CHALMERS
UNIVERSITY OF TECHNOLOGY

Data-Driven Decision Making in Requirements Engineering

How Simulated Data Can Be Used for Data-Driven Requirement Validation and Elicitation

Master thesis in Software Engineering

Axel Bergrahm, Oscar Johansson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master thesis 2020:June

Data-Driven Decision Making in Requirements Engineering

How Simulated Data Can Be Used for Data-Driven Requirement Validation and Elicitation

AXEL BERGRAHM
OSCAR JOHANSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Division Software Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

Data-Driven Decision Making in Requirements Engineering
How Simulated Data Can Be Used for Data-Driven Requirement Validation and Elicitation
AXEL BERGRAHM, OSCAR JOHANSSON

Supervisor: Richard Berntsson Svensson, Department of Computer Science and Engineering
Examiner: Regina Hebig, Department of Computer Science and Engineering

Master Thesis 2020:June
Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Data-Driven Decision Making in Requirements Engineering

How Simulated Data Can Be Used for Data-Driven Requirement Validation and Elicitation

Oscar Johansson and Axel Bergrahm
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Requirements engineering is a discipline that is exploring the opportunities in becoming data-driven. Since requirements engineering is a decision-rich field, it is drawing inspiration from the activities of data-driven decision making. The activities make use of data in different regards to improve the activities conducted within requirements engineering. This thesis makes use of simulated data from industry and investigates which factors that are important for improving decisions for requirements engineering. To achieve this, a method for data-driven requirements engineering is developed and is used as a catalyst for deriving these factors. The result shows a set of important factors to consider making data driven requirements validation and elicitation

Keywords: Requirements Engineering, Data-Driven Decision Making, Visualization, Data-Driven Requirement Engineering

Acknowledgements

We would like to bring out a special thanks to our supervisors Richard Berntsson Svensson and Markus Berglund for the support during the Master Thesis. It has been invaluable.

To Richard, we'd like to give you a special thanks for all the late evenings, and weekends, that you endured us and our pesky questions over Zoom. Also we need to thank you for sharing your knowledge and inspiration for sources that helped us make better decision in the thesis.

To Markus, we want to say thank you for all the hands-on support with all technical details, help with ideas, making us build a computer, and making certain that we always had the best conditions to push our work further.

Finally, we would also like to thank everyone at the Huawei Research Office for being so cooperative and participating fully in our studies, giving us support throughout the project with all positive energy. It has been a treat!

Oscar Johansson and Axel Bergrahm, Gothenburg, 7/June-2020

Contents

List of Figures

1
Introduction

Becoming data-driven is an industry trend that is finding its footing. The discipline of data-driven decision making (DDDM) has many definitions but the one used in this thesis is: "Data-driven decision making is seen as a continuum in which data must be transformed into information and, ultimately, actionable knowledge through a set of cognitive skills and processes [1]." Further breakdown of DDDM can be described as five activities, namely; data recording, data cleaning/integration/representation, data analysis, data visualization and decision making [27][30]. The DDDM activities are often juxtaposed with data science and big data with regard to decision making and their respective benefits [5], such as *productivity* or *profitability* [27][37]. Improvements from all these activities are also becoming apparent in a wide array of fields [6][7][8][21]. Moreover, as more fields are seeing improvements from becoming data-driven it is natural that others will attempt joining suite to reap their potential benefits.

Requirements engineering (RE) is a discipline for managing requirements [16]. As a discipline, RE is a decision-rich process and is being explored with activities from DDDM [46]. A single requirement in RE is defined as a condition or capability needed by a user to solve a problem or achieve an objective [42]. Holistically, RE consists of six activities for managing requirements, namely: requirement elicitation/inception, requirement analysis/negotiation, system modeling, requirement specification, requirement validation and requirement management [16]. This research investigates two of the activities closely: validation and elicitation, as they are closely linked to decision-making activities. Current research is investigating possibilities for making requirements data-driven, where RE and DDDM activities are being combined and aims to pinpoint the changes to the practice it would bring [21][24]. Emerging from these attempts comes the discipline of data-driven requirements engineering (DDRE) [20]. It is believed that making requirements more data-driven will bring benefits to the discipline but the challenge is to derive how [27].

To the best of our knowledge, there seems to exist a gap in the research where practical attempts are made to understand what factors are important for combining DDDM and RE. Most research centered around RE activities are based solely on software systems [22][14]. Therefore, to fill this gap, we conducted a case study [38] using design science [19] at Huawei, Gothenburg, where the decision-making process is done by the users. Conforming with the activities in DDDM and RE, a DDRE method is developed with a focus on validating and eliciting requirements with data.

1.1 Project Description

Huawei Sweden is currently developing a software artefact named *Coffee Grinder*, which is a simulation tool for 5G base-station placement. The simulations produce a lot of data and the need from the company's perspective is to be able to visualize the data and validate their internal requirements.
The project description from our perspective this is to create a partner software artefact to the Coffee Grinder's simulation results. Into the artefact, well-formed techniques from the research domain are matched with the functional needs of the employees from the company. An array of techniques are implemented and evaluated during the span of the project, if they are mismatched during an evaluation, other techniques are tested for fitness to the company domain.

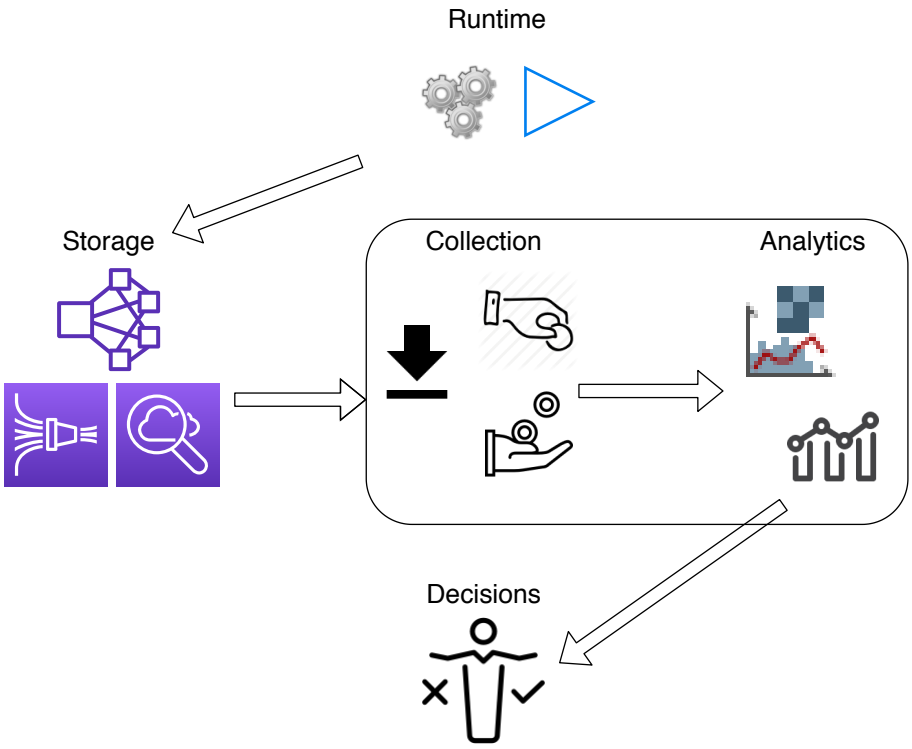


Figure 1.1: Project process, where the highlighted square contains the parts this research mainly focuses on

1.1.1 The Coffee Grinder

The Coffee Grinder simulator evaluates trillions of potential paths in a 3D environment, these paths are first reduced by GPU acceleration and then further reduced in CPU post-processing. Still, the output of the simulator contains vast amounts of information. To be able to make the right decisions for the users, this information needs to be presented in a way to assist the decision making. Statistics, 3D visualizations, graphs etc needs to be generated from a large number of different variables. The project aims to create a Python-based application that loads simulation results from Coffee Grinder and visualizes them according to the needs in the industry.

This method will be an essential feature of the simulator for the projects in 2020 and forward.

Given the Coffee Grinder, the problem statement would implicitly be that we implement the project description and embed a domain-specific decision support system. This support system will be confined to only collecting and implying data directly mapped to requested, necessary, decisions points. Moreover, the project implementation itself will be used for decision-making and thus we may have an interesting comparison point to be made if the proposed decision support system performs better or worse than the envisioned tool for visualization alone.

For Huawei, this will potentially give the stakeholders a stronger basis for making decisions about their 5G development in the future. The data will be used as performance indicators for 5G technology. Ideally, the data will be packaged in a way such that it can be used for new requirements and decision-making on some designed area of interest, both in traditional and novel ways. Finally, applying the thesis idea onto a system like this can imply assertions on how well suited the thesis is for similar projects. If the idea is successful it may serve as a steppingstone of encouragement to conduct further research in developing these product-specific subsystems that are separated from the concern of outside-acting variables. Figure 1.1 shows how our project process looks like and the highlighted square is where we have put most of the focus in the research. The simulations generate simulation data during run-time that is stored in a database structure. That data is then collected by the software and visualized so it can be analyzed by the users. The insights from the analytics are then used to make decisions regarding a certain algorithm or setup. This process can also be seen as a validation process because the KPI:s are used to see if a certain algorithm meets the customer needs. These are values that are attached to a requirement, which means that the users are checking if the KPI:s are within certain KPI target values and that reflects what the customer wants.

2
Background and Related work

This chapter provides background information and related work for the thesis. Important topics the thesis grounded inspiration from and is supported by is brought up in the following sections.

2.1 Data Driven Decision Making

Data-driven decision making (DDDM) has many definitions in the literature, according to Mandinach et al [40] DDDM highlights the salient role of data in decision making which refers to choosing a course of action from a set of options based on data. Despite a large amount of data available, decisions from managers are typically subjective, frequently inconsistent, and often lack explanations as well as links to which data and evidence they were based on [27]. The idea is that if there exists evidence for why a decision should be better than another, then the rationality is to conform with the evidence and let the data speak. The practice of DDDM can be divided into five activities, namely: data recording, data cleaning/integration/representation, data analysis, data visualization and decision making [30]. Data recording activity envelopes storage of data, data cleaning picks out the relevant data and removes unwanted data, data analysis is about getting an understanding of the data that is then visualized in the data visualization step. Finally, the insights given by the visualizations create the basis that the decisions are made upon. In practice though, it is not always possible, nor practical, to only make decisions on data and is:

> *"best used by those who reflect on their own practice as a mechanism for improvement."* [23]

Meaning that in the end, someone's judgement should be evaluating the data and the potential worst outcomes from decisions made. Most of the research made has been in an educational setting [39], and some research has been made in agile software setting [27], but more research has to be made with respect to other fields. Fields where data-driven practices are embraced to full extent include *Business Intelligence* (BI) [2][3], *Big Data* (BD) [4][5] and *Data Science* (DS) [5][13].

BI can be categorized as "a data-driven decision support system that combines data gathering, data storage, and knowledge management with analysis to provide input to the decision process" [9] and often is used to predict a business' success.

BD can be categorized as a set of practices or activities to achieve a goal based on the handling of a lot of data. Many of the activities in BD is overlapping with the activities in DDDM and the terminology but with respect to decisions the terminology used is usually *Big Data Driven* (BDD). When *BDD* is decomposed its end result practically become data analytics and decision making. [4]
DS can be categorized as "an amalgamation of classical disciplines like statistics, data mining, databases, and distributed systems" [13]. The DS discipline "requires both domain knowledge and a broad set of quantitative skills" 12 and when scrutinized, the work from the DS discipline could be interpreted as the narrative from data for decision making because it ultimately entails the activities from DDDM to a large extent [5].
These mentioned fields (and more) are all comprised of the activities derived from DDDM. Linguistically they all also seem to have in common are references to *key performance indicators* (KPI:s) [2][10][11]. A key performance indicator could be viewed as a composition of indicators that predicts performance onto a given model [2]. Meaning that a KPI model formed should be able to predict performance on some decision process or in other words "KPIs are quantifiable metrics which reflect the performance of an organization in achieving its goals and objectives" [15]. A lot of work is currently conducted in mapping out these KPI:s with respect to their field and implications, some fields are more mature in adopting DDDM practices and others are in a more exploratory phase in the adoption.

2.2 Requirements Engineering

Requirements engineering (RE) is a discipline for handling requirements. Single requirements are defined as a condition or capability needed by a user to solve a problem or achieve an objective [42]. The discipline of RE in its entirety entails *handling* requirements as the following six activities: Requirement elicitation/inception, requirement Analysis/negotiation, System modeling (coupled to software systems in the book), requirement specification, requirement validation and requirement management [16]. Naturally, all six activities take form depending on the environment the requirement discipline is practised in. For example, the requirement validation processes are not necessarily exactly the same from one company to the next and neither any of the rest of the activities. Apart from this, requirements engineering is a decision-centric process and most of the decisions in this context should intensively rely on empirical evidence and data [46].

Further RE abstraction separate requirements into *functional* and *quality* (non-functional) requirements [16]. A functional requirement in software engineering is a statement with respect to the functionality of a system, think of it as the functions the system performs [34]. Quality requirements can be seen as the adjective describing the functions of a system, a few examples could be usability, reliability and performance. A definition of quality requirements from an ISO standard [28] categorizes them as a discipline of RE that encapsulates requirement *attributes* such

as mentioned above.

With this said, research has been done identifying KPIs as measurements to validate requirements within 5G development. In that case the KPI:s serve its value as target values that will reflect the customer needs. In the research done by Tilemachos et. al [50] several use case requirements are used as speed for the networks, which has a KPI attached that tells how fast the network should be to fulfil the customer needs. According to Tilemachos et. al. [50] KPI values from vertical leaders in smart industry, city/health and energy sectors provide an indication of the expected added-value of 5G by these sectors and assist the telecom operators and vendors in designing their upcoming 5G networks and services. [50]

2.3 Data driven requirement engineering

Conventional Requirement Engineering(RE) decisions are usually based on user feedback from interviews, workshops or focus groups. Data-Driven Requirement Engineering (DDRE) instead focuses on how developers, analysts can use data as a basis to identify, prioritize and manage requirements. [21]

The data used for DDRE can be different depending on what type of decisions needs to be made, the data required is coupled with the type of requirement that is under scrutiny. Walid et. al. [21] addresses the importance of implicit and explicit data provided by user feedback. The explicit feedback is defined as feedback that the user gives intentionally (i.e ratings, reviews) and the implicit feedback is defined as feedback that is indirectly implies something (i.e clicks, purchases)[41]. This is two ways to gather data in which can be used to get a better picture of what the user needs and from that data, elicit requirements.

Apart from looking into the user feedback, data mining from the software system to improve software development processes. One of them is the Q-Rapids approach that gathers runtime data and use this data for creating dashboards [22]. The dashboards are then used to gain insights that will elicit requirements and add new tasks to the backlog. This is allowing the developers to create a basis for decisions and still have alternatives that the data points at. Another way of getting requirement insights is by monitoring a system's behaviour in order to get a better understanding of how to improve the performance of the system[54]. Research has also been made on how to automatize the decision process by using a graph-based approach to automatically elicit requirements [24].

Regarding potential future research in DDRE, Xavier et al.[49] addresses different directions the research in DDRE can take in the future. The different directions include for example a model-driven approach that accomplishes DDRE by making use of Domain-Specific Languages(DSL). Xavier et al. [49] also suggests a research direction to utilize advanced data analysis capabilities and decision support systems

where he addresses that analytic can help with aspects as customer satisfaction, risk, or time-to-market.

2.4 Visualization

Data visualization is one of the important aspects of data-driven decision making [30]. Apart from improving the quality of decisions by processing the data correctly, visualization also plays an important role in supporting decision-makers [48]. Previous research has been made on finding frameworks that supports visualization design [25] and finding ways to evaluate visualizations [26]. In the research conducted by Handcoff, the process of visualization design is described and provides a step to step process of creating visualizations. These steps can be summarized as project conceptualization, data characterization, visualization design, visualization development and development use. Handcoffs research also mentions important hurdles that are important during the design process. These hurdles are mainly adaption of data changes, anticipating edge cases, handle data-dependent interactions and handle of data mappings.

In order to know which visualization technique to use, we looked for patterns in the data should be considered to maximize the insights from it. Different methodologies and conceptual background is provided by [26]. These metrics are made for commonly used plots (Line Charts, Pixel-based Techniques, Geo-Spatial Data etc.). These plots were in this research evaluated by their ability to identify 4 different patterns, which included identification of Trend, Correlation, Outliers and Grouping of data. The importance of identifying visual patterns is illustrated by the following quote:

> "*The essence of effectiveness resides in the identification of interpretable visual patterns that contribute to the overarching analysis goal.*" [26]

Previous research has been conducted on creating data collection and visualization simultaneously [31], which is something this implementation grounded some inspiration from to make the bridge more dynamic between data collection and visualization [32].

3
Methods

This chapter entails all the methodological practices for producing the results of the thesis. It begins with stating a research objective and its following research questions. A general time-axis of the research methodology is presented with a visual for how the project panned out with respect to the method.
Finally, the prestudy, first, intermediate and final prototypes are dealt with in individual sections; respectively including the research practices, procedures, techniques for evaluating the results and how prioritizations are made.

3.1 Research Questions

From the activities of DDDM and RE, we see an opportunity to explore what is important for combining the disciplines. To conduct an investigation we propose the following research objective for finding the important factors when conducting DDRE:

> **Research Objective:** To formulate a method supporting data-driven decision making for requirements validation and elicitation.

Within the research objective, the aim is to investigate what is important to consider when mixing the disciplines. We adopt a data-driven approach in which we gather and analyse data from simulations to identify new requirements (elicitation) and to use the data to validate identified requirement. Therefore, we decompose the research objective into two research questions. We need to identify what is important to consider (called factors) in a method when adopting a data-driven approach for requirement validation (RQ1) and requirements elicitation (RQ2). The following research questions are investigated:

- **RQ1:** What factors are important to consider for data-driven requirements validation?
- **RQ2:** What factors are important to consider for data-driven requirements elicitation?

Further, an evaluation of the research objective is conducted to its usefulness. The usefulness is based on opinions from the users of the method and is compared to related research.

3.2 Design Science Research

The design science research methodology can be referred as the process of *learning through building* [18]. It is a method that is encapsulated by problem-solving from identifying organizational problems [19]. The design science framework used for this thesis is proposed by Hevner et. al., (2004) [19] and is shown in figure 3.1.

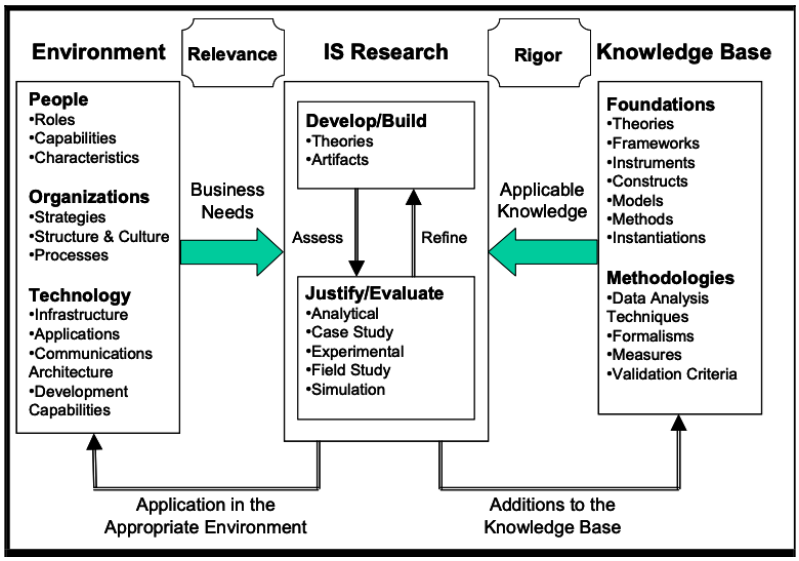


Figure 3.1: Design Science practice followed as proposed by Hevner et. al. [19].

Figure 3.1 tangibly show the connections between the environment, information system research and the knowledge base. Generally, design science couples domain knowledge (environment) and theory (knowledge base) to build a model (IS research) from which we learn from. The model is then refined iteratively and the learning's from each iteration are given back to the domain and theory. Decisions on the actions carried over to the next iterations are based on the research objective and its ability to answer the research questions. These refinings or actions are derived from a summarizing activity at the end of each iteration, also when the final cycle has been conducted a complete summary of the holistic result is made.

Design science has been chosen for this thesis because we are striving to match theory to a domain and their identified problem. The functionality of the developed method is elicited iteratively from the domain and matched with applicable theory. This project is encompassed with three iterations (design cycles), but first, a prestudy is conducted to attain domain knowledge. The design cycles include the following phases: First Prototype, Intermediate Prototype and Final Prototype, see figure 3.2 for a visual on the sequence of the project.

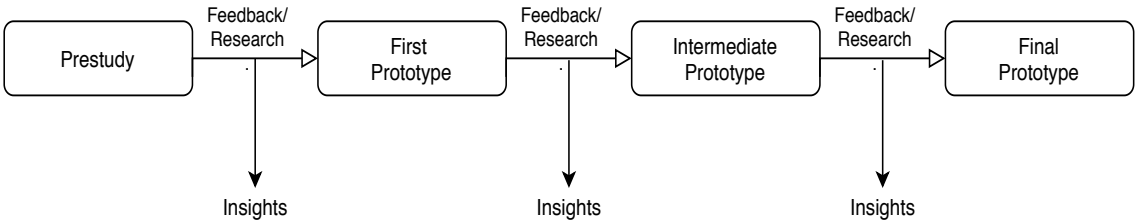


Figure 3.2: Research iterations for the different prototypes. Each of these phases are described in further detail in subsequent sections.

In short, each prototype is based on the knowledge gained in previous phase ergo the first prototype is based on knowledge from the prestudy, the intermediate prototype is based on knowledge gained from the first and the final prototype is based on knowledge gained from the intermediate prototype.

The participants in the interviews and the workshop are experienced engineers and researchers within telecommunication. They have between 4-20 years of experience within the field and most of the participants have PhD:s or masters degrees within disciplines such as electrical engineering, computer engineering and telecommunications. Due to historic events (see 6.4.4.1) taking place in the world, the two last cycles of the research (intermediate and final prototype) was conducted remotely.

3.3 Pre-Study

The pre-study is done by conducting semi-structured interviews and on-site observations. The interviewees have different backgrounds and responsibilities but due to agreement with the company, we can not disclose personally identifiable information. Five interviews are conducted with employees within the timespan of 20 to 40 minutes each and are recorded through dictation in writing. The questions during the prestudy included how the users handle their requirements and what kind of visualisations the users are familiar with. Apart from that the potential KPI:s was found by asking the interviewee about the importance of the different KPI:s, which is important for requirement validation [50]. In order to know which quality attributes to use for requirement elicitation, questions about system needs were asked for. The goal with these interviews is to get a better understanding of the domain and use that knowledge as insights for the visualizer during the set-up time required for creating the first prototype. Further investigation during the prestudy entails regular conversations and relaxed questioning of *wants* and *needs* regarding the method developed. All knowledge gathered in the domain is documented for our own use as feedback in the method. During this stage, we combined the domain knowledge with visualization design [26][25] in order to find the right visualizations to use in the software artefact.

3.3.1 Analysis

Feedback from the prestudy is used to find the right type of plots and what KPI:s that are important to prioritize. Further, the feedback also helped us to identify what data can be used when eliciting requirements. For functional requirements, a usage logger is decided to be used together with a performance visualizer that supports the elicitation for quality requirements. The performance attributes are divided into two parts, where we investigate memory usage and time efficiency for specific tasks. Also from the feedback, a sufficient understanding of the domain is established as a foundation the method to be developed. During the establishment of the understand, usable data is documented by us to make it easier to understand the KPI:s. This documentation is solely used for our own purposes such that we can match the domain needs [29].

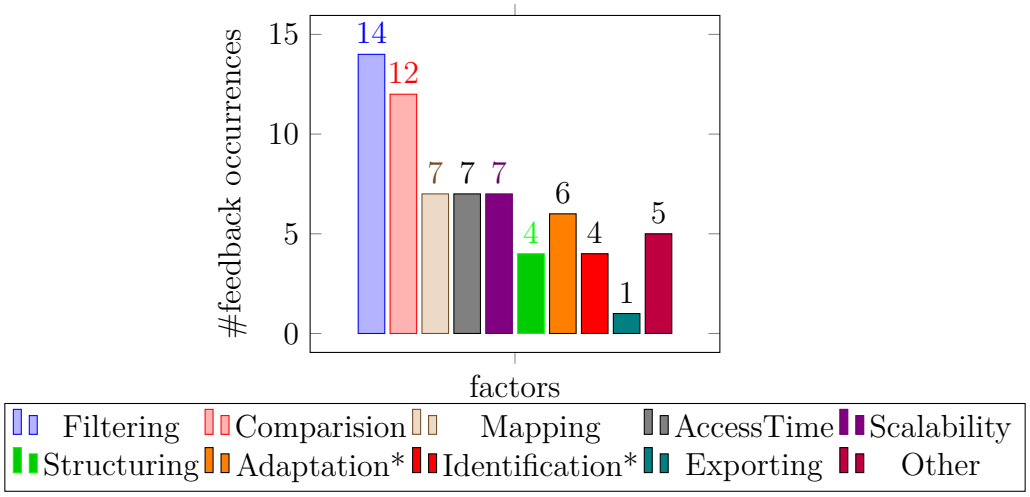
3.4 First Prototype

In this design cycle, we developed a method for DDRE based on the feedback we got from the prestudy. Again, following suit with the data gathering process from the prestudy, semi-structured interviews are used for data gathering again. Seven interviews are conducted within the time span of 60 to 100 minutes each, where six out of the seven participants also participated in the prestudy. The interviews are dictated in writing and the feedback can be found in Appendix A.

The interviews are conducted by first demonstrating the method to the user, and then eliciting the interviewees if it provides value for requirement elicitation and validation. Lastly, we ask what the user would like to add in order to accomplish better requirement validation and elicitation. The goal with the interviews is to verify that decisions on validation and elicitation can be made, to some degree, with help from the prototype.

3.4.1 Analysis

After the interviews are conducted, the feedback is dictated on notes and then categorized in factors via content analysis [17]. What this means is that factors are found by grouping the feedback and see what they have in common, which in this case is factors. The categorization of the feedback gave us an way to quantify how important some factors are to the users when validating and elicitating requirements. Both depending on frequency and relevance to the research, the factors are used for creating actions that are performed in the next design cycle. The feedback received in this design cycle allowed us to both strengthen our domain knowledge deciphered from the prestudy and helped us pinpoint which areas we still need to understand better. During this cycle, the several factors are identified (see Figure 3.3)



*=For requirement elicitation

Figure 3.3: The total number of found factors by looking at feedback occurrences per factor

3.5 Intermediate Prototype

The intermediate prototype is a continuation of the method from the first prototype. The elicitation part of the method was placed on hold because of low priority and therefore only requirement validation is tested in this cycle. This iteration, the data-gathering is conducted differently. To avoid bias if one feedback may be more important then others a *workshop* is conducted where a larger group of 14 employees is gathered in one session of circa 70 minutes, including six out of seven participants was interviewed during the first prototype cycle. The workshop consisted of a demonstration of the method, a feedback session and a prioritization session. During the demonstration we show off the software artefact in the state that it is in, and all of the modules that is created to support requirement validation. During the feedback session we take notes on the feedback and use that for categorizing them into factors via content analysis. The prioritization session is used to categorize what is important and was made by letting the group agree on which factors they saw as most important to focus on going forward in order to achieve efficient requirement validation. Since the setting is different for the data gathering this time, another priority categorization system is used. The intention of this session is to gather a wider array of feedback and have the participants discuss promising potential prioritization going forward. The prioritizations mainly involves the factors that the feedback is categorized within. A factor in this case could get a priority of high, medium or low priority. This both helped us to see which factors to focus on in the next iteration and which factors are important for the users.

3.5.1 Analysis

The feedback from the second prototype contains both the prioritization of different factors and a list of feedback regarding the artefact itself (see Appendix B). The prioritization showed that the priority pointed towards the same result as the feedback occurrences, but with some small differences. We chose to have a higher priority on the priorities session from the collection of users than the individual feedback occurrences because more stakeholder opinions are voiced in this session. A set of factors are presented and the stakeholder's voice which one they would like to see. Democratically the opinions are collected and compiled into priority clouds based on the frequency of the opinions (see Figure 3.4).

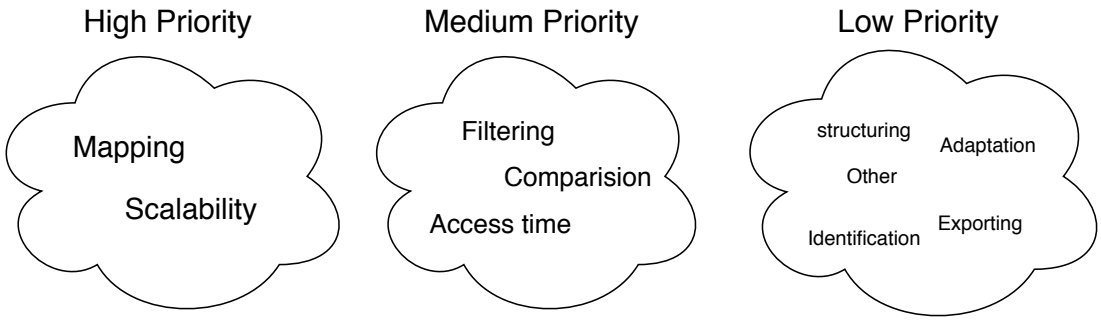


Figure 3.4: The different priority levels as a result from the conducted workshop

After the prioritization session, the attained knowledge from the domain points towards that most users wanted data mapping and scalability, these are prioritized for the final prototype. Medium priority items included more filtering and comparison options combined with optimizations to access time. As these items are not prioritized, they are given back to the company as knowledge for how to improve the method in the future. The same goes to the low priority cloud, which includes structuring, adaptation, exporting and identification.

3.6 Final Prototype

The final prototype is the last design cycle performed in the thesis. Each interview lasted around 30 to 50 minutes, where all of the participants also participated in the first prototype cycle. This cycle intends to act as a final evaluation and feedback cycle for the thesis (see Appendix B). For this cycle, we defaulted to the same data-gathering method as in the first prototype cycle. This included a demonstration of the method and different techniques. These techniques included a DSL model and a Python interface that both are developed for achieving mapping and scalability. After the demonstration, the users gave us feedback about which of these techniques they prefer and why. Apart from this, feedback about the DDRE method as a whole was dictated by taking notes. Again, the sample of interviewees have different backgrounds, and five interviews are conducted with a semi-structured technique.

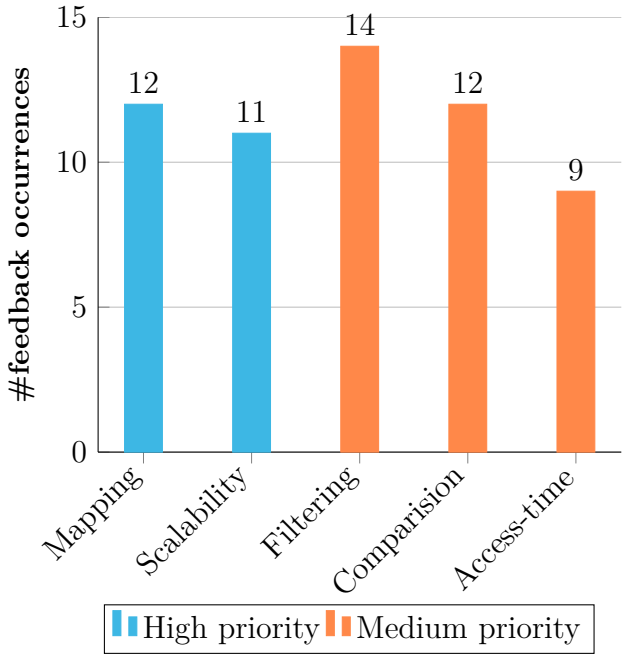


Figure 3.5: The five most important factors for requirement validation found by priority and feedback occurences

3.6.1 Analysis

During this stage, all of the data from interviews and the workshop is gathered and analysed to find the holistic answer for the research questions. This summarizing activity serves the purpose to find *all* the factors found for requirement validation and elicitation. The factors are mainly about requirement validation because only one design cycle handled activities for elicitation and those activities were deemed low priority. Below are the most common factors that were found after analysing the aggregated collection of feedback data for validation. The five most important factors can be found in Figure 3.5 and the feedback/categorizations can be found in Appendix C.

4
Development of Method

This Chapter details the software architecture of the developed method and each phase of development from first to final prototype. Each prototype contains details of the modules developed and finishes with the actions taken to the next iteration of the project.

4.1 Software Architecture

The prototypes for the DDRE method uses a Model View Controller (MVC) design pattern as a software design framework, an overview can be seen in Figure 4.1. The MVC pattern contains a model, which manages data, logic, and rules of the method. The controller dictates what the model should do and the view is the actual representation of the data. This architecture provides the flexibility to split the backend and the frontend so it is easy to add new features and work on several software artifacts at the same time.

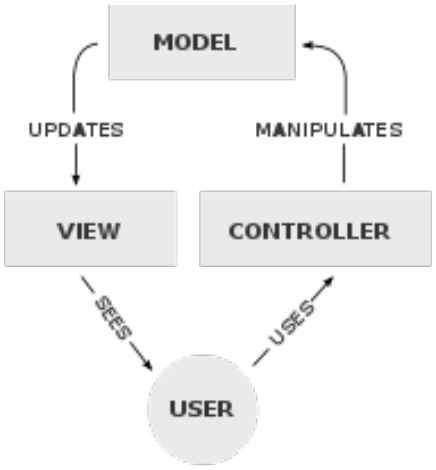


Figure 4.1: Traditional Model View Controller (MVC design)[52]

Even if we used the MVC pattern as inspiration for the design, the final architecture ended up being a MVC variant more suitable for data visualization, see Figure 4.2. The controller in this case was used as a mediator for all the components. The interactions from the users start from the view where the users interact with the GUI. The GUI itself contains several components where every component is a customized, self-contained, widget window. The controller keeps track of every

widget in order to know what each widget should do. When a interaction is made with the GUI, it goes through the controller that check what kind of mediation that should be forwarded.

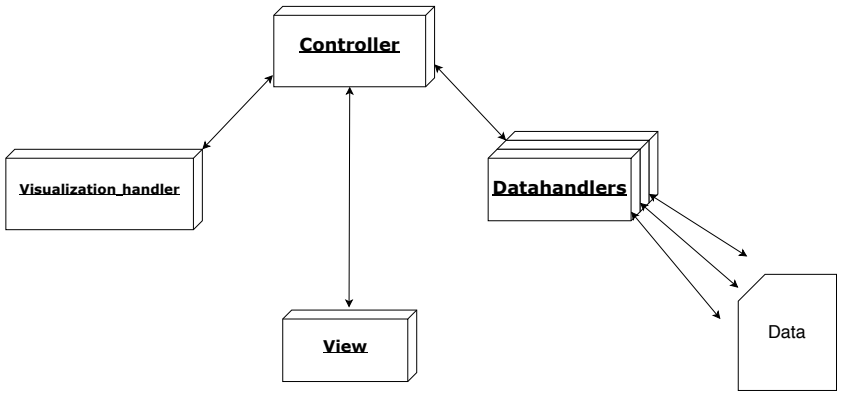


Figure 4.2: Software architecture used in for the DDRE method for the intermediate and final prototype

The data-handler collects a set of data and grabs the data from where it is stored. When a certain data is selected to visualize from the GUI, the controller tells the data-handler to prepare and structure the data, so it will be ready to visualize in the future. The reason why several datahandlers are used is to enable parallel execution and provide a faster execution during the preprocessing of the data. The parallel execution also let the user to do other tasks without having the software artefact freeze while operations are loading.

When the data is prepared in the datahandler, the controller uses the visualization handler to generate a certain visualization. Depending on what action the user performs in the GUI, the controller will grab that data from the correct datahandler and hand it over to the visualization handler. The visualization handler in turn will use that data to generate visualizations in form of figures. These generated figures are then sent back to the controller that will mediate to the GUI to visualize the figure. A design decision made to use only one visualization handler is made because the data is already prepared in the datahandlers and therefore does not take much time to execute.

Apart from the MVC pattern, inspiration is drawn from the Q-Rapids approach [22]. This model used mined data and dashboards that created value in order to give insights for creating new requirements. In this requirement validation approach that is not completely focusing on the software development, our design is a customized variant. Our design has datahandlers that collect the extracted data and the visualization handler that handles the dashboards and visualizations.
Finally, from the knowledge that internal goals may change during the scope of the research, a fleet-footed approach has been made with respect to the view-model.

The software's view model is developed with a component, *widget*, based modularity. This design pattern is inspired as proposed by Facebook's React JS documentation [53]. This architecture's strength lies in that if internal needs change, new self-contained widgets can easily replace, be added to, removed, and be changed if desired without having to alter any dependencies between the code artefacts.
In the first design science iteration, a prototype is developed to test some functional requirements elicited from the users. The three different things to test was data-driven functional requirements, quality requirements, and decision making in relation to requirements validation and elicitation, based on the prototype. Figure 4.3 shows how the flow panned out. Mind that the main task is to make a DDRE method, though two sidetracks are also incorporated that will help the user or developer to find requirements, namely a textual logger and performance visualization based on the Dask API. With the focus on understanding patterns and their corresponding visualizations, the visualization will help the user to validate the data and make it more insightful, which has potential benefits to the decisions made.

4.2 First prototype

The first prototype used the feedback we got from the prestudy. In order visualize important performance indicators, the prototype resulted in three modules that consisted of a Usage logger and performance visualization for elicitating requirements. Finally a KPI Visualizer was developed to support requirement validation. This stage of the prototype intends to verify that we have understood the domain knowledge and can meet some expectations for DDRE, both on validation and elicitation. Further, the prototype is used to elicit feedback from the stakeholders on how the method should be developed further.

4.2.1 Usage logger

The logger is one of the requirement insight tracks for this prototype, which its purpose is to show the user what is happening in the system and makes them aware of what is happening in run-time execution and what functions are used. The idea is to let the user and developer be more aware of system failures and what the user's preferred visualizations are. An example could be that a user frequently uses a specific visualization more often than others and provide insight for the developers on what visualizations to improve upon. These clues could potentially serve as self monitoring feature request just based on the frequency a feature is used and perhaps be indication whether further interactions should be implemented onto the popular feature.

Furthermore, failure reporting is more tailored to the developer to get clues on both where to find and how to fix bugs in the system. The data from the logger can be used to create reports concerning usage that will catalyze the insights. For this prototype basic bar charts are used to compare the frequency between functionality that is used. This can for example be how often a user uses certain plottypes.

4.2.2 Performance Visualization

The performance visualization gives the user an overview over tasks' time and resource efficiency. Time, is measured by how long a specific task takes and the resource efficiency is calculated by number of bytes used per task. A task in this context is mainly considered as a function. Due to the need of high computational power, each task in the system is distributed via Dask's distribution API, which is a framework for distributing computational work to remote data centers. Dask also provides an API to get diagnostic data about time and resource efficiency, which is used in this research. Furthermore, Dask also has some embedded visualizations for diagnostics, though these were not implemented in this research due to dependency issues. Data monitored for the thesis are solely the time and resource efficiency factors. These metrics in turn are used to make plots detailing the correlation between the simulation setup and the run-time execution.

Into the first prototype we decided to visualize the ten most memory draining and time consuming functions, using bar charts. This basic visualization is a means for us to collect feedback on where future potential improvements can be made to better answer our research questions.

4.2.3 KPI Visualizer

The overarching objective with this DDRE method is to handle and visualize generated data from simulations and that data was then used as KPI:s, used for requirement validation. Visualization is a technique for providing a user better insight into data from which they can derive decision grounds. If the visualizations help the users to understand the data, the software artifacts ultimately serves as a decision support system for validating and conceptualizing further requirements.
The validation of certain scenarios and requirements means that the setups are evaluated and compared between the simulations, drawing reasonable conclusions and decision points regarding which setup to go for. One has to keep in mind that there are different parameters to consider when comparing several scenarios, this prototype use these parameters and attempt yielding insight into the setup.

The first prototype allows the user to select which generated data to visualize and presents it to preference. Based on visualization techniques, and feedback, the available visualizations are quite limited but are in a state where they serve well as a proof of concept. Data filtering techniques are also implemented for this design iteration, though only a few are available. Based on what is visualized, the user thus has a certain degree of freedom for controlling what is visualized to their needs.

For this initiating iteration a large portion of the work entails structuring, or preprocessing, the data such that it is accessible architecturally. Based on the prestudy, our understanding from how the users work and would like to use the software led

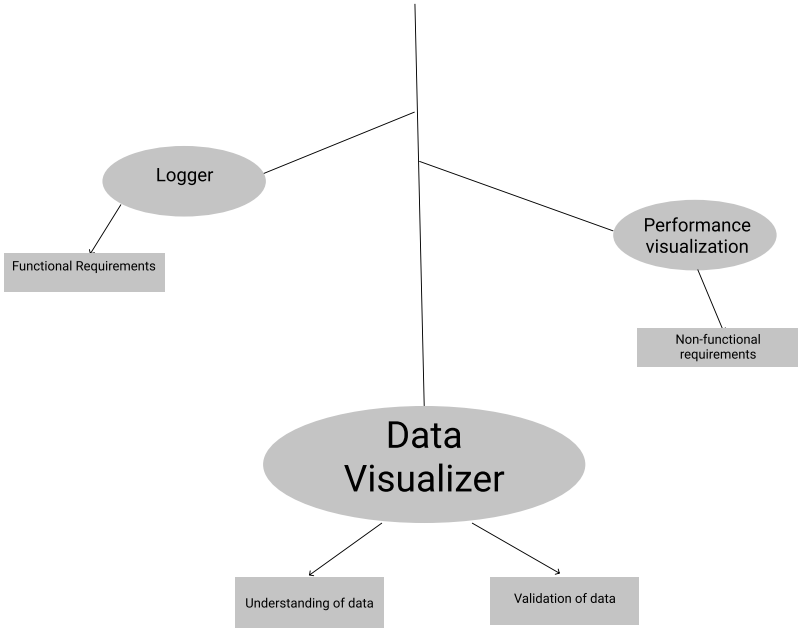


Figure 4.3: The first prototype design. The logger visualized the usage data, performance visualization visualized the resource efficiency and time efficiency while the data visualizer visualises the simulation used for validation

to our considerations for this prototype and how we design the method for further development. The state in which the product is in for this iteration, mostly basic visualizations are available such as the scatter-plots, pixel-maps and line-charts with few interactions are ready. The feedback on these visualization techniques is used for our own decisions on how to progress the research.

4.2.4 Actions for second cycle

From this iteration of the method, a complete list of feedback can be found in Appendix A. The actions we decided to focus on with respect to the complete feedback can be found in Appendix A.1, where each action also is categorized with an estimated value and estimated difficulty by us.

4.3 Intermediate prototype

Due to low priority, the elicitation modules in the method are put on hold. Instead, the KPI Visualizer is prioritized and the priorities within that module included filtering the data, mapping between visualizations, side-by-side comparisons and fast access time. The intention of this state for the method are to elicit the important factors for requirement validation and to elicit further feedback for the next design cycle.

4.3.1 Filtering datasets

Filtering is one of the techniques implemented into the second prototype and it provides the ability to sift data within plots interactively and according to specification. One example of filtering present for the second prototype is a *double slider*, where the user has the ability to *cut the ends off* for sorted data by setting a minimum and maximum accepted value. This slider filtering technique proved useful for most of the already existing plot types, and perhaps most useful in the pixelmap where the user would interactively see the data points disappear when they no longer align with the selected range.

4.3.2 Mapping between visualizations

The mapping between visualizations was implemented by letting the user click on points within the visualization and get the data associated with the points. An interaction which provides the user with an interface which reduces the work required to find and explore exact data points. Moreover a need for selecting multiple points as sought after and also implemented is a *lasso selector*, where the user can click and draw an area for which points to explore. To solve some of these challenging mapping interactions, decisions on the type of interaction could either be connected with the datatype or the type of visualization of the datatype [25]. Both variants are tested.

4.3.3 Comparisons

Comparisons is also a factor categorized as important for this iteration. Two techniques for categorization are considered, one being plots within plots and the other is creating side-by-side views. After some consideration we settled with implementing a grid-view type of solution as specifying what embedded plots are available for every plot type would be a whole lot of manual labor which practically would be infeasible for the time constraint given. Thus, the grid option is practically an extension to the current architecture where every plot-canvas widget are self contained components but are all encapsulated within a larger widget and our solution for this allows the user to create as many plots in the grid as they please. So accompanying the solution, the users have the ability to set up their plot grid to personal preference, both in size and locations for the plots, and that in itself may prove valuable to some extent.

4.3.4 Multithreading

Since the user now is given the ability to plot multiple visualizations, it made sense to architecturally change the data handler to plural data handlers, one per unique set of data handled. Each data handler is also created in their own thread to avoid having the software artefact freeze if multiple plots are created at the same time with different data handlers. In each thread, on top of the visualization widget and before visualizing the plot, a loading bar also is implemented which shows the user how many percent of the data handlers have been loaded into memory - giving the

user an idea of how long they have to wait before the preprocessing and creating the data handler is finished, which of course is different depending on the simulation sizes.

4.3.5 Additional Functionality

Finally, with regard to the new availability of multiple plot widgets, an interactive element where the user is able to drag and drop their visualizations to match their preference is implemented. This drag and drop functionality event is also sensitive to what data is being handled and provide different options for different *data-drop events* and this is good because all data forms will not be available, nor useful, in all visualization techniques.

Accompanying the new visualizations and interactions, a plot action bar is also added. Sometimes filtering is not the correct action as the data presented is already holistic, but having the ability to: zoom, pan, export and change visualization modes etc. the user can access parts of the data without updating the visualizations themselves. Noteworthy benefit; export photo to a set of file-types for presentations.

4.3.6 Actions for final cycle

Despite getting positive feedback from both performance visualization and the logger from the first design cycle, they are not tested further in this iteration. We reason that they are promising tracks but too few users will immediately find value from it, our time, therefore, is likely better invested in creating a solid architecture from the feedback attained for the upcoming iteration.

The complete list of feedback from this iteration can be found in Appendix B. From the workshop session, the compiled set of actions chosen for the final iteration can be found in Appendix B.1 with their respective estimated priority.

4.4 Final Prototype

The final prototype's status in this stage is that the software artifact is attaining more convenient interactions and techniques for which the users can understand their data for validation. Some of the feedback from the first test has also been dealt with directly and indirectly as some of the feedback have overlapping values in their nature. So the prototype is approaching a state for which it could be considered a minimum viable product that can be tested for live use. Apart from general fluidity updates to the method, the actions chosen for the final design cycle implemented are custom visualizations and further interactions. Further interactions mainly entail more ways one can handle the data through and practically is a continuation of the data filtering which is considered as an important factor. Therefore, though, it does not have an exclusive subsection like custom visualizations have. Compiling all ideas and feedback gathered throughout all prototype iterations, the general need for the company has and likely always will be further visualizations with more patterns and

visualization types. If all those needs are to be catered to, our risk-assessment is that the research likely would be ongoing for multiple years. Thus the interface for custom visualizations elicited from the workshop in the intermediate iteration.

4.4.1 Custom visualizations

This subsection consists of two parts; *DSL Design* and *Python Interface*.

DSL Design:
A large portion of the allotted time is put into developing a dynamic way of making customized plots i.e enabling mapping between data and visualization. The specific steps of the process had to be thoroughly planned before the development of the last prototype commenced. The implementation's purpose is to find ways for which the user can define scripts that collect the data and creates plots according to specification. One implementation for this interface is exemplified below in a Domain Specific Language (DSL)[35] developed for this prototype and also tested for this iteration:

- **visualization_tag**
- **visualization_type**
- **datasets**
- **functions**
- **data-handler-ids**

In the DSL example above each row is aggregated into one line and separated by a space. The **visualization_tag** represents the name that the user wants to use to identify this custom visualization. The **visualization_type** dictates which visualization type that this data should be presented within. **datasets** is a collection of data and it ends with an index number that decides what simulations to use and the whole variable name is the same name used in the function parameters. **functions** are used to manipulate the data from mapped with the specified data. Finally, **data-handler-ids** picks the datahandler that the user wants to use i.e the data that should be accessed. A screenshot of the DSL can be found in Figure 4.4

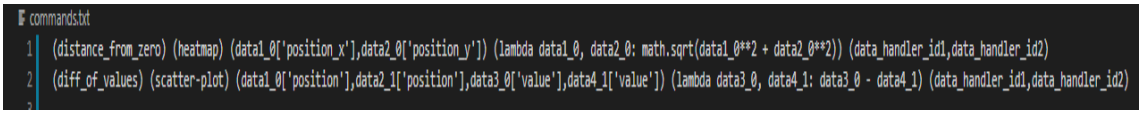


Figure 4.4: DSL format used in the final prototype. Each row represents one specific visualization of a specific datatype

This DLS practically works as a parser to handle different mapping and assumes parameters are provided on how the data should be handled for what visualizations. The software can from these parameters provided derive which data handler to use and how the function stack call chain puzzles together the correct visualization that has been specified. One could say that this DLS is a *sandbox mode* where it is possible to specify new visualizations and try them out *during runtime* with the need to restart the software artefact and reload code. Complying with feedback from earlier iterations, a drag and drop abstraction is implemented for this feature

as well, where the user can reload the text file containing all the DLS commands on request for new visualizations. This text file mentioned is our attempt to abstract every required combination through a DSL that can easily be learned with some practice. Each line in the text file serves as a DSL command and should be a complete query, at writing moment, few error handling cases are implemented and thus requires *precise input* because no error handling is implemented.

An exemplified use case could be when a decision has to be made between two different scenarios and the company may want to compare two collections to build an understanding of how the data sets are different and make a single visualization for it. A feature like this has the potential to make the system more extensible and usable if a combinatory work of any kind is needed, then it is, plainly put, enough to query the information required from which simulation, perform an action between them and visualize the result.

This DSL parser also provides a way of dealing with adaptation to format changes. Some of the responsibility will be laid in the user's control that they indeed are querying the correct visualization to their contentedness. So, given the scenario that the data storage format changes, if the user is adept enough to query what they need, even with respect to new changes, the generic interface can solve the problem as it only needs data and visualization types to perform its duty. Therefore assuming the user already is familiar with the data format before using the method [25], they can perform almost any task they are required.

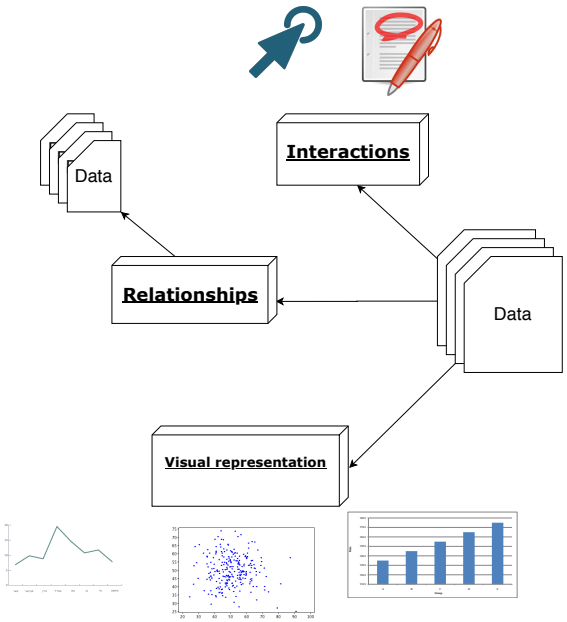


Figure 4.5: Dynamic data creation where the important mappings are shown. Depending on the KPI, it should be mapped to an interaction type, other data relations and a visual representation

Python Interface:

```
### Custom visualizations ###
CustomVisPython(tag="First visualization" , visualization_type="heatmap"          , function=func1)
CustomVisPython(tag="Second visualization", visualization_type="heatmap"          , function=func2)
CustomVisPython(tag="Third visualization" , visualization_type="advanced_scatter", function=func3)
CustomVisPython(tag="Fourth visualization", visualization_type="bar_graph"       , function=func5)
CustomVisPython(tag="Fifth visualization" , visualization_type="simple_scatter"  , function=func4)
############################
```

Figure 4.6: How custom visualizations are made in the final prototype. Same as [4.4] but now abstracted solely from Python objects and function pointers

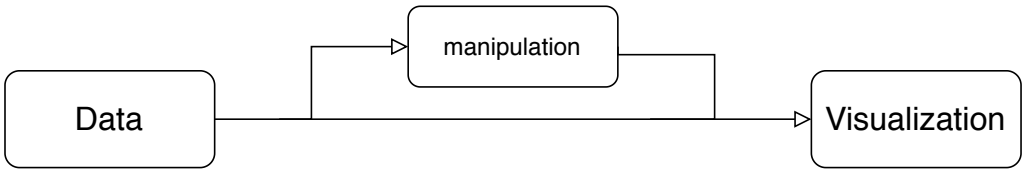


Figure 4.7: How the mapping combined with the Python Lambda manipulation works in the the final prototype

This research experimented with a generic python interface where the user could partially be responsible for creating the exact visualization they need. Therefore, also in this final prototype, an implementation where python lambdas are stored as function pointers is implemented, these function pointers can be passed as parameters to the generic visualization function. At the cost of sacrificing run-time changes to the visualization techniques, it is a choice that is likely going to be more sustainable over extended periods of time. Python itself is a very well established language with a feasible learning curve, even for beginners. Thus let's say new users having to learn *our* implemented DSL which they will only be accompanying *our* software product, they can instead learn a *Turing complete* [36] general-purpose language and create as simple or complex visualization comparisons as desired. Practically, if the user can apply or use shared pre-defined Python lambdas they can visualize anything that is mapped and manipulated data to contentedness. We don't either rule out that it is possible to add run-time lambda function pointers, just that we chose to be reasonable with our time left. The Python interface is shown in Figure 4.6

Moreover, the user has the ability to click and select multiple simulations, drag and drop the selected group, and then use their custom functions for visualizations. This interaction is necessary as it comes with the cost of not using the DSL approach where simulations are already specified. The following information had to be provided to create a mapping with manipulation, see Figures 4.5 and 4.6:

- **visualization_tag**
- **visualization_type**
- **function**

And each seperate information here have the same purpose as in the DSL.

4.4.2 Actions post final cycle

The final prototype has no continuation for this thesis. From this stage, the summarizing activity mention in Section 3.6.1 is conducted and the feedback collected from the design science methodology is given to the company. The results from the thesis are presented in the next Chapter 5.

5
Results

Below is the final result compiled from all of the design science cycles. Each cycle conducted allot a number of *feedback points* to each found factor with respective research question. A feedback point is the number of times an action from the compiled feedback is voiced by the stakeholders. Note that the second cycle is marked with priorities instead as that session was a workshop. From the workshop (Appendix B) a priority list was compiled onto actions. The factors and allotted points are compiled after the final design cycle was conducted. First, presented are all our findings concerning **RQ1**. Second, presented are all our findings concerning **RQ2**. Following the proposed results, the research objective and how the method works is presented. Finally, the compiled lists of the complete feedback can be found in Appendix A, B and C.

5.1 Factors important for data-driven requirement validation (RQ1)

The result from the study is compiled into five different factors, namely: Mapping, comparison, scalability, filtering and access time. These are the factors that are considered as most important for data-driven requirement validation. Below are the alotted feedback connected to the factors and priority orders:

Number of feedback occurrences in category (requirement validation)					
Cycle	**Mapping**	**Filtering**	**Comparison**	**Access time**	**Scalability**
Cycle 1	7	14	12	7	7
Cycle 2	HP	MP	1W & MP	2 W & MP	1 W & HP
Cycle 3	5	-	-	-	3
Total:	12 /w HP	14 /w MP	13 /w MP	9 /w MP	11 /w HP

Table 5.1: Table for found factors and respective feedback occurrences when performing data-driven requirement validation.

HP = High Priority
MP = Medium Priority
W = number of feedback occurrences from workshop
/w= "with"

From Table 5.1, the results are derived from content analysis from all data collected over the whole research when focusing on data-driven requirement validation. Each row in the table represents the result from the content analysis for a cycle. The final row shows the total compiled result, paired with the priority set from the workshop. Each factor is presented in further detail in subsequent sections.

5.1.1 Mapping

The mapping had a total of 12 feedback occurrences with a high priority and was therefore considered as an important factor.
Mapping is a factor that involves the connections made between data to visualization [25] and connection between visualizations. Below follows a list of how the users preferred to use mapping.

- **Mapping between visualizations**
  The mapping between visualizations gives the user the possibility to access datapoints from a visualization. This gives the user a better understanding of the underlying data that was used in the validation process. It supports the user in retrieving their dataset from a visualization and visualize it according to preference. The stakeholders especially found this useful when dealing with large datasets that is in need to be structured with the help of visualizations.
- **Mapping between data and visualization with manipulation**
  The mapping between the data and visualization allows the users to, in a flexible way, add new datatypes and use the manipulation to create new datatypes for additional insights used in requirement validation. This provides a way to compare different sets of data by combining their data and create new sets of data from them (e.g. the difference between two sets). Derived from the new information, validation of requirements can be performed. The mapping provides the users a lot of freedom to "play around with the data", explore it, and find new insights used for their validation processes.

5.1.2 Comparisons

The comparison amassed 12 feedback occurrences with a high priority during the workshop and is therefore considered as an important factor, refer to Table 5.1. Comparison involves the ability to compare different datasets and draw conclusions for requirement validaiton. In this research, two types of comparisons are identified.

- **Side by side comparisons**
  Through side-by-side comparisons, the user can validate data by comparing different datasets against each other. The number of plots and the size constraints of the visualizations differ depending on the scenario, therefore the size of each visualization and number of visualization is something the user finds important for requirement validation.
- **Multiple datasets in the same visualization**
  In order to compare datasets in detail, and to derive whether there are differ-

ences in the data, multiple datasets in the same plot is deemed as a solution to this problem. From the users, this is seen as a way to easily detect small differences between sets of data and from those derivations internal requirements can be validated.

5.1.3 Scalability

The scalability had 10 feedback occurrences with a high priority during the workshop and is therefore deemed as an important factor, refer to Table 5.1. Scalability is about adding new visualizations, datatypes, manipulations and mappings in order to work over several projects with different datatypes/KPI:s.

- **Add new visualizations**
  Even if a proper visualizaation can be found for a datatype by identifying patterns the users look for [26], the proper type of visualization is also individual depending on who uses the method. Familiarity plays a role in how people work and how they understand the data.
- **Add new datatypes**
  The datatypes/KPI:s could differ depending on which project the users wanted to validate their requirements within. Therefore the flexibility of adding new datatypes is important.
- **Add new manipulations**
  Manipulation of data is found to be connected to the mappings, but this also enabled the user to compare and see how datasets divert from each other (difference between sets of data). It also provides a way to create new formats of the data and make the data fit a specific type of visualization (right number of dimensions for example).
- **Add new mappings**
  This is connected to the Mapping factor dealt with in Section 5.1.1, but the ability to create new mappings in a scalable way is considered important for the users.

5.1.4 Filtering

The filtering had 14 feedback occurrences with a medium priority during the workshop and is also considered an important factor. Filtering helps the user to delimit the number of datapoints to the ones that is interesting to look at during the validation process. The two most important ways the user finds useful is by having the ability to look at a subset of data within a specific range or excluding outliers that are not interesting to consider in the validation process.

5.1.5 Access time

The Access time has 7 feedback occurrences with a medium priority during the workshop and because “time is money” it is deemed as an important factor as well, refer to Table 5.1.
The Access time is the time it takes for a visualization to render. We saw that

larger sets of data also resulted in longer time to render. This makes the validation process take longer time and just as mentioned in [30] is something that should be considered with smart algorithms. The users also wants the ability to *multitask* and be able to access multiple datasets at the same time, without having to wait for a dataset to finish loading before starting to load up the second, third, etc. This can be considered as a usability aspect that makes the user validate their requirements more efficiently.

5.2 Factors important for requirement elicitation (RQ2)

The result from the study is compiled into two different factors, namely: Adaptation and Identification. These are the factors that are considered as most important for data-driven requirement elicitation. Note that fewer design cycles are conducted on this track compared to the first research question. Below are the feedback point and priority orders:

Feedback points in category (requirement elicitation)		
Cycles	**Identification**	**Adaptation**
Cycle 1	4 /w LP	6 /w LP
Total:	4 /w LP	6 /w LP

Table 5.2: Table for found factors and respective feedback occurrences when performing data-driven requirement elicitation.

LP = Low Priority
/w = "with"

From Table 5.2, the results are derived from content analysis from all data collected over the whole research when focusing on data-driven requirement elicitation. Each row in the table represents the result from the content analysis for a cycle. The final row shows the total compiled result, paired with the priority set from the workshop. Each factor is presented in further detail in subsequent sections.

5.2.1 Adaptation

The Adaptation has 6 feedback points with a low priority. Despite the low priority, it is considered as an important factor, refer to Table 5.2. The potential for requirement elicitation is not only for developers but for other stakeholders too. The stakeholders want to have ways to display the most used functionality within the method so the knowledge is ready when it becomes a priority. The stakeholders with a more software centered background want to see in detail where to make changes and therefore have a lower abstraction level of detail, with respect to quality attributes and usage, than the other users. Therefore different "user modes" are considered as important.

5.2.2 Identification

The Identification factor has 4 feedback points with low priority. Also despite being a low priority from the workshop, it is seen as an important factor, refer to Table 5.2. For quality requirements, the main datatypes are memory usage and execution time. It is important for the developers to find the most time-consuming task and from displayed information make correct optimizations. This also applies to the usage reports, the most frequently used functionality is likely the most important one to extract from the visualization with higher quality.

5.3 Data-driven requirement engineering method (RO)

The research objective is *to develop a method for data-driven requirement validation and elicitation.* To achieve this objective, we have developed the resulting method based on the feedback attained from the research methodology's iterations. The method combines the activities in DDDM and RE and couples them to the factors by enabling modules. A **customizable gridview** enabled the users to customize how the data is displayed to preference and the grids allow for comparison, crossing the DDDM activities of data representation, analysis, visualization and decision making with requirements validation. A **slider** module implemented enables the user to manipulate the data during runtime, crossing the DDDM activities data representation and analysis with requirements validation. **Datapoint selection** allows the user to derive further information from single or multiple points and create new visualizations from the data maps on selection, this combines the DDDM activities of data representation, analysis, visualization, and decision making, and the RE activity validation. The **Python Interface** provides the users the ability to link data together and make use of mappings to derive further insights in their work, it is also a scalable interface which can extensibly add new mappings and visualizations to, this interface combines the DDDM activities of data representation, analysis, visualization and decision making with requirements validation. **Access time** is the time aspect relating to how long time it takes to perform a task, the module combines the DDDM activity of decision making with requirements validation.

The modules developed are further detailed in subsequent sections. The requirements elicitation factors are in their own subsection.

5.3.1 Customizable Gridview (Comparisions)

The gridview was important in order to do side by side comparisons between visualizations. As seen in Figure 5.1 the gridview contains several gridspots where the visualizations can be placed. The gridview also allowed size changes for each gridspot. This was good in order to make more prioritized visualizations larger or make the size appropriate for that specific plot. In order to visualize the datasets, the datasets are selected from the side-menu and dragged to the desired gridspot

(see Figure 5.4). This was important because some datatypes had visualizations that need larger screen real-estate in order to validate the values.

Figure 5.1: The red rectangle to the left represents the datasets and the grid within the right red rectangle represents the gridview

5.3.2 Slider (Filtering)

A slider was implemented to allow filtering. The slider was made with two handles that decide the lower and upper limit for the dataset (see Figure 5.2). This allowed the user to exclude unimportant datapoints and select a range of interesting datapoints that is interesting to look at in the validation process.

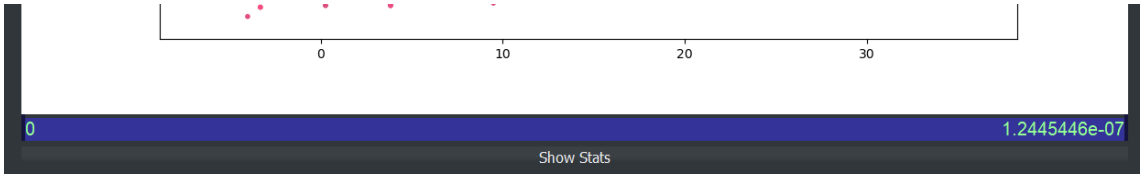
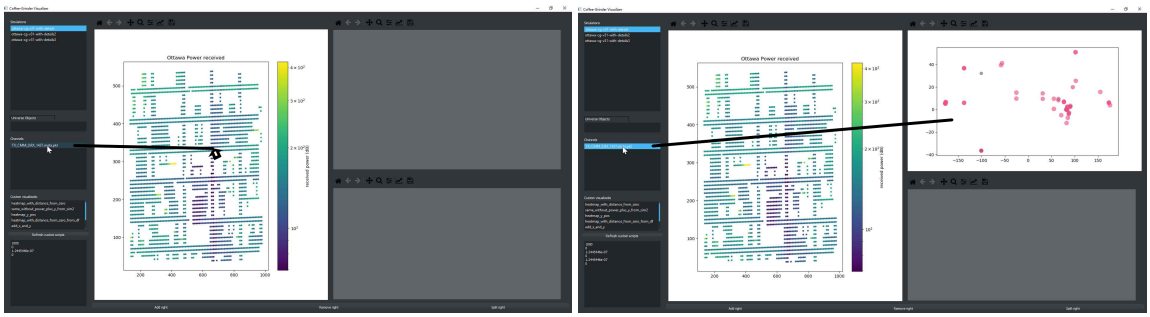


Figure 5.2: Show the slider that has a upper and lower limit for filtering

5.3.3 Datapoint selection (Mapping, filtering)

Datapoint selection handled the mapping between visualizations. This allowed the user to click a datapoint in a visualization and from there show a new visualization with a dataset connected to the clicked datapoint (see Figure 5.3). This is simply like a mapping technique that filters our visualizations from a visualization. In practical terms, this is also a way for the user to *filter* information within visualizations. If, for example, an outlier is interesting, they can select it and investigate it further in a vacuum.



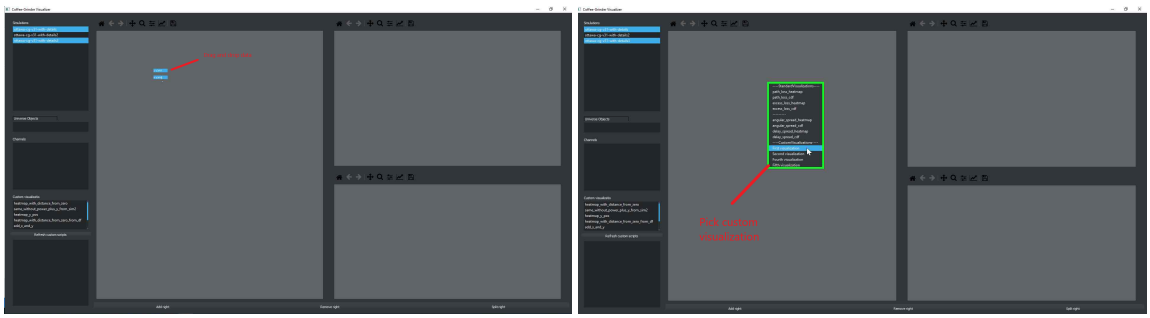
(a) Select the datapoint that is mapped to another dataset

(b) Drag that dataset to visualize the data from the mapped dataset

Figure 5.3: The steps of navigating between different datasets by visualization mapping via clicking. This helped the user get more details about the data and therefore make better validations

5.3.4 Python Interface (Mapping, Scalability)

The python interface gave the user a chance to add new datatypes, add new mappings, and add new manipulations. Apart from this the user also had to select a visualization from a subset of visualizations. This was made by creating python objects (see Figure 5.5) with certain parameters. The parameters contains a tagname, a visualization type and a function-pointer. The function/manipulation also needed to be specified by the user (see Figure 5.6). This let the user easily do mapping and add new datatypes and manipulations. The manipulation also enabled the user to combine several datatypes (by dragging several datasets) and visualize them in the same visualization (see Figure 5.4).



(a) Select the datasets that will be visualized. In this case 2 visualizations are selected and dropped on a gridspot within the grid

(b) When the datasets are dropped, only mappings (created in python interface) with manipulations with that exact number of parameters will be showed.

Figure 5.4: This shows how the python objects are presented in the user interface. The user could select the wanted datatypes and connect them to a custom visualization from the python interface

```
### Custom visualizations ###
CustomVisPython(tag="First visualization" , visualization_type="heatmap"          , function=func1)
CustomVisPython(tag="Second visualization", visualization_type="heatmap"          , function=func2)
CustomVisPython(tag="Third visualization" , visualization_type="advanced_scatter", function=func3)
CustomVisPython(tag="Fourth visualization", visualization_type="bar_graph"        , function=func5)
CustomVisPython(tag="Fifth visualization" , visualization_type="simple_scatter"   , function=func4)
############################
```

Figure 5.5: Shows the python interface and how a new mapping is created with a manipulation/function. The dataset used for these mappings are selected in the GUI

```
### Custom functions ###
def func1(dataset1, dataset2):
    data1 = dataset1['index']
    data2 = dataset2['index']
    return [data1, data2, abs(data1 - data2)]
```

Figure 5.6: Shows one defined manipulation as a function. This specific function takes two datasets (selected in the gui), picks out the interesting datapoints and then returns the axis (in this case x, y, z with z as the difference between the two datasets)

5.3.5 Multithreading(Access Time)

Multithreading let the user load multiple plots at the same time (see Figure 5.7). This was done by letting the loading of each dataset be done on separate threads. This both allowed several visualizations to load at the same time and also led to no freezing of the software artefact during loadtime. This led to a more time efficient validation process that gave a quicker access to the visualizations.

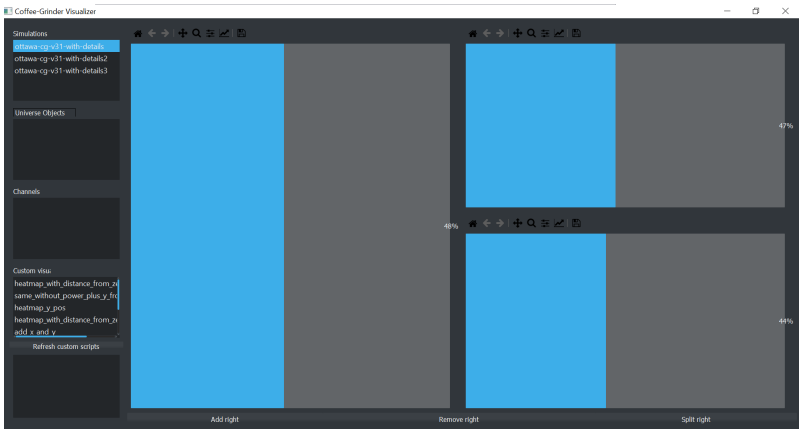


Figure 5.7: One dataset loads on one separate thread for each gridspot and therefore makes the validation process more time efficient

5.3.6 Data-driven requirement elicitation

All of the factors for software requirement elicitation are not covered in the prototype as the priority was toned down, but here is what was developed after the first cycle:

- **Data collector from Dask API (Identification)**
- **Sorted box-plot visualization of performance (Identification)**
- **Usage visualization (Identification)**

Below follows some screenshots of how the elicitation ended up in the prototype:

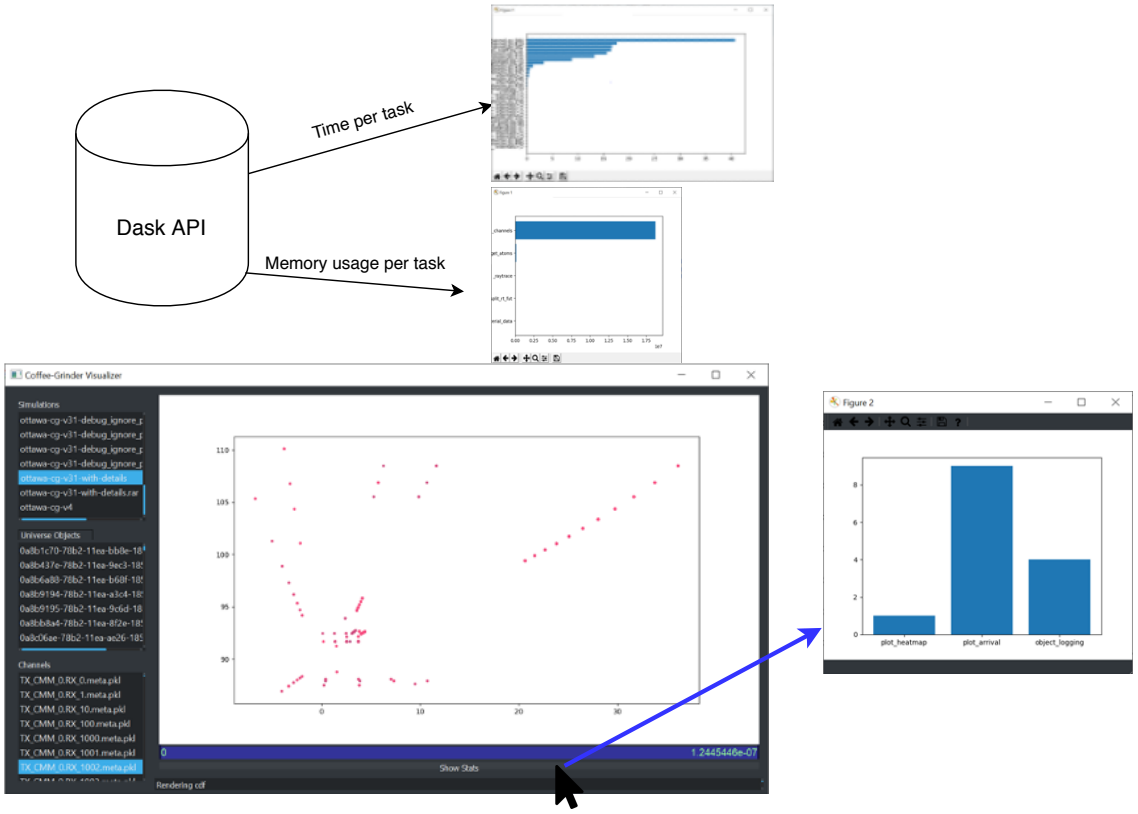


Figure 5.8: The software elicitation implementation of the prototype. The usage report was accessible in the software artefact and the memory usage/execution time was gathered from the DASK API during runtime

From Figure **??** it is shown how the identification is done for eliciting requirements. When features are used, the usage information both concerning runtime tasks and frequency of functional use is logged. The information is available for display to elicit requirements for possible optimizations or improvements.

5.4 Evaluation of DDRE method

The method is seen keenly by the users. They saw a big potential in the method and thought it would be a good method to use across several projects. The method is at an early stage of development and should have some more development time to be fully ready for production. The method was mainly developed to support requirement elicitation/validation and here is what we got from each aspect:

5.4.1 Requirement validation

First Prototype: The first prototype focused on visualizing the plots with the KPI:s found in the pre-study. The user found the plots to be sufficient in terms of their preference and how particular KPI:s should be visualized efficiently. This lets the users have a fixed connection between the given KPI:s and the visualization.

The feedback was that the method should perform more plots and be able to present the data in more diverse ways. Still, the prototype gave them a method that can be used to easily access the data, visualize it, and validate the KPI:s. The elicitation was done by having a usage logger and a performance visualizer and both of those aspects were seen as good implementations in the future but were not prioritized in the upcoming prototype.

Intermediate Prototype: The intermediate prototype contained drag and drop, where the user could drop the data on a given grid-spot and choose from a set of KPI:s. This was perceived as a good feature from the users, it enables the user to produce side-by-side comparisons and prioritize more important visualizations. This provides the user with an interface to compare different KPI:s and from there validate if there was an improvement or not. This also let them look at several KPI:s at the same time, which led to a more efficient validation process. Moreover, the users also said that it could potentially be a problem if the datasets are too large because the preprocessing time of the data would be too long in order to validate the KPI:s efficiently. The filtering technique with the slider provided a way for the users to exclude data that should not be a basis for requirement validation and more ways of accomplishing filtering was proposed. Finally, the multithreading was seen as a way to do several tasks at the same time while loading the visualizations and therefore validate the KPI:s more efficiently.

Final Prototype During the last prototype, two types of ways to accomplish scalability which included a DSL language and the Python Interface. During the interviews, the common opinion was that the Python interface was better because the users are familiar with Python and thought that it was a great way to be able to add new mappings between data and visualization manually. To allow the users to easily accomplish this, we added a way to create Python objects that took parameters to specify the mapping. The definition of the mapping provided a way to find a visualization that is suitable for that specific KPI and from there get the insights needed for validation. The parameters dictated the abstraction level of how much the user has to inform the system in order to make a mapping. Most of the interviewed users agreed that three parameters consisting of tagname, plottype, and manipulation was a good abstraction level to go for. The users also asked for making the mapping automatic, which was a bit too ambitious for this software artefact at the moment.

All in all, the software artefact was seen as a good method to validate the requirements for the users. Users saw the potential, especially when the requirements are quantifiable (as it is in this case). It also allows the user to see if there are any improvements when testing a scenario and validating it against the internal requirements. The users also appreciated the scalability of the software in order to use it across projects and be prepared if new KPI:s have to be added to the validation process. Even if the method covered many of the important aspects, it still lacked some important features like textual representation of the data and multiple datasets in the same graph. The textual representation was to some degree implemented, but not enough to be usable. According to the users, textual representation was impor-

tant for values like sample mean, median etc. Basically numbers that can represent a dataset collectively and from there validating the datasets. Apart from that the user also wanted to have the ability to implement new visualizations which was not implemented in this method, but rather used a subset that the user can choose from.

5.4.2 Requirement elicitation

First Prototype: We saw at an early stage that requirement elicitation is not prioritized in this research. Even if this is not prioritized in this research, some important factors are still identified. These factors are unfortunately not tested enough to get feedback about it. Even if the users are somewhat critical of the elicitation approach, we still provided an API that can be used and prioritized in the future. The main focus of this prototype is to enable identification for eliciting requirements. The identification of the performance attributes and the usage logger are not tested with the adaptation between the users and therefore difficult to get an opinion from the users regarding different visualizations for different users. The identification of different tasks memory usage and time efficiency was monitored to the user and was seen as too complex to understand. Derived from this, we got the adaptation factor. The usage logger is something that the users did not find value in because of the small userbase, which means that the usage data collection will not be large enough to draw conclusions from it.

6
Discussion

The discussion entails our thoughts around the project from different angles. First, we discuss the implications and our thoughts on the results from a requirement validation standpoint. Second, we discuss the implications and our thoughts on the results from a requirement elicitation standpoint. And finally, a section sharing our thoughts on the method developed and its result.

6.1 Data-Driven Requirement Validation (RQ1)

From the results, we conclude that a set of factors are important to consider for requirement validation. **Data mapping** is important as it is a way for which the employees have a shorter lead-time for making decisions regarding requirement validations. If correct data mappings are pre-defined then they will not have to make the connections themselves and can begin their analysis immediately. A correct mapping is also crucial in order to have appropriate visualizations for specific KPI:s. This can both be something the team agrees upon or is individual for each specific user. Data mapping between data and visualizations was also that was considered as a challenge in visualization design [25]. The main challenge is to know which data/KPI:s to connect to a certain visualization. This is mainly important for requirement validation because new requirements can be added with corresponding KPI:s and then this data has to be visualized in an optimal way in order to accomplish efficient requirement validation. More research should be done within this area in order to create frameworks that can support the user to add new KPI:s that can automatically be mapped to a visualization. This is something that also is related to the **Scalability** factor that is one of the most significant factors found with respect to the future. The ability to scale the application implicitly means as requirements change, and they will, the need for adding new KPI:s to visualize is crucial. If the system can not add new KPI:s, then it will not be possible to visualize it if a new requirement is added for validation. Scalability also added value in the way that the method can be used across several projects because each project has their own set of KPI:s to validate. The reason why this is seen in particular as an important factor for requirement validation is due to the need to add new KPI:s may be fluctuating, requirements can quickly change. **Comparisons** could be viewed as a form of benchmark between different scenarios. The ability to make decisions from the data side-by-side basically provides a way for visualizing the two or more decision points and pitch them against each other in a cohesive way for validating performance indicators. This also makes the validation process efficient because

the user can look at several KPI:s at the same time and come to quicker decisions. **Filtering** is the means in which the user has control over the data. Depending on their requirement they are working against, it may not be relevant to include the whole dataset and, for example, only the 5% most prominent data is interesting for the requirement validation. Having this control, the user is able to set the bounds for which data is presented and can from one visualization and change the filters to explore the information interactively. Some data may not be interesting to consider in the validation and therefore it is an important feature in order to limit the data within the boundaries that will be valuable for requirement validation. **Access time** is the time aspect from where an employee knows what needs to be done and it is accessible for visualization. So from a requirements perspective, this means how long time does it take to validate performance indicators.

One could question whether these factors are particularly important for validation, or important in any general visualization system. Comparison may be more important in requirements validation for this case because different KPI:s need to be compared against each other, for example, to find the best algorithm. Comparing several KPI:s may not be something that is important in other settings where the user only validates single scenarios, but that really depends on what one is trying to achieve. Apart from that, Comparison may be a factor that is important in many other ways than only validation, because the visualizations are commonly used as a measurement of performance in other fields as well. Access time is probably a factor that is important for most visualization systems, because of the ability to quickly produce a visualization from the data. Mapping is also probably a factor that is important in general for a visualization software in order to know what data should be visualized and how. Scalability and filtering are probably more important for validation. The reason for this is that requirement validation can be seen as a process where the user quickly wants to see if a specific requirement is fulfilled and meets the customer needs. The needs can quickly change and therefore the user that does the validation has to adapt and add new KPI:s to investigate. These KPI:s reflects the need from the customer, therefore Scalability is likely more important in requirement validation, but again; it all depends on what one is trying to achieve. Filtering is extremely important for requirement validation, but depending on what a general visualization method should do it could also be very important to have filtering functionality. Coupled to validation though, we know that performing some validations is not possible without having filtering techniques in place, while in a general case of just needing to visualizing something one may not need to filter anything. So once again, it all is depending on what one is trying to achieve.

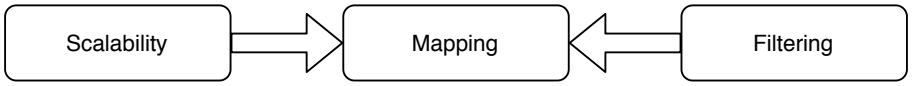


Figure 6.1: Both Scalability and fitering could provide new mapping functionality

Apart from this, it is also interesting to look into how the different factors relate

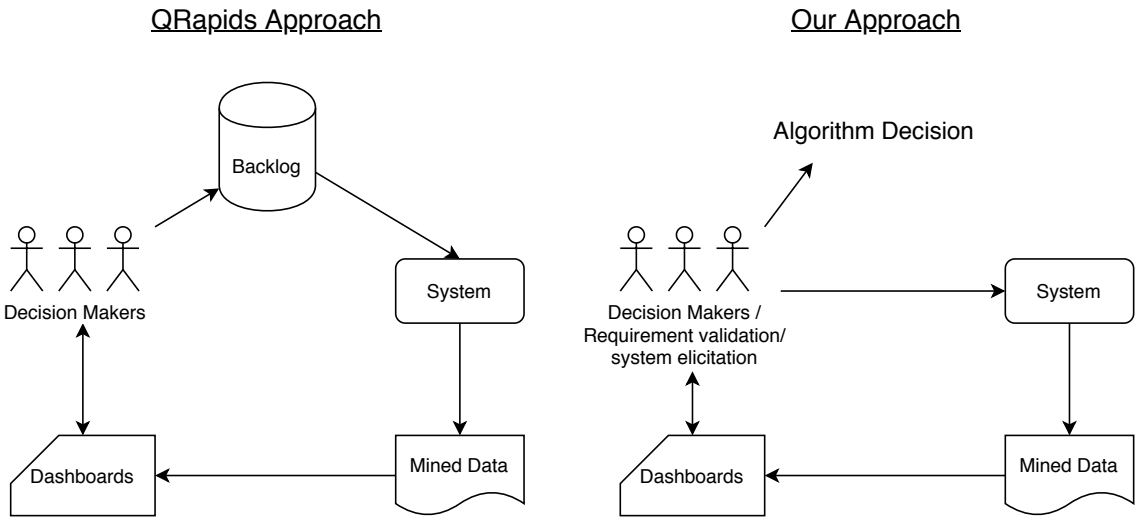


Figure 6.2: The QRapids approach vs our approach. Where the difference is what happens afer the decision making process

to each other. Figure 6.1 shows how the different validation visualization factors related to each other. They are connected in the way that a scaleable system can create new mappings and filtering techniques can also provide mapping functionality. The filtering mainly provided a way to filter by clicking at datapoints that contained datasets, which is a mapping between visualizations.

6.2 Data-Driven Requirement Elicitation (RQ2)

From the results we conclude that a set of factors are important to consider for requirement elicitation. **Adaptation** is identified as important because every stakeholder is different. Depending on what background or skillsets the stakeholders have, they have a different need to get out of the software. For some, the ability to just output something is most important for their work, and to others being able to make implicit deductions out of the method is most important. Thus, a need for different user modes could be sought after. **Identification** is also seen as important. The case study points towards that there could be potential value in gathering identifying data for future requirements, though the need explicitly is not there today.

During this research, the Q-Rapids model [22] was used as inspiration for the DDRE method. In the Q-Rapids model identification is important in order to eliciting requirements. This is probably because quality requirements are based on runtime data from the system that then can be measured that can be used to identify flaws. As far as we know, no research has been done by eliciting requirements by having adaptation i.e. different presentation layers between the users and the developer. This is something that can potentially work by letting the users understand the flaw

in a system and from there, communicate this to the developer, that then can use this information to elicit new requirements. Due to this, a new problem will arise and that is the communication between the user and the developer. This communication may be too time-consuming and difficult to maintain over time in order to get an effective requirement elicitation process.

These factors are just as the validation factors, probably to some degree important for any visualization software artefact. Though, as current research suggests [22], the attempts are about identifying flaws in a system in order to elicit new requirements. Identification can also be seen as a kind of identification of outliers, which is an important pattern to look for in visualizations [26], which is not only used in requirement elicitation. The Adaptation factor may be more unique because it is about having different visualizations depending on the user. As far as we know, there is no visualization tools that handles different abstraction levels in that way.

6.3 DDRE Method (RO)

The DDRE method includes several artefacts and each artefact supports the factors found. The method is seen as a good way to validate requirements and the software artifacts supported the factors found important for requirement validation. The elicitation part of the method was made by having performance measures taken from the DASK API [47] and having user logs to see what functionality is used the most. Both of these implementations fulfill the factor identification, but the adaptation factor was unfortunately not implemented due to time constraints. The method did not adapt to format changes either. In terms of the scalability factor, the indexing of the data needed to be documented in order to collect the right data. This is something that we see as a potential improvement in the future to have automatic adaption to format changes, or that a data storage layer is able to process data for usage to conform to data formats desired.

To the best of our knowledge, there is no software tools to validate internal requirements. Studies have been made looking into existing visualization software or tools [45] and these visualizations are commonly used for software development. Many of these visualization tools are used for software requirement elicitation by identification of software flaws. But as far as we know, no method has been developed that extends this idea and uses data for other purposes then software system requirements. The reason may be that it can be difficult to make some requirements quantifiable and therefore cant be associated with a KPI. If the requirements do not have KPI:s to measure the requirement, then it will probably be more difficult to visualize and validate. Therefore, more research should be done in visualization techniques for validating non-quantifiable requirements.

The design of the method was inspired by the Q-Rapids approach [22] and the difference between the DDRE method and Q-Rapids is shown in Figure 6.2. The validation leads to an algorithm decision and the elicitation leads back to the system for system improvements. By doing this, the method could both validate and elicit

requirements. The commonalities between our approach and Q-Rapids is that the decisions are based on visualizations. The type of visualization is therefore important to consider in order to make the right requirement decisions based on them. Visualization frameworks can be a potential way to get the most out of visualizations. In our research we mainly used inspiration from visualization design [25] and Quality metrics [26] in order to accomplish this. More research should be attempted to automatically apply visualization frameworks for certain KPI:s/data that can find the optimal visualization and improve the insights for requirement validation and elicitation.

Research has also been done addressing DSL approaches linked to DDRE [49]. In this research Franch et. al. addresses the potential research in the future for DDRE and one of them included a DSL model to treat requirements as functions. Functions that can be linked to the system and mine data during runtime and automatically validate the requirements. Comparing this to our approach shows that both look at the values of the quantifiable requirements, but instead of automatically validating the requirements in our system, we let the analyzer validate based on the visualized data. Even if the Python interface and Franch et al. proposed DSL is not exactly the same model, the Python interface is still a starting foundation for creating DSL models for supporting DDRE. This can provide value by creating methods catering to requirement validation, but still be customized depending on the requirement validation scenario.

6.4 Threats to Validity

The following subsections detail plausible threats to validity of this study. This discussion uses definitions for validity threats taken from Wohlin et al.[51].

6.4.1 Conclusion Validity

Conclusion validity is a threat with respect to how well one is able to interpret the results and draw the correct conclusion.

To minimize this threat, each participant in the study were subjected to the same demonstration of the software. Though, since the interviews were semi-structured, opinions are of course voiced at different times of the demonstration so further explanations and answers from us are provided when asked for. This means that the *exact same* interview was not conducted every time, even if they were planned to be the same. We do not think that allowing curiosity hurt the conclusion, rather it provided more well-rounded feedback from the participants.

Further, the sample size of the case study is too small to claim any generalizability, though that is not the scope of the thesis. Best efforts were made to bring as many participants available as possible to do a good job as possible.

6.4.2 Internal Validity

Internal validity addresses the causal relationship between the factors within the study.

The first validity threat to our result is that we conducted a workshop. From the feedback of the workshop, and comparing to the first prototype interviews, the mass just seems lackluster in comparison. This could perhaps be due to when people start voicing their opinions in a group, some part of the group becomes compliant with the opinion and perhaps do not want to contradict popular opinions. So there is a risk that some of the answers in this session are influenced by others. Since we expected a wide array of opinions from the workshop (like the first prototype), we did not prepare ways to encourage more outlying opinions either. So it is plausible that a group of opinions that could lead to more factors or other priority orders is maybe missing from that session.

Another threat to internal validity is that the selection of participants is not random. Though, we deem that since the expertise and backgrounds of the participants are largely different, differing opinions have been voiced and thus should not be a biased answer. The selection of participants is also done based on their availability, and their expertise, therefore different number of interviews were conducted in each step.

6.4.3 Construct Validity

Construct validity is concerned with how the operational measure reflects the purpose of the study. This study does not intend to prove generalizability of the results but does intend to analyze target stakeholder's preferences concerning the developed prototype.

The participants in the interviews were subject to demonstrations of the method where they were asked questions around our research questions and allowed to voice whatever came to mind during the time. To avoid the construct that they *only* answer questions that we've prepared and allow a limited scope of possible answers for factors, we believe that the freedom that comes with these semi-structured interviews allow the interviewee to feel that their opinion is heard and therefore a more truthful and unfiltered answer should result from the feedback. However, in order to maintain some order of scope for the breadth of answers, more specific questions are asked if they did not voice their opinions with respect to the investigation.
Finally, we don't either believe mono-operational bias is an issue for this study as all the participants have a wide array of different backgrounds, different sets of working, and projects that they are involved in.

With a larger scope of answers possible, the job of quantifying the feedback becomes a validity threat. We are ultimately responsible to make sure that their answers are understood correctly. To contest this threat, during the interviews when distinct opinions were voiced, confirming questions were asked in our own words to confirm

the understanding. Sometimes our understanding had to be corrected, but to the best of our ability we've confirmed that the opinions are current.

Last, we'd like to consider the second design cycle being compiled together with the other cycles. It could be seen as comparing apples to pears since the data gathering format was different. Though to the best of our ability, we attempted to elicit the feedback as openly as possible but this is of course a threat to validity, people do indeed act differently in groups and we cannot guarantee that they either are as comfortable to voice their opinions.

6.4.4 External Validity

Threats to external validity refer to the risk of generalizing the results outside of the scope of the study.

The most obvious external threat with this study is that we are doing the research for a company in a niche market of telecommunication. Every company is not the same and the results have therefore a risk of differing depending on where the research is conducted. To avoid this threat we tried to only implement modules in the method that can be used in a generic way. Regardless, the aim of the study is not to prove theories, but rather to obtain an understanding of the important factors at play.

6.4.4.1 Historical Validity

Historical validity refers to the events taking place in the world at the the time of the research.

This thesis was conducted during the pandemic outbreak of Covid-19. Halfway through the thesis, the data gathering was done with remote tools and we cannot say if the less intimate nature of video conferencing affects the results. Also, we cannot either tell if the change in work environment or stress from the historic event have any impact on the results.

7
Conclusion

Becoming data-driven is an industry trend, i.e., adopting data-driven activities onto other disciplines to make better decisions concerning the discipline. Research in making requirements engineering data-driven has been conducted, though most seem focused on eliciting requirements back to a software product. This study performed a case study using design science in industry. The data collected was conducted through 17 interviews and a workshop, the data is categorized through content analysis. Further, the study focuses on the requirement engineering activities of validation and elicitation, and what factors are important for making these activities data-driven. The result is produced utilizing a method that is used for data-driven requirements engineering.

The study resulted in five important factors to consider when performing data-driven requirement validation and two factors when performing data-driven requirement elicitation. The implications concerning data-driven requirements validation are that there is a need for interactivity and flexibility when handling data, to be able to from many angles analyze the data to understand it completely. The implications with respect to requirements elicitation are that the need for identifying data and have data adapt to the user is sought after but perhaps will not be used as immediate decision points.

The method developed is based on the input received in iterations from industry over the research scope. From evaluating the method with regard to validation and elicitation; we find that having specific techniques for data-driven requirements engineering is perceived as advantageous. Generally, the method was easy to use and understand, and high hopes for usefulness were reflected on the scalability of the functionality.

7.1 Future Research

The research made in this thesis will provide an insight into which factors that are important to consider to gain insights into data during visualization and make the decision based on the data. Even if this research developed a DDRE method based on the insights given, more research should be done on the different aspects of data visualization and what kind of interactions, graphical visualizations and relationships that can be added to make a more generalized concept. More research should also be made into the visualizations withing data-driven decision making to get

support on which visualization to use for a certain KPI to maximize requirement engineering efficiency and fulfil customer needs. Research should also be done in looking in which circumstances KPI:s can be used to validate requirements. The telecommunication industry has already established this in their field, but it would be interesting to see if it can extend to other areas.

The dynamic behaviour is also something that is an important aspect to make a product that can adjust if the format of the data changes and adds new data types which can easily be mapped to an interaction, visualization and relationships with other data types. This is a probable adding on the prototype and would be a good extension of this research.

A continuation of this thesis could perhaps also be trying to investigate the remaining activities from both DDDM and RE to derive more important factors when either combinatorically selecting a few or holistically testing all at once.

Bibliography

[1] Mandinach, Ellen B and Gummer, Edith S and Muller (2011) "The complexities of integrating data-driven decision making into professional preparation in schools of education: It's harder than you think", Robert D. Report from an invitational meeting. Alexandria, VA: CNA Analysis & Solutions

[2] Barone D., Jiang L., Amyot D., Mylopoulos J. (2011) Reasoning with Key Performance Indicators. In: Johannesson P., Krogstie J., Opdahl A.L. (eds) The Practice of Enterprise Modeling. PoEM 2011. Lecture Notes in Business Information Processing, vol 92. Springer, Berlin, Heidelberg

[3] Negash S., Gray P. (2008) Business Intelligence. In: Handbook on Decision Support Systems 2. International Handbooks Information System. Springer, Berlin, Heidelberg

[4] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang and W. Xiang, "Big data-driven optimization for mobile networks toward 5G," in IEEE Network, vol. 30, no. 1, pp. 44-51, January-February 2016, doi: 10.1109/MNET.2016.7389830.

[5] Provost F, Fawcett T. "Data Science and its Relationship to Big Data and Data-Driven Decision Making". Big Data. 2013;1(1):51-59. doi:10.1089/big.2013.1508

[6] Scott J.Lusher Ross McGuire René C. van Schaik C. David Nicholson Jacobde V. lieg. "Data-driven medicinal chemistry in the era of big data".

[7] Hedgebeth, D. (2007), "Data-driven decision making for the enterprise: an overview of business intelligence applications", VINE, Vol. 37 No. 4, pp. 414-420. https://doi.org/10.1108

[8] Marsh, Julie A., John F. Pane, and Laura S. Hamilton, Making Sense of Data-Driven Decision Making in Education: Evidence from Recent RAND Research. Santa Monica, CA: RAND Corporation, 2006. https://www.rand.org/pubs/occasional_papers/OP170.html.

[9] Solomon Negash, Paul Gray "Business Intelligence, Handbook on Decision Support Systems 2", 2008 ISBN : 978-3-540-48715-9

[10] Tardío R., Peral J. (2015) Obtaining Key Performance Indicators by Using Data Mining Techniques. In: Jeusfeld M., Karlapalem K. (eds) Advances in Conceptual Modeling. ER 2015. Lecture Notes in Computer Science, vol 9382. Springer, Cham

[11] Meinrenken, Christoph J., et al. "Combining Life Cycle Assessment with Data Science to Inform Portfolio-Level Value-Chain Engineering: A Case Study at PepsiCo Inc." Journal of Industrial Ecology 18.5 (2014): 641-651.

[12] Matthew A. Waller, Stanley E. Fawcett "Data Science, Predictive Analytics, and Big Data: A Revolution That Will Transform Supply Chain Design and Management"

[13] van der Aalst W. (2016) Data Science in Action. In: Process Mining. Springer, Berlin, Heidelberg

[14] G. Deshpande, C. Arora and G. Ruhe, "Data-Driven Elicitation and Optimization of Dependencies between Requirements," 2019 IEEE 27th International Requirements Engineering Conference (RE), Jeju Island, Korea (South), 2019, pp. 416-421, doi: 10.1109/RE.2019.00055.

[15] Bauer, Kent. "KPIs-The metrics that drive performance management." Information Management 14.9 (2004): 63.

[16] Ian S. "SOFTWARE ENGINEERING" Ninth Edition Chapter 4.

[17] Vaismoradi, Mojtaba and Turunen, Hannele and Bondas, Terese, Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study, 2013, pp. 398-405, https://onlinelibrary.wiley.com/doi/abs/10.1111/nhs.12048

[18] Vaishnavi, Vijay Kuechler, B. (2004). Design Science Research in Information Systems. Association for Information Systems.

[19] Hevner, A., March, S., Park, J., Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75-105. doi:10.2307/25148625

[20] W. Maalej, M. Nayebi, T. Johann and G. Ruhe, "Toward Data-Driven Requirements Engineering," in IEEE Software, vol. 33, no. 1, pp. 48-54, Jan.-Feb. 2016, doi: 10.1109/MS.2015.153.

[21] W. Maalej, M. Nayebi and G. Ruhe "Data-Driven Requirements Engineering - an Update"

[22] Xavier Franch1, Cristina Gomez1, Andreas Jedlitschk2, Lidia Lopez, Silverio Martin Nanezz, Marc Oriol1, Jari Partanen "Data-Driven Elicitation, Assessment and Documentation of Quality Requirements in Agile Software Development"

[23] Anthony Kelly "Decision Making Using Game Theory: An Introduction for Managers Cambridge University Press, 2003.

[24] Zuoxu Wang, Chun-Hsien Chen, Pai Zheng, Xinyu Li, Li Pheng Khoo "A novel data-driven graph-based requirement elicita-tion framework in the smart product-service system context"

[25] Jagoda Walny, Christian Frisson, Mieka West, Doris Kosminsky, Søren Knudsen, Sheelagh Carpendale, Wesley Willett "Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff"

[26] Behrisch, M.; Blumenschein, M.; Kim, N. W.; Shao, L.; El-Assady, M.; Fuchs, J.; Seebacher, D.; Diehl, A.; Brandes, U.; Pfister, H.; Schreck, T.; Weiskopf, D.; Keim, D. "Quality Metrics of Information Visualization"

[27] Richard Berntsson, Svensson Robert Feldt, Richard Torkar,
The Unfulfilled Potential of Data-Driven Decision Making in Agile Software Development

[28] ISO ISO/IEC 25010 "https://iso25000.com/index.php/en/iso-25000-standards/iso-25010"

[29] Siiri Fuchs; Mari Elisa Kuusniemi, Making a research project understandable - Guide for data documentation "http://doi.org/10.5281/zenodo.1914401"
[30] C.L. Philip Chen, Chun-Yang Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on Big Data, pp 314 - 347 http://www.sciencedirect.com/science/article/pii/S0020025514000346
[31] Datid Fetterman, "Collaborative, Participatory, and Empowerment Evaluation", davidfetterman.com
[32] Tarek Azzam, Stephanie Evergreen, "Data Visualization, Part 1 : New Directions for Evaluation", Number 139, chapter 1, pp 7-32
[33] Sourour Maalem ,Nacereddine Zarour, "Challenge of validation in requirements engineering", 2016, pp 15 - 21, http://www.sciencedirect.com/science/article/pii/S2352664516300025
[34] O'Regan, Gerard, "Concise Guide to Software Engineering: From Fundamentals to Application Methods", 2017, pp 47–60, https://doi.org/10.1007/978-3-319-57750-0_3
[35] Domain Specific Language https://en.wikipedia.org/wiki/Domain-specific_language
[36] Turing Completeness https://en.wikipedia.org/wiki/Turing_completeness
[37] Erik Brynjolfsson, Lorin M. Hitt and Heekyung Hellen Kim, "Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance?", 2011.

[38] Runeson, P., Höst, M. Guidelines for conducting and reporting case study research in software engineering. Empir Software Eng 14, 131 (2009). https://doi.org/10.1007/s10664-008-9102-8
[39] Data documentation https://ebookcentral.proquest.com/lib/chalmers/detail.action?docID=3317671

[40] Mandinach, Ellen and Honey, Margaret and Light, Daniel, 2006/01 A Theoretical Framework for Data-Driven Decision Making
[41] Gai Li and Qiang Chen, "Exploiting Explicit and Implicit Feedback for Personalized Ranking", https://www.hindawi.com/journals/mpe/2016/2535329/
[42] SM Wundenberg, "Requirement Engineering for Knowledge-Intensive Processes", 2015 https://link-springer-com.proxy.lib.chalmers.se/book/10.1007%2F978-3-658-08832-3
[43] Charles M. Rudd, Gary H. McClealand, Carey S. Ryan, "Data Analysis, A Model Comparison Approach" second edition
[44] Fowler, M. and Foemmel, M., "Continuous integration", 2006.
[45] Leonel Merino ; Ekaterina Kozlova ; Oscar Nierstrasz ; Daniel Weiskopf, "VISON: An Ontology-Based Approach for Software Visualization Tool Discoverability", https://arxiv.org/pdf/1908.04090.pdf
[46] M. Nayebi, "Data Driven Requirements Engineering: Implications for the Community," 2018 IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, 2018, pp. 439-441, doi: 10.1109/RE.2018.00058.

[47] Dask: Scalable Analytics in Python https://dask.org/
[48] Janssen, M., van der Voort, H., Wahyudi, A.: "Factors influencing big data decisionmaking quality". Journal of Business Research, 70, pp. 338–345 (2017).

[49] Franch, Xavier & Seyff, Norbert & Oriol, Marc & Fricker, Samuel & Groher, Iris & Vierhauser, Michael & Wimmer, Manuel. (2020). Towards Integrating Data-Driven Requirements Engineering into the Software Development Process: A Vision Paper. 10.1007/978-3-030-44429-7_10.
[50] Doukoglou, Tilemachos and Gezerlis, Velissarios and Trichias, Konstantinos and Kostopoulos, Nikos and Vrakas, Nikos and Bougioukos, Marios and Legouable, Rodolphe, Vertical Industries Requirements Analysis Targeted KPIs for Advanced 5G Trials, http://dx.doi.org/10.1109/EuCNC.2019.8801959 2019 European Conference on Networks and Communications (EuCNC), 2019
[51] Experimentation in Software Engineering Claes WohlinPer Runeson-Martin HöstMagnus C. OhlssonBjörn RegnellAnders Wesslén https://doi-org.proxy.lib.chalmers.se/10.1007/978-3-642-29044-2
[52] Model–view–controller https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93contr

[53] ReactJS https://reactjs.org/
[54] S. Fickas and M. S. Feather, "Requirements monitoring in dynamic environments," Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95), York, UK, 1995, pp. 140-147, doi: 10.1109/ISRE.1995.512555.

A

Appendix Design Iteration 1

Each point describes the a certain feedback from the user, followed by number of users addressing that feedback and finally the factor category:

1. Correlations between simulations visualization e.g. cost per performance. (1/7) -> QR-vis Identification
2. Set upper and lower data bounds and add more filtering interactions (5/7) -> Filtering
3. Fix errors in logarithmic scale for visualizations. (3/7) -> Other
4. Specific fix suggestions for visualizations, e.g. Inverted X-Y axes, building outlines, etc. (2/7) -> Other
5. Removal of uninteresting points by default in CDF-plot but with option to show them. And other good defaults. (1/7) -> Filtering
6. Logger with basic information is good as it is, could perhaps work on the presentation format for clarity. (4/7) -> Usage reports (Adaprion)
7. Added crash reports are important, though might be hard to collect them between users for the scope of this project due to administration and some services amd resources used are only available in Chinese. (1/7) -> Usage reports (Adaption)
8. Automatic usage reports and function memory usage may be good for making decisions on what features to optimize, and plain knowledge on what operations take time. (3/7) -> Performance visualizaion Identification
9. Add more ways to plot data. (7/7) -> Scalability
10. Trend spotting on performance between releases. Perhaps make use of continuous integration and continuous development tools in version control system. (1/7) -> QR-vis Identification
11. Request to be able to plot specific algorithm schedulers. (1/7) -> Mapping
12. Cross simulation visualizations, co-joining simulation results. (6/7) -> Comparison
13. Visualization comparisons, pitch one plot against another. (6/7) -> Comparison
14. File format change request for handling large storage better. (2/7) -> Access time
15. Request to attempt combining multiple stored files into larger ones. (2/7) -> Access time
16. More techniques for filtering data in visualizations requested (7/7) -> Filtering
17. Suggestions for data grouping, topologies and key performance indicators. (4/7) -> Data structuring
18. Suggested textual search function. (1/7) -> Data filtering

19. Export functions for visualizations. (2/7) -> Exporting
20. Multithread support to avoid application freezing. (3/7) -> Access time
21. Have between plot mappings with interactions (6/7) -> Mapping

Top categories & feedback points

1. Filtering (14)
2. Comparison (12)
3. Mapping (7)
4. Access time (7)
5. Flexibility (7)

A.1 Actions for next iteration

This appendix is a compiled list of actions that are chosen to be implemented for the next iteration of the project. It is based on the items from Appendix A and each item selected is categorized by us with value and difficulty.

- 2. Set upper and lower data bounds and add more filtering interactions (5/7) -> Filtering (High value / High difficulty).
- 3. Fix errors in logarithmic scale for visualizations. (3/7) -> Other (Medium value / Low difficulty).
- 4. Specific fix suggestions for visualizations, e.g. Inverted X-Y axes, building outlines, etc. (2/7) -> Other (Expected / Low difficulty).
- 8. Automatic usage reports and function memory usage may be good for making decisions on what features to optimize, and plain knowledge on what operations take time. (3/7) -> Performance visualization Identification (Medium value / Low difficulty).
- 9. Add more ways to plot data. (7/7) -> Flexibility (High value / Medium difficulty).
- 13. Visualization comparisons, pitch one plot against another. (6/7) -> Comparison (High value / Medium difficulty).
- 16. More techniques for filtering data in visualizations requested (7/7) -> Filtering (High value / Medium-High difficulty).
- 17. Suggestions for data grouping, topologies and key performance indicators.(7/7) -> Data structuring (Medium value / Medium difficulty).
- 20. Multithread support to avoid application freezing. (3/7) -> Access time (Medium value / Medium difficulty).

B

Appendix Design Iteration 2

Following are the general opinions elicited from Design Cycle 2 feedback session. Each point describes the a certain feedback from the user, followed by a category:

1. Delimitations of data handlers may be relevant as number of simulations increase. -> Access time
2. Preprocessing the data may cause a time issue when the byte size of simulations increase, how you considered other compression-storage algorithms? -> Access time
3. Could the plots be normalized in size? Have the x-y axis same even if size of widgets are uneven. -> Format
4. Overlaying plots. For some plot types, it would be nice to plot them on top of each other. -> Comparitions
5. 3-D plots support are sought after. -> Other
6. Plot specific improvements e.g. building outlines on heatmaps. -> Other
7. Interface for custom functions e.g. for plotting. -> Scalability

Top categories & Priority

1. Scalability, Mapping (High priority)
2. Comparison, Filtering, Access Time (Medium priority)
3. Other (Low priority)

B.1 Actions for final iteration

This appendix is a compiled list of actions that are chosen to be implemented for the next iteration of the project. It is based on the items from Appendix B and each item selected is categorized by us with value and difficulty.

- 6. Work on interactions between visualizations (High value / High difficulty)
- 7. Custom visualizations for mapping (High value / High difficulty)

C

Appendix Design Iteration 3

1. Python objects preferred over DSL. (5/5) -> Mapping
    - Expert users already familiar with Python and office aims to have all employees learning Python.
    - DSL may prove useful for beginners but in long-term Python will be more relevant in all regards.
2. (DSL feedback) Use interactions to select the data instead of having to specify the mapping. -> Already done through Python Interface. (4/5) -> Data abstraction
3. Possible further abstraction on the data input. (1/5) -> Data abstraction
    - Investigate different abstraction levels.
    - Maybe further abstraction is good/bad.
4. Scalability, make it easy to add new plot. (5/5) -> Scalability
5. Possibility to specify plots within plots with new Python interface. (2/5) -> Mapping
6. Further plot-specific visual requests (3/5) -> Scalability
    - Standard axis lengths
    - Plot Formats
    - Plot settings, e.g. Label titles, set color scales etc.
    - Label correctness

Top categories & feedback points

1. Abstraction (5)
2. Mapping (5)
3. Scalability (3)