CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

An Industrial Assessment of Software Framework Design:

A case study of a rule-based framework

Master's thesis in Software Engineering

Anton Lunden

Mustafa Hussein

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

An Industrial Assessment of Framework Design: A case study of a rule-based framework

ANTON LUNDEN
MUSTAFA HUSSEIN

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Supervisor: Imed Hammouda, Department of CSE
Advisor: Peter Eriksson, Magnus Strandar, Ericsson AB
Examiner: Jan-Philipp Steghöfer, Department of CSE

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Department of Computer Science and Engineering
Gothenburg, Sweden 2017

Abstract

Software frameworks provide reusable functionality and play an important role in increasing the productivity and maintainability of large software systems. Commonly the functionality and services encapsulated by the framework is accessed through an API, obscuring the inner workings of the framework.
Many companies have rules that their products and software systems are required to follow, these rules can have a wide range of technical depth and are often classified as business rules, guidelines, regulations and policies, some frameworks are driven by these rules. These frameworks are used in a system to implement a set of rules, in addition to providing the usual functionalities of a framework. We call such frameworks rule-based.
These frameworks can be used by companies to enforce rules in a production environment, one of such companies is Ericsson. Ericsson develops applications that must adhere to company-specific design rules, the use of a rule-based framework eases the common implementation of these rules. e.g. by reducing boilerplate code created by several applications implementing the same design rules.
Using Ericssons framework as a basis for the case study the purpose of the study was to identify and demonstrate what values rule-based frameworks can provide for different stakeholders. In addition to this the study provided insight into what factors should drive the design, development and usage of rule-based frameworks.
Data was collected throughout the case study through archival data, interviews and to a lesser extent, surveys. The data was collected from a variety of stakeholders to capture the difference perspectives and experiences. The data was analyzed and modeled using iStar Goal Modeling and e-3 Value Modeling to aid in understanding the ecosystem and values of the framework.

Keywords: Rule-based frameworks, frameworks, design drivers, design rules, goal model, e-3 value model.

Acknowledgements

The authors would like to thank their supervisors Imed Hammouda and Jennifer Horkoff, at Chalmers university of technology for their help and guidance during the study and making it possible. Juho Lindman from the Software Center at lindholmen for his guidance and feedback that helped us to conduct this study. Thanks to the examiner Jan-Philipp Steghöfer for his input and feedback on this study.

The authors would also like to thank Peter Eriksson and Magnus Standar, our supervisors at Ericsson AB and all the participants from the company for their support, expertise and sharing their knowledge with us, without it we wouldn't be able to conduct this study.

Mustafa Hussein & Anton Lunden, Gothenburg, June 2017

Contents

List of Figures

List of Tables

1
Introduction

Software frameworks provide reusable functionality and play an important role in increasing the productivity and maintainability of large software systems [1]. Commonly the functionality and services encapsulated by the framework is accessed through an API, obscuring the inner workings of the framework [1].

Many companies have rules that their products and software systems are required to follow. These rules can have a range of technical depth and are often classified as business rules, guidelines, regulations and policies. Some frameworks are driven by these rules [2], they are used in a system to enforce conformance a set of rules, in addition to providing the usual functionalities of a framework. We call such frameworks rule-based. Rule-based frameworks can be used by companies to enforce rules in a production environment, one of such companies is Ericsson. Ericsson develops applications that must adhere to company-specific rules, the use of a rule-based framework eases the common implementation of these rules. E.g. by reducing boilerplate code created by several applications implementing the same rules.

There have been studies covered in the literature to understand the value and benefits of using software frameworks in different contexts [3][4][5][6][7], and the problems and challenges with developing and using them [5][7]. The literature shows little information regarding rule-based framework specifically. We aim to compare the common benefits and challenges existing in the literature on frameworks in general, as well add anything new that may come from using this specific type of framework. The study will also contribute to be applicability of using goal models [8] and value models [9] to demonstrate these benefits and challenges in an ecosystem.

As for what drives the design and usage of rule-based frameworks, little research was found in the literature. A study was made to investigate the effect of using case-based reasoning to support the rule-based implementation in a framework [10]. By studying this framework, we hope to gain insight about the drivers behind developing and using it.

While the purpose of the framework utilized by Ericsson is to support the rules, is the design driven purely by the rules, or is it driven by application use cases? In the former case, the developers of the framework consider the rule assets, and develop from a bottom-up approach. The second alternative is to consider the use-cases of the application teams using the framework, a top-down approach. The latter may be more practical, but can lead to a framework getting too large and growing out of the original scope. As one of the goals of the framework is to make it simpler for developers to implement the domain specific rules in their application. One question is how well does it achieve this simplicity, the mechanism for information hiding is the API, does it sufficiently hide the knowledge required of rules?

This case study was part of a larger research project conducted in collaboration with Software Center in Lindholmen, Gothenburg for their API Strategy project (APIS). The purpose of this study is to identify and demonstrate what value rule-based frameworks can provide for stakeholders. In addition to this the study will provide insight into what factors should drive the design of rule-based frameworks and how well they can ease a common implementation of rules. Therefore, for the purpose of this study the following questions has been defined:

- **RQ1**: What are the values and challenges of having a rule-based framework?
  - RQ1a What are the values?
  - RQ1b What are the challenges?
- **RQ2**: What drives the development and usage of a rule-based framework?
  - RQ2a what drives the design and development of a rule-based framework?
  - RQ2b what drives and motivates the usage of a rule-based framework?

The outline of the thesis is according to the following structure; In the background section a description of the case study domain is given, including a description of the framework being studied. In addition to a theoretical background is given, explaining the concepts used throughout this thesis. As well as describing existing work that is related to the research questions at hand. Following this is a methodology section describing the methods used for collecting and analyzing data in this case study. Threats to the validity of the study are also addressed in this section. After this a results section present the findings of the case study. The findings are separated according to the research questions. Following this is a discussion section which relates the results to the existing work and analyzes the result. Finally, a conclusion section summarizes the thesis and any implications it has for academia and industry.

2
Background

This chapter describes the background on which the study is based. Firstly, a theoretical background which gives the reader an understanding of the concepts related to the study. Followed by the setting of the study described, which includes domain background and stakeholders related to the study.

2.1 Theoretical Background

This section describes the theoretical background of the study by giving the reader a description of frameworks and design rules. As well as a summary of related work that has been done on the benefits, challenges and design drivers of frameworks.

2.1.1 Frameworks

Frameworks are commonly, but not always, an object-oriented abstract design for a specific application and with an abstract class for the most major components in that application [11]. It can be defined as "A generic application that allows the creation of different applications from an application (sub)domain" [12]. Another definition for a framework is "A system that can be customised, specialised, or extended to provide more specific, more appropriate, or slightly different capabilities" [13].

A framework often consists of reused code taken from other applications which are called "frozen spots" and variable aspects in the framework of the domain application that is called "hot spots". A hot spot allows the implementation of a subsystem or a class either by selecting occurrence of variabilities existing in the framework or programming a class or subsystem to achieve that [12]. This means that a framework design can be classified as black-box, white-box or a combination of both (gray box) [3]. A common way to consider composition when developing frameworks is to maintain a set of APIs that encapsulates the services that the framework provides [14]. When communicating with a framework, the application only needs to know the functions and parameters that should be called, ignoring the inner workings, i.e hot spots and frozen spots, of the framework.

Framework development phase is considered the most effort consuming phase when introducing frameworks [5]. It focuses on the domain which the framework will be part of, where in our case the focus was a common solution for state, fault and alarm handling for the various products at Ericsson. One problem in designing and developing a framework is the domain size, if the domain is too large, it might

be difficult to show the usability of a framework, while a smaller domain might show it in a more efficient way [5].

The framework usage phase or also referred to as the application development phase [5] is where the developers include the core framework or part of the framework in their application. This can happen during the development of new applications or refactoring already existing applications to support the framework. Both cases are applicable to our case at Ericsson. Learning frameworks is often considered difficult and lack of documentation can increase the complexity of it [15]. To be able to use the framework correctly, application developers must know how to use the methods in the framework API to implement specific features as well as the design of the framework [15]. The framework evolution and maintenance phase ensures the changes in the framework or applications are handled correctly and provide support for new and old applications using the framework [5].

2.1.1.1 Rule-Based frameworks

When designing a framework, developers usually identify a need for it, most frameworks purpose is to provide free functionality and reusability for different use cases in the domain. These frameworks can have use cases that requires following a set of specific rules, but enforcing a set of rules is not the framework's main functionality. Rules are considered statements that establishes principles or standards, that aid in guiding or mandating actions. Rule-based frameworks are designed by identifying the need of enforcing a set of rules in the domain as the main functionality. While it's not the primary goal of these frameworks, they can also provide other functionalities such as software reusability and support of domain use cases that are not related to the specific set of rules implemented by the framework.

Rule-based frameworks are driven by the goal of encapsulating or enforcing a set of rules. These rules can typically be classified as business rules, guidelines, regulations and policies. The extent to which the rules drive a framework can vary drastically. Rule-engines are one type of rule-based frameworks, they are driven and created purely for supporting a set of rules. They typically consist of several components, such as a rule base containing all the rules and the data on which the rule-engine operates [16]. Their main usability is the ease of modifying rules separately from any application code, acting as a pluggable component to enforce rules.

Other rule-based frameworks may be driven by more than rules, providing other functionalities or implementing a common behaviour. The rules themselves may also directly influence how applications should be programmed, in which case the rules are supported directly through the interfaces.

2.1.1.1.1 Examples of rule-based frameworks An example of a framework that implement specific rules which can be related to this study is an automated negotiation mechanism in model e-commerce multi-agent system [17]. Rules are organized into a taxonomy: rules for participants admission to negotiations, rules for checking the validity of negotiation proposals, rules for protocol enforcement, rules for updating the negotiation status and informing participants, rules for agreement

formation and rules for controlling the negotiation termination [17]. The main goal of the framework is to create a model system where agents perform functions based on the negotiation rules [17].

Another example is the the oncology protocol management system. Developers of a knowledge base system must ensure that it will give the users an accurate advice or correct solutions to their problems [18]. To verify the system is accurate and reliable, they must ensure that the knowledge base contains all necessary information and verify that the program can apply this information correctly [18]. Rules specify the context in which they apply, in the oncology protocol management system (ONCOCIN) contexts are drugs, chemotherapies, and protocols. A rule which specify the dose of a drug may be specific to the drug alone, or to both the drug and the chemotherapy. The rule checking program described here was developed at the same time that ONCOCIN's knowledge base was being built [18]. The system helped them find missing rules and detect conflicting and redundant rules [18].

Final example of rule based system is a distributed data management system that implement internal consistency constraints directly within the software [19]. When the rules change over time, the software must be rewritten. A way to avoid that in data management system is to have a rule based system [19]. The architecture differentiates between the administrative commands needed to manage the rules, and the rules that invoke data management modules [19]. This generic solution will support rules for data placement, rules for controlling access, rules for presentation, rules for federation, and rules for data ingestion [19].

2.1.1.2 Types of frameworks

2.1.1.2.1 White-box frameworks In white-box frameworks the architecture of the framework is open to the application developers using it. This design knowledge is necessary for a user to adapt the framework to their application [3][11]. Since the mechanism that provides flexibility when implementing the framework is generally limited to inheritance. The user must therefore have knowledge of the architecture to customize the framework for their application needs. The application requires the creation of many new subclasses, and while most of these are simple, their number can make it difficult for the user to learn the architecture well enough [11]. In some cases, you can say learning to use the framework is the same as learning how it's constructed.

2.1.1.2.2 Black-box frameworks In black-box frameworks the architecture is hidden from the developer using it. The knowledge required to adapt it is limited to the description of the frameworks usage and its flexible elements. The mechanism for adaptation is limited to composition [3][11].

2.1.2 Related work

The research has covered the value and challenges of frameworks in general [3][4][5] [6][7], but there is few research on the value and challenges of frameworks that enforce a set of rules (rule-based frameworks).

Frameworks in general offer many benefits including code reusability and reduce time to market for applications [3][4][5][6][7]. Software reuse can help improving the productivity and quality of software systems [4][3]. A framework should take less time and effort to understand and use than developing and using an equivalent software without a framework [4]. This can only be achieved if the framework is easy to understand, a complex design and implementation of frameworks can impact the understandability and usability and is considered a critical issue [4]. The literature shows some issues that are well known for researchers, these issues include maintenance, understandability, change in requirements, verification, testing and debugging [5][7].

Jan Bosch et al. [5] studied the problems related to frameworks and organized them into 4 categories: framework development, usage, composition and maintenance. During the framework development, the framework should cover all relevant concepts in a domain, unlike the application which is concerned with its requirement only [5]. Lack of business models, verifying abstract behaviour and framework release are the identified problems for framework development. During the framework usage, the problems are related to applicability, estimations of the application development and debugging. The framework composition problems could be a mismatch in the architectural styles, overlap with the components and issues with legacy code [5]. As for the framework maintenance, choosing the maintenance strategy and react to business changes can be challenging for the developers of the framework [5]. Maintaining a framework may take different forms, such as adding functionality, removing functionality, and generalization [7]. Fixing errors is not an easy process, since the framework is used in many different application in the same domain [5]. A need to split the framework might be one solution if the error is too complicated to fix and there are many applications depending on the framework [5].

Another problem of framework-based application development is the ability to manage the framework usage phase. The application developers need to study and analyze if their application can be built based on the framework, especially if the framework does not support certain aspects of the application which might not make adapting the framework a possibility [5].

Dirk Riehle mentions in his dissertation "Framework Design: A Role Modeling Approach" [3] some problems with frameworks from different case studies and current practice regarding understanding and using existing frameworks. The four problems mentioned are: class complexity, complementary focus on classes and collaborations, Object collaboration complexity and Difficulties with using a framework [3].

As for the design drivers and the role of use cases and rules, Golding & Rosenbloom studied the possibility of improving rule-based systems through case-based reasoning [10]. Adding case based reasoning to a rule based system improves the coverage of the domain for the framework without affecting the rules [10]. Rules and cases have complementary strengths, rules capture the domain in general while cases are capturing the small gaps and exceptions in a domain [10]. A way to have

both cases and rules part of the system is to convert the cases into rules or rules into cases. But these conversions has its drawbacks since rules can't easily be changed specially if they are covering more domains, it might give an unreliable presentation for both [10]. An experiment was made to evaluate the effect of combining rules and cases in practice [10]. The primarily results showed that the system performed near the level of best commercial systems, the results also showed that the system could not have achieved this depending only on the rules [10].

2.2 Domain Background

The study was conducted at Ericsson, a large telecommunications company with more than 110,000 employees. They offer services, software and infrastructure in information and communications technology (ICT) for telecommunication and networking equipment. The study was conducted in cooperation with the Product Development Unit Baseband and Interconnect department at Ericsson Lindholmen, Gothenburg. The department provided the study with access to the source code of their state, fault and alarm handling (SFA) framework, the framework being studied in this case study, described further in the domain section. As well as documentation and personnel with user and developer knowledge related to SFA.

2.2.1 The Domain

The domain that this study was conducted in consisted of several software components related to RBS (Radio Base Stations). These components should all adhere to company-wide design rules related to state propagation and fault handling. The implementation of these design rules was modularized into a framework starting with one of the software modules in RBS. This framework became the SFA framework and has since been used partially or fully in several other software modules in the RBS.

2.2.1.1 SFA Framework In General

SFA is seen as a rule-based framework by Ericsson based on the definition given in the theoretical background. The purpose of this common state and fault framework is to have a software for supporting the domain design rules that can be developed, verified and delivered independently of the components which the rules apply for. By using a stricter description and implementation of the state and fault handling logic, the results is software that has a more robust behaviour. To achieve this goal, state, fault and alarm framework (SFA) was developed by Ericsson to encapsulate the state and fault mechanism and lessens the amount of boilerplate code for handling state and fault operations between entity objects (EO). It also encapsulates the management of state propagation clients and servers to the EO and handles the calculations of availability status. SFA is also responsible for handling new faults and holds the fault states for the EO, as well as raising and ceasing external faults to the upper layer for the faults that are concluded to be actual faults.

SFA API is an interface provided by the framework and fully implemented and supplied by it. The API acts as a single interface to access the underlying components in SFA. SFA Adapter API functions the same way as an SFA API and is used when a component not using SFA needs to propagate state and fault information to a component using SFA. An instance of the class SfaApi is created for every EO that requires state, fault or alarm handling. Each instance is responsible for connecting to its own dependencies by getting them as string identifiers from the dependency service which encapsulate the dependencies for an EO.

2.2.1.2 SFA Usage

To use SFA to its full potentials, the users need to configure the framework to their specific needs, this is mandatory to set up the framework and get the basic functionalities. The user does so by setting up the timer service, registering the dependency service and the state rules. A timer service is needed to get SFA to work, its configured by providing a timer implementation which should fulfill the interface defined by the TimerApi and that the application has been linked to a library containing the expected timer implementation. A dependency service configures each SFA API instance with its dependent one in both propagation and aggregation direction. State rules are required when an EO has at least one dependency, it evaluate state changes received from dependencies in both propagation and aggregation directions.

2.2.1.3 Fault Handling

Each fault evaluated by SFA needs configuration. The fault must be registered on an SFA instance to raise a fault. Each fault has a unique id for the instance but other faults can have the same id if they were registered on different SFA instances. It is not possible to change a register fault during run-time. When an EO detects a fault, it will call on SFA to evaluate it, the framework will coordinate fault candidates from all its dependencies and if it exists, the localization will pin-point to the correct faulty EO. If the fault is registered with a recovery action, the framework will request a recovery action and after completion the EO will notify the framework if the recovery was successful or not.

2.2.1.4 State Handling

The state handling process in SFA include propagating and aggregating state information and localization to its dependencies as well as calculating new states.

Figure 2.1 shows the state aggregation from a child to a parent and state propagation from a parent to a child with the help of SFA.

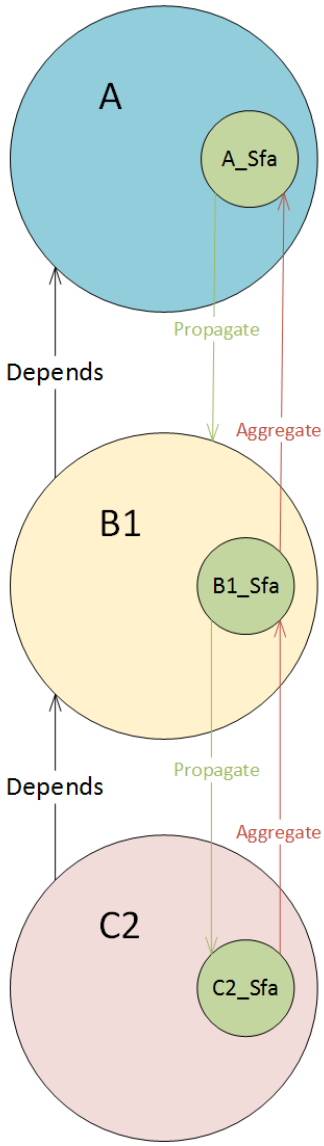

Figure 2.1: State propagation and aggregation between a parent and child using the SFA API interface

2.2.1.5 Alarm Handling

An EO that has been declared faulty should raise an external fault which is reported externally as an alarm and only one alarm at a time can be visible on each EO. Alarm forwarding happens when an EO is unable to report an alarm and delegate the reporting to another EO.

2.2.2 Stakeholders

This section describes the stakeholders that are affected by or affect the results of the study.

2.2.2.1 Users of SFA

Developers and other stakeholders in applications that have implemented and used the framework. They provide the study with necessary data through qualitative interviews.

2.2.2.2 Potential Users of SFA

Developers and other stakeholders in applications that could potentially use the framework, but are not currently doing so. They provide the study with data through qualitative interviews. The data gathered from them can be used find gaps in the needs of users and what's currently supported in the framework

2.2.2.3 Developers of SFA

Developers of the framework itself, these persons may also be users of the framework. They provide the study with necessary data through qualitative interviews.

2.2.2.4 Product Guardians

Product guardians (PG) are responsible for different frameworks and components. They can provide technical knowledge of the applications and framework, and can provide information on which teams have members who are suitable to interview.

2.2.2.5 Managers

The managers are mainly affected by the results of the study. They will be able to use the data to make decisions related to the design/strategy of the framework. The result may also help them show the value and issues in using the framework.

2.2.2.6 Systemizers

Systemizers are responsible for eliciting design rules and making recommendations and decisions on a system level.

2.2.3 Design rules

One definition of a rule is that it's a statement that establishes a principle or standard, serving a norm for guiding or mandating actions. The design rules at Ericsson for this specific domain are defined as: rules that describes principles and define rules for state, fault and alarm handling in the RBS system. These rules shall be followed or well-motivated why they are not followed. These rules do not describe rules or principles related to design faults or program/software errors.

In this study we use the term "design rules" to describe the specific rules their framework support as it is used by Ericsson. Design rules do not describe rules related to the design of applications, the framework or rules that should be followed during the development of either. The design rules are executed at run time and not during the design phase.

The design rules enforced by SFA vary in their scope and some could be classified as a type of business rules, as their purpose is to define constraints and support to behaviour in the system [20][21]. The design rules are created primarily based on customer needs and telecommunications standards.

Business rules usually define some aspects of the software to assert the business structure of the organization and focus on access control issues and usage permission. It can also help the developers to follow the policies of an organization. This is an example of a design rule related to alarm behaviour in the system, which we argue belongs to the categories described above *"It shall be enough for the first line staff support to categorize and make correct first priority of an alarm that is shown in the alarm viewer Thus the specific problem that caused the alarm should have explicit information and the affected functions"*.

Here are some other examples of the type of design rules used that are enforced by the framework:

- When a fault is reported as ceased by the controller, another component must notify the alarm component about it.
- When a component has restarted, that component should be considered fault free until a fault report is generated.
- When a component has a fault that made it completely non functional, it should be taken out of service.
- If a recovery action brings a component back to full functionality, no alarm should be raised.
- Hardware alarms should not be raised unless a hardware test was attempted and failed.
- Equipment alarms should pinpoint to the replaceable hardware.
- Alarms should only be sent when there is a need to inform the operator and there is a relevant action for the operator.
- Alerts should be used only when there are no other options, it should be the last option.
- If a recovery action brings a component back to full functionality, no alarm should be raised.

3
Methods

This chapter describes the research methodology used in conducting this study, the type of data collection methods, the data analysis techniques, the evaluation and finally addressing validity threats to this study.

3.1 Case Study

Case study is a well-known research methodology within software engineering and considered a suitable research methodology in this field where the object is considered a contemporary phenomenon (events in the present that can be observed and studied) [22].

This case study is an exploratory study which is what case study methodologies were originally used for, it aims to investigate, seek new insights and generate ideas and hypothesis [22]. Since our research is conducted in real world settings and requires flexibility, this methodology fits our criteria where no strict boundaries between the studied object and the environment exist.

In a case study, triangulation is an effective way to increase the precision of empirical research [22]. By triangulation we mean that different angles will be taken to collect the data by using different methods and data sources. The data collection methods used for this study to help answering our research questions are Interviews, surveys and archival data.

3.1.1 Interviews

Interviews and specifically semi-structured interviews are considered the most common data source for a qualitative research [23]. Questions are planned beforehand but not necessarily asked in the same order, this will give us the opportunity to steer the interview in the right direction at the same time as asking the relevant questions for the answers we receive. The questions are based on the topic of interest regarding the SFA framework and can be seen in Interview Questions. Both open and close ended questions were asked. Open ended questions being the primary questions to gather additional information and allow the participants to speak freely, share their knowledge and any information we missed. The interviews were recorded after acquiring the permission from the participants to transcribe it and validate their answers for future references.

The participants were selected based on their affiliation with the SFA framework, whether they were users, potential users, developers or software architects with the

help of our advisor at the company. The sample can be classified as a mixture judgement and convenience sampling. One member from teams that used or developed the framework were selected and contacted to schedule a 45 minutes interview. The participants received a brief introduction about the topic of this research to eliminate any misunderstanding or confusion during the interview. The questions were divided in various categories based on the type of participants and our research questions. For the type of participants, a general category for both developers and users of the framework, a user category for users and a developer category for developers. A fourth category was added for the potential users of the framework. Follow up questions were asked when necessary to further explain and understand what the participant meant at that point.

As for questions related to our research questions, a category regarding the advantages, disadvantages and perceived value of SFA to help us answer our RQ1. A second category regarding the domain use cases for their products, the evolution of SFA and design decisions where applicable to help us understand the design drivers and answer our RQ2. A third category regarding the knowledge of design rules in general and during the time of using SFA to help us answer our RQ1 and RQ2.

ID	**Role**
D01	Developer of SFA
U01	Product guardian
U02	Developer
U03	Developer
D02	Developer of SFA
D03	Developer of SFA
PU01	Product Guardian
U04	Product Guardian
PU02	Developer
SA01	System Architect
*U05	Designer
*U06	Product Guardian

Table 3.1: Participants interviewed:
D: Developer, U: User, SA: Software architect, PU: Potential user. (* Answered the survey, as they could not be interviewed)

3.1.2 Surveys

A survey was designed for users and other candidates who could not be reached for interviews or were otherwise unavailable for conducting interviews throughout the case study. The survey consisted of the same questions as the interview, to act as an alternative method. The survey was answered by two participants in total, and their data was analyzed together with the interview data.

3.1.3 Archival data

Archival data retrieved from the case company Ericsson includes documents from different development phases regarding the case study. These documents gave us a better understanding of the framework and helped us formulate relevant questions for the interviews.

Early design documents during the development of SFA will be used and compared to the current framework to understand the evolution and design decisions for it, this will help us answer our RQ2. Documents such as the SFA framework guidelines and it's use cases will help us understand the framework in general as well as the benefits and drawbacks of using SFA, this will help us understand the perceived value of having such framework and be able to answer our RQ1. Other documents such as the RBS architecture and the company design rules for state propagation, fault handling and alarm handling will help us understand the products using SFA and get a better understanding about the design rules and what effects they have on SFA.

3.1.4 Data analysis

During case study research, it is common to collect large amounts of qualitative data. This is especially the case when interviews are used as a primary method for data collection. To analyze qualitative data, there are certain preferred techniques [24]. To properly be able to draw conclusions based on the analyzed data, a clear chain of evidence should exist [25]. Several approaches exist to qualitative analysis, usually having coding and pattern finding in common [26]. The method for data analysis chosen in this case study was thematic analysis [27]. The reasons for choosing this method is related to its flexibility and effectiveness at extracting important information from data [27].

As thematic analysis does not require the detailed theoretical and technological knowledge of approaches such as grounded theory and DA, it can offer a more accessible form of analysis, particularly for those early in a qualitative research career. The main purpose of the method is to identify themes and patterns from the data. The method consists of six steps, a description of what we did in each step is given below.

1. Step 1 - Familiarize yourself with the data:
   In this step we familiarized ourselves with the data by looking at the transcript and notes, trying to identify patterns between the participants. At this stage we created a written summary of the relevant and interesting parts of each interview.
2. Step 2 Generating initial codes
   In this step we started coding the data. Statements in the interviews were assigned to patterns which they related to. An example of this would be a developer saying the framework was too large, this would be assigned to a pattern of statements relating to the size of the framework.
3. Step 3 - Searching for themes:
   After coding the data and identifying the patterns we started separating them into larger themes. The themes were themselves organized into initial cate-

gories based on our research question. In e.g the challenges category, we would group patterns in potential themes related to challenges.

4. Step 4 - Reviewing themes

   In this step we evaluated the credibility of the themes we had found, seeing if the data extracts assigned to each theme made sense. After this we we went over the data set again to add see we missed something relevant to the themes in the initial steps.

5. Step 5 - Defining and naming themes:

   In this step we refined the themes. Identifying fitting names for the themes and making sure the scope of the themes were not too diverse or complex.

6. Step 6 - Producing the report

   In this step we did a write up of each theme, describing and explaining the data extracts in them. The result of this is presented in section 4.

3.1.5 iStar Goal Modelling

Goal models is a technique found in requirement engineering. It models the goals of stakeholders in a system. The goals are defined as objectives which the system should achieve, through different tasks and actors [8]. The model shows a hierarchy of goals which can trace the high-level goals of the system to low-level elements of it. Goal models can help companies address concerns such as why the system is needed, i.e. displaying how the system achieves the goals it intends to achieve [28].

Creating goal models based on archival data and interviews can help answer the research questions related to the value and challenges of SFA. The model maps the stakeholder goals to the actors in the system, these dependencies can show which objectives SFA addresses. The goal model created during this thesis will be using the iStar notation [29]. The model will therefore as a supplementary to the thematic analysis, visualizing the values and challenges and displaying which actors are affected and/or affect them.

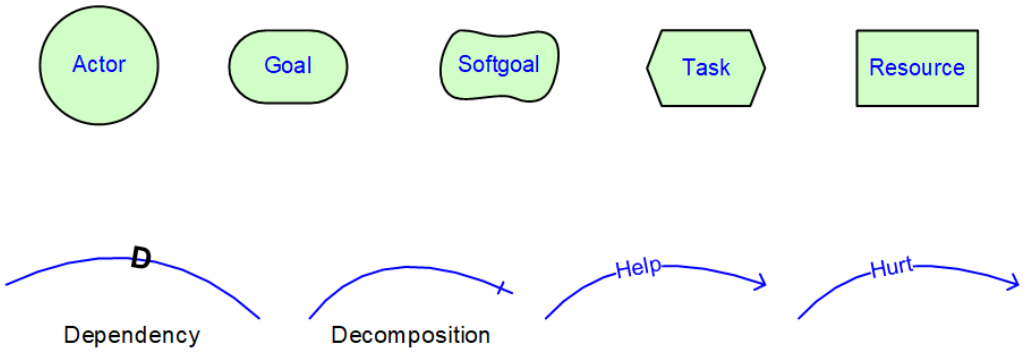

Figure 3.1: A simple legend for the goal model used in this study

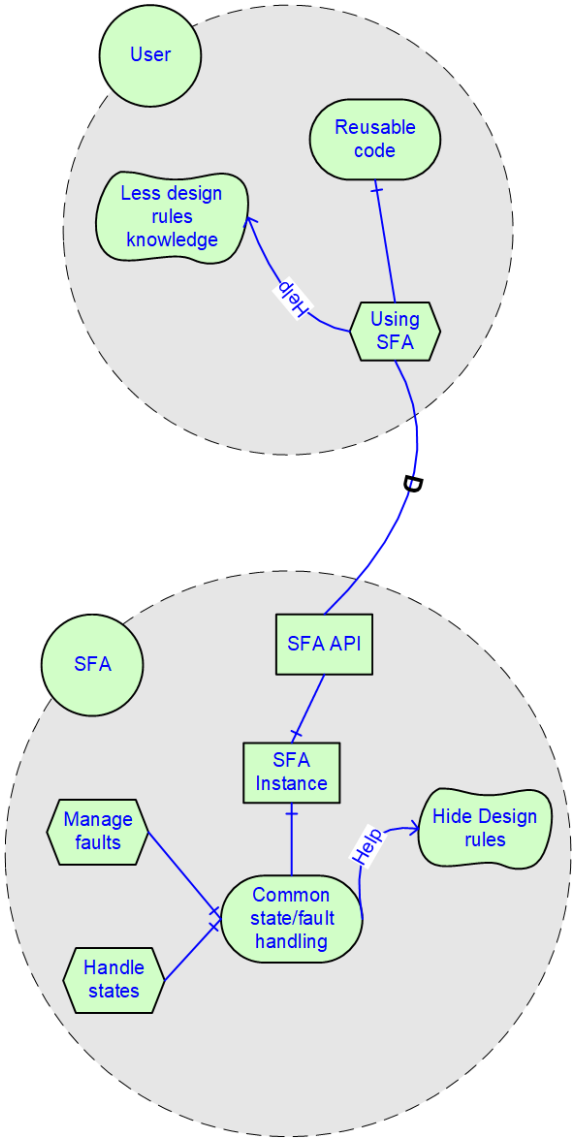

Figure 3.2: An example of a goal model in the context of this study

Figure 3.1 shows a simple legend to explain the notations used in our goal model. Figure 3.2 shows an example of a goal model where 2 actors are represented, the users and the framework. The users have a goal of having reusable code and a soft goal of having less design rules knowledge while using the framework. These two goals can be achieved by the task "using SFA" which has a dependency link to the second actor (SFA) through the "SFA API".

3.1.5.1 Creation of the model

The goal model was created by using the data gathered from interviews and archival data. From the interviews summary, we identified the different actors, goals, soft goals and tasks. Goals for the users and developers were identified based on the interview questions about the advantages and value of using the framework. All the answers related to that were mapped into goals and soft goals that the user has for

using the framework. as for the unachieved goals in the model, they are based on the answers we received regarding the challenges they had with the framework.

The goals for potential users are based on the interviews we conducted, asking them about what they expect to achieve by using the framework, what values they see and what might prevent them from using it. The data was analysed and checked with the other interview and archival data to investigate whether it is an achieved goal or not.

We could not interview people representing the managers and systemizers, but we did interview product guardians and system architects who has knowledge about the goals and interests of these actors, these goals are therefore interpretations of all the interview data collected.

As for the EO and SFA actors, the goals and tasks are identified with the help of archival data provided to us by Ericsson, by reading the SFA guidelines and other documentation during the development of the framework to identify the tasks required to achieve specific goals and the relation between SFA and the other actors. We also used interview data to identify more goals and tasks since we conducted interviews with early and current developers of the framework.

3.1.6 e3-Value Modelling

The e3-value model has been developed to identify the exchange of value objects between different actors. The e3-value model can be defined as "An ontology-based methodology for modeling and designing business models for business networks incorporating concepts from requirements engineering and conceptual modeling" [30]. Value objects does not necessarily need to be a physical good, in our case they usually represent a service, right or a quality attribute.

The basic concepts of this model are actors, value objects which belong to at least one actor, value ports with value interfaces which are used by actors to exchange values and value activities are the activities the actor must do in order to deliver the value object [30][9].

We chose this modelling technique due to its simplicity in presenting value exchange between different stakeholders. This model can help Ericsson understand the value objects for SFA that might have been overlooked by the managers, developers, users or potential users. This model will give us a better understanding of the perceived value of having a framework enforcing rules and answer our RQ1a.

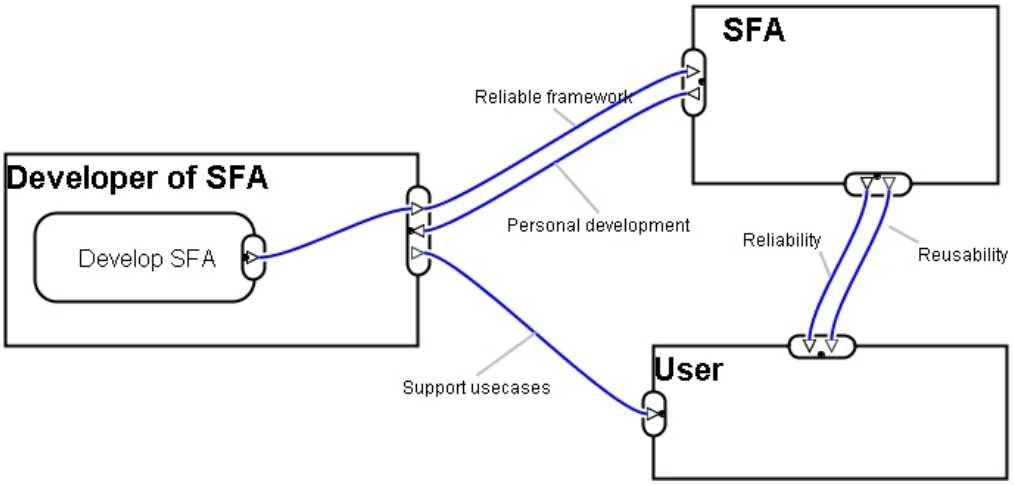

Figure 3.3: An example of an e3-value model in the context of this study

Figure 3.3 shows an example of an e3-value model where 3 actors are represented, the developers of SFA, the users and the framework. The developers give the value of reliable framework to SFA by having the value activity of developing SFA, while receiving the value of personal development. The developers also give the value of supporting use cases to the users of the framework. SFA gives reliability and reusability to its users.

3.2 Validation

In this section we present the methods used to validate the data collected through interviews, surveys and archival data.

3.2.1 Member Checking

Member checking is a well-known qualitative technique to improve the accuracy, credibility and validity of the data collected during an interview [31]. It is considered an informant feedback and external validity. During member checking, the researchers will summarize the collected data and present them to the participants. The participants will agree with the presented summary or disagree and provide feedback and reflections on it. Member checking can happen at the start of the data collection and analysis phase or towards the end of the project [31]. There are several drawbacks and other issues with member checking however, which should be accounted for. In the process of member checking the participants may be confused, and even change their mind regarding issues based on this. Different members may have different views of the same data [31].

In this case study, member checking was used to validate the models constructed, which were primarily based on interpretations of data collected from employees in interviews. The members were previously interviewed participants. 4 new interviews were scheduled with developers, users and potential users of the framework. Each interview lasted around an hour and during that we described the models and went

through them step by step. We received some good feedback as well as improvements and suggestions to the models, specifically the goal model. Overall there was no disagreement on the data presented to the participants.

3.3 Validity threats

The validity of this study shows to what extent the results are true and not biased [22]. The validity of the case study should be addressed during all the phases of conducting a case study [22]. The threats to validity, approaches and methods on reducing them are explained in this section based on four aspects of validity:

3.3.1 Construct validity

Construct validity shows to what extent the operational measures represents what the researcher wants according to the research questions [22]. One threat to the construct validity of this study is misinterpretation of the interview questions by the participants, to limit this threat, the interview questions were grouped based on the role of the participants whether it is a developer, user, or potential user. The participants were also informed beforehand about the research topic to give them time for preparations. Member checking was used on some participants to ensure the data collected and analyzed represent their opinions and answers.

3.3.2 Internal validity

Internal validity are factors that affect the research without the knowledge of the researcher [22]. Triangulation is one of the tools that help limiting this threat, as the researchers chose this method to cover more aspects of the study, by collecting data using interviews and archival data.

Interview participants were primarily located at Ericsson Lindholmen, and of those users interviewed they have all had on-site access to SFA developers or other persons with knowledge. Participants themselves indicated that this aided them when using the framework, and therefore had a positive effect on the time it took to implement the framework initially and the burden of knowledge required. This introduces a bias in the perceived disadvantages of the framework, as they're mitigated by nature of having high knowledge personnel on the same site. To limit this threat we tried to interview users who are not located at Lindholmen, these participants were not available for interviews but they answered a survey we designed that had the interview questions in mind.

3.3.3 External validity

External validity shows to what extent it is possible to generalize the findings and to what extent they are interesting to people outside the case [22]. One of the issues of this study is ensuring the results of the research can be applied to different cases and domains. To reduce this threat, other companies were involved in this case through workshops as part of a larger project (APIS) lead by Software Center at Lindholmen

in Gothenburg. Our findings are considered relevant to other case studies that has similar characteristics to our case, by having a common framework which is based on a set of rules.

3.3.4 Reliability

Reliability shows to what extent the data and analysis is dependent on the researcher [22]. The results should be similar if the study was conducted by other researchers. This study was conducted by two researchers which made it possible to avoid single bias. The data was analyzed and member checked with some of the participants to ensure the data and analysis are correct.

4
Result

This chapter presents the result of the study. The first section covers our RQ1, the second section covers RQ2 and the third section covers the evaluation models we used to answer both of the research questions

4.1 RQ1 - What are the values and challenges of having a rule-based framework?

In this section, we present the results related to RQ1. These results show the values for the stakeholders of SFA as well as the challenges of developing and using it. These results are gathered from the interviews and archival data. The archival data includes the SFA framework user guide, SFA user training and the state, fault and alarm handling design rules.

For this research question, we have identified two categories based on the sub-questions, values and challenges. Under the value category we identified 3 different themes based on our thematic analysis: Design rules support, reduce cost and simplicity. Under each theme we identified different patterns as shown in figure 4.1. As for the challenges category, we identified 4 different themes: Modifiability, testability, support and understandability. Under each theme we identified different patterns, large scope pattern is presented in two different themes as it affects different stakeholders (developers and users) in different ways, as as shown in figure 4.2.

4.1.1 Values

In this section, the value of using SFA is presented. The values found are divided into three distinct themes based on our thematic analysis (see section 3.1.4), as seen in the thematic map in figure 4.1.

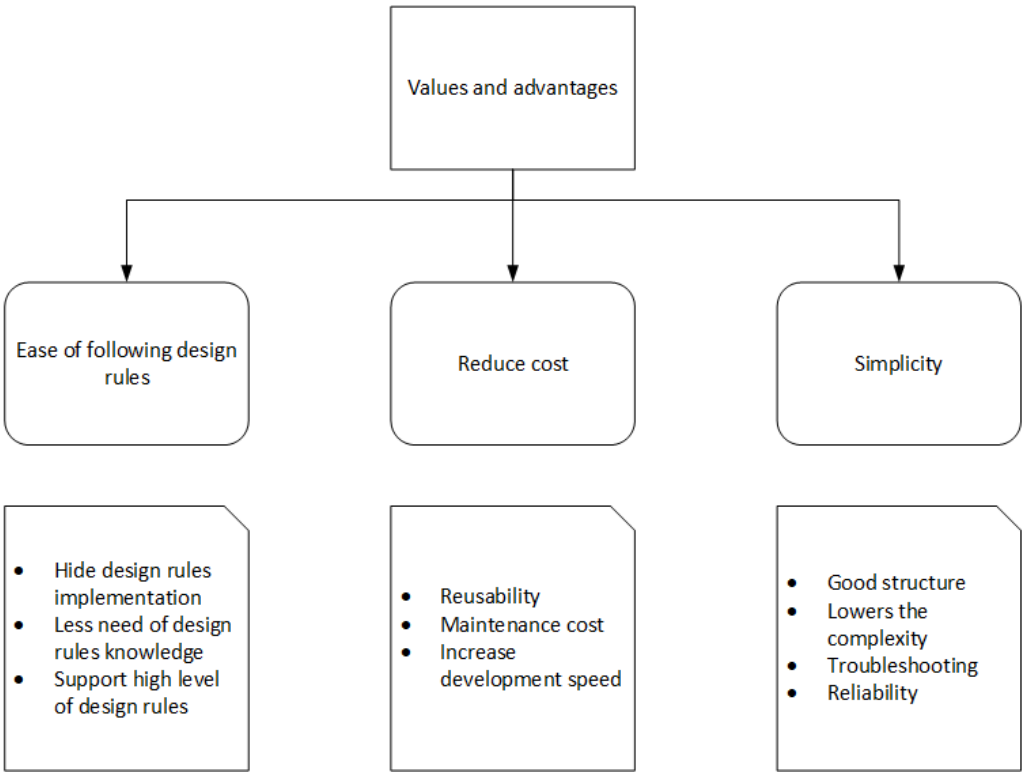

Figure 4.1: Thematic map of the values identified

4.1.1.1 Ease of following design rules

Hide design rule implementation
One participant mentioned that he noticed during the time he used the framework that the design rules were hidden inside the framework. There are a lot of design rules for state, fault and alarm handling which makes it a complicated process with coordinating these rules. Another participant explained that SFA hides the design rules and dependencies in a good way. It implements the design rules in a way that there is a standard behind it, things are not likely to change which makes it a stable framework.
Less need of extensive design rule knowledge
One participant mentioned that a user should have a fair knowledge of the basic propagation and aggregation design rules. There is a full traceability between the design rules and the framework, i.e knowing which design rules are implemented where. Another participant said that a user can use SFA without knowing the design rules, but understanding them and which part of the framework they are realized in can be helpful. From a generic user perspective, no extensive knowledge is needed.

One user mentioned that his design rules knowledge is not very high, he relies mostly on other team members who knows the rules better. He explained that one can use the framework in basic functionalities without full knowledge of the design rules. Some rules are covered by the application layer and some are covered by SFA. The framework can't cover all the design rules. Covering all of them will take more than one framework. Another participant said that in order to use the framework,

the users need to know the basic concepts of fault handling, they don't need to know all the details about everything.

Support high level of the design rules

Adding rules and behaviours in a consistent way is considered a value as one participant explained. Not following the design rules is seen as a cost inside Ericsson. SFA is helping the users to follow these rules and provide an easier way to review new design rules *"It's very good to have this framework to know you are following the rules and standards"* - U04. Easing the design rules review process can be seen as a value for the managers to manage and update their design rules.

4.1.1.2 Reduce cost

Reusability

SFA reduces the amount of boilerplate code for handling state and fault operations between entity objects (EO). Many of the participants explained that code reusability is one of the values SFA provide to them. One participant mentioned that SFA provides free functionalities to their product, another participant explained how the framework allowed him to reuse code in different components within their product.

Maintenance cost

SFA reduces maintenance cost for the products using it, this can also be seen by the potential users as one participant explained the perceived advantages of using the framework is that another team will take care of the maintenance.

> *"Someone else takes care of the maintenance, that is a driver for us to get rid of work, and if several products use the framework then it will be faster to resolve issues"* - PU01

Increase development speed

Using SFA help the users to speed up the development of new features and functionalities in their products. This is also seen by the potential users as one explained that the perceived value of using SFA is speed gain.

4.1.1.3 Simplicity

By using SFA, the users felt that it is easy to read and maintain code since they reduce code duplication, one participant said:

> *"It becomes simpler to add new fault type or new behaviour for states, when there is a fault scenario, there are few places to add code"* - U01

Reliability

Reliability is one of the values seen by the users, when a user adds a feature to any of the applications using SFA they have a framework they can rely on and it makes the process easier. One participant said:

> *"The framework gives the user a software that can be applied and do huge deployment and it just works, when the user get introduced to it they expect it to work and that is a big value when it comes to software"* - D03

Trusting the framework and rely on it in the future is something the users should do as a participant explained:

> *"The value of SFA for the users is that they get something they can trust after some time, at the beginning it might be tricky, the more you use it the more you will benefit from it, it's an investment"* - U04

SFA is one application that is dependant and not mixed with other applications, it has its own interface. A clear interface that forces users to use gives them a robust and stable framework. A potential user explained the perceived value for them using SFA is to have more robust functionalities.

Good structure of rules and dependencies between components

The application side has a good structure with rules and dependencies but the framework helps sharing it with different components as one participant explained. Another participant mentioned that SFA help the developers to write easy to understand code while separating the behaviour, instead of having one large class, the users can have many objects interacting with each other, this can increase the maintainability and understandability of the code.

> *"You can separate the behaviour, instead of writing a big class you can have many objects interacting with each other"* - U02

Lowers the complexity

When using SFA, the application becomes simpler from an application point of view, as one participant said. Another participant mentioned that SFA helped lowering the complexity and ease the development process, as she described it:

> *"One of the good things about using SFA that it take care of things that you don't need to think about, as a user you send an error and it's someone else's problem"* - U03

SFA takes a domain area that is difficult to handle in large products which has different states and dependencies and help reducing its complexity,

> *"SFA take a domain area that can be difficult to handle in large products which has different states and ease the process of how they react with each other, it takes care of areas that can be tricky and complicated"* - D02

The users don't need to care about all the details related to fault, state or alarm handling. SFA brings another layer of abstraction that is less complex than the full details as one participant mentioned:

> *"You don't need to care about the full details related to fault handling, the messy parts, you have another abstract layer that might be complex also but much less complex than the details."* - D03

the logic becomes more clear and compact when using SFA. A participant mentioned that while using SFA, the separation of the logic to state became easier, it helped easing the coordination of states between objects.

Troubleshooting becomes easier

When using a common framework across different products, troubleshooting becomes easier for all the users, one developer explained *"We had the possibility to implement troubleshooting and other things to help the system"* - D03. A user of the framework explained how it made it easier to review code and make sure you are following the correct specification

> *"The implementation you do on top of the framework will reflect on the problem, you can easily review the code and see its correct. Otherwise*

if you have to manually test it, it will be hard, you can easy see if the code is following the specs or not" - U04

4.1.1.4 Value model

From the results regarding RQ1a and the goal model, we have mapped the values to different actors in the context of this study. The value model has 6 different actors that exchange values between one another as shown in figure 4.2. Actors can have value activities that help generate specific value for the other actors. The actors we identified are similar to the actors found in the goal model which we will discuss about for the second research question.

As the model shows, there are values exchange between SFA and all the other actors in the model. There are also value exchange between other actors directly as the case with systemizer-Developer, Developer-User. The values are represented by a directional relation between the actors, the label above the relation shows what the value is.

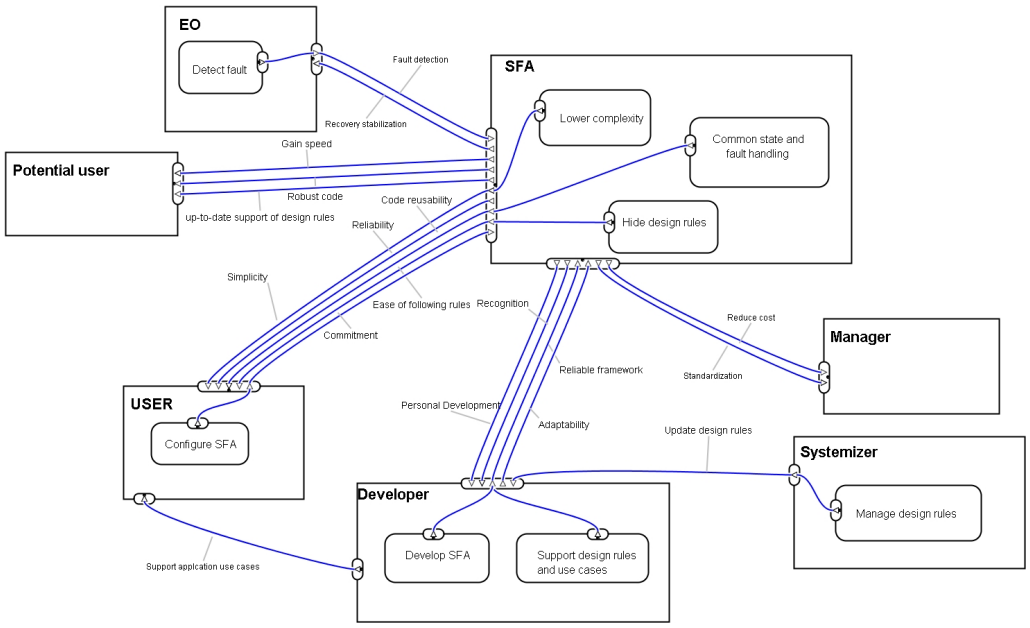

Figure 4.2: e3-value model of SFA showing the value exchange between the different actors

4.1.2 Challenges

In this section the challenges of using SFA is presented. The challenges found are divided into four distinct themes based on our thematic analysis (see section 3.1.4), as seen in the thematic map in figure 4.3.

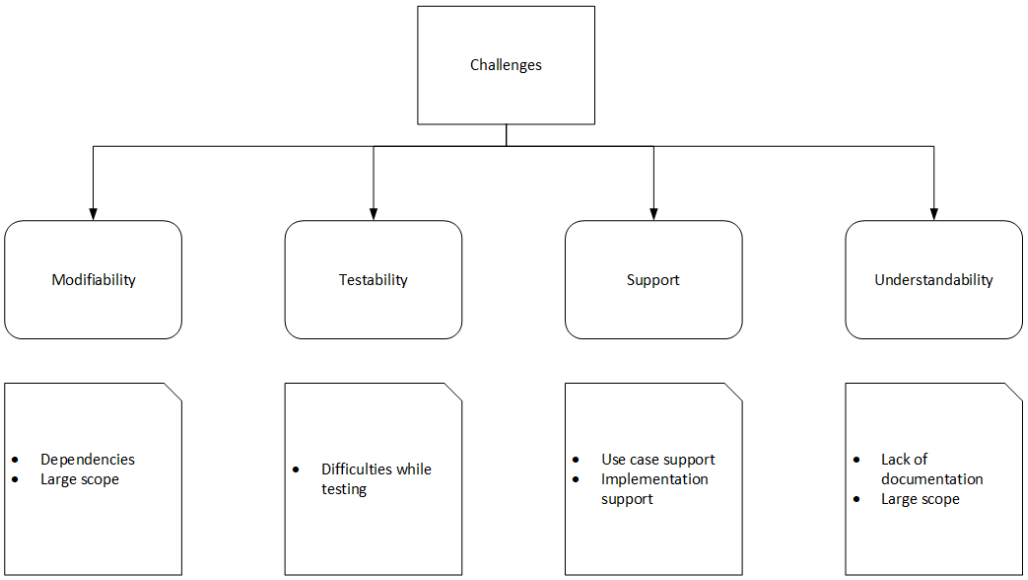

Figure 4.3: Thematic map of the challenges

4.1.2.1 Modifiability

Dependencies

Frameworks are commonly shared between several applications. A consequence of this is that there are several dependent applications being affected by changes made to the framework. An implication of this is the importance of making correct initial decisions to minimize the need of large changes down the line. A developer of the framework mentioned that it is difficult to modify SFA, since if there is a need to change requirements it's hard to remove the old legacy part since there are other users using it. A large number of dependencies can therefore cause a lock-in effect.

> *"If you have many users and one provider, this can cause a lock-in effect. Where the cost of change is very high since many users are using the API. it's important to get it right from the start, but everyone in development knows how hard it is to do that"* - SA01

Participants stated that any updates to common code must be stable, since it will have an effect everywhere. One developer recalled a very minor update for one software module, which ended up causing issues for several others, noting how hard it is to predict these things.

Large scope

Developers of the framework have stated there are difficulties in modifying the framework due to the large scope. The larger domain increases the time and effort needed to have complete understanding of the framework for the developer.

> *"There are quite a lot of interfaces that could be difficult to understand and how they are implemented, quite complex"* - D03

Developers have also stated that the scope makes it harder to provide an easy to understand API for the users. Several participants have stated that separating the framework would be beneficial, as even dependencies within the framework itself are becoming an issue.

4.1.2.2 Testability

Difficulties testing
One developer stated that there have been difficulties in testing the framework properly.

> *"It has been difficult to test the framework properly. Specially with different complex configurations. You can test smaller parts but not the entire framework".* - D02

Other participants have said that the lack of testing is mainly caused by the high cost, so it has not been prioritized. A product guardian said that these testing difficulties are to be expected when the development isn't test driven. When there are difficulties with testing, modifying the framework can be harder. One participant noted the issues of changing the API, as this changes the tests as well.

4.1.2.3 Support

Use case support
Several participants talked about the need of supporting specific use cases that may not be encompassed by the design rules, as products need these to function properly.

> *"Use cases are the important to support, not everything from the use case is in the design rules."* -D01

This could create issues for the developer and higher-level point of view, the scope of the framework is increased and the commonality is decreased. Despite this, it appears to be a necessary inclusion as users state that many problems they encountered during the implementation of the framework was due to some use cases not being supported yet. It is also a concern for potential users, as one states their concern that SFA may not support their use cases.

> *"We might only be able to use 20% of SFA so we don't know if it's worth using it then"* - PU01.

Implementation support
There is a need to write different API adapters for each product at Ericsson that does not support SFA currently, this can be time consuming and require an understanding of the framework. SFA has a lot of interfaces that can be difficult and complex to understand and implement.

4.1.2.4 Understandability

Lack of documentation
One issue that the users of Ericsson experience is the lack of documentation related to the framework. The little documentation that do exist felt directed towards the developers of the framework, rather than the users.

> *"SFA is not well documented, there is a lack of examples of real and complex scenarios of usage"* - U06

On-site users expressed the value of face to face communication with the developers when solving more complex issues, mitigating the need for documentation at times. One user talked about the lack of documentation related to tracability between use cases and their implementation, caused by the communication method.

> *"It was more face to face communication between the domain users and development. There was no formal way of doing it, we were the same team sitting in the same area."* - UO2

Large scope

Users have perceived the framework as being too large in scope. There's a high time investment to understand and learn the framework. Users who used the framework for shorter period of times, thought that the time they needed to invest in learning it was not worth the value gained. The framework could be separated into smaller components to help mitigate these issues, as stated by a software architect.

> *"We think of separating the framework into smaller components, the fault and alarm can be together and the state could be separated."* - SA01

A product guardian and potential user of the framework believe that dividing it into smaller components, would make it more likely to be widely used. Since only the state handling part is interesting for some products, including theirs.

4.1.3 Goal model

In this section we present the goal model and the validation process. The model visualizes and the challenges and values identified for the research questions and map them to the relevant actors. Tasks or lack of tasks can show why these challenges and values exist.

4.1.3.1 Validation of the model

During the member checking process, we presented our findings using the goal model to the participants. We started explaining the modelling language what the different shapes they represent, then we showed an overview of the entire model with the different layers, actors and dependencies as shown in figure 4.4. We zoomed in on each actor, explaining the different goals, unachieved goals and tasks. We had participants representing different actors in the model from developers, users, potential users and software architects.

All participants agreed with the goal model and the data represented in it, there was no conflict or misinterpretation of results besides manager and systemizer actors. From the evaluation, we discovered that a different actor is responsible for managing the design rules which is the systemizer, and the manager's interest is to have a common state and fault handling to help increase quality, reduce cost and gain speed. Other valuable feedback was given to extend on the model.

Some of the changes for the model after the validation process are:

- User: Adding soft goals "better documentation" and "common debugging"
- Potential user: Adding a goal "divide SFA into smaller components"
- EO: Adding a goal "more predictable behaviour"
- Developer: Adding soft goals: "divide SFA into smaller components", "Improve test coverage" and "Improve documentation". Adding a task "Support users troubleshooting"
- SFA: Adding a task "Log events (dumps)"
- New actors: Adding the systemizer and troubleshooting data actors to the model

4.1.3.2 The goal model in details

The goal model in figure 4.5 shows 4 different layers, a domain layer, an application software layer, an API layer and an assets layer. We have identified 9 actors in the context of this specific model related to SFA Framework to understand the drivers for the current design. On the domain level, we identified the current users of SFA as one actor, the potential users who might use SFA in the near future as a second actor, the managers as a third actor and the systemizer as a fourth actor. On the application software layer, we have identified the entity objects (EOs) that implement SFA and the developers which developed the framework. On the API layer, we identified the framework itself as an actor and finally on the business assets layer, we identified the design rules for fault, state and alarm handling and the troubleshooting data as actors.

There are dependencies between the actors as shown in the model, the systemizer manages and updates the design rules, the developers implement these design rules into the framework, the developers also depend on the use cases provided by the users to ensure that the framework supports them. The users and potential users depend on SFA to achieve some of the goals and perform some of the tasks they have. The EOs depend on SFA to evaluate the faults and check recoverability status. SFA depends on the developers to develop and maintain it, on the users to configure it and on the EOs to be provided with registered faults.

Under the actor boundaries, we have identified goals, soft goals, tasks that can help achieve these goals and resources that are used by the tasks and goals. The yellow goals and soft goals means that they are not achieved yet according to the actor.

Figure 4.4: Goal model of SFA before validation, showing the goals the different actors has from using the framework and which tasks help achieve such goals

Figure 4.5: Goal model of SFA after validation, showing the goals the different actors has from using the framework and which tasks help achieve such goals

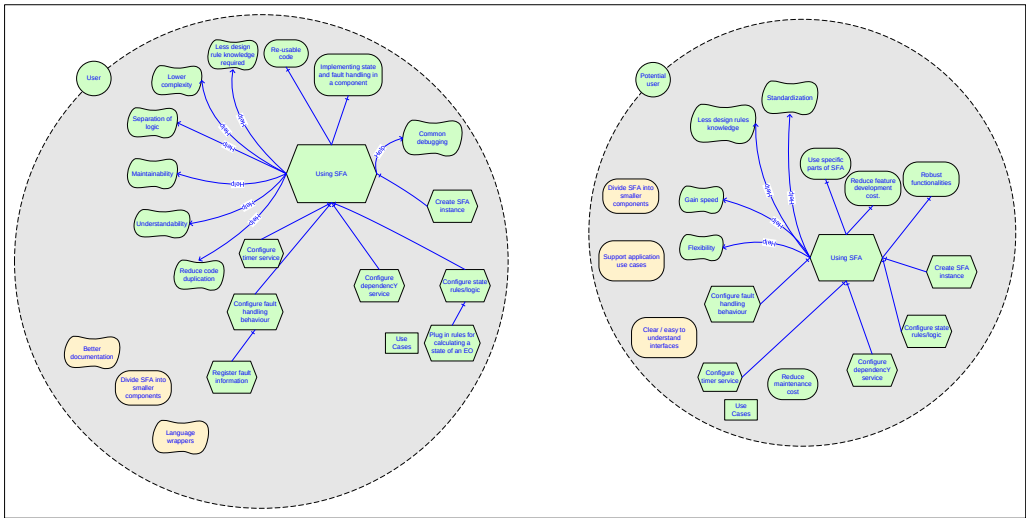

Figure 4.6: The user and potential user actors in the goal model zoomed in

Figure 4.6 shows the user and potential user actors in the model. The user has many soft goals or quality attributes that is achieved by using the framework. These goals are understandability, maintainability, separation of logic, less complexity, less design rules knowledge needed, common debugging and reduce code duplication. The user also has two hard goals that are achieved which are implementing state and fault handling in a component and having re-usable code. All these goals are achieved the task "using SFA" which has other smaller tasks. The user currently has some unachieved goals which are having better documentation, provide more language wrappers and divide the framework into smaller components.

The potential user has some similar goals to achieve as the user of the framework, with the addition of reducing feature development cost, using specific part of the framework and standardization. They also have unachieved goals such as dividing the framework, supporting their application use cases and having an easy to understand framework.

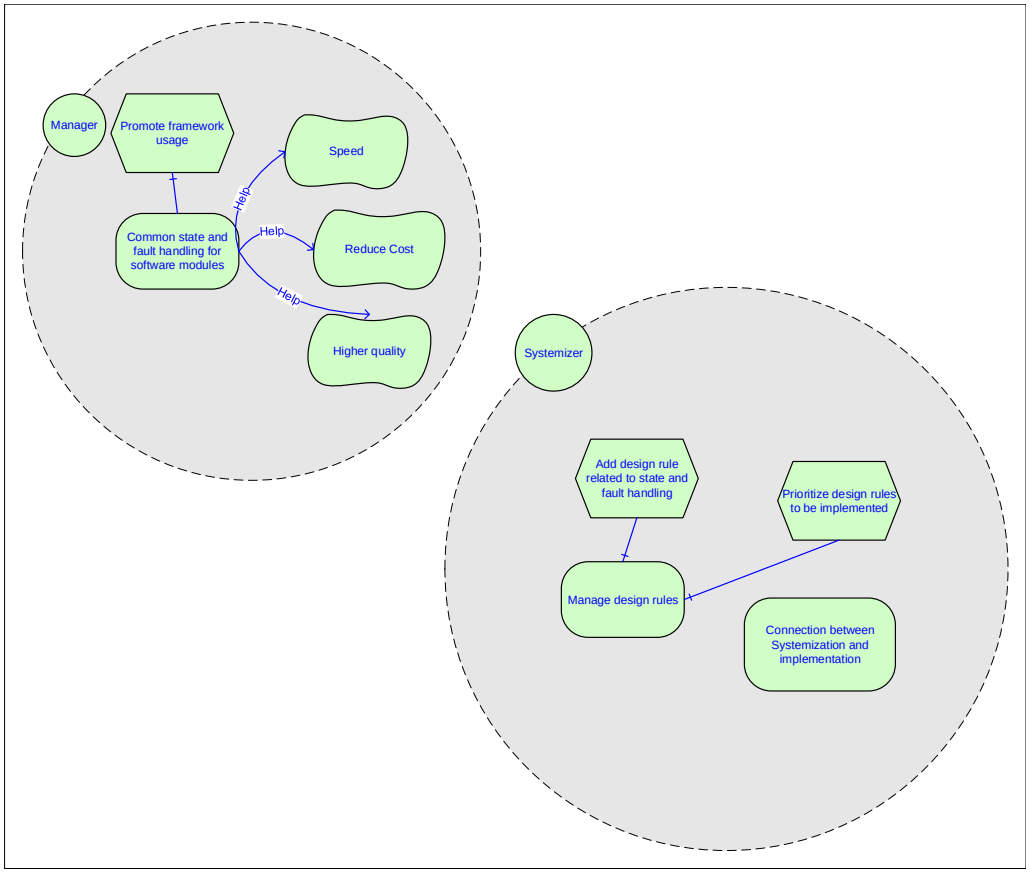

Figure 4.7: The manager and systemizer actors in the goal model zoomed in

Figure 4.7 shows the manager and systemizer actors in the model. The manager has a hard goal of having a common state and fault handling for software modules, which helps achieving other soft goals like speed, reduce cost and having a higher quality products. The systemizer has a goal of managing the design rules, which can be achieved by adding and prioritizing rules related to state, fault and alarm handling. The systemizer also has the goal of a connection between the systematization and implementation of the framework.

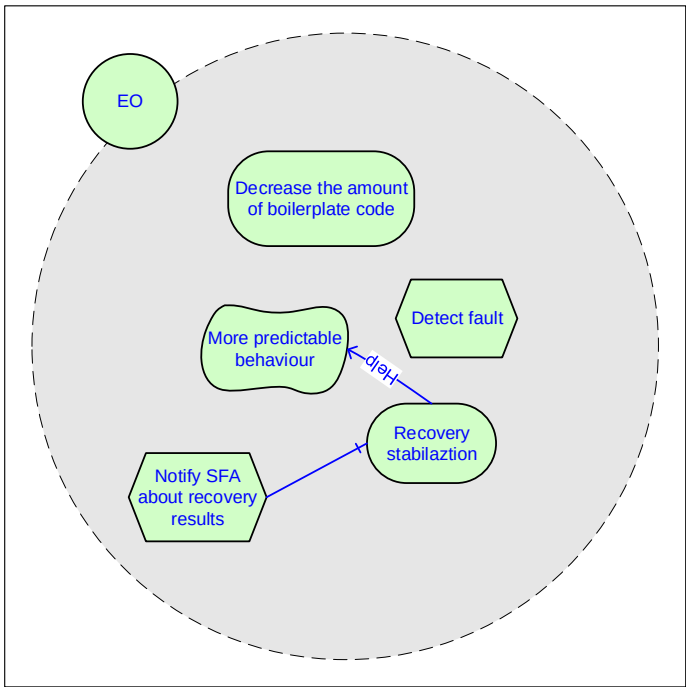

Figure 4.8: The EO actor in the goal model zoomed in

Figure 4.8 shows the Entity object actor in the model. The EO has a hard goal of decreasing the amount of boilerplate code and recovery stabilization which lead to a more predictable behaviour. The EO also has tasks that help the framework such as detecting a fault and notifying SFA about recovery results.

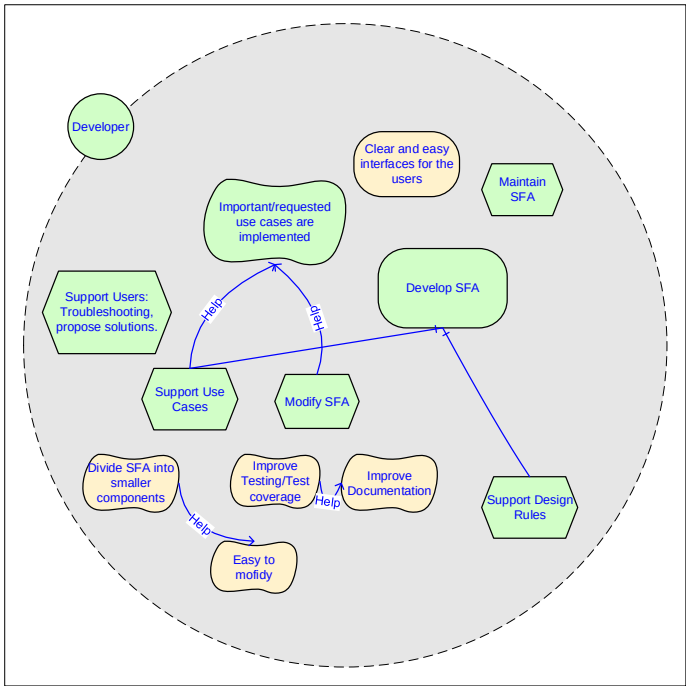

Figure 4.9: The developer actor in the goal model zoomed in

Figure 4.9 shows the developer of SFA actor in the model. The developer has

a hard goal of developing SFA and a soft goal of supporting and implementing use cases that should be implemented in the framework. The unachieved goals for the developers are dividing the framework into smaller components which can help making the framework easy to modify, improving test coverage and documentation for the framework. The developers have some tasks to do which is maintaining SFA, offer support and troubleshooting for the users, modifying the framework, support use cases and design rules.

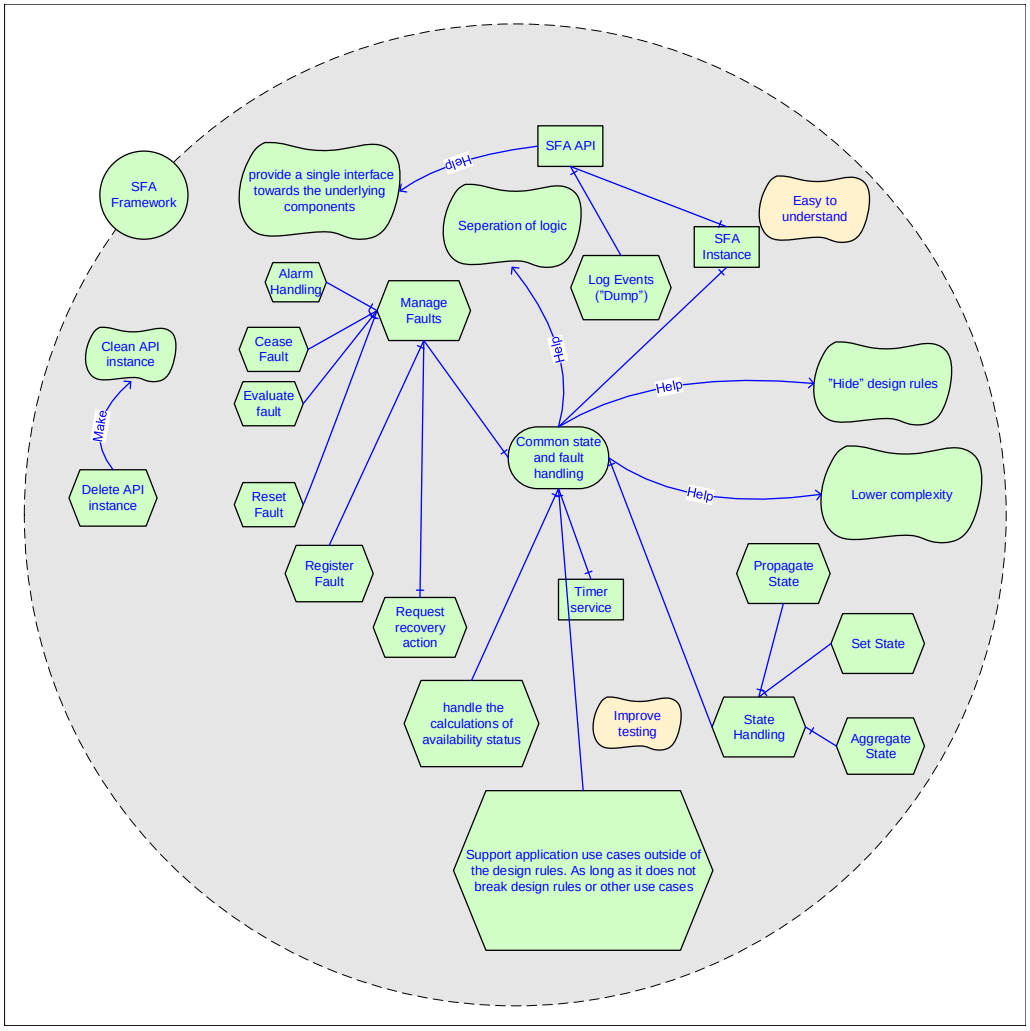

Figure 4.10: The SFA actor in the goal model zoomed in

Figure 4.10 shows the framework itself as an actor in the model. SFA has many tasks that support the goals of the framework and other actors. The goals of the framework are the common state and fault handling, lowering the complexity, hiding the design rules, separation of logic and provide single interface towards the underlying components. SFA has some unachieved goals, improve the testing process and make it easy to understand for both developers and users.

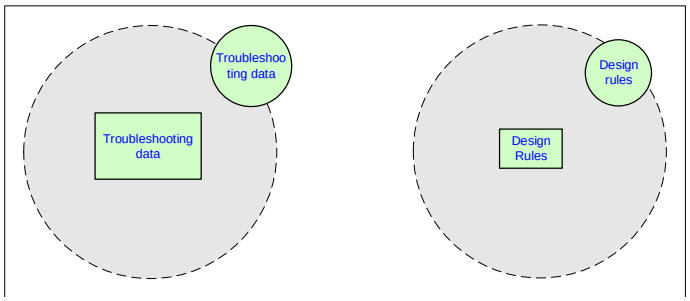

Figure 4.11: The assets actors in the goal model zoomed in

Figure 4.11 shows the troubleshooting data and the design rules as assets in this model.

4.2 RQ2 - What drives the design and development of a rule-based framework?

For RQ2 we have identified two themes to answer our sub-questions as shown in figure 4.12, the first theme is the design drivers for the framework, which include the initial and current design rules and use cases as patterns. The second theme is the potential user which has promoting SFA, cost and misconceptions as patterns.

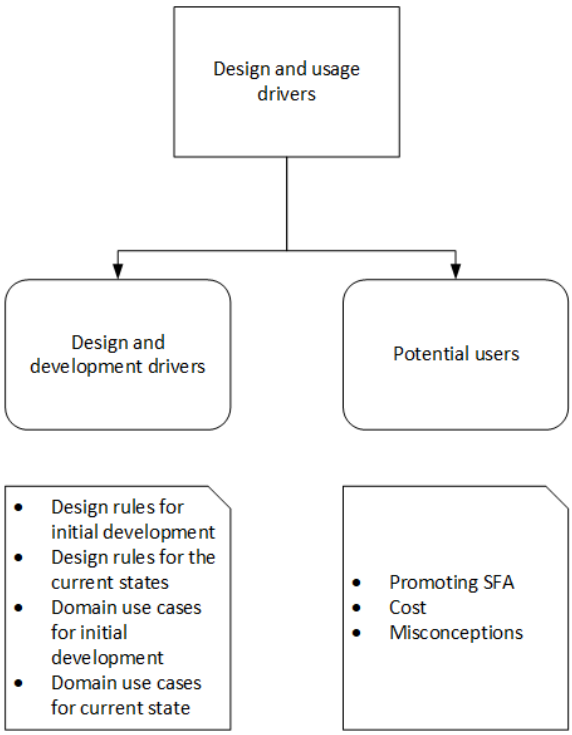

Figure 4.12: Thematic map of the design drivers

4.2.1 Design and development drivers

SFA went through many changes since the first version in 2012, In this section we present our findings related to these changes and what affect the initial and current state of developing and using the framework.

Design rules for the initial development

The relation between the design rules and the framework is an important one, SFA is built to support the design rules and ease the process of implementing them in different components. One participant who was an early developer mentioned that during the development of SFA, they relied mostly on the design rules documents for the state, fault and alarm handling, and checked them regularly.

A developer and a user of the framework said that the design of the framework should follow the design rules to ensure the user is following them. But SFA was driven and influenced by the application use cases since SFA was a part of an application at the start.

The inclusion of the design rules was step wise, there was a priority system based on which design rule to be implemented first and the developers aimed to fully support the design rules.

> *"The inclusion of DR was step wise, they were prioritized based on which to implement first, and we aimed to fully support the design rules. Some use cases affected which design rules to implement in the framework"* - U02

Design rules for the current state

Regarding the design rules, and how they affect the current development of SFA, a developer of SFA mentioned that in the past year, they modified parts of the framework which had faulty implementation of the design rules or missing rules. Ericsson has many design rules regarding state, fault and alarm handling. A developer explained that SFA can't support all the design rules:

> *"We don't have full control if SFA fulfill all the design rules, they take more than a framework to cover"* - D02

He also explained that some of the design rules are already implemented in the applications using the framework:

> *"Some parts of the design rules are covered in the application layers and some are covered in the framework"* - D02

When asking about how to determine which design rules to currently be implemented in the framework, a participant said:

> *"We look what is common between applications, that's how we decide which DR to include in the framework"* - U01

Domain use cases for the initial development

An early developer of the framework mentioned that domain use cases were more important to support than design rules. He explained that not everything regarding state, fault and alarm handling is covered in the design rules. One participant who is a recent developer of SFA said that the domain use cases were used as a basis to develop the framework while the development of the framework was mainly driven by the design rules.

At the beginning, SFA did not support all the use cases, some applications handled fault, state and alarm by itself, some of the recovery handling is application

specific and was hard to handle by SFA

> *"At the start we limited to the design rules, and at the end we saw that we should support the use cases. As long as we have clear interface and purpose then it should be fine to support more individual application use cases."* - D03

Some use cases affected which design rules to be implemented first. There were no domain use cases that were documented to implement the framework, but rather it was face to face communication between the domain users and the developers since both are working in the same area. One participant said:

> *"From a user perspective, the use cases drives the development of SFA and update the design rules since they are only guidelines"* - U02

Domain use cases for the current state

Regarding the use cases, and how they affect the current development of SFA, a developer of SFA mentioned that as long as SFA has a clear interface and a purpose then it should be ok to support more individual application use cases. One participant explained that most use cases for the latest product using SFA were checked to insure it is covered by the framework, and if a use case is discovered to not be supported by the framework, the team will try and modify the framework until it supports it.

> *"Use cases are the important to support, not everything from the use case is in the design rules"* - D01

SFA has not been tested on a large set of use cases for different products but it is expected to evolve over time and support different use cases. This will require the different teams to modify the framework based on their needs. There is an exception when it comes to special or rare use cases that should not be included in the framework.

> *"Not all of the use cases are supported by the framework, some applications handle themselves, and some are handled by SFA, they are application specific and might be hard to add them to the framework"* - U01

Another participant mentioned that there were some problems with use cases that were not supported by the framework. But as the framework evolved it started to support the important use cases. SFA's evolution is important and whenever there are use cases that are not supported and needed, the developers will try to implement it in the framework. There were also new use cases added to cover parts of the design rules and they were valid use cases from the application point of view.

One participant mentioned that the use cases are currently driving the development of the framework and they are being used as a way to update the design rules:

> *"As a user i would say the use cases drives the development of SFA and updates the design rules since they are only guidelines"* - U02

4.2.2 Potential users

Potential users have an effect on the usage and development of the framework, in this section we present our findings regarding the potential users and the current

state of SFA

Promoting SFA

Promoting the framework in a good way might bring more users and achieve the goal of SFA being a common state, fault and alarm handling framework at Ericsson. According to some potential users, there is lack of sharing information regarding SFA, one potential user mentioned that he heard about it a month ago while SFA was developed for several years and he explained:

> *"Based on my experience at Erickson we are bad at sharing knowledge and that is why we didn't hear about SFA. I don't know how to solve it. We should be better at promoting things"* - PU01

Another participant explained about barriers of using the framework:

> *"The highest barrier we have of not using frameworks is the culture, at Ericsson and the second one is the size of the framework"* - SA01

When creating a framework, there are few things in mind the developers need to keep, such as the ideal time and in which domain the framework belongs, in Ericsson's case which products to have in mind.

> *"Its hard to tell when its ideal time to create a framework and get everyone on board, you need command and control culture, and we don't have that, there is no top control entity to say to use the framework"* - SA01

A potential user explained his thinking of SFA and that it was meant for specific products and not their product:

> *"Probably they didn't think of our product when they developed the framework. We have slightly different way of handling alarms and such"* - PU01

Cost

When asking the participants about the usage of SFA and what prevent some users of migrating to the framework, cost was one of the issues specially for old products. One developer explained that it is expensive to migrate existing products to SFA:

> *"Why isn't it used? There are two ways to use it, one is to build new applications around it and that is done, the other is to replace existing solutions with SFA which is costly. Its low cost to add it to new product, its high cost to implement it in current products then you need to decide if its worth it"* - D03

A potential user explained that they should do their own analysis on the framework and that might be costly, but in most situations, he trust the people he works with at Ericsson. Their product is very old and currently supporting new features, they have not re written any code which lead to having a lot of legacy code, because of that they don't follow the current design rules. Using SFA might help them keep their product up to date with the design rules but at some cost.

Misconceptions

There are some misconceptions that the potential users have about using SFA, one potential user is worried about changing their product too much in order to use the framework:

> *"What I'm worried about is changing requirements if it's not possible to change the framework, perhaps we should change our requirements,*

> *we don't know yet. If we look at it as a whole it can be complicated"* - PU01

He then continue to explain that some specific use cases are not currently supported by the framework and might not be supported in the future due to priorities and other products using SFA. Another potential user said that the main reason for not using SFA in their product is that SFA was built and introduced specifically for 2 different products suitable for fault handling and that's what lead the development of it, their product focuses on state handling which is still a part of SFA. They wouldn't use SFA if their use cases are very special, or if the framework is limited regarding state handling, they are not sure how SFA handles states and how flexible the API is.

As for changing legacy code and refactoring in order to use SFA, one participant explained that it's never a problem and said:

> *"Changing legacy code or refactoring in order to use SFA is never a problem, it's always better to update and refactor, software development business is like this, we have to think far away from one year and it will cost at the start but will be beneficial for the future, it's the common sense of software development, there is no other way of doing it"* - PU02

5
Discussion

In this chapter we will discuss the values of having a rule-based framework, the advantages and challenges as well as the perceived value for current and potential users. We will also discuss the design drivers for such frameworks, and whether it is a top-down approach where the design is driven by the domain use cases or a bottom-up approach where the design is driven by the design rules. We will discuss these issues based on our findings in the previous section as well as the related literature and theory.

5.1 RQ1

In this section we discuss the values, advantages and challenges of having a rule-based framework. What is common between rule-based framework in this study and frameworks in general as well as looking at the differences between them.

5.1.1 Values

Our findings show several values and benefits of using SFA for different stakeholders. Encapsulating the state, fault and alarm handling to lessens the amount of boilerplate code was mentioned in most of the documentation regarding the framework. This was also supported by many different participants as the most common advantage for the framework. Frameworks in general are considered as "a generic application that allows the creation of different applications from an application (sub)domain" [12].

Providing free functionalities helped the users to improve the quality, productivity and reduce cost for their applications by having a common framework which focuses on a specific domain. Code reusability is one of the most common advantages of frameworks in general [3][4][5].

The framework enabled the users to lower the complexity of implementing solutions for state, fault or alarm handling in their products by providing a clear interface that hides the complex components encapsulated by the framework. It allows the users to configure it, and decide which parts of the framework to use in their applications. This is aligned with frameworks being software systems that can be customized, specialized or extended based on the needs of the developers using it [13]. SFA also helped the users to have a clear and compact code, ease the development process overall and separate the logic while dealing with the different parts of SFA. An application only needs to know the functions and parameters when using

a framework provided through the API and ignores the detailed implementation inside the framework such as the "hot spots" [14][12].

The related work of the benefits, advantages and value of using software frameworks which were described in section 2.1.2, are all found in the data collected. Among common themes identified were the previously known values such as reusability, increased development speed and reducing complexity. These findings thus serve to confirm that SFA achieves the known theoretical benefits of using a framework.

As for the design rules, the basis of the framework. All participants stated that the framework was hiding the design rules to some extent. In the domain of SFA, there are many design rules regarding the different parts of SFA (State, fault, and alarm handling design rules) and one of the advantages of using the framework is that the users don't interact that much with the design rules, having in mind the framework supports what is needed for their application, thus hiding the design rules gives them the ease of mind, trust and reliability. This value of enforcing conferment to design rules in the application was not found in the literature. This value could however be considered to be in the same category as the more general value of reducing complexity. Reducing complexity is a value brought by not requiring the users to implement a set of complex solutions themselves, as the framework provides these solutions. Conforming to design rules by using the framework means there is less work required of the users in learning and understanding the rules in depth, integrating them in their own solution and verify through testing that the rules are followed. While this do reduce the complexity for the users, we argue the value has more depth than only reducing complexity. Another value that was mentioned in the literature analyzed is the improvement of troubleshooting. The framework introduces uses an event log, which means all products using the framework will log any issues in a consistent and similar manner.

Applying e3 value modelling did not show anything new compared to the goal model and analysis, it only presents the values gained in a more direct and abstract manner. It can serve as a quick visualization of which stakeholders gain what value in a setting, but it does not explain *why* or *how* they gain it.

5.1.2 Challenges

Despite the advantages of having a rule-based framework like SFA, there are some issues and drawbacks that comes with it, some of them were addressed during the evolution process which we will discuss later. One of the issues is the scope of SFA, some users consider it too large and could be separated and divided into smaller components. This is mainly dependent on the specific use cases and usage areas of the products utilizing the framework. The domain size is a well-known issue in the literature when developing a framework, if the domain is too large it will be difficult to show the usability of the framework and might create confusion for the users [5].

Another issue with SFA is the time it takes to understand the framework, most of the participants said that it wasn't easy to understand the framework and it took a lot of time for some users to use basic functionalities. A framework should be easy to understand, it should take less time to use it than developing the solution directly for the applications [4]. Lack of documentation may increase the complexity

and affect the understandability and usability, in order for the users to use the framework correctly, they must understand the framework interface and how to use the methods in the framework API [15], SFA has a lot of interfaces that can be difficult and complex to understand and implement. Testing is another well-known issue for framework development [5], developers of SFA consider testing the entire framework a difficult process, mainly due to the different complex configurations that SFA provides, so instead the testing is done on parts of SFA.

Having a common framework that is used by different applications can be problematic as well, In the SFA case, there is a need to write different API adapters for each application that does not support SFA. There will be a need to modify the framework, the application using it, or both. Modifying SFA is an issue for the developers, there are a lot of dependencies with other products that makes it difficult to modify. Any common code updated wrong will affect all the products associated with it. Some of the applications that can use SFA have a lot of legacy code and need refactoring. The main question will be is it worth to spend the time and effort on modifying the framework and the applications to use SFA? There is no clear answer to this question since each product need to investigate and study the benefits and drawbacks of migrating to SFA. Another perceived disadvantage from the potential users is the loss of control over the relevant parts of their product. Since another team is responsible for maintaining and updating the framework, any changes needed by the users will have to go through that team.

Overall, these challenges found are known common issues that comes with using framework, as described in section 2.1.2. However, a number of theoretical challenges associated with frameworks were not found in SFA. These challenges include; mismatch of architectural styles, reacting to business changes, applicability to users and debugging. Some of these challenges may not be applicable for SFA, as it is an internal framework, commercial frameworks may have challenges that are unique to them. Also during the data collection method, we did not attempt to verify known challenges, only gather the ones which participants perceived to exist. Other known issues may therefore be present in the framework, but the challenges collected are the ones considered more important by the stakeholders.

5.1.3 Goal model

The goal modelling was used as a method to map the ecosystem of stakeholders and their goals. The aim of this was to have a model that can give an easy to understand overview of the values and challenges that exist in the system, and to which stakeholders these are clustered at.

The models were validated through member checking at the end of the case study, and positive feedback was given. While the main purpose of the validation was to validate the data itself, the models were perceived as easy to understand quickly and displayed the relevant information clearly.

The use of goal models as documentation for software systems is an area of research that could be explored further. It can serve several purposes, displaying the current value the software brings in order to "sell it in" or used for pre-study to display what is to be gained by creating the software.

5.2 RQ2

When creating a framework, typically there should be a need for its usage, a domain or sub-domain to be identified which will be covered by it, and finding the hot spots and what functionality it should offer to the users. In this study, we considered two aspects of approaching the design and development of the framework, driven by use cases or the design rules, top-down and bottom-up respectively. In this section, the results and how they fit into these two approaches is discussed. We will also discuss the drivers of promoting framework usage in a company environment.

5.2.1 Design Rules and Use Cases

The main purpose of developing SFA is to have a framework that can enforce the support of one set of design rules at Ericsson. This can ease the process of implementing the rules in different products and introduce a commonality of how they are implemented. The design rules chosen to be included in the framework has been driven by the use cases in the early stages of the framework.

The frameworks initial development began as a way to encapsulate the current state and fault handling solution for one software module. Since it was therefore necessary to ensure support of this module, the initial development and conception of SFA was driven by application use cases in addition to the design rules. The framework was simply a part of an application in these early stages.

Looking at the findings related to the current development of the framework, use cases seem to be the main driver of the framework. Support of design rules is driven by the needs of applications, as some are either not applicable or already supported in the application layer. The use cases of the applications using the framework can be unrelated to the design rules. The findings show that the use cases can be considered the main driver of the framework at present.

As discussed in relation to RQ1, there are several challenges associated with the usage and development of the framework. The findings have shown that the lack of use case support would be an issue for the users of the framework. The framework being driven by the use cases is therefore the way of solving these issues. However, a consequence of supporting a wide array of domain use cases is the increased scope that comes with it, compared to limiting the scope to the general set of design rules.

Further questions raised from this is if it's possible to solve these issues in other manners. Can the framework be limited in scope to be driven only by design rules and still be usable by these applications, or is the framework only valuable to the users if it can actually support the use cases. The rules should capture the domain in general while the use cases capture gaps and exceptions in the domain [10], a sort of complimentary strength. The use cases may therefore be necessary to have a complete coverage of a domain [10].

For the purpose of answering the research question at hand, the framework can be driven by use cases and design rules, top-down and bottom-up approach respectively. You can consider there to be a spectrum between these two approaches, as you follow one more than the other. From the findings, we argue it has been mainly driven by a top-down approach, while still having the design rules in consideration. The use

cases themselves can encompass design rules, as such the framework could still reach its purpose of supporting design rules despite not being mainly driven by them. In relation to RQ1 the implications of supporting use cases, which may or may not be part of the design rules, affects challenges and values. More functionalities are added, which if applicable for several products, can add to more reusability. The scope is increased however as use cases from every product is supported, making the challenges of understandability and modifiability worse.

5.2.2 Potential users

To get the most out of a framework, spreading it to as many users as possible in the domain should bring the most value from a management point of view. The findings have shown issues that may be obstacles in attracting new users.

Company culture is perceived as one such obstacle. Sharing of knowledge so that the existence of the framework is known in the first place, so it can be taken into consideration when discussing solutions is one such issue. A Lack of command and control culture where a top-level decision can be made to force usage of frameworks and other solutions also prevents the usage. Solutions to these issues are not very clear, and are harder to apply in larger organizations like Ericsson. One potential avenue to promote framework usage, and other software solutions in general within companies, is the use of open source within organizations, known as inner source [32]. One of the many benefits associated with inner source is the increased software reuse, as normally organizations other teams cannot access others teams' code [32].

For the other the identified obstacles for potential users, the challenges found in RQ1 may have an effect on them. The lack of documentation can make it harder to promote SFA, as documentation is necessary to demonstrate the benefits and purpose of migrating to the framework.

6
Conclusion

This study was conducted to identify the values and challenges associated with a rule-based framework, in addition to identifying what has driven the design, development and usage of the framework. 10 semi-structured interviews were conducted to answer these questions. In relation to RQ1a - What are the values? Some of main results include: 1) The frameworks make it easier to conform to the set of rules that the users must follow, requiring less extensive knowledge in order to implement them. 2) Costs are reduced through reusability and increased development speed. 3) The framework introduces a commonality across products which makes tasks such as troubleshooting easier to conduct. Regarding RQ1b - What are the challenges? The dominant challenges of the framework include: 1) Modifiability is difficult due to the large scope of the framework, and the many dependencies to the framework which are affected by any changes. 2) The understandability of the framework is difficult for both users and developers, due to the size and lacking documentation of the framework.

As to RQ2a - What drives the design and development of a rule-based framework? The findings show that it has been driven by both use cases and design rules, top-down and bottom-up approach respectively. But the main driver in practice has proven to be the use cases, while having the design rules as a basis for the idea of the framework. Consider these two approaches, what is prioritized more or less can be considered in a spectrum, where SFA is placed more towards the top-down approach. When it comes to RQ2b - What drives and motivates the usage of a rule-based framework? The results show that one of the main reasons the framework has not seen more usage, despite having value for potential users, can be contributed to a non-sharing company culture and lack of controlling entities that can force usage of frameworks. This has resulted in unrealized value as stakeholders who can benefit from using the framework, has not done so.

In conclusion, many of the advantages and values associated with the framework have been shown to be many of the commonly known benefits associated with using frameworks in general. As a rule-based framework it has also been shown to sufficiently ease the enforcement the that rules acting as a basis for the framework.

6.1 Implications for Ericsson

This study presents values, advantages and challenges present in a framework that is currently used by Ericsson. The models and other findings related to the benefits of the framework can aid in showing how the framework can be beneficial to use

for potential users. The challenges can help identify problem areas that should be improved upon by developers and other stakeholders involved in the framework. In order to promote framework usage in the company, goal modelling or other similar techniques to demonstrate the value given to the stakeholders may be used as a presenting tool, based on the positive feedback received during validation. Goal modeling can also identify gaps such as unachieved goals of the different stakeholders of a framework. In addition to this, general problems in regard to company culture that is not optimal for sharing of knowledge between teams has been identified as a larger issue within the company.

Improving documentation will alleviate challenges for many stakeholders. It will address understandability for developers and users, as well as help demonstrate the benefits of using the framework when promoting it. To address the lack of knowledge sharing in general due to company culture, larger organizational changes such as using inner source [32] can spread awareness and sharing of solutions internal to the company.

6.2 Contributions to academic research

This study adds to the existing body of academic knowledge within software engineering. The thesis reinforces current known benefits and challenges associated with the use of frameworks. Among the confirmed values are reusability and simplicity, with some known challenges being modifiability and understandability. The study adds knowledge in regard to what unique value rule-based frameworks in particular gives, and adds to the understanding of what rule-based frameworks offer. Showing that rule-based frameworks do in fact make it easier to follow a set of design rules while still benefiting from the other values known to frameworks in general. Additionally, the findings give a perspective on the challenges for a rule-based framework, caused by having to balance two needs, the use cases of applications using the framework, and the rules acting as a basis for the framework.

Applying goal modeling techniques can give an easy to understand overview of the values and challenges for developing and using frameworks. It can also help identifying missing values or solutions for challenges by identifying unachieved goals for different stakeholders. This study demonstrates the applicability of using this modeling technique in investigating values and challenges of frameworks and their APIs. This study adds the applicability of using this modelling technique in investigating values and challenges of frameworks in general. There has not been any previous research on the applicability of using goal modelling in the context of frameworks.

6.3 Future work

As the case study is focused on a single company and single rule-based framework. It would be of interest to study other companies and frameworks. Studying other companies should be done in order to see if the results are unique to Ericsson, or are applicable for other companies well, while also mitigating the effect of company culture. Other frameworks at the Ericsson can also be studied to identify if there is

a pattern in the company. For the framework at hand, SFA, the study was mainly limited to on-site users who had easier access to support when dealing with the framework. Comparing the findings of on-site users versus off-site users could be of interest.

Bibliography

[1] C. Manger, T. Trejderowski, and J. Paduch, "Advantages and disadvantages of framework programming with reference to yii php framework, gideon. net framework and other modern frameworks", *Studia Informatica*, vol. 31, no. 4A, pp. 119–137, 2010.

[2] R. A. Calvo and A. Turani, *E-learning frameworks = (design patterns + software components)*, Jan. 2009.

[3] D. Riehle, "Framework design", PhD thesis, 2000.

[4] G. Froehlich, H. J. Hoover, L. Liu, and P. Sorenson, "Hooking into object-oriented application frameworks", in *Proceedings of the 19th international conference on Software engineering*, ACM, 1997, pp. 491–501.

[5] J. Bosch, P. Molin, M. Mattsson, and P. Bengtsson, *Object-oriented frameworks-problems & experiences*, Blekinge Institute of Technology, 1997.

[6] R. E. Johnson, "Frameworks=(components+ patterns)", *Communications of the ACM*, vol. 40, no. 10, pp. 39–42, 1997.

[7] M. Fayad and D. C. Schmidt, "Object-oriented application frameworks", *Communications of the ACM*, vol. 40, no. 10, pp. 32–38, 1997.

[8] L. Liu and E. Yu, "Designing information systems in social context: A goal and scenario modelling approach", *Information systems*, vol. 29, no. 2, pp. 187–203, 2004.

[9] J. Gordijn, H. Akkermans, and J. Van Vliet, "Designing and evaluating e-business models", *IEEE intelligent Systems*, vol. 16, no. 4, pp. 11–17, 2001.

[10] A. R. Golding and P. S. Rosenbloom, "Improving rule-based systems through case-based reasoning.", in *AAAI*, vol. 1, 1991, pp. 22–27.

[11] R. E. Johnson and B. Foote, "Designing reusable classes", *Journal of object-oriented programming*, vol. 1, no. 2, pp. 22–35, 1988.

[12] H. A. Schmid, "Systematic framework design by generalization", *Communications of the ACM*, vol. 40, no. 10, pp. 48–51, 1997.

[13] D. Parsons, A. Rashid, A. Speck, and A. Telea, "A" framework" for object oriented frameworks design", in *Technology of Object-Oriented Languages and Systems, 1999. Proceedings of*, IEEE, 1999, pp. 141–151.

[14] M. E. Markiewicz and C. J. de Lucena, "Object oriented framework development", *Crossroads*, vol. 7, no. 4, pp. 3–9, 2001.

[15] I. Kume, M. Nakamura, N. Nitta, and E. Shibayama, "A feature model of framework applications", in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2013 14th ACIS International Conference on*, IEEE, 2013, pp. 511–516.

[16] C. Nagl, F. Rosenberg, and S. Dustdar, "Vidre–a distributed service-oriented business rule engine based on ruleml", in *Enterprise Distributed Object Computing Conference, 2006. EDOC'06. 10th IEEE International*, IEEE, 2006, pp. 35–44.

[17] M. Ganzha and M. Paprzycki, "Implementing rule-based automated price negotiation in an agent system", *Journal of Universal Computer Science*, vol. 13, no. 2, pp. 244–266, 2007.

[18] M. Suwa, A. C. Scott, and E. H. Shortliffe, "An approach to verifying completeness and consistency in a rule-based expert system", *Ai Magazine*, vol. 3, no. 4, p. 16, 1982.

[19] A. Rajasekar, M. Wan, R. Moore, and W. Schroeder, "A prototype rule-based distributed data management system", in *HPDC workshop on Next Generation Distributed Data Management*, vol. 102, 2006.

[20] B. R. Manifesto, "The principles of rule independence./ronald g", *The Business Rules Group*, 2003.

[21] A. V. Zitzewitz, "Designing quality software: Architectural and technical best practices", ello2morrow Inc, Tech. Rep.

[22] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering", *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[23] B. DiCicco-Bloom and B. F. Crabtree, "The qualitative research interview", *Medical education*, vol. 40, no. 4, pp. 314–321, 2006.

[24] C. B. Seaman, "Qualitative methods in empirical studies of software engineering", *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[25] R. K. Yin, *Case study research: Design and methods*. Sage publications, 2013.

[26] C. Robinson, *Real world research: A resource for social scientists and practitioner-researchers*, 2002.

[27] V. Braun and V. Clarke, "Using thematic analysis in psychology", *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[28] R. Ellis-Braithwaite, R. Lock, R. Dawson, and B. Haque, "Towards an approach for analysing the strategic alignment of software requirements using quantified goal graphs", *arXiv preprint arXiv:1307.2580*, 2013.

[29] F. Dalpiaz, X. Franch, and J. Horkoff, "Istar 2.0 language guide", *arXiv preprint arXiv:1605.07767*, 2016.

[30] R. Schuster and T. Motal, "From e3-value to rea: Modeling multi-party e-business collaborations", in *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, IEEE, 2009, pp. 202–208.

[31] M. Harper and P. Cole, "Member checking: Can benefits be gained similar to group therapy?", *The Qualitative Report*, vol. 17, no. 2, pp. 510–517, 2012.

[32] K.-J. Stol and B. Fitzgerald, "Inner source–adopting open source development practices in organizations: A tutorial", *IEEE Software*, vol. 32, no. 4, pp. 60–67, 2015.

A

Appendix 1

A.1 Interview Questions

General questions

1. What is your current role in the company?
2. What is your background at the company?
3. Which products are you working with?
4. What is your experience with SFA? (As a user, developer etc)
5. What are perceived disadvantages of the framework, if any?
6. What are perceived advantages of the framework, if any?
7. What is the perceived value for Ericsson by using the framework?

If they have experience as a user

8. How much knowledge of the design rules do you have?
9. When using the framework, to what extent did you have to know the design rules to implement or use it?
10. Do you know if all design rules are enforced by the framework?
11. When working on a product using SFA, do you feel like the use cases of the product are supported by the framework?
    - IF NO:
    - What type of use cases are not supported?
    - Which ones should be supported according to you?
12. Did you notice any evolution of the framework? What major changes happened in the framework since the time you first used it?
13. Did the framework adapt and support to the evolution of the product?

If they have experience as a developer

14. How much did the framework evolve over time, what changes has happened since the start.
15. How do you determine which design rules, and which properties of them, to support in the framework?
16. Do you have domain use-cases as a basis for developing the framework?
    - IF YES:
    - How do you decide which domain use-cases to support in the framework?

If they are potential users

17. To your knowledge, is there a specific use case in your product that SFA needs to support?
18. How much knowledge of the design rules do you have?
19. What motivates you to use SFA?

20. What prevents you from using SFA?
21. Is it an issue to refactor and replace legacy code in order to use SFA?
22. If you are considering using it now, why not before?
23. Who makes the decisions of using SFA for your product?