How to choose an operating system for an embedded system

Author: Justus Ingelhag
Abstract

The work of this thesis focuses on the critical task of choosing an operating system for an embedded system. The study focuses on features such as security, maintainability, and reliability. By conducting a mapping study and interviews, an evaluation method is proposed. This method helps to make an informed choice of an OS by considering aspects such as hardware support, security features and their overhead impact, kernel architecture, and documentation. The findings contribute to reducing costs regarding research time for a suitable OS, lowering the risk of the unfavorable effects of choosing an unsuitable OS, and making informed decisions. This thesis provides helpful insights and guidance for developers and researchers in optimizing OS selection for improved performance and protection in embedded systems.
Acknowledgements

I would like to express my sincere appreciation to my colleague, Jonathan Olsson, for his invaluable collaboration and positive attitude throughout the project. I would like to extend my gratitude to my supervisor, Alberto Giaretta, for his valuable insights, guidance, and unwavering support throughout the entire timeline of this project. His availability and willingness to address my questions have been instrumental in the successful completion of this thesis. Additionally, I would also like to thank Annika Hansson, my supervisor at Advenica. I am grateful for her insights and feedback during this project.
1 Introduction
   1.1 Background ............................................. 1
   1.2 Project ............................................... 2
   1.3 Requirements .......................................... 3
   1.4 Division of labour .................................... 3

2 Background
   2.1 Research approach ..................................... 4
      2.1.1 Design science ................................... 4
      2.1.2 Mapping study .................................... 5
      2.1.3 Semi-structured interviews ....................... 8
   2.2 Evaluation method ..................................... 10
      2.2.1 Threat modeling .................................. 10
      2.2.2 SWOT analysis ................................... 10
   2.3 Related work ........................................... 11
      2.3.1 Security evaluation regarding operating systems 11
      2.3.2 Evaluation of operating systems .................. 14
      2.3.3 Security features regarding operating systems .... 15
      2.3.4 Kernel architecture ............................... 16
      2.3.5 Assurance & Verification of operating systems ..... 17

3 Method
   3.1 Design science ......................................... 19
   3.2 Systematic mapping study ............................... 19
      3.2.1 Databases ......................................... 20
      3.2.2 Keywords & Search strings ......................... 22
      3.2.3 Filtering first instance ............................ 22
      3.2.4 Brief exploration of articles ...................... 22
      3.2.5 Full-text covering .................................. 23
      3.2.6 Snowballing ........................................ 23
      3.2.7 Set of papers ....................................... 24
      3.2.8 Limitations ......................................... 24
   3.3 Data collection ........................................... 24
3.3.1 Before the interviews .................................. 25
3.3.2 Interview session ..................................... 25
3.3.3 After the interviews .................................. 25
3.4 Developing the method .................................. 26

4 Implementation
4.1 Mapping study ........................................... 27
  4.1.1 Keywords & Search strings .......................... 27
  4.1.2 Searching in databases ............................... 29
  4.1.3 First instance ....................................... 31
  4.1.4 Full-text reading .................................... 31
  4.1.5 Compilation of findings ............................. 32
4.2 Semi-constructed interviews ............................ 32
  4.2.1 Before the interviews ............................... 32
  4.2.2 Interview sessions .................................. 35
  4.2.3 After the interviews ................................. 35

5 Results
5.1 Mapping study .......................................... 37
  5.1.1 Operating systems for embedded systems ........... 39
  5.1.2 Security features & aspects ........................ 40
  5.1.3 Performance, Maintainability & Reliability ........ 41
  5.1.4 Performance vs. Security ........................... 42
5.2 Interviews ................................................ 43
  5.2.1 Question 1 .......................................... 43
  5.2.2 Question 2 .......................................... 44
  5.2.3 Question 3 .......................................... 45
  5.2.4 Question 4 .......................................... 46
  5.2.5 Question 5 .......................................... 47
  5.2.6 Question 6 .......................................... 47
  5.2.7 Question 7 .......................................... 48
  5.2.8 Question 8 .......................................... 49
  5.2.9 Question 9 .......................................... 50
  5.2.10 Question 10 ......................................... 51
  5.2.11 Question 11 ......................................... 51
  5.2.12 Question 12 ......................................... 52
  5.2.13 Question 13 ......................................... 52
  5.2.14 Question 14 ......................................... 53
5.3 Answers to the research questions ...................... 53
  5.3.1 How could the security of an operating system for embedded systems be evaluated? ....... 53
5.3.2 What characteristics of an operating system for embedded systems could be considered for evaluating the quality of an operating system? 56

5.3.3 Can it be determined if an operating system fits the specification of a project well? If so, how? 57

5.3.4 Can one compare two or more operating systems with the information from RQ1, RQ2 & RQ3? If so, how? 58

5.3.5 How can one design a general method that considers RQ1-4 to select an operating system that can fulfill the specified requirements of a given project? 58

6 Discussion
6.1 Compliance with the project requirements 65
6.2 Social and economic implications 66
6.3 Project development 66

7 Reflection
Chapter 1

Introduction

This chapter aims to explain the difficulties of choosing an operating system for an embedded system, why a solution for the issue is needed, and how this thesis aims to solve it. It will start by explaining the current situation and how that affects the industry. After this, a solution on how this can be solved will be presented, why it can reduce the problem in context, and how it will be realized.

1.1 Background

The project aims to develop an evaluation method for operating systems in embedded systems, to guide and aid practitioners and enthusiasts in the field in choosing the most suited operating system for their specific project. This task is done together with a company called Advenica, which is a company that provides cybersecurity solutions and products[6].

Choosing an operating system for an embedded system can be a crucial task since it is the foundation regarding the security[30] and resource manager of the system. Depending on the purpose of the system, aspects like performance, security, and maintainability must be taken into consideration, and are directly affected by which operating system is chosen. A typical property of embedded systems is that they have reduced computing power compared to a regular computer, meaning that a general-purpose operating system (GPOS), like Windows or Ubuntu, might not be an optimal choice to put on such a system due to the computing power constraint or even feasible. It is also common with reduced resources like storage and RAM in embedded systems, meaning that the size of a GPOS with all the features they bring might not fit into the system. This makes it a time-consuming task with all the factors that play a significant role and must be considered when selecting an operating system. According to Advenica, who has a lot of experience
creating products from the ground up, this process is often repeated at the beginning of a new project. It can be very costly, both in time and money. This process often gets avoided, and Linux often becomes the operating system of choice. The advantage is that Linux is packed with features, fits in many systems, and since it is well-known, this choice saves time and money during the beginning phase of a project. However, because Linux is a monolithic kernel with a substantial amount of common vulnerabilities and exposures (CVEs), the choice might not be optimal since this can increase development costs and cause security vulnerability exploits later in the project. The Linux kernel is also significant in size and, due to that, can be harder to maintain. This signals a need for guidance at the beginning of a project to aid practitioners in choosing a safe and reliable OS that is done in an intelligent, cost-effective, and time-saving way. Our aim with this method is to assist people in making wise choices for their projects and try to lower the risk of failure by correctly choosing a suitable OS.

1.2 Project

The goal is to develop a method that can be used at the beginning of a project as an aid to bring forth essential aspects to look at to find a fitting operating system for embedded systems. Due to the resource restraint commonly present for embedded systems which limits OS choices, the limitation of only looking at embedded systems is interesting and reasonable to make and, by looking at current research about operating systems for embedded systems, is needed. This thesis expands its scope beyond the kernel architecture as the sole essential feature of an operating system. By considering a broader range of features, the aim is to enhance the understanding of the subject and explore its significance. The main focus of this thesis is centered around the broad collection and analysis of data relevant to the evaluation of operating systems for embedded systems. Extensive exploration and investigation have been conducted throughout the thesis to demonstrate the various aspects involved in assessing operating systems. In the end, an evaluation method build upon the information gathered is presented.

First, we want to conduct a systematic mapping study to look at current research about operating systems for embedded systems. We aim to determine what aspects researchers have found essential when deciding an OS’s quality and security. We also want to look at what functionality an OS can offer for these aspects and highlight those that are most significant for each element. With this, we want to minimize research time and overall project cost for people by headlining the essential aspects that the researchers suggest. Besides the knowledge gathered from a systematic mapping study, we want to benefit from the expertise of people at Adventica to further explore the field of operating systems in the form of interviews.
Here we get to ask them about their experience of choosing an OS, what they suggest is essential to look at, and things that might imply a completely different thing in practice. We plan to build the method upon these two different sources of information. With both knowledge from academia and Advenica’s experience, we aim to find the critical aspects to guide people to make a more informative, safer, and reliable choice of an operating system for their project.

1.3 Requirements

To be able to make use of this method and to utilize it to compare operating systems, a set of research questions will help us to fulfill these goals:

- How could the security of an operating system for embedded systems be evaluated?

- What characteristics of an operating system for embedded systems could be considered for evaluating the quality of an operating system?

- Can it be determined if an operating system fits the specification of a project well? If so, how?

- Can one compare two or more operating systems with the information from RQ1, RQ2 & RQ3? If so, how?

- How can one design a general method that considers RQ1-4 to select an operating system that can fulfill the specified requirements of a given project?

1.4 Division of labour

Two students from different universities participated in this project. Since both universities have different criteria for the thesis work, we will write our separate reports to ensure that the individual parts of each criterion are met and that it is clear who has done what. Both of our supervisors also suggested this. The segments of the project that must be planned and performed together, along with the parts that we are doing separately, will be clearly stated in this paper. The findings that both of us will discover will contribute to the method.
Chapter 2

Background

This is the background chapter, which aims to give the project’s theoretical and technical background. The methods used during the project are explained, and summaries of the research found during the mapping study are also found in this chapter.

2.1 Research approach

This section explains how design science works and the benefits this approach can have for the project. It also explains the concept of a structured mapping study and semi-constructed interviews as an approach to design science. These are the tools used for this project.

2.1.1 Design science

Johanneson et al.[19] describe design science as: "The scientific study and creation of artifacts as they are developed and used by people to solve practical problems of general interest." The authors define an artifact as an object, a solution humans make to solve a practical problem. A practical problem is defined as the gap between the current and desired states, looking at it in practice. Practice here means a set of human activities that are accomplished frequently and are seen as significantly connected to each other by people partaking in them. The authors also bring up the difference between design and design science, where a design approach sometimes is only relevant to local practice, whereas design science aims to produce an outcome for global practices, i.e., the research community or a group of local practices. There are three conditions that the authors highlight that should be met when making use of the design science method:
• A strategy for collecting data and knowledge from science and field practitioners using rigorous methods should be chosen.

• The outcome from the collection of knowledge should be connected to an already existing knowledge base. This is to make sure that the produced result is original and justified.

• Present the result to people in the field.

We see this as a suitable method for our project for what we want to accomplish. We want to create an artifact, a method, that practitioners can use to solve practical problems. It aims to produce an outcome of global practices. Design science can take many forms, from creating an artifact from scratch to improving an existing solution. To classify the different kinds of design science contributions, the authors look at two factors, Solution maturity, and application domain maturity. The first factor is the maturity of artifacts that could be utilized as an opening for finding answers. The second factor is the maturity of the approach for which the contribution is intended. The authors list the four different types of contributions:

• **Improvement** is something that addresses known problems and offers a new answer or a significant enhancement to an existing one.

• **Invention** is a radical innovation that can be compared to items like the first car or the first X-ray. It allows new conventions and a new foundation for new research areas.

• **Exaptation** is an existing artifact that is repurposed, or exapted, to an unknown problem context. An example of that could be gunpowder, initially used as a medical elixir but then repurposed as powering fireworks and firearms.

• **Routine Design** focuses on incremental innovation to already existing, well-known problems by making small transformations to an existing solution. Typically, this does not count as a design science contribution.

In our case, our contribution is aimed to be an improvement since it addresses a known problem and offers an answer by utilizing existing methods and practices.

2.1.2 Mapping study

Petersen et al.[26] explain that systematic mapping studies or scoping studies aim to provide a comprehensive overview of a particular research area. This is
done by categorizing and quantifying the contributions found within that area. An extensive literature search is conducted to identify the topics that have been addressed in the existing research and the publications where this literature has been disseminated. Farias et al. [13] describe a systematic mapping study (SMS) as the aspiration to underpin the latest about a particular object and aims to gather information about the literature in focus. SMS aspires to offer an overview of the literature in a growing research area and tends to be broader than other research methods since it focuses on several aspects. Farias et al. also discuss another type of study used to underpin a topic’s current status: the Systematic Literature Review (SLR). While the SMS aims to seek to present an overview of the current state, the SLR aims to bring an in-depth analysis of the studied subject by exploring all available empirical studies related to the topic. When a researcher seeks to go in-depth for each research question they have, the authors suggest SLR as a way to explore further. According to Petersen et al., systematic reviews aim to combine evidence and assess its strength, whereas systematic maps focus on organizing a particular research field. The SLR is more stringent than the SMS, and the quality assessment is more essential. Due to this comparison between an SMS and an SLR, we believe it is a better fit to conduct an SMS for this project. We want to get an overview of the scope of the research area to discover research gaps and trends rather than to do a detailed analysis of current research, which is how Petersen et al. describes a mapping study to be. We will take the research paper written by Farias et al. as inspiration for this project, and we aim to follow how they conduct their systematic mapping study.

The following steps are the main steps that we aim to use in the mapping study for this project:

- Objective and Research Questions
- Search strategy
- Criteria for inclusion & exclusion
- Process of extracting relevant articles
- Extraction of data

Each step will be carefully explained in each section below, and examples of how Farias et al. conducted theirs will be presented.

**Objective and Research Questions**

The main goals of their mapping study are to categorize primary research on measuring cognitive load in developers and to identify potential areas for further
research. To reach these goals, they set up a set of research questions to be answered. This is desirable to achieve the study’s research objective and highlight what to look for when conducting the study. This can be viewed as the central purpose of the whole study, to find an answer to these research questions.

**Search strategy**

After specifying the research questions, the authors proceed to present the search strategy for their study. The focus here lies on composing the search strings to ensure that relevant information is searched for and presented when conducting a search in a database, that synonyms for keywords are included, and how Boolean such as "AND" or "OR" is used when doing a search. The authors also mention which databases they use to conduct the investigations that can potentially give relevant information. They list two reasons for what they look for when choosing which to use. The first one is the relevance to the field of study, in this case, software engineering. The second reason is the effectiveness of retrieving peer-reviewed articles for SMSs.

**Criteria for inclusion & exclusion**

To decide which papers are relevant or not, the authors use specific criteria to look for in each paper. In their paper, they list that the paper must be in English and that the paper must be published in a book, journal, conference, or workshop, to name a few. Papers that are duplicates or irrelevant other than keywords found in the paper are some criteria used for exclusion.

**Process of extracting relevant articles**

The central part of an SMS is when one is conducting the searches, searching for information that can answer the research questions, using keywords related to these questions, in databases that can give present articles, and using criteria to include and exclude relevant information. The authors go through these steps by first doing an initial search using the created search strings. They include and exclude specific papers by looking at their set criteria. The authors proceed to look through the titles and abstracts to decide whether the papers are relevant. The authors then proceed to read through the full text and remove articles that are irrelevant or duplicates. As the last step, the authors use a technique called snowballing to increase the coverage of the selection process. This can be done in two directions, one backward and one forward. Backward looks at references in the article or paper currently analyzed that could be of interest, and forward looks if the paper or article is referenced in another source of information that could be relevant.
**Extraction of data**

The last part is finding the data that can answer the research questions from the beginning. The authors use a table to map the data to each research question in this paper. During all these different stages, the authors also keep track of how many articles they find, how many of each of those are excluded and for what reason, and which article maps to what data. All of this is presented in tables and figures, clearly showing the result of the findings.

**2.1.3 Semi-structured interviews**

This form of data collection technique is, according to Runeson et al. [28], of first-degree data, meaning collecting data in real-time in the form of interviews. It is one of the most used and necessary forms of data collection since one can gather knowledge from people’s minds in the study case, which is unavailable elsewhere. Even if a researcher were to study information available in the form of documents, he or she wants to ensure the quality of the information. If the information is of interest to people working in the field, or if the person being interviewed agrees with details within the document. The questions in the interview should be based on the research questions formulated at the beginning of the study. However, they should not be formulated similarly since they could be too difficult for the interviewee to answer. A question could be either open, which can bring a discussion and a more comprehensive answer from the interviewee, or closed, which is limited to only a set of answers that the interviewee can pick. The interviews can be structured in three ways: unstructured, semistructured, and fully structured. During an unstructured interview, the interviewer utilizes an interview guide outlining the main focus areas. The questions asked are open-ended and based on the researcher’s general concerns and interests. During a fully structured interview, every question is thoroughly planned beforehand and asked in the same order as outlined in the plan. The typical choice for studies in software engineering is the semistructured interviews, which are set up in a way where open and closed questions are mixed. The main focus is how individuals qualitatively and quantitatively experience the subject being studied, and the main objective is that it should be explicatory and clarifying. Semistructured interviews allow for asking the questions in a way that feels natural during the interview, and therefore the order of them can differ from each interview.

**Interview session**

An interview can be divided up into four phases. The first phase usually involves explaining the purpose of the case study and how the collected data are intended
to be used. One can ask permission to record the interview to ensure nothing is forgotten during data collection and because it can be hard to grasp what is essential or not when conducting the interview. After this, some preliminary questions about the interviewee’s background or the project, etc. This will hopefully make both the interviewer and the interviewees comfortable and allow them to continue with further questions. The part that will take up most of the interview is asking the questions. Beforehand, it is essential to review the interview format so that the interviewer ensures the interviewee’s confidentiality and integrity. It is also necessary that the questions given to the interviewee are formulated so that the interviewee can answer them without the risk of giving up any information about the company they work for that could damage it. All findings from the interview should be summarized at the end and confirmed by the interviewee so that data has been correctly understood and to see if there have been some misunderstandings. This is the last phase of the interview. Runeson et al. bring up some important aspects to consider when conducting interviews. An advantage could be conducting the interview in pairs if possible. One could have the role of an interviewer that focuses on what is being said, and the other person can focus on which questions have been answered and which are the remaining to ask. Another thing the authors lift as a typical error often standard with inexperienced researchers is that they might ask leading questions. The problem with this is that it can generate inaccurate answers. Inexperienced researchers might also be tempted to "help" an interviewee when unsure or thinking quietly about what to answer. One wants to avoid biasing the answer. The authors also suggest that a consent agreement can be used to clarify the purpose of the data collection, how the data is intended to be used, and how confidentiality will be ensured. This is done before the interview starts.

GQM

Goal Question Metric (GQM) is an approach that helps to find meaningful metrics in a measurement program[20]. The first step in this approach is to find a goal that one wants to be fulfilled, meaning the goal should state what one wants to accomplish with the measurements. To ensure a well-defined goal, the following four dimensions are desired:

- Should have a purpose/reason.
- Entity or entities that will be considered.
- Attribute or attributes that will be considered.
- A viewpoint from which the measure is taken.
From the goal, one wants to formulate a question or questions that help to define the goal as fully as possible[9]. The questions will then be redefined into metrics, which purpose is to answer these questions. One question can generate one or multiple metrics, depending on how it is framed. One wants to find measurable and realistic metrics to answer the questions they are derived from.

2.2 Evaluation method

This part aims to give some background information on tools used in our method, to use for the evaluation of features for operating systems and analysis of data that could be of interest to the project.

2.2.1 Threat modeling

Threat modeling is described by Owasp[7] as a family of activities for enhancing security by identifying dangers or threats and preventing mitigating by finding countermeasures to them. It generally includes some explanation of the matter being modeled, assumptions on what happens if the threat landscape changes in some way, what can be viewed as a threat to the system, and how to mitigate these threats. One also wants to ensure that these actions successfully can reduce those threats. The benefit of doing this type of analysis, if done right, is that it gives a justification for making an effort regarding the security of a project and makes decisions about safety more rational when looking at all available information.

2.2.2 SWOT analysis

According to Granulo et al., [14], SWOT analysis, which is an acronym for strength, weaknesses, opportunities, and threats, is a popular tool for decision-making and is sort of a guide looking at personal opinions and decisions. It can be used to evaluate a proposal for the future, and the goal is to find strengths and opportunities that the proposal has against weaknesses and threats that can come with it. The authors define each keyword like this:

- **Strength**, something that helps to achieve a goal and that can be influenced and measured.
- **Weakness**, something that is not advantageous to the goal, a factor that requires substitute or repair.
- **Opportunity** represents something that favors the goal and is good for the future. It will help to increase the chances of a favorable outcome.
• **Threat**, something that will decrease the odds of something good happening that can be bad for the future and should be avoided.

The authors see SWOT analysis as a beneficial tool to use as a defense mechanism, but it requires clear and intelligent thinking to make a good decision.

### 2.3 Related work

In this section, all summaries from the papers used to answer the research questions for this thesis can be found. Each paper lies under a set of subcategories to help to navigate the papers and for structure. *Security evaluation regarding operating systems, Evaluation of operating systems, Security features regarding operating systems, Kernel architecture and Assurance & Verification of operating systems* are the ones that are used.

#### 2.3.1 Security evaluation regarding operating systems

Mhamed Zineddine[34] says that the increase of digitalization in industrial control systems is encouraging the use of existing security solutions. This comes with risk since they are often not suited to be in those kinds of systems. With the help of a bat algorithm, the author managed to prevent Intrusion Detection and Prevention Systems without decreasing the data transfer quality of service between the systems. The author mentions that these systems must behave as expected, even if security features are implemented. He also mentions that improving security while maintaining the quality of service is paradoxical, meaning that introducing security will increase computing power, leading to performance overhead. Therefore, it is essential to consider and respect the worst-case execution time. The result shows while avoiding unwanted effects, the IDPS selects the best security policies effectively for all active roles and diminishes the worst execution time.

Profentaz et al.[27] center their paper on the absence of secure boot in embedded systems and emphasize the consequences that arise due to this issue. The secure boot has been one of the standard techniques to ensure each booting step’s integrity, ensuring that the kernel code has not been modified. Commonly, IoT devices often control physical objects, such as a car’s engine or a refrigerator’s cooling system. In the event of an attack and potential breach of these systems, an unauthorized user may gain access to sensitive data or manipulate executable system files, resulting in significant damage. Embedded systems often neglect to implement secure boot, leaving the boot process and internal software vulnerable to manipulation. The overhead of securing the boot process can affect these systems negatively regarding memory capacity, energy consumption, and boot time.
The authors aim to determine the impact of implementing secure boot through software and hardware on an embedded system. It was discovered that the introduction of secure boot software caused the boot-up time to increase by 4%, whereas a hardware implementation resulted in a 36% increase in time.

Hamdani et al.[16] analyze in their literature review different types of cybersecurity standards, what they consist of, how they are designed, and compare them. They discuss the importance of security since the dependency on computer systems is growing. Since all of the standards are general guidelines to achieve security in computer systems and not specific approaches to hardening an operating system, it is a tedious task for the end-user to ensure they are protected against cybersecurity threats. The authors also list a lot of alarming statistics about organizations that signal low prioritizing of security and call for a need to keep heads above water regarding cybersecurity. The result that the authors bring to the table from this paper is a bare-minimum set of necessities that is a must-have for ensuring protection regarding cybersecurity, focusing on the Windows OS. One significant challenge the authors bring up is that even if an organization follows one of these security standards, it might not be sufficient to fulfill its requirements. They suggest that further work needs to be done to merge and intersect multiple standards to accommodate the requirements regarding security for the organization.

Song et al.[30] discuss the importance of host security when looking at information security, which is the first layer of security in an information system. Host security relies on the operating system, which is the foundation of the security inside an information system and is the main focus throughout the article. The authors mention that the OS security methods consist of hardware and software techniques. The authors’ software mechanisms are user identification, access control, most minor privilege management, trusted path, covert channel, security audit, and virus defenses. With these aspects, the main objectives of the operating system security are to make sure that the user’s actions should be restrained according to the security policies, the users in the system should be identified, the system should be supervised, and the integrity of the system should be assured. The author’s problem regarding evaluation tools for host security is that they are not comprehensive enough and can only handle some vulnerabilities. However, the authors state that it is impossible for a system to be entirely secure, and the aim is to provide security for found susceptibilities. The authors suggest a broader approach that does not only look at the evaluation of the OS and lists different evaluation aspects that are more comprehensive, such as malicious code defense, invasion defense, and residual information protection.
Abbasi et al. [8] bring up exploitations on embedded systems as a problem regarding memory corruption and what techniques are used to prevent them, where the focus is on executable space protection, address space layout randomization, and stack canaries. The reason for this is that the current picture of how big the gap is regarding security mechanisms for embedded systems is unclear as to how they are compared to solutions for general-purpose computers. Forty-two different OSs were compared and evaluated to get an overview of embedded OS mitigation adoption. The authors saw a big gap for embedded OSs compared to GPOSs like Linux or Windows. The authors identified four groups of open problems the embedded OSs share: "Development Practices & Cost sensitivity," "Resource constraints," "Safety, reliability & Real-time requirements" and "Hardware & OS Limitations." The authors’ first open problem states that the world of embedded OSs is fragmented between vendors, leading to issues like patching issues, lack of general capabilities for security due to specific-driven purposes an embedded system usually has, and cost. The second problem concerns the smaller size of memory and storage and less computing power embedded systems typically have over a GPOS. This inhibits the inclusion of specific cryptographic algorithms and computationally heavy security measures. The third issue focuses on the fact that security measures must be robustly trustworthy and available while also stopping or mitigating any possible attack on the system. This balancing is tricky since the measure must be robust but not too aggressive so that they affect critical parts of the system negatively. The last problem that the authors bring up is the typical lack of modern security measures, such as advanced processing features like trusted computing or isolation of code or dedicated hardware chips like MPUs, MMUs, and ESPs. This is connected to the cost and fragmentation problems mentioned in the first open problem. To improve against these issues, the authors present an exploit mitigation baseline design called microArmor that helps improve security with no sizeable overhead impact.

Bekele et al. [10] examine hardware faults in the microkernel environment to see how the choice of architecture affects the system’s reliability. The authors state that reliability as a concept is closely associated with security in a system, and microkernels are one technique to ensure that faults do not propagate to critical components. However, these faults cannot be entirely eliminated. Furthermore, there is an authentication mechanism between elements in the system to work against non-secure communications to strengthen the system’s reliability. What the authors look for in this paper is how mechanisms in the microkernel architecture can resist hardware faults. They compare the seL4 microkernel against the Linux kernel, a monolithic architecture. The authors find that the microkernel reduces the number of silent data corruptions (SDC:s) and crashes compared to what
Linux managed to do. However, the number of hangs in the systems is higher. The authors believe this has to do with the performance constraint in the microkernel, which affects when a hang is reported. The authors mention that it is more advantageous with fewer SDCs than hangs since they can be detected and dealt with.

2.3.2 Evaluation of operating systems

Patel et al. [25] list several operating systems in their case study that is commonly used in Internet of Things (IoTs) devices and analyze what each OS provides in terms of kernel architecture, scheduling policies, network protocol support, memory allocation/footprint, and management, program language support, real-time support, etc. Due to the limited resources of IoTs regarding memory, computing power, and power management, general-purpose operating systems (GPOS) are not suited to be inside these systems. The authors state that the first aspect to consider when choosing an operating system for an IoT device is the kernel architecture since it significantly impacts the OS’s modularity. Other aspects, such as scheduling policy and network connectivity/protocol support, are also essential since they substantially influence the device’s energy consumption. For a device that needs real-time support, the authors mention that it is essential for the kernel to operate with deterministic run-time. The authors conclude that security and connectivity are top concerns regarding IoT properties, looking from a developer’s perspective.

Zikria et al. [33] examines the rapid growth of IoT devices, their impact on society, and what challenges they bring when trying to connect all of them to the internet, as we know. They can be usually found in use-cases like industrial automation and HVAC (heating, ventilation, and air conditioning) systems but also in more crucial infrastructure like mobile health and advanced metering infrastructure, like power grids. Since these systems are constrained by their size, computing power, energy, and memory capacity, they must fulfill demands such as real-time capabilities, energy efficiency, small memory footprint, interoperability, network connectivity, and security/safety. This makes the operating system a pivotal point in achieving and meeting these requirements. The OS kernel must operate in a deterministic way to achieve real-time assurance. The OS must provide cryptographic libraries and security protocols to meet security standards while preserving resilience and versatility.
2.3.3 Security features regarding operating systems

Mullen et al. [24] investigates the widespread problem with buffer overflows (BOF) and their impact on embedded devices. The primary cause of these exploits lies in the C language, typically used in these systems due to resource constraints and its absence of boundary checking in functions for processing data buffers. The authors compare Ubuntu against FreeRTOS regarding two types of attacks, return-to-lib-c and code injection, to see how sensible FreeRTOS is against these buffer overflow attacks. The authors found that in Ubuntu, there were two different prevention mechanisms, address space layout randomization (ASLR) and stack protector or stack guard, that the OS provided to prevent a buffer overflow attack. FreeRTOS, on the other hand, had no protection against BOF attacks by default. The authors mention that one cannot assume that operating systems for IoT provide the same functionality that GPOSs do and that one could use the methods used by Ubuntu and port it to FreeRTOS to make it protected against BOF attacks. The authors also push that security is one of the main problems with IoT; without it, it will likely fail.

Malenko et al. [22] discuss the issue of the complexity, size, and inconsistency that can come with device drivers and applications developed by different parties that occur inside embedded systems. Without proper memory isolation in the address space, these with OS services and libraries represent a big attack surface. Unlike commodity operating systems that usually use virtual addresses and Memory management unit-based isolation, low-end systems use some form of a memory protection unit (MPU) that protects address regions in a single address space. The authors highlight device drivers as one of the most vulnerable parts of an OS since third parties often provide them, contain more bugs than other code, and endanger OS reliability. Using fault isolation for these device drivers, the authors state that it can prevent global system failure and misbehavior. The authors propose a hardware-enforced approach to provide run-time protection by isolation the memory of device drivers.

Moghadam et al. [23] lifts buffer overflow as one of the most common memory safety vulnerabilities in the embedded world due to using C and C++, which can raise security exposures. When protection against attacks called arbitrary code execution was established, like stack canaries, a new paradigm of attacks appeared, called Code-reuse attacks. This type of attack hijacks the control flow of the program executing. This paper introduces the concept of Control-flow integrity. This technique is suitable against these attacks since it ensures that the program follows the control-flow graph, which is found when statically analyzing the program before execution. One can then check during run-time against this CFG to ensure
that the program execution is following this flow. The authors see that implementing this mechanism is poor for embedded and real-time operating systems. When they compare different solutions, they see a trade-off between security and real-time capabilities that either affects the performance or protection of these systems. According to the authors, a full-coverage solution with no overhead for embedded systems is still an open issue.

2.3.4 Kernel architecture

Liu et al. [21] compare the microkernel against the monolithic kernel in their paper, looking at how it is built and functions and why the microkernel is preferable in embedded systems. They mention the architecture as a critical argument for the microkernel to be more desirable. The main difference between the architectures is that the monolithic approach runs every service in kernel mode. In contrast, the microkernel separates crucial tasks and other services into kernel mode and user mode. This makes the monolithic kernel non-secure since all services operate at the highest authorization level. A crash of one service can affect other services running in kernel mode and, therefore, the system as a whole. According to the authors, separating the services makes the microkernel more secure, reliable, and maintainable. The authors also point out that embedded systems often have specifications such as real-time performance and limited resource capabilities that the microkernel provides with its implementation. As a drawback, the authors mention the poor performance that the microkernel usually has over the monolithic kernel due to the address space and context switching occurring between the modules for the microkernel.

Silva et al. [29] investigate operating systems for the Internet of Things (IoT) devices and how they compare against each other through benchmarks and analysis. It is commonly believed that no single operating system can cater to all needs and requirements. To explore this further, the authors analyze the most widely used open-source operating systems for IoT. They found two key features significantly impacting the system’s overall behavior: kernel architecture and scheduling policies. These factors affect the system's performance, determinism, and power consumption. What also affects the power consumption is the system’s memory and how the OS uses that memory. Using memory more dynamically can help decrease power consumption compared to an OS that uses static resources. The authors also highlight the importance of choosing a suitable OS for an application with real-time demands and state that it is crucial to meet that requirement. The authors utilized the Thread-Metric Benchmark Suite to evaluate performance, focusing on prevalent real-time operating system (RTOS) services and interrupt
processing mechanisms. The conclusion from their investigation is that different OSs fit into different applications, depending on the purpose of the system.

Hamad et al.,[15] lift the microkernel as a way to improve security in an embedded system due to its small kernel size with only the necessary parts running in privileged mode, different access control privileges for each application and the capability to involve system policies to manage application communications. However, it can not eliminate all safety concerns. Researchers have proved the network system inside embedded systems to be insecure in vehicles like aircraft and cars. The authors of this paper suggest a solution called IPsec, which uses two encapsulation protocols; Authentication Header (AH) and Encapsulation Secure Payload (ESP). They wanted to investigate the overhead impact of using IPsec inside an embedded microkernel system. After conducting their analysis, it was determined that the primary security concerns in communication within distributed embedded systems stem from inadequate authentication methods. This leads to unauthorized people being able to send false data or even take over the system entirely. They suggest using only AH since they can fix these issues with acceptable overhead. Using IPsec in a constrained system causes significant overhead and affects the system’s throughput. The authors believe that switching to a more efficient encryption algorithm can help improve the performance of ESP.

2.3.5 Assurance & Verification of operating systems

Chen et al.,[11] discuss the importance of semaphores in the embedded operating system since it is an essential key component that regulates the rules when multiple tasks access a shared resource between them. The authors explain that traditionally when testing out software, one only looks at whether the outcome of the execution is correct. But to ensure the correctness of the software, formal methods are used, which are based on a strict mathematical foundation. The authors state this is the only way to ensure the system is free from programming errors. This paper proposes a software-proof framework based on formal methods to test the functionality and security of a typically embedded operating system semaphore module. The results show that the module in context met the requirements given to the verification framework. The authors, however, highlight the lack of verification of the requirement document that is given, and future work will focus on how one can detect redundancy or conflicts in such a document.

Vanderleest,[31] discusses the importance of safety in avionics and how the industry focuses on reliability, lowering the number of faults that occur and minimizing their effect if they do transpire. Lately, the industry has more clearly
recognized that security issues can have a significant impact, and the authors bring up formal verification as a way to achieve the highest level of safety. This paper focuses on the formally proven microkernel SeL4, the first operating system kernel, with an end-to-end assurance of implementation correctness and security enforcement, to have this verification. The author mentions that the OS holds the highest level of authority in the system. The microkernel approach is utilized to minimize its feature set, resulting in a smaller kernel that reduces the cost of ensuring safety and security. According to the authors, the proof of correctness has limitations based on a set of assumptions. One of the assumptions is that no Direct Memory Access (DMA) devices are present. However, this is a significant omission since most modern embedded systems use DMA to enhance performance.
Chapter 3

Method

This chapter aims to explain all the tools and methods used in this project. Each approach will be motivated by why it was chosen and compared to other alternatives that have been discussed.

3.1 Design science

For the research method, we choose to use the design science approach. The goal is to develop a method, some artifact, that can be used by practitioners in the field of software engineering. The method’s purpose is to reduce the gap between the current state of choosing an operating system and the desired state of how to select one in a more informative way. As mentioned in the background chapter 2, this approach is suitable for what we want to accomplish and is a reasonable choice. Other research approaches were briefly overviewed, but the choice for design science was taken pretty instantly due to the nature of our project.

3.2 Systematic mapping study

At the start, we are conducting a systematic mapping study. This choice is made after consulting with the other student’s supervisor and doing some research comparing it to a systematic literature review. Due to time constraints and the broader approach that we want to take since we want to look at many different factors, a suitable choice for this project is to do an SMS. As explained in the background chapter 2, this is recommended if the aspiration is to collect state-of-the-art information about a subject, looking at several factors about a topic. This fits nicely into what we want to do and, thus, is a reasonable choice. The mapping study aims to evaluate, analyze and characterize all relevant research on a specific topic, research questions, or aspects of interest[13]. Inspiration has also been taken from
one other study that uses this type of methodology[18]. By examining relevant articles, books, etc., about the topic, we aim to find information about the current state of people in the field. Another desire is that it can help to find gaps in the recent research and that further investigation needs to be established. The goal of the approach is to get the most updated and relevant information about the general aspects of operating system kernels. For each section, I will present a general overview of how the mapping study and the data collection will be conducted. Second, an explanation of each step of the methods is given. Each section will explain how and why a particular approach was chosen.

3.2.1 Databases

I intend to utilize six primary databases for conducting the mapping study:

- IEEE Explore
- SpringerLink
- Science Direct
- ACM Digital Library
- MDPI
- USENIX

All of the databases are known in the field of computer science and are relevant to our study[1][17][2][3][5][4].

When using these different databases, a filter will be applied to narrow the search, and specific keywords will be used to obtain relevant information. This filters out all the matches obtained from the initial investigation. All articles collected will be examined by looking at each article’s title, abstract, and conclusion. A third filter will then be applied to ensure that the pieces are relevant to the conducted work. All articles will be reviewed throughout to get the whole picture of the paper and how it was completed. Backward snowballing will be used at this stage to complement the initial search to further examine relevant work in the field. Lastly, all the articles that have gone through the process will be summarized and used as related work. In figure 3.1, one can see what the workflow of the SMS will look like.

To filter out admissible databases, three different criteria have been utilized:

- The database should be relevant in terms of the area of the scope for the work, that is, computer science.
• The database should provide peer-reviewed articles.
• The database should be well-known as a source for technical articles.

Figure 3.1: Overview of systematic mapping study

The reason for the criteria listed above is to search relevant databases that are prominent in the field of data science and where peer-reviewed articles can be obtained to increase the probability of getting information that can be useful to our work. This will hopefully also increase the credibility of the themes found at the end of the process.
3.2.2 Keywords & Search strings

The choice of relevant search words will primarily be selected concerning their relevance to security and evaluation regarding the operating system and kernels related to embedded systems. All keywords chosen will be presented in a table so that others can recreate the searches.

3.2.3 Filtering first instance

The first filter obtains unique, updated, and understandable information for many people. It contains four different rules on what information that will be presented when searching using the different databases:

1. The paper should not be older than ten years from the initial search.
2. If the same paper is filtered out from the different databases, one will be removed due to duplication.
3. The paper should be written in English.
4. The paper should be available in full text.

What we want to achieve with these filters is that the papers should present the current research about the subject and that a full version of the report should be available to understand the study entirely. To make our project available to other researchers, we consider including English articles a reasonable requirement. We also need to consider the timeline and the scope of the project. The result from this stage will be presented in a figure showing how many papers were found from each database.

3.2.4 Brief exploration of articles

For this part, we will read through all the obtained papers individually. We look at the title, abstract, and conclusion for each paper. Doing this allows us to sort out relevant papers effectively and read through each paper more thoroughly later. The following six criteria are what we will use to determine whether to include a paper or not:

1. If the choice of kernel or operating system entailed problems and if so, do the paper’s authors discuss it?
2. Security vulnerabilities around the kernel or operating system have been discussed.
3. Kernel is relevant to embedded systems.

4. The process of choosing a kernel or operating system has been specified.

5. The paper should be about micro-kernels or monolithic kernels.

6. Have the authors discussed and proposed a solution to the issue?

These criteria decide the relevance of the paper and if the paper seems relevant to our study, which appears to bring up at least one of the six criteria listed above, I include it. If the form is somewhat pertinent to our research, meaning it’s hard to judge at first glance, a second review process will be conducted more carefully to determine its relevance. The paper is irrelevant and excluded if none of the criteria above is fulfilled. We will do this to all articles to ensure compatible information is found and included.

3.2.5 Full-text covering

After all important papers have been sorted out, we will go through each paper and read them from top to bottom. The above criteria will be used and compared more carefully at this level than in the previous stage. We will also add three more criteria:

1. Does the paper help us to answer the research questions?

2. Is the choice of research method relevant to our study?

3. Are there any flaws that the authors bring up?

The hope is to make sure that only papers related to our study are included and ensure a selection of knowledge with quality. Each piece found from this stage will be presented in a table, showing which keyword was used to find this paper, from what database, and what criterion it fulfilled.

3.2.6 Snowballing

To get a broader understanding of the papers we will go through, snowballing is used at this stage, precisely backward snowballing[32]. For each paper found, we will go through referenced articles to further explore the paper’s main idea in focus. These articles will be chosen based on relevance to our study.
3.2.7 Set of papers

Finally, each paper found will be summarized into approximately 200 words and put into the related work section. The summary aims to answer these questions and give concentrated information on what the paper is about:

1. Does the summary clearly describe the context of the study?
2. Does the summary clearly describe the study objectives?
3. Does the summary clearly describe the study findings?
4. Does the summary clearly describe the limitations of the study?

3.2.8 Limitations

All of the choices that we have made for this SMS come with risks and limitations, both due to the approach but also because of limited time. Having criteria that decide what is relevant or not makes it a limitation. Information that could be of interest might be excluded due to a lack of fulfillment of the criteria that we have decided upon. Since it is up to us to determine what counts as relevant or not, our knowledge and understanding of the subject can be a limitation and therefore result in the exclusion of pertinent information. Information that could be of interest might not be discovered due to the lack of relevant keywords, which we also see as a risk for this study. A similar problem can be found when we look at the databases chosen. Other databases might include papers with relevant information that will be missed out.

3.3 Data collection

After conducting a structured mapping study, we will bring the information obtained into the data collection part of this study. Our ambition with the mapping study will give more insight into the topic in context. It will hopefully develop a greater recognition of what type of data one wants and can collect. With the knowledge gained from the systematic mapping study, we aim to formulate questions relevant to the topic investigated and for people in the field working at Advenica. This will be in semistructured interviews, a common choice for case studies in software engineering, which was mentioned and discussed in the background chapter 2. This is done with a GQM method to ensure the questions are delimited to discuss admissible topics. Our ambition is to formulate goals that align with the study’s objectives. This translates into what we want to attain when conducting the interviews with Advenica. We wish to acquire questions that
characterize the goal in a way to be measurable and meaningful. This helps us to ask relevant questions to Advenica. The interviewees will be chosen with the aid of our supervisor at Advenica. After the interviews, we will analyze and discuss the results to use this in our method development.

3.3.1 Before the interviews

Before contacting the interviewees, we will construct an agreement that the subject will read through and make sure they understand the purpose of the data collection and how we intend to use it. We will ask the participant if they are comfortable with us recording the interview and let them decide if that is okay or not. We will also ensure they can withdraw their participation at any time. At last, we want to inform the participant that after the interview, we will send them a summary of what we wrote down. This is because they can ensure we understood them and correctly wrote down their answer.

We will also discuss the questions with a security officer at Advenica to make sure that the questions are safe to ask without risking the confidentiality of the participant by asking questions that could pinpoint who is giving a specific answer and that the questions will not jeopardize giving up secrets within the company.

3.3.2 Interview session

During the interview, we will both participate, where one of us will ask questions and listen while the other will take notes. We aim to switch roles for each interview so that we both do the same amount of times at each position. We aim to keep the interview session at about forty-five minutes. The first thirty minutes will be about the questions. Here we will try to ask the questions and listen to the participant’s words without attempting to "help" them. An example could be that they are not sure what to answer. If the question is unclear, we will try to explain what we mean. The last fifteen minutes will be dedicated to showcasing what we have so far regarding our method and getting input from the participants. We will ask if what we present as a way to choose an operating system is reasonable and how it can be improved.

3.3.3 After the interviews

After each interview, we will review and summarize our notes. We will ensure that we agree on the summarized answer for each question. If the participant permitted us to record the interview, we could listen to it to ensure that nothing of importance was forgotten or if some answers were unclear. All the answers to
the closed questions will be summarized and displayed in a table, showing graphs of how all participants answered each of them. For the open questions, I will use the summary mentioned above to give an answer to what the participants answered for each question in the form of text.

### 3.4 Developing the method

During this thesis, we intended to do it agilely, meaning that if insufficient data was found from the mapping study or if the data collected from the interviews were inadequate, we plan to have the opportunity to go back and gather more if so. In figure 3.2, one can observe the general workflow of this thesis.

![Figure 3.2: General overview of this thesis work](image)

Figure 3.2: General overview of this thesis work
Chapter 4

Implementation

This chapter will describe how I executed each step of the mapping study and the interviews. It will also show the results of what was found.

4.1 Mapping study

This part will go through each step of the mapping study, how I carried out each of them, and the results. Motivations about the choices made will also be presented. In figure 4.1, one can observe the statistics and an overview of the mapping study.

4.1.1 Keywords & Search strings

The keywords chosen to use for the search strings were the following:

- Operating system
- Kernel
- Microkernel
- Monolithic kernel
- Embedded system
- IoT

Operating system was a natural choice of a keyword since the whole purpose of this project was to investigate operating systems. Kernel, Microlkernel, and Monolithic kernel were keywords that were also mentioned in the early talks with Advenica and felt essential to include when talking about comparing different operating systems since it refers to the architecture of the OS. These are also the main
Figure 4.1: Result from mapping study

architectures when talking about operating systems. *Embedded system* and *IoT* are words related to systems with restrained computing power and of interest for our mapping study.
All of these keywords were combined into search strings, together with these specifiers:

- Security
- Comparison
- Evaluation

Combined with the specifiers listed above, we will be able to form strings that are relevant to our study. *Comparison* goes along with our goal of looking for ways to compare operating systems, which can be used to form self-explanatory search strings. *Evaluation* is also natural to use in this case since we want to know how to evaluate operating systems and what to look for. *Security* was a word brought up during the talks with Advenica. It was one of the key features we wanted to investigate when evaluating an operating system; thus, it felt natural to include it as a keyword. All of the search strings that we used for all different searches can be found in column three in table 4.1.

### 4.1.2 Searching in databases

After deciding which keywords we wanted to include, we conducted the searches. The search strings were made up by using the keywords and modifiers listed in section 4.1.1. They were created how we subjectively felt natural in a way to search for. Since we aimed to find research about comparing and evaluating kernels and operating systems, we constructed search strings, combining the keywords with different modifiers because of our talks with Advenica and to find different angles when we conducted the searches. In table 4.1, one can observe how the searches were conducted. I focused on inquiries regarding the security of operating systems and kernels for embedded systems, while the other student focused his investigations on the evaluation of operating systems. This choice was made to spread the searches for efficiency. I did five different searches in three different databases, *IEEE*, *ScienceDirect*, and *SpringerLink*. The order of my searches is the same as presented in table 4.1, from top to bottom. The choice of not searching in the other databases mentioned in the method chapter, for my part, was a combination of time constraints, and I found that the number of articles was sufficient. In the early talks with my supervisor at the university, he highlighted the importance of starting to search and read articles as soon as possible since this, in his experience, will take a lot of time. This was also mentioned by researchers when conducting a systematic mapping study. So before heading into this part of the mapping study, I knew that finding and reading through articles would take a lot of time. I first started searching in the IEEE database. Since I got a large
<table>
<thead>
<tr>
<th>Name of database</th>
<th>Number of articles from search</th>
<th>Search strings</th>
<th>Who conducted the search?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summon</td>
<td>948</td>
<td>Operating system security embedded systems</td>
<td>Jonathan</td>
</tr>
<tr>
<td>IEEE</td>
<td>795</td>
<td>operating AND systems AND security AND embedded AND systems</td>
<td>Justus</td>
</tr>
<tr>
<td>Summon</td>
<td>1121</td>
<td>Operating system IOT</td>
<td>Jonathan</td>
</tr>
<tr>
<td>Summon</td>
<td>62</td>
<td>&quot;operating system&quot; AND Evaluation AND Security AND &quot;Embedded systems&quot;</td>
<td>Jonathan</td>
</tr>
<tr>
<td>Summon</td>
<td>1161</td>
<td>operating system AND Evaluation AND Security</td>
<td>Jonathan</td>
</tr>
<tr>
<td>IEEE</td>
<td>169</td>
<td>&quot;operating system&quot; AND security AND kernel AND evaluation</td>
<td>Justus</td>
</tr>
<tr>
<td>Summon</td>
<td>127</td>
<td>operating system AND comparison AND &quot;embedded system&quot;</td>
<td>Jonathan</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>20</td>
<td>&quot;monolithic kernel&quot; AND security</td>
<td>Justus</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>107</td>
<td>&quot;microkernel&quot; AND security</td>
<td>Justus</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>223</td>
<td>&quot;microkernel&quot; AND security</td>
<td>Justus</td>
</tr>
</tbody>
</table>

Table 4.1: Searches in databases
number of hits in the database, 795 hits, it took me quite a long time to read through each article’s title, abstract, and conclusion. By the time I had done that whole process, many potential articles were filtered out that I would proceed to read in full text. This was repeated four times, with fewer hits with my searches on SpringerLink and ScienceDirect, adding more papers to the list of potential articles. Figure 4.1 shows how many documents were found in each database. The criterion set for the filter articles can be found in 3.2.1.

4.1.3 First instance

The number of papers gathered from each of the five searches in total was 1314 papers. After each search I did, I went through all of the articles, looking at the title, abstract, and conclusion, to decide if the paper in some way fulfilled at least one of the criteria listed in the method chapter. I gathered a list of the relevant articles and put the information about each piece in a table. This was to track which paper was found depending on the search string I used and in which database the document was found. For every database I used, there was a functionality that could export articles I found of relevance as an XML file, which was very convenient. In figure 4.1, one can observe how many articles were sorted from each screening. I found most of the articles from IEEE. This was probably due to the search strings I used and the fact that it is an extensive database. In total, 104 papers were included. All of them seem to fulfill either one or more of the criteria listed in 3.2.4.

Most of the articles that were excluded were irrelevant in some way. A substantial amount was not focusing their work on the keywords set from the start and was therefore excluded. A significant amount only mentioned something about operating systems or kernels, but no further information was given. Some focused only on hardware implementation regarding operating systems, which was out-of-scope for this thesis. Only two papers were unavailable in English, and only one was not in full text.

4.1.4 Full-text reading

All of the 104 documents that were gathered from the first instance of filtering were now read from start to finish. For each paper, I read it from top to bottom and highlighted relevant information that could help answer the research questions. The filter used in 3.2.5 has been applied here. Determining if the paper met the criteria was more accurate since it was reviewed in its entirety. Out of 104 documents, 21 of them were included.
4.1.5 Compilation of findings

After summarizing all the articles I found, I and the other student discussed our findings. We compiled all information, listing features and essential aspects of the articles. This was done to map out and compare our findings. From the mapping study, we gathered enough information that we felt we understood the current state of the research about operating systems and felt comfortable to proceed to discuss what type of questions to ask the participants for the interviews. In the end, I used a set of 16 papers that were found from our findings. This was done to keep the amount of information to a reasonable level for this thesis. The 16 articles were handpicked, and chosen by their relevance. These were also closely related to the research questions. The articles were summarized and added to the related work section of this thesis work.

4.2 Semi-constructed interviews

In this section, I describe how we conducted the interviews, listing each approach before, during, and after the interviews. It follows what was previously explained in the method chapter 3.

4.2.1 Before the interviews

After the mapping study, with recommendations from our supervisor from Advenica, we reached out to people at Advenica to interview. These are people with experience working on projects and with embedded systems. Each person was e-mailed with information about the project, the purpose of the data collection, and instructions on conducting the interviews. Eight of the twelve people our supervisor suggested agreed to participate in the interviews.

Questions

From what we discussed in the mapping study, we weighted aspects regarding working experience on projects and what we feel is essential when dealing with issues in practice and came up with 14 questions. It was done somewhat as the GQM approach suggested. We took our research question into consideration and formulated questions that could be used in the interviews and at the same time help us answer the research questions. In table 4.2, one can find the questions from our discussion. Seven of these fourteen questions were closed, and the rest were open. Question number one is about how the choice of OS is usually made, and we wanted to map out the standard way for practitioners to choose an OS. We decided
<table>
<thead>
<tr>
<th>Order</th>
<th>Question</th>
<th>Purpose</th>
<th>Scale type</th>
<th>Raw text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Do you select OS or kernel based on prior experience, evaluate for each project, or gut feeling?</td>
<td>Behaviour</td>
<td>Nominal</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>How have (1) affected the results?</td>
<td>Project Understanding</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Is it common to implement your features in the OS or kernel? Follow-up, if yes, why?</td>
<td>Development understanding</td>
<td>Internal</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>How long time does the selection of OS usually take?</td>
<td>Effort</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>How long time does the selection of OS usually take?</td>
<td>Effort</td>
<td>Internal</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Is there any major impact on selecting an OS or kernel? Follow-up, if yes, what?</td>
<td>Mitigation strategies</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Are there any theoretical aspects that might raise practical uncertainties? Follow-up, if yes, what are those?</td>
<td>Model improvement</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>What QoS characteristics of an OS are considered for selecting the quality of an OS kernel?</td>
<td>Experience, Unknown uncertainties</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>What key aspects could be used for comparing an OS kernel?</td>
<td>Security improvement</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Are there any theoretical aspects that might raise practical uncertainties? Follow-up, if yes, what are those?</td>
<td>Model improvement</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>What are common security concerns?</td>
<td>Security improvement</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>How could the security of kernels in embedded systems be evaluated?</td>
<td>Model improvement</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>What are common security concerns?</td>
<td>Security improvement</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>What key aspects could be used for comparing an OS kernel?</td>
<td>Model improvement</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Anything of value to consider while building the evaluation model?</td>
<td>Model improvement, reduce risk to miss information</td>
<td>Open</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Questions used for the interviews
to have three criteria that we think covered the regular ways to do so. The follow-
up question, number two, wants to investigate how this has affected the result. Do
people think that this has worked well or not, and if so, why? This question gives
us a natural opening to discuss and ask what would be preferable if the approach
negatively impacted the result. This can help us understand if a method is needed
and how it can aid. Question three is to understand the development process
and whether implementing its own features in the OS or kernel is ordinary. This
question comes from the findings from the mapping study, where the majority of
the papers present a problem regarding an OS or kernel and a solution to that
problem that the authors provide. We are interested to know how common it is to
provide your own implementations and, if so, if is it desirable. The fourth and fifth
questions focus on the time span of choosing an OS, both how the reality is and
how long it is desired to take. To create a method that aims to reduce the time
cost of looking into what OS to choose for a project, it is essential to investigate
how long this process takes and how long it is desired to take. The sixth question
investigates the optimal way to go when selecting an OS. Is it to first decide which
hardware to have and then the operating system, or the other way around? Either
way, choosing one of these ways will affect how one can proceed since an OS might
not be supported on specific hardware and vice versa. And if there is a desired
approach, why is that? This helps us understand if the order of decisions in the
method matter. The next question’s purpose, number seven, is to understand
what can go wrong if one picks an insufficient operating system for a project. Can
this affect the project, and if so, in what way? What are the consequences? Can
there be problems of great importance that can be avoided by choosing a suited
OS? Question number eight focuses on what features people consider essential to
evaluate an OS’s quality. We seek to find what people in the field look at and
find important to them. This is also interesting to ask and compare with our
findings from the mapping study. Question number nine investigates if there can
be features in an operating system that one wants to explore in practice. We want
to know if there can be attributes that one wants to test in practice to ensure
that what is claimed about the OS match in reality. Question ten asks for aspects
one can use to compare different OSs. Comparing this question to number eight,
one can find that they are similar, but we think that it can be some features are
more logical and easier to compare than others. Question eleven aims to discover
how the security of kernels in embedded systems can be considered. We want to
compare with our mapping study findings and see if other factors can be crucial.
It is interesting to look at the kernel because of our discussions with Advenica and
that it plays a big part in the security of the whole system. Number twelve asks
about security, in general, to find other essential security concerns when looking
at an operating system. The last two questions investigate how a person wants
this method to present the results. Is it preferable to give an OS based on the specifications given, or does one only want to have guidance on what to look for? What must be considered when creating an evaluation model according to people working in the field?

**Agreement**

Before each interview, we gave each participant an agreement document with information about the data collection, its purpose, and how we intend to use the data. In figure 4.2, one can see what this document looks like.

**4.2.2 Interview sessions**

In total, we conducted eight interviews. We both participated in each interview and took turns leading the interview and taking notes. Before we began each interview, we informed the participant about what we were doing as a thesis work, presented the agreement, and let them read through it. We made it clear that the recording from the interview would be removed after we did the interview summary and that they could read through it so that nothing was misunderstood or misinterpreted. All participants were okay with us recording the interview. All sessions were conducted similarly; we started with the questions, going through them in the order they are presented in table 4.2 and finished with what we had developed on our method to hear their thoughts on our approach. Question nine was the only question that some participants struggled to answer due to how it was phrased. We clarified with an example that one might want to see how much power the operating system draws when in an idle state to ensure that the system can supply that power if using a battery, for example. This made the question more straightforward for the participants to understand. Otherwise, there were no issues with the other questions, and each session was kept within 45 minutes.

**4.2.3 After the interviews**

After each session, we reviewed the notes that the person responsible for it had taken and summarized what the participant had answered. For all closed questions, we filled a table with what the person answered to be used later for displaying the data in a chart. For the open questions, we summarized the answer into keywords and phrases that were the main content of the person’s responses. This summary was then sent to each participant for them to review.
Agreement

The interview aims to collect information to create an evaluation model for selecting operating systems and kernels for embedded systems and your information will only be used for this purpose.

- The interview is estimated to last 45 minutes.
- You could at any time withdraw from the study and terminate your contribution.
- The researchers will do their best to keep your data anonymous and you have the right to review how your data is treated.
- I'm okay with the interview being audio recorded. □
  
  *Audio recorded data will be transcribed and the audio file will be deleted after transcription.*

- All instances will be stored without personal names.
- You are allowed to skip answering questions if there is a risk to reveal confidential data.

I have read and agree with the terms above.

Name: ____________________

Figure 4.2: Agreement for the interviews
Chapter 5

Results

This chapter presents the result of the mapping study and the interviews. All results will be presented and analyzed. The evaluation method will also be presented, and the motivation behind our choices will be given. The papers found in the mapping study are summarized and can be found in section 2.3.

5.1 Mapping study

In this section, the articles that were of interest are discussed and analyzed, and critical themes and patterns are presented. In table 5.1, articles brought up in section 2.3 are listed with which criterion was met during the collection of articles in the mapping study. In figure 5.1, one can observe a taxonomy of the finding from the mapping study.
<table>
<thead>
<tr>
<th>Title of paper</th>
<th>Has the choice of OS been challenging</th>
<th>Has a security deficiency of an OS been raised?</th>
<th>Is the process for selecting a kernel or Operating System specified?</th>
<th>Have researchers published a solution to the problem?</th>
<th>Does the work contribute towards solving the stated research questions for this work?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment of Buffer Overflow Based Attacks On an IoT Operating System</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Optimizing security and quality of service in a real-time operating system using multi-objective Bat algorithm</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Operating System, Applications and Protocols Design, and Validation Techniques</td>
<td>Yes</td>
<td>No</td>
<td>Metrics are mentioned</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Challenges in Designing Exploit Mitigations for Deeply Embedded Systems</td>
<td>Somewhat</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>A Survey of Operating System Microkernel</td>
<td>Yes</td>
<td>Yes</td>
<td>Metrics are mentioned</td>
<td>Yes</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Operating system support, protocol stack with key concerns and tested</td>
<td>Yes</td>
<td>Yes</td>
<td>Metrics are mentioned</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Control-Flow Integrity for Real-Time Operating Systems: Open Issues and Challenges</td>
<td>Yes</td>
<td>Yes</td>
<td>Not mentioned</td>
<td>Yes</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Device Driver and System Call Isolation in Embedded Devices</td>
<td>Yes</td>
<td>Yes</td>
<td>Not mentioned</td>
<td>Yes</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Evaluating the Impact of Hardware Faults on Program Execution in a Microkernel Environment</td>
<td>Somewhat</td>
<td>Yes</td>
<td>Metrics are mentioned</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Implementation and Performance Evaluation of Embedded IPsec in Microkernel OS</td>
<td>Yes</td>
<td>Yes</td>
<td>Not mentioned</td>
<td>Yes</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Operating Systems for Internet of Things Low-End Devices: Analysis and Benchmarking</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance of Secure Boot in Embedded Systems</td>
<td>No</td>
<td>Yes</td>
<td>Not mentioned</td>
<td>Somewhat</td>
<td>Somewhat</td>
</tr>
<tr>
<td>The open source, formally-proven sel4 microkernel: considerations for use in avionics</td>
<td>Somewhat</td>
<td>Yes</td>
<td>Not mentioned</td>
<td>Somewhat</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Cybersecurity Standards in the Context of Operating System Security and Host Vulnerability Evaluation</td>
<td>Yes</td>
<td>Yes</td>
<td>Metrics are mentioned</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security Verification Method of Embedded Operating System Semaphore</td>
<td>Yes</td>
<td>Yes</td>
<td>Not mentioned</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mechanism based on Coq</td>
<td>Yes</td>
<td>Yes</td>
<td>Not mentioned</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5.1: Papers found from mapping study & criteria protocol
In the majority of the papers found, it is highlighted that selecting an OS for an embedded system is crucial due to the resource restraints within these systems, and the operating systems that fit into these systems often lack general features that exist for GPOSs. It is crucial to exercise caution and thoughtfulness when making a decision regarding these systems, as they are responsible for overseeing critical infrastructure and can result in severe consequences if they fall into the wrong hands or fail to meet their intended purpose. Song et al.[30] mention that
the operating system is the foundation of security within these systems. Hamdani et al.\cite{16} highlight that security within these systems is essential since their dependency is growing. The article highlights the concerning issue of security being given low priority and emphasizes the critical importance of taking it seriously. Patel et al.\cite{25} underline that security is one of the main problems with IoT; without it, it will likely fail. Abbasi et al.\cite{8}, in particular, mention four interesting problems regarding embedded systems. The first problem addresses the fragmented world of embedded OS with different vendors that have led to matters like patching issues, cost, and lack of general features regarding security. The second problem addresses these systems’ reduced power and size, which causes fewer security features to be utilized. The third challenge is finding a balance between countering threats and avoiding negative impacts on the overall critical system. The last problem the paper brings up is the lack of modern security measures. When it comes to security, Song et al.\cite{30} outline a number of software tactics to guarantee it. These include user identification, access control, strict privilege management, trusted path, covert channel, security auditing, and virus defenses. The operating system aims to achieve several objectives, including monitoring user actions, verifying system integrity, and overseeing its performance. These goals are accomplished through the various implementations mentioned.

5.1.2 Security features & aspects

Just looking at the architecture of the kernel, the microkernel is considered to be more secure than the monolithic.\cite{21} This is due to the isolation of sensible processes that are put in kernel mode or privilege mode, with higher priority than processes running in user mode. The kernel is viewed as the trusted computing base of an operating system. The monolithic kernel runs all processes in kernel mode, all with the highest priority. Malicious processes can get access to the trusted computing base and crash the system. The microkernel’s smaller size lends to its reputation for reliability, which is closely linked to security. With the smaller size and real-time performance, the microkernel is more suited to be used for embedded systems. But the monolithic is more suitable if performance is of the essence due to the reduced number of context switching between address spaces. Regarding security concerns regarding embedded systems, memory corruption is one concern mentioned in the papers\cite{8}\cite{23}, for example, in the form of a buffer overflow. This is usually due to the lack of boundary checking in processing data functions and using the C language in these systems. What can be found to prevent this type of concern is executable space protection, address space layout randomization, stack canaries, and stack guard. Regarding stack canaries, Moghadam et al.\cite{23} mention that a new paradigm of the attack appeared from this, code-reuse attacks, that lets a threat actor take over the flow of execution within the sys-
tem. This type of attack can be avoided by adopting control-flow integrity as a prevention technique. Integrity is also important when talking about the kernel. To ensure that the kernel code has not been modified, one of the most common prevention mechanisms against ensuring each booting step of kernel code is called secure boot. Devices drivers and applications developed by other parties can be an issue within embedded systems, according to Malenko et al.\[22\]. Since embedded systems usually use some form of memory protection unit to protect against adversaries. The authors of the papers suggest the use of fault isolation that can protect against misbehavior and protect the OS reliability.

Security standards and protocols are suggested tools to achieve security in computer systems. Hamdani et al.\[16\] mentioned that it is a tedious task for end-users to ensure that they are protected against security threats since security standards are only guidelines suggested to follow to achieve security. Hamad et al.\[15\] mention that to secure the networking in an embedded system, one can use IPsec, which uses two encapsulation protocols, Authentication Header and Encapsulation Secure Payload.

One thing that is highlighted in some of the papers[29][25][33] is the importance of determinism for these embedded systems and that one can not just use existing security implementations without ensuring that they are suited to be inside such a constrained system. Mhamed Zineddine\[34\] highlights worst-case execution time and says that one must respect and take it into consideration when introducing security for embedded systems.

5.1.3 Performance, Maintainability & Reliability

To achieve real-time assurance, the papers from the mapping study suggest that it lies in the kernel to operate in a deterministic way[29][25][33]. One recommended approach to accomplish this is by utilizing the microkernel architecture. This type of architecture is considered better suited for implementation and more dependable and manageable than the monolithic kernel since it is smaller in size. Liu et al. \[21\] say that since the microkernel divides its processes between kernel space and user space, it is more reliable and maintainable than the monolithic kernel. However, the authors state that the monolithic kernel performs better due to context and address space switching occurring in the microkernel. Silva et al.\[29\] say that two factors decide the system’s power consumption, determinism, and performance; scheduling policies and kernel architecture. The power consumption is also primarily affected by how the OS uses memory, where using it more dynamically will preserve it. According to the same paper, selecting the appropriate operating system is essential for achieving real-time performance. Patel et al.\[25\] mention network connectivity/protocol support as an essential factor that can substantially influence the device’s energy consumption. While examining hardware
faults, Bekele et al.\cite{10} propose that the microkernel is more efficient in managing silent data crashes than the monolithic kernel, but it experiences more hangups than the monolithic kernel. The paper advocates for this approach since hangups can be identified and resolved while a silent data crash is harder to spot. Chen et al.\cite{11} and Vanderleest\cite{31} discuss the importance of proving that something works as advertised and intended. This is done by formal verification based on a strict mathematical foundation. According to one of the papers, this is the only way to ensure that the software is free from programming errors. However, the authors of that paper highlight that it also depends on the verification document that the formal method is given.

5.1.4 Performance vs. Security

Almost all papers discuss the overhead cost of introducing security into these embedded systems. This is because of the increased computing power required in implementing security measures. Mhamed Zineddine\cite{34} mentions in his papers that improving security while maintaining the quality of service is paradoxical, meaning that introducing security will increase computing power, leading to performance overhead. It is there for essential to consider and respect the worst-case execution time. Patel et al.\cite{25} highlight that for a device that needs real-time support, it is essential for the kernel to operate with deterministic run-time. Zikria et al.\cite{33} also state this. Liu et al. \cite{21} compare the microkernel against the monolithic kernel in their paper, where they state the microkernel is more secure due to how it is built, but the monolithic kernel has better performance. Abbasi et al.\cite{8} bring up that security measures must be robustly trustworthy and available while also stopping or mitigating any possible attack on the system. This balancing is tricky since the measure must be robust but not too aggressive so that they affect critical parts of the system negatively. They also suggest an exploit mitigation baseline design, which helps to increase security with no sizeable overhead. Moghadam et al. \cite{23} bring up that a full-cover solution for control-flow integrity in embedded systems is still an open issue and that the solutions they have looked at only offered a trade-off between protection and performance. Hamad et al.\cite{15} talk about the overhead of using IPsec, and using it in constrained systems causes significant overhead and affects the system’s throughput. Only using the Authentication Header can still help to protect with acceptable overhead. Profentaz et al.\cite{27} center their paper around the secure boot and say that a software implementation of it introduced a 4% overhead compared to a hardware implementation that had a 36% overhead impact. According to the authors, embedded systems often neglect to implement secure boot, leaving the boot process and internal software vulnerable to manipulation. The overhead of securing the boot process can affect these systems negatively regarding memory capacity, energy consumption,
and boot time.

5.2 Interviews

This section presents the result of the interviews. Each question will be carefully reviewed, as what the participants answered, and a summary with an evaluation of the content from this summary.

5.2.1 Question 1

Do you select OS or kernel based on prior experience evaluate for each project gut feeling?

![Figure 5.2: Result from question one](image)

All participants agreed that the most common path to take when choosing an operating system is to go with what has worked in the past. One participant says it is both past experiences and to do an evaluation that is typically used to select which OS to use. Many of the participants explain that this saves time at the beginning of the project due to factors such as experience working with a
How have the result from question one affected the results?

Figure 5.3: Result from question two

specific OS, reusing software developed in earlier projects, and the research time to investigate different OSs is expensive and often not worth it. However, most of the participants highlight that it depends on what type of project, what kind of experience there exists within the team working on the project, and how it is usually done within the company. A few of the participants highlighted that they would prefer to do some form of evaluation since this can prevent issues in the future if choosing an OS that is not suited. In figure 5.2, one can observe the result as a graph.

5.2.2 Question 2

For question two, there were a bit different opinions from the participants. The majority thought it was an excellent choice to rely on past experience since, for many, it has not introduced any unexpected problems, and they felt the project could be started relatively quickly. Some participants explained that, in some cases, the choice of OSs was limited due to hardware limitations. One other participant expressed that there will always be a better choice of OS. The person explained that poor choice of OS is when it introduces security flaws. The focus should be to look at and oversee the number of Critical Vulnerabilities Exploits (CVEs) that could affect features that will be used in the project. This participant also explained that the performance related to the choice of OS has more to do with how the software is written rather than OS. One participant said it was hard
Is it common to implement own features in the OS or kernel?

![Bar graph showing answers to the question](image)

Figure 5.4: Result from question three

to tell since they felt they lacked experience. In figure 5.3, one can observe the result as a graph.

### 5.2.3 Question 3

There was a big spread on what the participants answered in this question. This depended on how much experience the person had before working within an OS, what type of product the person was working with, and whether it was even necessary. Some participants said it was more common to fix security issues or optimize the kernel by removing what they felt to be unnecessary features, not adding extra features. They felt this was important to improve security, for example, if a security issue was detected and performed. One participant expressed that one should not add anything to kernel space since this can introduce security issues to the operating system. However, if performance is of the essence, one could add features to the kernel. Another participant said that Linux usually contains the features one needs and that it is more common to remove features that one is not using or slightly change a feature to function in a certain way. The rest of the participants said either that it was not desired since the OS already had the features that were wanted for the project or they did not want to answer the question. In figure 5.4, one can observe what the participants answered in a graph form.
5.2.4 Question 4

Many participants felt it was challenging to say how long it usually takes to select an OS for a project. It was either due to a lack of experience since they were not involved in that specific process or that it depended on what type of product they were working with. One participant mentioned that, in some cases, a version of Linux is shipped with a system that will be used in that case. This person also highlights that they think the decision is usually made too quickly and more time should be put into it. Linux seems to be a common choice since other participants mention that this is what is usually chosen and used in the projects. This makes it a quick process and decision, either taking less than a week or two to three weeks. Another participant mentioned that sometimes one wants to do a security analysis to find if the kernel is suitable and trusted, affecting the time it takes to choose an OS. One person says it could take up to six months if one wants to do it thoroughly. It also depends on the complexity of the problem that they have worked with, the experience within the team, and what has been done in earlier projects. In figure 5.5, one can observe the result in a graph form.
Almost all participants answered differently when asked about how long the process of choosing an OS is desired to take. The majority expressed that the context significantly affects how long this process should take. What type of product, the kernel size, and how quickly one must get started with the project due to customer’s requirements are given as examples from the participants. One participant mentions that if security is essential for the project, one might want to do pen-testing or read the kernel source code. But the person does not think it should take more than one sprint, about two to three weeks. One person highlights that the process should not be stressed and must take time to ensure a suitable choice. Another participant thinks starting the project as soon as possible is more important and that the process should not take more than two weeks. In figure 5.6, one can observe what the participants answered on question five using a graph.

5.2.6 Question 6

The majority of the participants explained that the hardware is what guides which OS one is using for a project. There are multiple reasons for this, according to
Figure 5.7: Result from question 6

the participants. One expresses that adapting the software to the hardware is often more manageable, but it depends on the current case. Sometimes the cost is deciding factor. Other participants say deciding on the hardware first feels more common. Another participant says that hardware with less power, meaning less CPU power, memory, and storage, limits the number of options of OSs one can run on that system, whereas going with an extensive and demanding OS, one can either remove features to decrease its size and demand. Those that suggest that both hardware and software mean that sometimes one wants to use a specific feature within an OS, and sometimes one chooses hardware first due to security aspects. One of the participants suggests that one wants to select hardware that is compatible with many different operating systems, not limiting the number of OSs to choose from. The same goes the other way around; one does not want to limit the hardware choice due to the OS one has chosen. However, this person highlights that hardware has more impact on what options one does have to choose from regarding OSs than the other way around. In figure 5.7, one can observe the outcome of this question in the form of a graph.

5.2.7 Question 7

Question seven focuses on what impact an insufficient choice of OS can have on a project. All the participants agreed that choosing an unsuitable OS or kernel
Is there any major impact of selecting an insufficient OS/kernel?

![Bar chart showing results](image)

Figure 5.8: Result from question seven

could significantly impact the project. Most of the participants gave similar answers and underlined the importance of choosing an OS carefully. Based on the responses of the participants, it was found that a wrong decision often results in an increase in workload. There are several reasons why the workload increases due to poor decisions. It could be due to patching the operating system, including addressing security vulnerabilities and software exploits, modifying the kernel, adding missing functionalities, or fixing bugs. One participant brought up licenses as an issue. They mentioned that if something is changed within the operating system, like own-developed features, one might be forced to give up them due to the requirements of an open-source license.

### 5.2.8 Question 8

There were varied responses from the participants concerning which quality of service characteristics to look for when evaluating an OS. The most common answer was to look at how active the OS is regarding update frequency, forum posts, and how many users use it. The number of users also tells how active it is. The point of it is that more activity regarding the OS is a good sign, according to some of the participants. This makes the OS more likely to be well-maintained and mature. Some participants highlighted the importance of security updates and the availability of support in some form, whether someone to call or support on the Internet. If competence is lacking within the team working on the project, it is crucial to have someone that can aid and avoid issues. How the manufacturers handle security generally is something that almost all participants bring up as a
Are there any theoretical aspects that might raise practical uncertainties?

![Figure 5.9: Result from question nine](image)

critical characteristic to look at. Some of the criteria for evaluating an operating system’s security include its support for secure boot, its design with security as a priority, certification to meet security requirements, the number of CVEs, and the speed and effectiveness of resolving security issues. Other participants mentioned the matter of hardware support. One participant mentions knowing if the operating system has real-time capabilities is essential if the system one is developing needs it. The participants also mentioned the support to modify the OS and how configurable the OS is. It can be interesting to know if it comes with a USB stack or network stack or if that is something one has to implement independently.

5.2.9 Question 9

As mentioned in the implementation chapter 4, this question was difficult for the participants to answer. Many of them said that there either had not been any need to make sure it worked as advertised and that they had assumed it would work. Linux is brought up as an example, where they state that it is well-used and makes it reliable. Others say they lack the experience to answer the question or do not want to answer. Some participants believe verifying the operating system’s functionality as advertised is crucial. They express the need to understand the mechanisms for safeguarding inter-processes and the security measures implemented to guarantee protection. Sometimes, one might want to read the kernel’s source code to see how it functions and validate that it works. Scheduling is also mentioned as something one wants to do sure works as promoted.
and that it fulfills the real-time requirements. In figure 5.9, the result from this question can be viewed as a graph.

5.2.10 Question 10

To compare OSs against each other and which aspects to use when comparing, the participants mentioned similar aspects that were brought up for question nine. Documentation and support are also mentioned here as essential aspects that can decide which OS to choose. The project’s time span plays a significant role when deciding on an OS and whether it is future-proof or not. Some participants highlighted that seeing how long the OS will be supported is vital. This is important to know so that one can get support from the manufacturer if needed, in case the documentation is insufficient, and to get security updates if bugs or other issues arise. One aspect that one participant gave was that one wants to make sure that if an OS is chosen, can one handle it if it turns out to have flaws and to be inadequate? Or will this jeopardize the whole project? Many participants also mention experience as an essential factor, how people in the project feel about working with the OS, their knowledge about it, and what they prefer. One participant says that gut feeling often decides what to go for. Other aspects that were mentioned were how easy it is to modify, remove unnecessary features, and harden the OS. A participant also mentions the availability of packages as essential and an aspect one can use to compare OSs.

5.2.11 Question 11

Some participants struggled with answering the question about how the kernel’s security could be evaluated. This was due to either a lack of experience, the difficulty in determining how to evaluate its security, or that they did not want to answer. For those who could answer this question, the number of CVEs was the most common answer from the interviewees. One wants to look for how many CVEs are open, meaning not resolved, and how many users the OS has. One participant said that the quota between the number of CVEs and the number of participants is interesting to look at since a lower number of CVEs does not say everything about the kernel’s security. The same person also says that looking at how often bugs are presented and how quickly they are resolved is interesting. However, another participant highlights the fact that if a functionality in the OS is likely to be used in an exploit, this particular feature might not even be used in the project. The participant means then it does not matter if there exists a CVE. One can remove these functionalities. The same person also says that the newest patch for an OS is not necessarily a more secure choice and that one must carefully compare, read about the patches and understand which of them suits
the project best. Another participant mentioned that looking at the OS’s intent could be interesting. Does it aim to be secure, and do the developers of the OS have that in mind during its development and with updates? The same person also mentions that knowing how isolation between the processes works and their policies and privileges is interesting. Two participants mentioned that one wants to look for old reports or investigations on the kernel in context to use as navigation to evaluate the security. One individual notes that conducting personal research can be pretty time-consuming. Additionally, they mention that comprehending a massive kernel such as Linux can prove to be challenging due to its size and complexity.

5.2.12 Question 12

Similar answers from question eleven are also found for the question about common security concerns. Participants mention process isolation and the number of CVEs as essential security concerns. One wants to ensure that one process cannot affect or take over another process. Another participant talks about similar issues but regarding interfaces. The person listed network, USB, serial ports, and debugging interface as concerning parts. Hackers can use these to perform code execution when programs are reading from these interfaces. Another issue that one participant brought up is the concern regarding memory being overwritten. They also added that it is vital for memory separation between processes. The person also highlights that it should be easy for a developer to do right. If the developer has made a mistake, it should be easy to find. One wants to detect if there are any security deficiencies within the OS. A participant has expressed the importance of transparency from the OS manufacturer, particularly regarding how vulnerabilities are handled. They believe that swift action and transparency are crucial in managing any potential vulnerabilities.

5.2.13 Question 13

When asked about their desired outcome from an evaluation method, the participants provided varying responses.; however, some of them were in the same ballpark. Some of the participants mentioned that they would like to have some guidance rather than just be given an OS from a requirement specification. This is desirable for the sake of transparency; one wants to know what to look for and why since that can also be used to convince a customer or a project manager. A list of the suggestions’ pros and cons should also be presented. It is also desirable to be part of the process to see how the method came to a particular conclusion, says one of the participants. One participant thinks that there should be some grading of what is essential for the project. Features that are for security might not be
necessary if performance is of higher priority. It would be nice to grade different
aspects, and the evaluation method gave different recommendations if one aspect
is more essential than others, according to this participant. Other participants
suggested that a list of all OSs that could be used, ranked after how much they
fulfill the requirements. All aspects and features that are used to evaluate the OS
can be listed next to it. This would be nice to have, according to these partici-
pants, also to compare different OSs. The person who suggested having a grading
system would also like to see a list of OSs that updates when the priority changes.
Another also thinks that an interactive evaluation model is preferable.

5.2.14 Question 14

For the last question about what can be of interest to look at when building an
evaluation method, most participants mentioned similar aspects discussed in the
earlier questions. Number of CVEs, what type of hardware support there is, the
update frequency of the OS, how active forums are regarding the OS, certificates,
documentation, support, licenses that one needs to know about, architecture, what
the OS is usually used for, can one modify it, security features that are supported
and what organizations that is behind the OS are things that the participants
mentioned that are interesting to look at. According to one participant, the last
thing mentioned is particularly interesting to know if supply chain attacks are
a concern. One participant suggested that all parameters the method suggests
should be explained and motivated by why it is essential to look at. One participant
recommended developing a streamlined version of the process, considering time
constraints in project management. The method should also be appealing and
efficient to use in such cases.

5.3 Answers to the research questions

5.3.1 How could the security of an operating system for
embedded systems be evaluated?

For this question, based on the findings from the mapping study and the inter-
views, several characteristics have been suggested to look at regarding the security
evaluation of OSs for embedded systems. In figure 5.10, a taxonomy of the findings
from the mapping study can be seen.
Figure 5.10: Taxonomy of the answer to research question 1
Security features

The support for features such as process and memory isolation, secure boot, hardware support for security features, user identification, access control, most minor privilege management, trusted path, covert channel, security audit, virus defenses, address space layout randomization, stack protector, stack canaries, control flow integrity, cryptographic libraries, IPsec(Authentication Header, and Encapsulation Secure Payload) is one way to evaluate the security of the operating system. These features that an OS can provide help to protect the system in various ways.

Architecture

By looking at the operating system’s architecture, one can use that as a feature to evaluate the security. The microkernel is considered to be more secure than the monolithic kernel due to its smaller size with only the necessary parts running in privileged mode, different access control privileges for each application, the capability to involve system policies to manage application communications and dividing of processes between user space and kernel space. This makes it more secure, reliable, and maintainable. The microkernel is also considered to be better at handling hardware faults.

Policies, Guidelines, Verification & Certificates

If the operating system does follow principles, protocols, and standards with security in mind, it can indicate that the system is more secure and the organization behind it takes security seriously. The system follows guidelines to improve security. If the operating system or features within it are formally verified, this tells that the software is free from programming errors and follows the requirements set for it.

General

To understand the purpose of an operating system, it’s essential to consider the manufacturer’s intentions. Are they prioritizing security? What measures are in place to protect users? These are crucial questions to ask and research when evaluating an operating system. One can look at the number of CVEs, how often CVEs are resolved, how transparent manufacturers are with these CVEs, and how they are resolved. Can the OS help developers to find mistakes made or security deficiencies? Comparing different versions of OSs can be used as an indicator. Is there a version that is more stable and secure than others?
5.3.2 What characteristics of an operating system for embedded systems could be considered for evaluating the quality of an operating system?

For this question, based on the findings from the mapping study and the interviews, several characteristics have been suggested to look at regarding the quality of an OS for embedded systems. Figure 5.11 contains a taxonomy of the key aspects.

Features within the operating system

What scheduling policies the OS uses tells how it handles real-time demands and how this affects the system’s power consumption. One can look for what stacks (Network, USB, etc.) come with the OS and what packages are available on the OS. When considering an operating system, it’s essential to assess its kernel architecture, network protocol support, memory allocation and management, and how these features are implemented. Regarding the kernel architecture, the microkernel is viewed as more secure, reliable, and maintainable than the monolithic kernel; however, the monolithic kernel performs better.
Maintainability

The OS’s size is a metric that tells how demanding it is and easy it is to maintain. One can examine how often the OS is updated and how long it will be supported. Documentation regarding the OS, if it exists, and how much there exists of it. Support, if issues arise, via chat, phone, or forums. How active is it in these forums and how users the OS has can also tell how mature and well-maintained it is. One can see if it is possible to modify the OS (implement its features or modify features within it) and how configurable it is.

5.3.3 Can it be determined if an operating system fits the specification of a project well? If so, how?

Yes, by looking at how well it fulfills the requirements for the project, both in terms of security and quality. The characteristics mentioned in research questions one and two can be used to determine how well it fits. One can look at the timeline of the product and how long an OS is supported. Is the OS open-source, or does it come with license costs? Is it possible to implement your own features without giving up due to a license? Is there a need for it? If open CVEs exist, are they affecting functionality intended to be used in the project? These questions can help decide if the operating system fits the specification.
5.3.4 Can one compare two or more operating systems with the information from RQ1, RQ2 & RQ3? If so, how?

Yes, looking at what is mentioned in research question three, one compares how many of the requirements each OS fulfills. Beyond what is mentioned in each research question, the knowledge and experience within the project group are aspects to consider. What is the preference, and how well do people know the operating system? How can this affect the project? One can also think about what consequences an insufficient choice can entail. Does choosing one OS reduce the impact risk if that choice is poor?

5.3.5 How can one design a general method that considers RQ1-4 to select an operating system that can fulfill the specified requirements of a given project?

Here, I will present the method I and the other student developed for this question. It will divide into two different approaches, where one approach is more suited for project groups and people who feel they have lots of experience with the product. A more lightweight method that looks at the most critical aspects and takes less time to go through. The other approach goes more thoroughly through the aspects to think about and what to look for regarding what to look for in an operating
system. Both methods can be seen as guidelines that bring forth requirements and specifications and asks questions about what is needed and essential. The outcome of how one answers these questions can then be used to compare different OSs and to find a suitable choice for the project. The motivation for having two approaches is that from the discussions had with Advenica, cost and time are highlighted as the main reasons for not doing extensive research about different OSs. When having options, project groups can choose to include more aspects to look at if there is time and they feel it can be of substantial impact. For each heading explaining an action in the method, there will be an (Optional), meaning that that section can be skipped if going for the lightweight approach. Figure 5.13 shows an overview of the proposed method.

**Product specification (Optional)**

For each question listed below, answer each of them as detailed as possible:

- What type of product is built and for whom?
- Are there any communication requirements for the product? If yes, which?
- Are there any requirements for security? If yes, which?
- Are there any requirements for reliability? If yes, which?
- Are there any real-time requirements?
- What is needed to solve the task?
- For how long is the product planned to exist?
- What type of experience is there in the project group?

All of these questions are asked to raise awareness and understand what product one is dealing with and what experience there is within the team. These questions help to understand the current situation and what needs to be achieved. This part of the method should also be kept open and agile since one should be open to changes for these questions mentioned if one sees a better way to approach the product.

**Threat modelling**

This part of the method should be kept on a high level and aims to acquire knowledge early in the project. It does not require to be complete since covering all possible threats, both known and unknown, is too big of a task and time-consuming. For the threat modeling, there are five steps to go through and answer:
• What type of data does the system contain?
• To whom can this data be interesting?
• How can this data be compromised?
• What can be done to mitigate these effects?
• Validating and verification of success of actions taken

The initial step is to comprehend the data present within the system. This understanding is crucial in determining what needs to be safeguarded and what is of lesser importance. It also aims to understand what importance security is for the system in context and how one can prioritize what data to secure and strategies to use. Question number two aims to list potential adversaries. This is to raise awareness of whom the data can be of interest to and how they would interact with the system if their aim is to compromise the data. The third step is to find how the data within the system can be compromised. Suggested is to list the common attacks that can be used to compromise the data and how that can be done. This gives insight into how they work and how it can be done for the system in context. This "variable", ways to comprise the data, should be updated and kept agile since adversaries constantly develop new ways to compromise data. The fourth step is to investigate how to protect against these adversaries. A list of security features can be found in the section called SWOT-analysis, and one solution there might help mitigate an attack, but these are probably not the only ones. Since the landscape of attacks and solutions constantly changes, this part should also be kept open and agile. The last step is to validate and verify these mitigation actions suggested. There are different possible options to do this. One can look for some accreditation of the suggested action. This puts the trust in external parties and takes low effort. Another way could be to read research papers about the mitigation action. This also puts the trust in someone else and that you, as a reader, have some knowledge about what you are reading and what to look for. This also takes more effort to do. The last suggestion is to conduct penetration testing, which requires much effort and knowledge but puts trust in oneself. When considering security measures, assessing whether the implementation cost justifies the value of what’s being protected. How much security is truly necessary should be carefully evaluated.

SWOT-analysis

In this part, one wants to go through each mitigation solution found in the section called Threat modelling and features that the OS should have which can be

60
the ones found in the section called *Product specification*, and perform a SWOT-analysis. This is to properly investigate what a feature, a requirement, or an aspect brings in terms of strengths, weaknesses, opportunities, and threats regarding the product in context. It can be viewed as an inventory of what is needed and what should be prioritized to have in the system. Depending on how many aspects and requirements were gathered from these two sections, it is also interesting to consider the features and aspects listed down below. Examples of security features to consider:

- Buffer overflow protection - Address space layout randomization, stack protector, stack canaries
- Secure boot
- Cryptographic libraries
- Network protocols - IPsec(Authentication Header, Encapsulation Secure Payload)
- Control flow integrity
- Virus defenses
- User identification
- Memory isolation
- Process isolation
- External hardware support for security features - Memory Protection Unit, Memory Managing Unit, Executable Space Protection
- Security audit
- Access control
- Most minor privilege management
- Trusted path
- Covert channel

There are undoubtedly other features that are not listed here. Make sure to investigate thoroughly what features are of relevance for the project. When looking at each security feature, the main thing to consider is understanding how it can help mitigate attacks from adversaries and what overhead the implementation
comes with. Depending on what features one adapts, it comes with more or less overhead that affects the system. Depending on what hardware one uses, this aspect can then be crucial to look at. One shall also consider general aspects regarding performance, maintainability, and reliability. The list below contains examples of such aspects:

- Kernel architecture - Monolithic kernel, Microkernel, bare-bone, others
- Scheduling policies - real-time demands, performance, power consumption
- Programming languages - Support for a specific language
- Support for specific compilers
- Support for specific network protocols
- Available stacks - USB, network
- Available packages
- Memory allocation
- Size of the OS

**Hardware requirements**

The things that to look at regarding hardware requirements are:

- CPU - Support for it
- Available memory - both volatile and non-volatile, Support for it
- Networking capabilities - Support for it
- Power supply - Battery or constant powered
- Interfaces and I/O
- Available hardware for software features

Additionally, the project group can add its own requirements if something is missing from those listed above.
Software requirements

A list of software requirements should be acquired from the section called SWOT-analysis. What is also essential to look at is the licenses that might come with the OS. It is critical to know if it comes with obligations, like sharing own developed changes one has done within the OS, or if one is even allowed to modify the code base. Additionally, the project group can add its own requirements if something is missing from those listed in section SWOT-analysis.

Set of requirements

From the sections called Hardware requirements and Software requirements, a list of requirements should be obtained both for hardware and software requirements. This can now be used to look at operating systems that meet these requirements. If more than one operating system is of interest to use for the project, one can proceed to the section called General requirements - Lightweight(Optional) or General requirements - Complete(Optional) to look at more aspects that can be used to find which operating system to choose.

General requirements - Lightweight(Optional)

At this point, two or more OSs could be of interest. In this case, one can take a look at what each OS provides regarding:

- Documentation - How much information does the manufacturer provide? Is it updated?
- Support & Outsourcing - How big is the community or company behind it? Are there any community forums? Are they active? Is there someone to call to get support if needed? External help to get?
- Team experience - How comfortable are people in the team with OS? Do they have good or bad experiences with it? Knowledge?
- How long will the OS be supported with patches and updates?
- Is the OS configurable? Can one configure it based on the project specification?
- How frequent are updates and security patches? Is the manufacturer transparent about vulnerabilities within the OS and how they are resolved?
- How many CVEs are there? Are those relevant for the functionalities that are planned to be used?
• Who owns the software? Are they likely to be the target of a supply chain attack?

• What characteristics is the OS/kernel aiming for?

These are requirements that can be used beyond the hard requirements of the project and aims to differentiate various OSs from each other.

**General requirements - Complete(Optional)**

Together with what was mentioned in the section *General requirements - Lightweight*, one can also conduct more practical and extensive testing of the OSs and their kernels. First, one can perform a static code analysis of a kernel or OS. What parameters to choose depends on what the project members found relevant to look at. The Chidamber & Kemerer metrics[12] or lines of code are some examples. Beyond doing static analysis, one can supplement with a dynamic analysis. The following list contains aspects to look at:

• Memory footprint - how much does the OS demand of the memory? How much is left for the application that will be running on the system?

• Performance - On which OS does the application run the fastest?

• Scaleability - How much more can the application grow in size, user throughput, network throughput, etc.)

• Crash analysis - What happens in reality when the kernel or the application crashes?

• Perform pen-testing on the OS

• Power consumption - how much does the system draw when idle or in a loading state?
Chapter 6
Discussion

6.1 Compliance with the project requirements

The goal of developing a method for choosing a suitable operating system for an embedded system was fulfilled. A general method was successfully created by looking at research papers that address important aspects and issues regarding operating systems in embedded systems, and by interviewing experts in the field at Advenica. The research questions asked at the beginning of this project and the goal of a method were also answered. The method aimed to facilitate developers to make smarter and more informed decisions at the beginning of their projects, which I considered to be met.

How the project was planned and executed followed the plan set from the beginning and is considered successful. However, the way all parts were done could have been more efficient. This was due to uncertainties regarding conducting a thesis work when two universities were involved. Different criteria and supervisors with various feedback on the situation made it hard to start the project immediately. Since there was no experience with this type of constellation from both universities, it made it difficult to know how each step during the research method could be issued. Some parts could have been done differently to save time and effort. More work and results could have been done and shared if there were clearer instructions on how this collaboration could have been administered. This caused it to have some downtime, waiting for instructions and approval to proceed.

Looking at the results, some conclusions can be drawn from the findings. The importance of security within embedded systems is brought up both during the mapping study and during the interviews, where it is highlighted that it is one of the most important aspects of embedded systems and at the same time one of the biggest challenges for it. The fragmented market with different vendors with different priorities for their operating systems and the performance overhead of
implementing security features make it a hard problem to solve. This makes the argument that was stated in the interviews that Linux is a solid choice most of the time. It has a lot of features, a big knowledge base of users, forums that one can look for answers to if facing problems, and a very active community. One other interesting insight that was found from the interviews was that the number of CVEs does not directly mean the OS is more insecure than another OS. The number of users, how active the community is, which features it is affecting, and if these are of relevance for the project are aspects to take into consideration when looking at the number of CVEs.

### 6.2 Social and economic implications

The method produced by this thesis helps people to in a structured way, find important features and aspects to consider regarding choosing an operating system. It touches many different areas to look at, both known aspects but some that are not common, all made possible with information from people with knowledge and experience about the topic. It gives an opportunity for project groups to either use a lightweight version when time is of the essence or a throughout version that takes more aspects into consideration. By reducing the time of choosing a suitable operating system, this time can be used elsewhere in the project. It can help to reduce costs by avoiding an unsuitable choice of OS, that can cause problems in the future and cause significant damage. By employing a systematic approach and making informed decisions, it is possible to reduce the financial and temporal expenses associated with bug detection, feature addition, and the need to reconsider the choice of an operating system. It can help to find other operating systems than Linux that can better fit the project one is working on. In its current state, it is more suitable for people with some knowledge of operating systems. By improving the method, it can be suitable for more users with lesser knowledge about operating systems. This can help more people with lesser knowledge to make informed choices.

### 6.3 Project development

As mentioned in section 6.2, the method in its current state targets people with at least some knowledge about operating systems and embedded systems. By looking at the different aspects and features mentioned in chapter 5, one could do more throughout the investigation of how these aspects affect a system. One could look at security features regarding how well they can protect what it is set out to protect, how much in general a security feature causes overhead to the system,
how much power consumption it causes, and which feature does a better job at protecting the system. General aspects such as documentation and support from the manufacturer could be interesting to try to evaluate and see if there can be more aspects to take into consideration.

One limitation of this thesis is that the information gathered from the interviews represents a collective understanding of the subject based on past and current work experience and not one specific use case or project. Advenica has restrictions on sharing internal knowledge, aiming to protect its expertise within the organization. Because of this, the information presented in this thesis represents a collective understanding without sharing specific statements from individuals. Therefore, it is not stated which profession the person who stated something has or how long experience an individual has. One other limitation is that the method developed in this thesis has not been tested in a real-world scenario. There has been no opportunity for practical application or external validation other than some small discussion during the interviews. As a result, the effectiveness and performance of the method in real-world scenarios remain untested. To test the method in a real-time scenario would be interesting to do. Another drawback of the method is that it does not have any cost analysis. How much is the cost of a license or how much is the cost of development also considering time can be important to look at. As mentioned throughout the report, time constraint was a factor that affected this thesis and also what type of method that could be implemented. In the interviews, some participants mentioned that it would be interesting to have a program, where a big list containing many different OSs with all interesting features in different columns would be presented. A set of requirements and parameters could be set by the user and the program would then list which OS it finds to be best suited for the purpose of the project. This program could also give some motivation on why it recommends a certain OS. This would be an interesting follow-up to this thesis, to develop the method even further. The time constraint also affected how much information that could be collected from the mapping study. More features and aspects could be found and added to the list mentioned in this thesis.
Chapter 7

Reflection

For this thesis, I believe I have met the requirements that were set as goals for this course. I have formulated a problem and delimited it by looking at embedded systems. I have used a research method suited for the purpose of creating an evaluation method. This has been done by using a mapping study and interviews. I have gained deeper knowledge, both from researchers and practitioners about OSs for embedded systems, kernels, and aspects regarding security. The information from these sources was evaluated from the criterion set at the beginning of the mapping study, structured, and presented. All combined knowledge has been used to answer the research questions and develop the method. I have reflected on the results and given suggestions on how one could improve what has been developed. After this project, my knowledge about how to think when choosing an operating system has increased by a large. Understanding how people think and resonate regarding this issue has been much appreciated to learn about. To further explore this topic, I would need to learn more about features common in OSs for embedded systems, how they fundamentally work, and which choices of them impact a constrained system the most. This can help to highlight the most important aspects and make the choice of choosing an operating system less complex.
Bibliography


