Tool support for risk-driven planning of trustworthy smart IoT systems within DevOps

Andreas Thompson

Thesis submitted for the degree of
Master in Informatics: programming and networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2019

Tool support for risk-driven planning of trustworthy smart IoT systems within DevOps

Andreas Thompson

© 2019 Andreas Thompson

Tool support for risk-driven planning of trustworthy smart IoT systems within DevOps

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

The Internet of Things is rising in popularity across many different domains, such as build automation, healthcare, electrical smart metering and physical security. Many prominent IT experts and companies like Gartner expect there to be a continued rise in the amount of IoT endpoints, with an estimated 5.8 million endpoints in 2020. With the rapid growth of the devices, the physical nature of these devices, and the amount of data collected, there will be a greater need for trustworthiness. These devices often gather personal data and as the devices have relatively less computational power than other devices, security and privacy risks are greater.

With the IoT systems often operating in highly dynamic environments, the development of these systems should often be done in an iterative manner. DevOps is an increasingly popular agile practice which combines the development and operations of systems to provide continuous delivery. This is well suited for the development of IoT systems.

However, there is currently a lack of support for risk driven planning of trustworthy smart IoT systems within DevOps. This thesis investigates currently available tools and methods for the planning of trustworthy smart IoT systems within DevOps. We also propose a tool-supported method with the purpose of assisting developers in the planning phase of DevOps with identifying security and privacy risks, and executing risk assessment algorithms. Furthermore we facilitate automatic real-time security and privacy risk assessment through our custom made API.

Moreover we conduct a case study where we apply both our method and tool in a real-life smart home case. Based on our initial result we argue that our tool-supported method: is easy to use and understandable for developers, supports the planning of trustworthy smart IoT systems in the DevOps practice in terms of security and privacy risk assessment and it is appropriate for use in the DevOps practice in terms of adapting to new plans and flexible in response to changes in the system.

Contents

List of Figures

List of Tables

Acknowledgements

The present work was carried out at the University of Oslo, the Department of Informatics at the Faculty of Mathematics and Natural Sciences and SINTEF, Oslo, between 2018 and 2019.

I would like to thank my supervisors Ketil Stølen from the University of Oslo and Sintef and Gencer Erdogan from Sintef. They have provided me with this amazing opportunity and excellent support during my work with this master's thesis.

Thanks to my friends who gave encouraging words when I needed it the most, and who also helped me with styling small parts of the tool.

I would like to thank my parents, Keith Thompson and Ingrid Randen, for an amazing upbringing, filled with knowledge, patience, and most importantly love. I'm sure my father would have been proud of me now.

Thanks to my dear cohabitant Ingvil Aigeltinger, for always being there for me with love and support.

Finally I would also like to thank the University of Oslo for giving me this opportunity.

Chapter 1

Introduction

In this chapter, we present the motivation and background for our work and describe the problem addressed in this thesis. Further, we present our main contributions, as well as an overview of the thesis.

1.1 Motivation

As the number of Internet-of-Things (IoT) devices grows ever more rapidly, systems and networks become more complex, and the need for proper security becomes important. According to Leuker et al. [1], risks related to security, trust, privacy and identity management are major challenges in today's IoT systems. Assessing the quality and security risks of IoT systems however is not a simple task, and due to devices having constraints on cost, time to market and functionality developers of these devices often disregard this assessment [2]. Because IoT systems typically operate in highly dynamic environments, they need to be able to continuously evolve and adapt, to ensure and increase their trustworthiness. The DevOps software engineering culture and practice aims at shorter development cycles, increased deployment frequency and more dependable releases, which makes for a more agile and dynamic development process. However, there is a serious lack of support for trustworthy smart IoT systems in DevOps [3, 4].

Since the DevOps practice is a continuous loop of planning, developing, releasing and monitoring, automating and streamlining the process is key. By developing a software tool for risk assessment of the IoT system architecture to be used in the planning stage of DevOps, we can help ensure trustworthy execution of IoT systems, as well as reduce manual labour in the DevOps cycle.

Throughout this master thesis we will explore the different aspects of IoT, trustworthiness, DevOps and risk management. The goal is to research and analyse these aspects, gain insight into the current state of the art and to see how the different aspects relate. This thesis will serve as a foundation to develop a risk-driven guidance tool to architects, developers, feasibility study engineers

and other potential stakeholders in the assessment of IoT system architecture to ensure trustworthy execution of IoT systems.

1.2 Thesis Overview

This thesis is organised in these seven chapters:

Chapter 1 - Introduction is divided into the following sections: Section 1.1 goes into the motivation for conducting the thesis. Section 1.2 provides an overview of the thesis. Section 1.3 provides a background and state of the art of Internet of Things, DevOps, Trustworthiness and Risk Management. Section 1.4 specifies the thesis statement and success criteria. Section 1.5 presents the main contributions of this thesis.
Chapter 2 - State of the Art is divided into the following sections: Section 2.1 describes Security and privacy risk management and related standards. Section 2.2 describes how security and privacy risk management is done in the DevOps practice. Section 2.3 presents examples of tool used for risk driven planning and describes these. Finally Section 2.4 describe how our thesis contributes over the state-of-the-art.
Chapter 3 - Research Method describes how the research in this thesis has been conducted. The chapter is divided into the following sections: Section 3.1 presents a technology research method aimed at improving or producing new artefacts. Section 3.2 presents categories of evaluation strategies, what they are and what properties they have. Section 3.3 provides discussion of which evaluation strategies are appropriate for this thesis. Finally, Sections 3.4, 3.5 and 3.6 describe the evaluation strategies applied in this thesis.
Chapter 4 - Tool-supported Method for Risk-driven Planning of Trustworthy Smart IoT Systems is divided into the following sections: Section 4.1 describes our tool. Section 4.2 details how our tool satisfies the criteria of the DevOps practice, and provides functionality to support this practice. Section 4.3 describes our tool-supported method for IoT orchestration, identifying security and privacy risks related to the orchestration, and executing risk assessment algorithms.
Chapter 5 - Applying Our Tool-supported Method In A Smart Home Case is divided into the following sections: Section 5.1 provides motivation for our case. Section 5.2 presents the smart home case and establishes the context of the case. In Section 5.3 we perform a data flow analysis of the case. In Section 5.4 we perform privacy and security risk modelling. In Section 5.5 we translate the risk models to executable risk assessment algorithms. In Section 5.6 we execute the risk assessment algorithms. Finally, in Section 5.7 we demonstrate how the tool and method supports changes to the system.
Chapter 6 - Discussion provides a discussion of our thesis by discussing the development process, as well as our achievements with respect to our success criteria.

Chapter 7 - Conclusion concludes the thesis, provides directions for future work and addresses threats to validity.

1.3 Background

In the following subsections we look into the background of Internet of Things, DevOps, Trustworthiness and Risk Management.

1.3.1 Internet of Things

As currently defined by ISO/IEC, the Internet of Things (IoT) is "an infrastructure of interconnected objects, people, systems and information resources together with intelligent services to allow them to process information of the physical and the virtual world and react" [5]. These interconnected objects are often called 'objects', 'things' and 'devices', and are linked with services and web-bases systems via the Internet. The use of these devices alongside intelligent services such as clouds and fog computing [6], create larger, connected systems and allow for many tools and services that benefit both economic actors, as well as end users. The term Internet of things was first coined by Kevin Ashton in 1999 and was used in the context of supply chain management [7]. The definition has however over the years covered a wider range of applications like healthcare, smart cities and homes, transport, enterprises and others [8–10]. In 2017 it was predicted that there will be 20 billion connected things by 2020, with endpoint spendings exceeding 2.9 trillion dollars [11].

These devices all come from various vendors, utilizing different communication protocols, varying data formats and processing mechanisms, which, alongside the varied environments these devices are found, makes for heterogeneous and complex systems. There are currently ongoing efforts for standardizations, but there has yet to emerge a widely used standard [12].

The devices are also limited with regards to computational power, battery lifetime and storage, and often have the need to have global connectivity and accessibility. Because of these factors, the devices and systems have a huge attack surface, and security in IoT systems becomes a big issue [10, 13].

1.3.2 DevOps

DevOps is often characterized as a culture or practice. It's a collaboration between development and IT operations, with the aim of providing more agile development cycles, and continuous deployment. The DevOps toolchain is a set or combination of tools that is used throughout the development cycle to help aid the delivery, development and management of applications and systems. Figure 1 shows the stages in a DevOps toolchain.

The DevOps ideals builds upon other agile software development practices such as Scrum [14], Kanban [15], Lean Development [16] and Extreme Programming [17]. These practices all fall under the agile software development

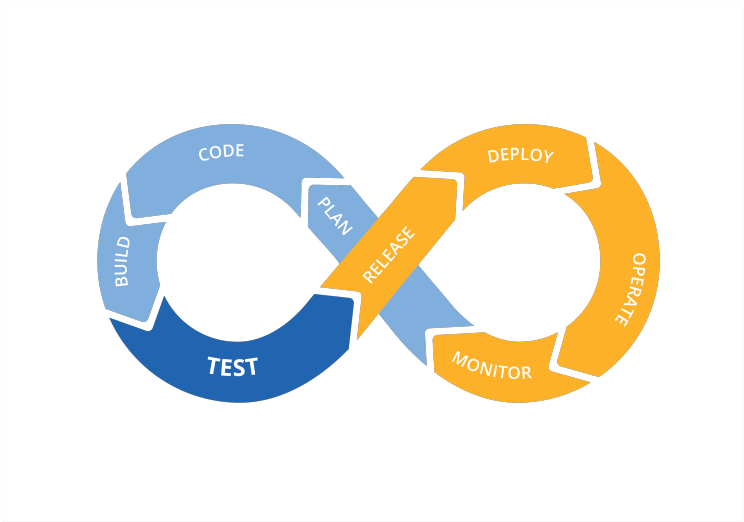


Figure 1.1: The DevOps development cycle.

category/approach, which has been popularized by Beck et al. [18]. The values of the agile software development paradigm is [18–22]:

- To be iterative, incremental and evolutionary
- Efficient and face-to-face communication
- Short feedback and adaptation cycles
- Focus on quality

The reason DevOps is arising with increasing success over other agile practices, is because of the large number of tools and information systems needed to manage data and processes, and the fact that these tools are becoming more essential to the development process. In many organizations, the development teams and operational teams are often separated, which causes communication difficulties and issues with an efficient engineering environment [23]. There are also often conflicts between the two teams, as the developers want change, and the operations want stability [24].

1.3.3 Trustworthiness

In computing literature, trustworthiness is defined as the system property that denotes the degree of user confidence that the system will behave as expected [25]. Trust is a complicated concept which relates not only to security but also to characteristics such as reliability, dependability and other characters of an entity. The social concept of trust is context-dependent. It is natural for A to trust B in certain contexts, mistrust B in some contexts, and neither

trusts nor mistrusts B in some other contexts. With respects to the use of IoT, only contexts for trusting them matters, as the contexts where there is no evidence of trust will be ignored. The most widely accepted definition of context and context-awareness in the research community was introduced by Abowd et al. [26]. In short, context is defined as any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computational object. Context-awareness is defined as the use of context to provide task-relevant information and/or services to a user.

When it comes to IoT systems, trustworthiness refers to the preservation of security, privacy, reliability and resilience of systems [27]. Throughout this thesis we will be focusing on security and privacy.

Trustworthiness and context together play an important role. Generally, an IoT system contains three layers, a physical layer that perceives physical environments, a network layer and an application layer. For a complete system to be trusted, it is not sufficient that individual layers of the systems are trusted, the cooperation between the layers need to be reliable as well [28]. For an IoT system to state the contexts in which it is available, it needs to be aware [29]. The two main types of awareness is self-awareness and context-awareness.

1.3.4 Risk Management

Before we dive into risk management, we should define risk. There are many definitions of risk, but throughout this essay we will be using the ISO 31000 standard. According to ISO 31000 risk is the combination of the consequences of an event with respect to an objective and the associated likelihood of occurrence. According to the same standard, risk management is "coordinated activities to direct and control an organization with regard to risk" [30]. The risk management process builds on basic principles which are necessary to understand. The ISO 31000 lists 11 of these principles. The process consists of risk assessment, monitoring and review, and communication and control [31]. Risk assessment is again broken down into three parts, risk identification, risk estimation and risk evaluation. The risk identification process consists of finding, recognizing and describing risks, which involves identifying sources of risk, areas of impact, events, their causes and their potential consequences. Risk estimation provides decisions on whether risks need to be treated, and on the most appropriate risk treatment strategies and methods [30]. Finally, risk evaluation is the process of comparing the results of risk estimation with risk criteria to determine whether the risk and/or its magnitude is acceptable or tolerable.

1.4 Thesis Statement and Success Criteria

1.4.1 Thesis Statement

As the problem of this thesis is concerned with creating both a method for risk-driven planning of IoT systems in the planning phase of the DevOps practice,

and tool support for this method, there are several issues to discuss. The tool needs to appeal to developers and other potential stakeholders. It should also be beneficial and usable, for even if the mechanics behind the tool are good, it won't matter if there are no users. The tool and method should support frequent changes to system.

1.4.2 Success Criteria

For the sake of fulfilling the requirements mentioned in the previous section, as well as the overall aim of the thesis, we need to identify and establish the success criteria:

Success Criterion 1. *The tool and method must be easy to use and understandable for developers.*

As the main beneficiaries of this tool are developers, the tool must be comprehensible, and the features should therefore be properly expressed.

Success Criterion 2. *The tool-supported method should support the planning of trustworthy smart IoT systems in the DevOps practice in terms of security and privacy risk assessment.*

By this we mean that the tool must provide the developers with risk levels based on the risk assessment. Such that the developers may prioritize the areas of the system that needs to be treated in order to mitigate the privacy and security risks the system is exposed to.

Success Criterion 3. *The tool and method should be appropriate for use in the DevOps practice in terms of adapting to new plans and flexible in response to changes in the system.*

For the tool to be efficient in a DevOps environment, it is important that it follows the agile paradigms. Features such as importing and exporting information will help the iterative nature of the practice.

1.5 Contribution

This thesis presents four kinds of contributions. First, it presents an adaptation of existing methods to create a refined method with the purpose of assisting developers in the planning phase of DevOps with identifying security and privacy risks, and executing risk assessment algorithms. Second, we develop a tool to support our method. Third, it provides a thorough case that aims to demonstrate the feasibility of the tool as well as the method in which the tool is used. Finally, we also provide a state of the art of concepts relevant to this thesis.

Chapter 2

State of the Art

In this chapter we provide a summary of the state-of-the-art in the domains of security and privacy risk management, DevOps and tool support for risk-driven planning.

2.1 Security and Privacy Risk Management

Security and privacy risk management is the process of risk management specialized towards security and privacy.

For security risk management, the ISO/IEC standard 31000 and the ISO/IEC 27005 are both well established and widely used. ISO 31000 is named "Risk management – Principles and guidelines" and provides generic guidelines on risk management. ISO 27005 is named "Information technology – Security techniques – Information security risk management systems" and provides guidelines for information security risk management in an organization. As illustrated in Figure 2.1, the risk management process provided by ISO 31000 consists of five steps: context establishment, risk assessment, risk treatment, monitoring and review, and communication and consultation.

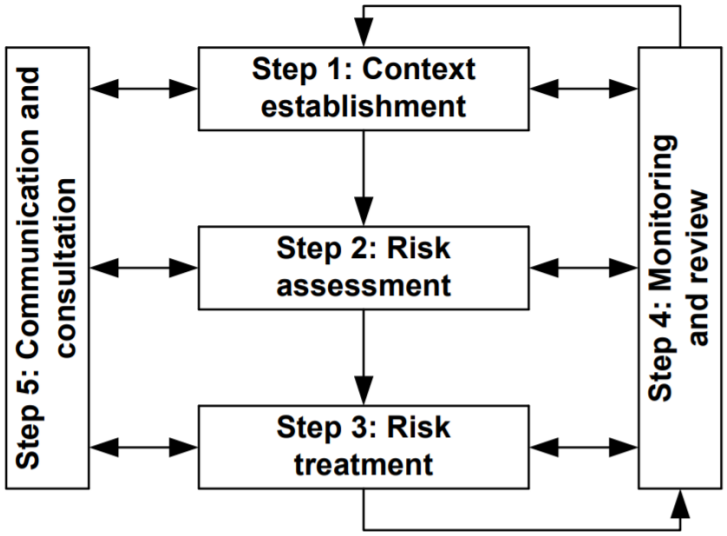


Figure 2.1: Risk management process adapted from ISO 31000

ISO 31010 is a supporting standard for ISO 31000, and provides guidance on selection and application of systematic techniques for risk assessment. It also explains how risk identification, risk analysis, and risk evaluation should be carried out.

ISO 27005 is a specialization of ISO 31000 and is designed to assist the satisfactory implementation of information security based on a risk management approach. The security risk management process provided by ISO 27005 differs from the general risk management ISO 31000, as it has a larger emphasis on iterating the risk assessment process, as well as the risk treatment activities. Figure depicts the security risk management process adapted from ISO 27005.

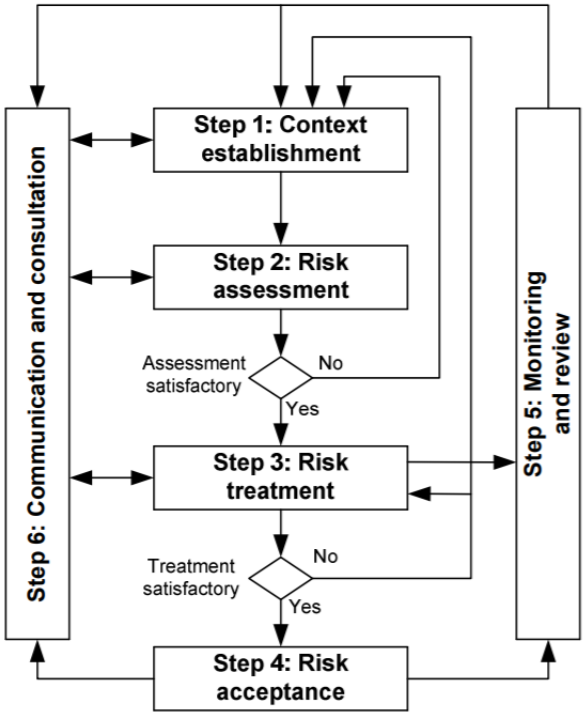


Figure 2.2: Risk management process adapted from ISO 27005

Privacy risk management is not new, and has been done similarly to security risk management. Back in 2004, Hong et al. [32] presented a model for both privacy risk analysis, and privacy risk management.

Today there are ongoing works to improve privacy risk management within organizations. NIST [33], the National Institute of Standards and Technology, are working on a tool for improving privacy through enterprise risk management [34]. This tool builds on their cybersecurity framework, and may be used either independently, or in conjunction with the cybersecurity framework. The framework is intended to be widely usable by organizations of al sizes, and agnostic to any particular technology.

Some organizations have a robust grasp of privacy risk management, yet there is still a widespread lack of common understanding of the topic [34]. There is also an ISO standard for privacy, namely the ISO/IEC 29100:2011 [35]. ISO/IEC 29100:2011 presents a privacy framework which specifies a common privacy terminology, and helps organizations define their privacy safeguarding requirements by:

- Specifying a common privacy terminology.
- Defining the actors and their roles in processing personally identifiable information.

- Describing privacy safeguarding requirements.
- Referencing known privacy principles.

2.2 DevOps and Security and Privacy Risk Management

Security in DevOps is a hot topic. A survey done by the HPE Security Fortify team [36] from 2016 show that many believe security should be a part of DevOps, but security is still not included in most DevOps programs. Gartner also estimates that less than 20% of "enterprise security architects have engaged with their DevOps initiatives to actively and systematically incorporate information security into their DevOps initiatives" [37]. Puppet [38], a company that delivers and operates software for infrastructure, has been doing DevOps surveys since 2012. In their newest survey [39] they surveyed nearly 3000 participants, and their key findings were:

- Doing DevOps well enables you to do security well.
- Integrating security deeply into the software delivery lifecycle makes teams more than twice as confident of their security posture.
- Integrating security throughout the software delivery lifecycle leads to positive outcomes.
- Security integration is messy, especially in the middle stages of evolution.

Part of their conclusion was that security needs to be built into the entire software delivery lifecycle, so it is not regarded as something "extra".

The term DevSecOps has been proposed to try help more DevOps users integrate security. This however treats security as an "optional" addition to DevOps rather than being an integral part of it [36]. In [40], Myrbakken et al. perform a literature review and find that the inclusion of security in DevOps offer several benefits:

- The inclusion of security experts from the start of the process makes it easier to plan and execute integration of security controls.
- This then helps lowering risk and time spent on errors, as well as making it easier to understand risks.
- The ability to measure and monitor security flaws early on decreases the cost of making mistakes, finding them and fixing them.

From their sources, Myrbakken et al. concludes that "Performing risk assessments from the first planning stage and continuously before every iteration is important as a way to prioritize risks, examine controls already in place and decide which are needed going forward"

The inclusion of security however also comes with challenges. One of the bigger challenges is that traditional security methods impairs the speed and agility of DevOps.

2.3 Tool-Support for Risk Driven Planning

For state of the art tool support for risk-driven planning of IoT devices and services, the choices are limited. There have been a lot of research on decision support systems (DSS) in general such as Zeleny et al. [41] and Bi et al. [12]. Smrati Gupta et al. [42] present an analysis of the state-of-the-art in decision support systems, and critical shortcomings in the existing tools. While not all their work is relevant to this thesis, the research is valuable to us. Currently there does not exist any clear mechanisms to collect data about different characteristics of available IoT devices and services, while ensuring trustworthiness and up-to-dateness. There is also a lack of tools to help users link the characteristics of IoT devices and services with the actual risks for their applications or infrastructures.

When it comes to tools, there are many available for use in the DevOps practice. For the different stages there are different tools, popular ones are Jenkins [43] or Maven [44] for testing and New Relic [45] for monitoring. These tools are valued as automation, monitoring and continuity are key in the DevOps practice. The planning phase does not have a lot of dedicated tools, there exists a few such as Jira [46], but teams often utilize backlogs and kanban boards to better gain an overview of what needs to be done when [24, 47]. Overall there is a lack of tool-support for risk-driven planning in DevOps [48].

There also exists general privacy and security risk management tools such Riskwatch [49] and ISRAM[50]. The issues with these are that they are both quantitative and not qualitative.

2.4 Our Advancement Over the State-of-The-Art

In this chapter we have discovered that security and privacy is not yet an integrated part of a typical DevOps process. This is both due to DevOps architects not properly applying security methods, but also because typical security methods slow down the DevOps process.

While there exists tools for use in other phases of DevOps, the planning phase does not have any specific tools to support it [48]. There exits general tools for both privacy and security risk however such as RiskWatch and ISRAM.

For our contribution we create a method for risk driven planning in the planning phase of DevOps, which support the agile nature of the process. We also develop a tool to support this method which can help automate some parts of the process.

Chapter 3

Research Method

Previously we described the main topic of our proposed thesis, and gave an overview of what is to be developed. We then defined a set of success criteria that our thesis aims to fulfil. In this section, we will describe the research method to be applied in the thesis, to fulfil these success criteria.
Research is defined by Merriam-Webster as:

> Investigation or experimentation aimed at the discovery and interpretation of facts, revision of accepted theories or laws in the light of new facts, or practical application of such new or revised theories or laws [51].

So what we seek is information that will either add knowledge, modify existing knowledge or find new uses for already existing knowledge. Researchers start by formulating hypotheses as a starting point for further investigation. They then test these with verifiability- or falsifiability-oriented experiments and observations. This testing is referred to as evaluation. Even if the hypotheses are strengthened through evaluation, it can never be ultimately proven. This approach is usually called *the scientific method*.

The approach is defined by Solheim and Stølen as classical research [52], and they define it as follows:

> Classical research is research focusing on the world around us, seeking new knowledge about nature, space, the human body, the society, etc. The researcher asks: What is the real world like?

This type of research is heavily rooted in what Solheim and Stølen call *basic research*, defined as "research for the purpose of obtaining new knowledge". The main steps of basic research is *problem analysis, innovation and evaluation.*

However, through our proposed thesis we will be using what Solheim and Stølen call technology research. This approach is more concerned with asking questions regarding technology, and finding better ways of solving practical problems. Contrary to basic research, technology research is mainly rooted

in what they call *applied technology* defined as "research seeking solutions to practical problems". According to Solheim and Stølen an artefact is an object manufactured by humans. The technology researcher aims to create and improve artefacts. Similarly to basic research, technology research is an iterative method that build on three main steps, *problem analysis, research-based design and evaluation*. This is similar to design science, where the objective is to design artefacts to interact with a problem context in order to improve something in that context [53]. Unlike classical research, which aims to understand reality, design science aims to develop artefacts that serve human purposes. Design science is also an iterative method that consists of three steps, which correspond to the steps in technology research. These steps are *problem investigation, treatment design and treatment validation.*

In the following sections, we will further explain technology research, give an overview of evaluation strategies, and give an overview of the selected evaluation strategies for our thesis.

3.1 Technology Research

A technology researcher is concerned about creating new artefact, or improving existing ones. Examples of this are, a new robot, an improved algorithm, a new construction method, etc. To start the researcher collects requirements concerning the artefact. This is the problem analysis and in contrary to classic research, where we ask *What is the real world like?*, which is focused on real world phenomena, we instead ask *How can we improve artefacts, or create new ones that benefit humanity in solving practical issues?* After the researcher has collected the requirements, he goes on to making an artefact that is supposed to satisfy these requirements. This can be tough as nobody has done it before, but the researcher must assume it is feasible. The overall hypothesis is therefore: *The artefact satisfies the need*

The artefact produced by the technology researcher is not always complete from a users point of view. We often create what is known as a functional prototype. When the researcher has produced this artefact, it has to undergo an evaluation, examples of evaluation strategies are given in Section 3.2. If the evaluation is successful, the researcher may argue that the artefact satisfies the need. However, if the results are not satisfactory, the researcher may have to adjust the requirements, and build a new artefact. Technology research is therefore an iterative process. In Figure 3.1 we illustrate the process. Positive evaluations confirm the hypothesis, but do not prove anything. Negative evaluations impairs the hypothesis, but also stimulates new iterations in the cycle.

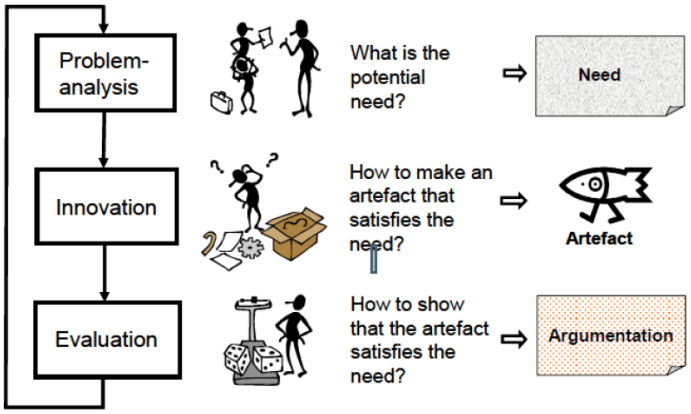


Figure 3.1: Method for technology research.

3.2 Evaluation Strategies

Evaluations are used to discover whether the predictions we had are true. There are different ways to do this, but we base ourselves on the most common strategies as explained by McGrath [54]. Figure 3.2 depicts the different strategies, as well as the three desired properties. These properties are:

- **Generality**
  A measure of the validity of results across populations.
- **Precision**
  The precision of measurement of the acquired results, and control of external variables that are not part of the study.
- **Realism**
  To what degree the evaluation reflects realism (if it was performed in a realistic context)

Even though one would want to chose a strategy that maximises all three properties, McGraths argues that it's not possible, and every research strategy is flawed. Different strategies have different flaws, one should therefore chose multiple strategies that complement each other, to attain acceptable levels for each property.

We will not be going over all the common evaluation strategies, but we provide a short description of each of the strategies depicted in Figure 3.2.

- **Field studies** make direct observation of "natural" ongoing systems, while disturbing on those systems as little as possible.
- **Laboratory experiments** attempt to recreate the "essence" of some systems in a context where the researcher has control of most extraneous

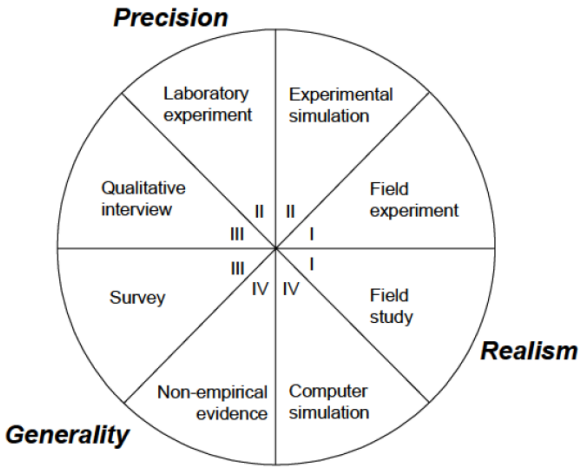


Figure 3.2: Evaluation strategies adapted from McGrath.

features of the situation, in order to maximize the essential features with precision.

- **Field experiments** are field studies where the researcher deliberately manipulates some features whose effect is to be studied.
- **Experimental simulation** is a laboratory study where one tries to recreate systems with a large degree of control, and the possibility to isolate the variables to be examined.
- **Surveys** are used to get information from a broad and carefully selected group of informants.
- A **Qualitative interview** is a collection of information from a few selected individuals. The answers are more precise than those of a survey, but cannot be generalized to the same degree.
- **Computer simulation** is operating on a model of a given system.
- **Non-empirical evidence** is a theoretical approach based on argumentation with logical reasoning.

These strategies are divided into four groups, as illustrated in Figure 3.2:

I The evaluation is performed in a natural environment.

II The evaluation is performed in an artificial environment.

III The evaluation is independent of environment.

IV The evaluation is independent of empirical measurements.

In addition to the above evaluation strategies, Wieringa [53] and Zelkowitz et al. [55] point out the following additional strategy.

- **Case Study** is an empirical inquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified [56].

3.3 Selection of Appropriate Evaluation Strategies

To figure out which evaluation strategies to choose, we should re-examine the success criteria we established in Section 1.4.2:

1. *The tool and method must be easy to use and understandable for developers.*
2. *The tool-supported method should support the planning of trustworthy smart IoT systems in the DevOps practice in terms of security and privacy risk assessment.*
3. *The tool and method should be appropriate for use in the DevOps practice in terms of adapting to new plans and flexible in response to changes in the system.*

Success criterion one is mostly concerned with the users perception of the tool. To what extent the tool can sufficiently help users decide on devices and services to create IoT systems is mainly based on the criteria given by the user. To evaluate success criterion one, we can perform a case study, which is further explained in section 3.4. Success criterion 2 and 3 are more concerned with the tool and methods effectiveness. With this in mind, there are multiple strategies to consider. We can in some extent use non-empirical evidence to support the tools decision making algorithm, and gain generality. We can also benefit from a strategy called prototyping to gain a better understanding of the requirements of the artefact. Prototyping would fall somewhere in between experimental simulation and field experiment, as we would try to simulate a potential user, while controlling certain factors for study. In Section 3.5 we further explain prototyping. To gain more precision as to the performance of the tool, we could simulate the tool in a controlled environment using experimental testing. We cover this in Section 3.6.

3.4 Case Study

Yin defines case study as:

> an empirical enquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident [57].

Which fits well in software engineering. According to Yin [57], "a case study allows investigators to focus on a “case” and retain holistic and real-life perspective." For example, when studying a method for security risk assessment, a software development life cycle, or organizational and managerial processes. The results of a case study can help determine to what extent an artefact is useful, comprehensible and scalable.

Runeson et al. concludes in [56] that:

> Case studies offer an approach that does not require a strict boundary between the object of study and its environment. Case studies do not generate the same results on, for example, causal relationships, as controlled experiments do, but they provide a deeper understanding of the phenomena under study.

3.5 Prototyping

A prototype is an initial version of a system [58], that represents the artefact created from requirements established initially in the problem analysis. The process of prototyping is concerned with writing programs for the purpose of learning about their optimal design and construction. The method can help us figure out what are the strengths and weaknesses of our tool early in development, and discover new requirements or success criteria. Prototyping is an iterative approach, and one can end up producing several prototypes to achieve a satisfactory understanding of the requirements.

3.6 Experimental Testing

The experimental testing, with regards to the evaluation strategies discussed in Section 3.2, aims to achieve precision. Experimental testing involves isolating certain variables to see what effects it has on the result. Tichy et al. [59] urges computer scientists to perform more experimental testing, and argues that experimentation can provide the following benefits:

- Experimentation can help build a reliable base of knowledge and thus reduce uncertainty about which theories, methods, and tools are adequate.
- Observation and experimentation can lead to new, useful, and unexpected insights and open whole new areas of investigation. Experimentation can push into unknown areas where engineering progresses slowly, if at all.
- Experimentation can accelerate progress by quickly eliminating fruitless approaches, erroneous assumptions, and fads. It also helps orient engineering and theory in promising directions.

Experimental testing can potentially take a lot of time, and this has to be taken into consideration.

Chapter 4

Tool-supported Method for Risk-driven Planning of Trustworthy Smart IoT Systems

In this chapter we explain our method for Tool-supported risk-driven planning of trustworthy smart IoT systems.

in Section 4.1 we describe the tool and the development of the tool. In Section 4.2 we provide information on how to properly apply our tool in the context of DevOps. In Section 4.3 we present our method for risk-driven planning of trustworthy smart IoT systems.

4.1 Tool Development

The tool we are developing is a combination of architectural modelling, and execution of risk assessment algorithms based on live data. The tool consists of two main components, namely the diagram editor, and the CORAS-DEXi module.

The diagram editor serves to model the system in focus with regards to communication between devices and services. It achieves this by allowing users to add and remove devices and services (depicted as blue squares with the device/service name), and connecting them up based on how the system is set up. Throughout this chapter we will be using an example of an SQL database and a server. Figure 4.1 depicts the diagram editor with the server and SQL Database communicating.

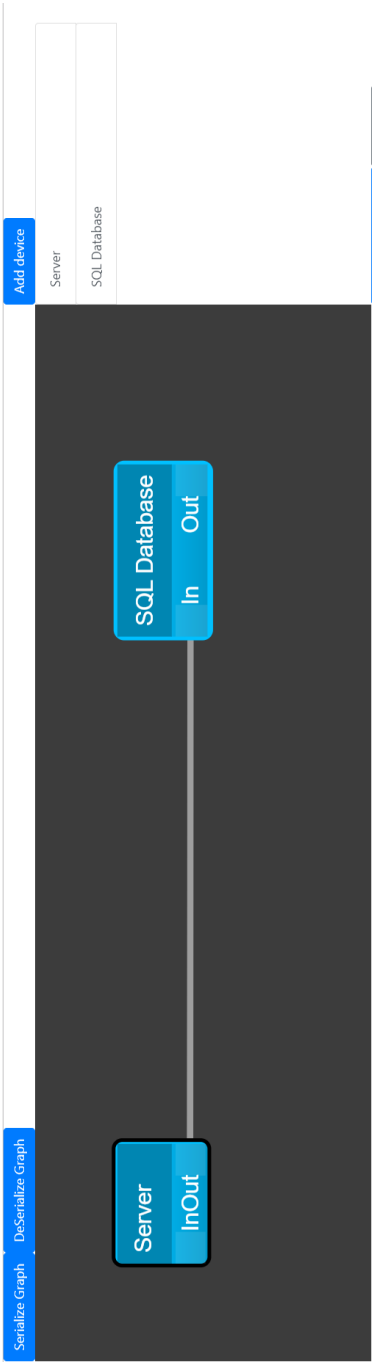


Figure 4.1: The diagram editor showing communication between a simple server and an SQL database

Each device/service has and "In" and "Out" node, which are used as a way to connect the devices and services, and there is no difference if the server is connected through its "Out" node or its "In" node. Above the diagram editor we have the Serialize and DeSerialize buttons, which are used to export and import the diagrams respectively. On the right hand side of the diagram editor we have the list of devices and services, with a button for adding new devices/services. A closeup of the list is depicted in Figure 4.2.

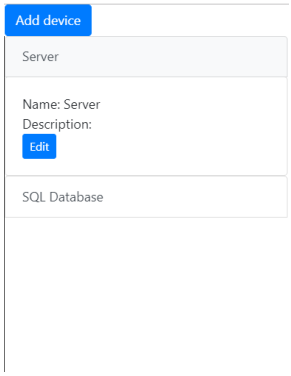


Figure 4.2: The list of devices and services in use in the current diagram

The second part of the tool is the CORAS-DEXi module which consists of the visualizer and the CORAS-Dexi list. This module is depicted in figure 4.3. The visualizer shows the user-imported CORAS diagram (more about CORAS in Section 4.3.3) alongside the inputs for the execution of algorithms, and similarly to the diagram editor, we have a list on the right hand side which includes he relevant CORAS-DEXi combinations.

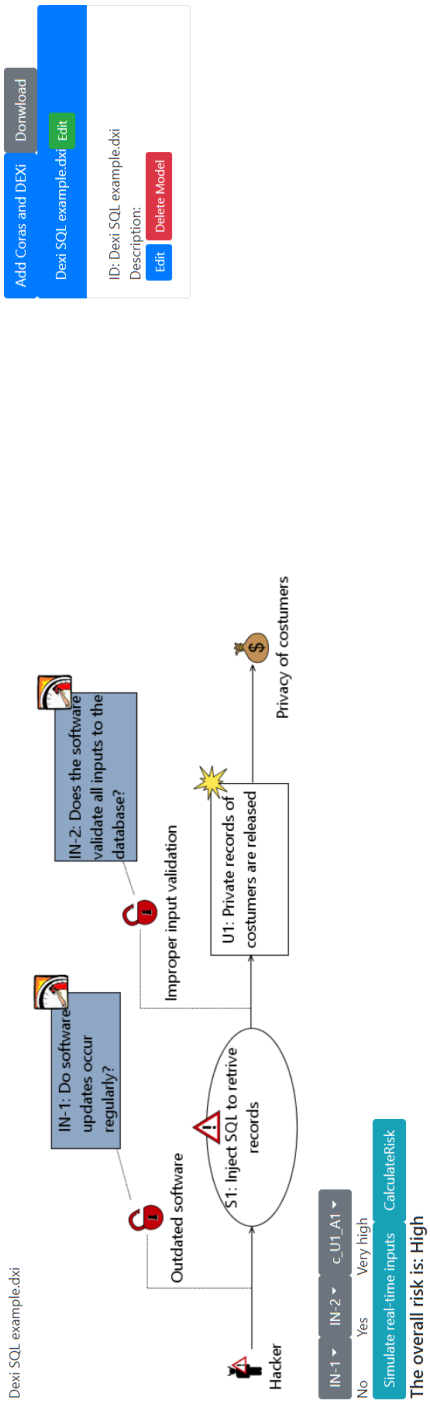


Figure 4.3: The CORAS-DEXi module for the server and SQL database example

The tool we develop through this thesis comes with constraints with regards to time. To create a functional and usable tool with this in mind we must rely on existing functionality such as libraries and frameworks developed by others. By building on existing functionality we reduce the amount of time necessary to develop the tool, and can focus on functionality required for the tool to be successful. To avoid having to buy licenses to use existing software, we consider resources which are open-source and non-proprietary. This also means that our tool can be open source, and allows other developers to further provide maintenance or additional functionality.

The tool we are developing will be a so called web-application, and is built on the React framework. The tool consists of a front-end using React and Typescript, and a back-end based on java, Spring Boot and MongoDB cloud. We go further into detail of the tool, and choices regarding frameworks and technologies used in Section 6.1.

4.2 Placing Our Tool in DevOps

The aim of our tool is to help support risk driven planning in the DevOps development cycle. This development cycle consists of the following steps: Planning, coding, building, testing, release, deployment, operation and monitoring. The cycle then repeats iteratively. The tool is to be used continuously in the planning phase of the DevOps cycle. Figure 4.4 shows the DevOps cycle, and how the planning fits into this cycle. Planning is the first step of every cycle, and in this the aim is to define criteria and functionalities to be fulfilled by the end of each phase. Planning is usually done without the use of distinct tools, in short iterations, and teams are planning with high-level objectives in mind. With this in mind, our tool-supported is suited for quick iterations, with a high-level of abstraction. The tool supplies a simple to use, high level modelling interface, and our method builds on use-case models, CORAS, a quick and iterative method for risk analysis, built with ISO standards in mind.

Our method is focused on trustworthiness, with a focus on privacy and security risks, which is often overlooked in DevOps cultures today [39, 40]. The tool and method also helps by making use of automatic monitoring for the use of risk planning, once deployment is complete, and monitoring is in use.

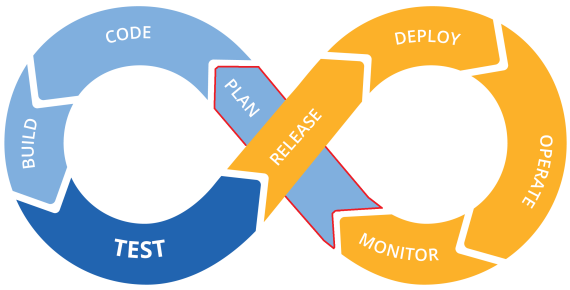


Figure 4.4: The DevOps Planning phase

4.3 Method for Risk-Driven Planning of Trustworthy Smart IoT Systems

The method consists of five steps depicted in Figure 4.5. Step 1 is context establishment. In this step the aim is to establish a context for a given IoT system, which provides the next steps with input. The output of step 1 is a relation model and use-case models. The relation model describes communication and high-level data flow between devices and services. While the use-case models are based on the relation model and describes scenarios of how the system is to be used.

In step 2 we create data-flow diagrams, we use relation models and use cases created from step one to create data flow diagrams that provide information such as information flow, and we use privacy policies to gather information related to privacy and security risks. We also use tables to structure this privacy and security related information.

Step 3 is privacy and security risk modelling. In this step we use the CORAS approach [60] to create graphical risk models based on the data-flow diagrams created in the previous step in order to capture privacy and security risks the target of analysis is exposed to.

These CORAS diagrams are then the input for step four, translating risk models to executable algorithms. Here we use DEXi [61] to translate our CORAS diagrams into executable algorithms in the form of .DEXi files.

In the final step, executing risk assessment algorithms using our tool, we use CORAS diagrams and the corresponding executable DEXi algorithms to create a depiction of the overall risk, dependent on given input from the user,

or gathered automatically from a monitor.

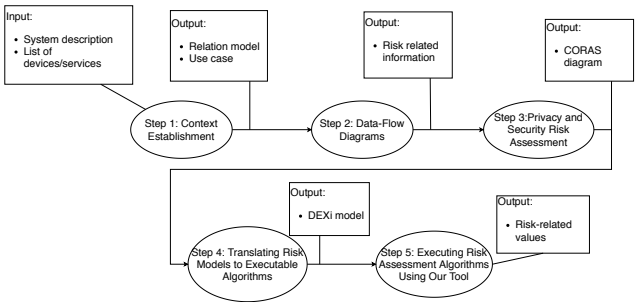


Figure 4.5: Steps of our tool-supported method

4.3.1 Step 1 - Context Establishment

The first step is context establishment which is the preparatory step for the subsequent activities. In this step we aim to establish the context following the principles of ISO31000 and ISO 27005. This is primarily done by having a meeting with the stakeholders/customers, and defining what devices and services to be used, as well as possible goals and assets the stakeholder/customer may have. In addition to establishing the general context, we also aim to provide all the input which is needed for the following steps. This step is crucial as the outcomes guides the rest of the process, and therefore has a major impact on the overall success. Context establishment is achieved by planning out which devices and services will be in use, and how they communicate. This can be done by creating a simple relation model using our tool of each of the devices and services, and how they relate to one another. Throughout this chapter, we will be using an example where a big company, aptly named "Big Company", needs help in planning a trustworthy smart system. Big Company owns two devices:

- A server
- An SQL database

With the two devices a relation model is created and depicted in Figure 4.6.

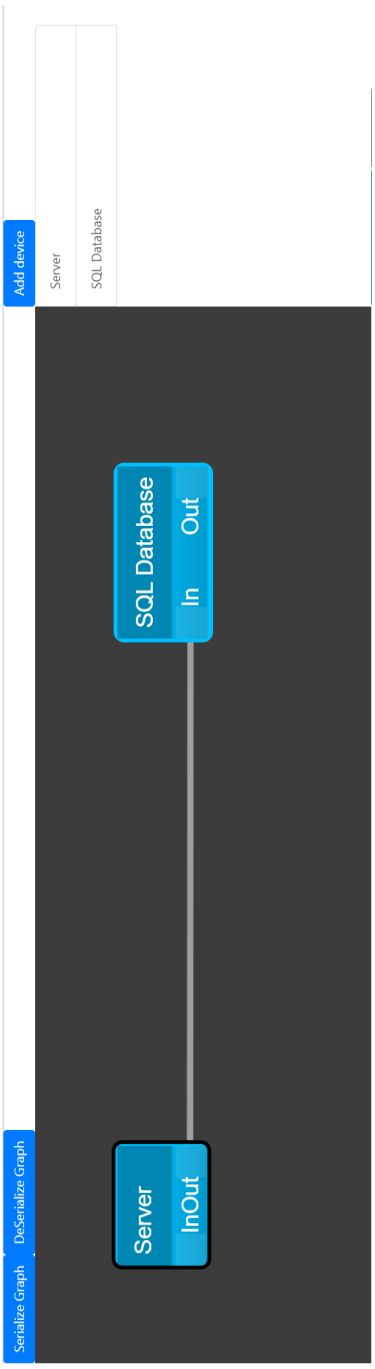


Figure 4.6: The relation diagram showing communication between the server and the SQL database

Once a model is created, we create use cases based on this model. The creation of use cases is done so that information flow between components is easier to understand, and to create input for the next step, creating data flow diagrams. A typical use case is depicted in Figure 4.7.

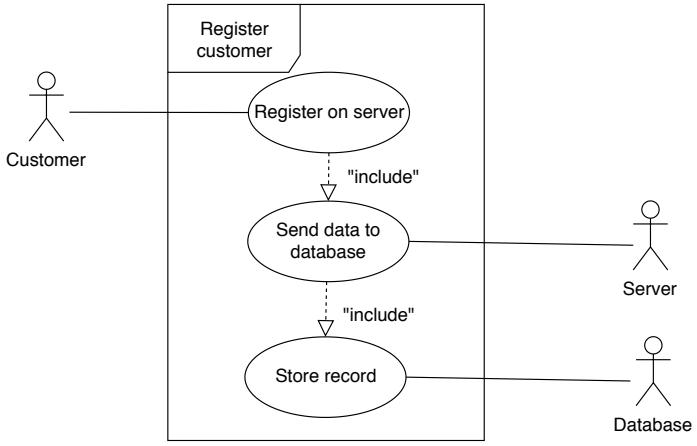


Figure 4.7: An example use case for registering a new customer

To help refining the objectives of Big Company, asset diagrams are used. Asset diagrams are another type of diagram provided by CORAS [60], which help both Big Company and the user of the method in clarifying and understanding what Big Company values. This is important as this is what motivates the conduction of risk analysis in the first place, and helps in getting an overview of threats and risks in later step 3. Big Company values the privacy of their customers, and an asset diagram of this is depicted in Figure 4.8.

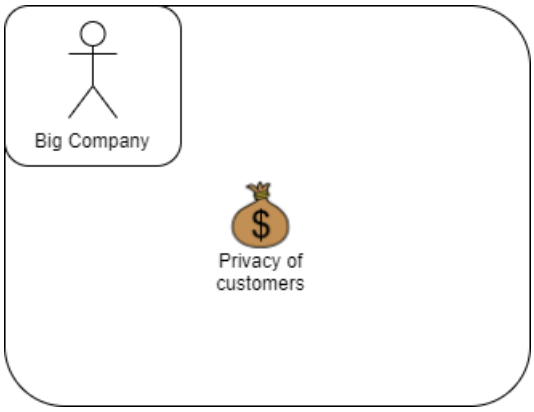


Figure 4.8: An example Asset diagram

With the assets defined, we create scales for consequence and likelihood. Defining the scales the last step of context establishment. One consequence scale is typically defined for every asset, for example monetary loss, however, this can be challenging where it may be hard to know the economic implication of risks. Therefore iy may be more useful to define consequence scales for each asset individually. The consequence scale is either quantitative or qualitative.

For the asset in Figure 4.8, we define the following scale:

Consequence	**Description**
Very High	More than 60% of records including sensitive information of customers is leaked
High	More than 30% of records are leaked, including some with sensitive data of customers
Medium	More than 15% of records are leaked, very few including sensitive information
Low	More than 5% of records are leaked, none including sensitive information
Very low	Less than 5% of records are leaked, none including sensitive information

Table 4.1: The consequence scale defined for our example asset

It is also necessary to create a likelihood scale. This scale differs from the consequence scale as it is used not for the assets, but for unwanted incidents and threat scenarios, more on this later in step 3. Similarly to the consequence scale it may either be qualitative or quantitative. Table 4.2 defines an example likelihood scale.

Likelihood	**Description**
Very High	Happens more than twice times a year
High	Happens between once a year to twice a year
Medium	Happens once every 2 years
Low	Happens once every 5 years
Very low	Happens once every 10 years

Table 4.2: An example likelihood scale

In Step 1: Context Establishment, our tool provides functionality for modelling support to aid in establishing context for a given system. The tool helps users create an overview of the devices and services used, and can be edited easily and iteratively if changes occur in conjunction with the highly agile DevOps cycle.

4.3.2 Step 2 - Creating Data Flow Diagrams

Once we have established the context and created use case diagrams we go on to step two, creating data flow diagrams. In the late 1970, data flow diagrams were popularized by Constantine et al. [62] and Gane and Sarson [63]. There exists different types of notations for these diagrams, and we will be basing ourselves on the Yourdon and DeMarco notation [64]. DFDs are created by mapping flow of information for a process or system. It uses a defined set of four symbols to show data inputs, outputs, storage points, processes and the routes between each destination. The different diagram elements that are used are depicted in Figure 4.9. A typical model includes squares which are external entities and communicates with the system from the outside. Circles which represent processes that transforms inputs to outputs. Arrows which are data flows that show the transfer of information. Two parallel lines that represent warehouses/databases, which is used to store data for later use.

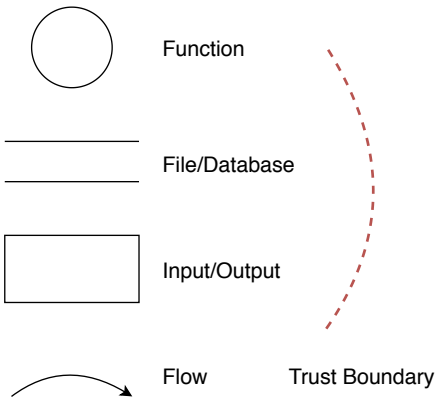


Figure 4.9: Data flow diagram notation

In these diagrams we will also include red dashed lines to indicate trust boundaries, points or areas where new actors are introduced, to further help distinguish the different parties involved. This helps identifying what information is available to the different parties, and helps identifying privacy and security risks in step 3.

For each of the use cases developed in step 1, we create a high level DFD model. Since the exact processes and information gathered from the devices and services may sometimes be unclear, our models may not reflect information flow as it truly is in real life, and is merely a simplified depiction. However, if we have more insight into the processes and information then we will be able to create more detailed DFD diagrams. For the example use case created in step 1 in Figure 4.7, we create a data flow diagram showing the flow of user information through the system, depicted in Figure 4.10.

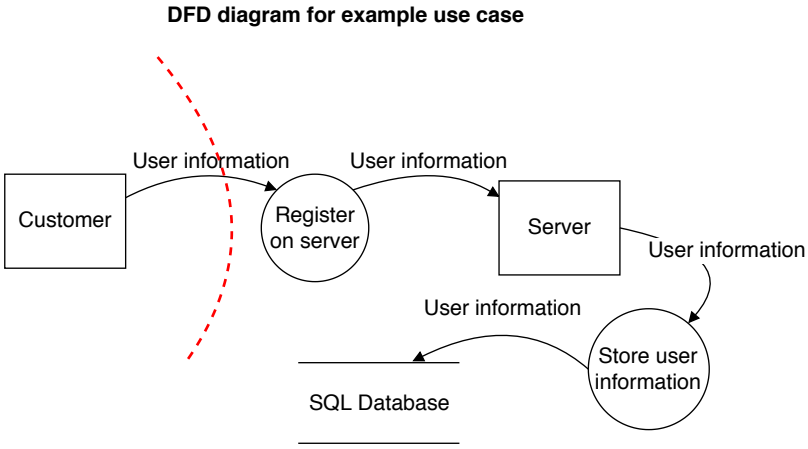


Figure 4.10: Data flow diagram for the example use case

In this DFD, the user information is the only information sent, and it is sent by the customer. The customers information crosses a trust boundary, as Big Company is a different actor from customer. The information then goes through the process "Register on Server" before being sent to the server. The server then receives the information, and sends it to the next process "Store user information". Finally the user information arrives at the SQL Database, and is stored.

With the knowledge of what information flows through the different actors, and the different parts device and services, we collect information regarding possible privacy and security risks in a template. Table 4.3 shows this template.

Device/ Service	**Source of Information**	**Information regarding Privacy**	**Information regarding Security**
Server	Big Company Official website and Big Company Privacy Policy.	"User information is collected for the sole purpose of registering users."	Susceptible to DDOS attacks
SQL database	Big Company Official website and Big Company Privacy Policy. News articles of recent SQL attacks	"User information is collected for the sole purpose of registering users."	SQL attacks are common attacks, especially when you don't use input validation.

Table 4.3: Security and privacy information gathered on the different devices, and their source.

To help with the next step, we use Microsoft's STRIDE method [65] alongside our DFD's. STRIDE is a acronym for a collection of threats, namely: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege. For each of the threats in STRIDE we consider how they could affect each part of the model. Each threat in STRIDE also have a set of countermeasures which may be applicable to these threats. The STRIDE method therefore helps us gather information regarding privacy and security risks, in the form of possible threats, what assets the attacker may want to harm, and how they may achieve their goal. This is used to assist risk modelling, and is the input for the next step.

4.3.3 Step 3 - Privacy and Security Risk Modelling

The next step then is privacy and security risk modelling. For this we use CORAS [60]. CORAS is an approach to risk analysis, supported by a graphical risk modelling language developed by Mass Soldal Lund et al. We use CORAS as it has been empirically shown to be intuitively simple for different stakeholders, and has been proven to be cognitively effective [66]. CORAS comes with a method which builds on ISO 31000 [30], and includes detailed guidelines for creating CORAS risk models.

CORAS risk models consist of unwanted incidents, threat scenarios, threats, vulnerabilities and assets. Each of these are depicted in Figure 4.11 and definitions are listed below.

- Threat: an action or event that is caused by a threat source and may lead to an incident.
- Unwanted incident: an event that harms or reduces the value of an asset
- Asset: anything of value to a party
- Vulnerability: a weakness, flaw, or deficiency that can be exploited by a threat to cause harm to an asset.

Figure 4.11: CORAS Diagram Elements

There is also a distinction between direct and indirect assets. Indirect assets are assets which are only harmed through harm of other assets.

There also exists three types of unbroken arrows in CORAS risk diagrams, and they are all relations. The first type is the *initiates* relation which goes from a threat to a threat scenario or an unwanted incident. The second type, *leads-to* relation which goes from a threat scenario or an unwanted incident to a threat scenario or an unwanted incident. Finally the *impacts* relation goes from an unwanted incident to an asset. The different relations are depicted in Figure 4.12.

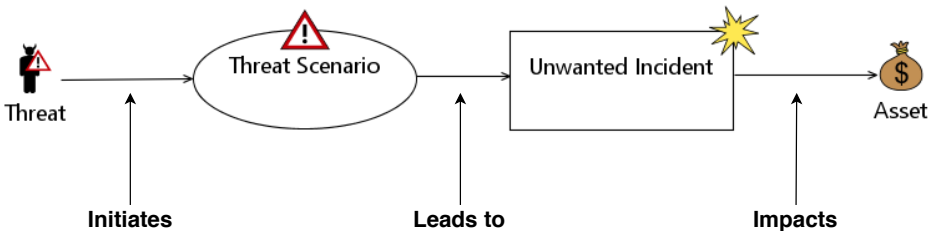


Figure 4.12: CORAS Unbroken arrows

A CORAS model takes input in the form of system diagrams, use case documentation, system manuals and similar. In our method we will be using a combination of use case diagrams, data flow diagrams, and information gathered in the previous steps. With this information we can then identify risks, threats and vulnerabilities.

There exists an extension of CORAS which also include the modelling of indicators [67]. Indicators can be attached to any relevant risk model element, and they help capture the dynamic behaviour of systems. Figure 4.13 shows the four different indicator types:

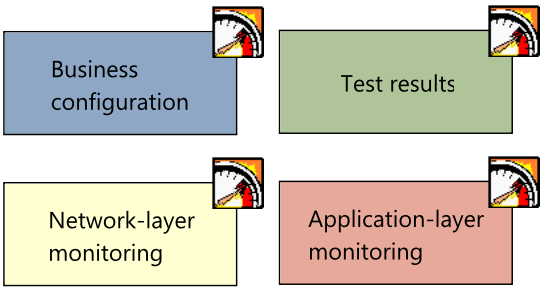


Figure 4.13: CORAS Indicators

In our method the blue business configuration indicators will be based on expert knowledge, and will be in the form of questions regarding either the system, privacy or security. The three other indicators, namely: test results, network-layer monitoring, and application-layer monitoring can receive input

automatically, but requires access to the source code. Because we do not have access to the source code throughout our thesis, we will only be using the blue business configuration indicators.

For each of the data flow diagrams created in the previous step, we create a corresponding CORAS risk model, each with one or more indicators. Figure 4.14 depicts a CORAS risk model, using CORAS notation. In this case it depicts a hacker injecting SQL, by exploiting the vulnerability "outdated software", which causes the unwanted incident "Private records of consumers are released" which harms the asset "privacy of costumers".

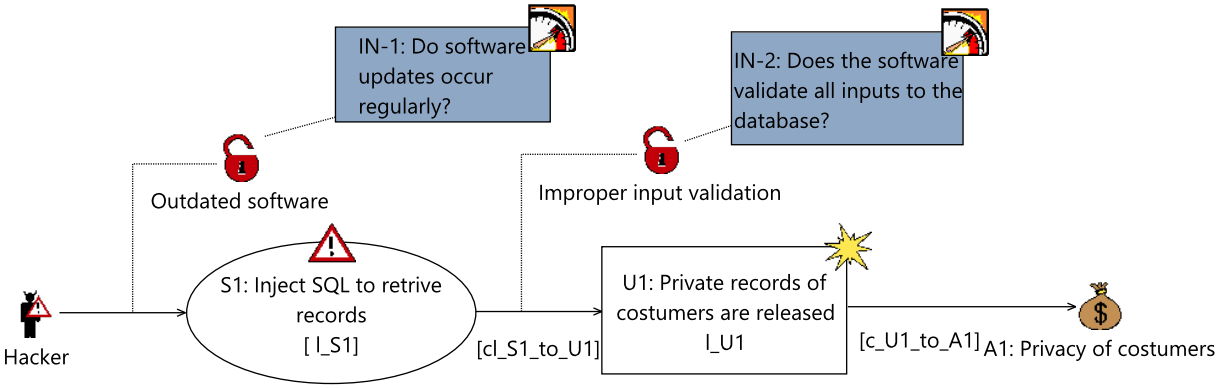


Figure 4.14: A CORAS diagram depicting a hacker injecting SQL to harm the privacy of costumers

The creation of CORAS diagram can be done using either the CORAS Wiser tool or the CORAS tool provided here:

http://coras.sourceforge.net/coras_tool.html

These CORAS models can then be exported as PNG images which serve as input to the next steps of the method.

4.3.4 Step 4 - Translating Risk Models to Executable Algorithms

Once we have our CORAS models, we use DEXi [61], a program for multi-attribute decision making, and a method for developing qualitative risk assessment algorithms to translate our risk models to executable algorithms. DEXi can create helpful decision models, as well as perform evaluation and analyses of these models to help with decision making.

DEXi defines "attributes", "scales", "tree of attributes" and "utility functions" to develop these models. Attributes are qualitative variables that represent decision subproblems. Scales are sets of symbolic values that can be assigned

to attributes. Tree of attributes is a hierarchical structure representing the decomposition of the decision problem. Utility functions are rules that define the aggregation of attributes from bottom to the top of the tree of attributes.

The use of qualitative attributes instead of quantitative ones is one of the benefits of DEXi, as our CORAS diagrams often times have indicators which are symbolic, and it is often difficult and sometimes impossible to obtain quantitative estimates [68]. The attributes are either basic attributes (terminal nodes of the tree, depicted as triangles), or aggregate attributes (internal nodes of the tree, depicted as squares).

Figure 4.15 depicts a small example of a DEXi model, where the root attribute is Car, the aggregate attributes are Car and Characteristics and the basic attributes are Price, Comfort and Safety.

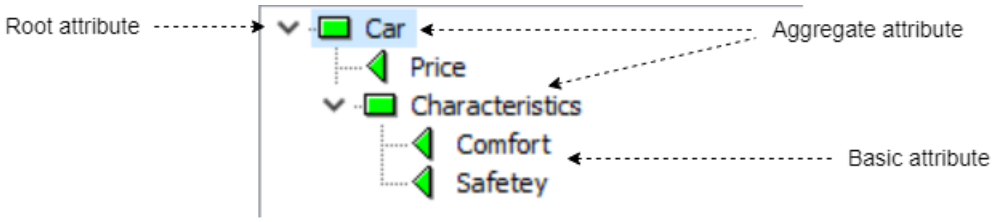


Figure 4.15: A DEXi model depicting a car

We utilize DEXi in our method, by using the method for developing qualitative risk assessment algorithms developed by Erdogan et al [68] on our CORAS risk models so we get executable risk assessment algorithms. This is done by mapping each of the different CORAS elements to either a basic attribute or an aggregate attribute, depending on what type of CORAS element it is, and if there are any related indicators attached to the element. As an example we will show how to build a risk assessment algorithm of model 4.14.

Step 1 - Risk: The risk is determined by the likelihood of the unwanted incident and its consequence for the asset in question. This corresponds to the unwanted incident U1 and the impacts relation to asset A1. The likelihood of U1 is denoted by l_U1, while the consequence of U1 for asset A1 is denoted by c_U1_A1. The DEXi representation is shown in Figure 4.16. Here risk is the top attribute, with two child attributes, one representing likelihood and one representing consequence.

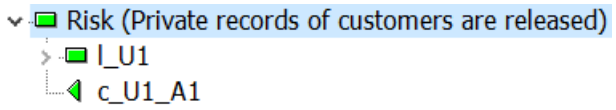


Figure 4.16: Screenshot of the DEXi tool, for step 1

Step 2 - Node with incoming leads-to relation: Figure 4.14 shows that the unwanted incident U1 has one incoming leads-to relations from S1. This means

that the likelihood of U1 depends on the likelihood contribution from S1. The DEXi representation is shown in Figure 4.17. The likelihood of a node with incoming leads-to relations is represented by an attribute with one child attribute for every incoming leads-to relation. In our example U1 has only one incoming leads-to relation, l_S1_to_U1 which therefore is represented as a child node.

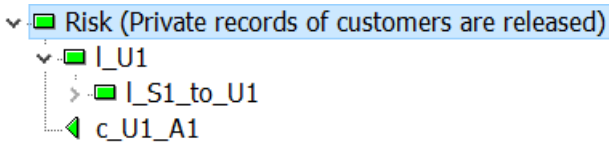


Figure 4.17: Screenshot of the DEXi tool, for step 2

Step 3 - Node with outgoing leads-to relation: The contribution from a leads-to relation to a target node depends on the likelihood of the source node and the conditional likelihood that an occurrence of the source node will lead to an occurrence of the target node. The latter is assigned to the leads-to relation. Figure 4.14 includes one leads-to relations from S1 to U1. The likelihood contribution from the relation from S1 depends on the likelihood of S1 and the conditional likelihood that S1 leads to U1. The DEXi representation is shown in Figure 4.18, where the likelihood of S1 is represented as l_S1 and the conditional likelihood of S1 leading to U1 is represented as cl_S1_to_U1.

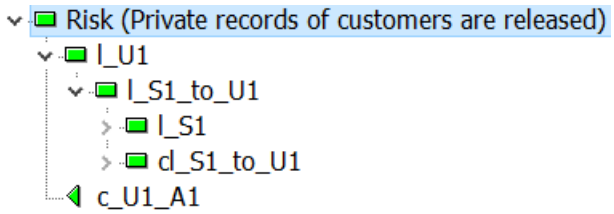


Figure 4.18: Screenshot of the DEXi tool, for step 3

Step 4: Indicators can be attached to a leads-to relation from one node to another to show that the indicators are used as input for assessing the conditional likelihood of an occurrence of the source node leading to the target node. This is typically done by attaching the indicators to a vulnerability on the relation, as such indicators normally say something about the presence or severity of the vulnerability. Figure 4.14 shows that indicator IN-1 is attached to vulnerability V1 and thus on the initiates relation going from Hacker to S1, while indicator IN-2 is attached to vulnerability S2, and thus on the leads-to relation going f rom S1 to U1. The DEXi representation is shown in Figure 4.19.

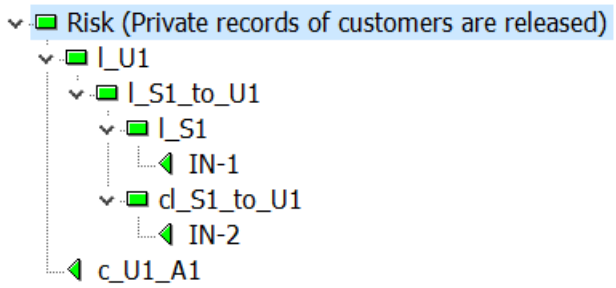
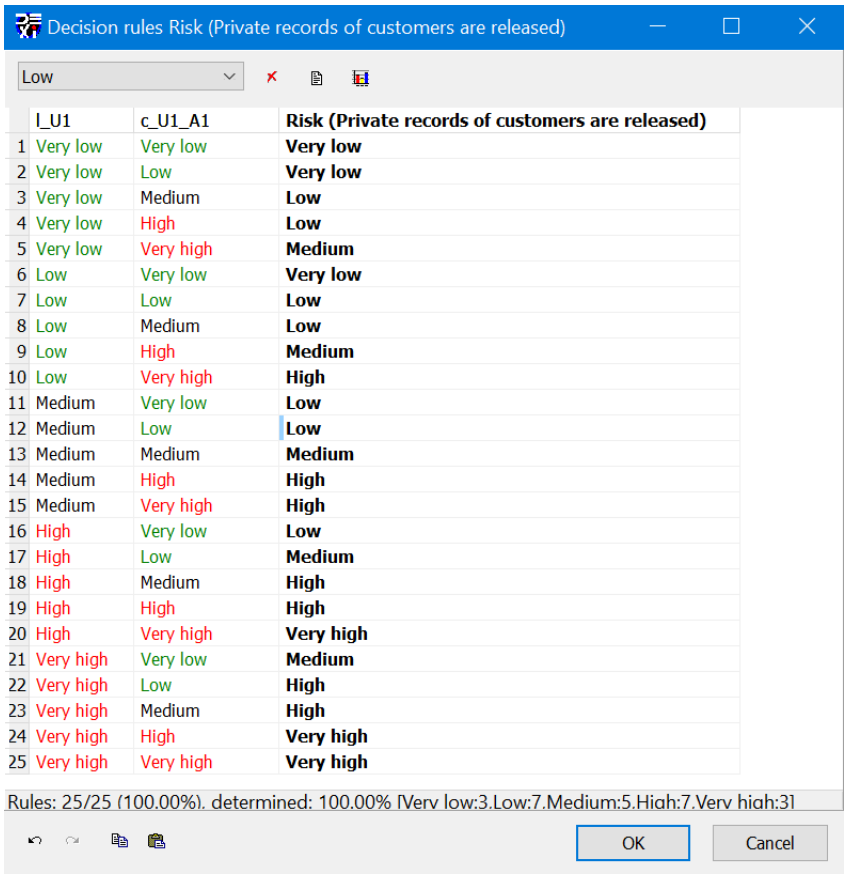


Figure 4.19: Screenshot of the DEXi tool, for step 4

Each of the DEXi models must have a defined scale and utility function for each attribute, and should be fully explicit to avoid undefined output values. In this example we use {Very low; Low; Medium; High; Very high} for all aggregate attributes. After defining scales, we define utility functions for each aggregate attribute. Figure 4.20 shows the utility function for the leads-to relation between S1 and U1 in Figure 4.14.



	l_U1	c_U1_A1	Risk (Private records of customers are released)
1	Very low	Very low	**Very low**
2	Very low	Low	**Very low**
3	Very low	Medium	**Low**
4	Very low	High	**Low**
5	Very low	Very high	**Medium**
6	Low	Very low	**Very low**
7	Low	Low	**Low**
8	Low	Medium	**Low**
9	Low	High	**Medium**
10	Low	Very high	**High**
11	Medium	Very low	**Low**
12	Medium	Low	**Low**
13	Medium	Medium	**Medium**
14	Medium	High	**High**
15	Medium	Very high	**High**
16	High	Very low	**Low**
17	High	Low	**Medium**
18	High	Medium	**High**
19	High	High	**High**
20	High	Very high	**Very high**
21	Very high	Very low	**Medium**
22	Very high	Low	**High**
23	Very high	Medium	**High**
24	Very high	High	**Very high**
25	Very high	Very high	**Very high**

Figure 4.20: Screenshot of the DEXi tool, for definitions of utility functions

The defined scale for the DEXi function equals that of the risk matrix depicted in Figure 4.21. The number of values for likelihood and consequence are typically within the range of three to five as a higher number of values often unnecessarily increases the complexity. If however you want a higher granularity, you may increase the number of values.

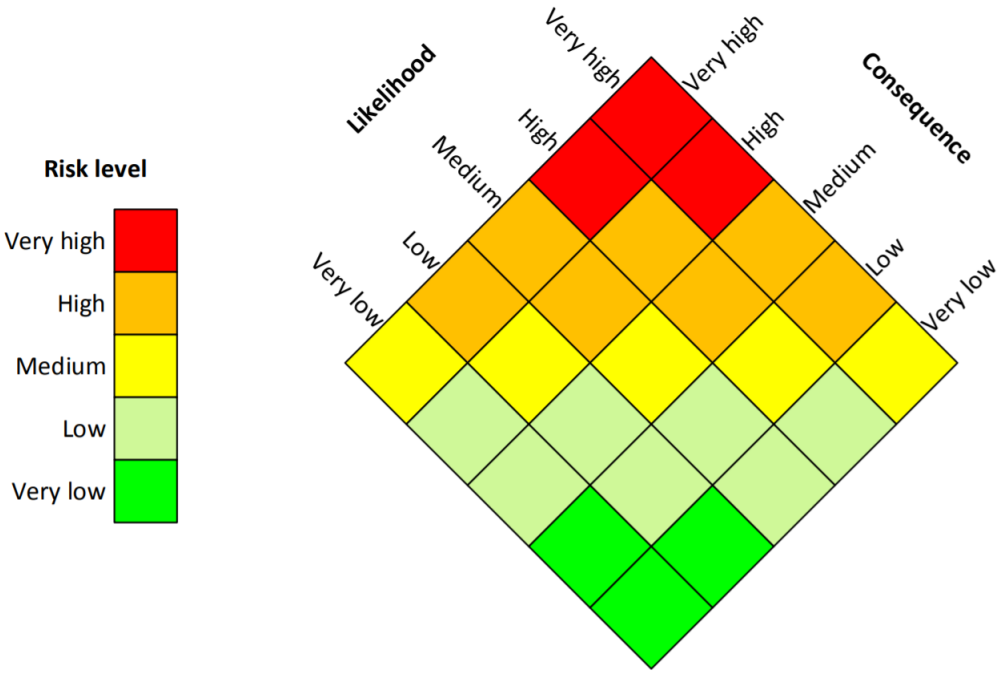


Figure 4.21: Risk Matrix

The reason for defining the scales for likelihood and consequence with five values is simply because it is standard[60].

4.3.5 Step 5 - Executing Risk Assessment Algorithms Using Our Tool

When all the previous steps are completed, we go on to the final step. Here we upload the CORAS diagram in combination with its respective DEXi model. The tool accepts CORAS diagrams as .PNG files and DEXi models as .DXI files. Once uploaded the tool provides the user with the depiction of the CORAS diagram, as created earlier. Alongside this it depicts the DEXi attributes in the lower left corner, which require input for calculating the overall risk. Once the user assigns each attribute with an input, the user may click calculate, and the tool depicts the overall risk depending on what inputs were given.

We will now present the tool usage for the execution of algorithms step by step. Figure 4.22 depicts the tool when the relation model is complete, and no other files have been uploaded.

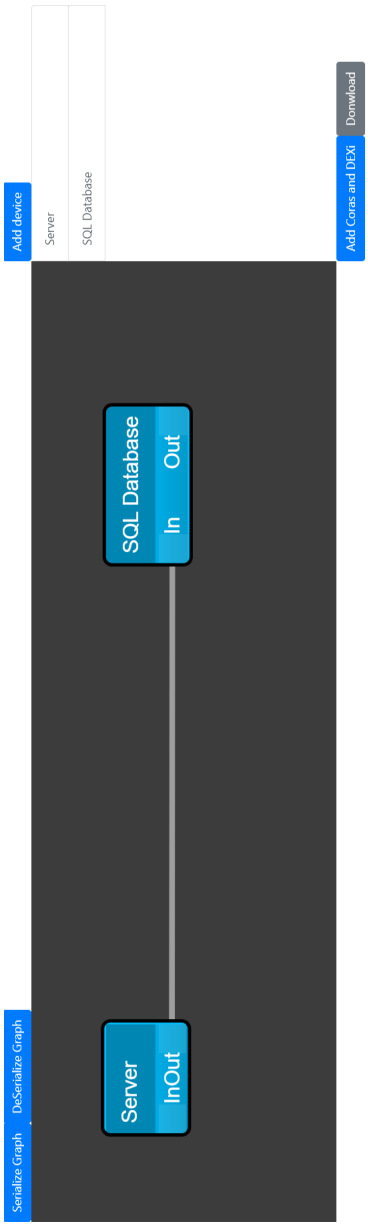


Figure 4.22: Overview of the entire tool

Figure 4.23 shows a zoomed in view of the list of devices and services, followed by the upload button for the CORAS models and DEXi algorithms.

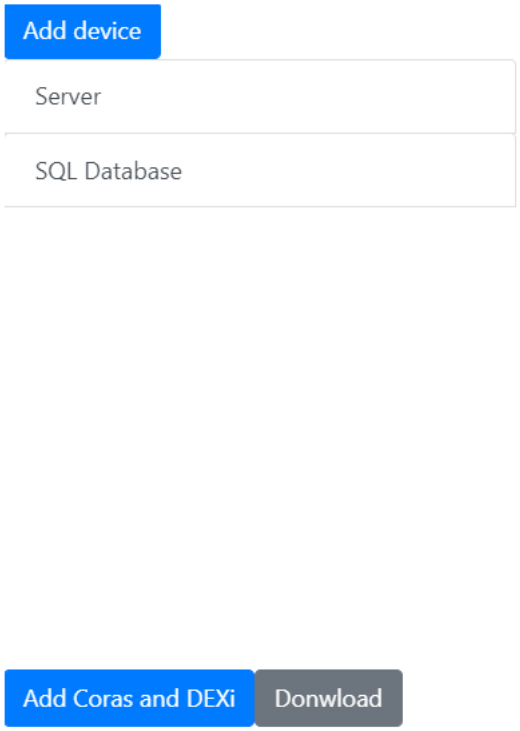


Figure 4.23: A zoomed in view of the device/service list and the input for CORAS and DEXi files

Figure 4.24 shows the inputs for the DEXi .dxi file, and the CORAS diagram, as a .png file. Once the user merges and uploads the files, a combination is created and added to the list of CORAS-DEXi combinations. This list is shown n Figure 4.25. Each CORAS-DEXi combination may also have a short description.

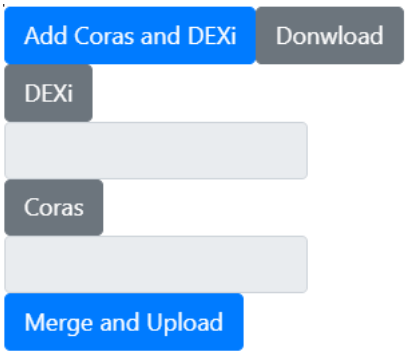


Figure 4.24: A zoomed in view of the input for CORAS and DEXi files

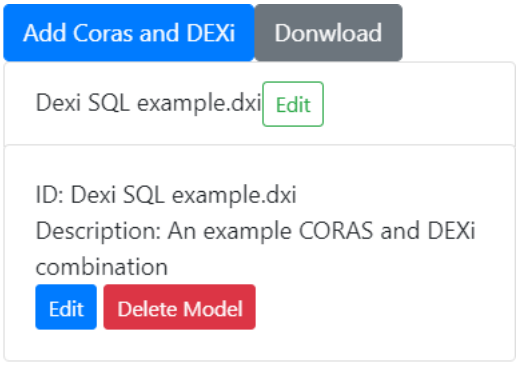


Figure 4.25: A zoomed in view of the list of CORAS-DEXi combinations, once the files have been merged and uploaded.

Once a combination is selected from the list, the tool shows the CORAS model, as well as inputs for every indicator and attribute requiring an input in the CORAS model. The inputs are then given either manually by the user, or given automatically from the API and can be used to execute the DEXi algorithms. Figures 4.26-4.28 show different views of the tool.

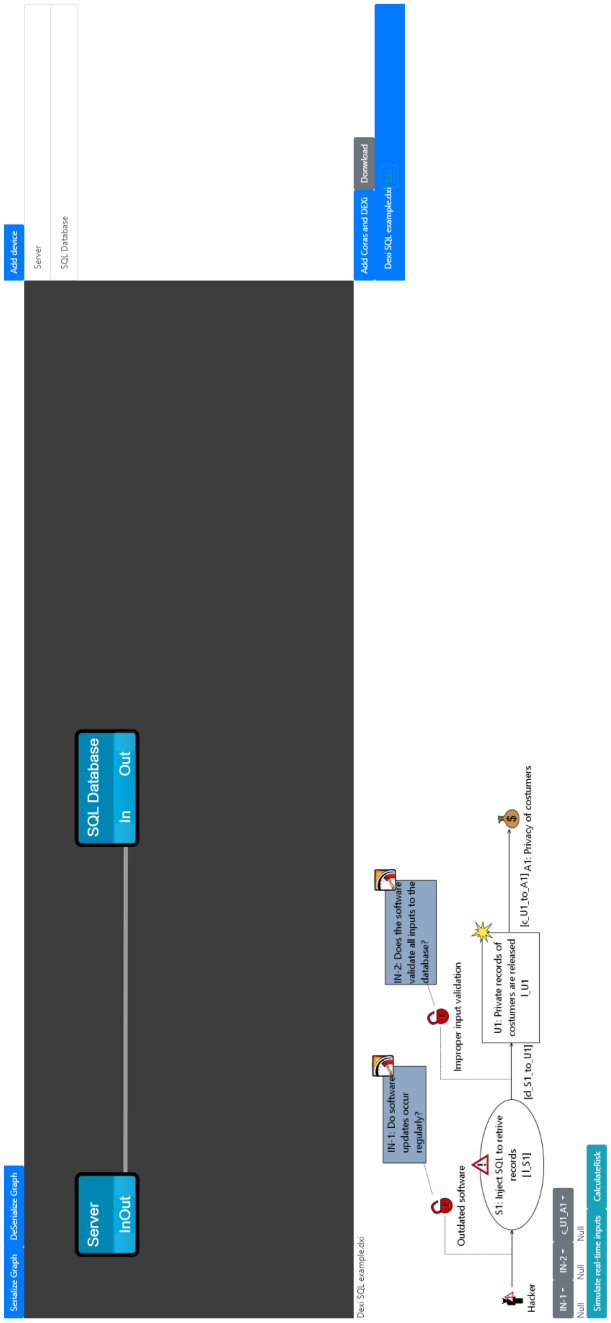


Figure 4.26: An overall view of the tool with both the relation model and the CORAS DEXi algorithms uploaded

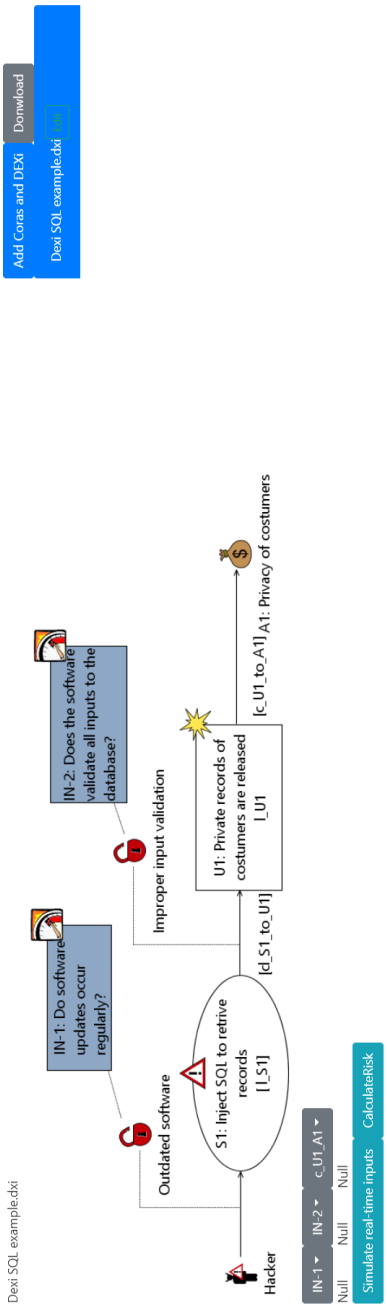


Figure 4.27: A zoomed in view of the CORAS-DEXi module, with the uploaded files

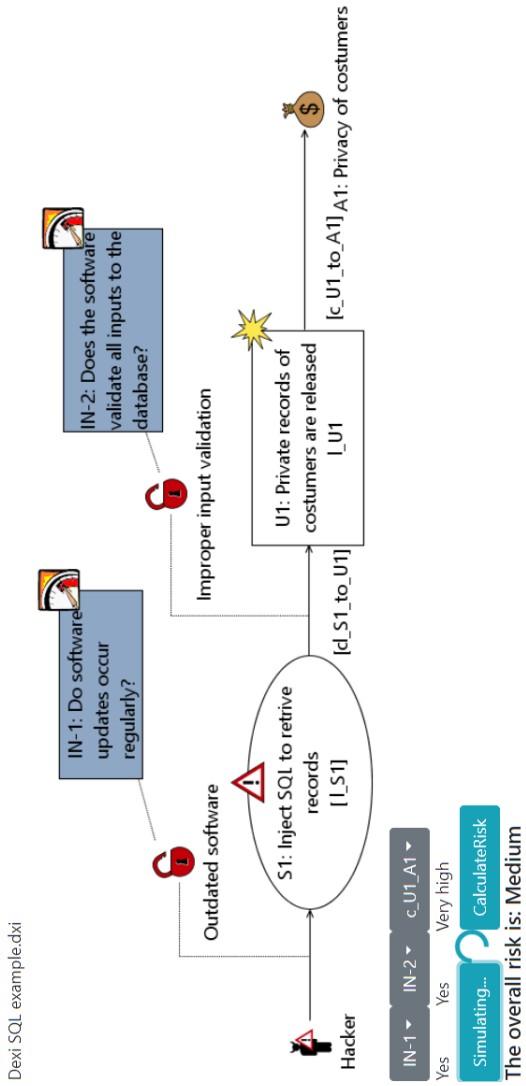


Figure 4.28: The CORAS DEXi module, with the uploaded files, simulating monitor API calls for the calculation of risk

Figure 4.28 shows the tool simulating API calls with different inputs, as to simulate a real-time monitor. In this case the inputs for both indicators were "Yes" and the consequence of unwanted incident U1 on asset A1 was very high. This input results in the DEXi algorithm outputting the risk with the value of "Medium".

Once risk values are presented by the tool, our method is complete, and it is up to the developer to use this information to further aid in the DevOps cycle.

The risk values indicate areas where there needs to be a form of mitigation. Mitigation is often done either by communicating with management to secure more resources, or further developing the part of the system which is susceptible to risk.

Later in Chapter 5, we present a case to give an example of how to apply our method, and how our tool fits in throughout the method.

Chapter 5

Applying Our Tool Supported Method In A Smart Home Case

In Chapter 4 we went through the process of developing our artefact, which is a method to be used in the planning phase of DevOps to help developers in identifying security and privacy risks, and executing risk assessment algorithms, and a tool to support this method. In this chapter we apply our tool supported method on a real-world smart home case.

5.1 Preface

With the increasing availability and number of smart devices in the market, smart homes have risen in popularity. The devices offer convenience and automation of simple daily tasks and help individuals with management of their home. Users can adopt these technologies and services to help monitor their power consumption, saving money on power cost, or they can use smart alarm systems to protect their home, with devices such as cameras, sensors and smart locks. The smart home also offers devices that help tedious daily tasks, such as automated robot vacuums and smart washing machines. Practical solutions are not the only use for the smart home, as it also offers entertainment with smart TV's and Smart audio systems.

Eureka! [69] A research company conducted a survey of nearly 1000 household owners in the U.K. to better understand consumer attitude towards smart tech. While they mention several key findings the ones most interesting were:

- The most popular product types for homeowners are related to security, lighting and heating.
- When it comes to installation of the smart technology, 59% prefer having

a smart tech expert install it, 14% prefer an electrician, and 13% would do it themselves.

These two findings bring out key elements: Firstly, the most popular devices have a direct physical dimension, which if control is in the wrong hands, could have lethal consequences. And secondly, most homeowners trust and rely on professionals for installation of the devices.

While smart homes are becoming popular, security and privacy is often an afterthought [70, 71]. There have been several successful attempt at hacking smart devices. Kafle et al. [72] found SSL-vulnerabilities in popular smart home devices Nest and Philips Hue. Roy Solberg, a norwegian system developer, found that communication in the Mill smart heaters was unencrypted, which meant that anyone could control any internet connected heater [73]. And a Tesla car, although arguably not a smart home device, was in 2016 proven to be remotely hijack-able [74]. The Tesla attack was luckily discovered by the security team at Tencent, and disclosed directly to Tesla, so that they could patch it quickly.

These attacks and vulnerabilities however open up discussion on the security of smart home devices.

The Open Web Application Security Project, OWASP for short, released in December 2018 an updated list of top ten things to avoid when building, deploying, or managing IoT systems. This list is shown in Figure 5.1, and is helpful to keep in mind.

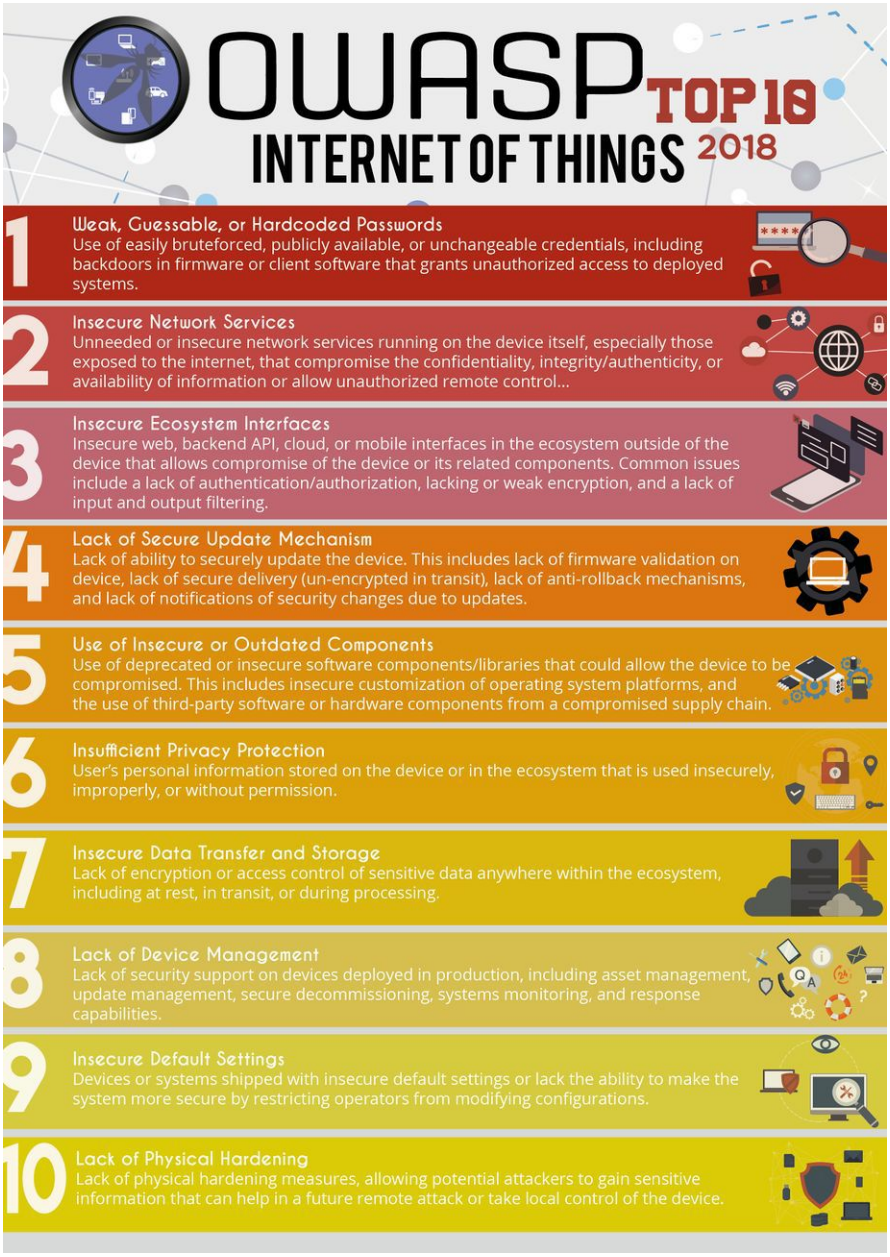


Figure 5.1: OWASP IoT Project

5.2 Context Establishment

We now present a smart home case to give an example of a possible use of the tool supported method we have developed. We start with the first step of our method, which is context establishment. Our smart home case starts with a family of three living in the outskirts of a large city. The family consist of a mother, a father and an 8 year old daughter. The father has recently picked up an interest for smart homes, and wants a large collection of smart devices, to help automate the family's daily lives, help save money on power, and keep their home and family members safe.

To help in designing, deploying and managing their smart home, they consult a professional IoT team from a service provider similar to what is currently on the market, such as Vivint [75] or Smartn [76].

The newly hired team begin the process of planning the smart IoT system. The team follow the DevOps agile approach, and will be using our method to carry out the planning of this system. The first step is therefore the context establishment. The team have to identify sources of risk, what the family value as assets, and possible unwanted incidents. The team conduct informal interviews with the parents to gather information regarding their preferences and requirements related to security, privacy and functionality, and consult their panel of experts for an expert opinion.

The family then presents the requirements for the smart home, in regards to functionality, security and privacy. The mother is skeptical of smart home technology and wants to keep the privacy high for herself and their daughter. The father is more concerned about their valuables, and requires a way to watch their home when not at home. Privacy is not a concern for the father. Both the mother and father own an android smart phone, which they wish to use together with services to control their devices, and to view their child's location and phone usage if there should be a need for it. For the family to achieve their goals they require a selection of devices, which come with several services, and as none of the family members are very tech savvy, they will require help from the professional IoT service provider.

The family gives the team a list of the devices they wish to use, and what purpose the devices serve. Table 5.1 shows the list of these devices and gives a short description of each of them.

Device	Description
Nest Hello Doorbell	Smart home doorbell which includes a camera, a continuous video log, and the possibility of alerting when people arrive at the door.
Nest Protect 2. Generation	Smart home smoke detector with alerts of which room there is smoke, and the possibility of sending warnings to your phone.
Nest Cam IQ Outdoor	High quality outdoor camera that alerts your phone when spotting a person.
Nest Cam IQ Indoor	High quality indoor camera with speakers.
Tail it kids	A smart watch for kids with functionality of tracking, phoning and an emergency SOS button.
Mill AV1200WIFI	Smart home heater which can be controlled from your phone.
Philips Hue Bridge	A smart home bridge which allows for scheduling of timers and control of your lights when away or from your phone.
Philips Hue E26 Bulbs	A smart light with the possibility of changing colours.
Philips Hue Lightstrip	A smart lightstrip with the possibility of changing colours.
Philips Hue Motion Sensor	A motion sensor which triggers lights when motion is detected.
Amazon Echo	A smart home device which plays music, makes calls, sets alarms, answers questions, controls other smart devices and can buy things online for you.

Table 5.1: The devices to be used and short descriptions of each

From the system description given by the family, the team create asset diagrams to identify what assets the family members have. Figures 5.2- 5.4 depict the asset diagrams created for the families assets.

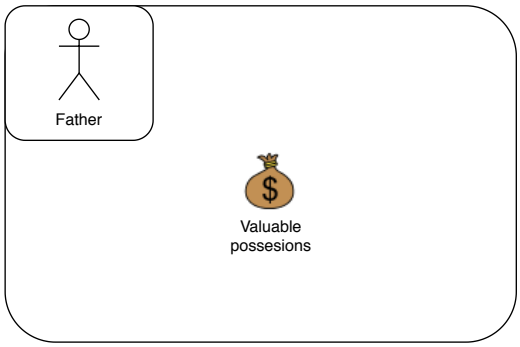


Figure 5.2: Asset diagram for the Fathers assets

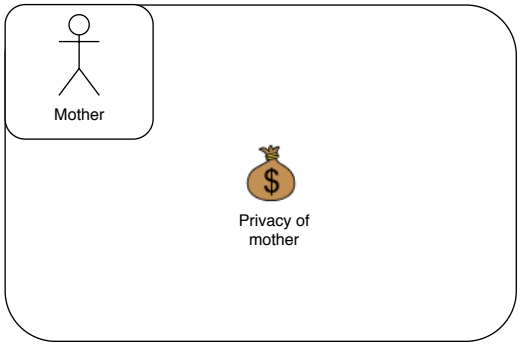


Figure 5.3: Asset diagram for the Fathers assets

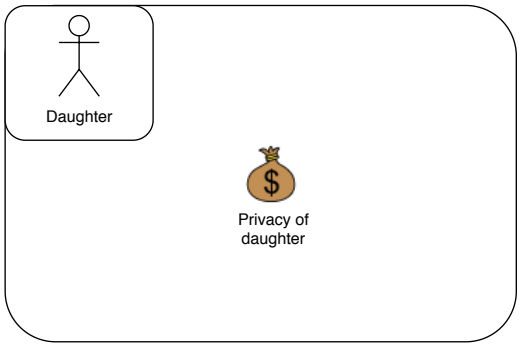


Figure 5.4: Asset diagram for the Fathers assets

After defining assets, the team can create a consequence scale and a likelihood scale. As the fathers valuables have a monetary value, a consequence scale is created to reflect this. However, as the privacy of the mother and daughter is

difficult to quantify in terms of money, a seperate consequence scale is made for their assets. Table 5.2 show the consequence scale for the fathers valuables, and Table 5.3 show the consequence table for the mothers and daughters privacy.

Consequence	**Description**
Very High	A monetary loss of more than 2500$
High	A monetary loss of more than 1000$
Medium	A monetary loss of more than 500$
Low	A monetary loss of more than 100$
Very low	A monetary loss of less than 100$

Table 5.2: The consequence scale defined for the fathers valuables

Consequence	**Description**
Very High	Sensitive information regarding health and/or beliefs is leaked or used inappropriately
High	Sensitive location data, or data gathered from devices in an automatic fashion is leaked or used inappropriately
Medium	Information that can be used to identify a specific person is leaked and/or is shared with a third party
Low	Information that can be used to identify a specific person is used for advertisement
Very low	Information that can be used to identify a specific person is collected by a company

Table 5.3: The consequence scale defined for the privacy of the mother and daughter

Likelihood	**Description**
Very High	Happens more than twice times a year
High	Happens between once a year to twice a year
Medium	Happens once every 2 years
Low	Happens once every 5 years
Very low	Happens once every 10 years

Table 5.4: The likelihood scale

Once the devices and assets have been identified, and scales for consequence and likelihood are defined, the team model some use cases to help expand upon the context. These use cases are listed below, and also illustrated in Figures 5.5-5.8:

1. The dad uses his phone to check on his daughter's location with the use of the Tail It Kids smart watch and the Tail It app.

2. The mother uses the Amazon Echo to buy a new hair product online.

3. The family drives to cabin and are expecting a package delivery while away. The Nest automated doorbell detects a person at the door, and alerts the father.

4. While on the way home from the cabin trip, the father wants to come home to a heated house. Using the Millheat app, he turns the heater to 22 degrees Celsius.

5. When they arrive home, the Philips Hue motion sensor senses motion and turns on their Philips Hue lighting.

Figure 5.5 depicts the first use case "Check on daughter", where the father, Tail It Kids, and the daughter are actors. The father begins by opening his phone which includes the use case of opening the Tail It App which again includes the final step of checking the daughters location. For the final step Tail It Kids and the daughter take part as the location of the daughter is collected in this step, and Tail It Kids utilizes this information.

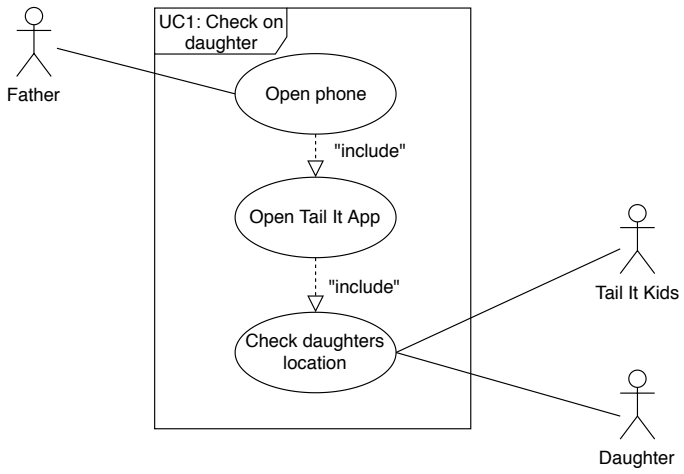


Figure 5.5: Use case diagram of the first smart home case

Figure 5.6 depicts the second use case "Buy product from Amazon", where the mother, Echo and Amazon Services are actors. The mother starts off by asking her Echo to purchase a specific item, which includes the use case of Echo processing this command, which again includes Amazon Services completing the purchase.

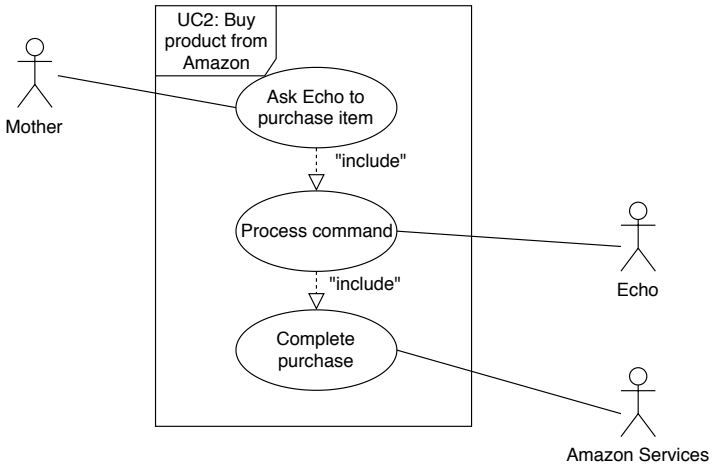


Figure 5.6: Use case diagram of the second smart home case

Figure 5.7 depicts the third use case where a Delivery Guy, and Nest Hello Doorbell are actors. The Delivery guy begins by delivering a package, which includes the Nest Hello Doorbell sensing motion, as well as alerting the homeowners phone.

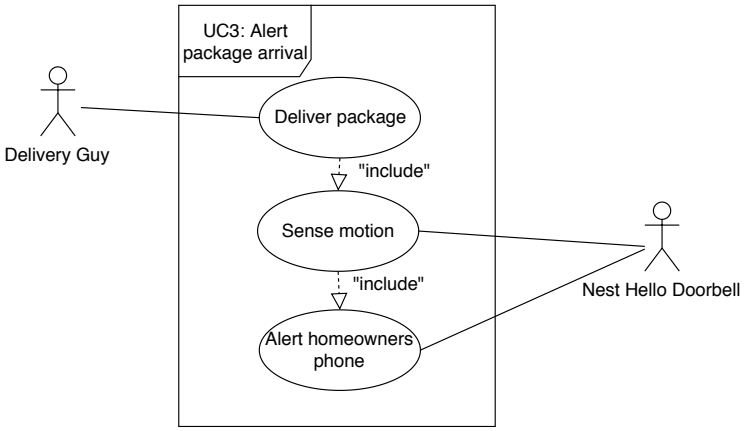


Figure 5.7: Use case diagram of the third smart home case

Figure 5.8 depicts the fourth use case, where the Father and Millheat are actors. The Father starts off by using the Millheat app on his phone to turn on the heater, which includes Millheat turning the heater on.

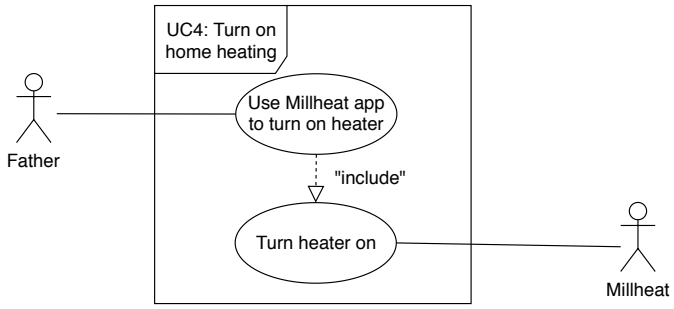


Figure 5.8: Use case diagram of the fourth smart home case

Figure 5.9 depicts the fifth use case where the Family and the Philips Hue Sensor, Bridge and Lights are actors. It starts off by having a family member arriving home, which includes the Philips Hue sensor detecting motion, which again includes the Philips Hue Bridge and lights turning on the lights.

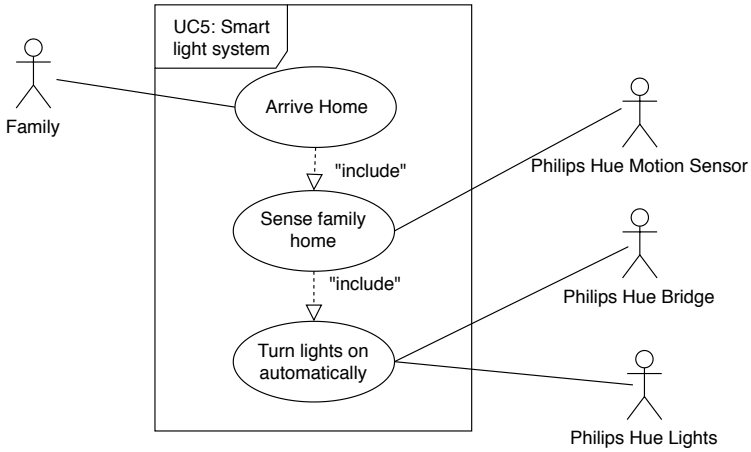


Figure 5.9: Use case diagram of the fifth smart home case

The newly hired IoT team then begin planning the smart home. They begin working, and start using the modelling part of our tool to help gain an overall picture of the finished system, as well as gather input to the further steps of our method. The finished model is depicted in Figure 5.10. The figure shows each of the different devices and to which device they communicate with. On the left side of the diagram the Nest devices are grouped together as many of those devices communicate which each other, and similarly on the right side we have Philips Hue devices grouped together. In the middle of the diagram we have the Amazon Echo and the Smartphone. These two devices can communicate with most of the other main components and act as a "command center" for control of all of the devices.

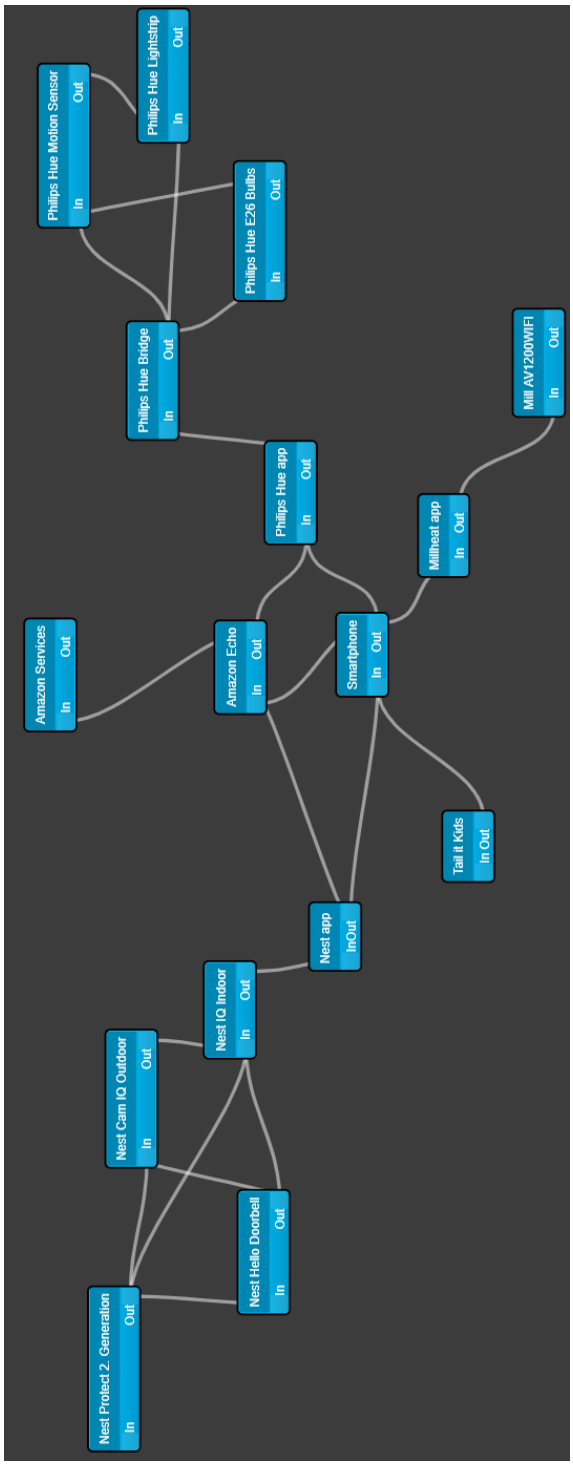


Figure 5.10: A context diagram including the devices and services to be used by the family

5.3 Data Flow Analysis

After performing the context establishment, the team continues with a data flow analysis to gather information related to security and privacy in the devices and services.

For each of the use cases presented in the previous section, the team create a corresponding DFD model. The diagrams modelled here are based on the different companies privacy policies in mind, as we do not have access to proper data flow within their systems. Some of these policies are vague and group different types of information into a general term "your data" or just "data" which is why the models are simplified and may seem very broad.

Combining the creation of DFDs, Microsoft's STRIDE method, and gathering information regarding privacy and security from privacy policies, official websites and online resources, the team also create tables for each of the DFDs which describes the devices and their related services. Column one of these tables contains the name of the device and/or service which they describe. Column two shows the source of the information gathered for the remaining columns. Column three and four contains information regarding privacy and security respectively.

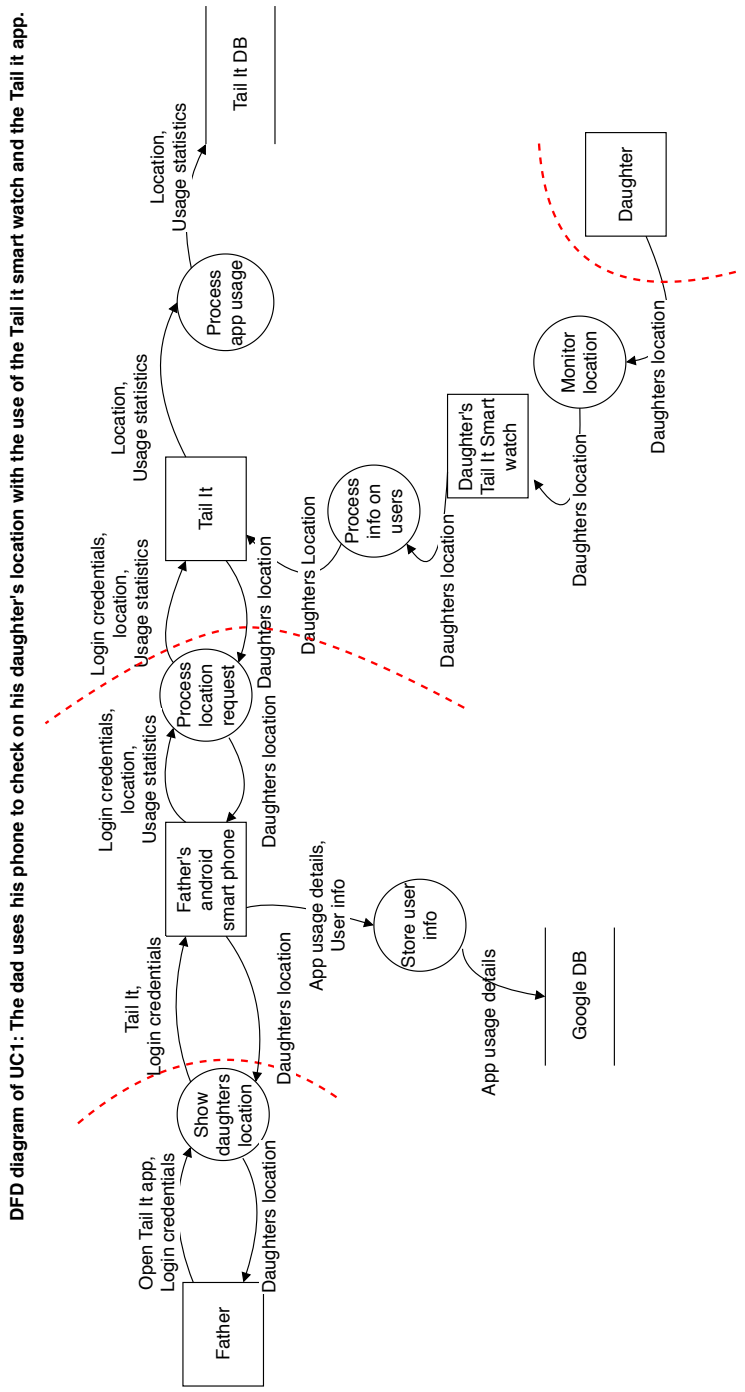


Figure 5.11: DFD for Use Case 1: *The dad uses his phone to check on his daughter's location with the use of the Tail it smart watch and the Tail it app.*

In Figure 5.11 the team has modelled a DFD diagram with trust boundaries for the use case considering when the father uses the Tail It app to check on his daughters location (UC1). The main external entities in this DFD are the father and his smart phone, the daughter and her smart watch, Tail It, and third parties, each of which are represented by a square. The father begins by opening the Tail It app on his smart phone and entering login credentials. Just by opening the app the Android smart phone sends app usage details to Google's database, this data therefore travels through a trust boundary between the father and the google android smart phone. The Tail It app also collects information from the fathers smart phone such as login credentials, location and usage, which traverse over yet another trust boundary between the android phone and Tail It. Tail It also collects location information from the daughters smart watch, which similarly to information from the fathers device, travels through a trust boundary between Tail It and the daughter. In Table 5.5 we describe the devices which are in use in this DFD, as well as information regarding privacy and security related to each device, and where this information has been gathered.

Device/ Service	Source of Information	Information regarding Privacy	Information regarding Security
Tail It	Official website [77], Privacy Policy [78].	Information about the person using their products is being collected as well as information about who is responsible for the use of the product, often parents of the user. One can choose to consent to the registration of the following information about them through their services: What services the person using the product have used. How a person have used the services. Location data about a person and their product in Tail It's systems. Photos, chats, friends, contacts and places you have added and visited.	The Norwegian "Forbrukerrådet" is a consumer protection organization which back in 2017 teamed up with the security firm Mnemonic [79] to investigate smartwatches for kids. They found significant security flaws at the time, which have now been patched. [80]

Continued on next page

Device/ Service	Source of Information	Information regarding Privacy	Information regarding Security
Android smart phone	Privacy Policy. [81] Official website. [82]	Google collects information a person provides directly to them; Information they collect about a persons use of their Services; and Information they obtain from third-party sources.	Built-in malware defense. Application sandboxing isolates and guards every Android app, stopping other apps from accessing your private information. Full on-device encryption.

Table 5.5: Security and privacy information gathered on the different devices, and their source for use case 1.

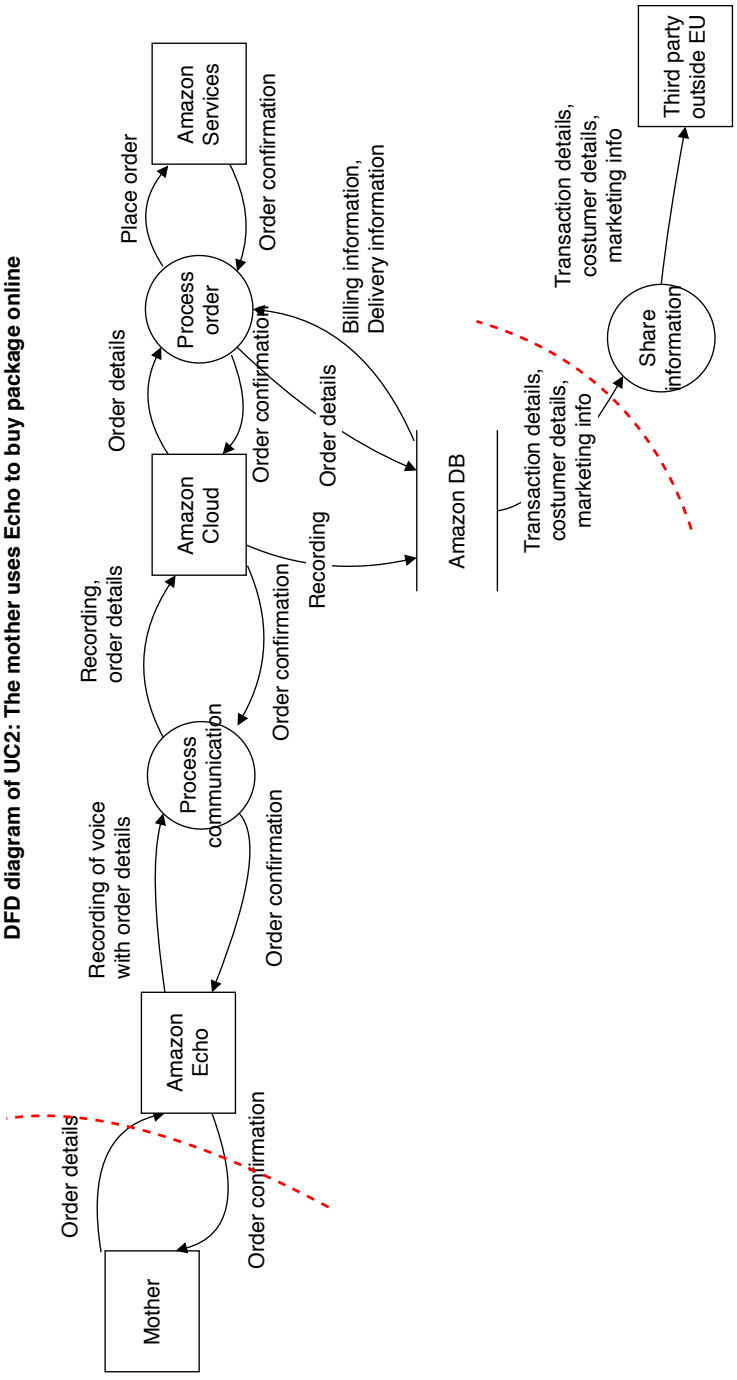


Figure 5.12: DFD for Use Case 2: *The mother uses Echo to buy package online.*

In Figure 5.12 the team present a DFD for the use case when the mother uses the Amazon Echo to place an order for online shopping (UC2). The mother speaks to the Echo, and a recording of the voice is sent to the Amazon Cloud to be processed. Amazon then stores the recording, transaction details and billing information which have been given by the mother earlier. Transaction details, costumer details and marketing info may then be shared with third parties.

Device/ Service	**Source of Information**	**Information regarding Privacy**	**Information regarding Security**
Amazon Echo	Privacy Policy. [83] Official website. [84] Report by Wired [85]. Paper by Kumar et al.[86].	Amazon receive and store certain types of information whenever a person interacts with them. Amazon shares user data with their parent corporation, Amazon.com, Inc., and the subsidiaries it controls, and they may share personally identifiable information with third parties. Alexa, a part of the Amazon Echo, also has third party “skills”, each with their own privacy policy.	Alexa has so called skills, which can be used to perform an attack called "Skill squatting". Another attack was disclosed by Chinese hackers at DefCon in 2018, where they could remotely control an Alexa. This attack has been patched.

Table 5.6: Security and privacy information gathered on the different devices, and their source.

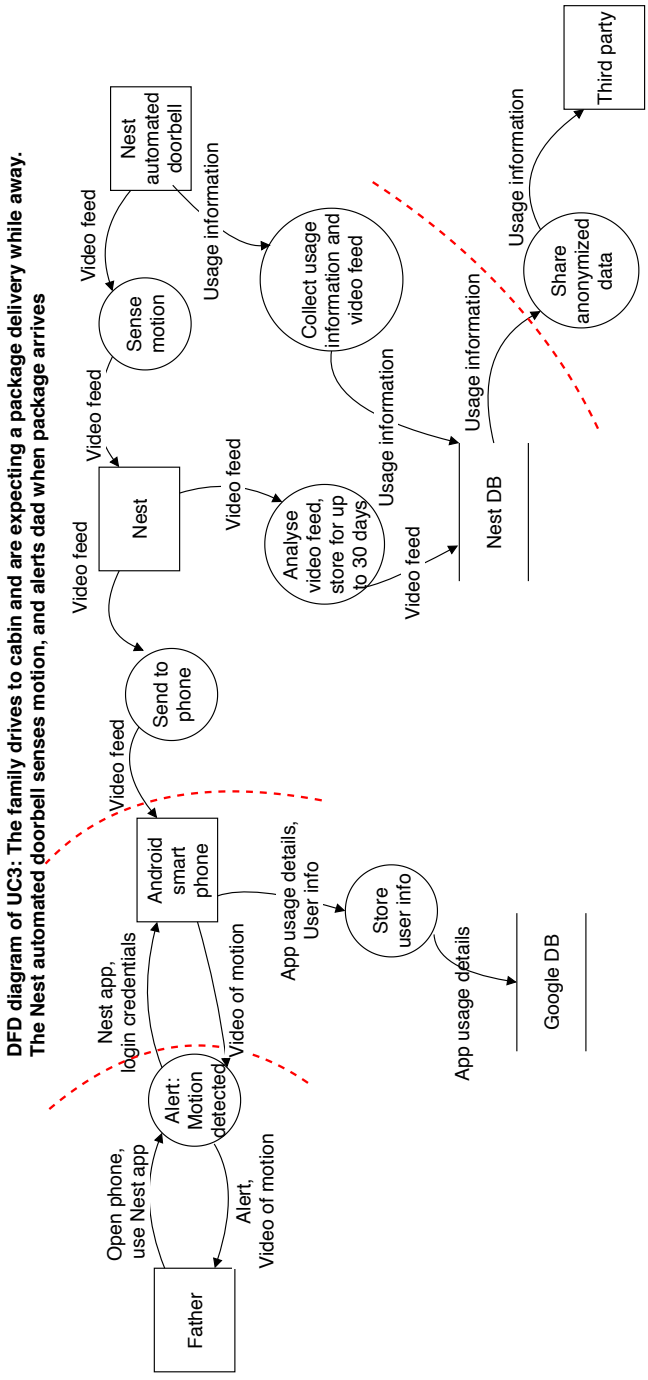


Figure 5.13: DFD for Use Case 3: *The family drives to cabin and are expecting a package delivery while away. The Nest automated doorbell senses motion, and alerts dad when package arrives.*

In Figure 5.13 we again have the father using a smart phone app, this time to access Nest. App usage details are again stored with Google. This time however the father receives information from his Nest security system, in the form of a video capture triggered by motion. Nest also receives a copy of the video feed which they may use for analysis, and may store for up to 30 days. Alongside this they collect usage information from the Nest automated doorbell. This usage information is then anonymised and shared with third parties.

Device/ Service	**Source of Information**	**Information regarding Privacy**	**Information regarding Security**
Android smart phone	Privacy Policy. [81] Official website. [82]	Google collects information a person provides directly to them; Information they collect about a persons use of their Services; and Information they obtain from third-party sources.	Built-in malware defense. Application sandboxing isolates and guards every Android app, stopping other apps from accessing your private information. Full on-device encryption.

Continued on next page

Device/ Service	Source of Information	Information regarding Privacy	Information regarding Security
Nest	Privacy Policy. [87] Official website. [88]	Nest collects: Setup information a person provides. Environmental data from the Nest Learning Thermostat's sensors. Direct adjustments to the device, including temperature or settings. Heating and cooling usage information. Technical information from the device.	Nest takes security seriously and cares about the integrity of your personal information. Nest uses commercially reasonable physical, administrative, and technological methods to transmit and store your data securely. However, Nest cannot guarantee that unauthorized third parties will never be able to defeat our security measures or use your personal information for improper purposes.

Table 5.7: Security and privacy information gathered on the different devices, and their source.

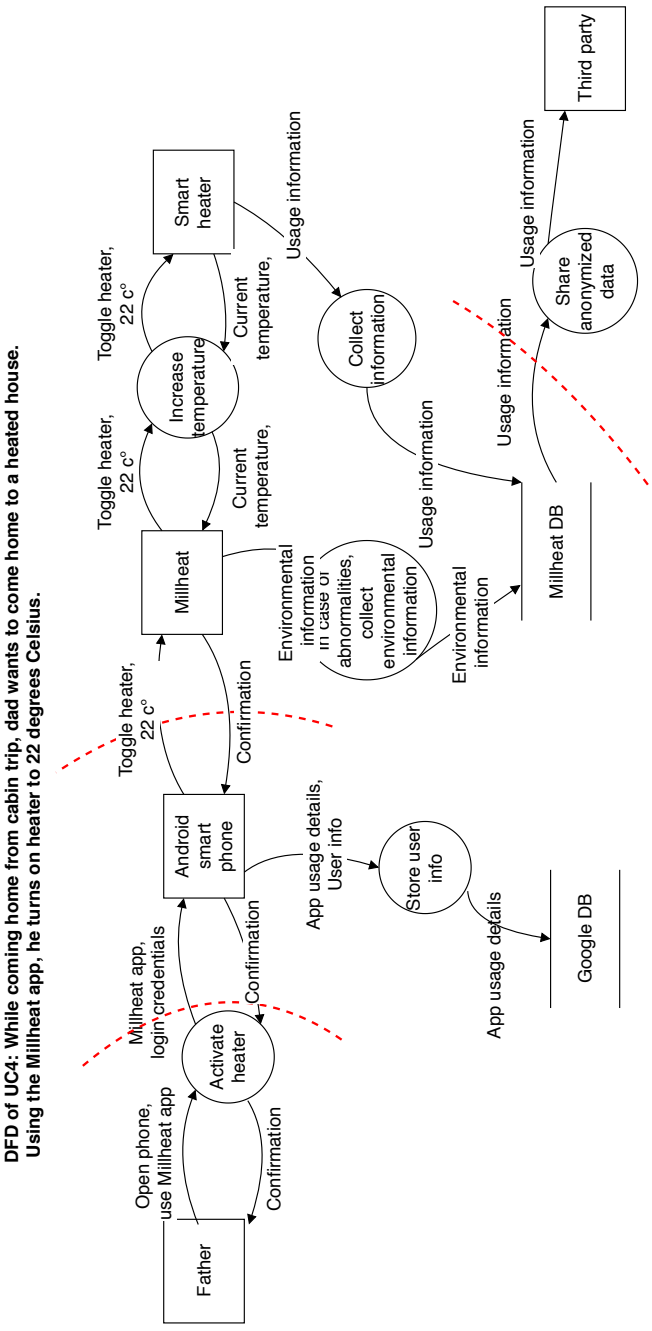


Figure 5.14: DFD for Use Case 4: *While coming home from cabin trip, dad wants to come home to a heated house. Using the Millheat app, he turns on heater to 22 degrees Celsius.*

In Figure 5.14 the family is on their way home, and the father uses the millheat app to turn on his smart heater. App usage data and user info is again stored with Google. Millheat gathers usage information on the Millheat devices, and if there are any abnormalities, they will also collect environmental information such as equipment ID, internet protocol address, routing data packets. They also share anonymous information with the public and their partners.

Device/ Service	**Source of Information**	**Information regarding Privacy**	**Information regarding Security**
Android smart phone	Privacy Policy. [81] Official website. [82]	Google collects information a person provides directly to them; Information they collect about a persons use of their Services; and Information they obtain from third-party sources.	Built-in malware defense. Application sandboxing isolates and guards every Android app, stopping other apps from accessing your private information. Full on-device encryption.
Millheat	Privacy Policy [89], Blogpost by Solberg [73].	In order to show the overall use of our products or services trends, Millheat may share anonymous information with the public and our partners. Millheat collects information about your name, telephone number and/or e-mail. They also collect some environmental information for diagnostic problems such as equipment ID, internet protocol address, routing data packets, etc.	Prior to october 2018, millheat only used https for authentication, all other communication was sent through http. This made it possible to control mill heaters worldwide. Millheat were quick to respond however and patched it quickly after discovery.

Table 5.8: Security and privacy information gathered on the different devices, and their source.

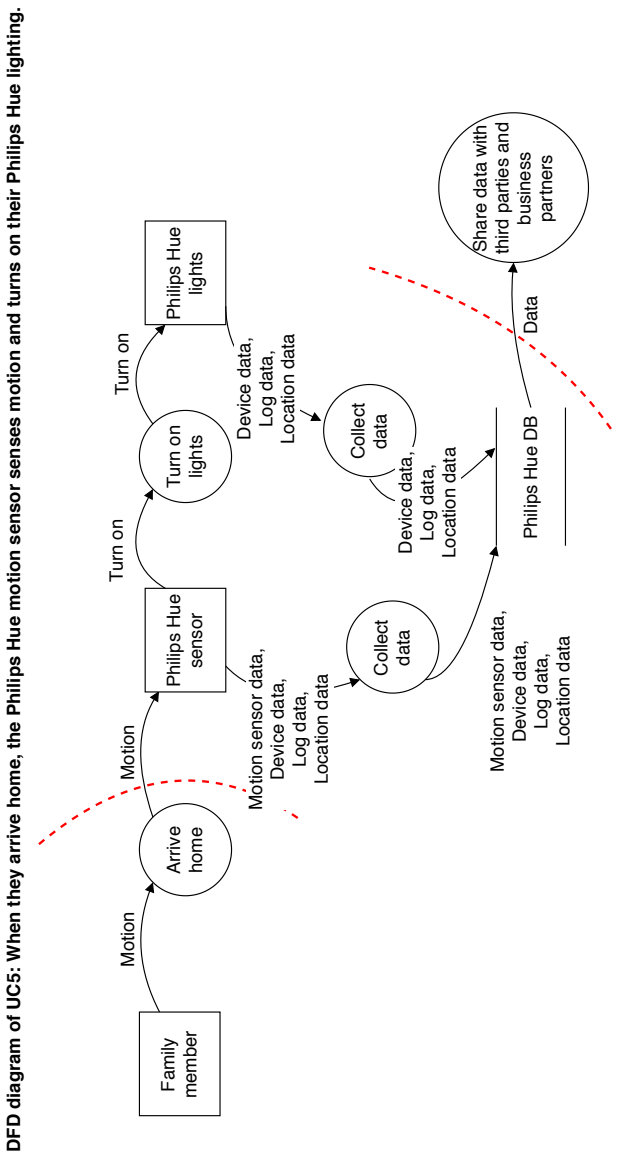


Figure 5.15: DFD for Use Case 5: *When they arrive home, the Philips Hue motion sensor senses motion and turns on their Philips Hue lighting.*

In Figure 5.15 there is only an indirect interaction between one of the family members and their Philips hue smart sensor, namely the motion sensor detects motion. This simple interaction however still becomes information that Philips hue collects in the form of motion sensor data, device data, log information and location data. And this data may then be shared with third parties.

Device/ Service	Source of Information	Information regarding Privacy	Information regarding Security
Android smart phone	Privacy Policy. [81] Official website. [82]	Google collects information a person provides directly to them; Information they collect about a persons use of their Services; and Information they obtain from third-party sources.	Built-in malware defense. Application sandboxing isolates and guards every Android app, stopping other apps from accessing your private information. Full on-device encryption.
Philips Hue	Privacy policy, official website and security advisory [90–92]. Paper by Ronen et al. [93]. PenTestPartners Blog [94].	Data collected: Email Address (Username), Password, Unique identifiers and configuration of products, Product usage and diagnostic information, Language preference, Usage and diagnostic analytics, Unique identifiers and configuration of products.	Implemented SSL connections in June 2018, used an AES encryption over HTTP earlier. Local communication is unsecured; the API key is sent through plaintext. A vulnerability in the ZigBee protocol allows for the creation of a worm which can brick Hue devices, jam wireless networks, infiltrate and exfiltrate data, or cause epileptic seizures.

Table 5.9: Security and privacy information gathered on the different devices, and their source.

With this information gathered for each of the use cases, the team moves on to the next step.

5.4 Privacy and Security Risk Modelling

After the team have performed a data flow analysis in conjunction with STRIDE, and have collected information regarding privacy and security through online

resources, the team can move on to the next step which is to create privacy and security risk models. For this step the team rely on their security and privacy experts to help with creating their CORAS models, and have to keep in mind the requirements regarding privacy and security of the family, and their assets.

So for each of the previous DFDs made, the team create a corresponding CORAS model.

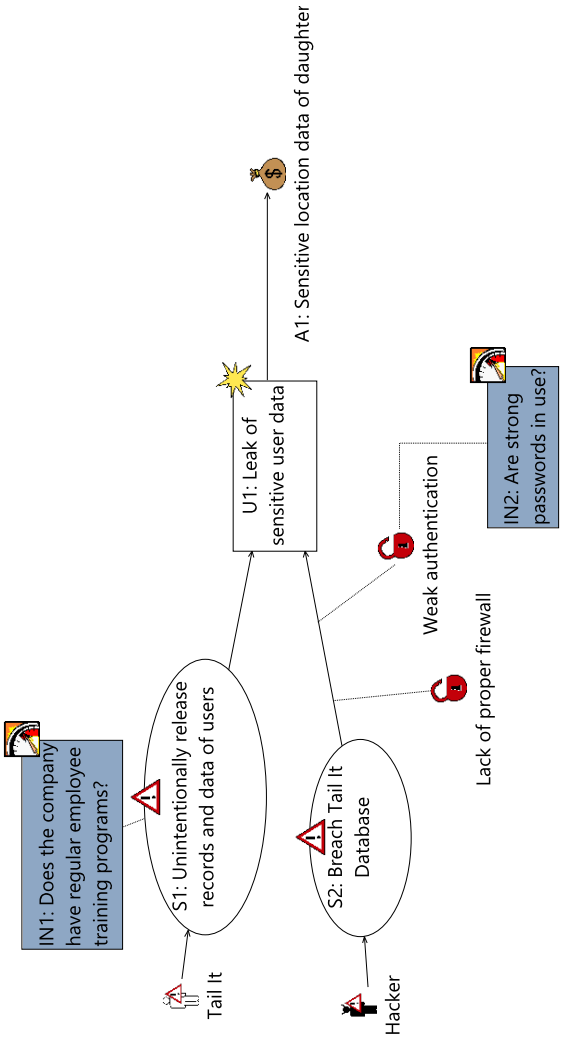


Figure 5.16: CORAS model for Use Case 1: *The first day of using the Tail It smart watch, the dad carefully checks on his daughter's location several times during her trip to school.*

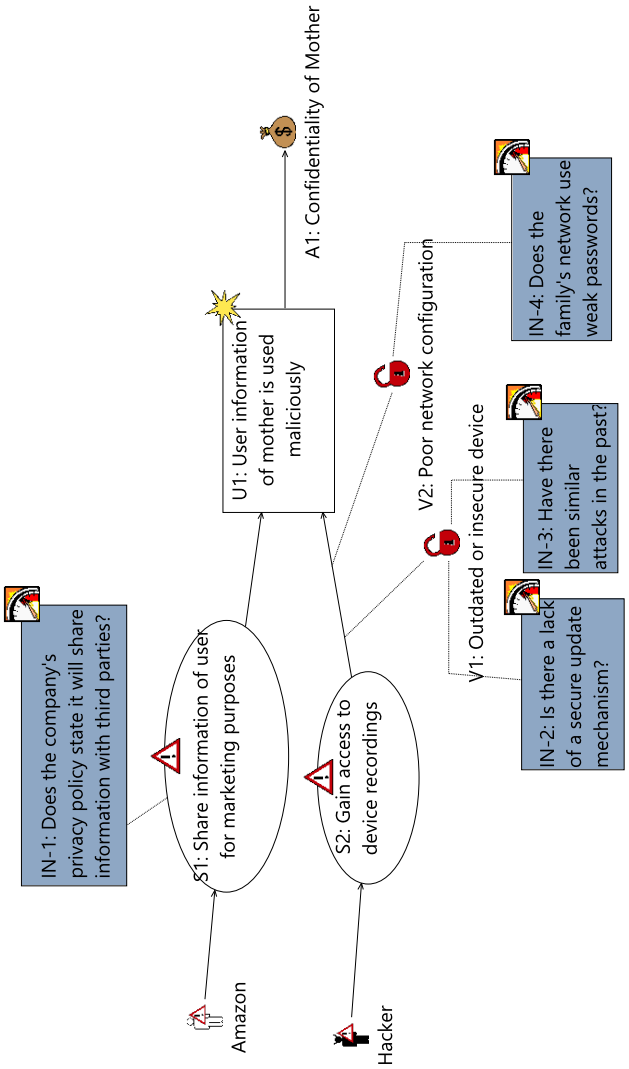


Figure 5.17: CORAS model for Use Case 2: *The mother uses Echo to buy package online.*

Figure 5.17 depicts a CORAS model derived from use case 2 and the DFD model. In the model we have one accidental threat, Amazon and one deliberate threat, a Hacker. Each of these are a source to a threat scenario. Amazon causes the threat scenario, S1: "Share information of user for marketing purposes", and the hacker is the source of the threat scenario S2: "Gain access to device recordings". Both of these threat scenarios lead to the unwanted incident, U1: "User information of mother is used maliciously", which again causes harm to the asset A1: "Confidentiality of mother".

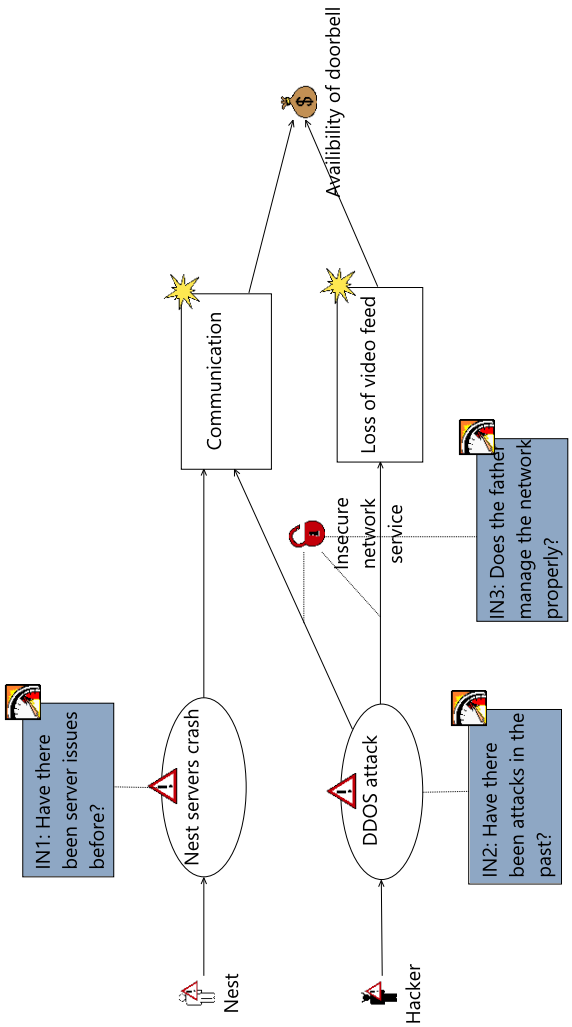


Figure 5.18: CORAS model for Use Case 3: *The family drives to cabin and are expecting a package delivery while away. The Nest automated doorbell senses motion, and alerts dad when package arrives.*

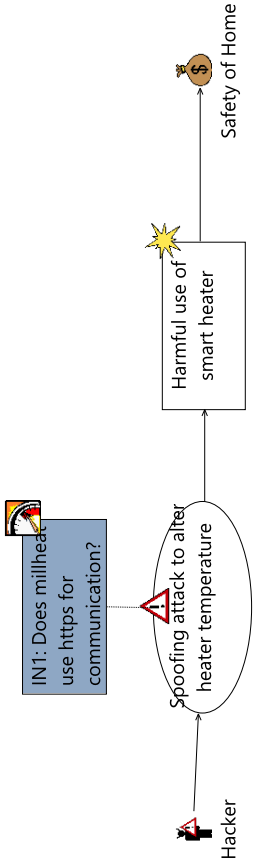


Figure 5.19: CORAS model for Use Case 4: *While coming home from cabin trip, dad wants to come home to a heated house. Using the Millheat app, he turns on heater to 22 degrees Celsius.*

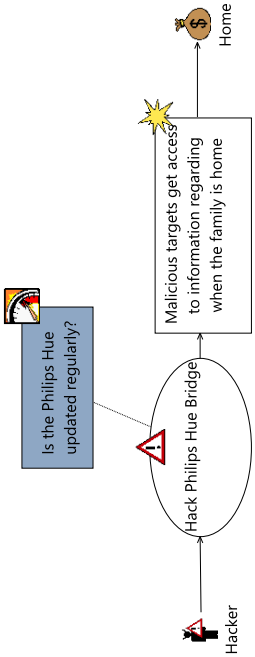


Figure 5.20: CORAS model for Use Case 5: *When they arrive home, the Philips Hue motion sensor senses motion and turns on their Philips Hue lighting.*

5.5 Translating Risk Models to Executable Algorithms

Once the CORAS models are complete, the team uses the method derived by Erdogan et al. in [68] and the DEXi tool to translate the CORAS diagrams into DEXi models. This is a straight forward method described earlier, and

The finished DEXi models are depicted in the following subsections. For all of the risk models the team will be using the scale {Very low, Low, Medium, High, Very High} to measure the overall risk.

5.5.1 DEXi Model for Risk model 1

Figure 5.21 shows a screenshot of the DEXi tool for use case 1: *The first day of using the Tail It smart watch, the dad carefully checks on his daughter's location several times during her trip to school.* The attributes IN-1 and IN-2 has the scale {Yes, No}, where no increases risk, and yes decreases risk. The other attributes will be using the scale {Very low, Low, Medium, High, Very High}. This is also the case for other 4 risk models.

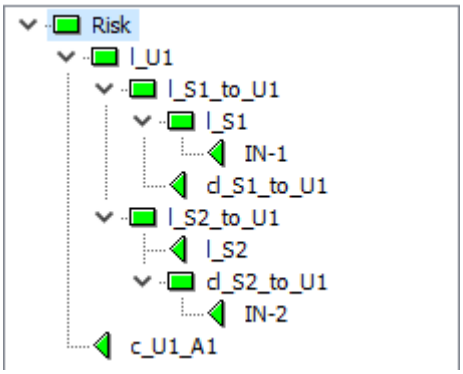
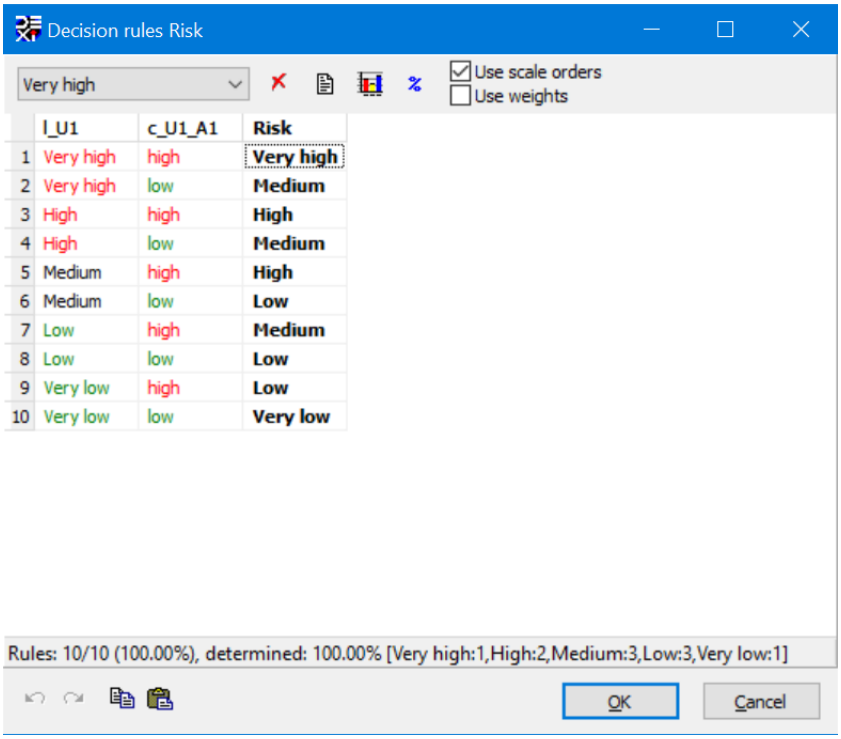


Figure 5.21: Screenshot from the DEXi Tool for Use Case 1

Figure 5.22 shows the decision rules for the root attribute Risk. All of the aggregate attributes have similar decision rules.



	l_U1	c_U1_A1	Risk
1	Very high	high	**Very high**
2	Very high	low	**Medium**
3	High	high	**High**
4	High	low	**Medium**
5	Medium	high	**High**
6	Medium	low	**Low**
7	Low	high	**Medium**
8	Low	low	**Low**
9	Very low	high	**Low**
10	Very low	low	**Very low**

Figure 5.22: Screenshot from the DEXi Tool for the utility function for risk

5.5.2 DEXi Model for Risk model 2

Figure 5.23 shows a screenshot of the DEXi tool for use case 2: *The mother uses Echo to buy package online.*

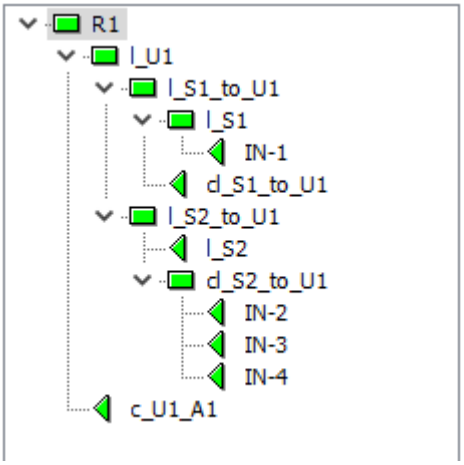


Figure 5.23: Screenshot from the DEXi Tool for Use Case 2

5.5.3 DEXi Model for Risk model 3

Figure 5.24 shows a screenshot of the DEXi tool for use case 3: *The family drives to cabin and are expecting a package delivery while away. The Nest automated doorbell senses motion, and alerts dad when package arrives.*

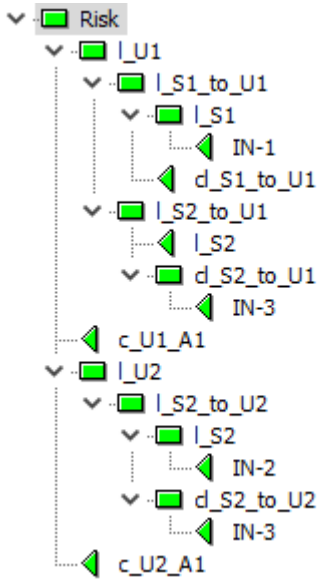


Figure 5.24: Screenshot from the DEXi Tool for Use Case 3

5.5.4 DEXi Model for Risk model 4

Figure 5.25 shows a screenshot of the DEXi tool for use case 4: *While coming home from cabin trip, dad wants to come home to a heated house. Using the Millheat app, he turns on heater to 22 degrees Celsius.*

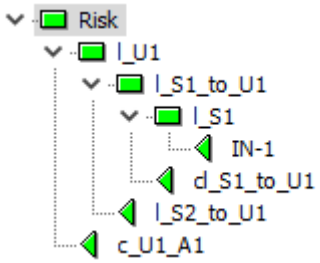


Figure 5.25: Screenshot from the DEXi Tool for Use Case 4

5.5.5 DEXi Model for Risk model 5

Figure 5.26 shows a screenshot of the DEXi tool for use case 5: *When they arrive home, the Philips Hue motion sensor senses motion and turns on their Philips Hue lighting.*

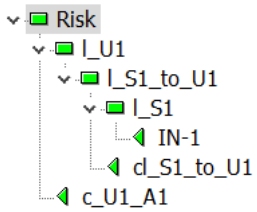


Figure 5.26: Screenshot from the DEXi Tool for Use Case 5

5.6 Executing Risk Assessment Algorithms Using Our Tool

For the final step, the team provides the tool with their CORAS diagrams, and their correlating DEXi models. The team then have a complete overview of the relations between the devices and services in the first part of the tool, alongside a depiction of the appropriate risk models, and their input variables in the second part of the tool.

Using the tool they may then select which Coras model and corresponding DEXi algorithm they wish to use for calculations, and then either manually provide input for the risk algorithms, or they may use the tool to automate the process by monitoring variables and send these to the tools API. This results in an automated risk analysis of the system based on the devices/services being monitored in the families home.

For a depiction of what the tool looks like in use, we provide a complete overview of use case 4 in Figures 5.27 and 5.28.

Figure 5.27: Complete overview of the tool part 1: The context modelling

In Figure 5.27 we see the relation between all of the components in the entire case, not just specific to use case 3. We also have a list of all the components on the right side.

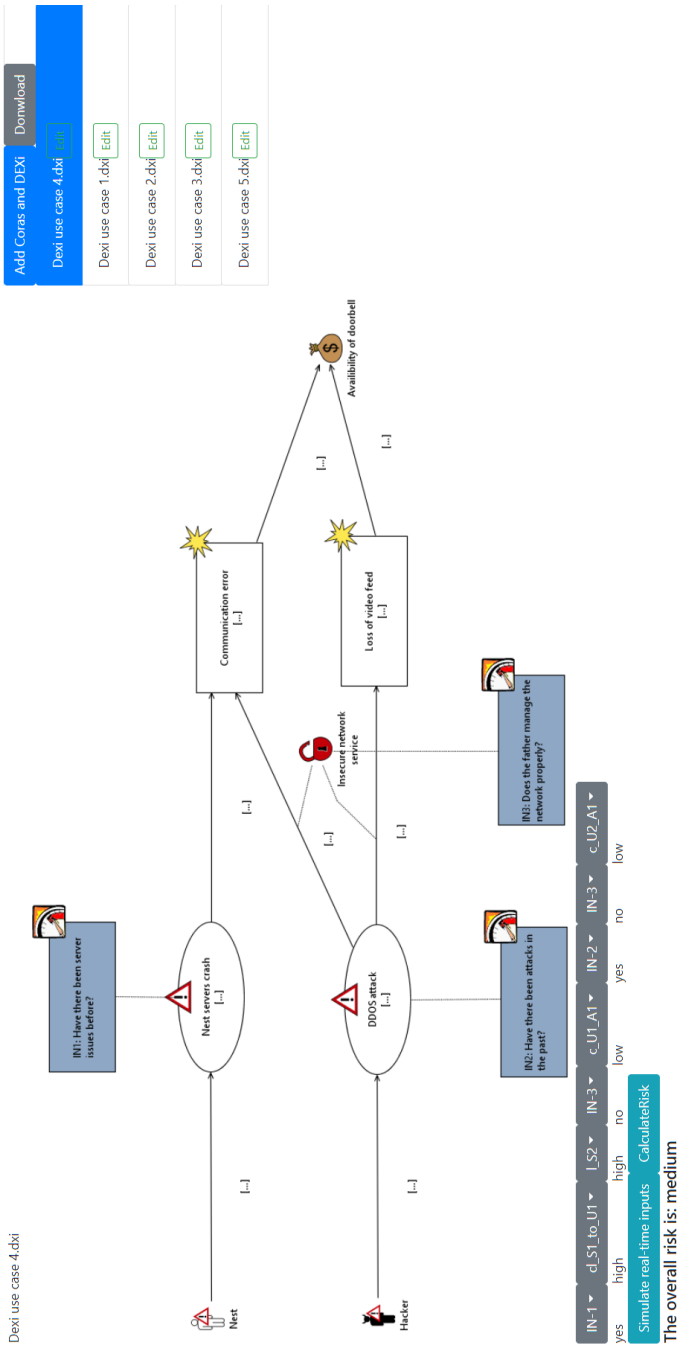


Figure 5.28: Complete overview of the tool part 2: The Execution of risk algorithms

In Figure 5.28 we see the CORAS diagram for use case 3, and right below the diagram we have inputs for each of the CORAS elements where the user may change the values as needed. Once the tool is given inputs for all variables, the tool executes the algorithm for the specific use case, and outputs the risk value. In this case the inputs were:

- IN1: Yes
- cl_S1_to_U1: High
- l_S1: High
- IN-3: No
- c_U1_A1: Low
- IN-2: Yes
- IN-3: No
- c_U2_A1: Low

Which gives the output: Medium. This output is derived from the DEXi model created manually by the team in the previous step.

5.7 Supporting Changes to the System

After having set up the families smart home, as well as deployed it, the IoT team starts to monitor the home. A week goes by and the family want to add a smart television to stream Netflix/Spotify on. They decide on the LG OLED55C8. The team following the DevOps practice, begin a new cycle, and the first step is again planning, so they start from the beginning of our method.

Step 1 is context management, the team already have a complete model of the families current smart home 5.10 and only have to add a single component which is done swiftly. An updated model is depicted in Figure 5.29.

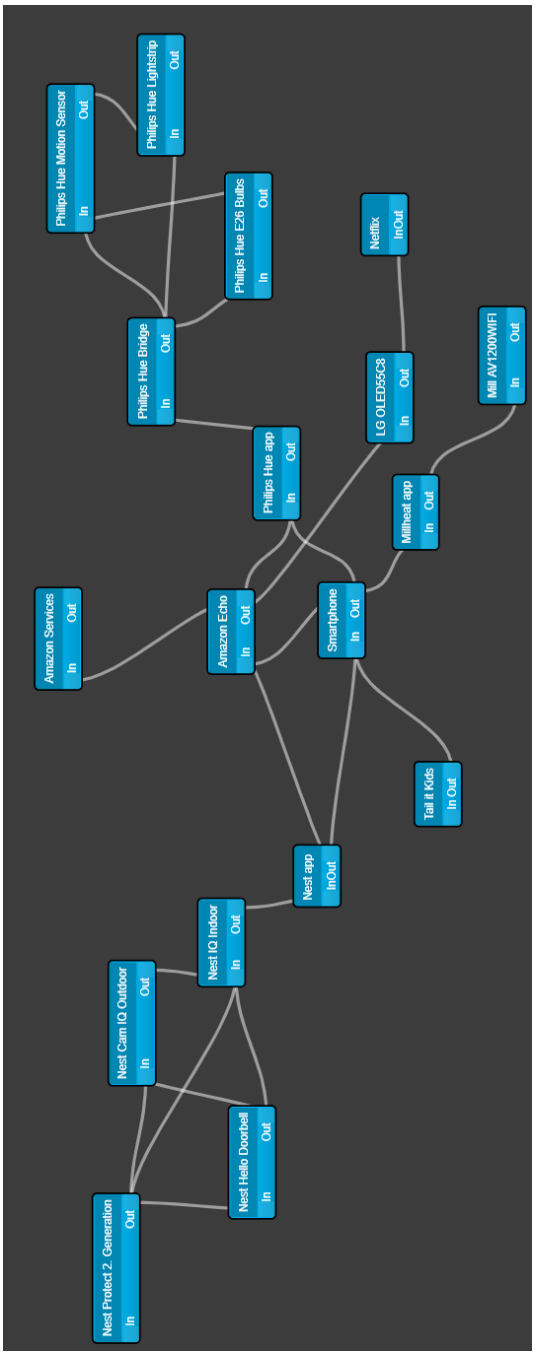


Figure 5.29: A revised context diagram including the devices and services to be used by the family

A new use case is made, listed below and depicted in Figure 5.30:

- The mother uses the smart TV to stream Netflix.

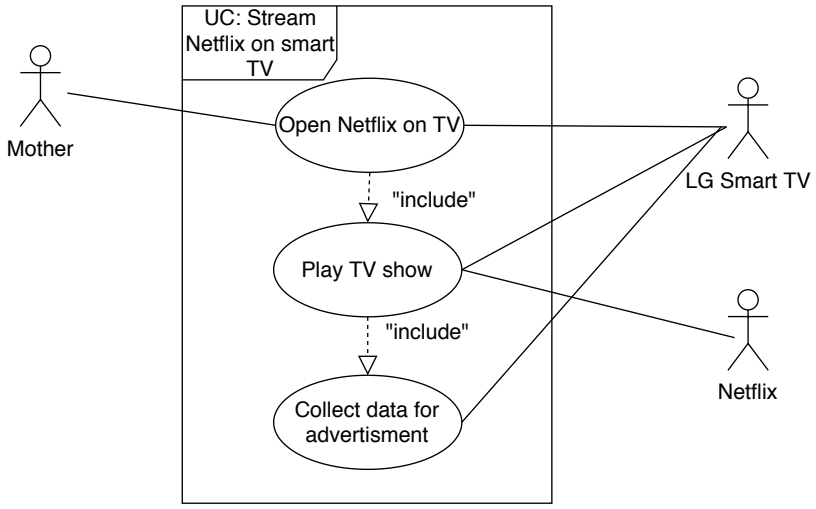


Figure 5.30: Use Case: *The mother uses the smart tv to stream Netflix*

Once the new context has been established, the team go on to step 2, creating a data flow diagram for the new use case, depicted in Figure 5.31.

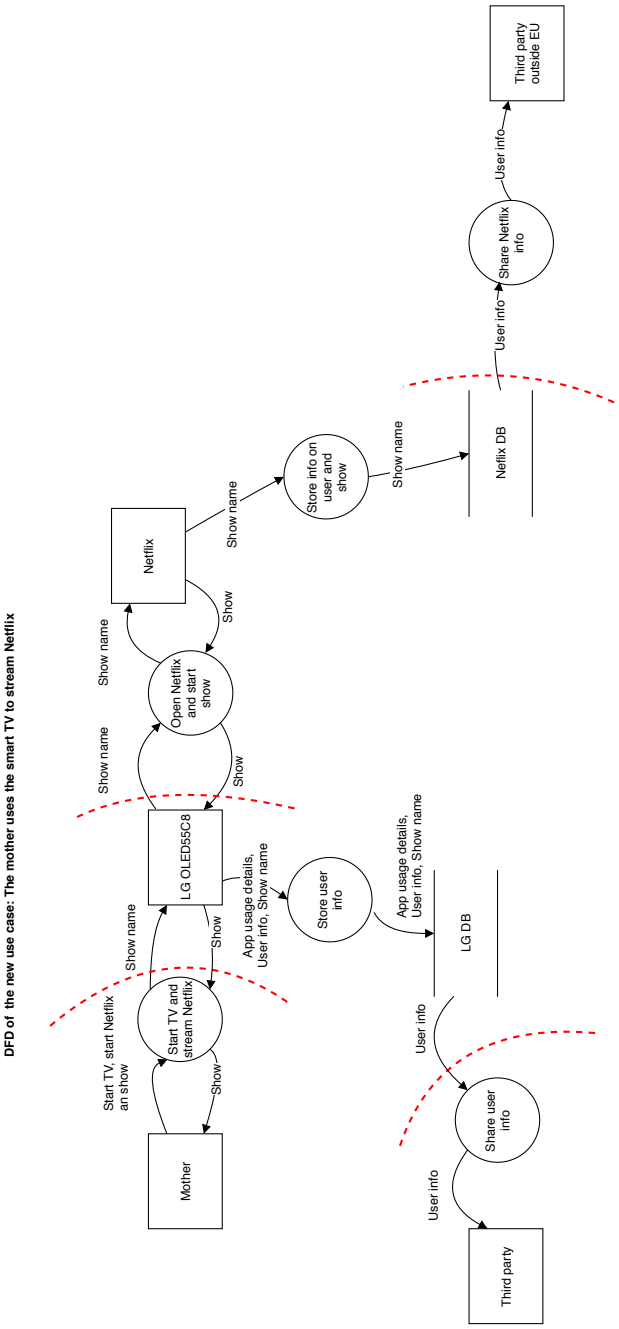


Figure 5.31: DFD for the new Use Case: *The mother uses the smart tv to stream Netflix*

In Figure 5.31 the team modelled a DFD diagram with trust boundaries for the use case considering when the Mother uses her smart TV to stream TV shows from Netflix. The mother boots up her TV and starts Netflix. Already information regarding her are use by both the LG smart TV and Netflix. Both companies store information regarding her, and what she watches, and both companies may share this information with third parties [95, 96].

Device/ Service	**Source of Information**	**Information regarding Privacy**	**Information regarding Security**
LG OLED 55C8	Privacy Policy.[96] Consumerreports.org [97]	The LG smart TV may collect information regarding the viewer, as well as what content the viewer is watching, regardless of whether or not the content is streamed. They may also share user information with third parties.	Consumerreports conducted in 2018 a test on security and privacy on many popular TV's including ones from LG, and found and unsecure API through the Roku platform.
Netflix	Privacy Policy. [95] Official website. [98] Article on her.ie [99].	Netflix collects your activity on the Netflix service, such as title selections, watch history and search queries. Netflix may also share information with third party partners.	There is not a whole lot of information regarding security when it comes to Netflix, however McAfee revealed that hackers had gained access to a number of accounts back in 2016.

Table 5.10: Security and privacy information gathered on the different devices, and their source.

After the DFD diagram is made, the team create a CORAS model depicted in Figure 5.32.

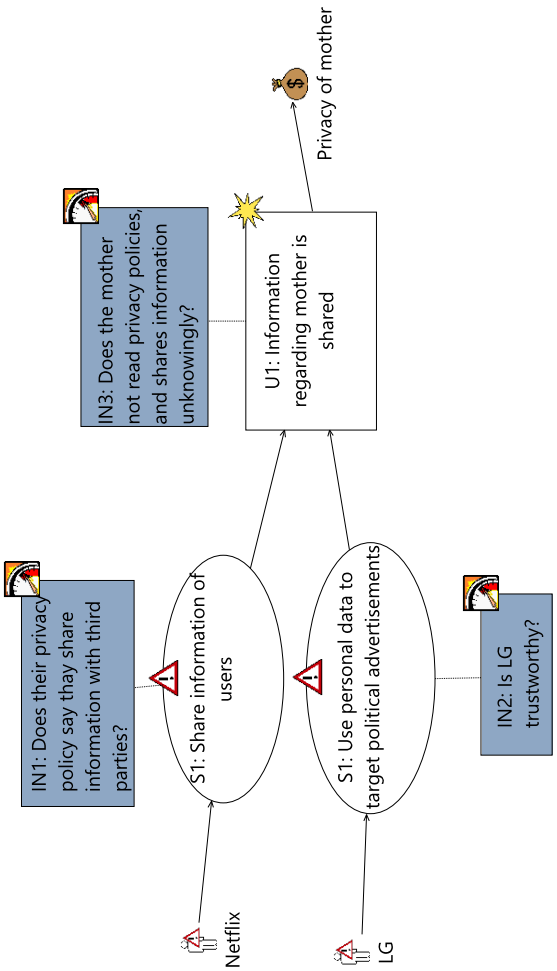


Figure 5.32: CORAS model for the new Use Case: *The mother uses the new smart TV to stream Netflix*

With a CORAS diagram made, the team can translate it into a DEXi algorithm depicted in Figure 5.33.

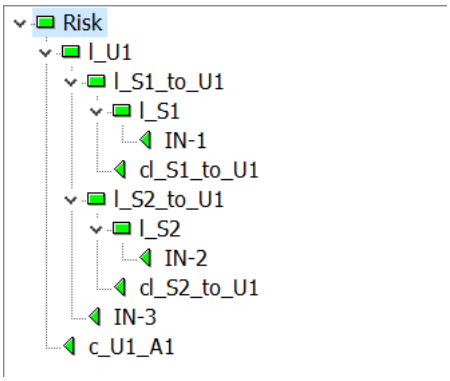


Figure 5.33: DEXi algorithm for the new Use Case: *The mother uses the new smart TV to stream Netflix*

Once the algorithm is made, the team may use the new CORAS model and DEXi algorithm with our tool, to determine risk, or automate risk calculations alongside the CORAS and DEXi models made previously.

Chapter 6

Discussion

In section 6.1 we go through the development process of the tool, what choices we made and why. In sections 6.2- 6.4 we discuss whether our thesis work has fulfilled the success criteria, and to what extent.

6.1 The Development Process

6.1.1 Selection of Frameworks

When choosing what frameworks to use with the proposed tool there are a few things to consider. Seeing as the tool will not be fully complete, and lightweight, we need a tool that is well documented and easy to understand. This will also help with having a quick prototype up and running, to further help guide the development of the tool, and help find areas of the tool which may require changes.

Another requirement for the tool is the possibility to import and export json. This will make it easier fo the tool to follow the RESTful architecture and easy to use with other API's.

Using a framework that is open source removes the need to depend on other people for continuous updates and allows us to build upon the tool for personal use. Depending on licenses used, it will also allow us to have our tool be open source as well.

We have also listed "free" as a requirement, this requirement is not as crucial as the other ones, but is a nice addition as it saves on cost. The final requirement is "No strict dependencies". This is with regards to large libraries such as React [100], Angular [101] and VueJS [102]. This requirement is mostly so I can develop the tool in a library I am comfortable with and allows for a slightly quicker development process.

The possible contenders for the modelling tool were: Draw2D [103], mxGraph [104], goJS [105], React Diagrams [106] and JointJS [107]. These are listen in the table below, showing whether or not they fulfil the requirements.

	Well documented	Import/export Json	Open Source	Free	No strict dependencies
Draw2D	✓	✓	✓	✓	✓
mxGraph	✗	✗	✓	✓	✓
goJS	✓	✓	✗	✗	✓
React Diagrams	✓	✓	✓	✓	✗
JointJS	✓	✓	✓	✓	✗

Table 6.1: Selection of frameworks

The only framework to fulfil all of the requirements was Draw2D, but after looking through documentation and the open source code, it was clear that further development of the framework was not a guarantee. Seeing as we may build upon our tool in the future, a more sustainable framework is a must. Comparing the other frameworks, React Diagrams and JointJS were the two final possibilities, and we ended up going with React Diagrams, as they offered a more flexible tool, with easier code to modify and build upon.

6.1.2 The Development

After choosing React Diagrams for our modelling component, we had to set up our JavaScript front-end, alongside typescript as this is one of the dependencies of React Diagrams. DEXi, being a part of the method, should also be integrated with the tool. DEXi is a stand-alone windows application, but the creators also offer open source libraries in both Java and C#. We decide on using JDEXi, the Java library as we have more experience with Java. We therefore had to setup a Java back-end server using Spring [108]. Spring is one of the most popular Java back-end frameworks, and easy to get up and running. Using spring it as easy to setup a restful API on the back-end, so that the front-end and other potential services could easily execute DEXi calculations. With the API setup, it was easy to simulate potential monitors with different risk values by calling on our API.

Spring would also have to extract and store data from files provided by the user, so the use of the cloud based database MongoDB Atlas [109] was easy to set up, and used to handle these files. As MongoDB Atlas is cloud based, database capacity is easy to expand upon if there should ever be a need.

Deployment is done using Maven with hosting on the Google Cloud Platform. This is done for quick and easy building of both the front-end and back-end, hosted on a cloud service for cheap hosting and good uptime.

Our tool is open source and located on github with example files provided:

https://github.com/ribako/risk-tool-frontend and
https://github.com/ribako/risk-tool-backend

The tool can be accessed on:

https://risktool-586bf.appspot.com/

6.2 Success Criterion 1

The first criterion states: *The tool and method must be easy to use and understandable for developers.*

The first module of our tool is the modelling tool. This module is used for context establishment, where the user creates system diagrams to depict the services and devices to be used in a given system. The modelling is fairly straight forward, and resembles many other popular modelling languages such as UML class diagrams. Developers often have experience with UML, it is therefore reasonable to argue our modelling tool is easy to use and understand by developers.

Developers don't always prioritize security, but they often need to consider security when developing software [110, 111]. For capturing risk models, we chose CORAS because it has been empirically shown that the language is intuitively simple for stakeholders with different backgrounds [66]. We may therefore argue that the risk-model section of the tool is reasonably easy to understand by developers. CORAS is also based on the international ISO standards like ISO 27005 and ISO 31000. As part of development activities, developers often use concepts such as threat, unwanted incident, vulnerability and risk [110, 111].

The design of the tool is made such that roughly half the screen is to be used for modelling of the IoT system, while the bottom half is for depicting the CORAS models, and executing their respective DEXi algorithms. This design follows the design of the method, in that the user starts by using the tool to designing the IoT system, then later uses the tool again to visualize the CORAS diagrams, and then execute the algorithms corresponding to the CORAS models. This helps make it pedagogical and easy to use.

6.3 Success Criterion 2

The second criterion states: *The tool-supported method should support the planning of trustworthy smart IoT systems in the DevOps practice in terms of security and privacy risk assessment.*

Our tool-supported method is constructed to fit within the DevOps planning phase, and with security and privacy risk assessment in mind. The planning phase of DevOps is the first activity of the DevOps cycle. This first step either has initial input from developers/domain experts (when first starting the DevOps process), or receives additional input from the final monitoring step of the cycle. The input for this step is therefore either initial input from developers/domain experts as part of initiating the DevOps process, or additional input from the monitor. The planning phase must however output something of value to the next step, coding. Figure 6.3 shows the DevOps cycle. As our tool and method may receive risk assessment values from the monitor, and supports changes based on this input, we may argue that our tool and method satisfies

the first criteria of the planning phase. Similarly, our tool and method produces risk values as an output, which may be used in the next step: "code" as to help decide what areas need to be worked on regarding privacy and security risks.

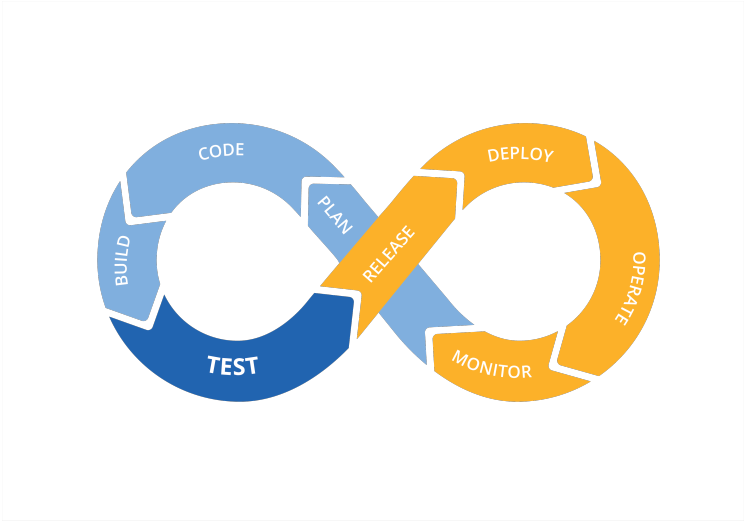


Figure 6.1: The DevOps development cycle.

The DevOps practice is all about automation, measurements, and tools. There exists many tools for DevOps users to use for the different steps of the cycle. Typical tools are Git [112] for code management, Gradle [113] or Maven [44] for build automation, Jenkins [43] for automation of building, testing and deployment/release, and Nagios [114] for monitoring. While these tools are powerful and greatly help the steps mentioned, there is a serious lack of tool support for the planning phase [48]. We mentioned Jira [46] as a popular planning tool, but other than this there are few popular planning tools. By building a tool to suit planning in DevOps for privacy and security risk assessment, we fill a gap.

Since our tool has an API for input and execution of our DEXi risk assessment algorithms, we facilitate automatic and real-time risk assessment. This can be seen in the tool by clicking on the "Simulate real-time" button, where the front-end makes frequent API calls to the back-end to simulate input from monitors. Thus, by using the DEXi API provided by our tool and feeding the risk model algorithms with risk-indicator values from a running system, it is possible to support automatic real-time risk assessment.

6.4 Success Criterion 3

The third criterion states: *The tool and method should be appropriate for use in the DevOps practice in terms of adapting to new plans and flexible in response*

to changes in the system.

We can argue that due to the flexible nature of the method, it is appropriate to use within the agile DevOps planning phase. If there are changes made to the system in regards to architecture after risk diagrams and risk assessment have been created, it is entirely possible to create new risk models and algorithms.

Our tool and method supports modular modelling. The user may focus on small parts of the system, creating smaller context diagrams, specific risk models for the given context, and risk assessment algorithms. This allows for the possibility of assessing risk with regards to specific parts of a system. It also allows for flexibility when making changes to the system, as not all parts depend on each other. On the other hand one may also create large context diagrams, encapsulating larger parts of the system. This may be helpful if we want to create a risk picture of a larger part of the system. One may then also create large CORAS diagrams, which results in fewer but larger risk assessment diagrams. Because of the possibility of creating either small diagrams and algorithms, large diagrams and algorithms, or a combination of large and small diagrams and algorithms, we may argue that the tool and method is flexible in response to changes to the system.

In Chapter 5 we presented an application of our tool and method, and showed that both the tool and method properly supports frequent changes to the system architecture. We did this by following the steps of our method in a smart home case. The first step was context establishment, where we presented the case and the targeted system. The case consisted of an imagined family recruiting an imagined DevOps IoT team, who applied our method. They created a context diagram using our tool, alongside use case diagrams. Further they created data flow diagrams, then risk assessment diagrams, and finally DEXi algorithms. Once the team had finished the first iteration of the DevOps cycle, the family requested a change to the system in the form of adding a new device, a smart TV. So the team begin anew with the planning phase. They already have a context diagram, and can easily add the TV to create a new context. The TV does not affect many other devices, so the team need only to focus on the new device, and create a DFD. From there they continue with privacy and security risk modelling, and finally translating those risk models to executable risk assessment algorithms.

Chapter 7

Conclusion

In our introduction, we established that risks related to security, trust and privacy is a big challenge for IoT systems, and that the need for proper security is important. Further, due to devices having constraints on cost, time to market and functionality, many developers disregard the assessment of these risks. IoT systems typically operate in highly dynamic environments, and need to be able to evolve and adapt. By examining the current state-of-the art of risk-driven planning of IoT systems, we concluded that there is a lack of support for trustworthy smart IoT systems, and also a lack of tool-support for these systems, in the DevOps practice [3, 4, 48]. Thus, in this thesis we developed both a risk-driven method and tool to support the trustworthy execution of IoT systems, and provide the following contributions:

- A method with the purpose of assisting developers in the planning phase of DevOps with identifying security and privacy risks, and executing risk assessment algorithms.
- A tool to support our method, and facilitate automatic real-time monitoring of risk.
- A case study to demonstrate the feasibility of the tool as well as the method in which the tool is used.

We applied our tool and method in a smart home case to simulate a real-life scenario, and to show how both the method and tool may be used, and how it suits the DevOps practice by supporting changes to the system. The tool may also simulate real-time input gathering from monitors though our custom made API.

In the following sections we discuss directions for future work related to out tool and method and threats to validity.

7.1 Future Work

We identify several directions for future work that may be of interest:

- Empirical investigations to determine the usability of the tool and method can be carried out. A usability study might uncover features that are not properly expressed to the intended user, or absence of behaviour required to fulfil the tool's potential. This can provide valuable insight toward how the tool can be refined to suit the needs of developers and potential stakeholders.
- Implement the real-time monitoring feature properly such that the privacy and security risk assessments can be calculated live, so that the user is given an updated, comprehensive view of the overall risk in the given IoT system.
- Implement a CORAS modelling component to the tool, so that one does not need to rely on external tools. This will also help in depicting large CORAS diagrams in the tool, as currently the tool only takes static images which can not be resized. This would also allow for the possibility of automatically generating DEXi algorithms "trees" without scales based on the CORAS diagrams. This way the developers would only need to create the scales and decision trees for the DEXi algorithms.

7.2 Threats to Validity

The tool supported method has currently been carried out in its entirety only by the author. This was deliberate in order to evaluate the applicability of our tool supported method. However, this is also a threat to validity in terms of the feasibility and applicability of our tool supported method by potential users and its appropriateness for DevOps. As already mentioned in future work, these aspects needs to be addressed in case studies and experiments. On the positive side, however, we know that CORAS has been tried out in practice in sharp industrial cases [60]. The DEXi tool has also been tried out in industrial cases as reported by the author of DEXi [61, 115]. The STRIDE method including DFD diagrams is one of the most popular approaches in identifying cyber risks with respect to data flow [116]. For example [117]. And finally, the schematic translation of CORAS risk models to machine-readable risk assessment algorithms have been tried out in full-scale industrial pilots in the EU projects WISER [118] and CYBERWISER.eu [119]. In other words, the different parts of our tool-supported method have been thoroughly tried out individually in real-life industrial settings, which in turn can support the feasibility and applicability of our method.

Our tool-supported method integrates the above-mentioned approaches in one tool-supported method, and we therefore do acknowledge that our tool-supported method must be thoroughly tried out in its entirety in real-life DevOps setting to assess its feasibility and applicability.

Bibliography

[1] *IoT 2020: Smart and secure IoT platform*. 2016. URL: http://www.iec.ch/whitepaper/pdf/iecWP-IoT2020-LR.pdf.

[2] Claudio A. Ardagna, Ernesto Damiani, Julian Schütte and Philipp Stephanow. 'A Case for IoT Security Assurance'. In: *Internet of Everything*. Springer, 2018, pp. 175–192.

[3] A. Taivalsaari and T. Mikkonen. 'A Roadmap to the Programmable World: Software Challenges in the IoT Era'. In: *IEEE Software* 34.1 (2017), pp. 72–80.

[4] *DevOps in the Internet of Things*. 2016. URL: http://events.windriver.com/wrcd01/wrcm/2016/08/WP-devops-in-the-internet-of-things.pdf.

[5] ISO/IEC JTC 1. *Internet of Things*. Geneva, 2014.

[6] Flavio Bonomi, Rodolfo Milito, Jiang Zhu and Sateesh Addepalli. 'Fog computing and its role in the internet of things'. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM. 2012, pp. 13–16.

[7] Kevin Ashton. 'That 'internet of things' thing'. In: *RFID journal* 22.7 (2009), pp. 97–114.

[8] In Lee and Kyoochun Lee. 'The Internet of Things (IoT): Applications, investments, and challenges for enterprises'. In: *Business Horizons* 58.4 (2015), pp. 431–440.

[9] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic and Marimuthu Palaniswami. 'Internet of Things (IoT): A vision, architectural elements, and future directions'. In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[10] Luigi Atzori, Antonio Iera and Giacomo Morabito. 'The internet of things: A survey'. In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[11] *Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016*. 2017. URL: https://www.gartner.com/newsroom/id/3598917.

[12] Zhuming Bi, Li Da Xu and Chengen Wang. 'Internet of things for enterprise systems of modern manufacturing'. In: *IEEE Transactions on industrial informatics* 10.2 (2014), pp. 1537–1546.

[13] Rodrigo Roman, Jianying Zhou and Javier Lopez. 'On the features and challenges of security and privacy in distributed internet of things'. In: *Computer Networks* 57.10 (2013), pp. 2266–2279.

[14] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*. Vol. 1. Prentice Hall Upper Saddle River, 2002.

[15] Corey Ladas. *Scrumban-essays on kanban systems for lean software development*. Lulu.com, 2009.

[16] Jim Highsmith and Alistair Cockburn. 'Agile software development: The business of innovation'. In: *Computer* 34.9 (2001), pp. 120–127.

[17] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[18] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Robert Martin C., Steve Mellor, Dave Thomas, Brian Marick, Ken Schwaber and Jeff Sutherland. *Manifesto for agile software development*. 2001. URL: http://www.agilemanifesto.org/.

[19] Vidas Vasiliauskas. 'Developing agile project task and team management practices'. In: *Eylean* (2014).

[20] Ron Jeffries, Ann Anderson and Chet Hendrickson. *Extreme programming installed*. Addison-Wesley Professional, 2001.

[21] Lisa Crispin and Janet Gregory. *Agile testing: A practical guide for testers and agile teams*. Pearson Education, 2009.

[22] Scott Ambler. *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002.

[23] Constantine Aaron Cois, Joseph Yankel and Anne Connell. 'Modern DevOps: Optimizing software development through effective system interactions'. In: *Professional Communication Conference (IPCC), 2014 IEEE International*. IEEE. 2014, pp. 1–7.

[24] Michael Hüttermann. *DevOps for developers*. Apress, 2012.

[25] A. Avizienis, J. C. Laprie, B. Randell and C. Landwehr. 'Basic concepts and taxonomy of dependable and secure computing'. In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (2004), pp. 11–33.

[26] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith and Pete Steggles. 'Towards a better understanding of context and context-awareness'. In: *International Symposium on Handheld and Ubiquitous Computing*. Springer. 1999, pp. 304–307.

[27] Edward R Griffor, Christopher Greer, David A Wollman and Martin J Burns. *Framework for cyber-physical systems: Volume 1, overview*. Tech. rep. 2017.

[28] Zheng Yan, Peng Zhang and Athanasios V Vasilakos. 'A survey on trust management for Internet of Things'. In: *Journal of network and computer applications* 42 (2014), pp. 120–134.

[29] Kaiyu Wan and Vangalur Alagar. 'Integrating context-awareness and trustworthiness in IoT descriptions'. In: *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*. IEEE. 2013, pp. 1168–1174.

[30] *Principles and Guidelines on Implementation*. Standard. Geneva, CH: International Organization for Standardization, 2009.

[31] Atle Refsdal, Bjørnar Solhaug and Ketil Stølen. 'Cyber-risk management'. In: *Cyber-Risk Management*. Springer, 2015, pp. 33–47.

[32] Jason I Hong, Jennifer D Ng, Scott Lederer and James A Landay. 'Privacy risk models for designing privacy-sensitive ubiquitous computing systems'. In: *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM. 2004, pp. 91–100.

[33] NIST. *National Institute of Standards and Technology*. 2019. URL: https://www.nist.gov (visited on 15/10/2019).

[34] NIST. 'PRELIMINARY DRAFT: NIST PRIVACY FRAMEWORK: A TOOL FOR IMPROVING PRIVACY THROUGH ENTERPRISE RISK MANAGEMENT'. 6th Sept. 2019.

[35] *Privacy framework*. Standard. Geneva, CH: International Organization for Standardization, 2011.

[36] Hewlett Packard Enterprise. *Application security and devops*. Tech. rep. Technical report, Hewlett Packard Enterprise, 2016.

[37] Ian Head and Neil MacDonald. *DevSecOps: How to Seamlessly Integrate Security Into DevOps*. Tech. rep. Technical report, Gartner, 2016.

[38] Puppet. *Puppet*. 2019. URL: https://puppet.com/ (visited on 15/10/2019).

[39] Puppet. *2019 State of DevOps Report*. 2019. URL: https://puppet.com/resources/whitepaper/state-of-devops-report (visited on 15/10/2019).

[40] Håvard Myrbakken and Ricardo Colomo-Palacios. 'DevSecOps: a multivocal literature review'. In: *International Conference on Software Process Improvement and Capability Determination*. Springer. 2017, pp. 17–29.

[41] Milan Zeleny. 'Multiple criteria decision making: Eight concepts of optimality'. In: *Human Systems Management* 17.2 (1998), pp. 97–107.

[42] S. Gupta, V. Muntes-Mulero, P. Matthews, J. Dominiak, A. Omerovic, J. Aranda and S. Seycek. 'Risk-Driven Framework for Decision Support in Cloud Service Selection'. In: 2015, pp. 545–554.

[43] Jenkins. *Jenkins*. 2018. URL: https://jenkins.io/ (visited on 10/10/2018).

[44] Maven. *Maven*. 2018. URL: https://maven.apache.org/ (visited on 10/10/2018).

[45] New Relic. *New Relic*. 2018. URL: https://newrelic.com/ (visited on 10/10/2018).

[46] Atlassian. *Jira*. 2018. URL: https://www.atlassian.com/software/jira (visited on 10/10/2018).

[47] Gene Kim, Patrick Debois, John Willis and Jez Humble. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution, 2016.

[48] Phu Nguyen, Nicolas Ferry, Gencer Erdogan, Hui Song, Stéphane Lavirotte, Jean-Yves Tigli and Arnor Solberg. 'Advances in deployment and orchestration approaches for iot-a systematic review'. In: *2019 IEEE International Congress on Internet of Things (ICIOT)*. IEEE. 2019, pp. 53–60.

[49] Riskwatch. *Riskwatch*. 2019. URL: https://riskwatch.com/ (visited on 20/10/2019).

[50] Bilge Karabacak and Ibrahim Sogukpinar. 'ISRAM: information security risk analysis method'. In: *Computers & Security* 24.2 (2005), pp. 147–159.

[51] *Merriam-Webster*. 2018. URL: https://www.merriam-webster.com (visited on 15/08/2018).

[52] Ketil Stolen and Ida Solheim. *Technology research explained*. Tech. rep. SINTEF ICT Technical Report: 2006-A313, 200.

[53] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[54] Joseph Edward McGrath. *Groups: Interaction and performance*. Vol. 14. Prentice-Hall Englewood Cliffs, NJ, 1984.

[55] Marvin V Zelkowitz and Dolores R. Wallace. 'Experimental models for validating technology'. In: *Computer* 31.5 (1998), pp. 23–31.

[56] Per Runeson, Martin Host, Austen Rainer and Bjorn Regnell. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[57] Robert K Yin. *Case study research and applications: Design and methods*. Sage publications, 2017.

[58] I Sommerville. *Software Engineering, Boston, Massachusetts: Pearson Education*. 2011.

[59] Walter F Tichy. 'Should computer scientists experiment more?' In: *Computer* 31.5 (1998), pp. 32–40.

[60] Mass Soldal Lund, Bjørnar Solhaug and Ketil Stølen. *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010.

[61] *DEXi: A Program for Multi-Attribute Decision Making*. 2018. URL: https://kt.ijs.si/MarkoBohanec/dexi.html (visited on 21/03/2019).

[62] Larry L Constantine and Edward Yourdon. *Structured design: fundamentals of a discipline of computer program and systems design*. Yourdon, 1978.

[63] Chris Gane and Trish Sarson. 'Structured Systems Analysis: Tools and Techniques, Improved Systems Technologies'. In: *New York, New York* (1977).

[64] Tom DeMarco. 'Structure analysis and system specification'. In: *Pioneers and Their Contributions to Software Engineering*. Springer, 1979, pp. 255–288.

[65] JD Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla and Anandha Murukan. *Improving web application security: threats and countermeasures*. Vol. 3. Microsoft Corporation Redmond, 2003.

[66] Bjørnar Solhaug and Ketil Stølen. 'The CORAS Language-Why it is designed the way it is'. In: *Proc. 11th International Conference on Structural Safety and Reliability (ICOSSAR'13)*. 2013, pp. 3155–3162.

[67] Lead Author Org, Atle Refsdal, Gencer Erdogan, Giorgio Aprile AON, Romina Colciago AON, Ales Cernivec, Alberto Biasibetti AON, Antonio Alvarez and Susana González. 'D3. 1 - CYBER RISK PATTERNS'. In: (2017).

[68] Gencer Erdogan and Atle Refsdal. 'A method for developing qualitative security risk assessment algorithms'. In: *International Conference on Risks and Security of Internet and Systems*. Springer. 2017, pp. 244–259.

[69] Dave. *Consumer attitudes to smart technology systems across the home*. 2018. URL: https://www.eurekaresearch.co.uk/single-post/2018/03/09/Consumer-attitudes-to-smart-technology-systems-across-the-home (visited on 10/09/2019).

[70] Earlence Fernandes, Jaeyeon Jung and Atul Prakash. 'Security analysis of emerging smart home applications'. In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2016, pp. 636–654.

[71] Franco Loi, Arunan Sivanathan, Hassan Habibi Gharakheili, Adam Radford and Vijay Sivaraman. 'Systematically evaluating security and privacy for consumer IoT devices'. In: *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. ACM. 2017, pp. 1–6.

[72] Kaushal Kafle, Kevin Moran, Sunil Manandhar, Adwait Nadkarni and Denys Poshyvanyk. 'A Study of Data Store-based Home Automation'. In: *arXiv preprint arXiv:1812.01597* (2018).

[73] *Case 20 It's getting hot in here*. 2019. URL: https://blog.roysolberg.com/2019/01/mill-heat (visited on 17/02/2019).

[74] *Car Hacking Research: Remote Attack Tesla Motors*. 2016. URL: https://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/ (visited on 17/02/2019).

[75] Vivint. *Vivint Smarthome*. 2019. URL: https://www.vivint.com (visited on 08/09/2019).

[76] Smartn. *smartn Smarthome*. 2019. URL: https://smartn.co.uk/ (visited on 08/09/2019).

[77] Tail It. *Tail It*. 2019. URL: https://www.tailit.com/ (visited on 10/09/2019).

[78] Tail It. *Tail It*. 2019. URL: https://www.tailit.com/privacy-policy/ (visited on 10/09/2019).

[79] Mnemonic. *Mnemonic*. 2019. URL: https://www.mnemonic.no/ (visited on 10/09/2019).

[80] Øyvind H. Kaldestad. *Elendig sikkerhet i smartklokker for barn*. 2017. URL: https://www.forbrukerradet.no/siste-nytt/elendig-sikkerhet-i-smartklokker-for-barn/ (visited on 26/08/2019).

[81] Google. *Google privacy and terms*. 2019. URL: https://policies.google.com/privacy?hl=en-US (visited on 10/09/2019).

[82] android. *android*. 2019. URL: https://www.android.com/ (visited on 10/09/2019).

[83] Amazon. *Amazon privacy notice*. 2019. URL: https://www.amazon.co.uk/gp/help/customer/display.html/ref=gss?nodeId=502584 (visited on 10/11/2019).

[84] Amazon. *Amazon*. 2019. URL: https://www.amazon.co.uk/ (visited on 10/09/2019).

[85] *Hackers found a (not-so-easy) way to make the amazon echo a spy bug*. 2018. URL: https://www.wired.com/story/hackers-turn-amazon-echo-into-spy-bug/ (visited on 14/02/2019).

[86] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates and Michael Bailey. 'Skill squatting attacks on amazon alexa'. In: *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 2018, pp. 33–47.

[87] Nest. *Privacy Statement for Nest Products and Services*. 2019. URL: https://nest.com/uk/legal/privacy-statement-for-nest-products-and-services/ (visited on 10/11/2019).

[88] Nest. *Nest*. 2019. URL: https://nest.com/ (visited on 10/09/2019).

[89] Millheat. *TERMS OF USE & PRIVACY POLICY MILLHEAT APP*. 2019. URL: https://www.millheat.com/privacy-policy-millheat-app (visited on 10/11/2019).

[90] Philips. *Philips Hue*. 2019. URL: https://www2.meethue.com/en-us (visited on 10/09/2019).

[91] Philips Hue. *Additional Privacy Notice for Philips Hue customers*. 2019. URL: https://www2.meethue.com/en-us/support/privacy-policy (visited on 10/11/2019).

[92] Philips Hue. *Security Advisory*. 2019. URL: https://www2.meethue.com/en-gb/support/security-advisory (visited on 10/11/2019).

[93] Eyal Ronen, Adi Shamir, Achi-Or Weingarten and Colin O'Flynn. 'IoT goes nuclear: Creating a ZigBee chain reaction'. In: *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE. 2017, pp. 195–212.

[94] *Hijacking Philips Hue*. 2016. URL: https://www.pentestpartners.com/security-blog/hijacking-philips-hue/ (visited on 14/02/2019).

[95] Netflix. *Privacy Statement*. 2019. URL: https://help.netflix.com/legal/privacy (visited on 10/11/2019).

[96] LG. *Privacy Policy*. 2019. URL: https://www.lg.com/uk/privacy (visited on 10/11/2019).

[97] *Samsung and Roku Smart TVs Vulnerable to Hacking, Consumer Reports Finds*. 7th Feb. 2018. URL: https://www.consumerreports.org/televisions/samsung-roku-smart-tvs-vulnerable-to-hacking-consumer-reports-finds/ (visited on 10/10/2019).

[98] Netflix. *Netflix*. 2019. URL: https://www.netflix.com/ (visited on 10/11/2019).

[99] her. *Netflix has been hacked*. 2016. URL: https://www.her.ie/business/netflix-has-been-hacked-heres-how-to-check-if-your-account-is-affected-268027 (visited on 10/10/2019).

[100] React. *React*. 2019. URL: https://reactjs.org/ (visited on 06/11/2019).

[101] Angular. *Angular*. 2019. URL: https://angular.io/ (visited on 06/11/2019).

[102] Vue. *Vue*. 2019. URL: https://vuejs.org/ (visited on 06/11/2019).

[103] freegroup. *Draw2D*. 2019. URL: http://www.draw2d.org/draw2d/ (visited on 14/01/2019).

[104] mxGraph. *mxGraph*. 2019. URL: https://github.com/jgraph/mxgraph (visited on 14/01/2019).

[105] goJS. *goJS*. 2019. URL: https://gojs.net/latest/index.html (visited on 14/01/2019).

[106] Project Storm. *React Diagrams*. 2019. URL: https://github.com/projectstorm/react-diagrams (visited on 14/01/2019).

[107] jointJS. *jointJS*. 2019. URL: https://www.jointjs.com/ (visited on 14/01/2019).

[108] *Spring*. 2019. URL: https://spring.io/ (visited on 26/06/2019).

[109] MongoDB. *MongDB Atlas*. 2019. URL: https://www.mongodb.com/cloud/atlas (visited on 06/11/2019).

[110] Jing Xie, Heather Richter Lipford and Bill Chu. 'Why do programmers make security errors?' In: *2011 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE. 2011, pp. 161–164.

[111] Signal Sciences. *DevSecOps Community Survey: 2019 Results*. 2019. URL: https://info.signalsciences.com/devsecops-community-survey-2019 (visited on 05/11/2019).

[112] Git-scm. *Git*. 2019. URL: https://git-scm.com/ (visited on 05/11/2019).

[113] Gradle. *Gradle*. 2019. URL: https://gradle.org/ (visited on 05/11/2019).

[114] Nagios. *Nagios*. 2019. URL: https://www.nagios.org/ (visited on 05/11/2019).

[115] Marko Bohanec, Martin Žnidaršič, Vladislav Rajkovič, Ivan Bratko and Blaž Zupan. 'DEX Methodology: Three Decades of Qualitative Multi-Attribute Modeling'. In: *Informatica 37* (2013), pp. 49–54.

[116] Microsoft. *Threat Modeling*. 2019. URL: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling (visited on 05/11/2019).

[117] Daniel Magin, Rahamatullah Khondoker and Kpatcha Bayarou. 'Security analysis of OpenRadio and SoftRAN with STRIDE framework'. In: *The 24th international conference on computer communications and applications (ICCCN 2015). IEEE, Las Vegas, Nevada, USA (3–6 Aug 2015)*. Vol. 38. 2015.

[118] A Refsdal, G Erdogan, G Aprile, S Poidomani, R Colgiago, A Gonzalez, A Alvarez, S González, CH Arce, P Lombardi et al. *D3. 4 Cyber risk modelling language and guidelines*. Tech. rep. final version. Technical report, 2017.

[119] Cyberwiser. *Cyberwiser*. 2019. URL: https://www.cyberwiser.eu/ (visited on 05/11/2019).