CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A quality-focused explainability approach for robot mission planning

Explaining the impact of quality attribute prioritization on self-adaptive systems' behavior

Master's thesis in Computer science and engineering

Erik Nilsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

A quality-focused explainability approach for robot mission planning

Explaining the impact of quality attribute prioritization on self-adaptive systems' behavior

Erik Nilsson

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

A quality-focused explainability approach for robot mission planning
Explaining the impact of quality attribute prioritization on self-adaptive systems' behavior
Erik Nilsson

Supervisor: Rebekka Wohlrab, Department of Computer Science and Engineering
Examiner: Gregory Gay, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2023

A quality-focused explainability approach for robot mission planning
Explaining the impact of quality attribute prioritization on self-adaptive systems' behavior
Erik Nilsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Self-adaptive systems adjust their behavior in response to changes in their environment and plan their actions by prioritizing different quality attributes, such as travel time, safety, and privacy. These systems are seen as black boxes, and stakeholders find it challenging to understand their behavior. This study aims to help humans better understand how the prioritization of quality attributes affects the actions of self-adaptive systems. We used design science research to identify, explore solutions for, and mitigate key challenges stakeholders perceive in understanding the behavior of self-adaptive systems. Our research led to the development of a tool that uses a combination of visual and text features to provide both a descriptive and a contrastive explanation of the system's behavior. The tool was evaluated in a think-aloud study, and our findings suggest that using our developed tool can significantly mitigate stakeholders' perceived challenges, especially the complexity.

Keywords: Self-adaptive system, prioritization, quality attribute, utility function, cost function, contrastive explanation, automated planning.

Acknowledgments

I want to thank all the participants in my interviews for their help in conducting this study. And special thanks to my supervisor Rebekka Wohlrab, who motivated me by providing plenty of expertise and support.

Erik Nilsson, Gothenburg, June 2023

Contents

List of Figures

List of Tables

1
Introduction

Self-adaptive systems are systems that adjust their behavior in response to changes in their environment, e.g., autonomous vehicles and robots. By adapting to changing conditions, these systems can minimize downtime, reduce errors, and optimize resource utilization [1]. Additionally, self-adaptive systems can enhance the user experience by providing personalized services and accommodating individual preferences.

A crucial part of self-adaptive systems is generating a plan of the actions to be performed. Self-adaptive systems plan their actions by considering multiple quality attributes, which are measurable properties of the system or its surroundings. Examples of quality attributes are travel time, safety, and energy consumption. To effectively balance and trade-off these attributes, they must first be encoded, most often in a *utility function* [2]. Utility functions map human preferences to a numerical output, commonly normalized between 0 and 1, where 0 is the worst-case utility value, and 1 is the best-case utility value [24]. Utility functions are used in economics and decision theory to numerically explain human behavior and decision-making.

A utility function is defined by the sum of weighted quality attributes in most related work [3, 24, 21]. For example, we may consider the utility function for a plan p

$$u(p) = x * U_{travel\_time}(p) + y * u_{safetey}(p) + z * u_{energy}(p)$$

, where $x, y, z \in \mathbb{R}^+$, and $x + y + z = 1$.

For self-adaptive systems to plan the optimal path in relation to their current preferences, they rely on an automated planner [1]. Concretely, an automated planner chooses the plan with the highest utility value. This automated planning is also constrained by hard requirements, e.g., preventing harm to humans and maintaining sufficient energy levels.

Creating a self-adaptive system involves multiple stakeholders with a multitude of different preferences. This difference in preferences varies vastly even within the same category of stakeholders (e.g., end users or engineers). The stakeholders find it challenging to determine the different quality attributes' effect upon automated planning [24]. Since the weights in a utility function are often manually tuned, the stakeholders require tools and decision-making techniques to assist their prioritization of quality attributes and reaching consensus on the prioritization [4, 5].

1.1 Problem Description

Self-adaptive systems are becoming increasingly ubiquitous for dynamic systems in the software engineering domain [3]. However, due to the complexity and dynamically changing properties of self-adaptive systems, stakeholders of these systems find it challenging to understand their behavior [24]. These systems are mostly seen as black boxes, where x goes in and y comes out, without understanding the reasoning for why the specific outcome was the result. This lack of understanding negatively affects the prioritization of quality attributes, as the stakeholders don't understand how to properly prioritize the quality attributes in order for the system to generate their desired output [7].
Improving human understanding of self-adaptive systems can enhance the decision-making process of prioritizing quality attributes, increasing these systems' efficacy [8]. Increased efficacy may contribute to the field of software engineering by aiding the broader adoption of self-adaptive systems, given their potential to adapt dynamically to changing conditions.
Increased efficacy may aid the broader adoption of self-adaptive systems, as they

1.2 Purpose of the Study

This study aims to help humans better understand how the prioritization of quality attributes affects the actions of self-adaptive systems. To achieve this, a tool has been developed using design science [9]. This tool is designed to help users visualize and understand how their prioritization of quality attributes affects the automated planning process in a self-adaptive system.
The effectiveness and usability of this tool have been evaluated using a think-aloud study, a method commonly used in usability studies to understand the user's thought process as they interact with a tool or a system [18].

Three key research questions have been defined to accomplish the goal of this study. They are divided into the phases of a regulative cycle: **RQ1** what are the perceived challenges, **RQ2** potential solutions, and **RQ3** to what extent can the potential solutions mitigate the perceived challenges [9].

RQ1: *What are stakeholders' challenges with understanding how the prioritization of quality attributes affects the behavior of self-adaptive systems?*
To develop a tool that helps stakeholders better understand quality attributes' priorities effect on automated planning, it is essential to understand what difficulties the stakeholders have with understanding them.

RQ2: *Which potential solutions exist to help the stakeholders better understand how the different prioritization of quality attributes affects the behavior of self-adaptive systems?*
There is no single supreme solution to this problem. It is therefore required to assess different solutions and choose one that's impactful enough and can be developed

with the given resources and the given time restriction.

RQ3: *To what extent can frequently occurring challenges with understanding the prioritization of quality attributes' effect on the self-adaptive systems' behavior be reduced?*
The goal is to help stakeholders assess the impact of quality attributes' priorities on generated travel plans. Therefore, it is necessary to understand to what extent the tool can help them.

1.3 Significance of the study

This study aims to benefit both stakeholders of self-adaptive systems and researchers in this research area. In addition, this research addresses a gap in the current understanding of the effect prioritization of quality attributes has on automated planning.

The contributions of this study are twofold and advance the state of the art in the field. Firstly, the stakeholders benefit from the contribution of an empirically evaluated prototype design artifact that helps users better visualize and understand how their prioritization of quality attributes affects automated planning. By bridging this gap, the artifact enables stakeholders to make better-informed decisions when prioritizing quality attributes.
Secondly, this study provides useful solutions to mitigate stakeholders' challenges of understanding how the prioritization of quality attributes affects the behavior of self-adaptive systems. Researchers may use these solutions to make further advancements within the domain of self-adaptive systems. They may also use this study's empirical insights to better understand how humans reason about quality attributes and priorities and how they understand robotic systems.

This prototype artifact is, to our knowledge, the only tool that helps stakeholders to assess the impact their utility function has on generated travel plans in self-adaptive systems for robotics.

1.4 Thesis Outline

Section 2 explains key terminology and provides an example of a robot mission plan. Section 3 presents related work. Section 4 describes the research method. Section 5 presents the design artifact and the rationale behind its implementation decisions. Section 6 presents our findings related to our research questions. Section 7 discusses our findings, describes threats to our research's validity and how we mitigated them, and provides insight into potential future work. Section 8 summarizes this thesis and provides key takeaways.

2
Background

This section explains key terminology related to self-adaptive systems and stakeholders' struggles with prioritizing quality attributes. It also provides an example of how a self-adaptive system generates its automated plan and how the prioritization of quality attributes affects it.

2.1 Self-adaptive systems

Self-adaptive systems are systems that adjust their behavior in response to changes in their environment, e.g., autonomous vehicles and robots. By adapting to changing conditions, these systems can minimize downtime, reduce errors, and optimize resource utilization [1]. These systems play a crucial role in applications where adapting to dynamic conditions is essential for the system's functionality and efficiency, such as robotics and cloud computing [6]. An example of this is a robot vacuum cleaner. The robot needs to adjust its path planning depending on obstacles it encounters, and it needs to always have sufficient battery to make the trip back to its charger. Another example is video streaming. Most video-streaming services can automatically adjust the quality of a video stream in real-time based on the condition of the end user's connection.

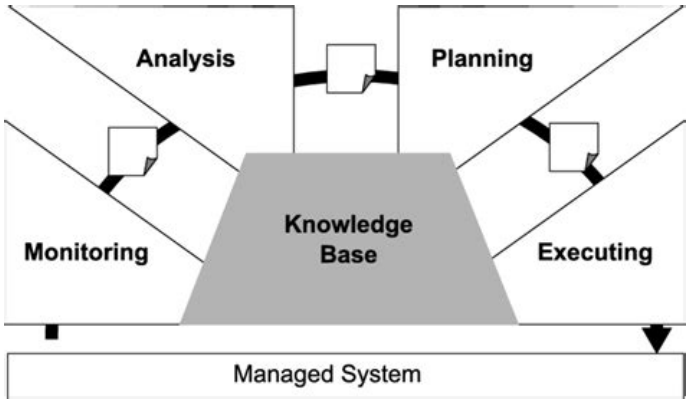

Figure 2.1: IBM's MAPE-K framework [12]

IBM's MAPE-K is a framework commonly used for self-adaptive systems [1]. The MAPE-K framework is a feedback control loop that enables a system to monitor, analyze, plan, execute, and store knowledge to improve its performance over time, see Figure 2.1. In this study, we focus mainly on the planning part of the framework. In the context of this study, planning most often refers to automated planning of a

path for robots to traverse. However, planning may also refer to a computer system's plan to reallocate resources to improve its health, e.g., Kubernetes[1] deploying self-healing pods in containerized cloud computing.

2.2 Quality attributes, utility functions, and cost functions

Self-adaptive systems use a multitude of different quality attributes for their automated planning. A quality attribute refers to a characteristic of the system that is used to evaluate its performance, e.g., speed, safety, and energy level. These quality attributes are necessary for ensuring that the system can adapt to changing conditions in the environment, such as changes in workload, resource availability, or security threats [11]. By monitoring and measuring these quality attributes, a self-adaptive system can make informed decisions about adapting to changing conditions to maintain or improve its performance [1].
Consider a self-flying robot that delivers packages. It has to be fast enough for the company's logistics to run smoothly, so it needs to focus on speed. But it's even more important not to damage the package, humans, and properties, so safety is required. The drone is also constrained by having enough battery to make the delivery and return home, requiring it to be energy-efficient.
Self-adaptive systems encode these quality attributes in a utility function, which enables the system to plan a path that depends on these quality attributes. Utility functions are usually constrained by some restrictions, e.g., don't deplete all fuel. A utility function is defined, in most related work [3], by the sum of weighted quality attributes. Stakeholders set these weights and will decide how the self-adaptive system plans its path.
A cost function is an alternative mathematical function to encode quality attributes. Cost functions represent the cost associated with a certain action or sequence of actions. For instance, in a path planning context, the cost might represent the distance traveled, energy expended, or risk encountered. A common cost function is Dijkstra's algorithm[2]. These functions aim to minimize the cost rather than maximizing a utility function, which is required when calculating optimal paths faster than $\mathcal{O}(n^2)$.

2.3 Stakeholders' preferences

Stakeholders are responsible for deciding the weights of different quality attributes for automated planning in self-adaptive systems. They must also enforce constraints upon the system, such as maintaining sufficient fuel levels. There are multiple stakeholders who all have different preferences regarding how automated planning should behave. Therefore, stakeholders in this thesis refer to everyone affected by a specific self-adaptive system: practitioners, end users, developers, etc., and not only owners

[1] https://kubernetes.io/
[2] Dijkstra's algorithm

and shareholders of a company. Some might only care about travel time, while others value safety highly. But it is not easy for stakeholders to understand how the weights of different quality attributes affect automated planning [10]. For example, a cost function that favors safety over travel time might not choose the safest path because an alternative path is much faster but slightly less safe. Therefore, stakeholders require tools and decision-making techniques to assist their prioritization of quality attributes and reach consensus [4, 5].

2.4 Planning example

Figure 2.2 depicts a map containing nodes and paths that a robot can choose from to reach its destination, where the number over each path is the distance between nodes. In this example, we have the quality attributes: *travel time*, *collisions*, and *intrusiveness* (of privacy).
Traversing a normal path yields a collision cost of 0, a partially occluded path costs 1, and an occluded path costs 2.
Passing a public location yields an intrusiveness cost of 0, a semi-private location costs 1, and a private location costs 2.

To calculate the optimal path, we define the cost function for a plan σ:

$$c(\sigma) = w_{tt} * c_{tt}(\phi_{tt}(\sigma)) + w_{col} * c_{col}(\phi_{col}(\sigma)a) + w_{int} * c_{int}(\phi_{int}(\sigma))$$

, where $w_{tt}, w_{col}, w_{int} \in \mathbb{R}^+$, and $w_{tt} + w_{col} + w_{int} = 1$.

c_* is the local cost function for each quality attribute, $\phi_*(\sigma)$ is the total cost of each attribute in a complete path, and w_* is the weighted priority of each quality attribute. The local cost function c_* is calculated by a complete path's cost for a certain quality attribute in relation to the least expensive path's cost for the same attribute.
The path with the lowest travel time is (1) → (3) → (4) → (5) with a cost of 3.414.
The path with the lowest collision cost is (1) → (2) → (5) with a cost of 1.
The path with the lowest intrusiveness cost is (1) → (3) → (4) → (5) with a cost of 1.
The local utility functions are therefore defined as:

$$u_t(\phi_{tt}(\sigma)) = \frac{\phi_{tt}(\sigma)}{3.414}, \qquad u_c(\phi_{col}(\sigma)) = \frac{\phi_{col}(\sigma)}{1}, \qquad u_i(\phi_{int}(\sigma)) = \frac{\phi_{int}(\sigma)}{1}$$

A self-adaptive system's automated planner chooses its path depending on the weighted priorities of quality attributes. Table 2.1 displays the attribute costs and cost function value for several sets of weights. E.g., in the case where only travel time is relevant ($w_{tt} = 1$), the path (1) → (3) → (4) → (5) is optimal. And if the system only cares about collision ($w_{col} = 1$), then the path (1) → (4) → (5) is deemed optimal.
Fully prioritizing a single quality attribute is easy, but mixing them all up does not always yield intuitive paths. E.g., for the weights $w_{tt} = 0.25, w_{col} = 0.5, w_{int} = 0.25$

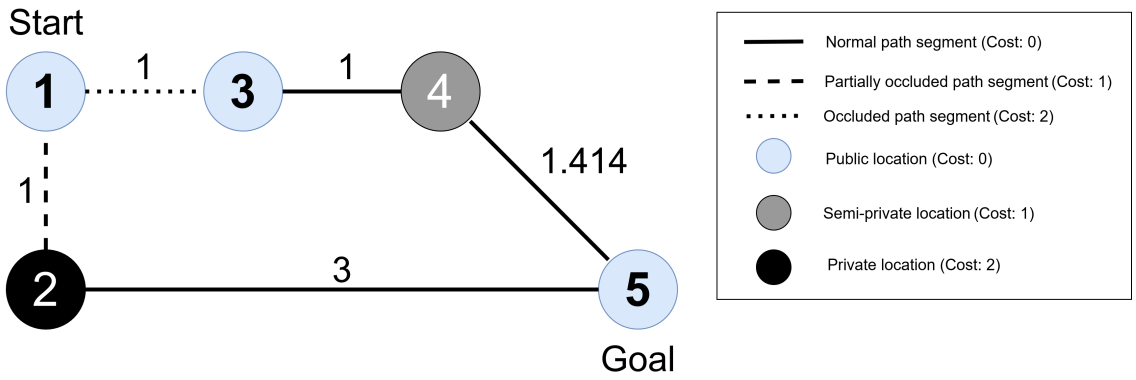

Figure 2.2: Example of an automated planning graph

the path ① → ④ → ⑤ is deemed optimal by the robot. Why did it choose that path? In this small graph, it is quite easy to calculate it by hand or intuition. However, plenty of minor mathematical calculations are combined, making it error-prone.
Real-world scenarios would have dozens of nodes and edges, making it nearly impossible to calculate the optimal path by hand or understand why it was deemed optimal by the robot. Therefore a tool is needed to aid humans in their task of understanding the self-adaptive system better, which yields increased trust in the system.

Table 2.1: Cost function value of paths for different weights.

w_{tt}	w_{col}	w_{int}	Locations	Optimal	ϕ_{tt}	ϕ_{col}	ϕ_{int}	c_{tt}	c_{col}	c_{int}	c
1	0	0	②,③	Yes	3.414	2	1	1	2	1	1
			④	No	4	1	2	1.172	1	2	1.172
0	1	0	②,③	No	3.414	2	1	1	2	1	2
			④	Yes	4	1	2	1.172	1	2	1
0	0	1	②,③	Yes	3.414	2	1	1	2	1	1
			④	No	4	1	2	1.172	1	2	2
0	0.5	0.5	②,③	Yes	3.414	2	1	1	2	1	1.5
			④	Yes	4	1	2	1.172	1	2	1.5
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	②,③	Yes	3.414	2	1	1	2	1	1.333
			④	No	4	1	2	1.172	1	2	1.391
0.25	0.5	0.25	②,③	No	3.414	2	1	1	2	1	1.5
			④	Yes	4	1	2	1.172	1	2	1.293

3

Related Work

This section covers related work for explainability, contrastive explanations, utility functions, and tool support in self-adaptive systems, with a particular emphasis on robotics. It concludes with a short summary of how this thesis' extends past literature.

3.1 Explainability for self-adaptive systems

Research has indicated the need for explainability in self-adaptive systems [5]. N. Li et al. [17] presents an approach to determine when to provide an explanation in human-involved self-adaptive systems. The approach is based on criteria considering system behavior and user expertise. They found that explanations increased users' understanding and trust in the self-adaptive system, which increases system performance.

Hellström and Bensch [20] discuss the importance of developing robots that are understandable to humans. The authors argue that as robots become more prevalent in society, it is crucial that people can easily understand how they work, what they are doing, and why they are doing it. They also emphasize that understandable robots are essential for building trust between humans and robots.

Reynolds et al. [19] explains how self-adaptive systems often are seen as black boxes, which is a problem as human users can't track or justify the systems' decisions. However, tracking everything that leads to a specific state within the system is not only computationally heavy; it's also too much information which leads to further confusion. Therefore, the researchers developed a system that captures key decisions, which a provenance graph uses to relate the system's actors, entities, and activities.

Generating explanations of self-adaptive systems can be categorized into two different types of explanations: the *what-explanation*, and the *why-explanation* [21]. The *what-explanation* describes the optimal solution to a particular planning problem, while the *why-explanation* provides a rationale of why that solution was deemed optimal. In this thesis, the design artifact first explains *what* happens with a specific prioritization of quality attributes and then provides a rationale of *why* that prioritization yielded the given outcome.

3.2 Contrastive explanation

A contrastive explanation is an approach that tries to explain why something is occurring in contrast to another occurrence, e.g., "Why is x happening rather than y?".

Contrastive explanations can help human users identify potential biases and errors in a robot's decision-making [23]. By providing information about why a particular decision was made, users can better understand the robot's underlying logic, resulting in improved transparency and trust in the robot. However, the contrastive explanation must be tailored to suit the intended audience's expertise.

Contrastive explanations have been proven helpful in aiding humans better understand other complex systems such as neural networks [27]. In the aforementioned case, the authors copied all inputs into the neural network, tweaked them, and highlighted the difference between the original and modified decisions. The modified instance is then presented to the user along with the original instance, allowing the user to compare and contrast the two and understand why the neural network made the decision it did.

Sukkerd et al. [22] use contrastive explanation to help end-users gain higher trust for multi-objective automated decision-making in robots. Their approach tries to explain how a robot reasons when selecting a planning solution by producing a quality attribute contrastive explanation. Their results show their contrastive explanation approach to impact end-users understanding of the trade-offs made by automated decision-making robots.

3.3 Defining utility functions

When dealing with self-adaptive systems, it is crucial to understand that specifying utility functions at design time is insufficient and that the elicitation and readjustment of preferences during runtime is often necessary [28]. Song et al. [29] proposed a self-adaptation approach that uses runtime models and constraint solving to adapt software systems based on end-user preferences. The approach involves the creation of a runtime model of the system, which is used to monitor the system and detect changes in its behavior. When changes are detected, the approach uses constraint solving to determine the best adaptation action based on end-user preferences.

Wohlrab and Garlan [24] developed a tool-supported negotiation technique to help stakeholders to elicit constraints and preferences for self-adaptive systems. Their tool is based on the Analytic Hierarchy Process [26] for pairwise comparison of quality attributes prioritization and is supported by a blackboard system. The negotiation technique was proven helpful when agreeing on prioritizing quality attributes, which defines the weights of a utility function. However, their tool doesn't help humans assess the impact of quality attribute prioritization on self-adaptive systems' automated planning.

3.4 Tool support

Handling multiple quality attributes in self-adaptive systems is an understudied research area [11]. Research concerning multiple quality attributes in self-adaptive systems generally focuses on utilizing decision-making techniques. However, studies have shown that combining these decision-making techniques with software tools can better understand quality attributes' effect on self-adaptive systems [24]. To our knowledge, there is currently no tool that helps stakeholders assess the impact their utility function has on generated travel plans in self-adaptive systems for robotics. Contrastive explanations have been proven beneficial to help understand autonomous systems decision-making [22], which could be an interesting feature to base a tool on. Concretely, it could help discern how a particular change in the priorities of quality attributes might lead to different optimal plans that an automated planner generates.

3.5 Extending past literature

This thesis focuses on developing a tool that supports humans in assessing the impact of quality attribute prioritization on self-adaptive systems' automated planning. This assessment is done by first generating a descriptive explanation of *what* decision was made by the system, followed by a contrastive explanation of *why* said decision was deemed optimal. This assessment aims to increase human understanding of self-adaptive systems, resulting in better quality attribute prioritization that yields higher system efficacy.

4
Methods

In this section, the research method is presented. The thesis was developed over two iterations. The first iteration focused on identifying perceived challenges (**RQ1**) and potential solutions to mitigate these challenges (**RQ2**). The second iteration focused on completing the implementation of the potential solutions, then evaluating to what extent they can mitigate the perceived challenges (**RQ3)**.

4.1 Design Science

Design Science has become an essential approach in Software Engineering due to its emphasis on solving practical problems through the development or improvement of artifacts [30]. In Software Engineering, artifacts include software systems, development methodologies, and other related technologies.
Wieringa proposes a framework for design science based on the idea that design science research involves nested cycles of problem-solving, with each cycle focused on solving a specific problem nested within a more extensive problem [31]. The framework consists of a regulative cycle containing four main stages: problem analysis, solution design, implementation, and evaluation (Figure 4.1).

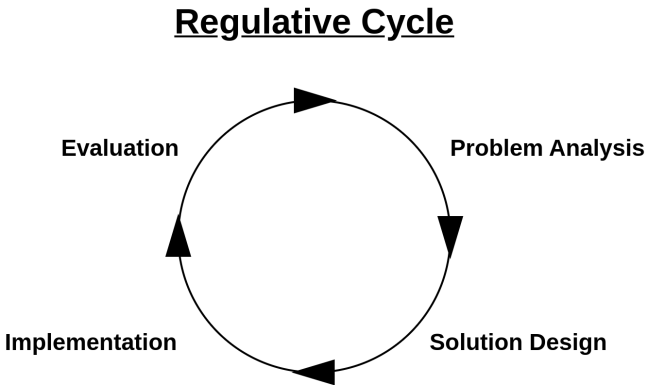

Figure 4.1: A design study's regulative cycle defined by Wieringa [31]

Knauss proposed seven guidelines for applying design science research for Master's Thesis work in industry [9]. Knauss' guidelines are heavily influenced by Wieringa's regulative cycle. The seven guidelines are:

- **G1:** Define the **artifact** early.

- **G2:** Work in iterations. Contribute to each research question in each iteration.
- **G3:** Define research questions with respect to the regulative cycle, i.e., one related to the problem, one related to potential solutions and their construction, and one related to evaluation.
- **G4:** Have regular meetings.
- **G5:** Shift emphasis between cycles. Work on each research question in each cycle, but put more emphasis on **RQ1** (Problem) in Cycle one, **RQ2** (Solutions) in Cycle two, and **RQ3** (Evaluation) in Cycle three.
- **G6:** Have a dedicated section to describe the artifact.
- **G7:** Write the thesis document as you go.

While these guidelines were created with a Master's Thesis in industries as a focus, they are excellent guidelines for any Master's Thesis using design science. Therefore this design science Master's Thesis adopted these guidelines with a few tweaks. It was identified early that the core functionality of the artifact would need more time than a single cycle. This led to the artifact being developed in two larger cycles instead of three, where the first cycle focused on **RQ1** and **RQ2**, and the second cycle focused on **RQ2** and **RQ3**, see Figure 4.2.

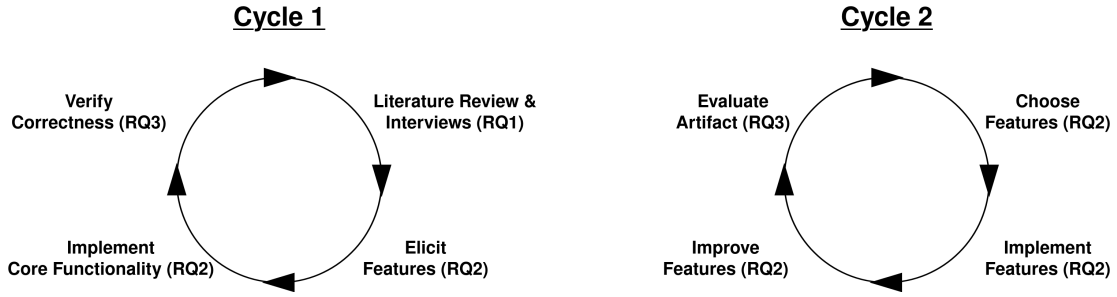

Figure 4.2: Overview of each cycle

Cycle one focused mainly on identifying challenges with understanding how the prioritization of quality attributes affects the behavior of self-adaptive systems (**RQ1**) and eliciting potential solutions to mitigate these challenges (**RQ2**). To achieve this, potential artifact users were interviewed in a think-aloud study, where the participants had to complete a survey afterward for data triangulation (see Section 4.3).
The data gathered from these interviews were coded into themes to link the commonality between each other's statements (see Section 4.2). The different themes were related to either identified challenges that the participant had perceived or solutions to these challenges. A set of potential features were elicited from these themes. A few of these features were identified as core functionality and were the first to be implemented in the artifact, e.g., generating a graph (map) and finding the optimal path.
Because of the importance of this core functionality and it requiring shortest-path algorithms, the first cycle concluded with extensive verification of the core functionalities' correctness.

Cycle two shifted its focus to discern which of the remaining features, from the set elicited in cycle one, would be most beneficial for the artifact and implement them. The selection process involved weighing each feature's usefulness and interoperability with the core functions.
After implementing a few of these features, the choice was made to refine existing features rather than implement more. This decision was made because it was deemed more important at this stage to improve the reliability and user-friendliness of the artifact.
After creating a satisfactory prototype of the artifact, another set of interviews was conducted as part of a think-aloud study (see Section 4.4). Similar to cycle one, the participants needed to complete a survey for data triangulation. These interviews aimed to determine to what extent the artifact could mitigate the perceived problems (**RQ3**) that were found in the previous cycle.
The data gathered from these interviews consists of both Likert scale rankings of the features from the questionnaire, as well as informative statements of why the participants found certain features helpful. The Likert scale rankings link features' relationship to the perceived problem they mitigated. The informative statements were coded into themes to accurately link them to the features' impact on the perceived challenges. See Section 6.2 for more details.

Table 4.1: List of interviewees.

Interviewee ID	Occupation	Years of work experience in tech
1	Engineering Manager	3-5
2	Software Development Student (5th year)	1-2
3	Software Engineering Student (5th year)	1-2
4	Software Developer	6+
5	Backend Developer	3-5
6	Cloud Engineer/Architect	6+
7	UX-design student (1st year)	0
8	Consultant Manager	6+
9	Software Architect	6+
10	Product Owner	6+

Table 4.1 contains a list of all interviewees and can be used to track their citations and participation. We expect most users of this tool to have a somewhat technical background, and therefore selected participants covering a mix of technical experience. The participants were selected from our established connections using convenience sampling, with a heavy focus on availability.
The pre-study had a total of seven participants (interviewees 1-7), which was satisfactory because the results quickly became heavily saturated. The evaluation of the artifact also consisted of seven participants (interviewees 1,2,5 & 7-10). However, a few results in the evaluation weren't saturated and would benefit from more participants. There also weren't any experts in the field of self-adaptive systems participating in this study. Only interviewee 10 had prior experience in work related to self-adaptive systems. These flaws were the result of both time and resource constrictions.

4.2 Qualitative Analysis

This section describes the methods used to analyze the qualitative data gathered from the interviews conducted in this study.

4.2.1 Coding & Thematic analysis

Coding qualitative data is a crucial step in the qualitative research process, as it helps to organize, analyze, and make sense of the data collected through interviews, focus groups, or other qualitative methods [14]. However, qualitative data can be overwhelming, as it is typically rich in detail and complexity, making it difficult to manage and analyze without some structure.
Coding involves categorizing and labeling sections of the data with codes or labels that capture the key themes within the data. This helps to identify patterns and relationships within the data, and researchers can increase the reliability and validity of their findings, ensuring that their interpretations are grounded in the data [14].
Coding qualitative data is an essential step in qualitative research. Various approaches to coding qualitative data have been discussed in the literature, including thematic analysis which is used in this study [13].
Thematic analysis is a widely used qualitative research method for systematically and rigorously analyzing data. It is a flexible approach that can be applied to various research questions and data sources, including interviews, focus groups, and written texts [13, 15].
Thematic analysis involves identifying and analyzing patterns and themes within the data. This is done through a series of steps that involve familiarization with the data, generating initial codes, searching for themes, reviewing themes, defining and naming themes, and producing the final report [13].
One of the strengths of thematic analysis is its ability to provide a rich and detailed account of the data while also allowing for a level of abstraction that enables the identification of broader patterns and themes [15].
To help researchers save time performing thematic analysis, the use of computer-assisted coding software to assist with the coding and theme generation is utilized [14]. One of these software tools is QualCoder, which has been utilized to code the data collected from all interviews in this study.

The main themes are our research questions. From these main themes, more increasingly specific themes were created, structured as a tree. Originally, at the lowest level of the trees, were the statements from interviewees that were deemed important. However, as the number of important thoughts and statements grew, the need to generalize them arose. Having singled out the important statements, it became easy to label them with the same code and then connect them with a similar theme of a higher level. E.g., the statements:

> *"It is far too complex to consider all possible routes."*
>
> (Interviewee 1)

"Adding additional quality attributes greatly impacts the difficulty."
(Interviewee 5)

could both be categorized within the same code, *Complexity*, and later linked to the theme *Problems*. This step was repeated at the higher levels as well. For each iteration, it became clearer that the interviewees' statements often had the same meaning.

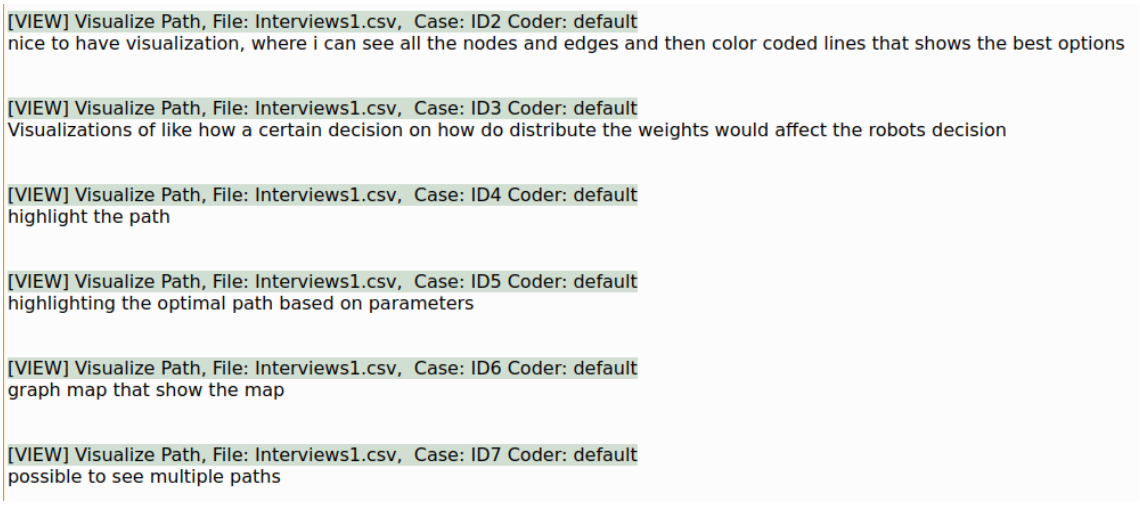

Figure 4.3: Generated QualCoder report listing all interviewees that thought *Visualize Path* was a good solution

After finishing coding themes for all the data, we generated reports to find which participants, and how many of them, saw eye to eye. An example of this can be seen in Figure 4.3, where multiple interviewees talked about potential solutions to their problems when completing their tasks. While all their statements differ, and even their core ideas differ slightly, it is clear to see that they are talking about similar solutions. This generalization of the participants' statements, turning them into themes, dramatically decreases the difficulty of understanding the thoughts they try to convey.

4.3 Identifying challenges and potential solutions

Qualitative interviews were conducted with potential users of the tool as interviewees (interviewees 1-7), as per guidelines for this study in a relatively unexplored area of research [14]. The purpose was to identify their difficulties with understanding how weighted quality attributes affect a self-adaptive system's decision-making and elicit potential features for the tool to help mitigate these problems.
The interviews were conducted one on one, where the participant was given an explanation of how quality attributes affect self-adaptive systems' decision-making. The participant solved a few tasks by calculating the optimal path for a self-adaptive robot in a graph shown to them[1][2]. The intention of these tasks was for the intervie-

[1]Pre-study Interview Guide: tinyurl.com/cwza7nd
[2]Pre-study Questionnaire Answers: tinyurl.com/37vwubt3

wee to describe their thought process, using think-aloud problem-solving. Think-aloud problem solving is helpful to elicit people's mental models [16]. Each participant was also given time to discuss the explanation given to them, to establish a common terminology and minimize misunderstandings.
After completing the tasks, the participant completed a survey with questions related to the problem presented to them. The questions were divided into two main categories. First, they answered questions about their perceived difficulties with understanding why the robot chose a specific optimal path. Then they were asked to elicit potential features for a digital tool to mitigate their aforementioned difficulties. During the survey, the participant continued to think aloud and was allowed interactive dialogue between the participant and the interviewer.
The data collected from the interviews were coded to analyze it and better understand what the participant meant, instead of what they were saying. The codes used to analyze these interviews were divided into categories related to **RQ1** and **RQ2**, and the most frequently appearing themes can be seen in Section 6.1's subsections.

4.4 Evaluation of the artifact

After the completion of a satisfying prototype of the tool, a second round of qualitative interviews (interviewees 1,2,5 & 7-10) was conducted, following the same guidelines established in the first set of interviews. The purpose of these interviews was to evaluate how effectively the tool-assisted users understand how weighted quality attributes affect the decision-making of a self-adaptive system.
The interviews were again conducted one-on-one, where each participant was given an explanation of the subject and a demonstration of the tool. The participant was provided with a similar set of tasks, albeit more complex, where the participant determined the optimal path in a graph[3][4]. The participant was first asked to complete the task by hand and then use the tool to check for correctness, utilizing its various features to better understand why a specific path was chosen. This approach directly compared their experience and understanding with and without the tool.
After completing the tasks, the participant completed a survey to gather feedback on the tool. The survey focused on questions related to how useful the varying features of the tools were in mitigating the perceived problems elicited from the first set of interviews.
The participant was encouraged to employ the think-aloud problem-solving method throughout the interview. This helped to capture their thought process and allowed for interactive dialogue between the participant and the interviewer, ensuring continuous and immediate feedback about their experience with the tool.

[3]Evaluation Interview Guide: tinyurl.com/2nps4dzz
[4]Evaluation Questionnaire Answers: tinyurl.com/bdfadwwj

5

An Explainability Tool for Self-Adaptive Systems

This section explains the construction of the artifact, a prototype of an explainability tool for self-adaptive systems' automated planning in robotics. The various features of the tool are covered, as well as why they were deemed beneficial to implement. The section is split into two: *Core Functionality*, and *Explanatory Features*. The *core functionality* consists of features required to display the optimal path in a graph, explaining *what* is happening. The *explanatory features* focuses on explaining *why* a specific path is deemed optimal.

5.1 Core Functionality

After studying the results of the thematic analysis of the first set of interviews and reviewing related literature, a set of core features was elicited.

1. Display a graph.
2. Pick start and end nodes.
3. Prioritize quality attributes.
4. Display the optimal path.

To create these features, it was necessary to have the data of a graph. Therefore, a feature that generated random reusable maps was considered. However, this feature was not deemed impactful to the study, and already existing graph data was instead incorporated to save development time. The graph data used in this tool was taken from a public repository[1] and contains the quality attributes: *Travel time*, *collision*, and *intrusiveness*.
The artifact was developed as a web-app, rather than a stand-alone program, to increase its accessibility and reach a broader audience. Figure 5.1 provides an overview of the tool.

5.1.1 Display a graph

To better understand how the prioritization of quality attributes affects automated planning, it was quite apparent to first display the graph it could traverse. The framework *vis-network* was chosen to display the graphs because of its feature to

[1] https://github.com/cmu-able/explainable-planning

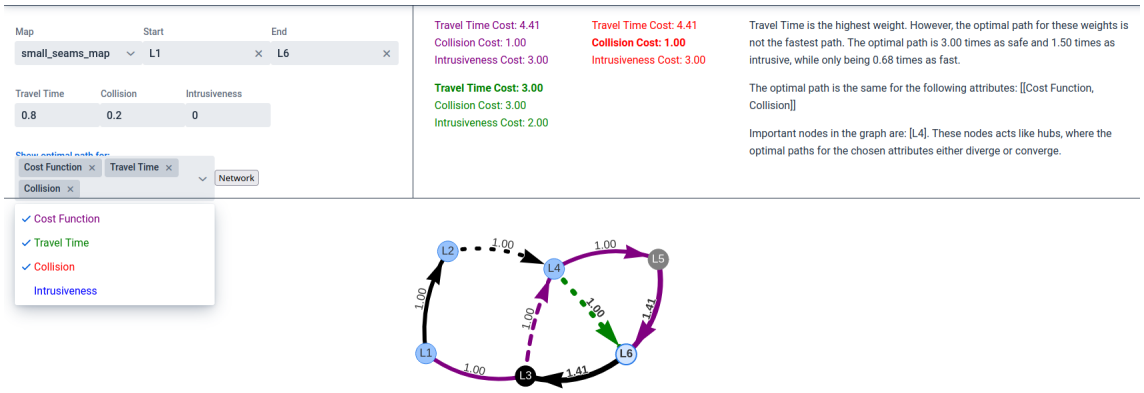

Figure 5.1: Overview picture of the tool

change the shape and color of edges and nodes. Using this existing framework was deemed a significant time save, rather than developing a tailored framework for this artifact.
The graph's visuals are based on the example's legend in Section 2.4 and can be seen in Figure 5.2. This visual representation of the different quality attributes was chosen due to its proven efficacy in similar examples in related works [25].

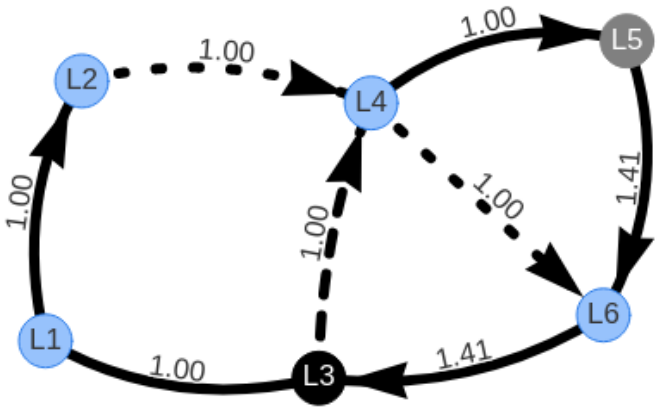

Figure 5.2: Displayed graph in the tool

5.1.2 Prioritize quality attributes

Weighting the quality attributes is straightforward, as the tool only needs an input field for each quality attribute. These inputs must also be sanitized to ensure they are non-negative numbers that sum up to 1.

The difficult part was to choose which mathematical function to encode the quality attributes. At first, a utility function was preferred since it is the most common function when encoding quality attributes in self-adaptive systems. However, utility functions focus on maximizing a value, which renders the artifact unable to utilize scalable searching algorithms. Therefore it was deemed necessary to use a cost

function due to them minimizing a value, which scalable searching algorithms can utilize.

Dijkstra's was the chosen shortest-path algorithm because of its ability to always find the optimal path. A* was considered, as it is superior in time efficiency compared to Dijkstra's. However, A* only finds the optimal path to a specified node, while Dijkstra's finds the optimal paths to every node. Knowing every optimal path was considered helpful in the development of other features, and this is why Dijkstra's was the preferred algorithm.

There was also the need to combat the fact that the quality attributes in a graph are often blown out of proportion compared to each other. E.g., travel time often has a substantially higher cost than intrusiveness. Therefore, all quality attributes in the cost function are normalized to deny attributes with a generally higher cost to dominate other attributes.

The chosen normalization solution compared the cost for each quality attribute in the current path to the lowest possible cost of each attribute. Normalizing against the minimal cost instead of the maximum cost was deemed optimal because it requires less computational power to find. The minimal cost can be found using Dijkstra's algorithm ($\mathcal{O}(n \log n)$), compared to using, e.g., Depth-first search ($\mathcal{O}(n^2)$) to find the maximum cost. See Section 2.4 for a concrete example of the chosen cost function.

5.1.3 Optimal path

Displaying the optimal path was the most crucial feature when enhancing the user's understanding, listed as the most apparent feature by interviewees in the pre-study. The interviewees wanted to view not only the optimal path based on the prioritization of quality attributes but also the optimal paths unique to each attribute. The paths for each optimal path have a different color to easier distinguish which attributes, or cost function, it is currently displaying. Examples of this feature can be seen in Figure 5.3 & 5.4.

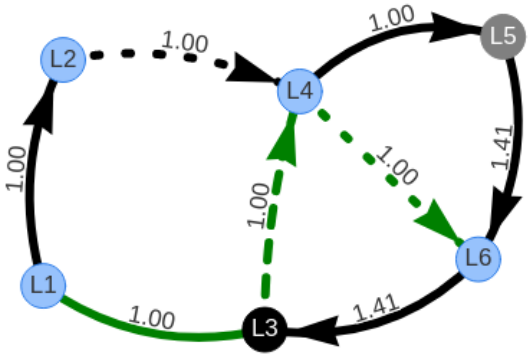

Figure 5.3: Lowest *travel time* cost

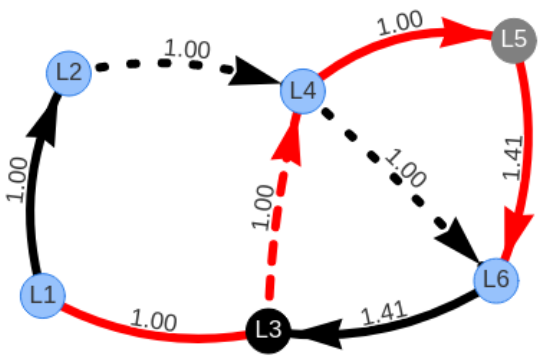

Figure 5.4: Lowest *collision* cost

5.2 Explanatory Features

After implementing the tool's core functionality, it became necessary to implement features explaining why a specific path was chosen. A list of the chosen features can be seen below.

1. Display multiple paths at once.
2. Mouseover tooltip information.
3. List costs of each optimal path.
4. Text explanation of why the optimal path was chosen.
5. List which quality attributes have the same optimal path.
6. List important nodes.

These features were chosen from the remaining features elicited in the pre-study. The general guidelines for which features were chosen came down to two criteria: how impactful the feature is and how much development time is required to implement the feature. With an infinite amount of time, the chosen set of features would likely be different, adding plenty of UI/UX-related features.

5.2.1 Visual features

Displaying multiple optimal paths at once was a highly requested feature in our pre-study. This feature aids users in comparing different optimal paths, leading to a better understanding since it facilitates a comparative analysis between alternative paths.
There is no limit to the amount of different optimal paths to show concurrently. However, when multiple paths overlap, it can become a visual clutter. Therefore other features, such as tooltips, can prove useful in distinguishing overlapping paths.
The mouseover tooltip feature was elicited to combat the time constraint interviewees felt during their tasks. The tooltip displays information about the node or edge's cost and which optimal paths overlap in an edge. This feature provides an instant overview, removing the need for constant reference to a separate legend for cost details.

Figure 5.5 shows an example of these two features. This figure simultaneously displays the optimal path for *intrusiveness* and *collision*, and a tooltip from hovering node *L3*.

5.2.2 Text features

In our findings, we recognized the importance of text-based feedback along with visual cues, especially to mitigate the increased complexity of larger graphs.
A list of costs was implemented at the request of a few interviewees in order to easily compare the costs between paths. The list color-codes the costs associated with each optimal path for clear identification, as seen in Figure 5.6. A few costs in the list are in highlighted bold, implying that it represents the lowest cost for a specific

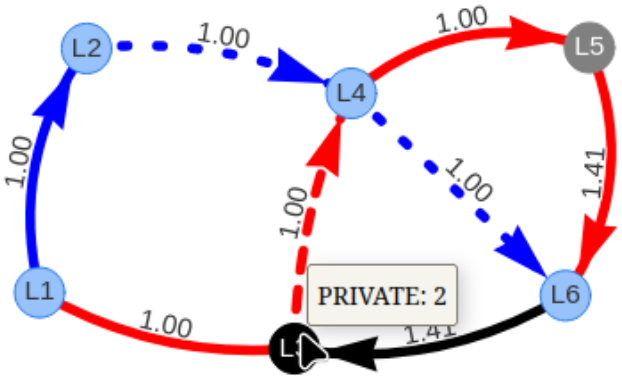

Figure 5.5: Multiple optimal paths and tooltip

quality attribute. This feature aims to mainly combat the complexity interviewees felt during their tasks, removing the need to manually calculate each path's cost. The list color-codes the costs associated with each optimal path for clear identification, as seen in Figure 5.6.

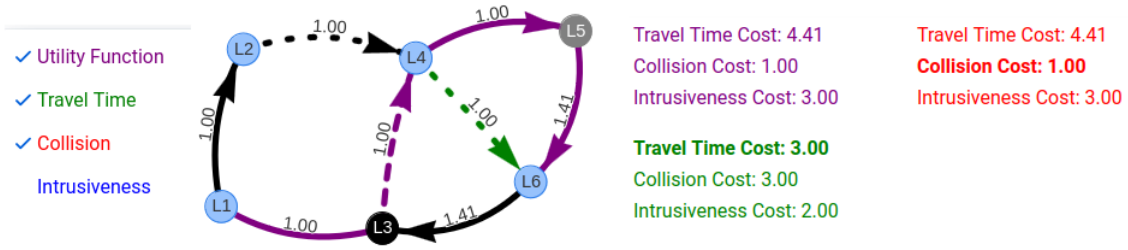

Figure 5.6: Color coded quality attributes ($w_t = 0.8, w_c = 0.2, w_i = 0$)

Travel Time is the highest weight. However, the optimal path for these weights is not the fastest path. The optimal path is 3.00 times as safe and 1.50 times as intrusive, while only being 0.68 times as fast.

The optimal path is the same for the following attributes: [[Collision, Utility Function]]

Important nodes in the graph are: [L4]. These nodes acts like hubs, where the optimal paths for the chosen attributes either diverge or converge.

Figure 5.7: Text explanations for the paths in Figure 5.6

In addition to the cost list, we incorporated sentence-based feedback from the system to clarify visual information. Three features were elicited to achieve this:

1. Descriptive Text: This feature explains why a specific path was chosen. As seen in Figure 5.7, this example explains that even though *travel time* was the highest weight, it chose to optimize a path for other quality attributes because the difference in cost for those attributes was larger than the difference in *travel time*. Due to development time constraints, the current version of this feature is limited to describing why the *cost function* is chosen in contrast to the highest weight. This feature would greatly benefit from the ability to

choose which two optimal paths to compare, resulting in further simplifying the cost comparison.

2. Equal Paths: This features states which quality attributes have the same optimal path. This feature simplifies the identification of identical paths and saves the user's time, especially in larger graphs.
3. Important Nodes: This feature lists important nodes in the graph. These nodes act like hubs, where the optimal paths for the chosen attributes diverge or converge. The features help reduce the complexity of larger graphs by segmenting them into smaller, crucial segments, enabling users to focus on a smaller portion of the graph. The start and end nodes are always deemed important, and it is therefore redundant to list them in this feature.

6
Findings

This section presents the findings from both our iterations. Section 6.1 first presents three key challenges perceived in understanding how weighted quality attributes affect a self-adaptive system's decision-making and then presents potential solutions to mitigate these challenges. Section 6.2 presents to what extent our design artifact can mitigate these perceived challenges.

6.1 Perceived challenges and potential solutions

This section presents the findings from the first set of interviews. Figure 6.1 contains the main themes from analyzing the first set of interviews which focused on identifying perceived problems (**RQ1**) and potential solutions (**RQ2**).
From **RQ1**, we could identify the three main problems: *Complexity*, *Math*, and *Time constraint*. We also acknowledged participants *misunderstandings* here, which often stemmed from overconfidence in their ability.
From **RQ2**, multiple potential solutions to the interviewees' perceived challenges were elicited. The solutions are divided into *Visual* and *Text* solutions. The two themes *Contrastive Explanation* and *Non-numeric input/output* fit in both *visual* and *text*, as they are not standalone solutions and require other solutions to combine with.
The arrows between themes in Figure 6.1 are their relations. E.g., the visual solutions were identified as a reasonable explanation by the participants who struggled with mathematics. Only the most prominent relations are displayed since adding all would clutter the figure extensively. The themes are explained more thoroughly in the subsections below.

6.1.1 RQ1 - Understanding the Prioritization

The codes below are related to the participants' problem with understanding how weighted quality attributes affect a self-adaptive system's decision-making.

6.1.1.1 Math

Math was a common problem for the participants (Interviewees 1,2,4-7). To find the optimal path, they had to calculate the cost for multiple quality attributes for each path and then calculate the cost function for that path.
The participants differed in what they found challenging about the math. Some,

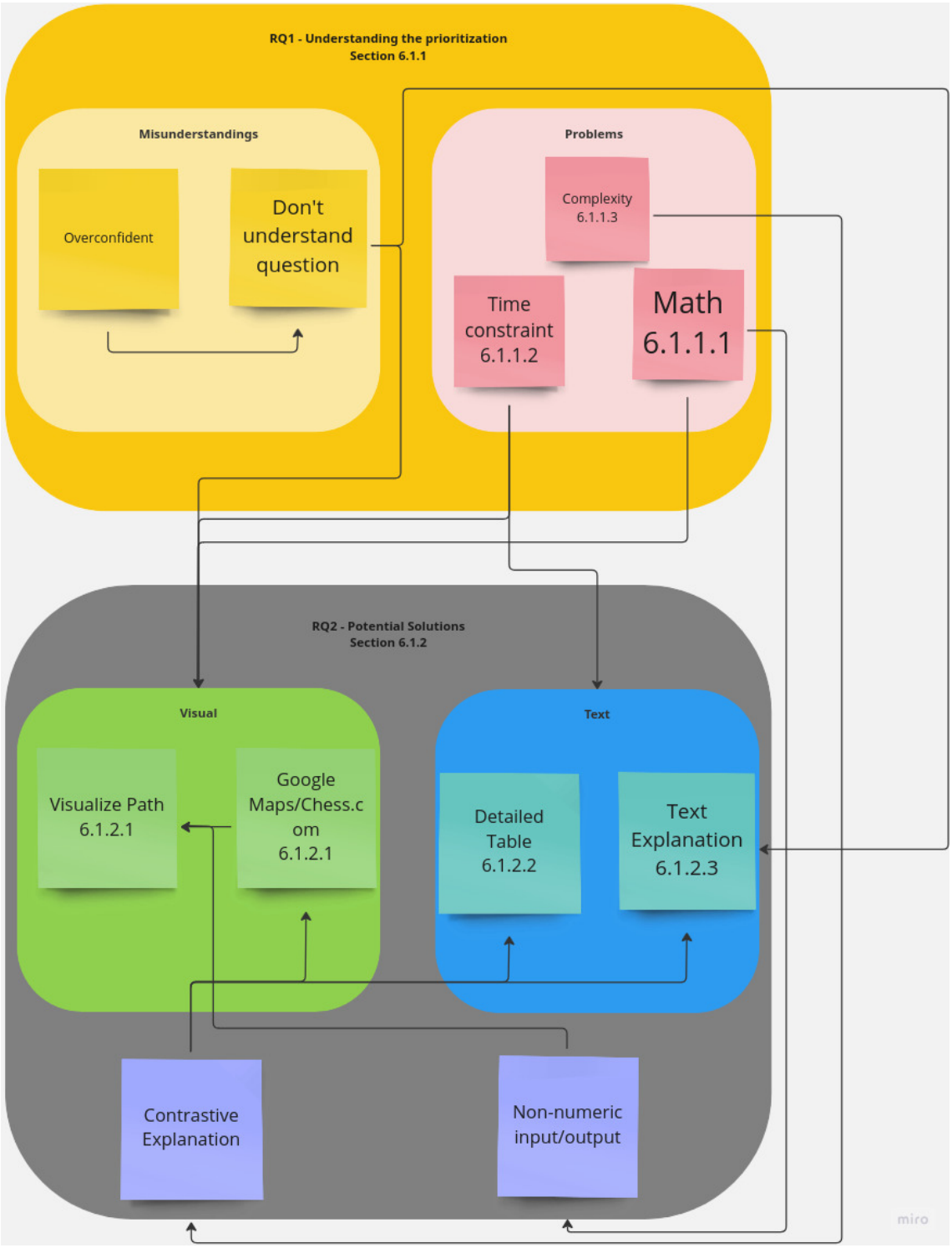

Figure 6.1: All themes and their main relationships from the first set of interviews. Notes contain the section covering it in detail

mainly the ones that didn't have a university degree consisting of much math, struggled with understanding the cost function itself,

> *"It was difficult to understand what the cost functions would yield*

when the weights were different." (Interviewee 2)

The participants that didn't have a problem understanding the cost function were still found to be prone to miscalculate due to:

> *"That the local cost functions also differ. Have to calculate them in multiple steps."* (Interviewee 5)

6.1.1.2 Time Constraint

Interviewees 1,5-7 stated that the time constraint hindered them from accurately solving the tasks. Each interview lasted 30-45 minutes, and the participants had 45 seconds to calculate the optimal path for each task. The tasks involved a very simple graph; hence the short time limit in solving the tasks to make the scenario more realistic.

> *"Having a time-constrained task doesn't give time to do the math for an exact answer, and when the values are fairly similar it makes it essentially a guessing game"* (Interviewee 1)

6.1.1.3 Complexity

The one problem everyone identified when asked if they could solve similar tasks for larger graphs, was how calculating the optimal path would rapidly grow too complex.

> *"It feels like the complexity grows exponentially, both when adding more nodes/paths and quality attributes"* (Interviewee 5)

> *"I think I would need to attempt to write a program to solve it for me."* (Interviewee 6)

6.1.2 RQ2 - Potential Solutions

The codes below are related to participants' ideas for potential solutions to the problems defined above.

6.1.2.1 Visualize Path

Every participant took for granted that it was necessary to visualize the optimal path with the current weights.
Several participants also wanted to show alternative paths that are optimal for different quality attributes:

> *"It would be nice to have visualization, where I can see all the nodes and edges and then color coded lines that shows the best options, e.g.,*

> *"purple" would be the cheapest time-wise, blue would be best collision wise etc."* (Interviewee 2)

> *"I would like it to be somewhat like Google Maps, where it is possible to see multiple paths. The paths should have info about in what aspect they are the optimal path."* (Interviewee 7)

> *"Highlighting each possible decision and its cost from a selected node, like Chess.com."* (Interviewee 4)

These are clear examples of contrastive explanations, where they want to compare the current and alternative optimal paths to gain a deeper understanding of the self-adaptive system's decision-making.

6.1.2.2 Detailed Table

A few participants (Interviewees 1,2,5) wanted a detailed table where the entries would contain different weights, the corresponding optimal path, and the value of the cost function.

> *"The tool should be able to provide exact details with some sort of table listing all calculated routes and their final score, so that I could compare the top 3,5,10 routes weighted."* (Interviewee 1)

This is another example of users wanting some contrastive explanation, where they want to see the optimal path for different weights and compare them against each other.

6.1.2.3 Text Explanation

Interviewees 1,4,5,7 wanted some text explanation of why a specific path was chosen:

> *"I think generating one or two sentences that justify the decision is enough to get a grasp on its decision"* (Interviewee 1)

They considered this a way to demystify the cost function, which many considered a problem.

6.2 RQ3 - Evaluating the artifact

This section presents the findings from the second set of interviews, which focused on evaluating the artifact. The varying features of the artifact were identified as helpful in mitigating the perceived problems identified in the pre-study. For example, participants could not give an accurate or satisfactory explanation of why a specific path was chosen when completing a task without the aid of the tool, except for the case when only one quality attribute was prioritized. However, participants could give increasingly better explanations after each task they utilized the tool, even in cases when they initially didn't know they had understood the decisions

themselves.

The artifact's features are ranked by the helpfulness in mitigating the perceived challenges with understanding the prioritization of quality attributes' effect on self-adaptive systems (**RQ3**). The findings are presented in Likert scale format, followed by citations from interviewees explaining their opinion of a feature.

6.2.1 Math

Text-related features and displaying multiple optimal paths were heavily favored when participants were asked which features had the highest impact in demystifying the perceived math challenge, as shown in Figure 6.2 and related statements below.

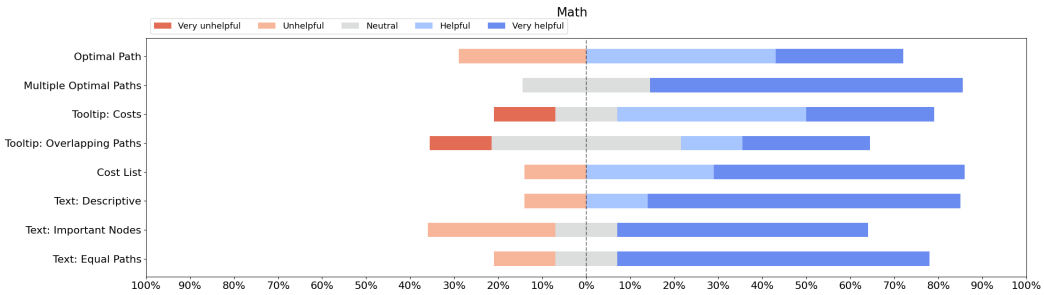

Figure 6.2: Likert scale measuring each feature's helpfulness in demystifying the perceived math challenge

Participants felt that comparing multiple optimal paths significantly reduced the number of times they needed to calculate the tedious cost function.

> *"Displaying multiple paths was really nice to quickly get an overview, instead of having to calculate them by hand."* (Interviewee 2)

The text features were especially helpful for participants with limited knowledge of graph theory.

> *"Calculating shortest-path algorithms is a bit alien for me. So I really like that there are plain text alternatives."* (Interviewee 5)

> *"The cost list feature provided the necessary information of each path to easily compare them against each other."* (Interviewee 9)

6.2.2 Time Constraint

The visual features and the equal paths text feature were overwhelmingly positive in battling the participants' time constraint, as showcased in Figure 6.3.
The visual features provided a quick and simple explanation of what the outcome was, removing the need for a separate legend.

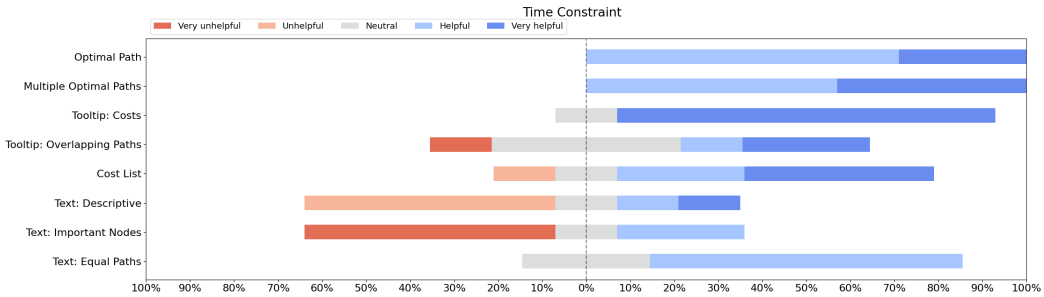

Figure 6.3: Likert scale measuring each feature's helpfulness in battling the perceived time constraint

> *"It was quick and easy to find the optimal path, and I could then use the other features to understand why it was chosen."* (Interviewee 7)

> *"The costs in the tooltip saved me a lot of time instead of looking back and forth in a legend."* (Interviewee 10)

Participants found the equal paths text feature to be time-saving due to them not having to find all paths themselves, and instead only the ones that differentiated from each other. Furthermore, if multiple optimal paths were equal, participants often considered these paths superior due to their multi-purpose.

> *"It was nice to see a list of which paths were equal, instead of looking through the whole graph and find each individual one."* (Interviewee 1)

6.2.3 Complexity

Figure 6.4 showcases that in the eyes of the participants, four features were outstanding in reducing the perceived complexity: Optimal Path, Multiple Optimal Paths, Cost List, and Descriptive Text.

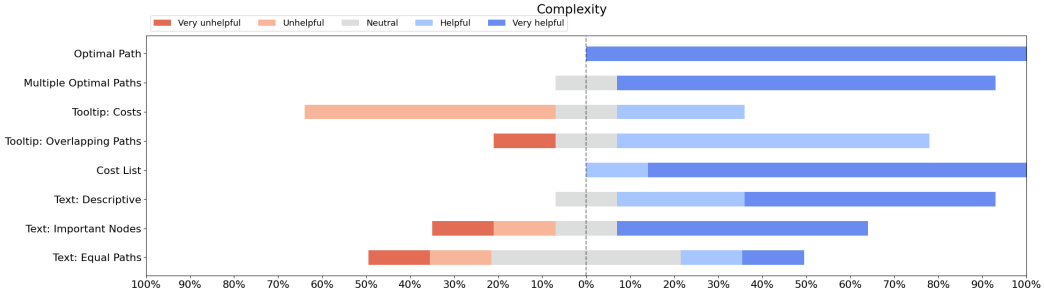

Figure 6.4: Likert scale measuring each feature's helpfulness in reducing the perceived complexity

The participants deemed it crucial to display the optimal path in large graphs, as they couldn't see themselves calculating it by hand, no matter the time limit.

> *"There is no way I could ever find the optimal path in the large maps by myself, let alone multiple optimal paths."* (Interviewee 1)

The participants found the cost list and descriptive text feature incredibly useful in reducing the complexity due to them providing an equal amount of text no matter how large and complex the graph grows.

> *"The graph gets larger, but the descriptive text explanation remains concise."* (Interviewee 9)

> *"I found the cost list super useful because it allows me to easily compare the costs no matter how complex the paths are."* (Interviewee 8)

6.2.4 General Perception & Favorite Features

The participants' general perception of the tool was positive, as seen in Figure 6.5.

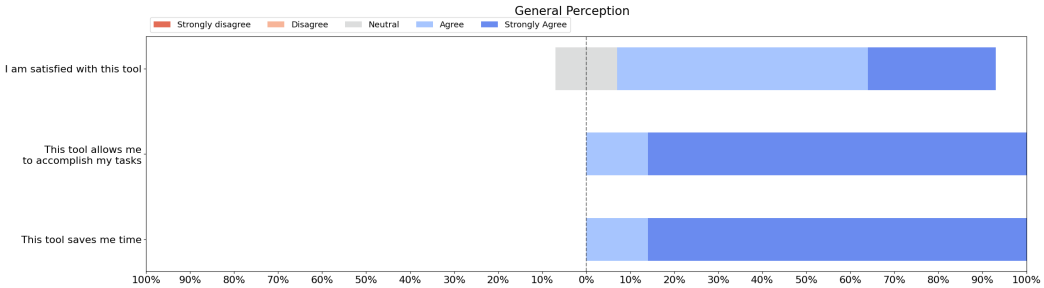

Figure 6.5: Likert scale measuring participants' general perception of the tool

Participants found the tool to significantly save time when trying to understand the effect of quality attribute prioritization on generated plans. However, they believe the tool could benefit from being more intuitive.

> *"The tool helped me complete my tasks. But I wish it got a UX/UI overhaul since some features were a bit hard to grasp."* (Interviewee 7)

When asked which features were their favorites, every single participant answered that the cost list was among their favorites:

> *"The list of costs - The way it explains why the "robot" took a specific route and the details around it."* (Interviewee 8)

> *"Display cost for each path, good with color differentiation and clear language and info."* (Interviewee 7)

> *"The color-themed path costs, easiest way to compare them. And displaying multiple optimal paths enhanced the cost listing feature by visualizing the costs."* (Interviewee 10)

Surprisingly few participants listed displaying optimal paths as one of their favorite features. When asked why, they stated that:

> *"Displaying the optimal path is such an obvious feature that I simply didn't think of mentioning it."* (Interviewee 1)

7
Discussion

RQ1 sought to identify stakeholders' perceived challenges in understanding how the prioritization of quality attributes affects the behavior of self-adaptive systems. Our findings indicated that stakeholders' major challenges are: *time constraint*, *math*, and *complexity*.
Without the aid of a tool, stakeholders are required to calculate cost functions for complex graphs within an arbitrary time limit. We believe these challenges overload stakeholders with information and stress, preventing them from understanding self-adaptive systems behavior. This is supported by Reynolds et al.'s [19] claim that self-adaptive systems are seen as black boxes, and humans can't justify these decisions of the systems without aid.
Notably, these challenges were perceived in the specific context of shortest-path automated planning for self-adaptive systems in robotics. They are not necessarily the top three challenges perceived in other types of self-adaptive systems. However, they are most likely still highly relevant in other types of self-adaptive systems as it is essential to calculate the cost or utility functions for any self-adaptive system within an arbitrary time limit.
However, the complexity of calculating the cost function for shortest-path problems is most likely higher than many other types of self-adaptive systems. It might not be equally dominant of a challenge in other types of systems.
The quality attributes used in our artifact were easy to understand for nearly every participant of our study. This might not be the case for other types of systems that uses more obscure quality attributes, which could lead to challenges in understanding the meaning of the quality attributes themselves.

RQ2 explored potential solutions that could improve stakeholders' understanding of how different prioritization of quality attributes impact the behavior of self-adaptive systems. Our research led to developing a tool that uses a combination of visual and text features to first provide a descriptive explanation answering *what* happens, and then a contrastive explanation of *why* the system behaves in that way.
This mixed-method approach ensures that stakeholders have a clear and nuanced understanding of how their prioritization decisions affect system behavior. The *what* and *why* approach can also explain all types of self-adaptive systems.
The text features, except important nodes, are generalizable to all forms of self-adaptive systems, as they compare quality attributes and optimal outputs, which are essential in all self-adaptive systems.
The visual features in the developed artifact are specific to travel path types of self-adaptive systems, and cannot be generalized to all types of systems. However,

other types of self-adaptive systems could benefit from explanations of similar visual features, where the system's decision-making can be visualized in a decision graph or tree.

RQ3 evaluated to which extent these identified challenges could be reduced. Our findings suggest that using the developed tool can significantly mitigate the challenges stakeholders perceive, especially the complexity.
The visual features had the most significant effect on battling the perceived time constraint. For example, the tool can display the optimal path in the blink of an eye, compared to manually calculating it. This lets stakeholders quickly understand *what* happens, and they could focus on understanding *why* that specific behavior occurred.
It was more challenging to find which features had the most significant impact on demystifying the math, since they all had similar Likert scale ratings. However, the thematic analysis showcased a favor for text features since they provided an explanation without requiring any calculation from the user. The analysis also showed that displaying multiple paths was still necessary because the text needed to be anchored to something more tangible.
Four features were outstanding in reducing the perceived complexity: Optimal Path, Multiple Optimal Paths, Cost List, and Descriptive Text. These features were expected favorites. Both the cost list and descriptive text provide equally concise explanations of *why* a decision was made, no matter the complexity of a graph, and the optimal paths are necessary to connect the explanation to *what* happened.
However, to our surprise, the important nodes feature wasn't considered a top feature. This may be because of poor implementation of the feature, as improvements for it was requested:

> *"Highlight the important nodes in the graph. They were a good indicator of what nodes should be focused on, but was a bit annoying looking at the list and then finding the node in the graph."* (Interviewee 10)

The successful use of contrastive explanation in the tool confirms both Sukkerd et al.'s [22] and Prabhushankar et al.'s [27] claim that contrastive explanation is helpful in aiding humans better understand complex systems.

7.1 Lessons learned

Stakeholders perceive challenges that overload them with information and stress, preventing them from understanding self-adaptive systems behavior. The perceived challenges can vary between different types of self-adaptive systems, and it is necessary to recognize stakeholders' perceived challenges for a specific self-adaptive system.
Stakeholders require tool support to mitigate their perceived challenges and better understand how the prioritization of quality attributes affects the self-adaptive system. We believe that it's necessary to give a contrast between a system's automated

plans, as how good a plan is can only be understood by comparing it against other plans. The mix-method approach of a descriptive explanation answering *what* is the optimal solution, and then a contrastive explanation of *why* that specific solution was deemed optimal proved a great success in mitigating all perceived problems in our study. This approach is generalizable and fit to explain all kinds of self-adaptive systems, even though the specific features explaining *what* and *why* may vary for each self-adaptive system. The contrast should focus on comparing the results of different plans, and their respective quality attribute cost.

Visual features are great at explaining *what* plan is deemed optimal. Visual features such as decision graphs are harder to generalize and require more development time compared to text-based features.

Text-based features are great at explaining *why* a plan was deemed optimal. They are also highly generalizable for different types of self-adaptive systems. One tool could potentially generate text explanations for all kinds of self-adaptive systems.

7.2 Validity

Threats to internal validity. The thematic analysis process is time-consuming, which can be a barrier for this study with limited time or resources. Potential researcher bias or misunderstandings while coding text in the thematic analysis may affect the results due to incorrect conclusions. This was mitigated by asking participants of the study to complete a questionnaire, which introduces data triangulation.

Threat to external validity. Stakeholders have different experience levels of eliciting quality attribute priorities for self-adaptive systems. A different set of participants involved in the study may lead to a different conclusion. To mitigate this, participants with a mix of experience was chosen. The external validity is also threatened by the study's focus on a specific problem domain, shortest-path self-adaptive systems in robotics, with a fixed cost function and fixed set of quality attributes.

Threats to construct validity. The participants in our study might not have the same interpretation of words such as quality attribute, self-adaptive system, generated plan, etc. This could lead to outlying results in the thematic analysis. It was therefore necessary to mitigate it by spending a few minutes with each participant to share a common terminology.

Threats to reliability. The reliability of this study may be influenced by our interpretations. This may have affected our insights and conclusions drawn from the data. To mitigate this threat, we tried to clearly describe our methods and keep a transparent chain of evidence throughout the thesis. For example, the interview guides and questionnaire answers were made public for both the pre-study[1][2] and

[1]Pre-study Interview Guide: tinyurl.com/cwza7nd

[2]Pre-study Questionnaire Answers: tinyurl.com/37vwubt3

the evaluation[3][4].

7.3 Future Work

The artifact in this thesis is only a prototype and would benefit from further development. Participants of the think-aloud study were asked which features needed further improvements, and what new functionality they desired in the tool. Most participants felt that the current features were enough, and wanted improvements of current features instead. Below is a set of potential improvements elicited from participants' suggestions:

1. Set coordinates for the graph: The framework used to display the graph is a version of vis-Network[5] ported to Java, which lacks the functionality to set coordinates for the nodes. Migrating the tool to JavaScript would allow for this improvement and access to many other features not available in the ported Java version.
2. Highlight important nodes: The important nodes are currently listed in text format. However, the participants found it tedious to look at the list of nodes, and then search for the nodes inside the graph. Highlighting the nodes visually in the graph would mitigate this problem.
3. Displaying cost function value: Add the cost function value of each optimal path to the list of costs. This feature would allow users to more easily determine how much more the system favors one path over another.

Alongside these improvements, the tool needs a UX/UI overhaul. The tool would greatly benefit from being more intuitive and pleasing to the eye. This could spark collaboration between the Interaction Design and Software Engineering fields.

Our tool is specific for robot planning missions, and doesn't currently support other areas of self-adaptive systems. However, while the visual features don't make as much sense in other areas, the text features would most likely still prove helpful in all areas of self-adaptive systems.

[3]Evaluation Interview Guide: tinyurl.com/2nps4dzz
[4]Evaluation Questionnaire Answers: tinyurl.com/bdfadwwj
[5]https://visjs.github.io/vis-network/docs/network/

8

Conclusion

In this thesis, we used design science research to identify, explore solutions for, and mitigate key challenges stakeholders perceive in understanding self-adaptive systems, specifically the impact of quality attribute prioritization on system behavior.
We found that these challenges overload stakeholders with information and stress, preventing them from understanding self-adaptive systems behavior.
Our research led to the development of a tool that uses a combination of visual and text features to first provide a descriptive explanation answering *what* happens, and then a contrastive explanation of *why* the system behaves in that way.
The tool was evaluated in a think-aloud study, where the participants had to complete tasks with and without the tool. Our findings suggest that using the developed tool can significantly mitigate the challenges stakeholders perceive, especially the complexity. Notably helpful features were the ability to concurrently display optimal paths for different quality attributes, and compare them with a list of their attribute costs.
Future work can improve existing features and overhaul the UI to create a more intuitive experience, further increasing stakeholders understanding.

Bibliography

[1] IBM. (2006). An architectural blueprint for autonomic computing: Autonomic Computing White Paper.

[2] Faniyi, F., Lewis, P.R., et al.: Architecting self-aware software systems. In: WICSA'14, pp. 91–94 (2014)

[3] Cheng, S.W., Garlan, D., Schmerl, B.: Architecture-based self-adaptation in the presence of multiple objectives. In: Proceedings of the 2006 International Workshop on Self-Adaptation and Self-Managing Systems (2006). DOI 10.1145/1137677.1137679

[4] Inverardi, P., Mori, M.: A software lifecycle process to support consistent evolutions. In: R. de Lemos (ed.) Self-Adaptive Systems, vol. 7475 LNCS, pp. 239–264. Springer Berlin Heidelberg (2013)

[5] Sawyer, P., Bencomo, N., et al.: Requirements-aware systems: A research agenda for RE for self-adaptive systems. In: RE'10, pp. 95–103 (2010)

[6] Frank D. Macías-Escrivá, Rodolfo Haber, Raul del Toro, Vicente Hernandez, Self-adaptive systems: A survey of current approaches, research challenges and applications, Expert Systems with Applications

[7] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed., Addison-Wesley Professional, 2012.

[8] R. Wieringa, Design Science Methodology for Information Systems and Software Engineering. Springer, 2014.

[9] E. Knauss, "Constructive Master's Thesis Work in Industry: Guidelines for Applying Design Science Research," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), 2021, pp. 110-121, doi: 10.1109/ICSE-SEET52601.2021.00021

[10] Bowers, K.M., Fredericks, E.M., Cheng, B.H.C.: Automated optimization of weighted non-functional objectives in self-adaptive systems. In: T.E. Colanzi, P. McMinn (eds.) Search-Based Software Engineering, pp. 182–197. Springer International Publishing, Cham (2018)

[11] Sara Mahdavi-Hezavehi, Vinicius H.S. Durelli, Danny Weyns, Paris Avgeriou, A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems, Information and Software Technology, (2017)

[12] Sampaio, A.R., Rubin, J., Beschastnikh, I. et al. Improving microservice-based applications with runtime placement adaptation. J Internet Serv Appl 10, 4 (2019). https://doi.org/10.1186/s13174-019-0104-0

[13] Virginia Braun & Victoria Clarke (2019) Reflecting on reflexive thematic analysis, Qualitative Research in Sport, Exercise and Health, 11:4, 589-597, DOI: 10.1080/2159676X.2019.1628806

[14] Flick, U. (2018). The sage handbook of qualitative data collection. SAGE Publications Ltd, https://doi.org/10.4135/9781526416070

[15] Guest, G., MacQueen, K. M., & Namey, E. E. (2012). Applied thematic analysis. SAGE Publications, Inc., https://doi.org/10.4135/9781483384436

[16] Robert R. Hoffman and Shane T. Mueller and Gary Klein and Jordan Litman (2019). Metrics for Explainable AI: Challenges and Prospects. arXiv:1812.04608

[17] N. Li, J. Camara, D. Garlan, B. Schmerl, Reasoning about when to provide explanation for human-involved self-adaptive systems, in: Proceedings of the IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2020, pp. 195–204. doi: 10.1109/ACSOS49614.2020.00042

[18] Charters, E. (2003). The Use of Think-aloud Methods in Qualitative Research An Introduction to Think-aloud Methods. Brock Education Journal, 12(2). https://doi.org/10.26522/brocked.v12i2.38

[19] O. Reynolds, A. García-Domínguez, N. Bencomo, Automated provenance graphs for models@run.time, in: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS), 2020, pp. 1–10.

[20] Hellström, T. & Bensch, S. (2018). Understandable robots - What, Why, and How. Paladyn, Journal of Behavioral Robotics, 9(1), 110-123. https://doi.org/10.1515/pjbr-2018-0009

[21] Sukkerd, R. (2017). Improving transparency and understandability of multi-objective probabilistic planning (Doctoral dissertation, The Open University).

[22] R. Sukkerd, R. Simmons and D. Garlan, "Tradeoff-Focused Contrastive Explanation for MDP Planning," 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Naples, Italy, 2020, pp. 1041-1048, doi: 10.1109/RO-MAN47096.2020.9223614.

[23] T. Miller, “Explanation in Artificial Intelligence: Insights From the Social Sciences,” Artificial Intelligence.

[24] Wohlrab, R., Garlan, D. A negotiation support system for defining utility functions for multi-stakeholder self-adaptive systems. Requirements Eng 28, 3–22 (2023). https://doi.org/10.1007/s00766-021-00368-y

[25] Wohlrab, Rebekka and Meira-Góes, Rômulo and Vierhauser, Michael. Run-Time Adaptation of Quality Attributes for Automated Planning. Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2022)

[26] Saaty, R.: The analytic hierarchy process—what it is and how it is used. Mathematical Modelling 9(3), 161–176 (1987)

[27] M. Prabhushankar, G. Kwon, D. Temel and G. AlRegib, "Contrastive Explanations In Neural Networks," 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 2020, pp. 3289-3293, doi: 10.1109/ICIP40778.2020.9190927.

[28] Kephart, J.: Viewing autonomic computing through the lens of embodied artificial intelligence: A self-debate (2021)

[29] Song, H., Barrett, S., Clarke, A., Clarke, S.: Self-adaptation with end-user preferences: Using run-time models and constraint solving. In: MODELS'13 (2013), pp 555–571
[30] Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS quarterly, 28(1), 75-105.
[31] R. J. Wieringa, "Design science as nested problem solving," in 4th Intl. Conf. on Design Science Research in Inf. Sys. and Techn., Philadelphia, 2009, pp. 1–12.