Olsson & Bosch, 2014b. p1). Of course, the proposed methods are valuable since decisions will be based on data from customer usage. However, the problem of opinion driven decision making is still present. The person responsible for prioritizing features still has to set a feature goal that is assumed to add customer value. Subsequently, the feature goal is measured and analyzed by means of the metrics set. The big conceptual flaw is that the feature goal is still opinion based. For example, the goal of a feature is to let the user have a faster login procedure. Metrics are set and evaluated. The feature meets its goal and is further developed. However, it is never measured whether the user really experiences an increased value with a faster login procedure. Maybe the login procedure is already fast enough and the user cannot be satisfied any further by this feature.

Concluding, the product can still lose the alignment with customer needs when the aforementioned methods are implemented.

2.2 METHOD OBJECTIVES

The goal of this thesis is to deliver a method that improves both aforementioned aspects relative to the current existing methods in literature. The purpose of this method is that it extends current methods by prescribing a procedure that makes sure that feature goals are also validated in terms of perceived customer value. Additionally, the proposed method needs to consist of a concrete set of activities and techniques resulting in easy adaptation by industry.

This thesis uses the Technology Acceptance Model (TAM) as proposed by Davis (1993), for two main reasons. TAM is originally intended by Davis as a model that is intended as an explanation of the determinants as the user acceptance of information systems. The purpose of TAM in this sense is further elaborated in Section 3.6. However, TAM has also been proven as a model that gives guidance in the acceptance of methods linked to information technology (Koc, Timm, Espana, Gonzalez, & Sandkuhl, 2016). TAM states that perceived ease of use and perceived usefulness will positively influence the behavioral intention to use an artifact (in this case the method). We derived the method's objectives from these two influences. The objective of feasibility is derived from perceived ease of use. The objectives of effectiveness and beneficially are derived from perceived usefulness. We assume that if the method can adhere to these objectives it will result in a behavioral intention to use it.

Following this path of reasoning, the following objectives for the intended method are drafted:

2.2.1 FEASIBLE

The method should be feasible for software producing companies. To guarantee the feasibility of the intended method, the following sub-objectives are formed:

- It should be possible to implement the method in a real-life context;
- There should be a well-balanced level of abstraction in the method's process description. Method steps should be specific enough to be actionable, but not too specific such that the method stays situational;
- It should be possible to align the method with an agile development environment;
- The execution of the method should utilize R&D as an experiment system.

2.2.2 EFFECTIVE

The method should have the following effects once it is implemented in a software company:

- Executing the method results in well-advised, data-driven, and short-term decision making concerning a product's feature portfolio;
- Executing the method actuates organizational learning concerning a product's feature portfolio and customer appreciation. This results in better understanding of the product, customer and user, leading to enhanced long-term decisions;
- The method relates features with actual customer value. Whereby customer value is defined as *"direct value perceived by the user of the product"*.

2.2.3 BENEFICIAL

The method can be feasible and effective, but the cost of execution should not outweigh the benefits.

2.3 RESEARCH QUESTION AND SUBQUESTIONS

This section presents the research questions which is a corollary of the aforementioned thesis goals.

The main research question is: *"What is an actionable method to maximize the customer value delivered by the chosen software features, by using R&D as an experiment system?"* This question captures both indicated aspects that are stated in the Section 2.1.1. The following sub-questions are derived from the main research question and related to the method's sub-objectives that are intended to make the method effective:

- *"What is a suitable technique to capture and model the link between perceived customer value and a feature?"*

  One sub-objective of the intended method is to relate features with actual customer value. Answering this sub-question is necessary to meet this objective and evaluate whether features are truly meeting their intended goals. Once a technique is created, or found, that models the actual- and intended value obtained from a feature, it can be used to answer the following sub-questions and other sub-objectives categorized under *effective*.

- *"How can the right direction of feature implementation be determined, based on testing different variations of feature configurations"*

  By creating the right set of activities to test different product configurations one can attain knowledge about the right product configuration for the right user segment. Furthermore, features can be compared with each other leading to enhanced decision making concerning a feature's lifecycle. Answering this sub-question will help in reaching the sub-objective that aims to aid short-term decision making concerning a product's feature portfolio.

- *"How can organizational learning be actuated concerning product, feature, customer and user knowledge?"*

  Making the right decision concerning a product's feature portfolio is mainly beneficial in a short-term period. The reasons are that the direct benefits of the decision will only last as long as the product stays in the company's product portfolio and because the user's perception of a feature's value changes over time (Kano, Seraku, Takahashi & Tsjui, 1984). However, if the intended method can actuate organizational learning and the company can store this knowledge, it can be used to better understand the customer and user. This will lead to better future decisions and therefore is beneficial on a wider timescale. Answering this sub-question will help in the realization of sub-objective concerning the enhancement of long-term decisions.

2.4 OUTLINE OF AND JUSTIFICATION FOR THE CHOSEN RESEARCH METHOD

The chosen research method is Wieringa's design science method (2014) whereby the evaluation strategy is derived from the FEDS framework as proposed by Venable, Pries-Heje and Baskerville (2016). The aim of design science is to design an artifact that solves a problem for a certain set of stakeholder in a specific domain. The artifact will be designed and subsequently investigated. Design science is a suited research method, since the goal of this study is to create a method (artifact) that helps software companies to develop software products that are aligned with customer's wishes.

The FEDS framework directs on how and when to evaluate the artifact. Due to temporal constraints, FEDS' lightweight "*quick and simple"* strategy is chosen, which is low cost and leads to quick conclusions. The "*quick and simple*" strategy starts with a set of formative evaluations in an artificial context (the iterations between the treatment design and validation) and moves relative quickly to a summative evaluation in a naturalistic context (the conducted case study).

Wieringa's design science method is called the engineering cycle that consists of five phases (Figure 2). First the problem is investigated whereby stakeholders, goals, causes, mechanisms and reasons are taken into account. During the Treatment design phase, requirements are specified for the artifact together with conceptual treatments for the problem that satisfy these requirements. Once a treatment design is created it is validated to check whether the designed artifact will effectively satisfy the requirements. If this is not the case, the scientist repeats the treatment design phase and adjusts the artifact's design. Once the treatment design is validated, it is implemented in the problem context. Subsequently, the implementation is evaluated to see whether the artifact has resulted in the desired effects/goals within the problem context. The latter may be the start of a new iteration through the engineering cycle.

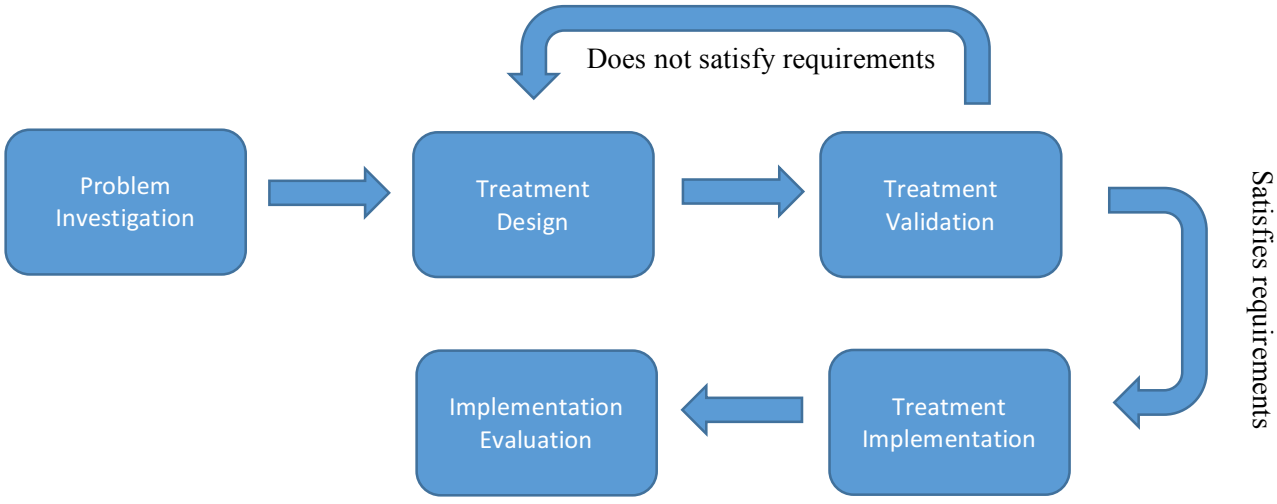


Figure 2. The engineering cycle (Wieringa, 2014).

The identified problem in this study is that, to the best of our knowledge, no method exists that guarantees product alignment with actual customer value. Additionally, only high-level practices are stated in the existing literature (Section 3.2). The stakeholders are software companies that aim to

keep a tight fit between the demands of the customer and the set of features the software product will offer. The artifact will be a method that, when applied during software development, satisfy all objectives mentioned in Section 2.2 which we consider as the artifacts requirements.

The treatment design in this thesis is created by means of knowledge gained from scientific literature and relevant stakeholder experience. With this knowledge, we attempted to answer the different research questions from Section 2.3. After possible answers to the questions were found, they were utilized to create a treatment design in the form of a method. The first iteration of this method was applied on an imaginary case study as a formative evaluation in an artificial context (Appendix A). During this evaluation, the method was validated with experts from industry and academics to conclude whether it would satisfy the objectives. Different iterations between the design of the method and the validation were made before the method was actually implemented in the problem context. The final iteration of the method is elaborated in the Chapter 4. Additionally, a rationale for the creation of every step is provided in this chapter.

Once the final treatment design was validated, the method was implemented in the problem context to conduct a summative naturalistic evaluation. The evaluation is done by means of a case study and is described in Chapter 5. The objective of the case study was to evaluate whether the designed artifact achieved the drafted objectives from section 2.2. Different data sources were used during the case study to realize data triangulation and therefore minimize validity threats (Runeson & Höst, 2009). These data sources are: observations, semi-structured interviews and analysis of data created by executing the solution method.

The final conclusions are stated in Chapter 7, together with the discussion and future work.

2.5 LITERATURE RESEARCH PROTOCOL

A hybrid approach is used as the literature research protocol. The initial idea of the thesis started with the paper: *"From Opinions to Data-Driven Software R&D"* (Olsson & Bosch 2014b). From this paper snowballing was applied to find relevant other literature. In addition, all related work of Olsson and Bosch has been analyzed to get a full understanding on data-driven software development. To complement the snowballing method on the initial paper, queries with relevant keywords concerning data driven and hypothesis driven software development were inserted in Google Scholar. By combining these methods, a wide spectrum of similar methods was acquired. The most promising methods are discussed in Section 3.1.

During the comparison of methods for data-driven development the problem statement was defined and with this research questions were drafted. Also, an initial high-level solution method was created which is discussed in Section 3.3. The high-level solution method should contain sub-methods and techniques on the areas of: customer value mapping to requirements, feature oriented development and software testing in terms of feature verification and validation. These topics were also explored by entering queries in Google Scholar and subsequently snowballing relevant papers. Furthermore, relevant journals were searched for relevant techniques to utilize within the solution method. The results of these topic searches are discussed in the literature review 3.4 – 3.6.

3 THEORETICAL FOUNDATION

The first half of this chapter (Section 3.1 & 3.2) compares and elaborates upon different methods that use R&D as an experiment system. The conclusion of this comparison, together with identified gaps, are stated in Section 3.2. Hereafter, a high-level overview of the primal envisioned solution method was created (Section 3.3) that could possibly address the gaps identified in section 3.2, and help with answering the research questions defined in Section 2.3. We created this high-level overview to use it as a guidance for finding possible techniques, or method fragments, that could be used for the solution method. The outcomes of this part in the literature research phase are stated in Sections 3.4 - 3.6.

3.1 METHODS TO FORM R&D AS AN EXPERIMENTING SYSTEM

The following subsections state different relevant methods found during the first half of the literature research phase. These methods intend to utilize R&D as an experimenting system. Every method is briefly explained together with its goal and position concerning the solution methods intended objectives. The last subsection summarizes- and compares these methods with respect to the intended objectives.

3.1.1 THE RIGHT MODEL

Fagerholm et al. (2017) propose the RIGHT model which focusses on delivering the *right* software instead of building the software right. The model aims to be a detailed framework to correctly guide the continuous experimentation process within development teams. RIGHT stands for Rapid Iterative value creation Gained through High-frequency Testing. The RIGHT model builds further upon the lean startup method, which includes the tripartite cycle: build, measure, learn (Ries, 2011).

The RIGHT model expands every of the three lean startup phases. During the build phase the business vision, or strategy, is translated into hypotheses. These hypotheses aim to validate the underlying assumption which a business strategy holds. Two parallel activities follow from these hypotheses; A minimum viable feature or product is created together with an experiment that aims to prove or disprove the hypotheses. Hereafter comes the measure phase in which measurement of the experiments are taken. Finally, during the learn phase, the experiment's results are analyzed and a decision is made concerning the course of the business strategy and product. These decisions are input for the subsequent iteration cycle.

Although Fagerholm et al. (2017) state that the RIGHT model is a detailed framework, a lot of detail is missing. No guidelines and practices are given to execute the model's phases. For example, no practices are given to prioritize and identify hypotheses. How should such a hypothesis be modelled? Concluding, the RIGHT model is a useful first step towards a detailed framework for continuous experimentation but stays superficial.

3.1.2 THE HYPEX MODEL

Olsson and Bosch (2014b) introduced a similar method called the "Hypothesis Experiment Data-Driven Development" (HYPEX) model. Olsson and Bosch argue that there is an "open loop" between management decisions and customer feedback because companies have too little mechanisms in place that capture customer usage data. The HYPEX model aims to close this open loop between

management decisions and customer feedback so that the company's product will not diverge from the customer's needs.

The HYPEX model (Figure 3) holds six practices that help companies run feature experiments with customers to improve data-driven decision-making. The six practices are:

1. Feature backlog generation: Features are generated that could potentially bring value to the customers and which can be experimented with.
2. Feature selection and specification: The highest priority feature is selected. The expected behavior of the feature is specified in terms of how the feature adds value to the customer and how it adheres to the business goals.
3. Implementation and Instrumentation: A slice of the feature's functionality, called the minimum viable feature, is implemented. The minimum viable feature's usage data is gathered after implementation.
4. Gap Analysis: The gap between the expected and actual feature behavior is compared. When the gap is sufficiently small, the team will continue development of the feature. When the gap is of considerable size, a hypothesis will be generated.
5. Hypothesis Generation and Selection: Different hypotheses are generated which possibly explain the gap between expected and actual feature behavior. In general, two main categories of hypotheses can occur: the feature slice has too little functionality to let the customer experience any benefits, or the feature does not adhere to the customer's needs. In case of the first hypothesis, the team can decide to further implement the feature. In the latter case, the team proceeds to the next practice. Note that this type of hypotheses is different than the ones mentioned in the RIGHT model.
6. Alternative Implementation: If the hypothesis states that the minimum viable feature does not meet the customer's needs, an alternative implementation of the feature will be selected, or the feature will be canceled. If an alternative implementation is selected, both implementations will be tested in the form of A/B testing. From this point on, the team returns to the gap analysis and repeats the circle until the gap is closed.

Although the HYPEX model is promising, no specific method for executing the six practices are given. Because of this, companies may struggle during the implementation of the HYPEX model. Specific techniques need to be contrived to actualize the HYPEX model so that companies have a clear notion on how to implement the HYPEX model. Furthermore, the model has a significant flaw. By applying the model, the practitioner solely verifies whether the goal fills the gap with the expected behavior. But what if the expected behavior adds little or no value to the customer? In this case the development resources will be wasted while it leads to little or no customer value.

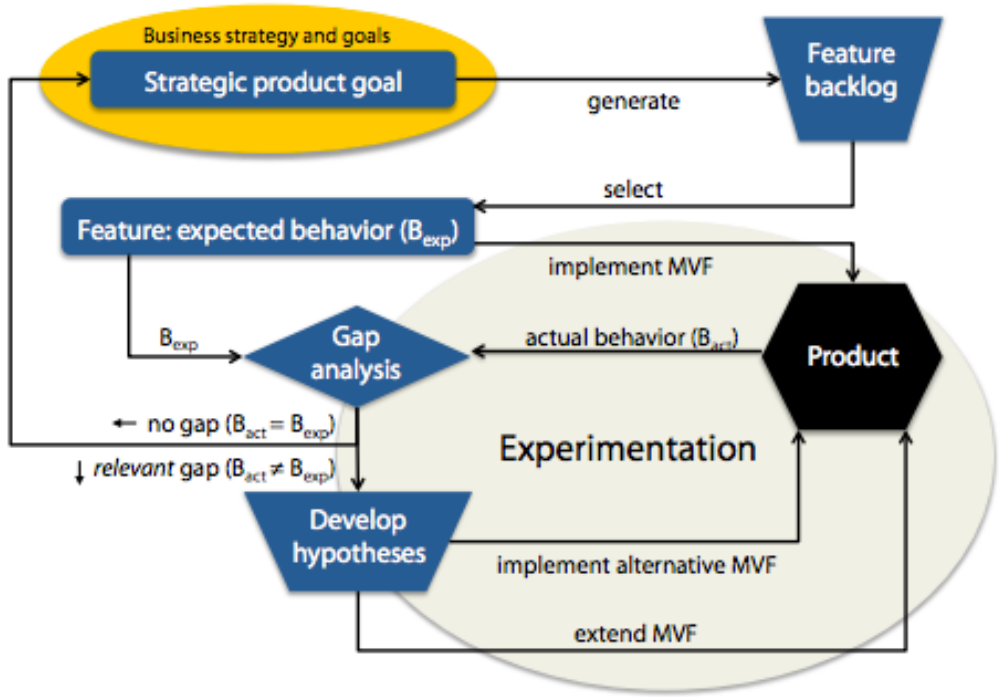


Figure 3. The HYPEX model (Olsson & Bosch, 2014, p. 5).

3.1.3 THE QCD MODEL

The "Qualitative/quantitative Customer-driven Development" (QCD) model (Figure 4), allows for continuous validation and re-prioritization of features, by means of qualitative and quantitative customer feedback techniques (2015). The model holds great similarities with the aforementioned RIGHT model. Hypotheses are derived from business strategies. After the Hypotheses have been prioritized, the company picks a data collection technique and validates the hypotheses. This gathered information helps the company to prioritize new features.

The QCD model stays very high-level and gives no practices to apply in a business context.

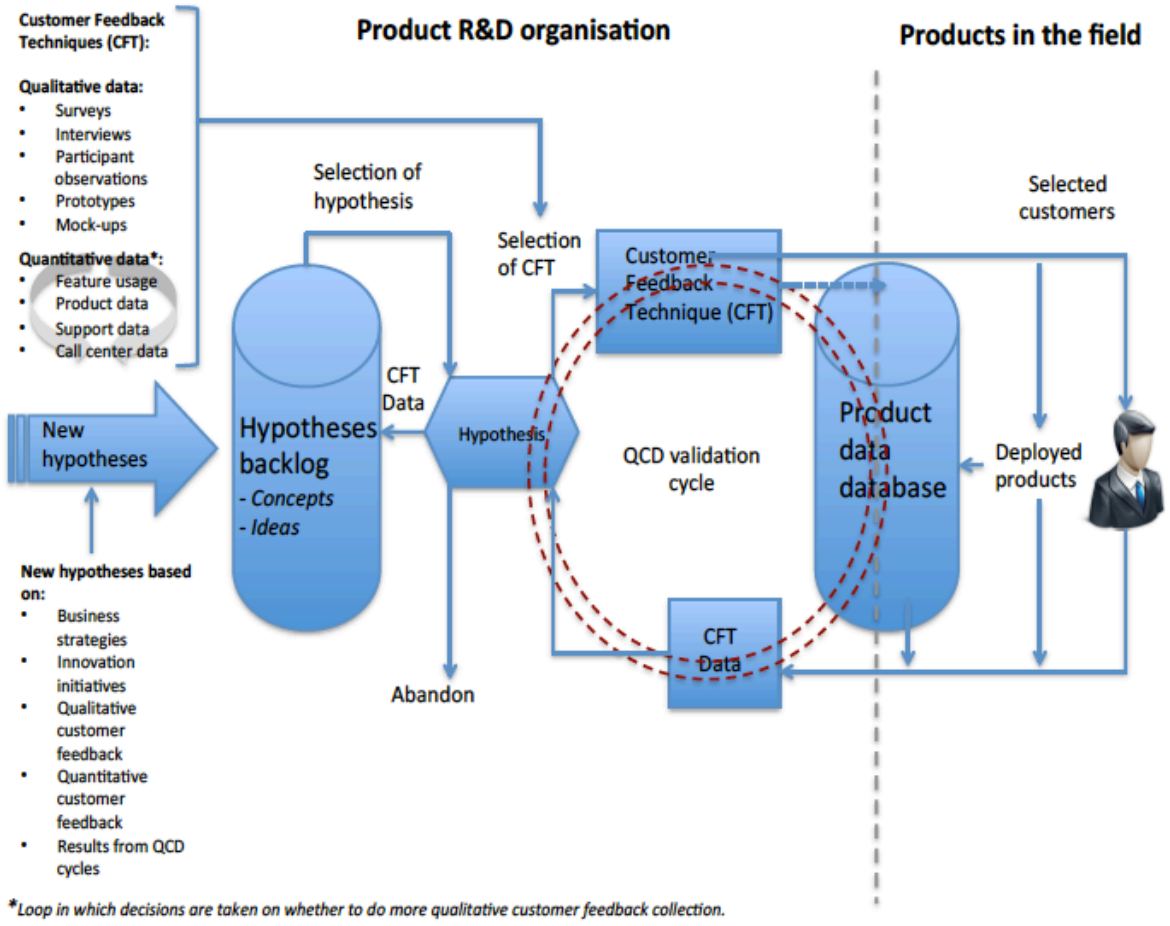


Figure 4. The QCD model (Olsson & Bosch, 2015, p. 64).

3.1.4 EVAP

"Early Value Argumentation and Prediction" (EVAP) is a technique that extends the QCD model. Fabijan et al. (2015) state that product management typically has to wait until the feature has fully been developed, until its value can be validated. EVAP is a technique that helps the company estimate the impact of a feature before it has fully been developed. The technique states that after the feature is selected, the expected value needs to be set. After this, several implementation iterations of the minimal viable feature (MVF) will be executed until the feature reaches its expected or unexpected value.

Although the EVAP technique gives a bit more direction on how to model a feature, it is not a strong approach to estimate the exact impact of feature value. For example, the EVAP technique holds a step which says that expected value needs to be evaluated. However, the technique does not prescribe a way on how the expected value can be evaluated.

3.1.5 DVOCE

The "Data-Driven and Value-Oriented Continuous Experiment" (DVOCE) process model is the first attempt to translate the high-level HYPEX model into a concrete and detailed procedure for the estimation of feature value (Ekström & Þorvaldsson, 2016). The model consists of the following steps:

1. Select: This phase is identical to the HYPEX model and aims to prioritize features for testing using value-based logic.
2. Model: The model phase consists of identifying and elaborating the feature's factors. Whereby the factor is a possible negative or positive measuring point delivered by the system. The model phase consists out of three sub steps:
    - Identify the expected functionality and affected factors.
    - Identify the starting state of the selected factors.
    - Define the value constants.
    - The authors use the Goal Question Metric (GQM) method during the first sub step to find metrics for a feature goal. Hereafter, if present, the starting state of the selected metrics are defined. Finally, the value constant is defined. The value constant is the amount of value a feature increases or decreases per difference in metric output. The value constant is typically based on domain expertise or available relevant data.
3. Predict: During this phase, the practitioner of the DVOCE model predicts the increase of feature value when implemented. Furthermore, the tolerated threshold gap is stated. The company will not further invest in the feature if its eventual contribution, in terms of value, is below this threshold.
4. Instrument: Data collection techniques are chosen by analyzing the metric's characteristics and product context.
5. Implement: During the first implementation step, the critical functionality of the feature is implemented. This is called the minimum viable feature. The authors state that this should approximately be 10-20% of the total feature's functionality.
6. Deploy: The authors of the DVOCE recommend pilot customers for testing the feature changes. This is done to minimize the risk of an unpredicted negative impact the feature can have on the product and company.
7. Monitor: After the feature is deployed, the actual value is decided by means of the measured values.
8. Analyze: The results of the collected data are evaluated and the gap between predicted value and actual value is measured. If there is no value gap the decision is obvious. However, when the value threshold is not reached the team has four options:
    - Gather more data
    - Commercially deploy the feature
    - Abandon the feature
    - Start a new iteration

    This phase is similar to the HYPEX hypothesis generation phase; The reason for the gap is hypothesized and a suited further approach is chosen.

The DOVCE model is based on data gathered from semi-structured interviews conducted with relevant experts of the industry. The model was drafted in several iterative cycles. After every cycle, the model was proposed to the experts and feedback was processed. Additionally, a prototype for a supporting tool was created which supports the modeling and analyzing phase of the DVOCE model.

Despite the fact that the model has conceptually been proposed to experts of the industry, the model was never validated by means of actual implementation.

3.2 METHODS COMPARISON

Objective \ Method	RIGHT	HYPEX	QCD	EVAP	DOVCE
FEASIBLE					
Possible to implement in real-life context	?	√	√	√	?
Well balanced level of process abstraction	X	X	X	X	√
Possible to align with an agile context	√	√	√	√	√
Utilize R&D as an experiment system	√	√	√	√	√
EFFECTIVE					
Better short-term decision making	√	√	√	√	√
Better long-term decision making (organizational learning)	√	X	√	X	X
Relate features with actual customer value	X	X	X	X	X

Table 1. A comparison between methods in terms of the solution method's objectives.

The objective to let the method be beneficial is not taken into account because this could not be deduced from the information provided in the papers.

The question mark depicts the unproven possibility that the method fulfills that objective. The HYPEX, QCD and EVAP method are studied in a real-life business context and are therefore checked with a "√". The RIGHT model has conceptually been proposed but not been studied in a real-life context. The DOVCE method has only partially been studied in a real-life context. Only the DOVCE method has clear steps that are adjustable to the method's context. The other method's stay to high-level to replicate in a study, or implement in a real-life context. All methods are designed to be workable within an agile context by using R&D as an experiment system.

All methods have the potential to help with short-term decision-making concerning feature development. The RIGHT and QCD methods also recommend drafting pre-development hypotheses that are linked to the organization's business strategy. Subsequently, the hypotheses are validated. Because of this, a reflection is conducted upon the business rationale that possibly results in better long-term decisions.

No method explicitly prescribes the linking of customer appreciation to functional feature goals. This concludes that no method tests whether implementing a feature, results in actual value for the customer.

3.3 A HIGH-LEVEL OVERVIEW OF THE PRIMAL SOLUTION METHOD

As explained in Section 3.2 , many methods exist that aim to continuously verify whether a feature is implemented correctly. The proposed solution is a method that combines the continuous verification and validation of building the right features and building the features right. In Figure 5 a high-level overview of the primal proposed solution method is depicted. The primal solution method is a loop whereby each phase is executed with a different frequency. This primal high-level overview is created to guide in the further search for possible useful methods and techniques in the literature.

The first phase is the mapping of customer values to a set of requirements. The assumption is that the chance of user acceptance is higher when the features are selected by keeping the types of valued dimension per customer segment in one's mind. A considerable candidate method fragment for this is elaborated in the section: “Map customer value to requirements”.

Once the requirements are prioritized they are input for the proposed method. Before the requirements are implemented, one should set the hypotheses, questions and metrics per feature. Once the metrics are determined, the development can start with building the features together with a functionality that implements the metrics.

After the features are implemented, they can continuously be monitored by means of the metrics to test whether they fulfill their goal. With this attained information, and other possible test techniques, the features are verified.

Figure 5. A primal high-level overview.

If the features are correctly implemented they can be validated whether they are valued by the user. For example, this can be fulfilled by implementing a simple in-product survey which shows only three icons that stand for satisfied, neutral, unsatisfied. This information can be gathered to measure the user’s actual value for a feature.

A significant part of the method is closing the loop between the actual customer values with the mapped customer values. The idea is that the customer value mapping is slightly adjusted every time the circle is completed. This creates organizational learning and helps in the prioritization of new features. Furthermore, the research done during the initial mapping of customer segments does not need to be repeated every iteration. Every phase of the primal high-level overview is elaborated in the following sections.

The last three elements of the solution method are possibly suited for integration with agile methods. Current agile sprints are mainly focused on the delivery of software. When developing features, it is important to already keep in mind how the functionality will be tested. This results in an easier implementation of metrics. Also, the verification and validation of implemented features can occur

during every sprint. Every sprint which is integrated with the solution method should therefore always hold two parallel activities which are input for the other type of activity for the next sprint (Figure 6). The developed features from the previous sprint create a context on which data can be harvested to verify and validate whether the right software is build. The outcome of this analysis leads to new feature hypotheses which are input for the next development cycle. Therefore, the last three phases can be repeated every sprint and give direction for the subsequent sprints. The first phase of the solution method is conducted less frequent and sets the prioritization of new features on a higher level.

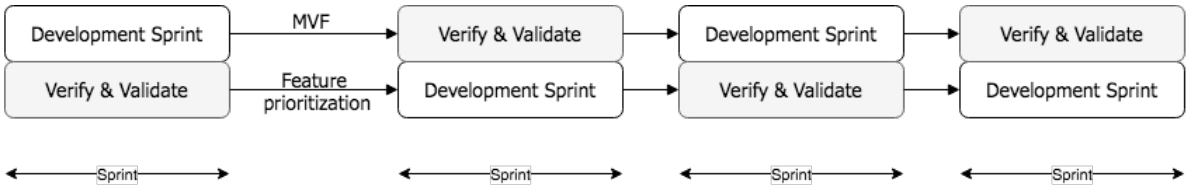


Figure 6. An agile alignment.

The holistic high-level overview of the solution method shows similarities with the continuous * view of Fitzgerald and Stol depicted in Figure 7 (2014). Fitzgerald and Stol argue that the Continuous * view is not yet saturated. The mapping of customer values to customer segments could be located in the business strategy area, as it delivers a prioritization of requirements which could be used as input for continuous planning. Develop and verify features are phases which show strong links with the development area. The validation of features is done during the operations by means of continuous use and continuous run-time monitoring. The continuous innovation is a similar feedback loop as the solution method.

The solution method could be seen as a specific instance of some elements of the more abstract view of Fitzgerald and Stol. However, there are slight differences in the goal. Both share the same lean philosophy of Ries (2011). The holistic view is mostly reactive whereas the solution method is more balanced in terms of proactive and reactive selection of features.

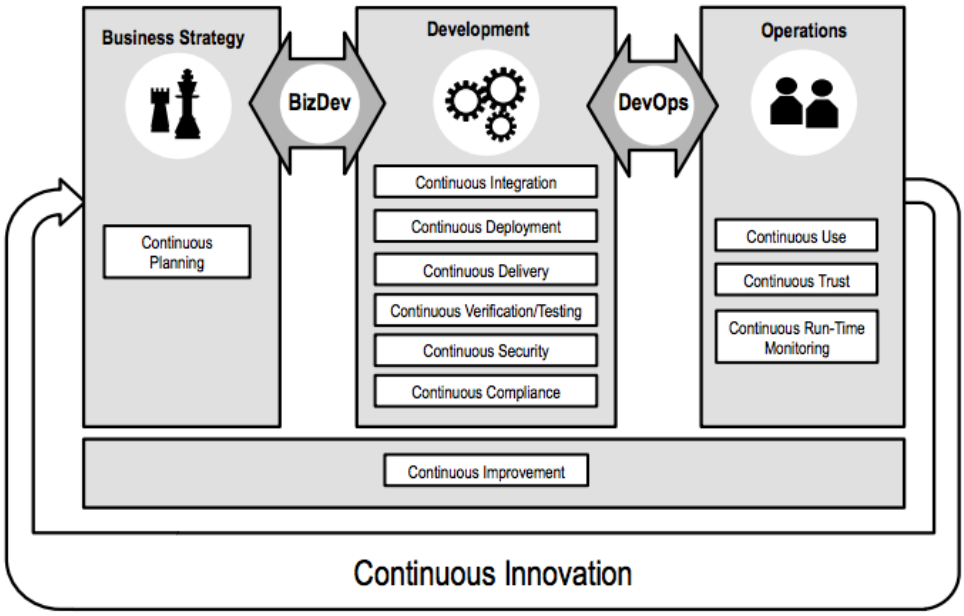


Figure 7. Continuous * a holistic view (Fitzgerald, Stol, 2014, p. 5).

3.4 MAP CUSTOMER VALUE TO REQUIREMENTS

Since user acceptance testing is an intrusive activity, a different approach for aligning with customer's wishes needs to be sought. Different customers have different values which they desire to see in the product they use. Customer satisfaction is linked to the difference between perceived and received customer value (Woodruff, 1997).

The concept of customer value can hold many different meanings. First of all, the concept can be divided in value *for* the customer and value *from* the customer (Smith & Colgate, 2007). Value from the customer is the value the customer generates for the company. This could be monetary value but also other forms of value such as product promotion among other people. Value for the customer is the value the customer perceives by utilizing the service or product of the company. The latter type of value is linked to customer satisfaction. If a company can deliver value to the customer it can possibly result in an increase in value from the customer. Salem Khalifa (2004) states that customer value is a considerable factor for competitive advantage and long-term business success. Furthermore, according to Slater (1997), delivering a high amount of customer value in general will increase the performance of the business.

The perception of a service or product value is context dependent and also depends on the specific values of the customer (Holbrook, 1999). The customer's own values are used as a criterion for conscious or subconscious judging whether the product brings any value for the customer. By mapping different values to different customer segments, a decision can be formed on the prioritization of requirements to develop. Mapping the right requirements to different customer segments will deliver more value to the customers. Subsequently, methods that use R&D as an experiment system (Section 3.1) can verify whether these validated requirements are correctly implemented as product features.

Several external and internal sources within the organization should be consulted to get a good overview on the hypothesized values of different customer segments. Such sources could be marketing, sales, support, R&D, customer, product and project managers, etc. Techniques to elicit the values of the customer are also present. A suited technique in this context is proposed by Zdravkovic, Svee and Giannoulis (2015). Their solution is a conceptual model that supports the elicitation of customer values and the translation of these to a set of product features which are tuned to the values of different customer segments. The benefit of this model is that the chance of selecting the *right* requirements for a higher customer satisfaction is increased without the continuous intrusion of validating every feature.

The conceptual model consists of two main steps (Figure 8). Firstly, the preference of the customer is captured by means of a consumer preference meta model (CPMM) in combination with a value framework taking user input, the product line and other context in consideration. Secondly, the consumer preference is translated into a goal model which is subsequently translated into a product specific feature model.

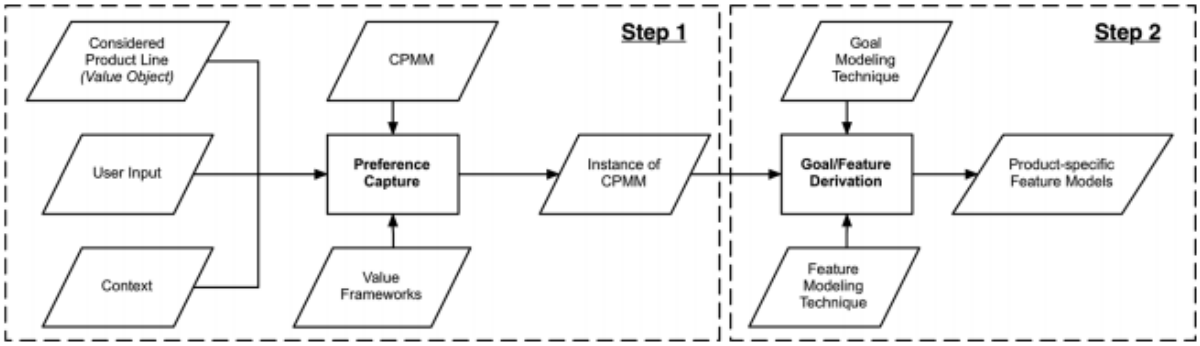


Figure 8. The method overview (Zdravkovic, Svee, & Giannoulis, 2015, p. 73).

Zdravkovic et al. (2015) discuss three possible value frameworks to capture the consumer's preference. The first is the hierarchy of human needs as introduced by Maslow (1954). The framework suggests that there are categories of needs in which any human finds value. Only if the needs of a preceding category are satisfied can a human proceed to the following category. For example, a human will only feel the need for love if he is physiologically doing well and has a feeling of security. The Maslow framework adds conceptual value but is hard to concretize for capturing consumer preference.

Another discussed value framework is Schwartz's Value Theory (SVT) (1992). According to Schwartz all values can be categorized in the following categories: Power, Universalism, Achievement, Benevolence, Hedonism, Tradition, Stimulation, Conformity, Self-determination, and Security. Different people can hold a different composition of preferences for these categories. Schwartz has created the Portrait Values Questionnaire (PVQ) as lightweight artifact to elicit what values are important.

A more consumer focused value framework is proposed by Holbrook (1999). The framework refines value concepts into customer values by focusing on the values people take in consideration during a value exchange. The values are divided in three dimensions: Extrinsic/Intrinsic, Self-oriented/Other-oriented, and Active/Reactive.

Zdravkovic et al. (2015) have taken the best of Schwartz's value theory and Holbrooks value framework and mapped to two together. The result is depicted in Figure 9. The customer preferences

are categorized according to this new value framework by means of qualitative and quantitative research.

		Extrinsic	Intrinsic
Self-oriented	Active	**Efficiency** (*Conformity, Security*)	**Play** (*Self-determination, Stimulation*)
	Reactive	**Excellence** (*Achievement*)	**Aesthetics** (*Hedonism*)
Other-oriented	Active	**Status** (*Power*)	**Ethics** (*Universalism*)
	Reactive	**Esteem** (*Power, Achievement*)	**Spirituality** (*Benevolence, Tradition*)

Holbrook's values are indicated by bolded Roman text, while Schwartz's values are italicized

Figure 9. Schwartz's Basic Values mapped to Holbrook's Consumer Values (Zdravkovic, Svee, & Giannoulis, 2015, p. 77).

The acquired customer data is placed in context of the CPMM (Figure 10). The CPMM is the theoretical and methodological for framework for capturing customer preference and translating it into the right requirements. A value object (the software product line) is delivered to the customer and needs to be aligned with the consumer values of the customer. The customer can be divided in different segments based on demographic data and context of use. The drivers of the customer can be measured in a qualitative and/or quantitative manner. Concluding, the values and demographics of different customers will be measured in a qualitative and/or quantitative way. Subsequently, customers will be divided in different segments which possibly can hold different consumer values.

The second step is the transformation of the captured consumer preferences to feature models. However, first an intermediate step will be taken by linking the preferences of consumers with requirements for the software product line. Different goal models can be used for this intermediate step but the authors suggest the i* goal modeling technique (i* Wiki, 2017). Subsequently, the goal model is transformed in a feature model (Silva, Borba & Castro, 2011). The feature models depict all the different features per consumer segment. These feature models give insights in which features should be implemented in every product configuration and which should only be implemented for specific customer segments.

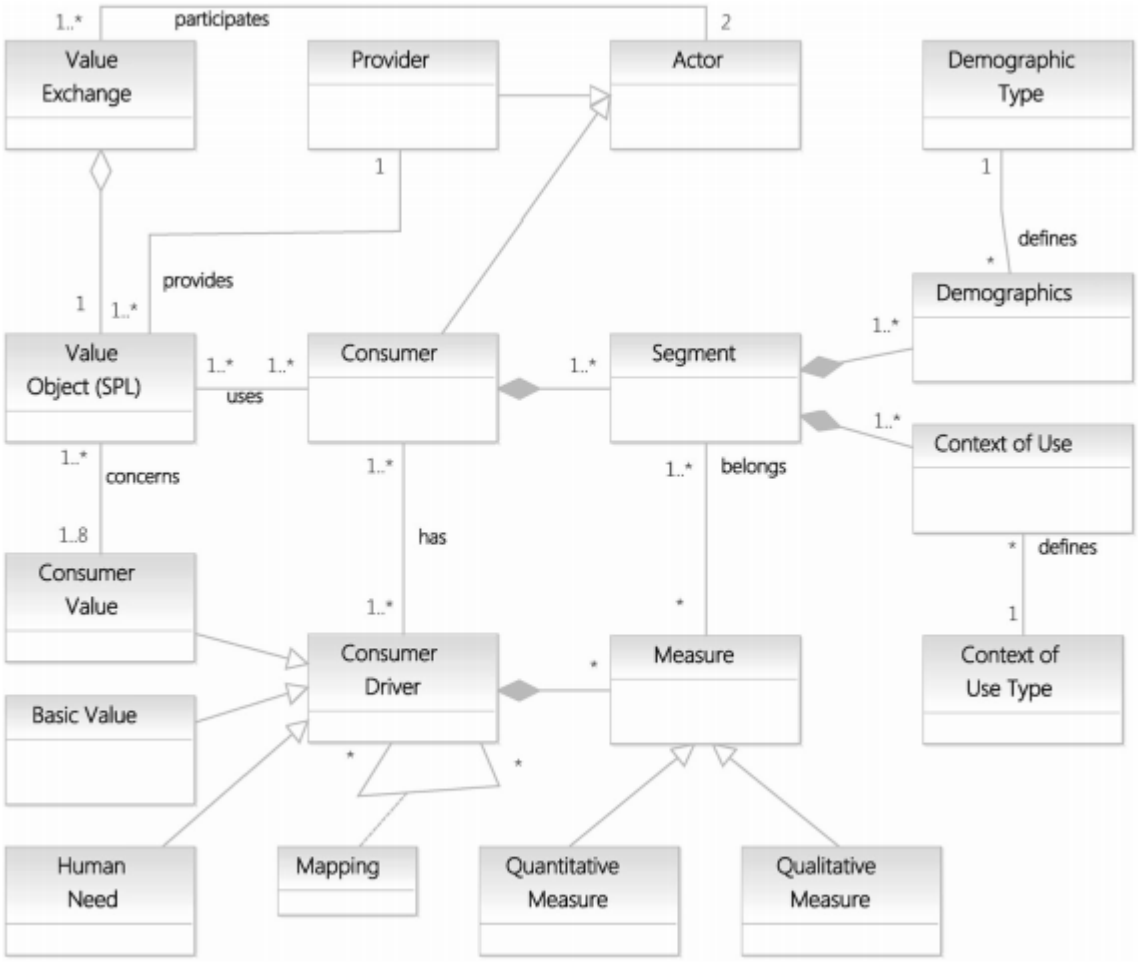


Figure 10. Consumer Preference Meta-Model (CPMM) (Zdravkovic, Svee, & Giannoulis, 2015, p. 76).

3.5 FEATURE ORIENTED DEVELOPMENT

This section provides an overview of the most prominent methods for feature oriented development. Feature oriented development aims to understand the software product line's problem domain, models a solution and ultimately transforms this into a solution architecture. Every method holds characteristics that could differently fit a company's situational factors and aspirations concerning software development.

Feature oriented development goes hand in hand with software product line management. By analyzing and modelling features, commonalities and differences within a software product or within a software product line can be identified. This results in a better reuse of code and saves unnecessary use of resources. Some methods focus on exhaustive documentation and tooling and are less agile but give a clear guidance for maintaining a proper architecture; other methods are lightweight and could therefore be used in a more agile environment.

Feature oriented development generally consists of four aspects:

1. **Domain analysis**: The requirements of different stakeholders are analyzed and typically translated to feature models.
2. **Domain design and specification:** The behavior of, and relations between, different features are analyzed and added to the feature models.
3. **Domain implementation:** How is a feature implemented in the best way? Different languages and tools for the implementation of features are developed by researchers (for example, aspect-oriented programming).
4. **Product configuration and generation:** Once features are implemented, different products are managed in terms of feature configuration.

3.5.1 FODA

Feature-Oriented Domain Analysis (FODA) is primarily focused on the modelling analysis of the product domain (Kang, Cohen, Hess, Novak & Peterson, 1990). It is known as the method that introduced feature models to domain engineering (Czarnecki, Eiseneckerr, Ulrich, 2000). Less emphasis has been put on the translation from feature model to architecture; specific rules for transforming feature models to architecture are absent within FODA. Because it is only focused on modelling features it is relatively lightweight and suited for an agile work environment.

3.5.2 FORM

Feature-Oriented Reuse Method (FORM) builds further upon FODA and provides guidelines for the creation of feature models, design and implementation phases (Sochos, Philippow, & Riebisch, 2004; Kang, Kim, Lee, Kim, Shin, & Huh, 1998). This method could be seen as an evolution of the FODA. However, it does not describe concrete processes to transform from feature model to architecture. The high-level processes that FORM does describe are depicted in Figure 11.

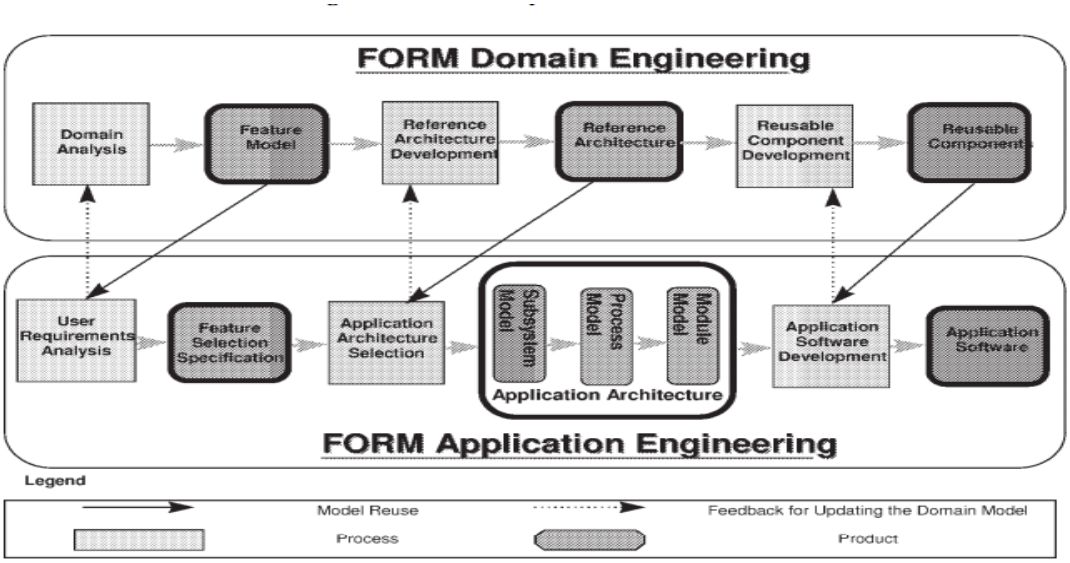


Figure 11. FORM engineering processes (Kang, Kim, Lee, et al., 1998, p. 149).

3.5.3 FEATURSEB

Reuse-driven Software Engineering Business (RSEB) is a systematic model-driven approach aimed to fully utilize software reuse (Griss, 1997). The approach is not specifically aimed at features. FeatuRSEB combines practices from RSEB and FODA. Within FeaturRSEB, every feature is linked with a usecase. These links are called traceability links. Because the use cases are connected with classes, these traceability links map features onto architectural elements. The disadvantage of this method is that the amount of traceability links exponentially increases when the product line increases in size.

A proposed solution for this is called the HyperFeatuRSEB. This method does not make use of traceability links but uses hyperspaces. This results in better scalability. However, the method also holds several disadvantages. It is hard to maintain and has little tooling support (Sochos, Philippow, & Riebisch, 2004)

3.5.4 FARM

Feature Architecture Mapping (FArM) is proposed as method that both closes the gap between feature models and architecture as well the quality of the product line in terms of maintainability and scaling (Sochos, Philippow, & Riebisch, 2006). The method aims to solve the maintainability problems of aforementioned methods by means of a modular plug-in architecture. A Modular plug-in architecture results in loosely coupled features that can more easily be added or removed from the architecture. The method is supported by several tools.

3.6 VERIFICATION AND VALIDATION

Davis and Venkatesh (2004) differentiate between two types of software defects that can occur during the development of software. Firstly, the defined requirements can be faulty implemented due to coding or design errors. Secondly, defects in the alignment between specified user requirements and true implemented features can occur. They further state that software development practices have made great improvements in preventing, identifying and eliminating the first type of defects. However, techniques for validating the correctness of requirements are lagging behind. Testing the first type of defects is the verification of requirements and testing the second type of defects is the validation of the software. As stated before, many methods which guide to form R&D as a continuously experimenting system are focused on the verification of requirements and features by means of user data. Unfortunately, little focus has been put on the validation.

The ultimate goal of any software system is that it is accepted by its intended users. Davis introduced the technology acceptance model (Figure 12) which states that actual system use is influenced by the user's attitude towards the software system (Davis, 1993). This attitude is influenced by the user's perceived usefulness and perceived ease of use of the software system. The perceived ease of use has a causal effect on the perceived usefulness. However, perceived usefulness is a significantly stronger determinant on attitude than perceived ease of use (Venkatesh, Morris, Davis, & Davis, 2004). The perceived usefulness and perceived ease of use are, in their turn, determined by the system's design features. Therefore, to make sure that a software system is accepted, it has to be tested whether it is perceived as useful and easy to use.

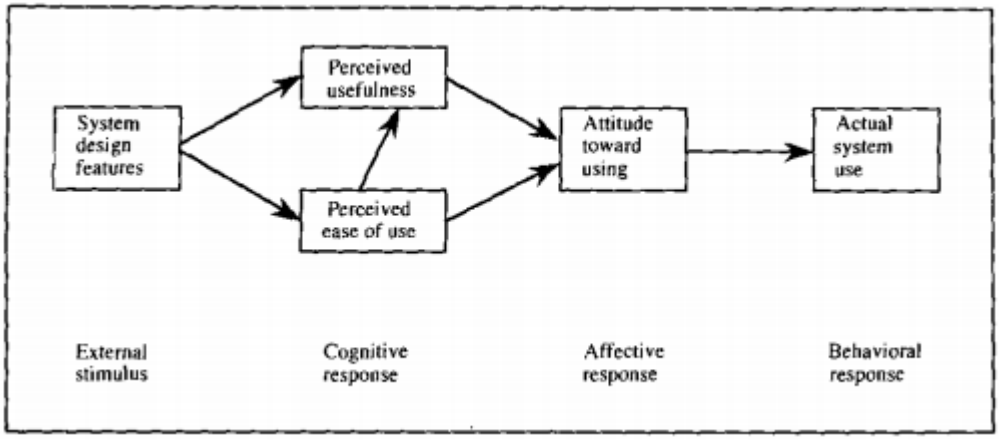


Figure 12. The technology acceptance model (Davis, 1993, p. 476).

The perceived ease of use and usefulness of a software system hold a relation with the aforementioned two types of defects a system can have. If a software system's requirements are faulty implemented, its ease of use will be reduced. However, when there is poor alignment between user requirements and true implemented features the system's perceived usefulness will be low. It is remarkable that, in general, relatively little effort has been put on the validation of features and requirements since the perceived usefulness is the greatest determinant of the user's acceptance.

A common testing technique is acceptance testing. Acceptance testing has four goals; verify the man-machine interactions, verify that the system operates within the specified constraints, check the system's external interfaces and validate the required functionality of the system (Hsia, Kung, & Sell, 1997). Acceptance testing is not user centered and does not put many emphases on the user's

perception of the software product. User acceptance testing (UAT) is a more specific type of acceptance testing which aims to use actual system user's feedback (Larson, 1995). Therefore, UAT is a possible answer for testing the user's perceived usefulness of a product.

Although UAT is a suitable technique for discovering the perceived usefulness, it has some serious disadvantages in the context of a method to form R&D as a continuously experimenting system. Namely, UAT requires a continuous user involvement throughout the development lifecycle. The lack of user involvement is being identified as a major problem in agile (Collins, 2012). Certain opportunities have been proposed to tackle this problem. For example, the creation of a wiki holding the developed changes where users and developers can asynchronously work on and test new features (Otaduy & Diaz, 2017). However, considering that continuous development, and thus continuous testing, is on an even higher pace than agile, classical UAT is not a desirable option for implementation within the solution method.

4 FINAL METHOD ITERATION

After the creation of the primal high-level overview of the envisioned method different iterations were made upon the creation of the solution method. One early iteration is described in Appendix A. The final method iteration that was implemented for evaluation is described in this chapter. Each section elaborates upon the defined activities per method phase. As this is the final version of the solution method, we coin the name "Learning as an Organization to Optimize the Product" (LOOP) method.

For documenting, communicating and designing the LOOP method we availed ourselves of the method engineering domain. Method engineering is defined as: *"the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems."* (Brinkkemper, 1996).

The standard notation in this domain introduced by Weerd and Brinkkemper (2008) is the Process-Deliverable Diagram (PDD). The PDD consists of two main elements. The left-hand side of the PDD holds the activities executed during the method. The right-hand side of the PDD shows the deliverables (concepts) produced by fulfilling the activities on the left-hand side. The PDD syntax is based on the Unified Modelling Language. For a more thorough understanding of the syntax, one should consult the paper of Weerd and Brinkkemper (2008).

Every phase accompanied with a PDD to provide a direct oversight of the method's activities. The PDD of the total method overview can be found in Appendix B.

4.1 LOOP PHASE 1: FEATURE SELECTION

The goal of this phase is to make ideas and opinions about features explicit. By rigorously documenting the hypotheses about features upfront, the post-development data of features could be compared to the initial hypotheses. One can then reflect whether the conducted reasoning and assumptions made concerning the features were valid. This reflection therefore helps in organizational learning and better feature prioritization during the next iterations.

First, the product manager needs to indicate suitable features for the method. These features will be the input of the first phase. The output of the first phase is a feature tree in which different features can possibly be linked to different customer segments. Furthermore, every feature holds a goal and rationale that includes the intended value of the feature. For every feature, a statement is made by the product manager about why it adds value to that specific customer segment. This information is important for later reflection once the feature is actually implemented.

In order to go from the input to the output of the phase, a set of activities needs to be executed. These activities are:

1. Indicate suitable features for the method;
2. Gather stakeholder assumptions on features;
3. Perform market analysis *(Optionally)*;
4. Divide users in different segments *(Optionally)*;
5. Develop or adjust the feature tree (*Optionally*);
6. Add a feature goal to every feature;
7. Add a hypothesis (per user segment) to every feature;
8. Decide on which product configurations are going to be (partially) implemented.

The optional activities do not have to be executed in every context. The prerequisites for conducting these activities are discussed and elaborated upon in the following sections.

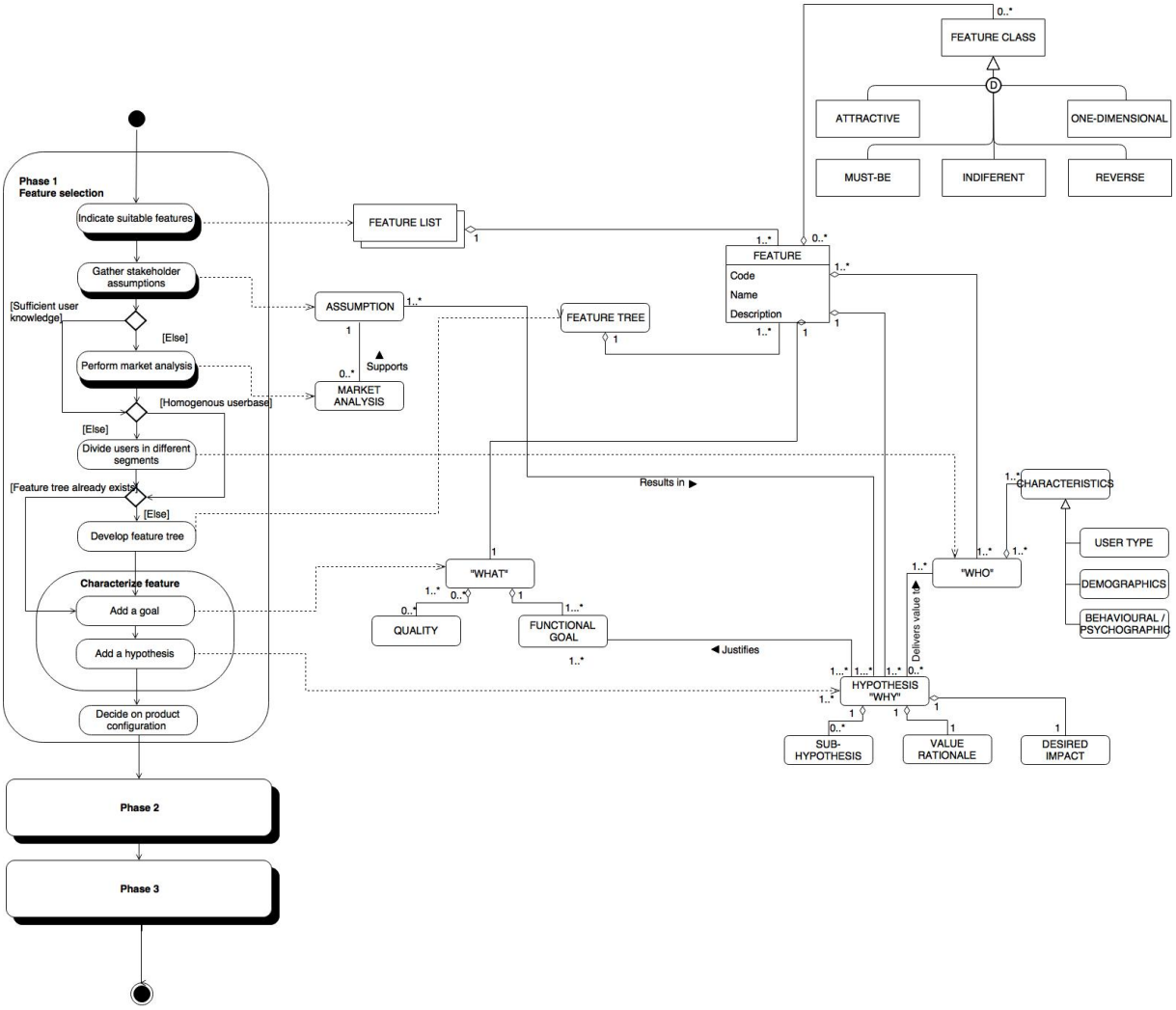


Figure 13. PDD LOOP phase 1.

4.1.1 INDICATE SUITABLE FEATURES FOR THE LOOP METHOD

The software product manager receives and defines a continuous stream of possible features to implement. Since there is only a limited set of implementation resources, these features need to be prioritized. For example, a possible method to obtain prioritization is the Kano model (Figure 14) (Kano et al., 1984; Sauerwein, Bailom, Matzler, & Hinterhuber, 1996). Features are classified by estimating the features' behavior in terms of implementation effort and perceived value. Possible feature classes are:

- *Must-be*: When implemented the customer is neutral. When the feature is not implemented the customer is very dissatisfied.
- *One-dimensional*: This feature results in satisfaction when completed and dissatisfaction when not implemented.
- *Attractive*: When the feature is not implemented the customer is neutral. When the feature is implemented the customer is very satisfied.
- *Indifferent*: The implementation of the feature does not affect the customer's satisfaction in any direction (not in the original model).
- *Reverse*: The more the feature is implemented, the more the customer is dissatisfied (not in the original model).

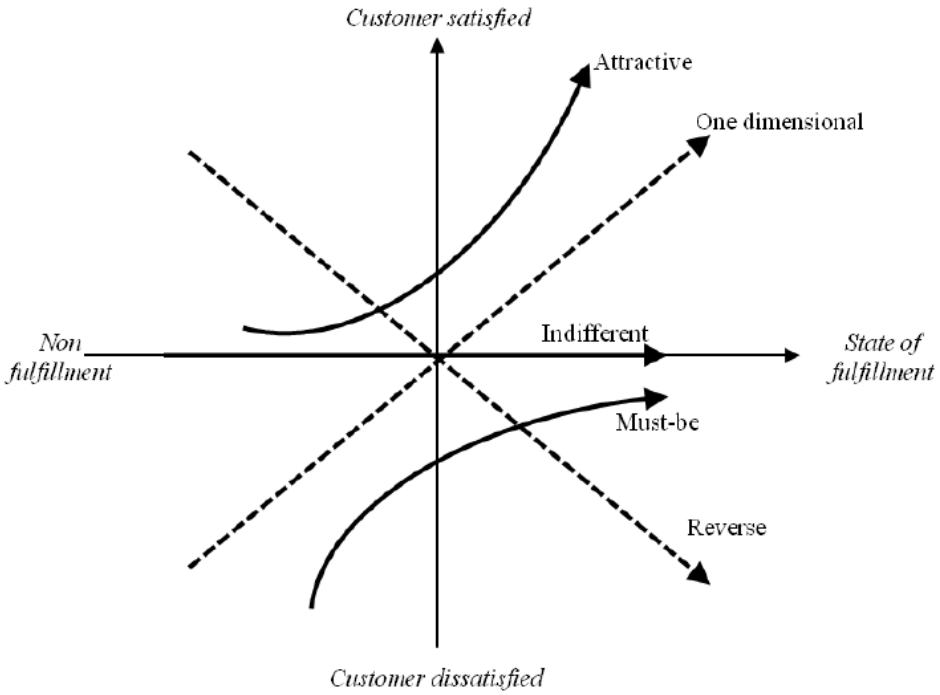


Figure 14. The Kano diagram (adapted from Kano et al., 1984).

The features which are not categorized as *must-be* are optional, but do nonetheless make the difference between a sustainable successful product or not. It is therefore important to choose the right

set of features, and reflect upon the rationale for these features once implemented. Before the development phase, the product manager can only estimate whether a feature belongs to a certain class or not. Any feature could be input for the LOOP method. However, since the LOOP method is costly, not every feature should be validated by means of the LOOP method because the benefits of the acquired knowledge about this feature will not outweigh the cost of resources. We recommend the type of features with the following characteristics as input for the LOOP method:

1. Unimplemented features from which the product manager is not sure to which class they belong, but look promising in terms of user satisfaction based on stakeholder assumptions.
2. Equipollent features that solve the same requirement by different means (if the current implemented equipollent feature is not sufficient).
3. Features that are implemented, but from which the product manager is not sure whether they are (still) delivering value to the user.
4. Implemented features that are implemented for a specific type of user but could possibly deliver value to another type of user.

The first two types of feature are suited because there is an indication that these features are promising but the product manager cannot be sure whether this indication is right until they are implemented. After implementation, the product manager can check whether the rationale for choosing these features was correct or not. By doing this, the product manager fosters the decision process for future features. Furthermore, the actual behavior of these features and the perception of the user on these features is documented. This can also aid in future decision making.

Features are solutions to solve requirements. Features are part of the solution domain and requirements are part of the problem domain. Multiple requirements can be captured by one feature and vice versa. So even if a requirement is mandatory for the next release, alternative equipollent features could be implemented to include this requirement. These features may differ in the degree of a requirement's fulfillment. Furthermore, features can also differ in the set of (quality) requirements they satisfy.

The third type of features is suited because the product manager can test whether features are or have become obsolete. "Sunsetting" the right features, helps against software bloat and therefore keeps the product lean. According to the Kano model theory, feature classes are not temporal by nature. A feature which once was an attractor could by time become a feature that every customer needs and therefore be a classified as "must-be". Even features in the must-be class can become obsolete for the user. Because of this, it is interesting to also examine this type of features.

The fourth type of feature is suited because the product manager needs to make sure that every type of user is exposed to an optimal configuration of features. By using the LOOP method, features can be reflected per user segment. If the exposure of a feature increases the satisfaction of a user segment it can be included, if not, it can be excluded. By hypothesizing why features add value to this user segments and subsequently validating these hypotheses, the company's understanding of the different user segments is evolving. This helps in the optimization of the product's feature configuration.

4.1.2 GATHER STAKEHOLDER ASSUMPTIONS ON FEATURES

Every feature is in the backlog because a certain stakeholder assumes that the feature will add direct or indirect value to a specific set of user segments once implemented. Features that add indirect value a user segment contribute to the quality of the product, but do not result in a perceivable change in product behavior for a user segment. Features that do add direct value, will result in a perceivable change in behavior for a user segment. Features that are hypothesized to add direct value to a user segment are potential for validation and verification by means of the LOOP method. Every of these features has a certain goal. The stakeholder that advocates for the feature needs to write the goal of the feature down together with the rationale why this goal will deliver value to the customer once satisfied. Whether the goal is satisfied will be *verified* during the third phase. Whether this goal actually adds value to the customer is *validated* afterwards.

4.1.3 PERFORM MARKET ANALYSIS

The necessity of this activity is dependent on the company's context. If there is sufficient knowledge about the product's (potential) customer and user base, a market analysis is unnecessary. In case of little knowledge, the company can choose to perform a market analysis. By doing this, a better understanding of the user's/customer's values linked to possible segmentations can be acquired. A very suitable technique that can be applied in order to find value per customer segment is the one proposed by Zdravkovic et al. (2015) (Section 3.4). Other marketing research techniques can also be suited if they can capture knowledge about user segmentation with accompanying values.

4.1.4 DIVIDE USERS IN DIFFERENT SEGMENTS *(OPTIONALLY)*

If there is heterogeneous user base, the base should be divided in different segments. These segments should be made on the right level of abstraction. If the division is based on too much user characteristics, every separate user could become a segment; whether if the division is too high-level, the segments could have no meaning in their differentiation. The reason to divide is that users in different segments could have variations in the features they value. Therefore, different configurations of the product or product line should be tuned to the different needs of each user segments. How to specifically divide the user base in separate segments is context dependent. The user base could for example be divided based on demographics such as age, gender or geographic data, but could also be divided based on the type of end-user they are, or on the type of customer who is paying for the end-user's usage (Table 2).

Demographics
Age
Gender
Geographical data
User type
Type of device
Authorization
Customer
Behavioral/ Psychographic
rate of usage
benefits sought
personality
Attitude towards product

Table 2. Example characteristics for dividing a user base

4.1.5 DEVELOP A MAIN FEATURE TREE MODEL FOR THE PRODUCT *(OPTIONALLY)*

Originally, feature models/trees are developed to visualize the commonalities and variability between a software product line (Kang et al., 1990). We argue that feature tree models are also a suitable technique for modelling the variability and commonalities within a product in order to depict different product configurations. Every leaf should contain different features that add direct value to the customer. The higher leafs are abstractions of the features that all address the same goal.

An example feature tree model of a SaaS based human-resource planning tool is depicted in Figure 15. In this example two high-level features from an imaginary software product are included: *Navigation* and *Plan Tool*. A sub-feature of navigation is the means to log-out once the user is in the main menu. Two possible versions of this feature could be implemented which are *Uncomplicated* and *Fast.* The fast feature uses a keyboard shortcut to log out. The uncomplicated feature makes use of visual clues to click through the product. This feature could further be implemented in two alternative manners. A navigation could be similar to the old interface or a new interface could be implemented to navigate to the logout menu. For the built-in planning tool, different visualizations of the planning are possible to implement. The development team can choose between a detailed visualization or a simpler uncluttered visualization.

To link features in a database a code is ascribed to every feature. Except for the first layer, all features are labeled with this code. First a feature gets the code from its parent feature. Concatenated to this code a feature receives a .1 if it is the most left child of its level, a feature is concatenated with a .2 if it is the next feature of its level, etc.

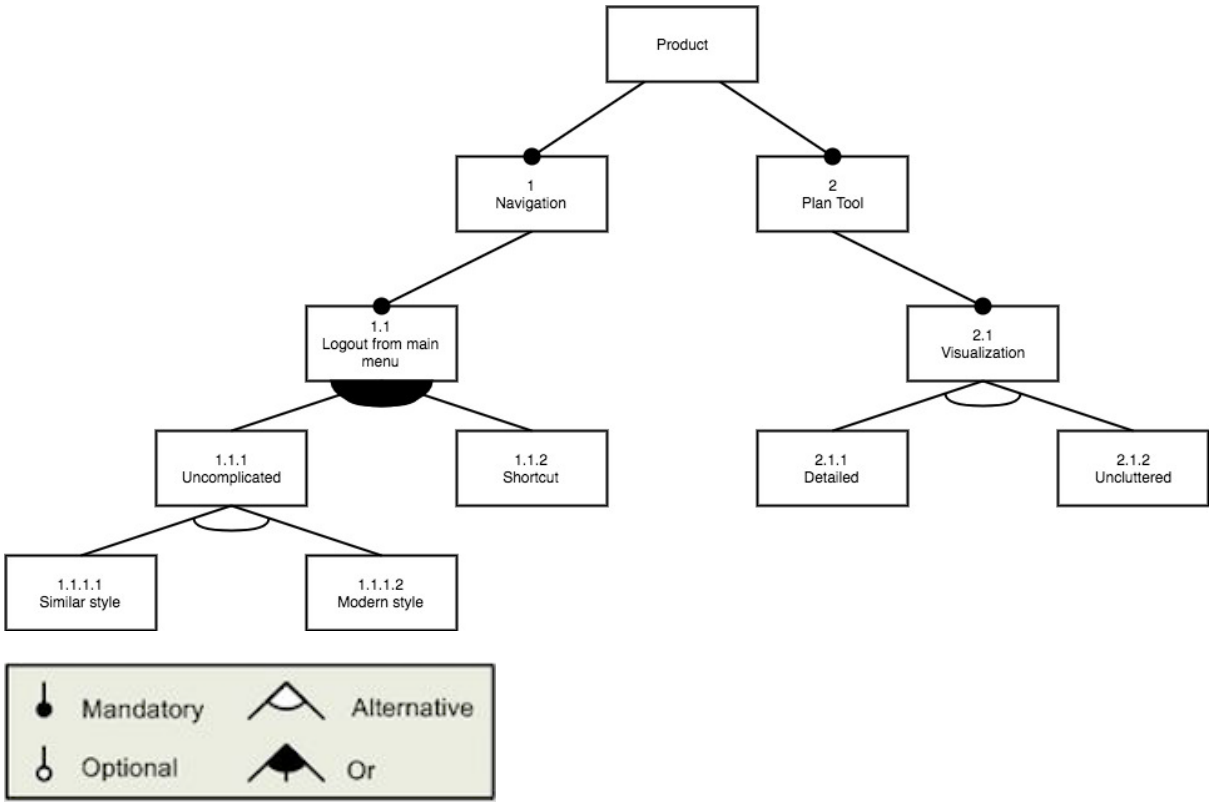


Figure 15. An example feature tree model

4.1.6 ADD A GOAL TO EVERY OPTIONAL FEATURE

Every optional feature should have a goal that distinguishes it from the alternative optional features. During this phase, only a high-level goal should be specified per feature until the feature is chosen for implementation. Every goal holds two aspects which should be elaborated: What the feature should do and how the feature should do that (the description). The qualities and functionalities that are improved by implementing the goal should be made explicit within the "*what*" section. Possible goals are illustrated in Table 3 for the optional features that are present in the example of Section 4.1.5.

Name	Code	Description	Goal "what"
Uncomplicated: Similar style	1.1.1.1	Buttons on the same place, click-through path through same menus	A similar navigation compared to the old interface. Qualities: Familiarity, ease of use
Uncomplicated: Modern style	1.1.1.2	New interface with less clicks to logout. Very different compared to the old version	A faster navigation compared to the old version. Qualities: Speed
logout: shortcut	1.1.2	Offer a keyboard shortcut	Fast logout from main menu. Qualities: Speed
Visualization: Detailed	2.1.1	All data about the planning is visualized or could be accessed by means of clicking on data	Detailed presentation of planning data. Quality: Providing Information
Visualization: Uncluttered	2.1.2	Uncluttered table with only basic aggregated information.	Easy and fast oversight of planning. Quality: Speed, ease of use

Table 3. Example goals linked to the leaf features.

4.1.7 ADD A HYPOTHESIS (PER CUSTOMER SEGMENT) TO EVERY FEATURE

Where the goal and the description state what a feature should do, and how a feature should act, the hypothesis states why it should be like this and also for whom this feature delivers value. It is important to clearly distinguish the difference between these concepts.

The feature's goal "What" is the intended change in software behavior and quality. The hypothesis "Why" is the reasoning for intended change in user perception as a reaction to the change in software behavior and quality. The latter is achieved when the feature's goal in terms of intended change in software behavior and quality is realized and if the hypothesis is true (Figure 16).

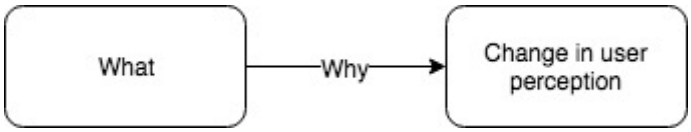


Figure 16: Depicted relationship between a feature's goal, hypothesis and user perception impact.

Once a feature is implemented, it can perfectly fulfill its intended goal but still add little to no value to the user. The hypothesis links the (type of) user to the goal. The type of user should be a previously selected user segment. If the hypothesis is right, the realization of the feature's goal will indeed increase value for the user. However, if the hypothesis is wrong, the realization of the goal can have no effect or even detract value to the customer. Therefore, *the goal* and *the hypothesis*, are both prerequisites to make a feature successful in terms of user value. The real assumption about a feature

lies within the hypothesis. Different stakeholders can have different assumptions about which features add value to the user. Therefore, it is important to evaluate the hypotheses for features after every iteration. By doing this, the organization will learn about implemented assumptions, and with this make better future assumptions.

The goal should be verified after implementation. This is because, if the goal is not truly implemented, one can take the wrong conclusions about the hypothesis. Once the goal is verified, the hypothesis should be validated. Different techniques for this will be elaborated in following sections.

The verification of goals is widely done in industry. For example, if the goal of a feature is to make it faster, it is common practice to actually test whether the feature has made a certain aspect of the software faster. However, the actual validation of the feature's hypothesis in a systematic manner is far from common. Consequently, features could be built that add little or no value to the user of the software product.

THE TYPE OF RELATION

The hypothesis can go further than only stating why the change in software functionality will deliver value for the user. It can also describe the type of relationship between the fulfillment of the goal *"what"* and the change in perceived value for the user. For example, if a feature can make a process faster with certain degrees, one could hypothesize that only the improvement of speed within a certain ratio (Figure 17, between the dotted lines) will increase the satisfaction of the user because no difference in speed is perceived within certain thresholds (i.e. once a process is loaded in "the blink of an eye", the user will not perceive any added speed on top of this because the speed is above the user's perception threshold).

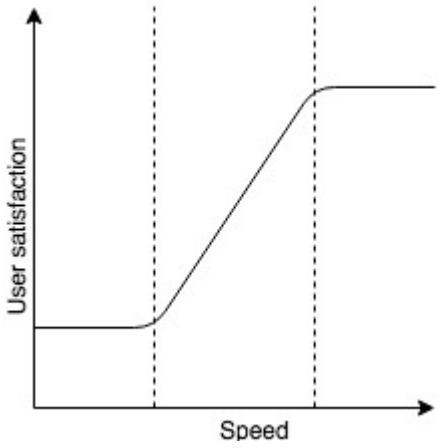


Figure 17. A possible relationship between functionality and user perception.

Some goals cannot be fulfilled in degrees but can only be implemented or not implemented (for example, a print functionality). Hypotheses may also hold relational information about features with these binary fulfillment levels and changes in user perception (Figure 18). One can predict a big impact in user satisfaction which is depicted in the Figure 18 as the *Delta*. It can also be interesting to look at the difference of delta values in user satisfaction created by one single feature over a certain time. The two illustrative examples of relationships are by no means exhaustive, and many different relationships between the fulfillment of a goal and user satisfaction are thinkable.

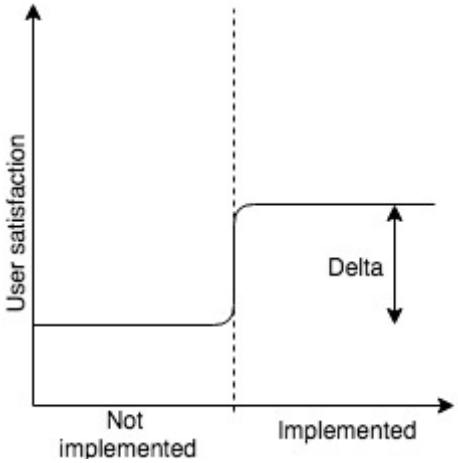


Figure 18. A possible relationship between functionality and user perception.

THE FEATURE VALUE MODEL

A feature goes through several phases before it will effectively deliver value over time to the user. Based on these phases, we have coined the Feature Value model (Figure 19). The first phase is the introduction of a newly introduced feature within the software product. From there on, in an ideal situation the user will become aware, starts using the feature, perceives value from the usage of the feature and retains this value over a certain period of time. However, during every transition between these different phases, the user can "take" an alternative path that leads to little or no value delivered by the feature. For example, the user can be aware but chooses not to use the feature whatsoever, and thus the feature delivers no value for this user.

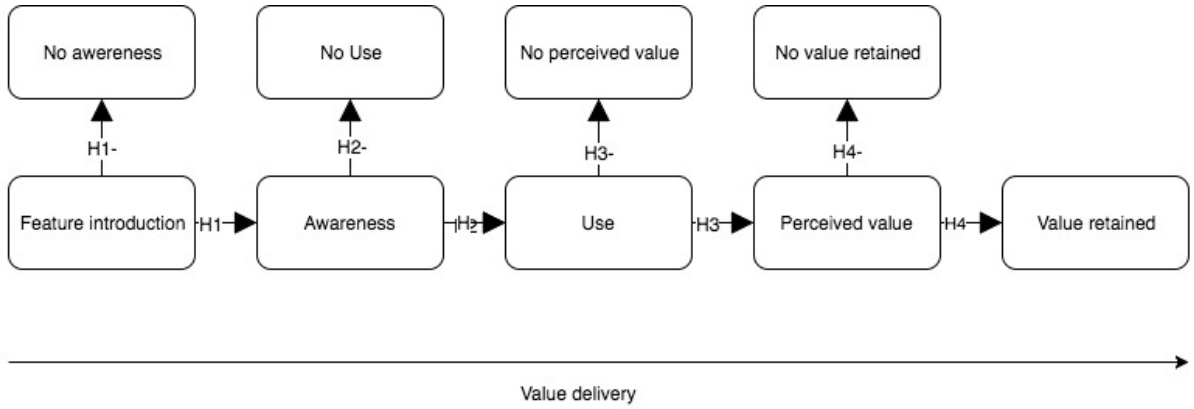


Figure 19. the Feature Value model.

As stated before, the goal of a feature states the intended functionality. The main hypothesis of a feature states the general rationale why the intended functionality will deliver value to the user (Figure 16). In certain cases, the hypothesis of a feature should be refined into more fine-grained hypotheses that are validated separately. The feature value model helps with dissecting the feature's main hypothesis into smaller hypotheses. These smaller hypotheses are a possible rationale for the transition between the different phases in the model.

Hypotheses about every transition between a phase are always made (implicitly or explicitly) during the design phase of a feature. The H1 in Figure 19 stands for the possible hypotheses why the user will become aware of the feature. Possible examples of these hypotheses can be:

- the user can easily find the feature in the navigation of the software product;
- the user is confronted by the new feature when logging in;
- the user will receive a notification about the new feature;

When a feature does not show the intended behavior, one can draft a negative hypothesis about the transition between phases. These are depicted with a "-" sign within Figure 19. Possible examples of H1- are:

- the user cannot find the feature within the navigation because it is in an illogical place;
- the user is not notified about a new version of the product holding the feature;

The hypotheses going from awareness to use (H2), or nonuse (H2-), concern whether the user will, or will not, start using the feature after becoming aware. The hypotheses from use to perceived value (H3), or no perceived value (H3-), are about why there is an increase in value for the user after he starts using the feature. The last phase transition is linked to hypotheses holding the rationale why this value will be long lasting (H4), or not (H4-). Whether these hypotheses are correct can be tested by means of the LOOP method.

Not all sub-hypotheses need to be explicitly stated in every context. For instance, if a user is always exposed to a feature during the usage of the product, it is not relevant to state whether users are aware of the features existence.

LINK TO USER STORIES

The segment, goal, and hypothesis, are closely linked with user stories (Folwer & Highsmith, 2001). The user story format is: As <persona>, I want <what?> so that <why?>. The *persona* could be linked to a user segment. The *what* is linked with the goal. The *why* could be seen as the hypothesis. Therefore, if phrased correctly, user stories could also be used for the method in an agile context. However, stating the features according to Table 4, gives the product manager more expressive power since he does not need to adhere to the strict user story format.

To illustrate possible hypotheses, we add these to the features and goals (see Table 3). In the first phase, the hypotheses can still be very high level and do not have to be very specified. This will only be necessarily during the second phase when the product manager has committed to the feature.

Name	Code	User segment “Who”	Hypotheses "why"
Uncomplicated: Similar style	1.1.1.1	Users who make irregular use of the application such as senior management	Learning new interfaces takes extra time which is not beneficial for irregular use. Therefore, it is more time-efficient for this group to keep a similar interface.
Uncomplicated: Modern style	1.1.1.2	Users who make exhaustive use of the application such as the operational manager	People who use the application more often will get a benefit in terms of time by using the new navigation. However, there is a small learning curve in the beginning which could be unattractive for irregular users.
logout: shortcut	1.1.2	Users who make exhaustive use of the application such as the operational manager	Short-cuts make the exhaustive use of the application more efficient which results in extra time for the user
Visualization: Detailed	2.1.1	Operational management needs to know the exact allocation of resources and spends more time in the application.	By knowing the exact allocation of resources the operational manager can save the waste of resources.
Visualization: Uncluttered	2.1.2	Senior Management wants to spend little time in the application and only needs a high-level overview of resource allocation and costs.	A high-level check of resources allocation is sufficient for the higher management. The ability to do this fast will result in less waste of time (which is costly) from the senior management.

Table 4. Example hypotheses and segments

4.1.8 DECIDE ON WHICH PRODUCT CONFIGURATIONS ARE GOING TO BE (PARTIALLY) IMPLEMENTED AND TESTED

Once the hypotheses are stated per feature, all features are linked to a user segment. These segments should be coded in the feature tree. The coding is done by adding an extra symbol to every leaf tree in the table. For the example (see Figure 20). By coding per segment, the product manager has a better visual oversight for making decisions on implementation. If a leaf belongs to two segments, one should introduce a new color per combined segment. Alternatives for using color are the use of patterns or the coding of leaves per segment type. In case of a homogenous user base (one customer segment), all hypotheses should apply to the whole use base.

The feature tree is a preliminary attempt to visualize feature optionality. Other methods for coding feature optionality can be used to support the documentation and decision making in the LOOP method. With a feature tree, he product manager can compare potential features for implementation. At this point, it is also possible to identify gaps in the set of potential features. What if both customer segments are equally important but it is not possible to expose both to a different configuration of the product? For example, under the visualization node an additional feature could be added that is hybrid. In this feature information is initially depicted high-level but one could click through and open a more detailed information visualization. Maybe, this feature satisfies both user segments or maybe it

dissatisfies both or one of the segments. This could of course be tested by adding this feature to an iteration.

After the product manager has identified possible gaps and have selected a set of features from the tree, the next phase of the LOOP method can be initiated.

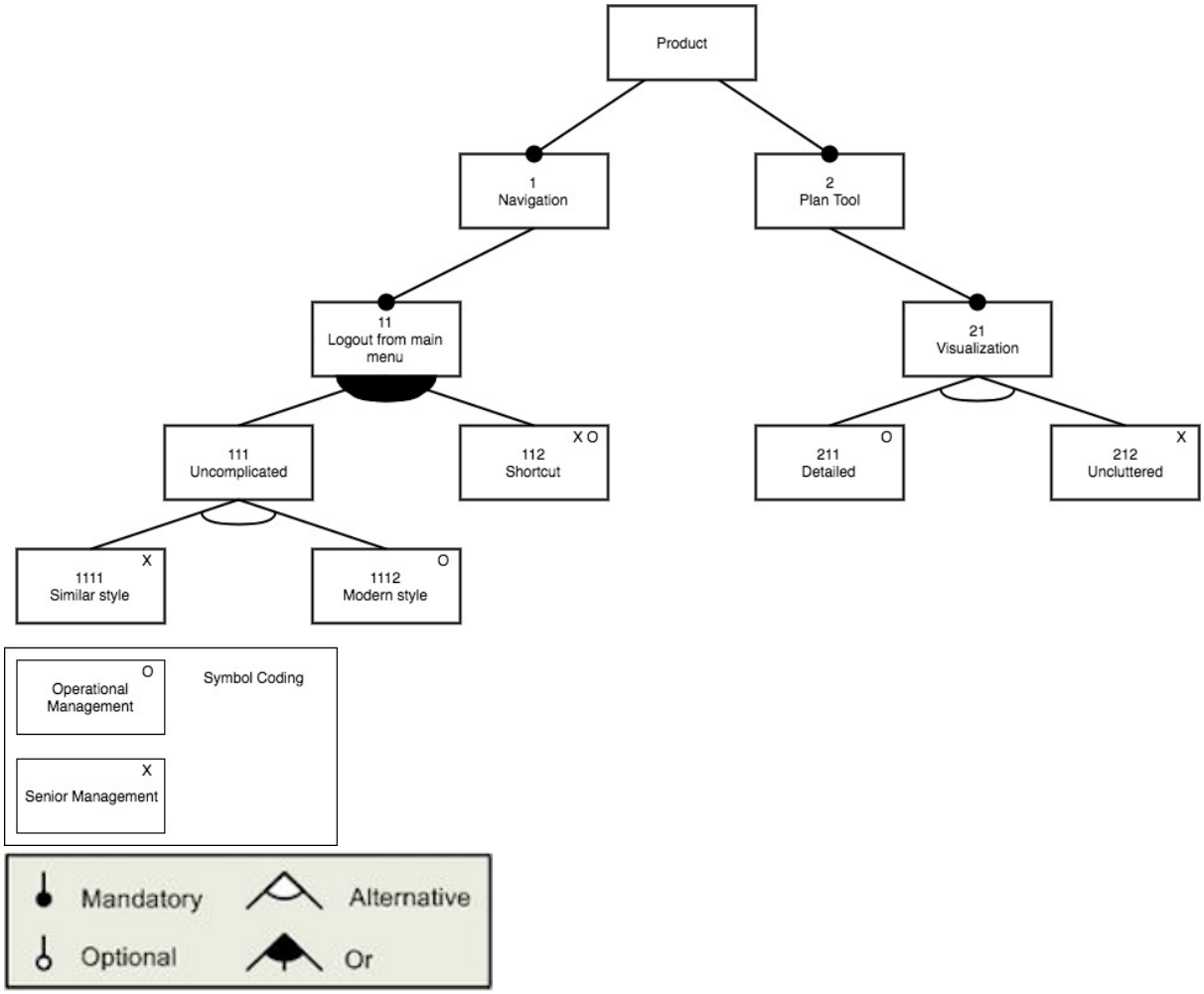


Figure 20. Leaf features with color coding per user segment.

4.2 LOOP PHASE 2: IMPLEMENTATION OF FEATURES AND DATA COLLECTION MECHANISMS

The input for the second phase is a set of features with their goal, description and hypotheses. This input will be transformed into two output elements:

- A set of implemented features within the software;
- A set of implemented data collection mechanisms.

Due to the situational characteristics of a software company's: personnel, requirements, application, technology, organization, operation, management and business (Clarke & O'Connor, 2012; Boehm & Turner, 2003), in this thesis, we will not cover the specific activities on how to implement features within the software. However, it should be considered during the implementation of the features that the LOOP method requires the software product to be configurable without excessive effort. Because of this, the features should be developed in a way that they are loosely coupled and modular. Otherwise, it will be harder to expose different configurations to different types of user segments. Furthermore, there should be an underlying infrastructure that supports the exposure of different configurations to different user segments. Only then, valuable insights on the feature hypotheses can be deducted.

The transformation from the set of features (holding a goal, description and hypotheses) to a set of (implemented) data collection mechanisms is based on the Goal Question Metric (GQM) method as proposed by Basili, Caldiera and Rombach (1994). The GQM method is suited because it can either be conducted exhaustively or as a lightweight method (preferable in an agile context) that allows the user to quickly translate feature goals and hypotheses in relevant data collection mechanisms (Wangenheim, Punter, & Anacleto, 2003). The original GQM method was introduced to aid with the measurement of organizational wide goals related to software development as a whole. We use a specific instantiation of the GQM method related to the measurement of feature goals. How this specific instantiation is executed will be explained in the elaboration of the second phase' activities.

According to the general GQM method, the goal is defined for an object that could be a product, process or resource. This goal should hold a purpose, issue, object type and a viewpoint. In context of our proposed LOOP method, an adjusted GQM method can be applied to both the "what" and the "why" of a feature (I.e., both can be classified as a GQM goal). The "What" holds a functional goal and the "why" holds a goal related to user perception. We have mapped the similarities and differences between the conceptual GQM goal and the solution method goal and hypothesis in Table 5.

GQM	LOOP method Goal	LOOP method Hypothesis
Purpose	What (functional change)	Why (change in user perception)
Issue	What (software quality)	Why (user satisfaction)
Object type (product, process or resource)	Product, process or resource	User perception
Viewpoint	Who (user segment)	Who (user segment)

Table 5. A mapping of goal concepts

During the second phase, the feature's goals and hypotheses will be further specified. The reason to specify the features extensively in this phase (rather than in the earlier phase) is that the product manager has now chosen to implement this feature. If features are extensively specified before they are approved for implementation, there is a risk that the benefits of this effort are never utilized.

In order to transform the input to the output of the FOD phase, the following activities should be conducted:

1. Ask questions about the hypotheses and goals
2. State metrics per feature that can answer the questions
3. Select data collection mechanisms for answering the questions
4. State the initial situation before the feature is exposed to the user
5. State the desired post-exposure situation of the feature
6. Implement/expose features and their data collection mechanisms

The following sections will elaborate on each activity. Subsequently, all activities but the last one will be clarified by means of an example that builds further upon the example of the previous phase.

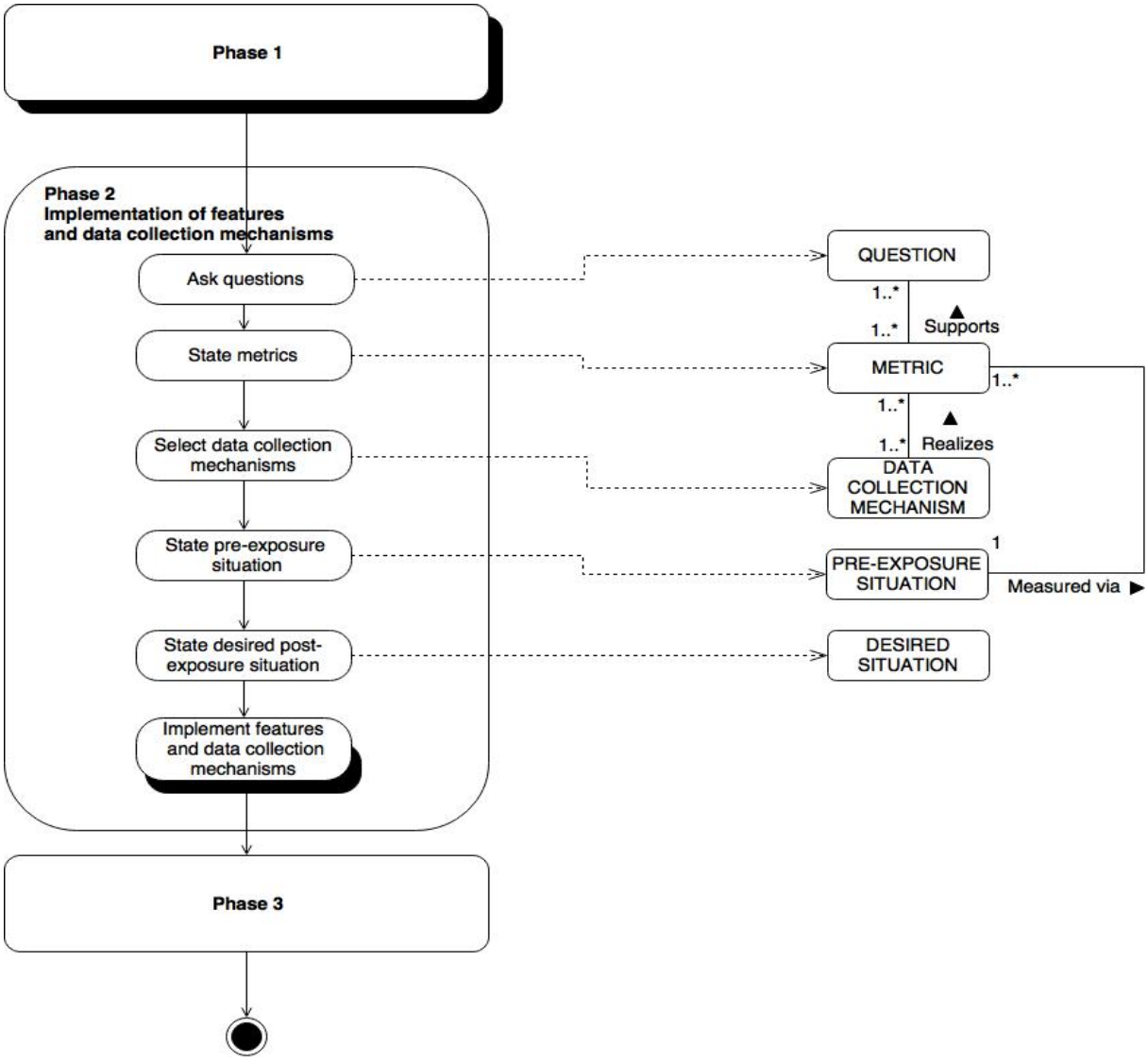


Figure 21. PDD LOOP phase 2

4.2.1 ASK QUESTIONS ABOUT HYPOTHESES AND GOALS

In order to tell whether the exposure of a feature to a specific user segment is effective, the delta between the initial and the desired situation should be analyzed. To measure this effectiveness, the right aspects of the feature's behavior should be examined. These aspects should be related to the level of fulfillment in terms of the feature's goal "What" and expected result of the feature's hypothesis "Why". For example, it would not make sense to focus at the change in aesthetic value of the product if a feature has the sole purpose to optimize user capacity.

One should start asking questions that characterize the feature's "What" and "Why" in a quantifiable way. The answers to these questions are needed to conclude whether the hypothesis is likely to be valid or not. Furthermore, by asking these questions, the development team makes the first step to find suitable metrics. These metrics are the means to answer the drafted questions. During the questioning, the context of the feature and relevant viewpoints (the user segments) should be taken into account.

To illustrate the application of questions to hypotheses, an example is given about the first two example features of phase 1. The goal of feature 1.1.1.1 is to give a new modern look to the interface but at the same time keep the old familiarity of navigation, whereby efficiency of use will not be reduced for any type of user. The hypothesis states that especially for people who do not use the product that often, for instance senior management, an increased level of satisfaction is expected. Questions which should be asked about this feature are:

1. Is the navigation interface still similar to the old interface?
2. Is efficiency of use not reduced?
3. Is user satisfaction of senior management increased?
4. Is user satisfaction of operational management the same?
5. Which user uses the product (to answer 3&4)?

The goal of feature 1.1.1.2 is to enhance the navigation speed by means of a new interface type. People who will frequently use the product will perceive a higher increase of satisfaction. Questions which should be asked about this feature are:

1. Is the new navigation interface faster than the old one?
2. Is user satisfaction of operational management increased?
3. Is user satisfaction of senior management the same?
4. Which user uses the product (to answer 2&3)?

4.2.2 STATE METRICS PER FEATURE THAT CAN ANSWER THE QUESTIONS

A set of metrics will be drafted for each feature so that the questions can be answered. We state the metric as a function that returns data about the specific behavior of a feature which answers the drafted questions. According to Basili et al., a metric can be objective or subjective. A metric is objective if it depends only on the measured object and not on the viewpoint from which it is taken. A metric is subjective when it depends both on the measured object and the viewpoint from which they are taken. If there is a high level of perceptual heterogeneity among one viewpoint (user segment), one should consider whether the right characteristics for segmentation are taken in consideration.

As can be seen in Figure 22, different metrics can be used as means to answer one or more questions which give an indication of how far a goal is achieved. As mentioned before, we separate the concept

of goal from Basili et al. into a feature's goal "What", hypothesis "Why" and user segment "Who". The validity of the "Why" and achievement of the "What" should be answered. For example, if the feature's "what" is to make a certain process faster, and the "why" is that this will make specific users ("who") more satisfied. Both the speed and increase in satisfaction should be measured by answering questions. Therefore, we have added an extra layer on top of the GQM model of Basili et al. (Figure 23). Although we do not advise to visualize the model for every feature due to time constraints, it is wise for the product manager to keep the relations between the model's elements into mind. Often, metrics for one feature can be reused for other features and therefore save resources. Figure 24 illustrates an instantiated GQM model for the goal, hypotheses, questions, metrics and data collection mechanisms that are specified for feature 1.1.1.1.

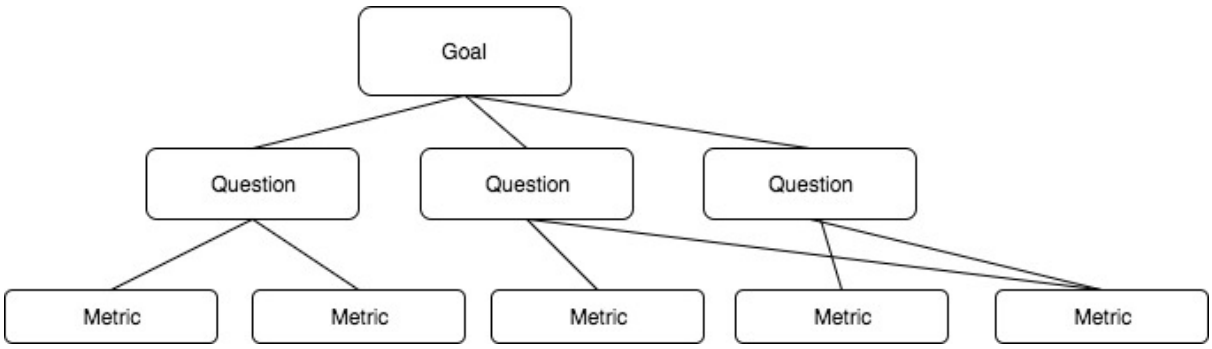


Figure 22. A GQM model (Basili et al., 1994).

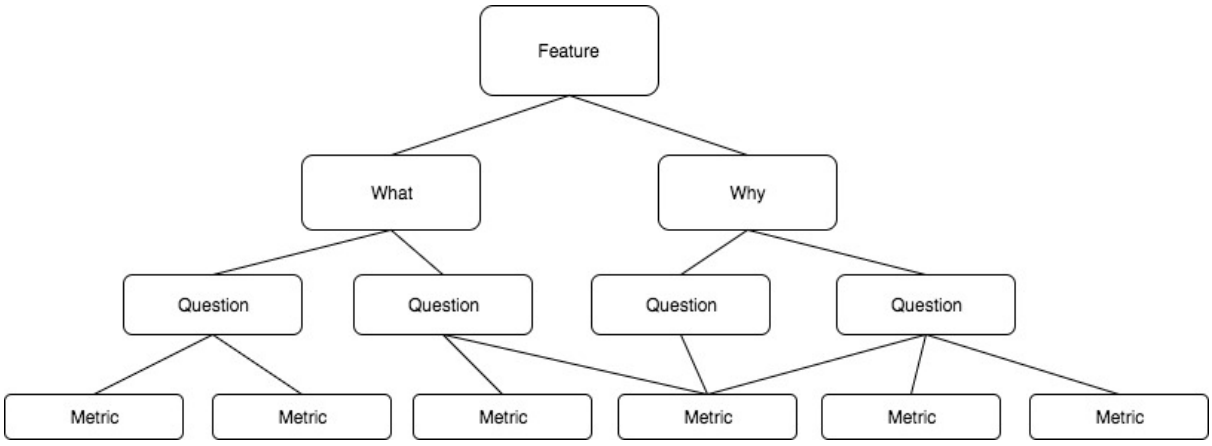


Figure 23. A general GQM model tailored to the LOOP method.

To create metrics for feature 1.1.1.1, one starts by investigating the questions. The first question is rather subjective and should be agreed upon by testers. The second question can be answered by measuring the average logout time per user segment. The third question is subjective and can be answered by asking the right user segment about their satisfaction. The type of user should be registered to answer the fourth question. The metrics per question will be:

1. Interface similarity
2. Average logout time per user segment
3. User segment satisfaction
4. User type (operational and senior management)

The same is done for Feature 1.1.1.2. Since both features are equipollent, they have a certain degree of overlap in questions and metrics. The first question is answered by measuring the average logout time per user segment. The second question is subjective and should be answered by asking the right user segment about their satisfaction. The third question can be answered by registering the type of user. Therefore, the metrics per question will be:

1. Average logout time per user segment
2. User segment satisfaction
3. User type (operational and senior management)

4.2.3 SELECT DATA COLLECTION MECHANISMS FOR ANSWERING THE QUESTIONS

Once the metrics are specified the product manager needs to select data collection mechanisms that provide data for answering the questions. Implementing these data collection mechanisms depends on the context of the product and company. A comprehensive list of possible data collection mechanisms:

- *Interviews (for subjective metrics)*: During an interview, the interviewer can address multiple new features, and can go into depth concerning the validation of stated hypotheses.
- *User observation (for subjective and objective metrics)*: By observing users that are exposed to new features, one can analyze whether the feature's goal is met and whether the user is satisfied with the new goal.
- *Theater sessions (for subjective metrics)*: If features are innovative, a company can choose to showcase the new features to users in a theater session. By doing this, direct feedback from the users can be acquired.
- *User bootcamps (for subjective metrics)*: User bootcamps are trainings where the usage of new features is explained to users. One can observe whether the feature's goal is met and if the user is satisfied with the new feature.
- *(in-product) Surveys and questionnaires (for subjective metrics)*: The opinion on new features is asked from a high number of users.
- *Logging (for objective metrics)*: Automatically measure product behavior to test the fulfillment of the feature's goal.

All the aforementioned techniques have their own benefits and drawbacks. Therefore, not every technique is suited for every context. Most mentioned techniques are used for validation but require user involvement and are intrusive. In most software development companies, the verification of goals is already widely executed by a test team. As mentioned earlier, the novelty of this thesis is that feature verification is linked with validation of an upfront stated hypothesis.

The data collection mechanisms which should be implemented in the product for feature 1.1.1.1 and 1.1.1.2 are:

1. Logging of user type
2. Logging of time per page
3. In-product survey that measures logout interface satisfaction

For feature 1.1.1.1 also an additional data collection is needed which cannot be built within the product. To answer the first question, a team should test whether the new interface is really similar to the old version.

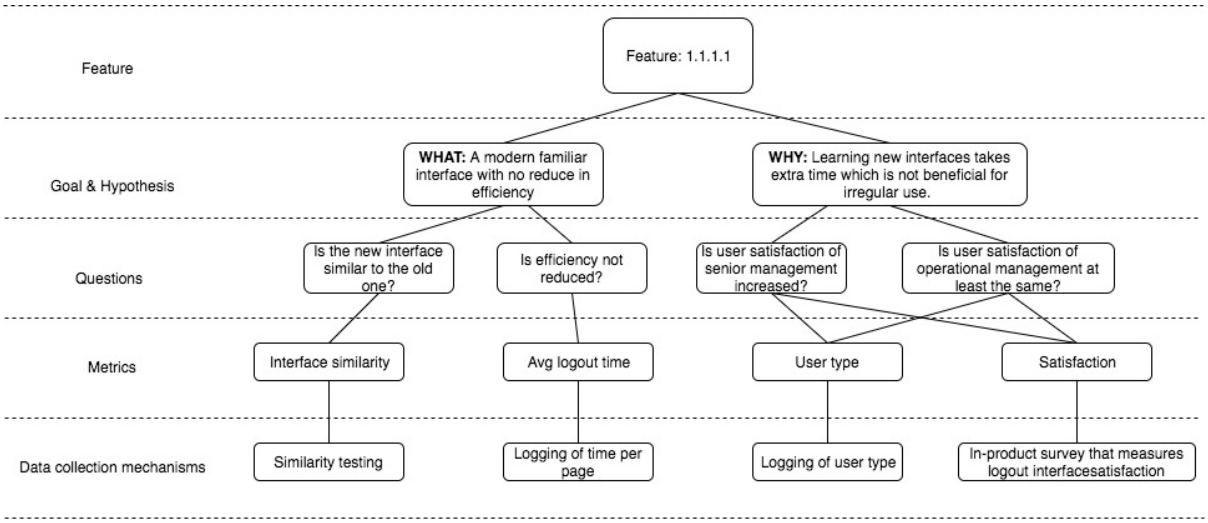


Figure 24. A specific GQM model for Feature 1.1.1.

4.2.4 STATE THE INITIAL SITUATION BEFORE THE FEATURE IS EXPOSED TO THE USER

Before a feature is implemented, the specific behavior with regard to the goal ("what") and user perception as a result of the hypothesis ("why") should be documented in terms of the drafted metrics. Because of these drafted metrics, one now knows which characteristics of the feature should be written down to state this specific behavior. For example, if a feature is selected that aims to make the navigation to the logout menu faster, the current average speed per user segment should be documented. If there is no data present concerning the current performance of the software, data collection mechanisms should be implemented that measure the current performance.

The initial state of performance is the benchmark that is required to eventually evaluate the features performance after implementation. The data about the initial situation should contain both the current perception per user segment and the current performance per feature. In some cases, measuring the current state of user perception is not suited because the introduction of the feature adds a new functionality in the software product which cannot be compared to the any old one. In these cases, one can only state desired post-exposure behavior.

4.2.5 STATE THE DESIRED POST-EXPOSURE SITUATION OF THE FEATURE

Before implementation, the development team should think about the post-exposure behavior of the feature in terms of the stated metrics. This is already informally captured in the previous phase. If the motives of executing the LOOP method are exploratory by nature, stating the post-exposure situation is not mandatory. (i.e., if a product manager wants to find out how a feature is doing but has no upfront idea or goal how it should be performing, stating the post-exposure situation is not mandatory). It is also possible to state the desired post-exposure situation only in terms of a subset of all metrics. This can be desirable if some outcomes are exploratory whereas others hold a certain threshold to be considered successful.

As an example, possible goals for both features are stated:
For feature 1.1.1.1 this contains a similar navigation together with an identical logout time per user type (the "what)" and a decrease in the dissatisfaction of 50% for infrequent users and 20% decrease for frequent users (result of the "why"). The feature is kept similar to not dissatisfy the infrequent users but the higher-level goal of 1.1.1 is to decrease dissatisfaction by implementing a modern uncomplicated look.

Feature 1.1.1.2 has the same higher-level goal as feature 1.1.1.1 (which is that of 1.1.1). The feature aims to reduce the dissatisfaction of the frequent users by 50 % and 30% for the infrequent user (the result of the "Why". However, for this feature, this is done by means of reducing the logout time (the "what").

The pre- and post-exposure situations are made explicit in Table 6.

4.2.6 IMPLEMENT FEATURES WITH THEIR DATA COLLECTION MECHANISMS

The former activities need to be conducted prior to development. It is not only wise to upfront specify how the feature will need to behave in terms of software and user perception; it is necessary because certain metrics need to be implemented in the code and therefore should be specified before development. As we have already stated, the specific way to implement features is outside the scope of this thesis.

Name	Code	Questions	Metrics	Data collection mechanisms	Pre exposure Situation	Post exposure situation
Uncomplicated: Old	1.1.1.1	Is the navigation interface still similar to the old interface? Is efficiency of use not reduced? Is the satisfaction increased? Which user uses the product?	Interface similarity Average logout time per user segment User segment satisfaction User type	Beta/ prototype testing to see whether people perceive the new interface as modern. Logging of user type Logging of time per page In-product survey that measures logout interface satisfaction	Avg logout time snr management = 8 seconds. Avg logout time opr management = 4 seconds. 60% of the infrequent and frequent user is not satisfied about the logout interface.	Same logout time per user. Only 10% of the infrequent users and 40% of the frequent is not satisfied with the logout interface. 40 %.
Uncomplicated: New	1.1.1.2	Is the new navigation interface faster than the old one? Is user satisfaction for frequent users increased? Which user uses the product?	Average logout time per user segment User segment satisfaction User type	Logging of user type Logging of time per page In-product survey that measures logout interface satisfaction	Avg logout time snr management = 8 seconds. Avg logout time opr management = 4 seconds. 60% of the infrequent and frequent user is not satisfied about the logout interface.	Avg logout time snr management = 5 seconds. Avg logout time opr management = 2 seconds. Only 10% of the frequent user and 30% of the infrequent user is not satisfied with the logout interface.

Table 6. Specified pre- and post-exposure situations per feature.

4.3 LOOP PHASE 3: VERIFICATION AND VALIDATION

During the third phase, the data from the data collection mechanisms are collected and analyzed. From this data, different conclusions can be drawn about: the feature's actual behavior; the fulfillment of the feature's goal and the validity of the hypotheses. With this knowledge, a reflection can be made on development rationale and decisions. This reflection can lead to organizational learning and possible better future decision making. The following set of activities should be conducted:

1. Conclude whether the goal is met in terms of functionality (verification)
2. Compare realized situation with initial situation and intended situation in terms of user impact
3. Conclude whether the hypotheses are likely to be valid (validation)
4. Document findings and decide on the feature's future lifecycle
5. Learn from wrong assumptions and reasoning

Ideally, the feature is implemented correctly and shows the desired effect on the user. However, not in all cases, the prerequisites are (completely) fulfilled for a feature in order to deliver value. If a feature does not deliver the intended value it can be due to (partial) failure of reaching the features intended functionality or a flaw in the rationale of the hypothesis. Feature goals that can be integrated up to a certain degree can partially fail in fulfilling the goal. Feature goals that have a binary nature of fulfillment can either be fulfilled or not. It is only interesting to look whether a feature has delivered the hypothesized value when its functional goal is fully or partially implemented. Therefore, the first task at hand in this phase is the verification of the functional fulfillment of the feature.

As mentioned before, the intended change in user perception is most likely only reached when the goal is realized and the hypotheses are valid. The change in functionality and user perception are measured by the implemented data collection mechanisms. Once the fulfillment of the feature's goal is indeed realized, it can be tested whether it also has led to the desired effect on the user's perception. In the ideal situation, the effect is exactly or better as anticipated. One can then conclude that the feature adds value to the user at that moment in time and, more important, that the rationale for choosing this feature was most likely to be correct (it is never possible to be absolutely certain that a hypothesis is correct since there could always be another reason why the user perceives value). However, if the functional goal is realized but the intended effect on the user is not met, one should dissect the hypothesis as described in following section.

When it is hypothesized that there will be a change in user perception once a feature is exposed to the user, it holds the positive sub-hypotheses as shown in the Feature Value model (Figure 19). When the feature's goal is indeed implemented but the user is not perceiving long lasting value, one or multiple sub hypotheses can be invalid. Examples of reasons for not delivering value to the user can be:

- No awareness of the feature (no awareness H1-);
- The user's perception of the feature's functionality holds no benefits (awareness but no use H2-);
- The user uses the feature but it does not fully meet up to his desires (use but no satisfaction H3-);
- The user uses the feature and is satisfied but only for a short time because the feature is not as useful as expected. (short-term satisfaction H4-).

Furthermore, it could be possible that there is a relation between implemented functionality and user perception, but that the expressed relationship is not fully correct. For example, users become more satisfied when there is a novel functionality but it is not the desired delta (see Figure 18). In this case, one can use this knowledge in further decisions on the feature's lifecycle. To illustrate, it could be a possibility that the gained value does not outweigh the reduction in performance of the product.

Under any circumstances, knowledge and possible new hypotheses about the feature are gained. With this knowledge, the product manager can decide to further invest in the feature or not. It could be a scenario that the feature fully functions but the users are not satisfied. By looking at the acquired data, the product manager found out that users are not using the feature. He realizes that the assumption that users will find the new feature without guidance may not be valid. A possible action to undertake could be sending a mail notification that introduces the new functionality. If the use of the feature then increases together with the satisfaction of the user, all other sub hypotheses were valid. If this would not be the case, the product manager can choose to further investigate the invalidity of hypotheses or suspend the feature. The performance of the feature in terms of its change in user perception should always be document for future decision making.

The true value of the LOOP method does not solely lay in gaining knowledge about a feature. When a new feature idea arises, it is based on a rationale posed by certain stakeholders. Not only the functionality of the feature is tested but also the underlying rationale is made explicit and validated. By reflecting upon this rationale, one can find out where mistakes are made in the making of assumptions concerning features and users. The LOOP method combines business, development and operations and let the executor reflect upon decisions made in all these areas. By learning from this feedback, one can not only realize better decisions about the specific tested feature, but also make better future assumptions, and thus decisions, concerning the introduction, or deletion, of other features. This will lead to a strong feature portfolio of the software product.

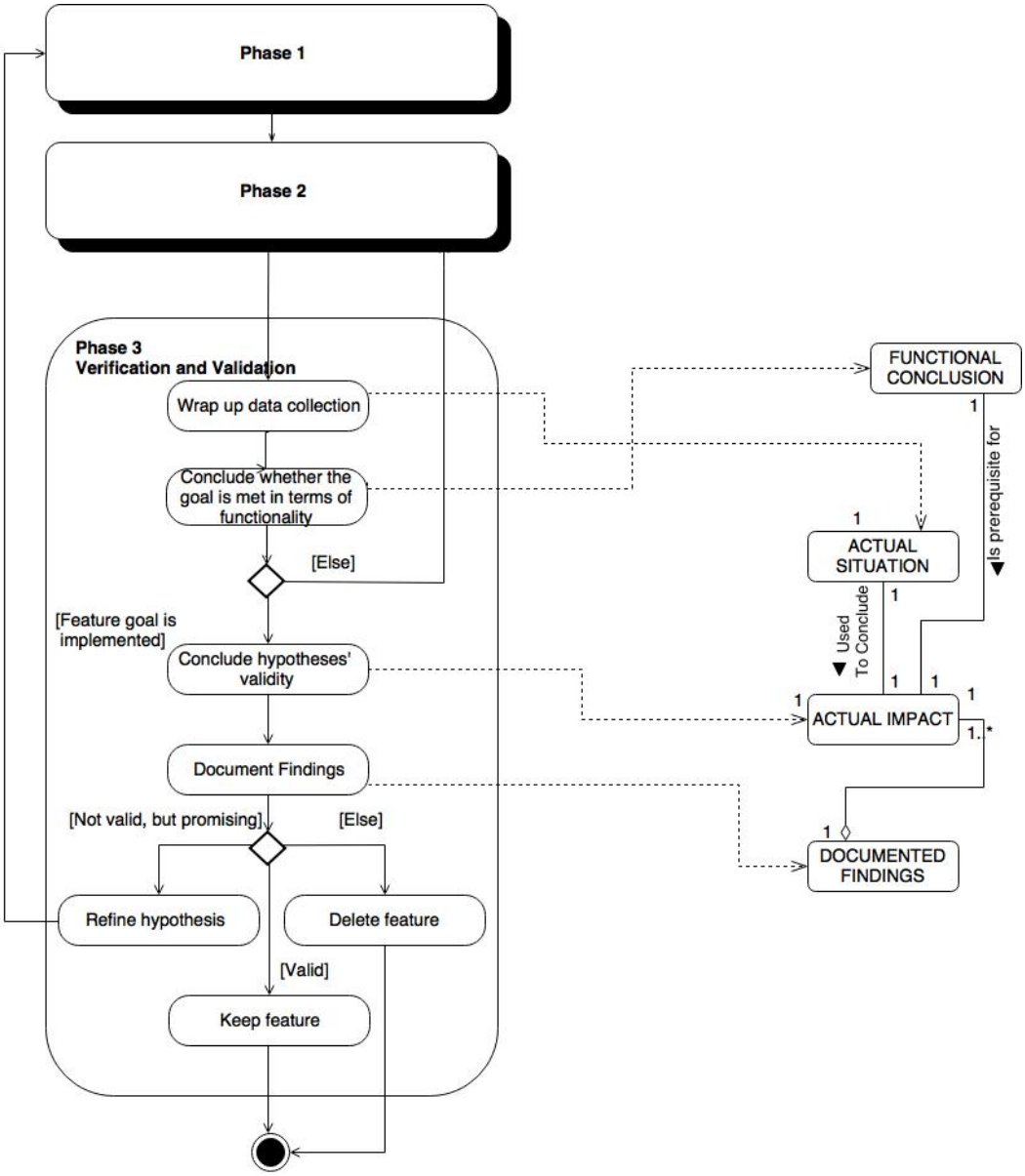


Figure 25. PDD LOOP phase 3

5 LOOP CASE STUDY

This chapter introduces the case company and elaborates upon the application of the LOOP method in the context of this case company.

5.1 COMPANY

The coined LOOP method has been conducted at a small size software company named Retail Solutions (RS). RS has one product which is called PMT which is a cloud based personnel management tool for retail stores. The product differentiates itself from similar products in the market because it not only supports the planning of personnel, but also offers legislations checks (collective labor agreements) and shows the planned versus actual costs and revenues per department of a retail store. Furthermore, management/planner functions are not the only type of user, the other personnel in retail stores use PMT to get insight about their work schedule via a browser based interface or mobile application.

RS has existed for seven years and is rapidly growing with a current workforce of 16 FTE that is partly offshored. The company holds a big market share in the Dutch supermarket sector (>750 retail stores). The software is licensed to both full supermarket chains as well to individual franchisers.

The situational factors of RS make it a suitable company for a case study. Firstly, because the product is cloud based which makes it easy to collect user data. Secondly, the company has a relatively big customer base compared to its workforce. This makes it hard for the management to get a clear vision on their different customer segments. Lastly, PMT is still a growing product and the management of RS is considering different paths for the product to go to in terms of functionality. The created method can help in making decisions considering the PMT's feature portfolio.

5.2 APPLICATION LOOP PHASE 1

5.2.1 INDICATE SUITABLE FEATURES FOR THE METHOD

During the meetings with the product manager and owner of Retail Solutions, a wide set of potential features for the LOOP method were mooted. However, due to time constraints two features were chosen that were already implemented in the software product but were not exposed to all users (in the right manner).

The first feature, to which we from now on will refer to as the *"schedule integration feature*", was already available to all users but was hard to find within the software product. This feature facilitates the possibility to synchronize your work schedule with external calendar applications such as Google Calendar and ICal.

The second feature, named *"activity dashboard"* was also already implemented within the software but is only exposed to a specific set of users. This feature is a functionality on the landing page of PMT, and shows a dashboard with pending tasks the user has to fulfill for that day and week within the product.

5.2.2 GATHER STAKEHOLDER OPINIONS ON FEATURES

Before the *schedule integration feature* was introduced, both the product manager and owner of Retail Solutions had high expectations for this feature (they expected it to be a genuine "Attractor", (see Figure 14). However, since the feature was implemented, only a small part of the user base has been using it. The owner and product manager suspect that the feature is not easy findable within the product and therefore users are not aware of it. They want to determine whether this is the case, or rather users are aware but just choose to not use the application.

The *activity dashboard* feature has been a demand of a few customers and is implemented for this specific set of users. Both the product manager and owner want to know whether the feature is desired by other users and if the satisfaction is long-lasting.

5.2.3 DIVIDE USERS IN DIFFERENT SEGMENTS

Retail solutions offers their product to a wide variety of supermarkets. For certain features, it is important to segment the user base on a customer level. For some type of supermarkets, the full supermarket chain is a customer and all supermarkets within the chain can use PMT. Retail solutions also has a great set of individual supermarket proprietors which directly buy a license. For these reasons, the users can be segmented on the type of store and whether or not they are part of a chain or individual store owners.

The users for a given store are divided per job function. This segmentation is compulsory because different job functions hold different levels of feature authorization. The different type of users based on their job role are:

- Admin;
- Supermarket manager;
- Team leader;
- Planner;
- Standard employees;

5.2.4 DEVELOP THE FEATURE TREE

The feature tree depicts the different possible configurations the software product can have.

The *activity dashboard* feature is placed on the landing page of PMT. The *schedule integration* feature is a feature which is part of the work schedule feature and therefore is a child of the work schedule feature. We have chosen to depict the different navigation possibilities as alternative leaf nodes. This is to clarify the difference between product configurations with the current navigation and emphasized navigation for finding the *schedule integration* feature. On the higher level the main other main features of PMT are depicted. The Employee page shows all the functionalities which are accessible to all users. The Department page facilitates schedule management per store department. The Store page facilitates all the management concerning the store such as: employees, distribution of schedules, KPI's and store rapports. The organization page is for clients that are part of a supermarket chain and facilitates configurations on level of the full chain.

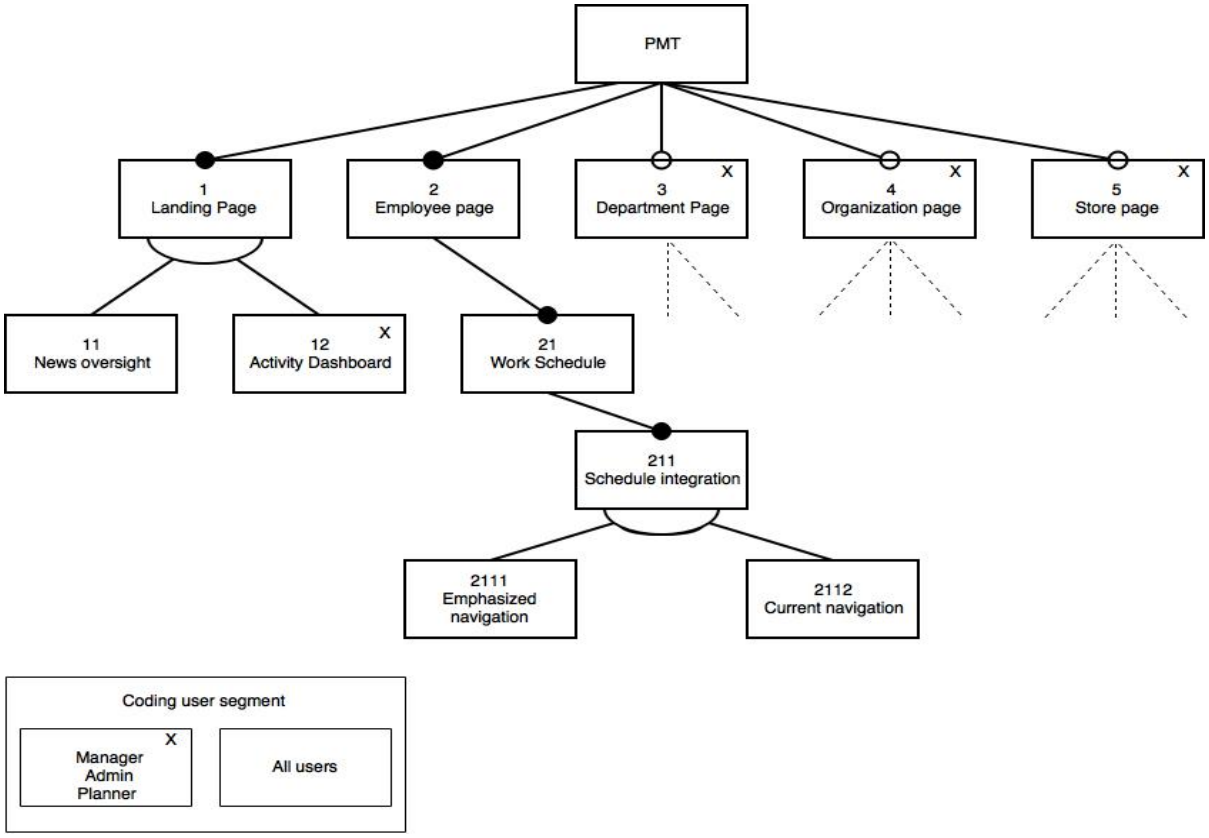


Figure 26. A feature tree of PMT.

5.2.5 ADD A FEATURE GOAL TO EVERY FEATURE

The goals and descriptions for both features are made explicit in this section.

ACTIVITY DASHBOARD

Goal 1:

- A dashboard that shows pending tasks to the user so they can easily see, and navigate to, tasks which need to be fulfilled.
- Qualities: Efficiency, Usability.

Description:
Two tables are shown to the user. The first one contains the upcoming weeks, showing pending activities that the user needs to execute. The other table is similar but shows the pending activities per day in the upcoming week. All activities hold a link which navigates to the feature where these activities can be executed. Figure 27 is a screenshot of the activity dashboard with on top the week oversight consisting of different tasks and at the bottom the daily oversight.

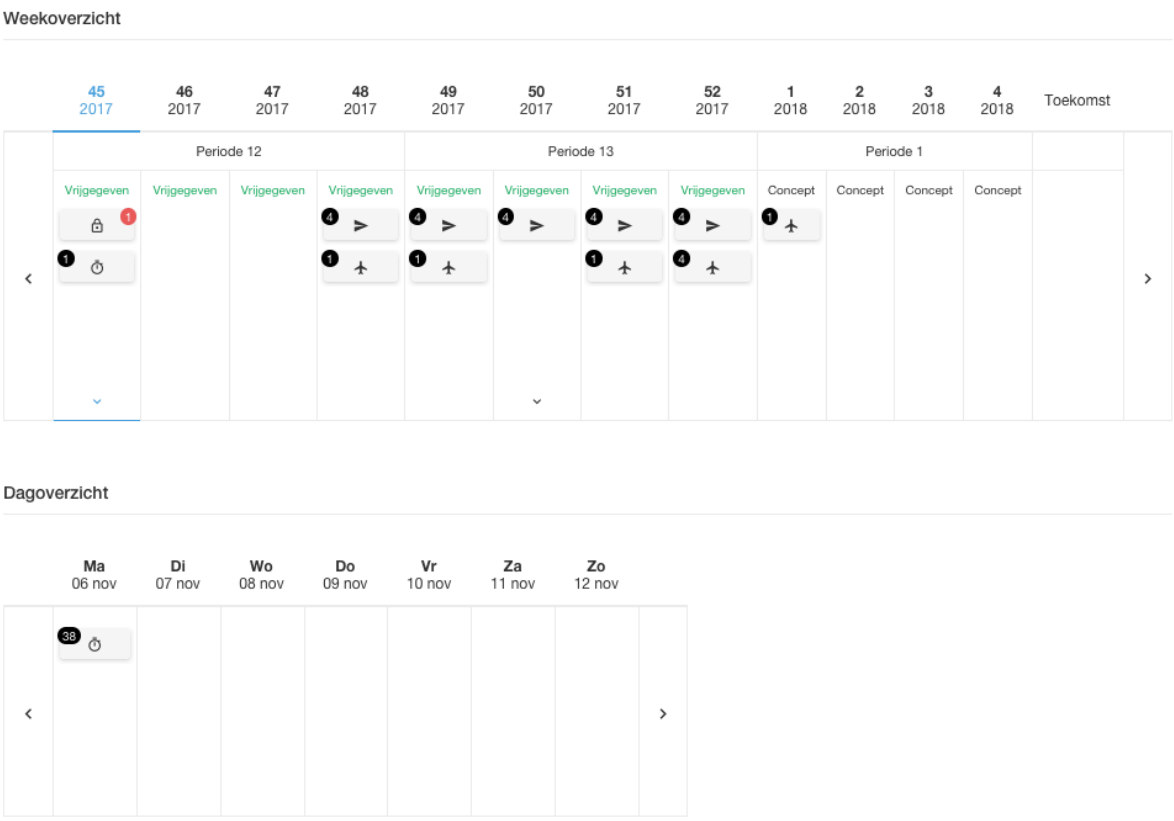


Figure 27. A screenshot of the activity dashboard.

SCHEDULE INTEGRATION

Goal 1:

- A functionality that allows the user to integrate their work schedule with external calendars such as google calendar, Ical, etc.
- Qualities: Interoperability, Usability.

Description:
This feature lets the user integrate his working schedule with an external calendar. The feature is already built within the product. However, the product manager of RS suspects that this feature is currently hard to find for users. By introducing the feature to users who do not use it via a direct link from schedule tab to the feature's location, usage is expected to increase.

5.2.6 ADD HYPOTHESES

The hypothesis links the goal of a feature to change in user perception

ACTIVITY DASHBOARD

Who
Users of the type supermarket manager and planner.

Hypotheses
H1: By showing the tasks at hand, the user gains more oversight and navigates faster through PMT. This will increase efficiency and therefore satisfaction among the targeted users.
H2: The satisfaction of the activity dashboard is long lasting.

SCHEDULE INTEGRATION

Who
All users except admins.

Hypotheses
H1: Users of the type employee are not well aware of the functionality.
H2: By introducing them to the functionality more employees will use it.
H3: There will be an increase in satisfaction when employees become aware.
H4: This increase in satisfaction is long lasting (not possible in the time scope of this study).

5.2.7 DECIDE ON WHICH PRODUCT CONFIGURATIONS ARE GOING TO BE (PARTIALLY) IMPLEMENTED

Due to the scope of this study only the two mentioned features are going to be implemented.

5.3 APPLICATION LOOP PHASE 2

5.3.1 ASK QUESTIONS ABOUT THE HYPOTHESES AND GOALS

The questions which should be asked to answer whether a feature fulfills its goals and hypotheses are stated in the tables below (Table 7 & 8). Each table lists the identifier of the question in the left column (Question #). In the middle column (Question), the actual question is depicted. The right column (Goal/hypothesis) contains the goal or hypothesis to which the question is linked. The goals for both features are worked out together with their questions, metrics and data collection techniques. However, the goals are only included for clarity. Since these features are already tested and implemented within the software, stating these elements is unnecessary.

ACTIVITY DASHBOARD

Question #	Question	Goal/Hypothesis
1	Is the list with pending activities correct?	G1
2	Do the navigation links from the pending activities to the related functional features work correct?	G1
3	What is the satisfaction of users concerning the feature?	H1
4	Which type of users are exposed to the feature?	H1, H2
5	Is there a difference between newly exposed users and those who are longer exposed?	H2

Table 7. The question table for the activity dashboard feature linked to the goals and hypotheses.

SCHEDULE INTEGRATION

Question #	Question	Goal/Hypothesis
1	Does the integration work with all the targeted agendas?	G1
2	Which and how many employees do not use the feature?	H1, H2
3	What is the current satisfaction of employees who are not aware of the feature?	H3
4	Which and how many employees do use the feature?	H1, H2
5	What is the current satisfaction of employees who already do use the feature?	H4
6	What is the degree of satisfaction for employees who have started using the feature after introduction?	H3, H4
7	What is the difference in satisfaction between employees who already used it and those who started using it after exposure?	H4
8	What is the increase in satisfaction from employees who start using the feature after introduction?	H3
9	What is the increase in numbers of employees who start using the feature after introduction?	H1, H2

Table 8. The question table for the schedule integration feature linked to the goals and hypotheses.

5.3.2 METRICS

The metrics are ordered in the following tables (Table 9 & 10). In the left column (Metric #), the ID of the metric is stored. In the middle column (Objective/ Subjective Metrics), the metrics are stated which help answering the questions that are stated in the right column (Question #) by their ID number.

ACTIVITY DASHBOARD

Metric #	Objective Metrics	Question #
1	The passing of all functional tests.	1,2
2	*User type*: The type of user should be stored in terms of the customer supermarket. If the customer supermarket is known, it can be deduced whether the user is newly introduced to the feature or not. This is eventually important to answer H2.	3,4,5
	Subjective Metrics	
3	*Feature satisfaction*: The satisfaction asked to every user concerning the activity dashboard.	3,5

Table 9. The metrics table for the activity dashboard feature linked to the questions.

SCHEDULE INTEGRATION

Metric #	Objective Metrics	Question #
1	The passing of all functional tests.	1
2	*The user ID & Type:* The user ID is important because the data from different campaigns can be linked via the user ID. The type helps in segmenting.	2,3,4,5,6,7,8,9
3	*Feature usage:* The number of users that uses the feature.	2,3,4,5,6,7,9
4	*Start time usage:* By recording the time a user starts using the schedule integration, the effect of exposure to the usage can be measured.	5,6,7,8,9
	Subjective Metrics	
5	*User satisfaction:* What is the user's satisfaction concerning scheduling functionalities	3,5,6,7,8

Table 10. The metrics table for the schedule integration feature linked to the questions.

5.3.3 SELECT ADDITIONAL DATA COLLECTION MECHANISMS FOR VERIFICATION AND VALIDATION

For both features the data collection mechanisms are stated in the following tables (Table 11 & 12). In the left column (Data collection mechanism) the data collection mechanism is stated. In the right column (Metric #), the metric id is stated to which the data collection mechanism is linked.

ACTIVITY DASHBOARD

Data collection mechanism	Metric #
Acceptance tests	1
Logging the user ID/ Type	2
Pop-up survey that asks for the user's satisfaction concerning the activity dashboard. (thumbs up, thumbs down)	3

Table 11. The data collection mechanism table for the activity dashboard feature linked to the metrics.

SCHEDULE INTEGRATION

Data collection mechanism	Metric #
Acceptance tests	1
Logging the user	2
Logging the feature's usage	3,4
Pop-up survey that asks for the user's satisfaction concerning scheduling functionalities after logging in. (thumbs up, thumbs down)	5

Table 12. The data collection mechanism table for the schedule integration feature linked to the metrics.

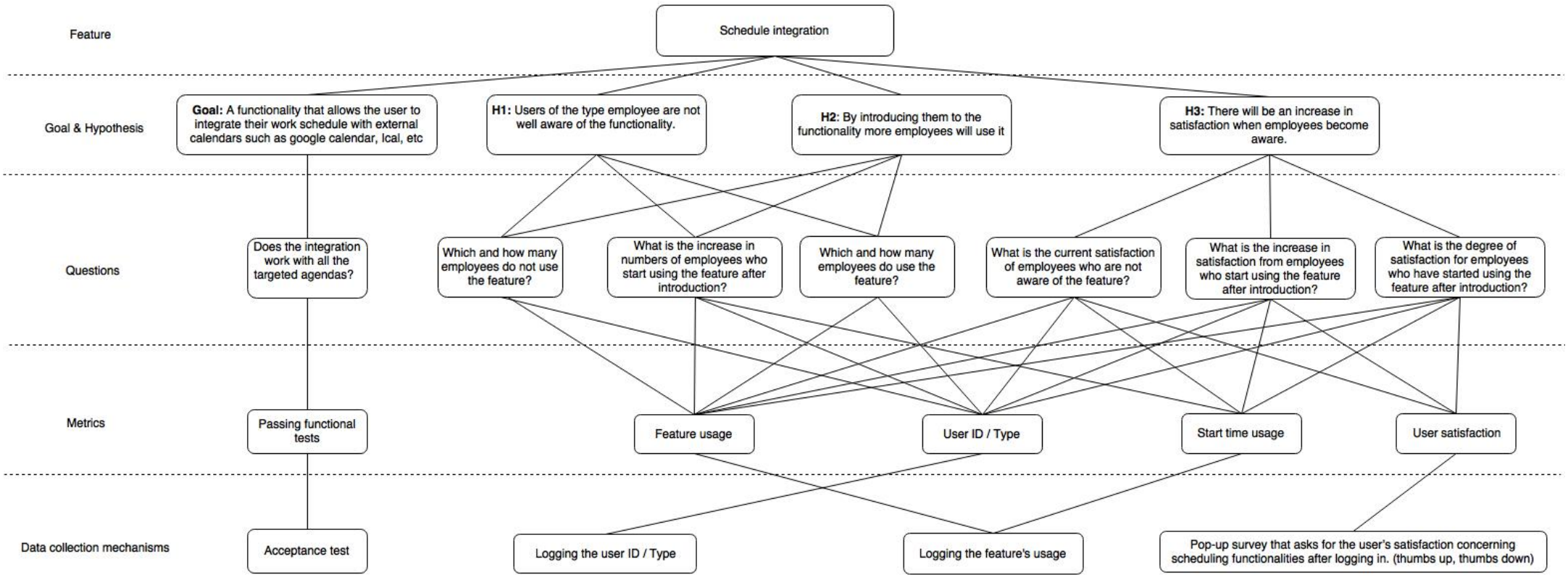


Figure 28, a GQM model of the schedule integration feature.

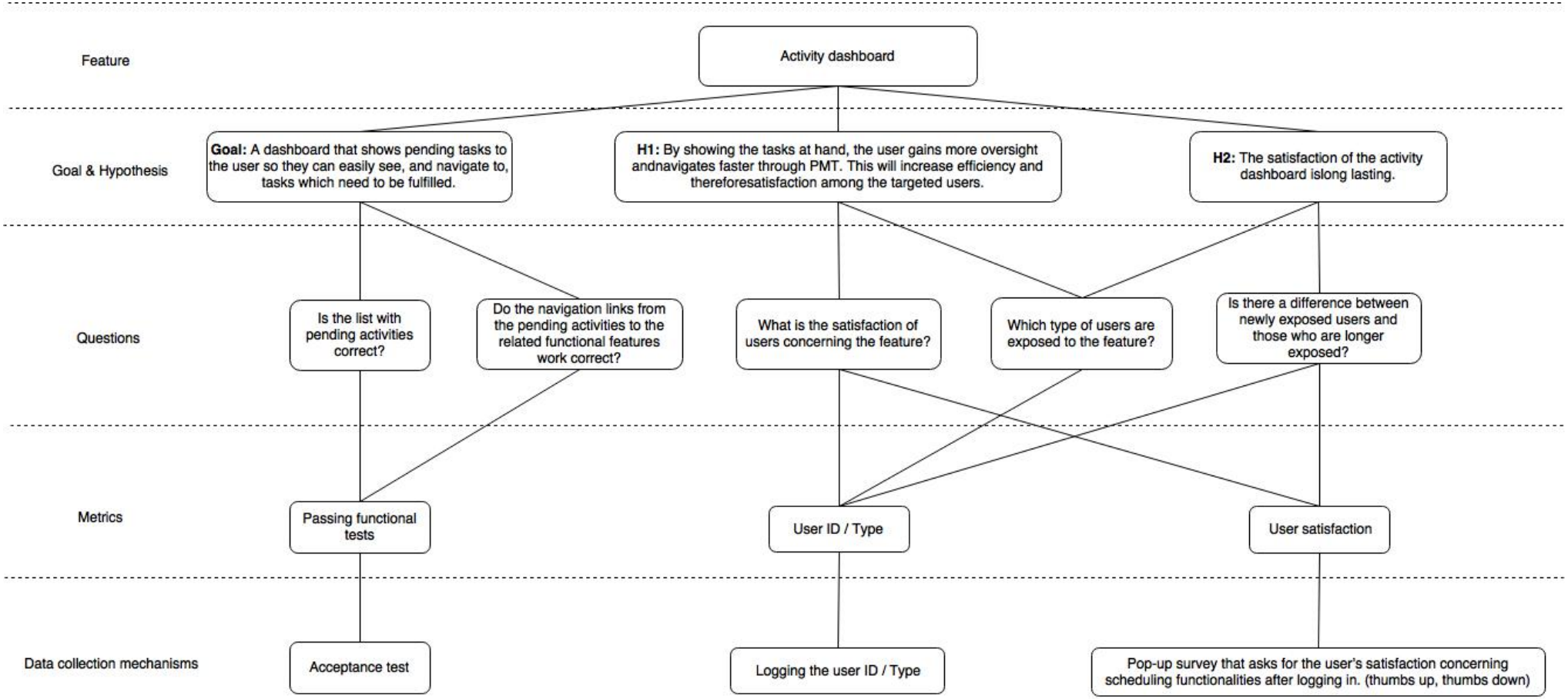


Figure 29, a GQM model of the activity dashboard feature

5.3.4 STATE THE INITIAL SITUATION BEFORE THE FEATURE IS EXPOSED TO THE USER

Because this is the first iteration that RS executes the LOOP method, no data about the initial situation is known. For the activity dashboard feature this is no problem because it is exploratory by nature. The schedule integration feature, on the contrary, needs an initial situation description. Therefore, after all the data collection techniques are implemented, data about the initial situation will be collected during the first couple of days of the data collection period (Figure 30). The results of this data collection will be elaborated in Section 5.4.

5.3.5 STATE THE DESIRED POST-EXPOSURE SITUATION

The product manager of Retail Solutions has stated the following desired post-exposure situations. Because this is the first execution of the LOOP method at RS, this estimation is intuitive. If a person/company has executed the method more often, it is possible to extrapolate recorded data and see what a realistic desired post-exposure situation for a feature is.

ACTIVITY DASHBOARD

The activity dashboard is considered successful if at least 70% of the answers is positive (thumbs-up).

SCHEDULE INTEGRATION

The possibilities to see your schedule is considered successful if at least 60% of the answers is positive.

The notification to make users aware of the schedule integration functionality is considered to be successful if there is an increase in 20% in positive answers after exposure. For example, if the current satisfaction would actually be 60%, the notification will be considered successful if 80% of the answers will be positive after exposure.

5.3.6 IMPLEMENT FEATURES WITH THEIR DATA COLLECTION MECHANISMS

Due to the time limitations of this study, the chosen features were partially chosen because they were already implemented. However, the feature that notifies users about the schedule integration and other data collection mechanisms needed to be built within the software. Because both features are already implemented, they are already tested and thus verified.

The software architect of RS designed a back-end feature that makes it possible for the product manager of RS to create campaigns. A campaign is considered as the in-product mini survey to ask for a user's opinion. (Figure 30) The parameters for the campaign are the question, the way to answer the question, the start and end date of the campaign. The feature automatically stores the answers and relevant accompanying data in a database. The creation of such a feature/infrastructure is important because the LOOP method can now be repeated a with minimal usage of resources. An additional benefit of this feature is that also the notification of the schedule integration feature could be done by means of a campaign. In this campaign, the user was not shown a question but a link to the schedule integration feature. This campaign also stores which users do or do not click the link.

Figure 30. An example campaign in PMT.

PMT already facilitates the functionality to store user-ID and the status whether a user used the schedule integration or not.

The data is collected with a timespan of one month (also one sprint within RS). At the start of this month, three supermarket chains were already using the dashboard. The activity dashboard was deployed to the targeted user segments of other customer supermarket chains on the first day until the last day of the month (Figure 31). After five days, everybody who has access to the dashboard activity feature received an in-product campaign question named: *"How would you rate the activity dashboard?"*. Users were exposed to this question until the end of the month.

On the first day of data collection, users were asked the question: *"How would you rate the possibilities to see your schedule?"*. This continued for eight days. After these eight days, all users were exposed to the campaign that pointed out there was a feature to integrate their schedule with their agendas. The campaign showed the question: *"Would you like to integrate your work schedule in your agenda*?" together with an accompanying link directing to the feature. The users were exposed to this campaign for another eight days. Subsequently, the users were exposed to the campaign asking: *"You have clicked the link, and found the schedule integration with your own calendar. How would you rate the possibilities to see your schedule now?"* until the end of the month.

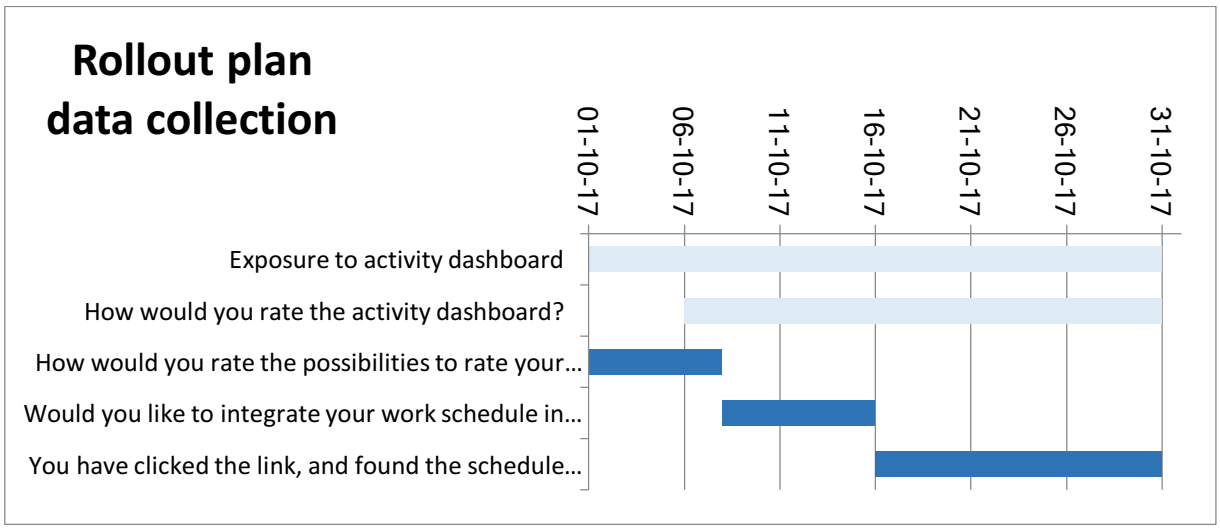


Figure 31. A Gantt chart of the data collection rollout plan.

5.4 APPLICATION LOOP PHASE 3

5.4.1 CONCLUDE WHETHER THE GOAL IS MET IN TERMS OF FUNCTIONALITY (VERIFICATION)

Both features are already tested and deployed within PMT (the activity dashboard was not yet exposed to all customers). However, the feature that facilitates the creation of campaigns, and stores customer feedback in a database, still needed to be created and tested. At the moment of implementing the LOOP method at RS, the basic functionalities for creating campaigns and collecting customer feedback were working and tested.

5.4.2 COMPARE REALIZED SITUATION WITH INITIAL SITUATION AND INTENDED SITUATION IN TERMS OF USER IMPACT

ACTIVITY DASHBOARD

For the activity dashboard, no initial situation was observed. Therefore, only the intended and realized situation can be compared.

As stated in the section of phase 2, the activity dashboard is considered successful if 70% of the users answers with a positive reaction.

148 users replied to the campaign. From this set, 55 users (30 familiar users and 25 new users) answered negative, and 93 users (58 familiar users and 35 new users) answered positive. Therefore, the actual situation is that 63% of the people liked the activity dashboard and 37% did not like it.

Additionally, a chi-square test was conducted to check whether there is a significant difference between familiar and new users of the activity dashboard. The result of the chi-square test was a P value of 0,35 which concludes that there is no significant difference between these two groups of users.

SCHEDULE INTEGRATION

This feature is considered a success if at least 60% of the users gives a positive answer. The notification of the feature is considered successful if at least an increase of 20% is realized.

For the schedule integration, the initial situation was measured at the beginning of the data collection period (Figure 31). Hereafter, the users were exposed to the campaign which notified them about the agenda integration and offered a navigation link to this feature (*"Would you like to integrate your work schedule in your agenda*?"). In total 6082 users answered this campaign from which 4434 users (73%) clicked the link and accessed the schedule integration feature.

The users that answered the first campaign and clicked the link to the navigation were asked the same question as the first campaign again with a small explanation (*"You have clicked the link, and found the schedule integration with your own calendar. How would you rate the possibilities to see your schedule now?"*).

Before the notification, 6411 users used the schedule integration. The notification resulted in an increase of 2786 users which resulted in a total of 9197 users using the feature. The total number of

users is estimated on 50.000. Therefore, we conclude that the notification resulted in an increase of approximately 6 % of the total user population.

The total number of users that answered to both campaigns and clicked the link is 465. This set of users is an interesting sample because a comparison can be made between the pre- and post-exposure situation concerning the schedule integration notification. We do want to highlight that this sample is therefore selective and non-representative for the whole user population. Before the schedule integration notification got exposed, 393 users answered positive (85%) and 72 answered negative (15%). After the notification, 341 answered positive (73%) and 124 answered negative (27%). This means that there was a shift of 12 % to more negative answers after the notification. The change was checked for significance by means of a chi-square test. We conclude with a P value of 2,9037E-05 that there is a significant negative shift in this group after notification.

5.4.3 CONCLUDE WHETHER THE HYPOTHESES ARE LIKELY TO BE VALID (VALIDATION)

ACTIVITY DASHBOARD

H1: *By showing the tasks at hand, the target user has more oversight and navigates faster through the software. This will increase efficiency and therefore satisfaction among the targeted users.*

We conclude that the feature probably is appreciated by users because they experience the benefits of it. However, it should be noted that the amount of people who appreciate the feature is 7% lower than the expected 70%.

H2: *The satisfaction of the activity dashboard is long lasting.*

The chi-square indicated that there is no significant difference between users (P=0,35) who already used the feature for a considerable time and newly exposed users. This indicates that the satisfaction of the feature is consistent.

SCHEDULE INTEGRATION

H1: *Users of the type employee are not well aware of the functionality.*

73% of the users showed interest when they were notified about the schedule integration. This is a considerable amount of people that clicked the link. We assume that indeed a big part of the user base was not aware of the schedule integration feature.

H2: *By introducing them to the functionality more employees will use it.*

We concluded that the notification had a considerable effect numbers of users that used the feature (increase of 6% of the total user population). Therefore, we assume this hypothesis valid.

H3: *There will be an increase in satisfaction when users become aware of the schedule integration feature.*

The number of satisfied users who answered both campaigns, and clicked the notification link to see the schedule integration feature, dropped with 12% significantly (P = 2,9037E-05). This unexpected result is a direct lead for further investigation. As mentioned before, this specific sample of users is not representative for the full user base. However, we can conclude that the notification of the schedule integration feature does not result in an increase of satisfaction.

5.4.4 DOCUMENT FINDINGS AND DECIDE ON THE FEATURE'S FUTURE LIFECYCLE

ACTIVITY DASHBOARD

We have concluded that all hypotheses concerning the activity dashboard were valid and as expected. Users do indeed appreciate the feature and this appreciation is long-lasting. The product manager of RS concluded that it is beneficial to keep this feature in the upcoming product versions and keep investing in it.

SCHEDULE INTEGRATION

The first two hypotheses seem to be valid. However, the schedule integration feature is not as appreciated as was anticipated by the users. The product manager of RS has decided to not further invest in the schedule integration feature and also maybe start to phase the feature out in future releases of the product.

5.4.5 LEARN FROM WRONG ASSUMPTIONS AND REASONING

The activity dashboard mainly confirmed the expectations of RS' product manager and owner. Although positive feedback on correct reasoning is very valuable, it confirms current ways of thinking and does not result in new insights.

The schedule integration feature on the contrary, initiated a new way of thinking for RS' product manager and owner. The notification showed to be a means to put emphasize on a feature and that not many people were aware of an agenda integration functionality. The product manager and owner learned from this that the place where they position a feature in the software is very important. Furthermore, they concluded that putting emphasize on a feature by means of a notification does indeed work and let more people use it. They find this knowledge valuable since they want to put emphasize on new functionalities within future product releases.

The method has shown that users currently are very satisfied with the possible ways to see their schedule. However, the product manager and owner of RS did not anticipate on a negative appreciation by people who found the schedule integration feature after notification.

The owner of RS hypothesized that it could be a possibility that people who are notified about something have higher expectations than people who find a feature by themselves. The product manager of RS gained a different revelation and realized that they have never thought about how much of the targeted users had an external calendar to integrate their schedule with. It could also be a sign of frustration from users that they were linked to a feature which they could not use due to not having an external calendar. Most users who work at supermarkets are teenagers between 15 and 21 years old and could possibly only have a school agenda. This realization was a reason for the product manager to do future market research on this topic. Furthermore, the owner of RS putted forth that it

is wise for future features on which they are not sure whether they will be appreciated by users, to create a minimum viable feature or mock-up feature for which the users will be asked to give their opinion about by means of the LOOP method.

6 CASE STUDY EVALUATION

The case study is executed to see whether the created artifact in the form of a method would fulfill the drafted requirements/objectives. This chapter will state the attained case study results from conducted observations and interviews about the case study and subsequently evaluates whether the method is feasible, effective and beneficial (Section 2.2).

6.1 OBSERVATIONS AND EVALUATION

This section states our own observations with regard to the case study. We partially assisted the case company with executing the method. We supported the documentation of the phases and analysis of the data collection results.

Relative much time is used for the visualization and documentation of possible sets of features (feature tree). The rationale for the visualization and documentation is to provide the company with a clear oversight on possible feature configurations to validate per customer segment. However, the method is less attractive to execute for features with relative small user impact. The number of suitable features for validation by means of the method is therefore limited. Because of this, the benefits of documentation and visualization of different features is not outweighing the costs of conducting it. However, we do keep this as an optional activity in the method to tackle the possible scenario that the method is conducted on a large scale and there is a risk of losing oversight.

The greatest identified risk that could influence the feasibility of the method was that the end-user was not willing to give his opinion. The benefits of the method in terms of gained knowledge would be weaker if the amount of collected data would be low. Fortunately, a large number of end-users (N = 6082 which is approximately 10% of the whole user base) were willing to share their opinion on features via the mini-surveys built into the product.

The newly developed functionality that lets the product manager create campaigns to gather user data was designed and implemented in a short amount of time. Therefore, the gathered data was not flawless. The data contained duplicate entries, and campaigns were ordered with different identifiers per supermarket which made the aggregation and analysis of data time-consuming. This event led to the realization that there is a risk in the data collection phase. If one does not think the design of the data-collection mechanisms carefully through, there is a potential that the collected data is hard to analyze, or even worse, useless.

When we presented the analyses of the data collection to the owner and product manager of RS we saw that they mainly focused on translating the results into a short-term action plan. This plan concluded the actions taken upon the further lifecycle of both features. Moreover, the product manager also realized that there were incorrect assumptions about the user base. This attained knowledge can be used for better future decision making.

6.2 INTERVIEWS

We have chosen to conduct a semi-structured interview with the owner and product manager from the case study company. The interview is conducted with the both interviewees at the same time with the rationale that they could complement each other and that not only the communication between the interviewer and interviewees could be observed but also the conversation between interviewees about the topic could be stimulated and observed.

Additionally, two owners of another software producing company were interviewed to minimize validity threats. These owners were informed about the method by means of a presentation which used the actual case study as an example. We chose to use the case study as an example to expose both groups to the same potential of the method. To prevent a bias, these interviewees were told that this was an artificial example. Because of this, the interviewees were forced to conclude for themselves whether the method would also be feasible in their own opinion. When the interview was concluded, the interviewees were informed that the example was an actual instantiation of the method.

A semi-structured interview is suited because it is mainly focused on addressing pre-drafted themes and eliciting the interviewee's opinion about these themes while maintaining a certain degree of freedom during the conversation to give room for new findings (Cohen & Crabtree, 2006). A more rigid interview in the form of a survey, or questionnaire, would give the interviewee no room for elaboration; the risk of an open interview is that possibly not all relevant topics will be addressed.

6.2.1 QUESTIONS

The themes are derived from the method's three main objectives. The different questions per objective are stated in the next section. The purpose of these questions is to ignite a discussion with the interviewee about the relevant topic. Some questions are closed, but the rationale for the given answer will be asked as a follow-up question.

FEASIBLE

Do you consider the method's steps realistic for implementation in a real-life context?

Would you consider certain steps too abstract, or too specific, for implementing at your company?

Do you believe that executing the method will cost less resources in following iterations?

How well do you think this method aligns with an agile development environment?

Do you think the method could be synchronized with every sprint?

How fast do you think you can execute one iteration of the method?

Do you have a gross estimation of resources that the execution of the method will cost in terms of man-hours in your company?

Do you consider the execution of the method to be difficult?

Do you consider the method as feasible?

EFFECTIVE

Why do you (not) think that executing the method will result in a better product for the user?

Why do you, or do you not, think the method actually relates user value to features?

Why do you, or do you not, think, that better future decisions will be made by the knowledge gained from executing the method and reflecting upon stated hypotheses?

Do you think that the organization will learn from the gained knowledge?

Do you think that executing the method will potentially result in extra customers and revenue?

Do you see other benefits of executing the method?

Do you believe the method is effective?

BENEFICIAL

Why do you, or do you not, think that the benefits of executing the method would outweigh the cost of executing it? (I.e. is there a positive cost-benefit ratio)

OTHER QUESTIONS

What aspects/phases/ideas about the method would you improve?

Are there other remarks you have concerning the method?

6.3 INTERVIEW EVALUATION

The conducted interviews are summarized in section 7.3.1. In this section, the link between every given opinion and question is emphasized by stating the related question's number between brackets in the text. For every question in Tables 13, 14, 15, the interviewee's position is summarized as positive, negative, not given or neutral. The concluding interpretations per objectives are given in subsection (interpretation).

Feasibility			
Question #	**Question**	**Interview 1**	**Interview 2**
Q1	Do you consider the method's steps realistic for implementation in a real-life context?	Positive, the implementation is proven during the case study.	Positive, with the right context and resources.
Q2	Would you consider certain steps too abstract, or too specific, for implementing at your company?	Not given	Positive, when aligned with current processes.
Q3	Do you believe that executing the method will cost less resources in following iterations?	Positive, for the data collection mechanisms in phase 2. The other activities from the method only over longer period.	Positive, but only on the mid to long term.
Q4	How well do you think this method aligns with an agile development environment?	Output of the outcome knowledge fits with agile. However, too much upfront thought could be non-agile.	Mixed opinion because the outcome knowledge fits agile but the variability within a sprint makes drafting upfront hypotheses hard.
Q5	Do you think the method could be synchronized with every sprint?	Negative. The time for data collection is unpredictable, and the method requires more time than one sprint.	Negative. Data collection is unpredictable. The method should be conducted on an epic level.
Q6	How fast do you think you can execute one iteration of the method?	Preferably more than 5 weeks.	At least one month.
Q7	Do you have a gross estimation of resources that the execution of the method will cost in terms of man-hours in your company?	Positive, the method is not costly. However, it can be scaled up and will cost more.	Negative, the method is costly if it is done correctly.
Q8	Do you consider the execution of the method to be difficult?	It is considered do-able and will get easier over time.	Yes, to correctly execute the method. Especially, for the long term attaining of customer satisfaction.
Q9	Do you consider the method as feasible?	Positive, certainly but preferably over a longer period.	It is feasible in situations with enough resources and a large user base.

Table 13, the feasibility questions linked to the interviewees' opinions.

Effectivity			
Question #	**Question**	**Interview 1**	**Interview 2**
Q10	Why do you (not) think that executing the method will result in a better product for the user?	Positive, because they now pro-actively integrate end-user opinion.	It could be positive, but there is a lot of risk in the processing of the attained knowledge and only if the right features are chosen.
Q11	Why do you, or do you not, think the method actually relates user value to features?	Positive, because they now directly ask the end-user.	Positive, but only if the data is analyzed carefully. Sometimes the user does not even know what he wants.
Q12	Why do you, or do you not, think, that better future decisions will be made by the knowledge gained from executing the method and reflecting upon stated hypotheses?	Positive, they learned that they should reflect upon their assumptions which is done by executing the method.	Positive, documenting the hypotheses and reflecting upon these helps in the understanding of the product.
Q13	Do you think that the organization will learn from the gained knowledge?	Positive, help all employees validate their opinions and creation.	Big organizations are a threat to the feasibility of the method. But the insight gained makes an organization more knowledgeable.
Q14	Do you think that executing the method will potentially result in extra customers and revenue?	Positive. Listening to the end-user results in better word-of-mouth advertising and thus in more customers.	Not extra customers, but possibly a higher retention rate and more sales, up-sales and less down-sales. Also, no costs are wasted on superfluous features.
Q15	Do you see other benefits of executing the method?	User engagement.	Organizing features by the hypothesized value. Structuring the development process.
Q16	Do you believe the method is effective?	Positive, A result of the positive answers to the previous questions.	Positive, if executed with caution and enough resources are available.

Table 14, the effectivity questions linked to the interviewees' opinions.

Beneficial			
Question #	**Question**	**Interview 1**	**Interview 2**
Q17	Why do you, or do you not, think that the benefits of executing the method would outweigh the cost of executing it?	Positive, but you have to find the right balance in terms of what features have potentially enough impact for the user, and what features do not.	Positive, but only on the mid to long term for bigger companies with enough resources. Furthermore, only if the method is applied on the right features.

Table 15, the beneficial question linked to the interviewees' opinions.

6.3.1 INTERVIEW 1

Interview 1 was conducted with the owner and product manager of Retail Solutions and took approximately one hour. The product manager (PM) of Retail Solutions has heavily been involved in the implementation phase of the method and was the dominant speaker during the interview. However, the owner of Retail Solutions held a more outspoken opinion on the company's vision and values and how these relate to the method and the effect on their product and user base. The interviewees' positions concerning questions are highlighted by stating the question number between brackets.

SUMMARY

Do you consider the method's steps realistic for implementation in a real-life context?

PM: I do believe that the method is feasible. If we incorporate this with our current workflow of designing and prioritizing features it is very feasible (Q1, Q9). I also think that if we had more time for executing the method, we could have implemented the method even more thoroughly and test more features and designs (The implementation of the method took approximately five weeks) (Q5, Q6). Before executing the method, I had my doubts if it would be feasible in the given timeline, and we had to speed up some things, but in the end, I concluded that it was very feasible (Q9).

PM: Upfront I had doubts whether the users would react the way that we would like; and how they would receive the questions, and how and if, they would answer the questions. But these concerns were proven wrong and I was happily surprised by the feedback (Q1, Q9). I also think that RS can very well use this method in the future to decide upon which features to implement and how. I am looking for a way to integrate it with the regular design phase and roadmap. I do not want to use the method to compare similar features and choose the best one between them, but I want to use it to validate a mockup, prototype or fully functional feature with a small group of users and then if it is successful expose it to the full user base and test the features again. Furthermore, I also want to use users feedback more upfront in the process, and for example, ask users to give their preference from a list of features and afterwards test by means of the method whether this upfront market research was valid or not.

Do you believe that executing the method will cost less resources in following iterations?

PM**:** I believe the actual method will get easier to execute over time (Q3, Q8). However, it needs more upfront planning than we did, because you already need to start thinking about it a long period before you implement it. For example, asking user groups for opinion, get feedback, talking to stakeholders is a process that needs to be incorporated in a bigger timeline (Q5, Q6). I believe it is better to execute the method with relative little effort over a longer timeline, than a lot of effort in a short timeline (Q5). Of course, practice will help and it will maybe become a second nature if you fully adopt it, but the learning curve in the beginning is high (Q3). However, on the technical side we have a good basis now to expand and repeat this process (Q3). Furthermore, I believe this method mainly helps in validating features in retrospect, and I also want to use more upfront validation of ideas and designs.

PM: We will not execute this method on a large scale. But when we do it, we would like to have richer feedback from the user than just the feedback of a thumbs up or a thumbs down button. This means that we also want to design follow-up questions that ask the user why they are not satisfied if that is the case. I was very surprised about the amount of feedback we received, and this is very valuable but it gives the company no information for the next direction.

Owner: The direction would be to investigate what is the user's need is or how they would want it different.

PM: I agree, and I also want to see how the user would like to have features differently. I want to use the method in the future, but I want to enhance the technique to collect user feedback. But we have to take in mind that the scope of executing a method iteration was short for this iteration (Q6).

How well do you think this method aligns with an agile development environment? And do you think the method could be synchronized with every sprint?

Owner: I think that it has a perfect fit with an agile environment. Because you want to develop the features that have the most value for the use. Once you receive the feedback you see whether a feature gives value or not. And this feedback can perfectly be used for the prioritization of the next sprint (Q4). I do not think that it is desirable to synchronize the method because you do not know how much time there is needed to collect user data (Q5) .

PM: We not only want to use the information on sprint level but also on a long-term level (Q12).

Why do you (not) think that executing the method will result in a better product for the user?

PM: I agree because you directly ask feedback from the actual user, letting that opinion count for your future decisions will result in a better product (Q10). The hardest issue is that we mainly have two stakeholders concerning the product. On the one side we have the customers who pay our bills and with whom we talk and share our ideas with. But we also have the end-users of our product, and we are not always engaged with the latter. With this method we can incorporate their feedback within the actual design process. And that will result in a better product (Q10). Right now, the end-users cannot give much input in the design of the product. Of course, the end-user does not always want the same thing as the actual paying customer wants to pay for, and that is a balance we need to find. We can design a perfect product based on the usability for the end-user but it is not always what the customer wants to pay for. Therefore, I believe this method helps with balancing the needs of these stakeholders.

Owner: I believe it results in a better product. With the feedback for the users, we can built better features within the freedom we get from the paying customer (Q10).

PM: The method helps you in actively improving a feature instead of passively waiting if somebody posts a new bug or responds with negative feedback (Q10).

Why do you, or do you not, think the method actually relates user value to features?

PM: I agree and really do believe you make a link with the user but also especially engage the user. The user feels he has a part in the decision process and that he is heard (Q11).

Owner: What I wanted to mention in the previous question and what is also related to this one, is that whether the product will result into a better version because of the method, will depend on your follow-up on the received user feedback. The method will only be effective if you do something with the feedback. The user will otherwise stop responding because he has the feeling he is not heard. Furthermore, this will result in a decrease of satisfaction. Our core principle is to put the user in the center. Therefore, in the communication with the paying customer, the user is the central point for RS.

PM: The challenge is to communicate with the end user about what is done with their feedback. A possible visual and functional response is designing wat is requested. But if you are somehow not able to do it in this way, how will you communicate that their feedback is heard?

Owner: Maybe we should create some kind of community. Like the App-store where user can see what kind of updates are done. You also see some tools which have an online community where everybody can post questions or wishes. But that is a lot of work to realize.

Pm: We can use the user feedback to motivate our decisions to the customer. We could also use release notes in PMT itself, readable for the end user.

Owner: this will help in the effectiveness and user value and the method will keep its value. Otherwise the user feedback will decrease and the value of the method will decrease.

Why do you, or do you not, think, that better future decisions will be made by the knowledge gained from executing the method and reflecting upon stated hypotheses?

PM: I certainly agree that this will greatly affect future decisions. When we saw that some hypotheses were proved wrong this affected the way we thought about the product because we might need to do a wider market analysis within our user base before implementing a feature. I learned that we should validate the requirements more before we actually make the feature (Q12).

We normally generalize the opinions we get from a small user group and we have learned to not take these opinions from dominant users as a truth for all users. Furthermore, we want a broader stakeholder involvement were the end-user group gets a bigger say. We concluded this based on the calendar integration feature from which we thought that it was an attractive feature that would increase user satisfaction but which was proven wrong. We therefore had wrong assumptions about our users when designing the calendar integration feature. For a future update of the product we will study if the user wants a different way of integrating the work with their calendar, or maybe even do not feel the need to integrate their schedule (Q12).

The method it is basically reflecting upon yourself. You assume something but you do not assume that you are right. Think again and rethink. And that is maybe in contrast to agile software development

because that is learning to developing so you do make mistakes and updates. And in agile you have to be careful to give it to much thought because then nothing happens (Q4). However, this method influences the way you look at future features for me.

Do you think that the organization will learn from the gained knowledge?

Pm: I think this method would benefit especially in validating all the employees in the organization (Q13). It helps to see if we did the right thing, and that is always a nice reward for the invested effort.

Owner: I think executing this method results in very valuable information for the application engineers. Because every application engineer has its own customer/platform to which he customizes the product (Q13).

PM: I think this sort of method could also be executed on an organization level. Not validating an isolated single features but also the full product.

Do you think that executing the method will potentially result in extra customers and revenue?

Owner: I think that would be a logical result. Our growth is a result of satisfying the users. All our marketing is done by word-of-mouth (Q14).

PM: Less negativity from the users will result in positive advertising for the organization. And yes, that would affect the organization (Q14). The strategy of the product is faced at the user. We want to make PMT very practical, usable, not to fancy and especially time-efficient.

Do you see other benefits of executing the method?

PM: We see user engagement as one of the biggest benefits (Q15). It could result in users returning to your application more often because of curiosity for new questions. Maybe this is farfetched, but the method certainly is engaging users, and letting them react to the software product, instead of letting them passively using it. They may start using it more often. Especially if they receive feedback upon their reactions.

Do you think it is a costly method to implement in terms of resources?

PM: No, looking at software development as a whole in terms of costs, versus the costs of implementing this method the costs are very small (Q7). However, it depends on how thorough you want to implement this method. If you want to hire a fulltime data-analyst for analyzing all the data, thinking about designs, and thinking about user engagement in a positive manner, and if you need other resources like a social media expert to guide you in how to ask questions to the users and in what form, then this could become a costly method (Q7). It all needs to be weighed against the benefits. But in essence it is not a costly method. We have executed the method within a couple of weeks and created a base framework for data collection. Especially repeating the technical side of the process would require a little amount of resources (Q3). And the data that you receive, far outweighs the cost of the method (Q17).

Why do you, or do you not, think that the benefits of executing the method would outweigh the cost of executing it?

PM: It is difficult to express the feedback and benefits of the method in value. But I think it is beneficial if you have the right balance. You should not do this for every feature because you get masses of data to analyze. However, I think we will definitely benefit by doing this effectively on certain big and new ideas (Q17).

Owner: I think it is costly on the short term, but on the long term it is different (Q7). However, because we as a company have the principle that we want to put the user at the center, we want to know this feedback, to know whether we do the right things. Secondly, we have a high number of users, and I believe this is the most cost-effective way to get their feedback (Q17). However, if we were a company with only one hundred users, then we would not implement the method like this. We would use the method but collect our data in a different way.

PM: There is more to receive from the method than the feedback, such as the user engagement and the increased quality of the product .

Do you have any other remarks about the method you would improve or any other remarks in general?

PM: In hindsight I would give the design of the data collection a bit more thought. For example, thanking the user after he gives his answer. If I would repeat the method, I would put more emphasis on the first phase to thoroughly think about our hypotheses and our expectations, and incorporate that the data collection phase. But as a result of the limited time we had, this was not possible in the first iteration.

Owner: If you want to execute this method, it has to be in your nature as a company. We really appreciate the user and put the user in the center and are willing to set resources aside.

6.3.2 INTERVIEW 2

The second interview is conducted with a young startup company called Your Next Concepts (YNC), that at the time of the interview, exists for two years. YNC is a young start-up which focuses on the academic market. The vision of the company is to enrich education with data, and with this improve educational institutions. The main product of the company is called Academy Attendance. The primary function of Academy Attendance is to register the attendance of students. YNC aims to get the most value out of student attendance data. This information helps the institute in living up to attendance policies but also gives insight in possible pain points within programmes/courses.

The first iteration of the method (Appendix A) was created by taking the context of YNC as an example situation.

The company is founded by three young post-graduates. The software product of YNC holds similarities with the product of Retail Solutions in the sense that the customer is not the end-user. We believe that this interview is a valuable data source because YNC is a small company with limited resources (only two fulltime software developers), and therefore, the owners will possibly have a different viewpoint on whether the method is feasible, effective and beneficial.

The interview was conducted with all three owners of YNC which are depicted as Owner 1, Owner 2 and Owner 3. The interview took approximately 50 minutes. Before the interview was conducted, a presentation of the final method iteration was given in which the application and results of the case study at RS were presented. However, the owners were told that this case study was an example and not real data. This was done in order to let the interviewees critically think about the feasibility of the method from their point of view.

SUMMARY

Do you consider the method's steps realistic for implementation in a real-life context?

Owner 1: I believe these are realistic but it depends on the organizational context in terms of structure and hierarchy. It is realistic if you have enough time to implement the method (Q1).

Would you consider certain steps too abstract, or too specific, for implementing at your company?

Owner 3: It is doable if you map the method on a company's development and product cycle (Q2).

Owner 2: And you should have the technical infrastructure in place to facilitate the data collection.

Do you believe that executing the method will cost less resources in following iterations?

Owner 1: Yes I believe that is trivial (Q3).

Owner 3: I think one gets better at setting up the hypotheses and data collection mechanisms over time. And for example, the infrastructure that facilitates questionnaires will be reusable (Q3).

Owner 2: The method forces you to think about features in a certain manner. And I think this is something that you have to do multiple times to get better at. Especially in a team (Q3).

Owner 1: But the steps will go faster, because you do not need to build your infrastructure multiple times. And you will often have need the same metrics for measurement (Q3).

How well do you think this method aligns with an agile development environment?

Owner 1: I do not think this method is dependent on an agile way of working or any other software development method (Q4). I think it is applicable to all of them. You always start development, deliver a feature, and then want to test it. Therefore, it is also applicable in a very straightforward conventional method. It may be even easier in a conventional method context than in an agile context. Because in that case, you have the steps and do not change your features during this development phase, while in agile, you change your features slightly, and that could have a huge influence on the effectivity of the method (Q4).

Owner 3: If you compare the method to the waterfall method, it directly maps on the plan, implement, and test phase. An agile development environment is less structured (Q4).

Owner 1: On the other hand, your results are far more applicable on an agile environment than on a waterfall environment. It is counterintuitive that you have a method that maps more easily on a traditional way of developing software, but the output of the method is far more usable in an agile development (Q4).

Owner 2: I think the method causes more overhead in an agile environment (Q4).

Owner 1: A risk is that your hypotheses and metrics could change during the development (Q4).

Do you think the method could be synchronized with every sprint?

Owner 3: I think this method is more centered around features and user stories. Since this method is not relevant to all features, the decision to use this method should be made at the moment when a user story or feature is selected (Q4, Q5).

Owner 1: I think the "epic" in agile is the right level of abstraction to conduct this method. Whereby, an epic exists of multiple requirements which are linked to user stories (Q4, Q5).

Owner 3: What could be an issue if you include this method with an epic or feature and the data collection takes longer than expected (Q5).

Owner 2: I think it depends on whether you want to synchronize it with a sprint. Because where does a normal sprint start? If features are defined before sprints, I agree that this method is more applicable on an epic level. This is because when you make the hypotheses too specific, you have the risk that the feature changes during the sprint and your hypothesis becomes worthless. However, if you use the method on the level of an epic, you as an organization are more flexible in development freedom. Concluding that it is very important for the success of the method that the right level of abstraction for the hypotheses is selected (Q5).

Owner 1: I think this mainly happens when the product owner, or manager, is prioritizing the features during the creation of the backlog. In our case prioritizing features is mainly driven by what the customer wants.

Owner 2: Another additional benefit I see is that the company can categorize user stories based on the hypotheses. I can imagine that multiple stories have the related hypothesis. And sometimes features can be complementary and you choose to validate them both as a set (Q15).

How fast do you think you can execute one iteration of the method?

Owner 1: I think you can implement the method in one week.

Owner 3: But taking the data collection period in consideration, it should take at least one month (Q6).

Owner 1: Yes you are right.

Do you have a gross estimation of resources that the execution of the method will cost in terms of man-hours in your company?

Owner 2: I think it is a costly method because you have to deeply think about the first phase which includes setting up the hypotheses. And even in the ideal situations of having a good infrastructure for in-product testing you need to fine-tune this on every iteration (Q7).

Owner 3: I also think it is time-consuming to align all hypotheses between relevant stakeholders (Q7).

Owner 1: A you need at least one meeting with quite a few people to determine a hypothesis. After this, you need to explain to the development team how to implement the method (Q7).

Owner 3: I think it takes time to apply the GQM method from the second phase in a correct manner. Moreover, making the GQM phase really explicit and actionable is the hard part (Q7, Q8).

Owner 1: We do not say the method is not valuable. You can decide if it is worthwhile doing when you have observed the cost and benefit of the method over a given time (Q17).

Owner 2: I think this method can structure your meetings when you discuss the hypotheses and goals. And when you document that this will prevent future discussions .

Owner 3: Yes this will prevent repeating the same discussions (Q15).

Do you consider the execution of the method to be difficult?

Owner 1: It is difficult to implement the method in a correct manner. It is easy to write down a hypothesis, but it is really difficult to write down a good hypothesis and actionable measurable goals that really help in your understanding of the user (Q8).

Owner 3 It would require some iterations. A possible pitfall is that within one of the first iterations you give up on the method because it does not add enough value (Q8, Q3).

Owner 1: You have the same type of challenges as when you change from waterfall development to agile development.

Owner 2: What also could be a difficulty is how to measure the satisfaction. Can you bother the user for every little change you make? This would negatively impact their satisfaction (Q8).

Owner 3: This issue is also a challenge towards your feasibility because you cannot unlimitedly collect subjective opinions from your users. It is a constraint of the method that user interaction is required. Furthermore, your user group should be big enough. Also in terms of user segments (Q9).

Owner 2: If we want to do this, we have to come up with creative ways of measuring user satisfaction without explicitly asking the user. Maybe a few times to see whether you can relate the hypothesis to functional goals, and from there on automate it.

Owner 3: I am a bit skeptical in how you can do this “unaware” satisfaction measurement in an objective manner. I think that is quite difficult to do (Q8). But maybe that is another topic.

Owner 3: Defining the constructs for satisfaction is very difficult (Q8).

Do you consider the method as feasible?

Owner 3: It is hard to tell. The topics we talked about could be constraints on the speed of iteration (Q9).

Owner 1: It is feasible to do, but if it outweighs the costs is the question (Q9, Q17).

Why do you (not) think that executing the method will result in a better product for the user?

Owner 1: Reasons that let it not result in a better product are that users do not always respond or act in a logical way and you cannot always explain user behavior. In that sense, you can be easily mislead by the results. So there is a risk of getting knowledge from which you do not know whether it is good knowledge. This can be worse than knowing that you do not know a thing (Q10, Q11).

Owner 2: It will be risky when you have to discard a hypothesis because you cannot conclude that the opposite is true (Q10).

Owner 1: I think that another risk is that you want to test features fast in the current agile methods. So you have your results within a few weeks. But if you change big things in your application, people have the tendency to resist the change in the beginning. So therefore, a feature may therefore start with a lower user satisfaction but over time the satisfaction will increase because they get used to the new features (Q10, Q11).

Owner 2: It is very valuable to come up with hypotheses that cover all four phases of the proposed feature value model. This takes the temporal aspect of a features satisfaction into consideration (Q10, Q16).

Owner 1: If we would apply this method it could be valuable if we apply it with the right features. But you should not apply this to all of your features all the time. It should be one of your tools to cover your blind spots (Q10).

Why do you, or do you not, think the method actually relates user value to features?

Owner 3: Given that the method is applied in a correct manner, I believe that the method does help to understand your user better, and that should lead to a better product (Q10, Q11) .

Owner 1: Especially, for the "quick wins" it is valuable to measure via the method, but if you talk about big innovations then sometimes the user value is not always well determined by the user itself. So you have the risk that the method could slow down big innovations. (Q11)

Owner 1: I think the method shows a clear path on how to get the knowledge but you need to upfront determine whether you want the knowledge and spent so much time on it or not (Q17).

Owner 3: And how you should use this knowledge.

Owner 1: Therefore, the success of the method will be determined by whether you choose right the features for the method (Q17).

Why do you, or do you not, think, that better future decisions will be made by the knowledge gained from executing the method and reflecting upon stated hypotheses?

Owner 2: Yes, it is nice to document your hypotheses so you have a better understanding of why you built things instead of documenting only what you are making. Additionally, you can look back at these hypotheses on a later moment in time. (Q12)

Do you think that the organization will learn from the gained knowledge?

Owner 1: In the end you want to know your customer or user. If you know your user in the best possible way, you can create a product that suits his or her requirements best. Indeed if you repeat the method indefinite, you know your customer and user very well, and you can built features that really help them in achieving their goals. In that sense the method will help (Q10).

Owner 2: The complexity increases with every person involved. Therefore, it is very hard in a large organization to make change possible. Concluding, that I believe that in larger organizations the learning is not possible (Q13).

Owner 3: However, getting more insight on your customer in general should make the organization more knowledgeable (Q13).

Owner 2: That is true.

Do you think that executing the method will potentially result in extra customers and revenue?

Owner 1: No, but I think that the number of users that will stop using your product over time should decrease slower (Q14).

Owner 2: Given that this method will help you better understand your customer, you can target them more effective via marketing.

Owner 1: It depends on whether your customer is your end-user.

Owner 3: Still this information is a good input for your marketing. But the value is attained very indirect and on a long term (Q14).

Do you see other benefits of executing the method?

Owner 1: It ensures that the product owner makes better decisions during the prioritization of the feature backlog. This benefit is a result from executing the first phase of the method (Q16).

Owner 3: It creates awareness on customer value (Q16).

Owner 1: Yes I think that is the biggest gain.

Owner 1: Creating the feature tree is also a beneficial. I believe a lot of companies have features with intermediate interactions but do not know how map these. I think we as a company also do not do this sufficiently (Q16).

Owner 2: The method helps structuring the development process (Q15).

Do you believe the method is effective?

Owner 3: Yes

Owner 1: the answer is yes, if we take everything into account that we discussed during this sessions (Q16).

Owner 3: It depends on how much time and effort you decide to put in the method and if you implement it correctly (Q16).

Why do you, or do you not, think that the benefits of executing the method would outweigh the cost of executing it?

Owner 1: Yes it is possible, but only on the mid to long term when you as a company have practiced the method and set up a good infrastructure. There is a learning curve (Q17).

Owner 3: This is very situational because for example, I have doubts that for our organization the method would be beneficial since I believe that we have a good understanding of our customer and I doubt that you require the method to do that effectively in our context. However, if an organization is larger, it would be more natural to conduct this method, since employees are not able to share all the knowledge on the customer as we do. In that case you want to have a more structured way to get these results (Q17).

Owner 1: In my opinion it is in the DNA of today's startups to apply the lean principles of building, measuring, and learning, on a high level. However, bigger and older companies do not do this by nature whereas it is more easy for them to cherry pick the features because they have a larger user base (Q9).

Owner 1: If your turn rate goes down you earn more money (Q14).

Owner 3: Furthermore, depending on the product, you can make more new sales, up-sales or less down-sales (Q14).

Owner 1: A company can also reduce costs by not building superfluous features (Q14).

Owner 2: This method will not be the right one for startups because it gives to much overhead. Startups have other techniques in place to validate their product. Concluding that the method is more suitable for larger companies (Q9, Q17).

6.3.3 INTERPRETATION

In this section, the results from the interviews are analyzed and interpreted. These findings are combined with our own observations, and are used to reflect upon the achievement of the method's objectives. Firstly, the sub-objectives are analyzed with the attained knowledge. Secondly, we conclude whether the three main objectives are met.

OBJECTIVE: FEASIBILITY

The method should be feasible for software producing companies.

- *It should be possible to implement the method in a real-life context.*

  Both interviewed companies were unanimous convinced that it is possible to implement the method in a real-life context. Additionally, the case study has proven that the method can be implemented, whereby we want to underline that some activities were assisted by the researchers of this study. However, the fact that the implementation is possible in a real-life context does not mean that it is implementable in every context.

  Identified situational requirements for application of the method are that the company should have enough recourses for implementation and a considerable large user base. We conclude the former because YNC believes the method is very costly and that it does not have the manpower and time to implement the method. RS per contra, thinks the basic method is not costly and is scalable in terms of resource consumption. Additionally, RS had sufficient resources for implementing the method.

  The size of the user base is important for multiple reasons. Firstly, if the user base is small the method could still be suitable, but the data collection mechanisms are possibly less structured and more informal. For example, if a company only has one hundred end users, validation of hypotheses can be done by periodical customer interviews and no technical data collection mechanisms need to be implemented. Secondly, a large user base will logically result in more responses and therefore makes analysis more powerful. Thirdly, the variety of user segments also influences the size of the potential set of users for testing. For example, the user base can have a considerable size, but the number of admins can be a handful. In that case, other means

can be more suitable to obtain their feedback. Finally, having a large user base makes it easier to keep the method's execution as non-intrusive as possible. If the user base is considerably large, one can cycle validation experiments on subsets of the user base and the user is less often exposed to validation experiments. Although, this rationale is based on the assumption that users experience giving their opinions as negative. RS believes that if there is a right follow-up protocol in place that pays tribute to the end-user, and informs them what is done with their opinion, validation experiments could also be a pleasant non-intrusive experience for the end-user.

- *There should be a well-balanced level of abstraction in the method's process description. Method steps should be specific enough to be actionable, but not too specific such that the method stays situational.*

  No difficulties in implementing the method's steps occurred at the case company. The activities up to the stating of the data collection mechanisms have proven to be straightforward. It is a conscious decision to give the executer of the method a lot of liberty in how to collect the data. Because of this, the method is applicable in wide diversity of company context. On the contrary, the step to come up with a data collection mechanism needs a certain amount of creativity. YNC foresees that in this phase possible difficulties could occur. Both companies agree that the steps are executable but should be aligned with the current development process of the company, and also this is very situation, and therefore requires a certain amount of creativity to realize.

- *It should be possible to align the method with an agile development environment.*

  Both companies agree that the knowledge gained from executing the method is very valuable and perfectly suits the agile way of working. We have also tested whether the method could be conducted in alignment with sprints. This is harder to realize because the data collection period is unpredictable.

  An indicated risk from RS is that too much upfront thought could be non-agile and therefore be contra productive.

  YNC believes that the method can align on an epic level with agile development but indicated a possible risk in this because features can change during a sprint and therefore lose their connection with hypotheses. We argue that this is dependent on the level of abstraction one uses for the goals and hypotheses. For example, the activity dashboard feature has as one goal to make the user more efficient by showing the tasks at hand. The hypotheses connected to this goal is that the user will appreciate this timesaving. The design of the activity dashboard still has a large degree of freedom within this formulation. Therefore, whether the activity dashboard is red or blue or big or small does not matter for the effectivity of the method in this case. On the other side, when one decides to change a feature significantly enough so that the relation to the goal and hypotheses change, we believe that it is beneficial to write down the new hypotheses for this change.

- *The execution of the method should utilize R&D as an experiment system.*

  Before, during, and after the development and deployment of a feature, research is done. Concluding, the whole method is designed as an R&D experiment system. Therefore, this is not explicitly tested during the interviews. A noticeable difference between any other R&D

methods, is the speed of the iteration. A R&D method that does not ask for the user's feedback can be applied several times a day. In the LOOP method, the user is actively engaged via the software product to give his opinion and this takes time. Furthermore, we argue that the high-level goal for any feature is to ultimately deliver value to the customer. After the implementation of a feature, it takes time to effectuate this high-level goal. For these reasons, the method is not executable on a daily basis. We want to emphasize that the LOOP method is complementary to other R&D methods, such as the RIGHT and HYPEX models, and not a replacement. To illustrate, once a hypothesis about functional characteristics of a feature is proven, other R&D methods can do tests on this feature's characteristics on a daily basis. For example, once it is proven that speed has a positive impact on user satisfaction, different fast iterations on the features speed can be tested by use of another R&D method, assuming that this will also increase the satisfaction of a user.

- *Concluding on feasibility.*

  Taking all aforementioned evaluations in account, the method has proven to be feasible. Although the method can be perceived as (too) costly for a relative small startup, it was not concluded as costly during the case study and took relative little resources compared to the whole software development process. Also, the steps were clearly executable during the case study. Both companies thought the method was feasible to execute, although situational factors could influence whether the method's benefits would outweigh the cost of executing. It is concluded that the follow-up actions on the method can determine whether the method would stay feasible over a longer time due to customer involvement. This is a possible topic for future research.

EFFECTIVE

The method should have the following effects once it is implemented in a software company:

- *Executing the method results in well-advised, data-driven, and short-term decision making concerning a product's feature portfolio.*

  Both companies have no doubts that the knowledge gained from the method will result in better decisions concerning the product's features. YNC foresees risks that the attained data can be wrongly interpreted and therefore incorrect follow-up actions are undertaken. This is indeed a possible risk but outside the scope of this method. The method is a guidance to attain data about a company's product, customer, and end-user, but does not give much guidance on how to interpret the data and translate it to knowledge and on how to undertake correct follow-up actions. Moreover, it is necessary to properly design mechanisms for retrieving data because otherwise data can be falsely interpreted.

- *Executing the method actuates organizational learning concerning a product's feature portfolio and customer appreciation. This results in better understanding of the product, customer and user, leading to enhanced long-term decisions.*

  The execution of the method at the case company has confronted the product manager and owner of RS with the incorrectness of assumptions they held about their end-user. Therefore, they gained new insights about their end-user and product. Moreover, they also learned to reflect upon their assumptions. To foster this learning on one owns rationale is exactly what the method intended to do, and what results in better future decision making. Both companies

were positive that explicitly upfront stating of hypotheses will aid in better understanding the product and therefore making better future decisions. This knowledge will benefit the whole organization assuming that it will be dispersed correctly.

- *The method relates features with actual customer value. Whereby customer value is defined as "direct value perceived by the user of the product".*

  All interviewees believe that this is the case because the company now directly asks the end-user's opinion. YNC indicated the risk that sometimes the user does not even know what he wants himself and that sometimes users have to get used to new innovations. The former can partly be prevented by giving the user the option to give a neutral answer. It is imaginable that the user does not know what he wants, but it is less likely that the user does not have an opinion about a feature that is presented to him. That a user's perceived satisfaction is not set in stone is indeed probable. The feature value model gives aid in reasoning about the temporal characteristics of a feature's appreciation by the end-user. The risks indicated by YNC are risks that come with interpreting the data gained from the method. The goal of the method is to facilitate the obtainment of valuable data. However, how a company reflects upon data and interpret it is situational and not prescribed.

- *Concluding on effectiveness.*

  By actually relating features to user value and reflecting upon the feature's hypotheses, executing the method indeed fulfills its intended short- and long-term benefits. Both companies believe that executing the method in the correct manner and using the gained knowledge will result in positive benefits for the company such as: new customers because a better word-of-mouth advertisement, more (up-)sales and a higher customer retention rate as a result of a better understanding the customer's and user's wishes, and lastly, no resources will be wasted on superfluous features. Additional benefits which are identified is the possible engagement of users, structuring of the development process and organization of features by hypotheses.

BENEFICIAL

The method can be feasible and effective, but the cost of execution should not outweigh the benefits.

- *Concluding on whether the method is beneficial or not.*

  The most important requirement that should be met to make the method successful is that it is applied on the right features. The method consumes resources and can be costly if applied thoroughly. All interviewees agree that if a company uses the method for features that could have a significant impact on the user's perception the benefits will outweigh the costs. The perceived cost of executing the method is relative to the capacity of the company in terms of resources. If the company is too small, the costs of the method will not outweigh the benefits. On the contrary, when the company is big enough and the user base is large enough, the method is perceived as a very cost-effective way of gathering valuable knowledge. Furthermore, it also depends on the values of the company. In case of RS, the opinion of the user is considered as very valuable. If a company does not see any perceived benefit in knowing the user, the method should not be executed. Concluding, if a company can find the right balance in selecting significant features, the method is beneficial.

7 CONCLUSION

This thesis addressed the limitations of the current methods that use R&D as an experiment system, concerning the alignment with customer and user wishes (Section 2.1). We have sought for a solution that complements these methods and better aligns a software product with the desires of the end-user and customer by using R&D as an experiment system. From these shortcomings, the main research question arose: *"What is an actionable method to maximize the customer and user value delivered by the chosen software features, by using R&D as an experiment system?"*.

We utilized design science as a research approach (Section 2.4), and with this, created the LOOP method. The intended effects as a result of implementing the LOOP method were stated. These effects entailed that: a true link between customer/user value and features should be captured, the knowledge gained from the method should help in creating a better software product on the short term and a better understanding of the customer and related decision making on the long term (Section 2.2). We argue that if these effects could be realized, the "open loop" (problem) will be closed.

Through several iterations, the method was created and enhanced, resulting in the final version elaborated in Chapter 4. We conducted a case study (Chapter 5) to conclude whether this final version of the LOOP method would realize its intended effects in a real-life context (Chapter 6). The findings from this chapter are used in the following section (7.1) to answer this thesis' sub- and main-question(s).

7.1 DISCUSSION ON RESEARCH QUESTIONS

After discussing the research sub-questions one by one, the obtained conclusions are aggregated to conclude on the main research question of this thesis.

- *What is a suitable technique to capture and model the link between perceived customer/user value and a feature?*

  In our method, we argue that by writing down a feature's hypotheses and goals, the intended value of a feature is documented and modeled (LOOP phase 1, Section 4.1). Subsequently, by using the adjusted GQM technique, a company can find and implement suitable data collection mechanisms (LOOP phase 2, Section 4.2). These data collection mechanisms can be used to capture and analyze the actual relationship between customer/user value and features (LOOP phase 3, Section 4.3).

  These techniques were implemented during the case study that checked applicability in a real-life context. The method has proven to be feasible (Chapter 5). Hereafter, we discussed the topic during the semi-structured interviews (Chapter 6). Since the concept "customer/user value" can hold different interpretations, we have defined it as: *"direct value perceived by the user of the product"*. To validate whether they perceive that execution of the method will relate user/customer value to features, we asked the interviewees *"Why do you, or do you not, think the method actually relates user/customer value to features?"* (Section 6.2). All interviewees agreed that this was the case (Section 6.3). Therefore, we conclude that the technique is valid.

- *How can the right direction of feature implementation be determined, based on testing different variations of feature configurations?*

  We integrated the creation of a feature tree into the first phase of the LOOP method. By means of the feature tree modelling technique, different variations of feature configurations, and their relationship to user segments, can be depicted. A variation of feature configurations can be seen as the implementation of different equipollent features (Section 4.1.1), or the difference between implementations with or without a feature. After the modelling of possible different feature configurations, the different configurations are implemented (LOOP phase 2), tested, and analyzed (LOOP phase 3). We theorized that by analyzing the features in this way, a software company can make better decisions on feature implementations.

  Because of the limited scope of the case study, the method was only applied to examine the difference in user perception between pre- and post-exposures of features (i.e., no equipollent features have been examined). Therefore, we can only verify that it is feasible to measure whether a feature has impact or not by analyzing its pre- and post-exposure behavior.

  Thereafter, we validated by means of the semi-structured interview whether experts from industry think these activities and data can help in determining the right set of features for the product (Section 6.2.1 Question 10). We found that all interviewees have no doubt that the knowledge gained from the method will aid in making better decisions concerning the product's features. The main remark is that it is still depended on the interpreter of the data whether a better decision is actually made. We therefore conclude that the method indeed helps in attaining the knowledge to make better decisions concerning software product's set of features. However, if this knowledge is wisely utilized to actually make better decisions is depended on the person executing the method.

- *How can organizational learning be actuated concerning product, feature, customer and user knowledge?*

  The first phase of the LOOP method documents the rationale that explains why a feature delivers value to a customer/user, and how this relates the functional goal of a feature. After implementation (LOOP phase 3), this rationale is validated and reflected upon. Because one reflects upon former rationale by taking the actual results in account, we theorized that learning is actuated. If a hypothesis is shown to be valid, the correct "way of thinking" is solidified. If a hypothesis is shown to be incorrect, wrong assumptions are emphasized, and if documented well, will not be made anymore in the future. Additionally, a better hypothesis considering the users wishes can be sought.

  The case study verified and validated that the method realizes this objective. Invalid hypotheses were found during the case study (Section 5.4.3) and this helped the software company to reflect upon their way of thinking about the software product in relation to the customer and end-user. This effect of the method was further validated during the semi-structured interviews whereby both interviewed companies confirmed that upfront stating of the hypothesis behind a feature and subsequently reflection upon this hypothesis will actuate organizational learning. We therefore conclude that the method indeed actuates organizational learning concerning product, feature, customer and user knowledge. Whether this attained

knowledge and learnings will be long lasting depends on the follow-up actions from the company and is outside the scope of this thesis.

- *What is an actionable method to maximize the customer and user value delivered by the chosen software features, by using R&D as an experiment system?*

  We proposed the answer to this question as the final method iteration. The method has proven to be actionable and effective during the case study.

  We conclude that the method is actionable because it is explicitly tested on whether it is beneficial and feasible. In Section 7.3.3 we concluded that the method at least adheres to these two requirements in the context of the case company. Furthermore, we have shown that the method is probably also feasible and beneficial in other companies, provided that these companies have a large enough user base and sufficient resources.

By aggregating the positive answers on the former sub-questions, we conclude that, under the circumstances of the case study, executing the method helps in maximizing the customer's and user's value delivered by chosen software features. We argue that this value, in all probability, will be maximized because the company is provided with the information about what the user values, and therefore, the most suitable features can be included within the software product on the short term. Moreover, the company is provided with information to learn to better reason about the customer's and/or user's wishes, and can therefore also maximize the software product's value in the long-term. However, the feasibility, effectiveness and beneficial aspect of the method has only been concluded for this case study. We have identified situational factors that can act as constraints on the implementation of the LOOP method in other contexts, such as available resources and a large enough user base. Furthermore, since the theorized benefits on the long term can only be actually validated on the long term, further research is required to confirm the concluded believes on this aspect.

Although not explicitly mentioned in the initial research questions, we did not only study how the LOOP method would create a better software product for the end-user, but also explored other possible benefits that could occur for the software company while executing the method. These possible benefits are: an increase in customer retention, more new customers, more (up-)sales, and lesser waste of resources on superfluous feature.

7.2 THREATS TO VALIDITY AND LIMITATIONS

7.2.1 INTERNAL VALIDITY

The scientific study is considered internally valid when the causal conclusion between two variables is most likely to be true (Bhattacherjee, 2012). In other words, the scientist should minimize the risk that an unknown third variable could influence the causality. In the context of this thesis project, we attempted to minimize the risk that the perceived effects of the method are not caused by the method.

We argue that the risk of internal validity for certain observed method effects is low because these effects could be directly related to the execution of the method. For example, the method prescribed to draft hypotheses and reflect upon these. As a result, we saw that the product manager of RS was confronted with the actual and hypothesized value of a feature.

For other evaluated effects this risk was higher; during the questioning of the semi-structured interview, we asked about the potential results of the method over the long term. There is a significant risk of validity in these conclusions because the foreseen effects, such as a better understanding of the product, are subjective and intangible by nature and only occur over a longer period of time. The best way to mitigate this risk, is to observe the effects of the method in a longitudinal study, which was not possible due to the scope of this project.

Furthermore, we concluded that executing the method provides information from which an organization can learn and possibly make better future product decisions. However, whether the information gained from the method will actually lead to a better product is depended on the skill of the interpreter and is outside the scope of this study.

We also identified the risk that only questioning the case company about the final method iteration could lead to a bias. This risk is one of the reasons why an extra interview was conducted with a software company that was not involved with the final iterations of the method (Section 6.3.2).

The last risk considering internal validity is that we partly assisted the case company with the implementation of the method. We documented all the phases and analyzed the data collected from the data collection mechanisms. However, we let the product manager and owner or RS decide themselves how to interpret the analyzed information about the features, and validate the upfront stated hypotheses. This was done to mitigate the risk that our support had too much impact on the effects of the method.

7.2.2 EXTERNAL VALIDITY

The external validity refers to the extent that the concluded effects of the method are generalizable to other contexts than the case study company. Because of the limited scope and time constraints of this research project we could only conduct one case study. We could conclude that the method can have the intended effects because these occurred during the case study. However, because the method was only applied in context, the concluded effects of the method cannot be generalized to different contexts.

However, in order to get as much clues on the feasibility of the method in other contexts as possible, we took the following measures: Firstly, the feasibility of the method was made an explicit objective of the method. We asked all interviewees to not only think about the feasibility of the case study or example but also to think about applying the method in other contexts. Secondly, we also interviewed a young startup company (YNC), with different characteristics than the case company such as limited resources, to see whether they could implement the method. They concluded that in their context, the method was not feasible, but in other contexts could possibly be. Thirdly, during the interview with YNC, we did not tell that the example of the method was actually executed to also see their perceived feasibility of the method in this imaginary context. Lastly, we thoroughly thought about the level of abstraction of the method's steps. We theorized that by keeping certain steps more abstract, such as stating the data collection mechanism, the method will not become to specific and can be aligned to different contexts. A possible disadvantage of this approach is that companies have to be creative in aligning the method with their development process. However, in Section 6.3.3, we concluded that the method is perceived to be feasible and aligned with software development.

7.2.3 RELIABILITY

We documented our research approach with additional research techniques to ensure that later researchers can follow the same procedures (Section 2.4 & 2.5). The creation of the method is hard to replicate because it was an creative process. However, we documented the primal method solution (Section 3.3), an early iteration of the solution method (Appendix A) and the final LOOP method (Chapter 4), our case study (Chapter 5), and case study interview Chapter 6, as thoroughly as possible, so the study can be repeated at different companies in future studies.

7.3 FUTURE WORK

Due to temporal constraints on this research project, we were limited to apply the devised LOOP method to one case study, at one company and on two features. We concluded that the application of the LOOP method could potentially be beneficial for other companies as well. This gives opportunities for future research in the following directions:

Firstly, the method has been applied on only one company. It will be interesting to discover the situational factors where the method is feasible, effective and beneficial. It appeared during the case study interviews that the method is not applicable in every situational context. Different means to collect customer satisfaction can be further studied in future case studies at other companies. Factors such as available resources, user base size, customer as end user, can be analyzed in relation to the LOOP method. Furthermore, effectiveness of the method on testing equipollent features can be tested during future studies.

Secondly, we indicated possible risks and benefits when the LOOP method would be applied for multiple iterations during case study evaluation. A longitudinal study whereby the LOOP method is iterated several times could be conducted to get insight in: the learning curve of executing the method; the cost of executing the method over time; the reaction of the user base when frequently asked for their satisfaction; the overall product performance in terms of (up-)sales, customer retention rate, customer satisfaction; the effectiveness of the method in learning from hypotheses reflection.

Thirdly, the method could be supported by a tool that helps in documenting and analyzing feature goals and hypotheses. Also, the data collection mechanisms implementation can be semi-automated by supportive tools as such as the campaign building feature in the case study.

8 BIBLIOGRAPHY

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). Experience factory. *Encyclopedia of software engineering*.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61-72.

Boehm, B., & Turner, R. (2003). *Balancing Agility and Discipline: A Guide for the Perplexed, Portable Documents*. Addison-Wesley Professional.

Clarke, P., & O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, *54*(5), 433-447.

Cohen, D., & Crabtree, B. (2006). Qualitative research guidelines project.

Collins, E. F. (2012, June). Software test automation practices in agile development environment: An industry experience report. In *Proceedings of the 7th International Workshop on Automation of Software Test* (pp. 57-63). Zurich, Switzerland: IEEE Press.

Czarnecki, K., Eisenecker, U. W., Goos, G., Hartmanis, J., & van Leeuwen, J. (2000). Generative programming. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, *15*

Davis, F. D., & Venkatesh, V. (2004). Toward preprototype user acceptance testing of new information systems: implications for software project management. *IEEE Transactions on Engineering management*, *51*(1), 31-46.

Davis, F. D. (1993). User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. *International Journal of Man-Machine Studies*, *38*(3), 475-487.

Ekström, D., & Þorvaldsson, I. (2016). *Predicting and Analyzing Feature Value when R&D is an Experiment System* (Unpublished master thesis). Chalmers, Gothenburg.

Fabijan, A., Olsson, H. H., & Bosch, J. (2015, December). Early value argumentation and prediction: an iterative approach to quantifying feature value. In *International Conference on Product-Focused Software Process Improvement* (pp. 16-23). Springer International Publishing.

Fagerholm, F., Guinea, A. S., Mäenpää, H., & Münch, J. (2017). The RIGHT model for continuous experimentation. *Journal of Systems and Software*, *123*, 292-305.

Fitzgerald, B., & Stol, K. J. (2014, June). Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering* (pp. 1-9). ACM.

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development*, *9*(8), 28-35.

Griss, M. L. (1997, June). Software reuse architecture, process, and organization for business success. In *Computer Systems and Software Engineering, 1997., Proceedings of the Eighth Israeli Conference on* (pp. 86-89). IEEE.

Holbrook, M. B. (1999). *Consumer value: a framework for analysis and research*. Psychology Press.

Hsia, P., Kung, D., & Sell, C. (1997). Software requirements and acceptance testing. *Annals of software Engineering*, *3*(1), 291-317.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education.

i* Wiki: i* Guides < http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=200 >. Last Access in May of 2017

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). *Feature-oriented domain analysis (FODA) feasibility study* (No. CMU/SEI-90-TR-21). Carnegie-Mellon University Pittsburgh Pa Software Engineering Inst.

Kang, K. C., Kim, S., Lee, J., Kim, K., Shin, E., & Huh, M. (1998). FORM: A feature-; oriented reuse method with domain-; specific reference architectures. *Annals of Software Engineering*, *5*(1), 143.

Kano, N., Seraku, N., Takahashi, F. and Tsjui, S. (1984). Attractive quality and must-be quality. Hinshitsu 14(2), 147–156.

Koç, H., Timm, F., España, S., González, T., & Sandkuhl, K. (2016, June). A method for Context Modelling in Capability Management. In *ECIS* (p.43).

Larson, G. B. (1995). The user acceptance testing process: A case study. *Journal of Systems Management*, *46*(5), 56.

Maslow A (1954) Motivation and personality. Harper, New York.

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and Organization*, *17*(1), 2–26.

Olsson, H. H., & Bosch, J. (2014a). Climbing the "Stairway to Heaven": evolving from agile development to continuous deployment of software. In *Continuous software engineering* (pp. 15-27). Springer International Publishing.

Olsson, H. H., & Bosch, J. (2014b). From opinions to data-driven software R&D: a multi-case study on how to close the 'open loop' problem. In *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on* (pp. 9-16). IEEE.

Olsson, H. H., & Bosch, J. (2015, June). Towards continuous customer validation: a conceptual model for combining qualitative customer feedback with quantitative customer observation. In *International Conference of Software Business* (pp. 154-166). Springer International Publishing.

Otaduy, I., & Diaz, O. (2017). User Acceptance Testing for Agile-developed Web-based applications: empowering customers through wikis and mind maps. *Journal of Systems and Software*.

Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Business.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, *14*(2), 131.

Salem Khalifa, A. (2004). Customer value: a review of recent literature and an integrative configuration. *Management decision*, *42*(5), 645-666.

Schwartz, S. H. (1992). Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries. *Advances in experimental social psychology*, *25*, 1-65.

Silva C, Borba C and Castro J (2011). A goal oriented approach to identify and configure feature models for software product lines. In: Proceedings of the WER'11, Rio de Janeiro, Brazil

Slater, S. F. (1997). Developing a customer value-based theory of the firm. *Journal of the Academy of marketing Science*, *25*(2), 162-167.

Smith, J. B., & Colgate, M. (2007). Customer value creation: a practical framework. *Journal of marketing Theory and Practice*, *15*(1), 7-23

Sochos, P., Philippow, I., & Riebisch, M. (2004, September). Feature-oriented development of software product lines: mapping feature models to the architecture. In *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World* (pp. 138-152). Springer Berlin Heidelberg.

Sochos, P., Riebisch, M., & Philippow, I. (2006, March). The feature-architecture mapping (farm) method for feature-oriented development of software product lines. In *Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on* (pp. 9-pp). IEEE.

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: a framework for evaluation in design science research. *European Journal of Information Systems*, *25*(1), 77-89.

Venkatesh, V., Morris, M. G., Davis, G. B., Davis, F. D., DeLone, W. H., McLean, E. R., ... & Marcolin, B. L. (2003). User acceptance of information technology: Toward a unified view. *MIS QUART*, *27*(3), 425-478.

von Wangenheim, C. G., Punter, T., & Anacleto, A. (2003, January). Software measurement for small and medium enterprises. In *Proceeding 7th International Conference on empirical Assessment in Software Engineering (EASE)*.

Weerd, I., & Brinkkemper, S. (2008). Meta-modeling for situational analysis and design methods. In M. R. Syed & S. N. Syed (Eds.), *Handbook of research on modern systems analysis anddesign technologies and applications* (pp. 35-54). Hershey, PA: Idea Group Publishing.

Wieringa, R. J. (2014). What Is Design Science?. In *Design Science Methodology for Information Systems and Software Engineering* (pp. 3-11). Springer Berlin Heidelberg.

Woodruff, R. B. (1997). Customer value: the next source for competitive advantage. *Journal of the academy of marketing science*, *25*(2), 139-153.

Zdravkovic, J., Svee, E. O., & Giannoulis, C. (2015). Capturing consumer preferences as requirements for software product lines. *Requirements Engineering*, *20*(1), 71-90.

APPENDIX A: FIRST METHOD ITERATION

This chapter states the first idea of a method that guarantees customer value alignment with the software product by means of an example which is coherent to Your Next Concepts' software product named: Academy Attendance.

The classification of the minimal viable feature (MVF) helps stakeholders in deciding whether the MVF needs to be further developed or "sunsetted". The specific class also gives a priority in doing so. Note that for a new MVF only two data points are known. Every time a feature undergoes a method iteration, a new data point can be added with regards to customer value and implementation effort. When a feature has more data points, the feature can be classified with more certainty.

BACKGROUND INFORMATION

In the following sections, all steps from the early solution method (Figure 5) will be elaborated by means of an example which is coherent to Your Next Concepts' product Academy Attendance (see Section 6.3.2).

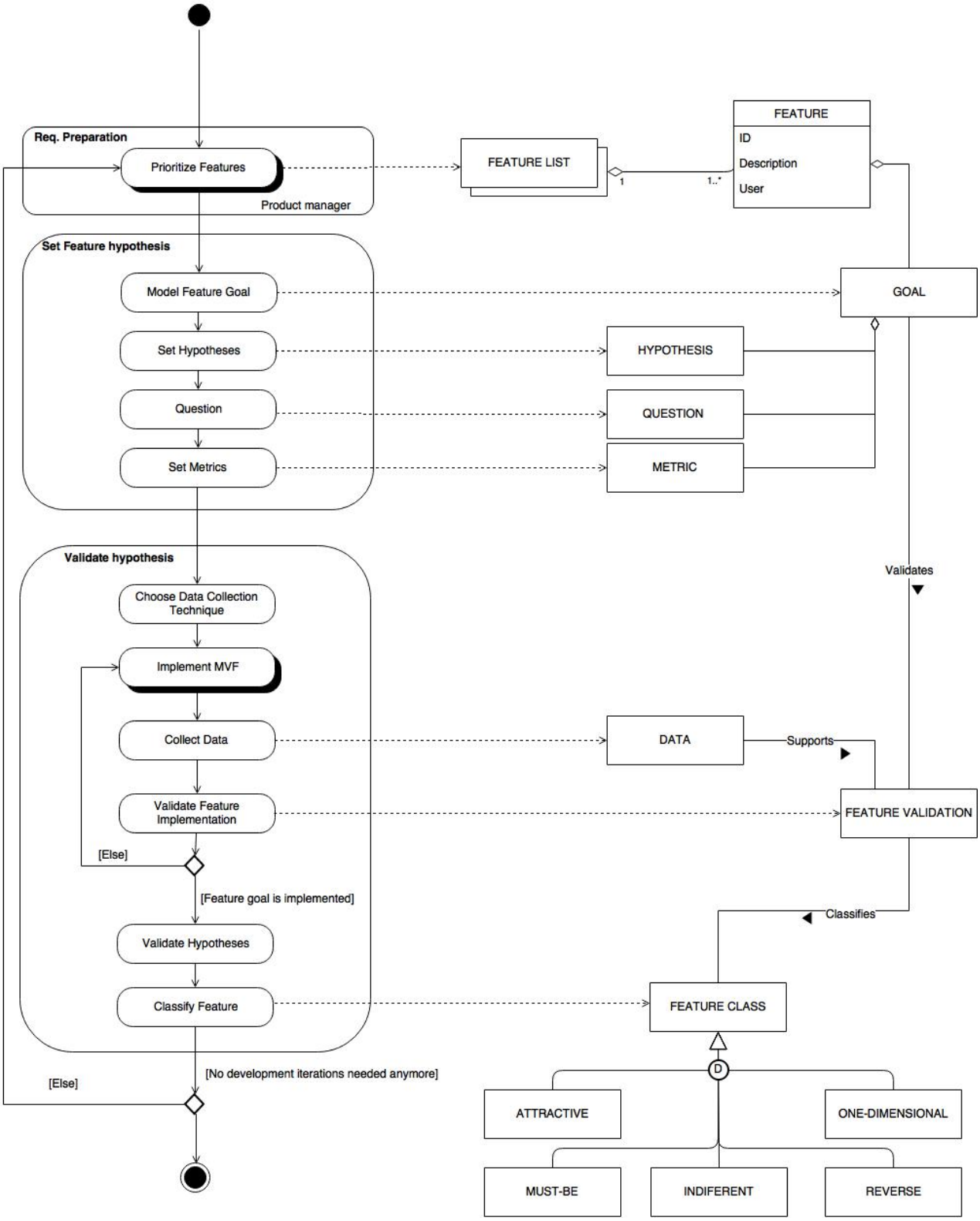


Figure 32. First solution method PDD

PRIORITIZE FEATURES

Because validating features by means of the solutions method is a resource-intensive task, only a small set of features can be validated per cycle. Therefore, features need to be prioritized. The features which are suited for the validation are the features on which the company has doubts if it will have the desired effect for customers or user in terms of value.

In case of Your Next Concepts only a subset of the end-users are customers. Academy Attendance holds different stakeholders which are: students, lecturers, system administrators and management of an educational institution. All of these are end-users of the system. However, the management only uses the system infrequently for reporting. The student, lecturer and system administrators are end-users which use Academy Attendance on a daily basis. Only the management is the customer of Your Next Concepts but it is also the smallest group of end-users. Furthermore, if the other end-users do not use Academy Attendance anymore, the product does not deliver value any more to the management, since it has no useful information to represent. Therefore, it is important for Your Next Concepts to make Academy Attendance as attractive as possible for all its end-users.

One customer has requested Your Next Concepts to invest in the reliability of the attendance data. Students can register their attendance for a class via different ways. One possibility is that the lecturer releases a specific code during class that the students use to register their attendance. The benefit of this approach is that the lecturer only has to release the code and not register every student at the beginning of the class. The drawbacks are that this approach is vulnerable for fraud which is a threat to the reliability of the attendance data.

Your Next Concepts wants to know whether this feature is interesting for all customers and is worth the investment of developing. This feature is therefore prioritized for validating using the solution method. The prioritization of features is very context dependent and therefore will differ in any situation. It is often done by the product manager.

MODEL FEATURE GOAL

The goal of the feature is the starting point for the validation process. If the feature goal is not clear it cannot be tested. The feature goal should be a high-level statement which is related to the business strategy. This goal should be decomposed into smaller goals/questions during later phases of the method. A set of hypotheses need to be derived from the goal.

The feature goal in case of the Your Next Concepts is: *"Increase the reliability of attendance information by 20%"*.

SET HYPOTHESES

Every feature goal has an underlying hypothesis which is direct or indirectly related to customer value. One customer of Your Next Concepts has requested the improvement of reliability. For this customer, we know that investment in reliability will deliver customer value. However, it is necessary to state hypotheses about the feature goal which address other customers and other users of the product. In certain cases, different hypotheses can be derived from one feature goal.

Hypotheses which can be derived from the earlier mentioned feature goal are:

1. Investing in a 20 % increase in reliability of attendance information adds value to all customers.
2. Investing in a 20 % increase in reliability of attendance adds value to end-users of the type lecturer.
3. Investing in a 20 % increase in reliability of attendance adds value to end-users of the type student who is motivated to attend lectures.

Hypothesis 1 states the external validity of the feature. I.e. does the feature value only account for this customer or for a greater set of customers.

Hypothesis 2 states whether the feature will add value to users of the type lecturer. If lecturers see no benefit of a feature that helps in the verification of student attendance, it will be likely that the feature is rarely used by this group of end-users. In the case of Academy Attendance, this feature may be demanded by the customer (management of the educational institute) but if the lecturer does not use it, the feature will deliver no value to the customer.

Hypothesis 3 seems different than the other hypotheses. How will the system bring value to the student if it is aimed at controlling this set of users? This hypothesis can best be explained by means of a metaphor; For a car driver who politely drives conform the maximum speed it is very frustrating if others, who do not adhere to the speed rules, are never punished for their deeds. If a student registers conform the rules of the lecturer, it is frustrating to see that other students who illegally login from their home get the same attendance rate.

QUESTION

During the question phase two different aspects of the feature need to be questioned with regard to the feature goal and the feature hypotheses.

First, two questions about the feature goal should be questioned. The first is of the form: *"How can a minimum viable feature be implemented in order to fulfill its goal?"*. Applying this to the Your Next Concepts example the question would be: *"How will the minimum viable feature be implemented in order for the customer to have more reliable attendance information?*". The next question will be of the form: *"How can we measure whether the implemented MVF fulfills its goal?"*. Applying this to Your Next Concepts will result in the following question: *"How can we measure whether the attendance information is more reliable as before?"*.

Secondly, the hypotheses related questions should be stated in the form of: "*How can the feature hypotheses be answered?"*. For Your Next Concepts this will be:

- Hypothesis 1: "How can we measure if all customers experience value in the improvement of attendance information reliability?";
- Hypothesis 2: "How can we measure if improving attendance information reliability will add value to the end-user of type lecturer?";
- Hypothesis 3: "How can we measure if improving attendance information reliability will add value to the end-user of type student?".

DEFINE IMPLEMENTATION AND SET METRICS.

During this phase, the development team answers the first question and sets metrics for answering the subsequent questions.

As can be seen in the previous section, the first question is a different kind than the other questions. The first questions are asked from a development perspective. The team needs to think about a specific solution to answer the features goal. In certain cases, the MVF implementation solution needs to be created from scratch. In other cases, a relevant stakeholder has a clear vision on how to answer the questions. The latter is the case for Your Next Concepts.

Your Next Concepts already created a feature which assigns a certainty level to an attendance registration. This certainty level is based on the location of the check-in, which is derived from the IP address, and the time of check-in. In order to create an even higher level of reliability, Your Next Concepts has the idea to create an additional feature which automatically asks the lecturer to verify attendance registrations which are classified as unreliable. The notification is send by mail. This mail holds a link to the Academy Attendance platform where the lecturer can verify whether the student really did visit the lecture.

After the answer on how to realize the feature goal is stated in the form of an implementation plan, one can start thinking on setting metrics to answer the following question. We have already stated that the second question was: *"How can we measure whether the implemented MVF fulfills its goal?"*. Useful metrics which can aid in the answering of this question are the following:

1. The click-through rate of the lecturer to the verification page;
2. The number of times the lecturer really verifies an attendance administration;
3. The number of times lecturers who did not get a notification go to the verification page;
4. The number of times a lecturer will go to the verification page without the link notification, after he once verified students.

By verifying the attendance of students, the lecturers make the attendance data more reliable. The first two metrics measure the increase in reliability by means of a mailing feature. The third metric measures whether there is also an increase in verification by lecturers if there is no mailing feature implemented. The fourth metric measures whether there is a learning curve after the first mail with link has been send.

Subsequently, the metrics need to be set for answering the remaining questions. The metric can be derived from the questions. When looking at the question: *"How can we measure if all customers experience value in the improvement of attendance information reliability?"*, a suitable metrics would be the perceived product value from customers which have the feature goal implemented and the perceived product value from a control group that has no solution or a different solution to the feature goal implemented. Note that in order to answer this question, these metrics need to be combined with the former metrics which are linked to the feature goal fulfillment.

In a similar way, metrics can be derived from the other questions which result in:

- The amount of perceived product value from end-users of the type lecturer which have the solution to the feature goal implemented;
- The amount of perceived product value from end-users of the type lecturer which do not have the solution to the feature goal implemented;

- The amount of perceived product value from end-users of the type student which have the solution to the feature goal implemented;
- The amount of perceived product value from end-users of the type student which do not have the solution to the feature goal implemented.

CHOOSE DATA COLLECTION TECHNIQUE

After the metrics are set, one can start thinking which data collection technique is the most suitable in order to implement the metrics. For metrics which are related to customer value, qualitative data collection techniques are the most suitable. Metrics which are related to the feature goal implementation can most often be implemented by means of a quantitative data collection technique.

In case of Your Next Concepts, the metrics which are derived from the feature goal are implemented by a combination of logging and A/B testing. For example, the metric: *The amount of mail notifications a lecturer will receive,* is implemented by logging the amount of notifications a lecturer will receive. By logging these metrics, Your Next Concepts can deduce the amount of times a lecturer has successfully verified attendance information and compare this with the control group.

The metrics which are related to perceived value will be realized by using qualitative techniques. In an ideal world, Your Next Concepts would have implemented an in-product survey which automatically registers data. However, the developing resources are scarce within Your Next Concepts. Therefore, a real-life survey is chosen.

IMPLEMENT MVF

In this phase, the MVF is implemented together with logging functionalities that realize the drafted metrics.

COLLECT DATA

After the MVF is implemented, usage data will be collected that are linked to metrics which can validate the MVF goal. Data about customer value is collected by means of conducting surveys with relevant users. In case of Your Next Concepts these users will be lecturers and students.

VALIDATE FEATURE IMPLEMENTATION

Before validating the hypotheses that are set around the feature, the implementation needs to be validated. If the MVF is not correctly implemented, the validation of the hypotheses cannot be done. For example, if no perceived value is found between users that are exposed to the MVF or not, the feature goal may be indicated as not valuable. However, it may be the case that the feature goal holds a great potential but the MVF is not correctly implemented.

In case of Your Next Concepts, one can investigate by looking at the metrics whether the lecturers actually confirmed attendance. If this is the case, the attendance information is more reliable as before. Therefore, the feature goal is fulfilled to a certain degree. Note that the fulfillment of a goal is not always binary of nature.

VALIDATE HYPOTHESES

In this phase, the hypotheses concerning the feature are validated or rejected. This phase can only be initiated when the feature goal is somewhat satisfied. The specific manner on how to validate hypotheses is dependent on the metrics and data collection technique.

CLASSIFY FEATURE

The Kano model is a direct inspiration for the classification of features in this phase. Features are classified by looking at the features' behavior in terms of implementation effort and perceived value. The first two data points can be derived by comparing the perceived value of a feature that is not implemented with one that is. Possible feature classes are:

- Must-be: When implemented the customer is neutral. When the feature is not implemented the customer is very dissatisfied.
- One-dimensional: This feature results in satisfaction when completed and dissatisfaction when not implemented.
- Attractive: When the feature is not implemented the customer is neutral. When the feature is implemented the customer is very satisfied.
- Indifferent: The implementation of the feature does not affect the customer's satisfaction in any direction.
- Reverse: The more the feature is implemented, the more the customer is dissatisfied.

The classification of the MVF helps stakeholders in deciding whether the MVF needs to be further developed or "sunsetted". The specific class also gives a priority in doing so. Note that for a new MVF only two data points are known. Every time a feature undergoes a method iteration, a new data point can be added with regards to customer value and implementation effort. When a feature has more data points, the feature can be classified with more certainty.

APPENDIX B: PDD

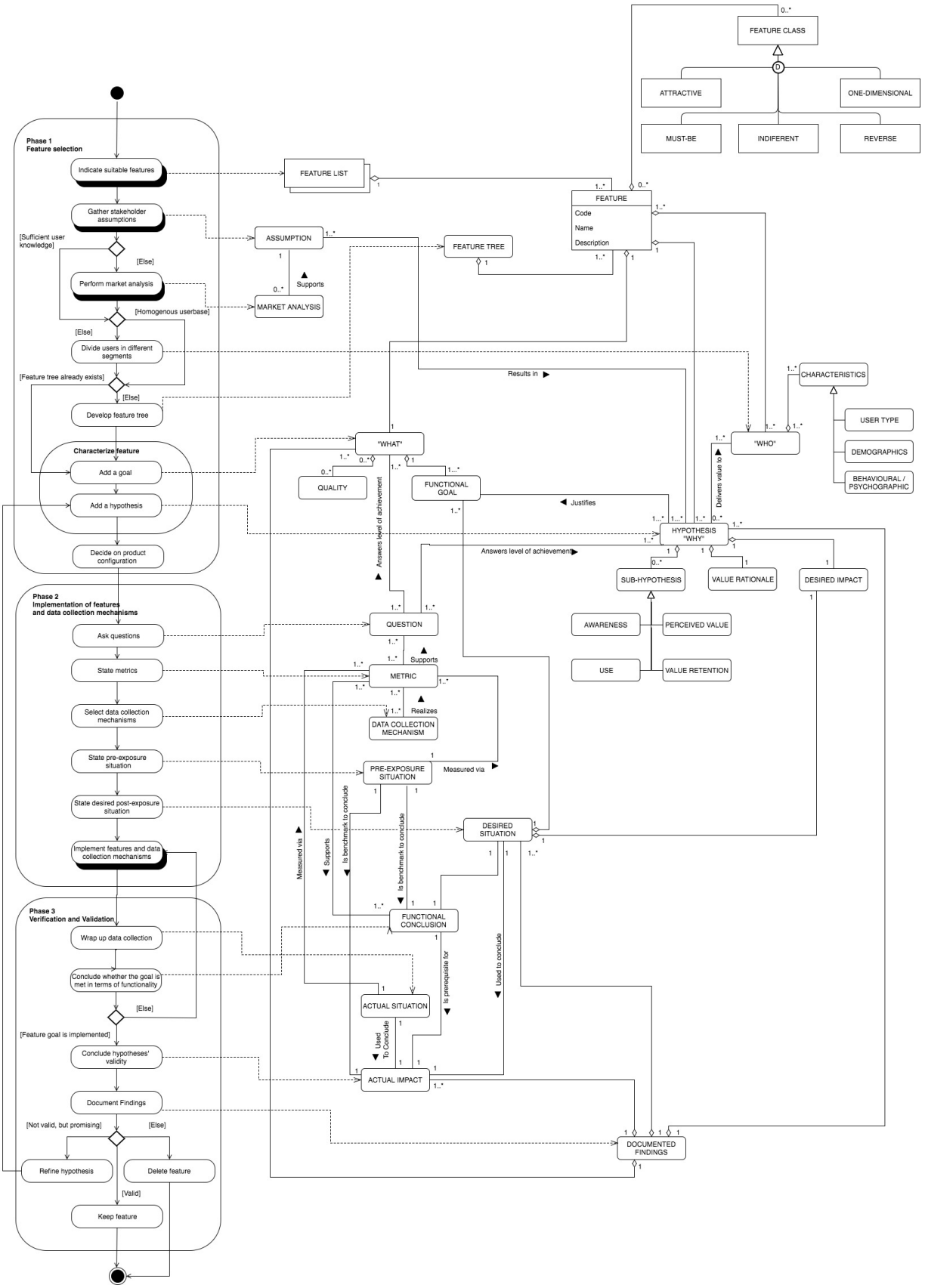


Figure 33, A Process Deliverable Diagram from the final method iteration