Threat, risk, and vulnerability analyses during the development of IT systems in the Swedish Armed Forces

Ola Andersson

February 1, 2010
Master’s Thesis in Computing Science, 30 credits
Supervisor at CS-UmU: Jerry Eriksson
Supervisor at FOI: Jonas Hallberg
Supervisor at FOI: Johan Bengtsson
Examiner: Per Lindström

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE
SE-901 87 UMEÅ
SWEDEN
Abstract

This master’s thesis describes how two models from the Swedish Armed Forces; the risk management model and the IT lifecycle model can be combined. An example is then presented for how the risk management model can be extended for threats, risks, and vulnerabilities related to information technology. The combination and extension of the models are based on a literature study that lists and compares models and methods for threat, risk, and vulnerability analyses, as well as an analysis of threats related to information technology.

From the combined and extended model, a design proposal for how to implement the identified functionality was identified. Based on an evaluation that showed that the program NTE and the plugin EASTER were suitable as the foundation for this implementation, the program NTE and the plugin EASTER were extended with further functionality and resulted in the implementation of the plugin ASCENSION. This was evaluated and resulted in ideas for a possible re-design and examples of the future potential of ASCENSION.
### Contents

1 Introduction ........................................ 1  
   1.1 Short description of the task ..................... 1  
   1.2 FOI, Swedish Defence Research Agency ............. 1  
   1.3 The FoT project ................................ 2  
   1.4 Disposition .................................... 2  
   1.5 Terminology .................................... 2  
   1.6 Abbreviations .................................. 4  

2 Problem Description ................................. 5  
   2.1 Problem Statement ............................... 5  
   2.2 Goals .......................................... 6  
   2.3 Purpose ........................................ 6  
   2.4 Methods ........................................ 6  
   2.5 Related work ................................... 7  

3 Models and methods for threat, risk, and vulnerability analyses .... 9  
   3.1 A brief introduction to threat, risk, and vulnerability analyses .......... 9  
      3.1.1 Use of terminology .......................... 9  
      3.1.2 Properties of threat, risk and vulnerability analyses ............... 10  
   3.2 Models for analyses in the Swedish Armed Forces ....................... 11  
      3.2.1 The risk management model ................... 12  
      3.2.2 The IT lifecycle model ....................... 15  
   3.3 Models for analyses in other areas .......................... 18  
      3.3.1 Swedish Emergency Management Agency .................. 19  
      3.3.2 Swedish Rescue Services Agency .................. 21  
      3.3.3 Legal, Financial and Administrative Services Agency ............. 22  
      3.3.4 County Administrative Board of Stockholm .................... 23  
      3.3.5 Swedish Security Service ........................ 25  
      3.3.6 Swedish Road Administration ...................... 25  
      3.3.7 U.S. Department of Energy, Office of Energy Assurance ............ 26  
      3.3.8 British Telecom ............................... 29
| 3.3.9  | National Institute of Standards and Technology  | 29 |
| 3.3.10 | Attack graphs                                     | 31 |
| 3.3.11 | Miscellaneous methods                            | 33 |
| 3.4   | Comparisons between models                       | 34 |
| 3.4.1 | Step P2 of the IT lifecycle model                | 35 |
| 3.4.2 | Swedish Emergency Management Agency              | 35 |
| 3.4.3 | Swedish Rescue Services Agency                   | 36 |
| 3.4.4 | Legal, Financial and Administrative Services Agency | 36 |
| 3.4.5 | County Administrative Board of Stockholm         | 37 |
| 3.4.6 | Swedish Security Service                         | 37 |
| 3.4.7 | Swedish Road Administration                      | 37 |
| 3.4.8 | U.S. Department of Energy, Office of Energy Assurance | 38 |
| 3.4.9 | British Telecom                                  | 38 |
| 3.4.10| National Institute of Standards and Technology   | 39 |
| 3.4.11| Attack graphs                                    | 39 |
| 3.4.12| Miscellaneous methods                            | 40 |

| 4   | Threats to IT systems                            | 41 |
| 4.1 | Technical threats                                | 41 |
| 4.1.1| Collecting and exploiting information            | 41 |
| 4.1.2| Attacks                                         | 41 |
| 4.1.3| Malicious code                                   | 42 |
| 4.1.4| Adding or modification of hardware              | 42 |
| 4.1.5| Exposure of source code or blueprints            | 42 |
| 4.1.6| Wireless communication                           | 43 |
| 4.1.7| Wired communication                              | 43 |
| 4.2 | Social threats                                   | 43 |
| 4.2.1| Insiders                                        | 43 |
| 4.2.2| Social engineering                              | 44 |
| 4.3 | Human-computer interaction and security          | 45 |
| 4.3.1| The human factor                                | 45 |
| 4.3.2| User centered security                          | 45 |
| 4.3.3| Security policy and human behaviour             | 46 |
| 4.3.4| Human memory and passwords                      | 46 |

| 5   | Combination of models from the Swedish Armed Forces | 47 |
| 5.1 | Evaluation of how the risk management model can be combined with the IT lifecycle model | 47 |
| 5.2 | Extending the risk management model for IT         | 49 |
| 5.2.1| Step 1 - Establish basic values for the analysis  | 49 |
| 5.2.2| Step 2 - Concretize and estimate the threats      | 49 |
| 5.2.3| Step 3 - Identify protections and estimate vulnerabilities | 50 |
CONTENTS

5.2.4 Step 4 - Estimate the risk ........................................ 50
5.2.5 Step 5 - Risk management decision and plan for follow-up .... 50

6 Design for the new implementation ................................... 51
6.1 Description of NTE .................................................. 51
6.2 Design guidelines .................................................... 53
   6.2.1 General rules and guidelines for user-centered design ......... 54
   6.2.2 Guidelines for risk analyses ................................... 56
   6.2.3 Requirements from the Swedish Armed Forces ............... 57
6.3 Design proposal ..................................................... 58
   6.3.1 Three views .................................................... 58
   6.3.2 General design decisions ..................................... 61
6.4 Implementation of the combined and extended model ............... 62
   6.4.1 Input of data .................................................. 62
   6.4.2 Analysis of data ............................................... 65
   6.4.3 Visualization of data ......................................... 65
6.5 Evaluation of NTE .................................................. 65
6.6 Functionality that should be implemented .......................... 66

7 Implementation ....................................................... 69
7.1 General description of ASCENSION ................................ 70
   7.1.1 Data structure ............................................... 71
   7.1.2 GUI structure ............................................... 71
   7.1.3 Structure of the analysis .................................... 72
   7.1.4 Methods for reading and saving data ......................... 72
   7.1.5 Method for creating and removing relations ................. 73
   7.1.6 Methods for presenting data ................................ 74
7.2 Detailed description of some parts of ASCENSION .................. 75
   7.2.1 Data structure ............................................... 75
   7.2.2 GUI structure ............................................... 76
7.3 Evaluation ......................................................... 77
   7.3.1 Features that were implemented ............................. 77
   7.3.2 Features that were not implemented .......................... 78
   7.3.3 Issues with NTE and EASTER ............................... 79
   7.3.4 Limitations of ASCENSION ................................. 80
   7.3.5 Design problems for a re-design of ASCENSION ............. 82
   7.3.6 Future potential of ASCENSION ............................. 85

8 Conclusion .......................................................... 87
8.1 Achievements ....................................................... 87
8.2 Limitations ........................................................ 88
8.3 Future work ......................................................... 89
<table>
<thead>
<tr>
<th>9</th>
<th>Thanks</th>
<th>91</th>
</tr>
</thead>
<tbody>
<tr>
<td>References</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Translations</td>
<td>97</td>
</tr>
<tr>
<td>B</td>
<td>Figures</td>
<td>99</td>
</tr>
</tbody>
</table>
# List of Figures

3.1 The risk management process described with the terminology used in the Swedish Armed Forces risk management model .................................................. 11
3.2 The risk matrix used to present risks, where the different colours represent a risk value from (1) to (5) (adapted from the Swedish Armed Forces [1]). ... 13
3.3 An example of how the tool used for the risk management looks like at present (adapted from Palm [7]). ................................................................. 15
3.4 The seven steps and the structure of the Swedish Armed Forces IT lifecycle model (adapted from the Swedish Armed Forces [2]) ................................. 15
3.5 A more detailed description of step P2 in the model that shows the different parts that make up this step (adapted from Bengtsson & Hallberg [16]). .... 17
3.6 The different methods presented in this report shown along an axis from qualitative to quantitative methods (based and adapted from the classification in the reports by Nilsson and Nystedt [22, 24]. ........................................ 19
3.7 The different steps in the County Administrative Board of Stockholm’s model (adapted from their web site [27]). ....................................................... 24

5.1 A diagram that shows how the two models can be combined. The structure of the model is adapted from the report by Bengtsson & Hallberg [16]. ...... 48

6.1 A diagram showing how NTE is constructed, the relation to the database and the relation to the plugins (adapted from Bengtsson & Brinck [3]). ...... 52
6.2 The main user interface of EASTER before the new implementation .......... 53
6.3 A dialogue window where users can modify profiles for the entities by filling in security requirements. ......................................................... 53
6.4 A mockup of how the security demands can be motivated in the program by presenting the user with a form to fill in and submit. ....................... 68
6.5 A mockup of how critical point and threats can be represented in the graph view of the program. ......................................................... 68

7.1 A diagram showing how the implementation of EASTER looks like and which classes that are part of it or are disabled. ......................... 69
7.2 A diagram showing how the new implementation is connected to the existing classes in EASTER, keeping the same layout as Figure 7.1. 70

7.3 This is a class diagram of the classes that have been added to the implementation of EASTER. 70

7.4 A figure that shows how the different objects in the data structure are related to each other and which limitations that have been made to the relations in the GUI. 73

B.1 This is how the text view of the prototype looks like. 99

B.2 This is how the matrix view of the prototype looks like. 100

B.3 This is how the presentation of data in the prototype looks like. 101

B.4 This is a diagram that shows the expanded classes in the class diagram for ASCENSION. In addition to the information presented earlier, this also shows the methods in each class. 102

B.5 A diagram that shows how the two models can be combined. The structure of the model is adapted from the report by Bengtsson & Hallberg [16]. This version is enlarged so the details are easier to see. 103
Chapter 1

Introduction

The development of information technology in the Swedish Armed Forces should follow an established model in order to ensure that the developed system is secure against all threats that have been identified. However, a new model for risk management in general has been developed, but the model for information technology has not been updated to use the newly developed methods in this model. The development of information technology also lack the support of a tool that can help to improve and simplify the process. This master’s thesis has looked into this problem to see what options there are to improve the situation, if the existing solutions could be improved, and how.

1.1 Short description of the task

The task was divided into four parts and the first was to do a literature study on models and methods for threat, risk, and vulnerability analyses. The second part was to evaluate if, and how, two models used by the Swedish Armed Forces could be combined: the risk management model [1] and the IT lifecycle model [2]. It was also to present an example of how the risk management model could be extended for information technology. The third part was to evaluate if a program called NTE [3] would be used as the basis for the implementation of the combined and extended model. The evaluation should also result in a list of extensions to the program or functionality from the program that should be used in a new one. The fourth part was to choose some functionality from this list and implement these in a software tool, either NTE or a completely new program.

1.2 FOI, Swedish Defence Research Agency

FOI, Swedish Defence Research Agency is a research institute in the field of defence and security. FOI is responsible to the ministry of defence and is financed on contracts’ basis. It was established in 2001 as the previous governmental agencies FOA, National Defence Research Establishment and FFA, Aeronautical Research Institute was merged. Among the things that FOI researches and develops, there are sensor systems, signal processing, systems for crisis management, control and command systems, and methods for IT security. FOI is based in Stockholm, Linköping, and Umeå and had about 950 employees in 2009.
1.3 The FoT project

The work done in this master’s thesis was part of a larger project at FOI. This is called the FoT project and the aim of this project is to provide methods and tools for the assessment of information security, where the focus is the use of information systems. A main reason for assessing information security is to acquire knowledge about risk management processes. Therefore, the connections between risk and security assessment is vital to explore.

1.4 Disposition

The report is divided into the following chapters:

- Chapter 2 presents the task in more detail. This includes a problem statement, goals, purpose, the methods used, and related work.
- Chapter 3 presents a literature study about different methods for threat, risk, and vulnerability analyses. It describes and compares methods from the Swedish Armed Forces with other methods from the Swedish Government and other organizations.
- Chapter 4 presents different threats that can affect IT systems.
- Chapter 5 presents how the two models from the Swedish Armed Forces, the risk management model and the IT lifecycle model can be combined. It also describes how the risk management model can be extended for use with IT systems.
- Chapter 6 presents a design proposal for how the extended model can be implemented and an evaluation if the program NTE would serve as the basis for this implementation. It also presents a list of possible features to implement.
- Chapter 7 presents how the implementation of some of the proposed changes to the program was done. This is described both on a general level and more detailed for some parts of the implementation. This also presents an evaluation of the features that was implemented.
- Chapter 8 presents a conclusion to the results and lists the achievements, limitations and areas suitable for future work.
- Chapter 9 presents thanks to those who have given help or support during the work presented in this report.

1.5 Terminology

The following list describes the terminology that is used in this report. In many cases there exist several definitions to each of the notations used. The aim of this report is to use the terminology that is commonly accepted but in the field of risk analysis and risk management, the use of terminology is not homogeneous. In the cases where there exist several definitions, this report will use these that are in use in the Swedish Armed Forces, as much of the work presented in the report focuses on the methods and models from the Swedish Armed Forces.

- Accreditation decision is a form of decision to make certain that the IT systems of the Swedish Armed Forces handle information in a correct way [2]. It is taken early in the lifecycle of an IT system and is part of all stages in the IT lifecycle model.
- Certification decision is a form of decision that is used to support, concentrate, coordinate, and control the Swedish Armed Forces IT activities [2]. It can give the
1.5. Terminology

organization experience and be used when making a proposal for an update of the set of regulations. It can also be used for an early assessment if the IT system can be realized and shall be taken before all development.

- **Demands for approved security functions** is a set of security demands that an IT system within the Swedish Armed Forces shall fulfil [4]. Depending on the security class of the information in the system, different levels of security and numbers of security demands needs to be fulfilled. The security demands can be divided into access control, security logging, protection against exposing signals, protection against unauthorized wire-tapping, protection against trespassing, detection of trespassing, and protection against malicious code [4].

- **EASTER** is a plugin for the program NTE. It builds upon the code for another plugin, SANTA and is described in more detail in chapter 6.

- **Exposing signals** are according to the Swedish Armed Forces the electromagnetic radiation that surrounds all electric equipment or acoustic signals produced equipment or activities [5]. The electromagnetic radiation can be detected and analysed with special equipment that are connected to electrical cables or other hardware. The acoustic signals can be detected with microphones and then analysed manually or with the help of other tools.

- **Information security class** is a way to divide information in different security classes, depending on if it is secret or not. According to the Swedish Armed Forces, information can be classified as open or be part of one of four internationally defined security classes [4]. These are restricted, confidential, secret, and top secret.

- **Modus** or modus operandi is a notation used in the Swedish Armed Forces risk management model and is defined as the course of action for affecting an asset worth to protect [1].

- **NTE** or New Tool Environment is a tool prototype that has been developed as part of a master’s thesis done at FOI by Bengtsson & Brinck [3]. NTE implements the method XMASS in a graphical plugin called SANTA where it is possible to model networks and computer systems.

- **Social engineering** is according to McClure et al. a technique to deceive and persuade the users behind every security system to help or let an attacker through the security system [6].

- **Risk** is a term for which there exist several definitions, some are more common in everyday speech and other are more formal. According to Palm, risk in everyday speech can be seen as an unwanted event that could happen but does not need to, or as a probability that an event could happen [7]. A more formal definition of risk is according to both Palm and Holmgren, the combination of the probability of an unwanted event occurring and the consequence if it occurs [7, 8]. More definitions of risk can be found in Palms report, for example from statistics or decision theory. In this report, the notation of risk is used based on the formal definition as a combination of probability and consequence.

- **Risk analysis** can be defined in many ways but the one that is used by the Swedish Armed Forces is that risk analysis is the systematic use of information to identify and estimate risks [1].

- **Risk assessment** can be seen as way to put a value on a risk. This value can then be used to measure the risk against one or more criterions. The Federation of European Risk Management Associations have a more formal and specific definition of risk assessment as the "overall process of risk analysis and risk evaluation" [9].
– **Risk estimation** can be seen as the estimation of a risk. The Federation of European Risk Management Associations defines risk estimation as a more general assessment that usually is qualitative in nature [9].

– **Risk management** has many definitions but one of these that are used by the Swedish Armed Forces defines risk management as coordinated activities that can be used to aim and control an organization with the focus of risks [1].

– **Threat** is closely related to risk and can in many ways be the same thing but according to Palm, risk is more commonly used for accidents or disasters while threat is often used to describe antagonistic actions [7]. An example of an antagonistic threat is a programmer who creates a computer virus. The creation of the virus has a purpose compared to if the same programmer creates a security hole in a software by mistake. If a person would use this security hole, it would be a threat. The same is true if the programmer created the security hole on purpose. Some of the methods and models presented in this report make a difference between risk and threat while others do not.

– **Uncertainty** is a part of all types of risk analyses since the analysis is a way to predict future events. Uncertainty can then be seen as how large the probability is that this prediction does not happen. According to Ingvarsson & Roos, it can be divided into three types: uncertainty in the parameters, uncertainty in the model, and uncertainty if all risks in the system have been identified [10].

– **Vulnerability** can according to both Holmgren and Christiansson be seen as a collection of properties in a system that weakens it or lowers its ability to keep its functionality when exposed to a threat[8, 11]. The threat can come both from inside or outside the systems boundaries.

– **Vulnerability analysis** can based on the definition of vulnerability, be seen as a way to analyse the weaknesses or vulnerabilities in a system. The Swedish Emergency Management Agency makes a similar conclusion and sees vulnerability analysis as a way to systematic evaluate and estimate vulnerabilities [12].

– **XMASS** or eXtended Method for Assessment of System Security is a method for assessment of security in IT systems that has been developed by Hallberg et al. [13]. It is an extension and further development of the method MASS, developed by Andersson [14] and assess security in a system through security profiles and relations between entities in a graph model.

### 1.6 Abbreviations

The following abbreviations are used in the report.

– **GUI** - Graphical user interface
– **H Säk IT** - Manual for the Swedish Armed Forces Intelligence Service, information technology
– **H Säk Hot** - Manual for the Swedish Armed Forces Intelligence Service, threat estimation
– **IT** - Information Technology
– **KSF** - Demands for approved security functions
Chapter 2

Problem Description

In this chapter, the problem that should be solved during this master’s thesis is presented. This starts by presenting a problem statement that describes the task that should be performed and divides it into a list of six sub-tasks. These are then used for identifying five goals with this master’s thesis. Then the purpose and different methods that were used during the work presented in this report are described. Finally, the related work that exists in this field and was used as the foundation and the starting point for this report is presented.

2.1 Problem Statement

The task was to evaluate if and how two models used by the Swedish Armed Forces could be combined: the risk management model [1] and the IT lifecycle model [2]. More specifically it was if the risk management model could be used for the risk analysis in part P2a and P2b from the IT lifecycle model. This would include an analysis if the risk management model could be extended for IT. The program NTE should be evaluated to see if it would serve as the basis for an implementation of this combined and extended model. A list of functionality to extend NTE with or a list of functions to use from NTE should then be presented. Some of the functions on this list should then be implemented, either in NTE or in a new program. The task can be divided into the following list of sub-tasks:

1. Read related documentation about the subject from the Swedish Armed Forces [1, 2, 4] and FOI [15, 16].
2. Do a literature study of methods for threat, risk, and vulnerability analyses.
3. Evaluate if the general risk management model can be used for the risk analysis in part P2a and P2b of the IT lifecycle model.
   (a) Evaluate if and how the general risk management model can be extended for IT.
   (b) Evaluate what extensions that needs to be made.
4. Evaluate if NTE, or parts of NTE, should be used as basis for the implementation.
   (a) Investigate what support the Swedish Armed Forces is interested in.
5. Implement functionality for the five steps in the Swedish Armed Forces risk management model in the three categories:
   (a) Input of data
i. Establish basic values for the analysis
ii. Concretize and assess the threats
iii. Identify protection and assess vulnerabilities
iv. Assess the risks
v. Risk management decision with a plan for follow-up

(b) Analysis of data
(c) Visualization of data

6. Present a selling example of the tool prototypes potential, showing results that are not self explained if possible.

2.2 Goals

Based on the task that was presented in the problem statement, five goals with this master’s thesis were identified.

1. The first goal was to do a literature study on threat, risk, and vulnerability analyses.
2. The second goal was to evaluate if and how the Swedish Armed Forces risk management model could be combined with the Swedish Armed Forces IT lifecycle model and used for the risk analysis in step P2a and P2b. This should include an analysis if the risk management model could be extended for IT.
3. The third goal was to evaluate NTE and see if it could be used as the basis for an implementation of the combined and extended model. If NTE would be used as a basis for the implementation, a list of functionality to extend it should be presented. If NTE would not be used, a list of functionality to use from NTE, as well as a list of functionality required to implement the extended model should be presented.
4. The fourth goal was to choose some of the identified functionality and implement these in a software. Either in NTE or a new program.
5. The fifth goal was to present the potential of the tool prototype.

2.3 Purpose

Security is important for IT systems and the purpose for achieving the goals of this master’s thesis was to improve the threat, risk, and vulnerability analyses of the IT lifecycle model and the software that had been used. If the IT lifecycle model could benefit from the threat, risk and vulnerability analyses in the risk management model, the systems developed with this model would have the possibility to be more secure against risks. By improving the program and implementing more support for the model, the process of developing or testing new systems could be improved even further. The program could help to make the threat, risk and vulnerability analyses of the development more structured and simplify the process by adding support for input, analysis, and visualization of data.

2.4 Methods

In this section, the methods that were used to accomplish the work presented in this report are described. During the literature study, some related reports were studied. From this, more references were found and lead to an understanding of the field and the terminology.
2.5. Related work

When this had been accomplished, the information was compared so that differences and similarities between different models and methods could be found. During the combination of the two models from the Swedish Armed Forces, the differences between them was used in combination with the information that had been identified earlier. This lead to an understanding of which parts that needed to be changed in order to combine the models. When the risk management model was extended, the information that had been identified earlier was used again to extend the model for risks related to IT. During the design proposal, the foundation was the information, the combined and extended model, and some established guidelines in the area. This was then expressed as rough sketches for how the design would look. The sketches were then translated into text, compared with the guidelines and extended. Some parts of the design proposal went through this process several times. When the design was starting to be more stable, it was described more detailed as text in the report. During the implementation, the work started by following some principles for object-oriented programming, for example to create weak coupling between the classes and declare variable as restrictive as possible. The implementation started bottom-up as individual components were developed, tested and then combined into a more complete prototype. This was then evaluated based on principles of object-oriented programming and resulted in a set of design problems for a possible re-design.

2.5 Related work

Much of the focus in this report was the study of different models for threat, risk, and vulnerability analyses as well as work with the program NTE. The report by Palm presents a description of the Swedish Armed Forces risk management model, how it was developed and compares it with other models for threat, risk, and vulnerability analyses [7]. The literature study in this report used her report as the foundation for different models and further sources. Another report that lists and describes different methods for threat, risk, and vulnerability analyses is the report by Christiansson [11]. The reports by Bengtsson & Hallberg describe assessment aspects related to IT security in the Swedish Armed Forces [15] and the second step in the IT lifecycle model in more detail [16]. These reports and the two manuals from the Swedish Armed Forces, DIT04 [2] and H Säk IT [4] gives a good understanding of the IT lifecycle model and how it is used. The program NTE, the method XMASS that was implemented in it and the method MASS that served as the foundation for this are described in several reports. The first of these are the report by Andersson [14], that describes MASS and the software ROME2 that implemented this method. An extension of MASS was then described in a report by Hallberg et al. where XMASS was presented [13]. The report by Bengtsson & Brinck then presents how the program NTE was developed and how it implements the method XMASS as the plugin SANTA [3]. The report by Sundmark then presents an evaluation on the implementation of XMASS and some improvements to the calculations [17].
Chapter 3

Models and methods for threat, risk, and vulnerability analyses

This chapter describes different models and methods for threat, risk, and vulnerability analyses. First from the Swedish Armed Forces and then from other organizations. A comparison is then performed between the Swedish Armed Forces risk management model and the other models or methods presented in this report.

3.1 A brief introduction to threat, risk, and vulnerability analyses

Threat, risk, and vulnerability analyses are a way to analyse threats, risks, and vulnerabilities, for example in organizations or systems. This section gives an introduction to these analyses, the use of terminology and some properties and perspectives of the analyses.

Threat, risk, and vulnerability analyses are used to identify threats, risks, and vulnerabilities in general or in more specific cases. They can be used by an organization to identify potential threats and risks in the daily work or in more specific cases, like a certain project or threats and risks tied to a specific area. The models or methods for these analyses can be general and be used for many different cases or developed for a specific area. Many of the models presented in this chapter contain a number of well defined steps. These steps contains different types of analyses but in general the models start with a broader identification and then move on to threats, risks, and vulnerabilities. When these have been identified, the threats and risks are estimated and measures to reduce these are developed.

3.1.1 Use of terminology

The terminology used in the field of threat, risk, and vulnerability analysis is not homogeneous and many of the notations have more than one definition or are used in different ways. This section gives a brief description of how some of the more important notations in this report are used. This is not a complete description of this issue and many of the models and methods presented in this report describe and use the notations in different ways.

The notion of threat and risk are used in different ways by the models and methods presented in this report. Some do not make any difference between threat and risk and use the notions interchangeable, for example the model from the Swedish Rescue Services Agency.
Some make a clear difference between them, for example the model from the National Institute of Standards and Technology [19], while others do not use the notion of threat at all, for example the Swedish Road Administration [20]. Some models estimate the risk by first identifying threats, then vulnerabilities and use this to estimate the risk, for example the Swedish Armed Forces risk management model [1]. Others make no difference between threat and risk and identify these first, then identifies the vulnerabilities, for example the Swedish Emergency Management Agency [12]. Another use of the notion of threat is the difference between antagonistic threats and regular threats. An antagonistic threat is caused by a human actor, but it is the intention of a conscious action that separates if from a regular threat, not the human actor. An accident or a mistake that causes one, that is caused by a human actor but is not done as a conscious action is therefore not an antagonistic threat.

The notation of vulnerability is closely connected to threat and protection. One issue that comes from the definition that is used in this report is that when an asset is affected by a threat, a weakness in the protection for it is seen as a vulnerability. If the threat is removed, the same weakness remains in the protection but is no longer seen as a vulnerability.

Besides the definitions of risk analysis and risk management given in chapter 1, they can be used in other ways. According to Palm, a risk analysis can be accomplished by identifying existing risks or threats and then assess the probability that they will happen and the consequence if they do [7]. Holmgren describes another way of looking at risk analysis; it can be seen as a systematic use of available information to identify sources of risk for humans, properties and environments [8]. Risk management can according to Palm and Holmgren, be seen as a combination of risk analysis, risk estimation and the measures taken [7, 8]. Holmgren presents another more formal definition of risk management as a systematic application of management policy and processes to analyse, estimate and reduce risks [8]. Risk management can also be seen as a process that contains both the analysis and management of risks. The management of risks is then a smaller step in the larger process and is not the same as risk management.

3.1.2 Properties of threat, risk and vulnerability analyses

Two manuals from the Swedish Armed Forces: the risk management model and the manual for estimating antagonistic threats, describe the relation between risk analysis and science [1, 21]. In the risk management model, it is stressed that risk analysis is not a science and that the result is an estimation, not an objective truth [1]. This is also described by Palm, who writes that the values for probability and consequence are subjective estimations that are combined to a risk value that is even more uncertain than the original estimations [7].

In the manual for estimating antagonistic threats, it is described that risk analysis is an art with both scientific and intuitive elements [21]. Despite this, both manuals describe that a risk analysis can benefit from a scientific approach. If those that perform the analysis look and reflect around the available material in a critical way, work structured and document their work so it can be repeated, the analysis should rest on scientific principles [1, 21].

Another property of risk is the social perspective described by Nilsson [22]. He writes that the technical definition of risk misses several important aspects that can be hard to capture in a risk analysis. This can be the interactions between people, that people do not assess risks with equal weight for probability and consequence, and that differences between individuals are missed when data from large surveys are combined [22]. He also writes that other factors that are important are if the risk is voluntary, lack of experience from similar risks and if the effects caused by the risk is delayed in time [22]. According to Nilsson, the estimation of risk is not objective and is affected by the persons involved [22].
3.2 Models for analyses in the Swedish Armed Forces

This section describes threat, risk, and vulnerability analyses from the Swedish Armed Forces. This is done by describing two models: the risk management model and the IT lifecycle model. The risk management model is the newer of these two models, while the IT lifecycle model is older and has gone through several changes. The version described in this report is from 2004 [2]. The development of the risk management model began in 2005 after two Swedish soldiers were killed in Afghanistan. The model was finished in 2007 and was presented in a publication from the Swedish Armed Forces in 2009 [1]. This publication also contains a larger number of appendices that describe how it is used and how it can be expanded. The model is general but can be expanded with further appendices for more specific areas of use. In its current form, it is adopted for international military operations.

The notations of threat, risk, and vulnerability in the Swedish Armed Forces are defined more specific than in most models or methods described in this report as the notations of threat and risk are clearly separated. Threat is defined as a possible, unwanted event with negative consequences for the activity, risk as a combination of the probability that an event could happen and its consequence, and vulnerability as lack of protection of an asset exposed to threat [1]. Three older manuals from the Swedish Armed Forces define risk and vulnerability in a slightly different way, but this does not affect the models described in this report to any large degree, see DIT04, H Säk IT and H Säk Hot for further definitions [2, 4, 5]. A way of looking at the risk management process described with the notations used in the Swedish Armed Forces can be seen in Figure 3.1. This shows an iterative process where new threats are continuously introduced and forces the activity to adapt and change.

![Figure 3.1: The risk management process described with the terminology used in the Swedish Armed Forces risk management model](image-url)
3.2.1 The risk management model

The model is made up of five steps that are performed on different levels within the organization. Step one and five are according to the Swedish Armed Forces performed by the command and the commander in chief and if they do not take an active interest in the work, the whole model will fail [1]. Step two to four are done by a team from the staff and it is important that they meet, discusses and solves the task together [1]. The model is described below and is a summary of a more detailed publication from the Swedish Armed Forces [1]. In some steps there are further explanations from a report by Palm, as she was part of the team that developed the model [7].

Step 1 - Establish basic values for the analysis

In the first step, the basic values for the analysis should be established. This means that the task is defined and some basic questions are answered. This should at least contain the following questions:

- What shall be done?
- Who shall do it?
- Why should it be done?
- Where shall it be done?
- In what time shall it be done and when should it be finished?

When the task is defined, the assets that are worth to protect are identified. This can be for example personnel, material, information, or trust. After that, the different types of threats that should be analysed in step two are established and a scale of consequence in ten steps for the analysis is defined. The scale decides how the threats are estimated and are graded from (1) neglectable to (10) extremely serious.

Step 2 - Concretize and estimate the threats

The threats that have been decided upon shall now be broken down to unwanted events, occurrences, or modus for attack. In this report, these three notations are referred to as events. The threats shall be broken down to such a level that it is possible to identify what protection exists for the specific event and what vulnerabilities that exists. Each event is then analysed and its level of threat is estimated on a scale of five, from (1) no apparent threat to (5) very high threat. According to Palm, the threats are estimated from intention, capacity, and opportunity [7]. She writes that intention can be seen as the will, the motive and goal of an antagonist [7]. Capacity can be seen as the resources and ability while opportunity can be seen as when and where an antagonist can attack an asset [7]. The level of threat should not be seen as a probability that the threat can happen as this depends on others factors, but as properties in the environment where the asset worth to protect is.

Step 3 - Identify protections and estimate vulnerabilities

The protections that should be identified are these that correspond to the threats that have been identified in step 2. The protections can be active, passive, preventive, restoring or recovering. Regardless of the type of protection, the goal is to reduce or eliminate the threat by reducing the probability that the threat occurs or reducing the consequence if it does. The vulnerability is then estimated based on the level of protection that the asset has against one or more threats. The level of vulnerability is estimated on a scale of five,
from (1) no visible vulnerability to (5) very high vulnerability. According to Palm, the vulnerability is estimated by dividing the protection into security awareness, exposure, and resources [7]. Security awareness is the will to protect oneself, for example by behaviour or education. Resources are the equipment an organization has and exposure is where and how long someone or something is exposed to a threat.

Step 4 - Estimate the risk

The identification of threats, protections and vulnerabilities in step 2 and 3 serves as the foundation for the estimation of the risks. The combination of probability and consequence is in the model known as risk. The probability is estimated from the threat itself, behaviour, resources for protection, security awareness, and exposure. The probability is estimated on a static pre-defined scale of ten that is described in appendix 14 for the risk management model. The consequence is estimated from three criteria: if the threat penetrates the protections, the consequences for the asset and if the consequences can be reduced by some protective measure. The consequence is estimated on the scale of ten that was defined in the first step. Both the probability and consequence are then transferred to a risk matrix where the rows represent probabilities and the columns consequences. The elements in the matrix represent different risk values from no apparent risk to very high risk, which are separated by different colors as can be seen in Figure 3.2. The risk values in the matrix are not spread out symmetrically, as they are weighted towards consequences. This means that risks with high consequence but low probability gets a higher risk value than risks with low consequences but high probability.

![Figure 3.2: The risk matrix used to present risks, where the different colours represent a risk value from (1) to (5) (adapted from the Swedish Armed Forces [1]).](image)

Step 5 - Risk management decision and plan for follow-up

In this step, the command and the commander in chief decides if the risks that have been identified are acceptable, if some measures must be taken or if the decision should be sent to the commander in chief higher up in the hierarchy of the organization. This is done if the
measures required to deal with the risk cannot be accomplished with the current resources. With the decision, there could be a plan for follow-up. This could include how and when a new analysis should be done or how measures against the risks should be carried out. When measures have been decided upon, they should according to Palm be analysed in the model again to ensure that they do not lead to new risks that are higher than those that exist [7]. She also writes that this step is the one in the model that is least developed and that no methods exist that describe how the management of risks should be done in more detail [7].

Methods

The model describes what analyses that should be done in the different steps and also when, where and who that shall do them but not how they should be done. This is instead described in separate methods in the appendices to the model. The model can be extended with new methods for making threat, risk, and vulnerability analyses in different areas. A developed method must fulfil several demands. There must be a method for estimating the threats, a method for estimating the vulnerabilities and a method for estimating the risks. These methods are connected as the protection, vulnerability, probability, and consequence are connected to the selected threat. The methods should estimate this on a scale of five, have criterions for each level in the scale and be able to give account for how the result of the analysis was accomplished.

The tool

At present, the tool that is used to support the analysis is a spreadsheet in Microsoft Excel with pre-defined fields and a risk matrix. When the analysis is presented, this is shown on a large screen or a projector. If the analysis has generated a lot of data, it is hard to get a good overview of the result. To solve this problem, three proposals for how to present the information has been presented in Appendix 8 to the model.

1. The first proposal is to present a part of the spreadsheet and hide what is not relevant at the moment. In this way, it is possible to show what is interesting, for example assets and the threats that could affect them. Information on how this was identified and more details can be revealed if necessary. The problems with this proposal are that it is hard to show what causes the threats and if a single threat affects several assets. An example of this proposal can be seen in Figure 3.3.

2. The second proposal is to present the risk matrix in order to deal with the problems with the first proposal. In the risk matrix it is possible to view the different risks, how serious they are and see patterns among them. The drawbacks with this are that the matrix can become cluttered if there are many risks and that it is hard to see if a threat affects several assets. If the risks are presented separately for each asset, this problem is solved but the matrix can still become cluttered.

3. The third proposal is to present the threats geographically on a map. In this way, it is possible to see where there are heightened risks and how they are located in relation to each other. The drawbacks are that another analysis is required to present the information in this way and that much of the information that were presented in the first and second proposal cannot be presented in this one.
3.2. Models for analyses in the Swedish Armed Forces

<table>
<thead>
<tr>
<th>Threats</th>
<th>Assets</th>
<th>Identify protections</th>
<th>Estimate vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Preventive protections</td>
<td>Active protections</td>
</tr>
<tr>
<td>Concrete unwanted event Occurance</td>
<td>Threat level</td>
<td>Protection A</td>
<td>Protection B</td>
</tr>
<tr>
<td>Modus for attack</td>
<td>Threat A</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Threat B</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 3.3: An example of how the tool used for the risk management looks like at present (adapted from Palm [7]).

3.2.2 The IT lifecycle model

The Swedish Armed Forces lifecycle model is described in a document from the Swedish Armed Forces and the model describes the whole lifecycle for an IT system, from the development of the requirements of the system until the system is phased out [2]. The IT lifecycle consists of seven steps and seven decision points, where the steps serve as the foundations of the decisions about the system. The first four decision points are certification decisions, the fifth and sixth are accreditation decisions and the last is a decision that is based on the previous accreditation decisions. For more information about the different types of decisions, see section 1.5.

A general description of the model

The model is presented below in seven steps based on the description in the document DIT04 from the Swedish Armed Forces [2]. The seven steps of the model can be seen in Figure 3.4.

Figure 3.4: The seven steps and the structure of the Swedish Armed Forces IT lifecycle model (adapted from the Swedish Armed Forces [2])
- **Step P1 - Preparation of requirements**
  In the first step, the requirements that an IT system should satisfy should be identified and analysed. If there are identified requirements, this step is the basis for the decision if the development process should start. An accreditation process starts in the first step, this continues throughout the whole IT lifecycle model and leads to several accreditation decisions. In the first decision point, the result of the first step is used as the foundation to the decision if the development process should start. In this decision, the object that shall be certified is classed as a system, a function, a service, or an activity. A foundation for the decision points later in the model is also established. An example of this is a communication system, where the object is a system and the foundation for the decision points are evaluations of existing systems.

- **Step P2a and P2b - Generate and assess a concept**
  In the second step, it should be evaluated if the work with the system should continue and in which way. The first that is done is to make an analysis of the current activity and remove those requirements that cannot be realized, while those that remain are established as activity demands. These are the demands that exist for the IT system and acts as the foundation for how it will be developed. Then an analysis is made to see if the new activity can result in changes to the organization or the way work is done in the organization. Finally, a concept to how the IT system can be realized is generated and then assessed from several perspectives, for example personnel, technique, and economy. A more detailed description of this process and more perspectives can be found in DIT04 [2]. In the second decision point, based on the work that has been done, it is decided if the work shall continue and if the third step should be done.

- **Step P3 - Define**
  In the third step, the activity demands that was established in step P2a serves as the basis for a system specification and system architecture. This results in a proposal to how the system can be realized and is done on a more detailed level than the concept in step P2a. In the third decision point, it is decided if the system should be developed or not. If the system is to be developed, the system specification, the system architecture, and the proposal on how it shall be realized should also be approved.

- **Step P4 - Acquire - Design**
  In the fourth step, the system is realized based upon what was decided in step P3. This is done by buying an existing system, reusing an existing system, adapting a system, or developing a new system. It can also be a combination of any of these. In the fourth decision point, the decision is based on the finished system and it should be decided upon if it should be integrated with the existing system, if there is one.

- **Step P5 - Acquire - Integrate**
  In the fifth step, the system is integrated, reviewed and integration tested. In the fifth decision point, it is decided if the system should be used or phased out. Three accreditation decisions are used as the foundation for the decision: a central accreditation decision, an accreditation decisions for system security, and one for basic conditions during operation. If the system is accepted, the decision could include new additions or changes to it.

- **Step P6 - Use or Phase out**
  In the sixth step, the system is put to use or phased out. If the system is put to use, accreditation decisions must be taken in the sixth decision point before the system can be used. These specify details for how the system should be put in use and how it should be used. If the system is phased out, there is no need for a sixth decision point and the model stops here.
### 3.2. Models for analyses in the Swedish Armed Forces

- **Step P7 - Usage and maintenance**
  In the seventh step, it is the decisions in step B5, step B6 and the regulations of the Swedish Armed Forces that governs how the system can be used. New requirements for the activity can be generated during use and leads to the seventh decision point. The seventh decision point is placed inside the last step and new decisions are made continuously while the system is in use. If these decisions are within the decisions in step B5 and B6, they can be performed; otherwise the requirements found during operation can lead to a new preparation of requirements and the model continues with another iteration based on the existing system.

**An in-depth description of step P2 in the model**

This section gives a more in-depth description of the two steps P2a and P2b in the model, which are called step P2 from now on. In a report from FOI by Bengtsson & Hallberg, the step P2 in the model is described further [16]. They divide P2 into several parts, of which the following are the most central: analysis of the activity, security analysis, constitutional analysis, preliminary system description, as well as threat, risk, and vulnerability analyses [16]. These parts and other steps that make up P2 are shown in figure 3.5 and are then presented in more detail.

Figure 3.5: A more detailed description of step P2 in the model that shows the different parts that make up this step (adapted from Bengtsson & Hallberg [16]).

Bengtsson & Hallberg describe the analysis of the activity as a way to analyse the existing or future activity and establish *information security classes* [16]. The classifying of information is done by estimating which security class the information in or around the system belongs to and creates a foundation for the security analysis [16]. The following security analysis shall then according to Bengtsson & Hallberg estimate if the information that the IT system handles should be counted as secret and perform an estimation of the possible consequence that can occur [16]. They describe that the estimation of the consequence shows what happens if the information is made available [16]. Examples of this could be if the information is given out to the public or another states intelligence services. After the security analysis, a constitutional analysis should be performed. This should according to Bengtsson & Hallberg identify which laws, regulations, and rules that affect the IT system [16]. They also write that the constitutional analysis results in a number of constitutional
demands that together with analysis of the activity and the demands from the KSF results in a set of security demands [16]. For a further explanation of the KSF; the Demands for approved security functions, see section 1.5. The analysis of the activity might also lead to a preliminary system description, this is according to Bengtsson & Hallberg decided in step P1 [16]. The preliminary system description is used for certain IT systems, can give a more clear view of the system and is performed before the threat, risk, and vulnerability analyses.

The final steps before the decision point are the threat, risk, and vulnerability analyses. These should according to Bengtsson & Hallberg be done separately, but are often done as a single analysis [16]. They describe that the threat analysis identifies the threats that exist, that the risk analysis identifies the risks that exist from the given threats and that the vulnerability analysis identifies the vulnerabilities that exist based on the security demands that have been formed from the demands in the KSF, the information security goals from the analysis of the activity, the security analysis, and the constitutional analysis [16].

The description of step P2 that is presented in Bengtsson & Hallbergs report [16] is more detailed than it is described in the documents from the Swedish Armed Forces, DIT04 and H Säk IT [2, 4]. According to Hallberg, this is because their report was based on both the documents above and interviews with personnel within the Swedish Armed Forces [23]. While the description in their report is more extensive, it does not take all parts of step P2 in account. The parts that are left out of their report are if the work should continue in step P3, the removal of requirements that cannot be realized, the analysis of changes to the organization and the assessment of the concept.

3.3 Models for analyses in other areas

This section first describes how models and methods for threat, risk, and vulnerability analyses can be divided into different classes and then presents a number of models and methods for threat, risk, and vulnerability analyses. Most of the models presented are from different government authorities, some are more general and others are more focused on IT security. Most of them are described in manuals or reports from the different governmental authorities that developed them, but some are described, in part or in whole based on the reports from Palm [7] and Christiansson [11]. This is done to emphasize things that Palm and Christiansson describe differently from the governmental authorities reports, as a complement to the reports, or because there are no first hand sources to these models.

Since the reports from the Swedish Emergency Management Agency [12] and the Swedish Rescue Services Agency [18] were published, the Swedish Emergency Management Agency, the Swedish Rescue Services Agency, and the Swedish National Board of Psychological Defence have been combined into a single government authority, the Swedish Civil Contingencies Agency. Because the manuals that describe these models were published before that, the models from the Swedish Emergency Management Agency and the Swedish Rescue Services Agency are presented under the agencies older names as well.

Classification of models and methods

Models and methods for risk analyses can be classified and divided into different classes depending if they are qualitative or quantitative. According to Nilsson, qualitative models and methods rank, compare and describe risks in words, like high or low [22]. Examples of these are a coarse or preliminary analysis, checklists, and some types of risk matrices. According to Nystedt, quantitative models and methods describe the risks in quantitative terms, for example the probability of an unwanted event and presents the result as combination of
probability and consequence [24]. Examples of these are fault tree analysis and reliability theory [24]. Between the qualitative and quantitative models and methods, there are according to Nilsson, *semi-quantitative models and methods* [22]. They are more detailed than the qualitative and have some numerical measurements for probabilities and consequences. An example of these is a risk matrix with numerical values on the axis instead of text.

It can be hard to know which method to choose for an analysis, but according to Ingvarsson & Roos, the selection can be simplified if the one who orders the analysis states why it should be done [10]. They also write that it is better to choose a more coarse qualitative method early in the analysis and then switch to a quantitative method [10]. The drawback with quantitative methods is according to Ingvarsson & Roos that they need numerical data and if this is lacking then it is better to choose a quantitative method for the analysis [10].

The methods presented in this report belong to both of these groups but most of them are found among the semi-quantitative methods. How they are grouped can be seen in Figure 3.6. The exact placement of the models on the axis is an estimation of how they relate to each other and some models are so similar that they should be much closer to each other than shown in Figure 3.6.

Figure 3.6: The different methods presented in this report shown along an axis from qualitative to quantitative methods (based and adapted from the classification in the reports by Nilsson and Nystedt [22, 24].

### 3.3.1 Swedish Emergency Management Agency

The model from the Swedish Emergency Management Agency consists of five steps and is according to them guidance for government authorities [12]. The description of the five steps below are based on their manual for risk and vulnerability analyses [12].

**Step 1 - The authorities role and area of responsibility**

The first step in the model deals with identifying the type of risk management that shall be done and the areas where the authority is responsible for the risk management. When
choosing the type of the risk management, both internal and external threats that have a low probability but have large consequences should be considered. When deciding the area of responsibility, one important criterion is if there is activity that is important for the society within the proposed area of responsibility.

**Step 2 - Identification of threats and risks**

In the second step, the threats and risks that exist are identified. The model presents four proposals for how to do this. The first is supervision and control, if the authority performs this, they will get a natural basis for an analysis. The second is by learning from accidents or other events that have happened, both those that affected the authority itself and other authorities or organizations. The third is exercise activities and simulation models, from which the authority can learn how to improve risk and vulnerability analyses as well as test situations that have been analysed earlier. The fourth is by mapping activities that are important for society and the critical dependencies that exist. This is important since a threat or risk can affect several activities. It is also good to identify threats and risks within the area of responsibility that other authorities should handle or threats and risks outside of the area of responsibility that can affect it.

**Step 3 - Assessment of threats and risks**

In the third step the probability and consequence of threats and risks are estimated. Probabilities are estimated either by quantitative means like empirical methods and statistics, or qualitative ones like subjective estimations and expert opinions. Consequences are seen as direct or indirect negative effects that can happen. They can be estimated through established classification systems or more comprehensive goals like people's life and health, basic values, and the stability in society. There are also other goals like for example the environment or the economy. When the probability and consequence for the different threats and risks have been estimated, they are ranked in different classes in a risk matrix. The Swedish Emergency Management Agency also describes that it is better to focus on the consequences of events and how to handle these rather than the probability of them happening.

**Step 4 - Estimation of capacity and analysis of vulnerability**

In the fourth step, the authorities capacity to manage threats or risks and how vulnerable it is to these are estimated. In order to do this, the capacity to handle the consequences of an extraordinary event is estimated. The capacity can be divided into three parts: crisis management capacity, operative capacity, and the capacity to withstand serious disturbance to activities that are important to society. The capacity should be estimated on a scale of four and depending on the foundation it is based upon, it is given a different validity. An event that occurred recently is given a very high validity while analytic material without empiric data is given a low validity. If the capacity is estimated to be low, the authority is considered to be vulnerable.

**Step 5 - Need of measures and presentation of the result**

The last step deals with the presentation of the result, the measures that should be performed and a plan for financing these measures. The result should include not only a presentation of the analysis but also the measures that have been proposed and the possible demand for further measures. The authority that has done the analysis should then present the result
3.3. Models for analyses in other areas

The ability to present the result of the analysis could be as important as the analysis itself. If the result and presentation are not clear and easy to understand, the work done during the analysis is wasted.

### 3.3.2 Swedish Rescue Services Agency

The Swedish Rescue Services Agency has developed a model for risk management, which describes the risk management process together with several methods for analyses [18]. They also describe that the comprehensive goal with risk analyses are to illuminate where and how accidents, incidents and interruptions can occur [18]. This is then used as basis for risk assessment and risk reducing measures. Their model is based around five steps which are described below. Unless noted, the description of the model is based on the Swedish Rescue Services Agencies manual [18].

**Step 1 - Goals and limitations**

In the first step, the goals and limitations of the risk management are specified. Without clear goals, it is hard to specify the purpose of the analysis, how detailed it should be and decide what limitations that exist for the analysis. During this step, those that perform the analysis should also start to think about the criterions that will be used for the risk assessment later in the process.

**Step 2 - Identify**

In the second step, the risks are identified. This is an important element of the analysis as risks that are not identified are not analysed and this can lead to that the need for measures are underestimated. The primary goals with the risk identification are to identify all relevant risks and to make use of knowledge from earlier experiences in different areas. The secondary goals are that the process should be well documented, structured, and effective.

**Step 3 - Analysis**

In the third step, the risk level for each risk is set by estimating the probability that it occurs and the consequence if it does. The probability can be calculated from empirical estimations, logical systems, or expert estimations. Empirical estimations build on statistics; logical systems breaks down events into sub events and expert estimations can complement the other estimations or be the only available means of estimate the probability. The consequences can be estimated by prediction of the possible damages and consequences that can happen. This can be done using both quantitative and qualitative methods. The result of the risk analysis is then presented in one of two ways. One is to focus on the consequences of well defined accidents and the other is to focus on the result, presented as a risk matrix.

**Step 4 - Assessment**

In the fourth step, the risks are compared against criterions to evaluate how serious they are. In the earlier steps, the risks have been identified and analysed from an engineering viewpoint, but are now evaluated from other assessment aspects. The estimation of the risks could for example be transferred from numerical values, like statistics, to other estimations like low risk. Risk criterions can provide a guideline to this and are based on some principles. Four of these are listed in the manual and these are the principle of reasonableness,
the principle of proportionality, the principle of distribution, and the principle of avoiding disasters. The different principles are also described in a report by Nystedt, but this report does not mention the model from the Swedish Rescue Services Agency [24].

1. The principle of reasonableness is that an activity should not contain risks that can be avoided or lessened with reasonable means.
2. The principle of proportionality is that the risks of an activity should not be unproportionally larger than the benefit it gives.
3. The principle of distribution is that the risks should be distributed within the society in relation to the benefit an activity brings.
4. The principle of avoiding disasters is that risks should be realised as accidents rather than large catastrophes.

Step 5 - Security measures

In the last step, measures to reduce or eliminate the threats and risks are identified. The measures can reduce or eliminate the threats or risks by both reducing the probability that they happen or reduce the consequence of them. When the measures have been identified, it should be decided which measures that should be performed.

3.3.3 Legal, Financial and Administrative Services Agency

The model from the Legal, Financial and Administrative Services Agency is focused on risks from an economic viewpoint. According to Palm, the model is rather general and can be used for other risks as well [7]. The model has five steps and is presented on the agencies web page [25]. The agency also provides tools and support for governmental agencies with risk management on their web page [25]. Based on their description, the five steps of the model are presented below.

Step 1 - Risk identification

In the first step, the risks that exist should be identified by first identifying the activity they can affect. This is important as how an activity or organization works in theory might be different from how it looks in reality. Examples of this are that things can be unregulated, there can be informal information channels or informal key personnel. In order to identify the risks, the activity must be described like it looks in reality, not how it should look in theory. When this is done, it is possible to identify the risks that can affect it.

Step 2 - Risk assessment

In the second step, the risks should be assessed from the perspectives of probability, consequence, economic significance, and priority. This is done through four questions: how often a risk happens, what consequences it brings, what cost it leads to and what significance it has in relation to other risks. Probability and consequence are estimated from the authorities own knowledge and are estimated on a scale with three to five steps. Priorities of risks are used for assessment of the risks in relation to each other and sort them in groups. How the assessment of economic significance is done is not described further by the model.
Step 3 - Risk management

In the third step, a decision is made on how to handle the risks. This can be done in several ways. Risks that have a high probability and consequence should be eliminated. Risks with either a high probability, a high consequence, or a medium probability and consequence should be lessened with preventive or recovering measures, transferred to someone else, for example an insurance company, or kept as they are. Risks with a medium probability or medium consequence should be kept as they are, while risks with low probability and consequence are neglectable.

Step 4 - Carry out the decision

In the fourth step, the measures are carried out and it is made sure that the right measure was carried out. It is also noted that measures that changes the routines within an organization are hard to carry out but can be very effective and that a high risk awareness is an important factor for good risk management.

Step 5 - Follow-up

In the last step, all the other steps in the model are evaluated. This should be done continuously and a guiding principle is if the organization does things right or does the right things. A follow-up should always be done, as new risks will appear and the probability and consequences for risks will change over time. To evaluate if the decision was carried out correctly, some questions are presented:

- **Execution**: Has the decision been carried out as it was meant to be?
- **Result**: Did the decision get the desired effect?
- **Adapt**: Is the effect still desired?
- **Adjustment**: Is there a need to change the decision?
- **Learning**: Has the organization learned anything about risks and how they are managed?

3.3.4 County Administrative Board of Stockholm

The model from the county administrative board is used for risk and vulnerability analyses on both local and regional level within municipalities and the county administrative board of Stockholm. The model can in short be described as a parallel process where the municipalities and the county administrative board perform risk and vulnerability analyses, report the result, take measures against the risks and performs a follow-up. A summary of the process can be acquired from the county administrative board of Stockholm’s web site [26].

The model can be divided into four steps: preparation, implementation, reporting and risk management and can be seen in Figure 3.7. The descriptions of the four steps of the model that are presented below are based on the information available from the county administrative board of Stockholm [27]. The tool IBERO is based on other sources, these are presented in the text below.

Preparation

During the first step, both the municipalities and the county administrative board select a group to do the analysis. The members of the group should have different competences and the size of the group can change depending on the tasks during the analyses.
Implementation

The implementation step starts with the identification of the assets that are worth to protect. Based on the identified assets, the municipalities perform a coarse analysis. Coarse analysis is a shorter type of analysis that is given further explanation in section 3.3.11. The county administrative board then put together a list of events that are sent to the municipalities. They are combined with the analysed events and the three most extraordinary ones are selected out and analysed further with the help of a computer based tool, IBERO.

IBERO is according to the county administrative board used for assessment of preparedness by both individual and multiple actors, for listing the existing risks and generate reports [28]. According to the county administrative board, IBERO analyses events by breaking them down into smaller parts, these parts are then analysed before they are put together again to identify if there exists any dependencies between them [28]. Further information about IBERO can be acquired from the county administrative board of Stockholm [29] and the tool itself together with templates can be downloaded from them as well [30].

The result of the analysis in IBERO is sent to the county administrative board and is then analysed further with the use of IBERO. This leads to a report on risk groups and then an analysis of risks and vulnerabilities by the county administrative board. This takes place after the municipalities has finished their analyses.

Reporting

The result of the municipalities analyses with IBERO is used as basis for a report. With the help of this, the municipalities can put together reports of the risks, vulnerabilities and measures that need to be taken. The county administrative board makes its own report based on the result of their analysis, but also takes the result from the municipalities in account. In contrast to the municipalities, this report is focused on the whole county.

Risk management

The municipalities reports are then sent to the different municipality board to be used as a basis for the decision of measures and follow-up for the analysis. The measure taken in the municipalities and the result of the county administrative boards analysis influences what measures the county administrative board takes for the whole county. These measures are then carried out, supervised, and followed-up by both the municipalities and the county administrative board.
3.3. Models for analyses in other areas

3.3.5 Swedish Security Service

The security analysis model from the Swedish Security Service makes a distinction between threat and risk and is therefore similar to the models from the Swedish Armed Forces. These models focus on antagonistic threats, which many of the others do not. The description of the four steps of the Swedish Security Services model below is based on a guidance from the Swedish Security Service [31].

**Step 1 - Identify assets that are worth to protect**

In the first step of the model, the assets that are worth to protect should be identified. This can be done by identifying the persons that have a security classed position as this might give guidance to important activity and those that handles it. Information that is related to the state’s security, protection against terrorism or important parts of the infrastructure should also be looked through. There are also many pieces of open information that can reveal new secret information when put together.

**Step 2 - Identify threats against the assets**

In the second step, the threats that exist are identified. The threats that the Swedish Security Service consider to be security threats are espionage, sabotage, criminality related to security, infiltration, and technology as a threat. They are all described in more detail in their guidance.

**Step 3 - Analyse risk and vulnerabilities**

In the third step, the risks that the threats pose and the vulnerabilities that exist are analysed on the basis of probability and consequence. The risks are estimated on a scale of four and a risk level is developed from the combination of probability and consequence.

**Step 4 - Create a plan for measurements**

In the last step, the risks are estimated, prioritized and measures are taken. Measures can be taken immediately, in a short period of time, about one year, or in a long period of time, up to three years. A risk with a high threat level should be fixed immediately, a risk with a medium threat level should be fixed in a short period of time and a risk with a low threat level should be fixed in a long period of time.

3.3.6 Swedish Road Administration

The risk management model described in this report is the model described in a report from the Swedish Road Administration [20]. This model is a more applied version of their more general scenario model and is based on a specific interest or asset. From this, dangers, risk factors and possible consequences are identified. The risk management model consists of six steps and is presented below, based on the description in the Swedish Road Administrations report [20].

**Step 1 - Identify which assets or objects that can be damaged**

In the first step, the assets that exist are identified. Each identified asset is connected to a type, that specifies which one of five pre-defined categories it belongs to. Each asset might
also be connected to a special place in a geographical map, a so called object. The objects that are interesting for the Swedish Road Administration are those that contain large values.

**Step 2 - Identify which dangers that can damage the assets or objects**

In the second step, the dangers that can damage the assets or objects are identified. The dangers can be objects in the road system, objects in the surroundings or objects in the road system that are caused by events in the surroundings. The dangers that are essential are sorted out and analysed further, but it should be documented why the rest was ignored.

**Step 3 - Identify the risk factors that can cause a danger**

In the third step, the risk factors that can cause a danger are identified and estimations of the probability that they happen and the development of the event caused by the risk are done. The estimation can build upon statistics, but in many cases it needs to be estimated by other, less reliable means.

**Step 4 - Describe the extent and the consequence of the damage for each category of assets when exposed to danger**

In the fourth step, the extent of the damage to the assets, both for the expected case and the worst possible damage that can happen are described. The consequences for each of the categories of assets are also summarized.

**Step 5 - Describe the total risk level**

In the fifth step, the total risk level is determined by presenting the information in a risk matrix together with the probability and consequence classes for each category of assets. The probability and consequence classes are presented on a scale of five, from (1) to (5).

**Step 6 - Determine possible risk reducing measures**

In the final step, the risk reducing measures that should be taken are determined. What determines which measures that are taken depends on the cost and the benefit of each of them. To do this analysis, the Swedish Road Administration has another model that is used for weighting costs against benefits for different measures [32]. This model is given a short description by Palm and she writes that the model determines this by weighting the economical income and benefit for society with the economical cost [7]. This means that in order to perform a measure, the benefit of it should exceed the cost. The benefit can, for example be that a specific road section becomes safer, which results in fewer deaths or injuries. The cost is what the society in general can accept to pay for this benefit.

The results from the analysis are presented on special forms and risks are presented in a risk matrix. The Swedish Road Administration are also developing a way to view dangers geographically using GIS technology. In this form, it is possible to see where a specific danger is, the geographical relation to other dangers, and get more information about each danger by clicking on it on the map.

### 3.3.7 U.S. Department of Energy, Office of Energy Assurance

The vulnerability assessment method is described in detail in a report from the Office of Energy in the U.S. Department of Energy and focuses on qualified antagonistic threats [33].
3.3. Models for analyses in other areas

When compared with the terminology used in this report, it is as Christiansson writes, a method for risk analysis with elements of vulnerability analyses [11]. The model is very extensive, which according to Christiansson, makes it both expensive, time consuming and that the result quickly will be out of date [11]. He also describes that extensive models have some advantages as well, where one is that it is possible to skip some parts of the model [11]. If a step in the model is skipped, it is clear what is not covered in the analysis, which can be very important to know as well. Another variant that he describes is to perform a few extensive analyses and then make follow-ups on these with less extensive ones [11].

The analysis contains three parts: a pre-vulnerability analysis, a vulnerability analysis and a post-vulnerability analysis. The three steps are described below, based on the report by the Office of Energy in the U.S. Department of Energy [33]. In some steps, there are further explanations from the report by Christiansson [11].

Pre-vulnerability analysis

The first phase contains several activities that are important for the success of the analysis and is divided into three steps:

1. **Define objectives and Scope of Assessment**
   In the first step, the scope of the assessment and the assessment objectives are defined. This can be many things, for example to find all critical vulnerabilities and develop responses or measures to them.

2. **Establish Information Protection Procedures**
   In many cases, an assessment makes use of external competence. It is therefore important to define policies on how to handle sensitive data that is collected during the assessment.

3. **Identify and Rank Critical Assets**
   In the third step, all the critical assets should be identified and ranked. This might include vital systems, facilities, processes, and information. This is done to focus the assessment and support the risk analysis.

Vulnerability analysis

The second phase contains the main part of the analysis and is divided into ten steps:

1. **Analyse Network Architecture**
   In the first step, the topology of the networks, the protocols, policies and procedures are analysed. This is done with the help of documentation, interviews with personnel, and by visiting the related facilities.

2. **Assess Threat Environment**
   In the second step, the threat environment, the different antagonistic threats and how they make use of existing vulnerabilities are analysed. The analysis is performed through cooperation with local law enforcement agencies and security personnel.

3. **Conduct Penetration Testing**
   In the third step, penetration testing of the system is performed. Penetration testing can find weaknesses in the network and identify connections to other networks. According to Christiansson, this is done through both technical methods like port scanning and penetration of firewalls, as well as social methods like social engineering [11]. For further descriptions of social engineering, see section 4.2.2.
4. Assess Physical Security
In the fourth step, the physical security of the related facilities are analysed. This can be for example fences, locks, alarm systems, energy sources, or communication systems. The analysis should focus on those facilities that have been marked as critical and those that have been ranked as most important.

5. Conduct Physical Asset Analysis
In the fifth step, physical assets are analysed for vulnerabilities. Earlier incidents can be analysed further as well as trends in maintenance, investments or field personnel.

6. Assess Operations Security
In the sixth step, the operations security is assessed. Operations security is the process of making sure that information about capabilities and intentions does not reach a potential antagonist. This is done by identifying, controlling and classifying information.

7. Examine Policies and Procedures
In the seventh step, the organizations policies and procedures are analysed. Policies are used to give an understanding on how an organization protects its assets and provide a clear guidance on how to achieve this. An analysis of policies should focus on if they address key factors in security, are implemented in an effective way, conform to established standards, and give a clear guidance. They should also define and communicate roles, responsibilities, authorities and accountabilities.

8. Conduct Impact Analysis
In the eighth step, an analysis is done of what the consequences would be if someone would get unauthorized access to the organizations facilities or assets.

9. Assess Infrastructure Interdependencies
In the ninth step, the infrastructure dependencies between the organizations facilities and other infrastructures are analysed. This can be for example telephone communications, water or financial systems.

10. Conduct Risk Characterization
In the tenth and last step, recommendations for several task areas are prioritized. This is done using the foundation of threats, vulnerabilities and consequences that have been identified.

Post-vulnerability analysis
The third phase consists of prioritizing recommendations, developing an action plan, capture lessons learned, and conducting training. It is divided into four steps:

1. Prioritize Recommendations
In the first step, the recommendations are prioritized based on their cost. Recommendations that have a low cost or provide cost savings are selected out since they are easier to perform.

2. Develop Action Plan
In the second step, an action plan is developed. It contains a time-line, the team who shall do the work, a budget and how to implement the proposed recommendations.

3. Capture Lessons Learned and Best Practices
In the third step, lessons learned from the process as well as how the process can be improved are captured and documented.

4. Conduct Training
In the fourth and last step, training, workshops and other activities are conducted.
3.3.8 British Telecom

According to Christiansson, British Telecom has two programs for security, one for antagonists with low capacity, the regular and commercial IT security function and one for antagonists with high capacity, the information security program [11]. Only the information security program is presented in this report and is based on the report by Christiansson [11].

According to Christiansson, the purpose with the program is to protect critical assets against qualified electronic attacks from attackers with both motive and capacity to perform them [11]. It shall according to Christiansson be used for attacks that have some of the properties that are presented below [11].

- Attacks from attackers with high capacity
- Attacks that have serious consequences
- Attacks where historical data is lacking
- Where there are no indications for investments of protections
- Where protections for the assets have been established in cooperation with several government agencies

The information security program contains three parts: an activity model, a vulnerability model and an antagonist model. The activity model is used for identifying critical assets and processes together with the goals that are interesting for a potential attacker. The vulnerability model is used to identify vulnerabilities by breaking down assets and processes in smaller parts as well as conducting penetration testing. The antagonist model is used for assessment of the threats that exist by estimating the probability that they will happen and analyse a potential attacker’s motivation, opportunity, and capacity. The work of modelling an attacker is often based on very limited information but can benefit from information about earlier attacks and their effects on real systems. The antagonists are divided into several classes with different initial values and technical capacity. The classes can then be detailed further if it is necessary. The classes that exist are criminals, competitors, states, terrorists, hackers, and dissatisfied employees.

3.3.9 National Institute of Standards and Technology

The National Institute of Standards and Technology has developed a model for risk management that is described in a report by Stoneburner et al. [19]. The model is very detailed and focuses on IT systems. It contains three parts; the first is a risk analysis in nine steps, the second focuses on how risks shall be prioritized and how the consequences can be lessened while the third deals with how the risk management can be evaluated. In this report only the first part is described as it is this part that is most related to the other models described in this chapter. The nine steps of the model that are described below are all based on the detailed description in the report by Stoneburner et al. [19].

Step 1 - System Characterization

In the first step, the system, resources and information concerning the system are defined. In order to perform the analysis, information related to the system is collected from a number of sources including hardware, software, users, and environment. The information can be collected through forms, interviews, documentation, and software. This can be done both for systems that are already implemented or are at the planning stage.
Step 2 - Threat identification
In the second step, the threats that exist are identified. The model sees threats as vulnerabilities that are exploited, either intentionally or unintentionally and the threats can be natural, human or depend on the environment around the system. The report from Stoneburner et al. also contains further examples of conceivable threats and where to find more information about these [19].

Step 3 - Vulnerability identification
In the third step, the vulnerabilities in the system that can be exploited by the potential threats are identified. They are identified through the use of three methods: identifying sources of vulnerabilities, testing of the system security, and the creation of checklists. The identification of vulnerabilities are also different depending on if the system is in use, is being developed or has not been designed yet.

Step 4 - Control analysis
In the fourth step, the controls that minimise or eliminate the threats that exists against the system are analysed. The model makes a distinction between technical and non-technical controls, both of these can also be broken down into preventive or detective controls. Preventive controls ensure that threats do not happen and the detective controls gives a warning when they do.

Step 5 - Likelihood Determination
In the fifth step, the probability that vulnerabilities are exploited is estimated. It is estimated based on the motivation and capacity of the source of the threat, the nature of the vulnerability and the existence of controls. The probability is estimated in one of three levels: low, medium or high.

Step 6 - Impact analysis
In the sixth step, the consequence that happens when a vulnerability is exploited is estimated. In order to do this, the sensitive data concerning the system must be identified. When this is done, the model describes three events that can damage the data and the system: loss of integrity, loss of availability, and loss of trust. The consequence is then estimated in three levels, low, medium or high.

Step 7 - Risk Determination
In the seventh step, the risk level of the system is determined. The determination is done as a combination of the probability that a vulnerability is exploited, the consequences if this happens, and what controls that exist to lessen the consequences. The result of the determination is then presented in a risk matrix.

Step 8 - Control Recommendations
In the eighth step, a recommendation of which controls that shall be used to take down the risks against the IT system to an acceptable level is presented. In most cases, there exist many different types of controls and in this step they are weighted against each other based on the cost versus benefits for each of them.
3.3. Models for analyses in other areas

Step 9 - Results Documentation

In the ninth and final step, the result of the analysis is documented in a report. The result should be presented in a systematic and analytic way so those that have ordered the analysis can make use of the information.

3.3.10 Attack graphs

An attack graph is according to Ingols et al. a way to represent how an attacker does to attack a network or a computer system by representing the stages in the attack as a graph [34]. A more detailed description can be found in a survey from Lippmann & Ingols [35]. They write that an attack graph is used to determine if the goal state or goal states can be reached from any of the initial states [35]. According to them, the nodes and arcs represents actions taken and the network state is the result of those actions [35]. They also write that the full attack graph then represents all possible sequences of actions that can be taken to reach the goal [35].

A survey of the field

There exist many papers and reports about attack graphs, Lippmann & Ingols have made a survey about 18 of these papers [35]. Their survey presents and compares papers in the area as it was in the year of 2005 [35]. The papers in the survey can be divided into three categories based on their goals. They write that the first category are those that constructs attack graphs to analyse network security, the second are those that present a formal language to describe attack graphs and the third are those that use attack graphs to group alerts in intrusion detection system together [35]. The purpose of papers in the first and second category is rather straight forward, but the third category needs a bit more explanation. The purpose of papers in the third category are to combine the alerts from the intrusion detection system with an attack graph, predict that an attack is underway based on the steps taken by the intruder and alert the security before the attack is completed [35].

Of the papers that the survey describes, a paper by Schneier [36] is described as one of the first descriptions of attack graphs and how they can be generated [35]. Schneier describes what he calls attack trees as a way to think and describe security, build databases, capture and reuse expertise, and to make decisions when improving the security [36]. He describes that this can be represented as a tree, where the root is the goal of an attack and the nodes and leaves are possible steps in the attack [36]. This view is very similar to how a Fault tree analysis is done. For a further description about Fault tree analysis, see section 3.3.11.

A paper that is presented in the survey from Lippmann & Ingols [35] and is referred to by many other authors in this field is a paper from Sheyner et al. [37]. This paper is referred to by, for example Ammann et al. [38] and Ou et al. [39] and contains an early, but thorough and detailed approach with a formal model for attack graphs using a model checker [35]. According to Clarke, a model checker is a way to verify finite-state reactive systems automatically, described in temporal logic and modelled as a graph [40]. The use of a model checker for attack graphs was presented in an earlier paper by Ritchey & Ammann [41] but the paper by Sheyner et al. [37] presents a more developed model checker, an algorithm for automatic generation of attack graphs and a tool which implements these two functions.

Another early approach that was more focused on internal processes instead of networks was presented by Ramakrishnan & Sekar [42]. According to Ramakrishnan & Sekar, most vulnerabilities in computer systems originate from unexpected interaction between system components [42]. They describe that many existing methods for detecting weaknesses in
computer systems are based around rule-based systems which can only handle known weaknesses and that expert knowledge is required in order to put these systems together [42]. Since few experts have knowledge about all weaknesses that can exist in a computer system, Ramakrishnan & Sekar instead proposed a model based system [42]. Their idea was that this system should model the components and processes in a system and make it possible to identify weaknesses by analysing the interaction between these automatically [42].

Ramakrishnan & Sekar later performed tests on this system on a simplified model of a Unix system, with the use of a model checker [43]. The result of their study was that for this simplified system, the model based approach could find vulnerabilities, some that were unknown to them before they performed the study [43]. The benefits of using a model based approach are according to Ramakrishnan & Sekar that it can find both known and unknown weaknesses, that it is modular so new components can be added, that it can generate patterns for intrusion detection and that it can create rules for rule based systems [43]. The drawbacks that they have identified are that the part of their model that checks the different states, the model checker, does not work on large systems due to problems with complexity and scale [43]. They also write that it is hard to create good models of system processes [43]. Their proposed solution to the first problem is to find new methods for implementing the model checker and for the second problem they propose that models could be created by an automatic analysis of the source code for system processes [43].

The results from the studies of Ramakrishnan & Sekar [42, 43] are similar to the results from several of the papers that are described in the survey by Lippmann & Ingols [35]. One paper that describes the first problem in more detail is the paper from Ammann et al. where they describe that the model checker cannot handle realistic networks as the algorithm for the model checker has an exponential complexity [38]. The method that Ammann et al. proposes is to simplify the attack graph by assuming monotonicity [38]. They describe that this means that the actions an attacker does, never interferes with the ability to make other actions [38]. According to Ammann et al. this simplification lowers the complexity of the problem from exponential to polynomial and a system with five hosts and eight weaknesses that previously generated a graph with 5948 nodes now generate 229 nodes [38].

Problems with attack graphs

The survey by Lippmann & Ingols also describes four problems with attack graphs and areas where more work is needed [35]. The first problem that Lippmann & Ingols describe is that most algorithms do not scale very well [35]. The algorithm from the survey which had the best complexity was from the paper of Ammann et al. [38], which had a complexity of \(O(N^6)\) [35]. This was still a huge improvement from the earlier algorithms as it was of polynomial complexity, compared to earlier algorithms which in some cases were NP-complete. The reason for this is according to Ingols et al. that a fully connected attack graph is of complexity \(O(N!)\) [34]. Lippmann & Ingols write that a realistic network should include ten to a hundred thousand hosts, and that none of the algorithms presented in the survey is useful for these types of networks, as only an algorithm of linear or quadratic complexity can handle networks of this size [35]. They present two ways to find a possible solution; the first is to use a restricted graph that answers specific questions about the security and the second is to group the hosts together to reduce the factor \(N\) in order to use algorithms with a higher complexity [35]. The second problem they describe is the difficulty to obtain details about attacks and vulnerabilities [35]. They write that the information must be inserted by hand in many cases, which takes a lot of time and can produce many errors [35]. They present two ways of how to solve this: one is to use databases with data
about vulnerabilities that is stored in a format that is easy to use or to require that a limited amount of information is filled in by hand and fill out the rest with default values [35]. The third problem they describe is to compute reachability in the network [35]. This is how to compute which host can reach each other within the network. They write that in a larger network of thousand of hosts, firewalls and routers, this is a computationally complex problem and that many methods assume that this information is known and available [35]. They also write that in order to compute this, rules and configuration files from all devices in the whole network must be analysed [35]. The fourth problem that Lippmann & Ingols describe is how to generate recommendations for improving the security from the attack graphs [35]. This is because attack graphs can become very large and complex even for networks with fewer than 20 hosts and it is hard to analyse and understand them. The solution they present is to both generate and analyse the graph automatically and then present recommendations to how the security can be improved based on this analysis [35].

Many later articles build upon the results from Ammann et al. [38] and use this to get around the scaling problem. Both Ou et al. [39] and Ingols et al. [34] focus on modelling realistic networks. Ou et al. describe how an attack graphs consisting of 1000 hosts were constructed and how the vulnerabilities in the network were detected and analysed with algorithms that were of polynomial complexity, but of a lower degree than before [39]. Their algorithm was of complexity $O(N^2)$ given some assumptions and in the worst case, of complexity $O(N^3)$ [39]. Their solution builds upon monotonicity in a similar way to Ammann et al. [38], but does this by ignoring low-level details about the attack. Ingols et al. describe a method that can reduce the complexity further by removing redundant structures in the graph and scales with the size of the network in a nearly linear fashion [34]. They describe that this was tested on both a real network with 250 hosts and on a simulated network with 50 000 hosts [34]. In the later case, the graph was constructed and fully analysed in under twelve minutes [34]. This can be compared to the earlier approaches that are described by Lippmann & Ingols, where a graph from a network of four or five host could take hours to construct and analyse [35]. The program from Ingols et al. can also import data from certain firewalls or program that scans the network for known vulnerabilities, but the data then needs to be processed a certain amount by hand in order to be useful for the program [34]. The improvements that build upon the work by Amman et al. [38] are made by simplifying the graph to reduce the complexity. According to Ammann et al. monotonicity does not hold for all types of attacks, but these can be modelled to relatively high fidelity anyway [38]. This might introduce new problems as the algorithm does not check all states in the original graph and solves a similar but simplified problem that assumes monotonicity.

### 3.3.11 Miscellaneous methods

In the report by Nystedt [24], several different types of analysis methods are described. They are shorter or more specialised methods compared to the more extensive models in the rest of this section. Some of these methods that are either qualitative or quantitative are described in the following section.

#### Coarse analysis

A coarse analysis can according to Nystedt, be seen as a preliminary analysis where no attention is given to the details and the goal is to get an overview of the risk sources that exist [24]. He describes that the coarse analysis is often done in the beginning of a larger risk analysis and is then complemented by more detailed analytic methods [24]. According
to Nystedt, the method starts with limiting what shall be analysed and then the harmful
events that can happen as well as the cause for them are identified [24]. After this, he writes
that the probabilities and consequences for the events are identified, the consequences are
then assessed and proposals for measures are given [24].

Checklists

A checklist is according to Nystedt, based on previous experience and helps to identify
known risks [24]. They can according to Nystedt, be either general or more adapted to the
present activity and is one of the fastest and most effective ways to identify known risks
[24]. He also describes that they can be used when an activity is starting up, when it shall
be inspected or when it shall be phased out [24]. The disadvantage with checklists that
Nystedt presents is that the one who creates them must have very good knowledge about
the activity they are going to be used for [24].

Fault tree analysis

Fault tree analysis is described by Nystedt as a method to identify the events that leads
to a specific event, a so called top event by creating a logical diagram of the events [24].
The fault tree is generated by identifying the top event and then the events that leads to
it. Then the events that lead to the other events are identified until all events have been
found and the whole tree is constructed. According to Nystedt, the events that lead to the
top event can be identified through the study of flow charts and similar methods [24]. He
also writes that the events are connected through several types of logical gates [24].

Nystedt writes that just because an event occurs, it does not mean that the top event
needs to occur and all events do not need to occur for the top event to occur [24]. Cut
sets and minimal cut sets can according to Nystedt and Rausand & Høyland be used to
calculate the chain of events that leads to the top event [24, 44]. The difference between
them is according to Rausand & Høyland that a cut set includes all the steps that can occur
and lead to the top event, while the minimal cut set only includes those events that need to
occur to cause the top event [44]. The minimal cut set is therefore a subset of the cut set.

Reliability theory can be used to calculate the probability that the top event occurs. This
is according to Nystedt done by calculating the probability that the specific events in the tree
occurs and then calculating the probability that the top event occurs [24]. Reliability theory
can also be used to calculate other things, see Rausand & Høyland [44] for more examples.
In order to calculate the probability for the events, some statistic data for the components
in the tree needs to be available. According to Nystedt and Rausand & Høyland, this data
is then used in equations for the reliability function of the system, which is used to calculate
the total reliability of the system [24, 44]. This is then an estimation of how reliable the
whole system is. Based on the equations presented in the book by Rausand & Høyland, it
can be seen that a system that is connected in series is much more vulnerable than a system
that is redundant and connected in parallel [44]. If a system lacks redundancy and a single
component fails, then the whole system fails as well.

3.4 Comparisons between models

This section describes a comparison between the Swedish Armed Forces risk management
model and the other models and methods that have been described above. The comparison
focused on how the models were structured, which parts that were similar and which parts
that were different in order to find new ways to extend the risk management model in chapter 5. The comparison was in most cases done step by step, one or more steps in a model was compared to a similar step in the Swedish Armed Forces risk management model. In other cases, there were no clearly defined steps and then several parts of a model was compared with a step in the Swedish Armed Forces risk management model.

### 3.4.1 Step P2 of the IT lifecycle model

Step P2 was the step of the IT lifecycle model that was most similar to the risk management model; therefore it was this step that was compared with the risk management model and both differences and similarities were found. There were also parts of step P2 that were outside of the risk management model and some steps of it that were outside of step P2.

Step P2 of the IT lifecycle model had a different layout than the risk management model as it did not have a set of numbered steps. Besides this, those parts that were most similar to the first step in the risk management model were the analysis of the activity, the information classification and the preliminary system description. This was rather similar to identifying the assets that are worth to protect. The first part in the first step of the risk management model, to establish the basic values of the analysis, was not part of step P2 but was made in step P1 instead. The estimation of threats was part of step P2, in form of the threat analysis but the second step in the risk management model described this in more detail. The security demands and the vulnerability analysis could be seen as the identification of protections and estimation of vulnerabilities in the third step in the risk management model. The estimation of risks in the fourth step of the risk management model was part of the risk analysis in step P2, but there existed differences. The last step in the risk management model was not part of step P2 itself but was more similar to the second decision point.

The primary difference between step P2 of the lifecycle model and the risk management model was the order of the threat, risk, and vulnerability analyses. In the risk management model, the risk analysis built upon the threat and vulnerability analyses. In step P2 in the IT lifecycle model the vulnerability analysis built upon the threat and risk analyses. The risk management model also described the threat, risk, and vulnerability analyses in more detail. This can be seen if the description of the risk management model [1] is compared with the description of the IT lifecycle model [2] and the threat, risk, and vulnerability analyses [4]. Parts of the first and last step in the risk management model were also outside of step P2 in the IT lifecycle, for example the basic values of the analysis and the follow-up.

There were also some parts of step P2 that were outside of the risk management model. In step P2, it should be decided if the work should continue at all. This might fit into the risk management model but in most cases this should already have been decided upon before the analysis is started. Another thing that was outside of the scope of the risk management model was to remove those requirements that could not be realized and analyse if there would be changes in the organization or how the work was done in the organization. The last part that did not fit into the risk management model was the generation of a concept and the assessment of this concept from several perspectives.

### 3.4.2 Swedish Emergency Management Agency

Both the model from the Swedish Emergency Management Agency [12] and the risk management model [1] consisted of five steps. They were similar but in the model from the Swedish Emergency Management Agency the threat and risk analyses were done before the vulnerability analysis. The model from the Swedish Emergency Management Agency made
a difference between the notations of threat and risk, but other than that, they were used interchangeable in the model itself. The first and second steps of the models were very similar but it was described in more detail from where the Swedish Emergency Management Agency got knowledge about threats and risks. They also described that it was important to identify threats outside of the agencies area of responsibility. The third step in the Swedish Emergency Management Agencies model was similar to the fourth step in the risk management model and both prioritized consequences before probabilities in the risk analysis and used a risk matrix. In the fourth step, capacity was estimated which can be compared to the estimation of protection in the risk management model. The capacity served as the foundation for the vulnerability analysis, similar to how the protection did in the risk management model. One difference was that in the Swedish Emergency Management Agencies model, capacity was estimated after crisis management capacity, operative capacity, and the capacity to withstand serious disturbance to activities that are important to society. In the risk management model, protection was estimated after security awareness, exposure, and resources. The fifth step in the model was in all very similar but there were no further description of methods or tools for the analysis.

### 3.4.3 Swedish Rescue Services Agency

Both the Swedish Rescue Services Agencies model [18] and the risk management model [1] consisted of five steps, but the Swedish Rescue Services Agency did not make any difference between threats and risks. The first and second steps were similar but in the Swedish Rescue Services Agencies model, risks were identified and documented while in the risk management model, threats were identified and broken down into smaller events. The third step in the Swedish Rescue Services Agencies model was similar to the fourth step in the risk management model where the probability and consequences for different risks were estimated. In the fourth step, the risks were assessed against criterions, and guidelines for these criterions were presented, which did not exist in the risk management model. The fifth step was similar in both models. The Swedish Rescue Services Agencies manual also contained many methods and support for risk management but not a tool like the one that could be found in the publication from the Swedish Armed Forces.

### 3.4.4 Legal, Financial and Administrative Services Agency

Both the Legal, Financial and Administrative Services Agencies model [25] and the risk management model [1] included five steps but the model from the Legal, Financial and Administrative Services Agency was more focused on economical risks. Even though it was focused on economical risks, it was very general and can according to Palm, be used to identify other risks than economical ones [7]. The first step dealt with the identification of risks and was similar to the second step in the risk management model. One difference that was especially noted was that the activity should be described like it was in reality, not how it should be in theory. The second step was similar to the fourth step in the risk management model since it dealt with prioritizing risks but it was done from probability, consequence, economic significance, and priority. The third step dealt with the decisions that should be taken and was similar to the fifth step in the risk management model. The fourth step in the model was to carry out the decisions and this step did not exist in the risk management model. There it was instead described that the decisions should be carried out as an order according to the Swedish Armed Forces usual routines [1]. The fifth step was a continuous follow-up that could be compared to the fifth step of the risk management model. If the
whole model was compared to the risk management model, the largest difference was that it did not focus as much on the first stages of the risk management process but described the last stages in more detail.

3.4.5 County Administrative Board of Stockholm

Palm describes that the County Administrative Board of Stockholm’s model [27] is similar to the risk management model [1] as it is concrete and describes each step in detail, but that it is more complicated as well [7]. The model consisted of two parallel tracks which had four steps each and were done at the same time but slightly out of phase. The first step was similar to first step in the risk management model but the second step was similar to the second, third and fourth step in the risk management model. The main difference between the models were that it was the three most extraordinary events that were selected out and analysed with the help of a computer based tool, IBERO. In the risk management model, all threats were broken down into smaller events and then analysed, but without a computer based tool. The third step involved documentation which was not covered specifically in the risk management model as it was done gradually with the tool that was used. The fourth step was similar to the fifth step in the risk management model as it involved measures to the identified risks.

3.4.6 Swedish Security Service

According to Palm, the Swedish Security Services model [31] is very similar to the risk management model [1] since both focus on antagonistic threats and make a difference between threat and risk [7]. The model consisted of four steps and the first was to identify assets that are worth to protect. This was similar to the first step in the risk management model, but there the basic values for the analysis and a scale of consequence was established as well. This was not described in the Swedish Security Services model and it is possible that this is established before the analysis begins. In the seconds step, the threats were identified from certain categories but they were not broken down into smaller parts as in the risk management model. The third step can be seen as combination of the third and fourth step in the risk management model but was not as detailed. The fourth step was similar to the fifth step in the risk management model but the Swedish Security Services model did not mention anything about methods or tools.

3.4.7 Swedish Road Administration

The largest differences between the model from the Swedish Road Administration [20] and the risk management model [1] were that the model from the Swedish Road Administration did not mention threats and had a separate model [32] to weight the cost against the benefits for different measures. The first step of the model was to identify assets but in contrast to the risk management model they were sorted after pre-defined categories and which place they belonged to. The second step of the model was to identify the dangers that existed which were similar to the second step in the risk management model. The third step of the model was to identify the risk factors that caused the dangers and the probability that they happened. This step did not fit directly into any of the steps in the risk management model but included some parts that could be found in the second, third and fourth step. The fourth step of the model described the consequences that could occur and was rather similar to the fourth step in the risk management model but in the Swedish Road Administrations model, both the expected and maximal consequences were estimated. The fifth step was to
estimate the total risk level, which was similar to the fourth step in the risk management model but both a risk matrix and consequence classes were used. The sixth step in the model was similar to the fifth step in the risk management model as the measures against the risks were established, but with the addition of a cost versus benefit analysis for them.

Together with the model there were three ways to present the result: parts of the analysis in a pre-made form, a risk matrix, and geographically on a map. These three ways to present the result were similar to the ways the results of the risk management model were presented. The difference was that support for a GIS enhanced tool was being developed. In this it should be possible to see the dangers on a map and then receive more information about them. The ability to view data on a map was discussed in the appendix to the risk management model but no tool has been developed for this.

3.4.8 U.S. Department of Energy, Office of Energy Assurance

The model from the U.S. Department of Energy [33], was similar to the risk management model [1] in that it was extensive and focused on antagonistic threats, but what set it apart was that it was also focused on a specific activity. The model consisted of three parts where the second part was the most extensive one. The first part was similar to the first step in the risk management model but also described how data from the analysis should be handled and protected. The first sub-step in part two dealt with how to analyse network architectures and that was not covered by the risk management model. The second sub-step dealt with the analysis of threats and weaknesses and was similar to the second and third step of the risk management model. What set it apart was that it was done in cooperation with local law enforcement agencies. The third sub-step dealt with penetration testing and was not covered in the risk management model. The fourth, fifth and sixth sub-steps dealt with different types of security which was similar to the third step of the risk management model. The seventh sub-step dealt with policies and was not covered in the risk management model, but it might fit into the third step. The eighth sub-step dealt with impact analysis and was related to the fourth step in the risk management model. The ninth sub-step dealt with the analysis of dependencies to other parts of the infrastructure and was not covered in the risk management model. It might fit into vulnerabilities and could therefore be part of the third step. The last sub-step in part two dealt with risk characterization and was similar to the fourth step in the risk management model. The third part of the model was similar to the fifth step in the risk management model but also dealt with training and lessons learned.

3.4.9 British Telecom

The model from British Telecom [11] was different from the risk management model [1] since it only contained three steps and focused on the attacker’s perspective. It was similar in the way that both models focused on antagonistic threats. The first step dealt with the identification of critical assets and conceivable goals which made it similar to the first step in the risk management model. The second step dealt with identifying vulnerabilities and breaking them down in smaller parts; this was similar to the third step in the risk management model but used the methodology from the second step. In the second step, penetration testing was also done and this was not part of the risk management model as it was more general in nature. The third step dealt with the assessment of the probability that a threat was carried out, from the perspective of the attacker. This was done by analysing the threat using similar terminology from the second step in the risk management model.
3.4. Comparisons between models

The attackers were then divided into different classes and this was not part of the risk management model. The model also assumed that the threats had serious consequences from the start and did not analyse this in the model itself. It also left out large parts of the fourth and fifth step in the risk management model.

3.4.10 National Institute of Standards and Technology

The model from the National Institute of Standards and Technology [19] was very similar to the risk management model [1] as it contained similar steps, made difference between threats and risks, was focused towards antagonistic threats but was more detailed and focused towards IT. The model was the first part in a larger risk management process and consisted of nine steps. In the first step, the system was defined which was similar to the first step in the risk management model but important assets were not identified until later in the model. The second step dealt with the identification of threats and was similar to the second step in the risk management model but the threats were not broken down into smaller events. The third step dealt with the identification of weaknesses which was part of the third step in the risk management model. The fourth step dealt with the identification of protections which was part of the third step in the risk management model but was done before the identification of weaknesses. In the fifth step, the probabilities that weaknesses were exploited were estimated which was similar to the fourth step in the risk management model. In the sixth step, important assets were identified and the consequences if they were exploited were estimated. The identification of assets were done in the first step in the risk management model and the estimation of consequences were done in the fourth step. In the seventh step, the risk was calculated from the estimations that had been developed in step five and six; this was also part of the fourth step in the risk management model. The eighth step dealt with measures against the risks, which was part of the fifth step in the risk management model but not as detailed as in this model. The last step dealt with the documentation which did not have a step in the risk management model as it was done gradually with the tool.

3.4.11 Attack graphs

Attack graphs, as described by for example Lippmann & Ingols, Schneier or Sheyner et al. [35, 36, 37] were very different from the risk management model [1]. Instead of analysing an activity by identifying threats, vulnerabilities and risks, attack graphs were used to build a model of a computer network and analyse this. The model was focused around a single or several threats, often that an attacker gained root or administrator access to one or more computers in the network. They were similar to the risk management model as they focused on antagonistic threats, but they did not analyse other threats and were only applied to networks and computer systems. Another difference was that they assumed that there were an antagonistic threat and did not analyse this further, as the risk management model did. The attack graphs did not analyse risks as the combination of probability and consequence either, only vulnerabilities in the network or computer system. They were also a quantitative method in contrast to the risk management model and was similar to fault tree analysis. The attack graphs could be compared to a small part of the risk management model, that was done on a very detailed level as they analysed a single or a few threats, the protections and the weaknesses in these. Measures against the risks were not mentioned either, but could easily be identified based on the result of the analysis.
3.4.12 Miscellaneous methods

The coarse analysis [24] had a similar layout as the risk management model [1] and described the different steps but was not as detailed. The analysis could be performed very fast and could be part of the first step in a larger analysis. Checklists [24] were very different from the risk management model as they could not identify any new risks and only find those that had been identified before. They could be used to verify the security in a system and ensure that it was maintained at a high level. Fault tree analysis [24] and reliability theory [24, 44] were similar to attack graphs as they were quantitative and modelled a system that was evaluated instead of making a direct analysis. They modelled a top event that could be compared with a chosen risk and focused on a specific but detailed part of the analysis in the same way as attack graphs. The difference was that instead of a list of vulnerabilities, the result was a numerical value on how reliable the system was. Compared to the risk management model, the analysis focused on the probabilities of risks but ignored the consequences. The analysis also ignored protections, vulnerabilities, and measures against the risks.
Chapter 4

Threats to IT systems

In order to evaluate how the Swedish Armed Forces risk management model could be extended for IT, related threats needed to be identified. This chapter gives examples of threats that can affect IT systems divided into three categories, technical threats, social threats, and threats related to human-computer interaction and security. Technical threats are those threats that are closely related to use or abuse of hardware or software. Social threats are threats that are more related to the users of the hardware or software. Finally, threats related to human-computer interaction are threats that occur from the interaction between the users and the hardware or software.

4.1 Technical threats

There are several types of technical threats that can affect IT systems, some threats are more related to hardware and other more related to software. The threats can be connected to certain technologies, systems, or security functions that are affected by an attacker. Some examples of these are presented below, divided into several categories.

4.1.1 Collecting and exploiting information

According to the Swedish Armed Forces, the goal with attacking or exploiting a system can be to get access to information about the system or information in the system [5]. The Swedish Armed Forces also describe that the purpose of exploiting the system could be to make sure that later attacks have a higher chance of being successful [5]. In order to attack a system, it is important to know as much about it as possible. McClure et al. describe several methods for getting information about a system, for example by searching for digital footprints, looking for open ports, identifying the operative system, and identifying weaknesses [6]. An attacker that has knowledge about a specific system can then make more focused attacks against it.

4.1.2 Attacks

Two types of attacks with the potential effect of being serious threats have been identified by the Swedish Armed Forces: mass attacks and aimed precision attacks [5]. Both of these can according to the Swedish Armed Forces abuse functionality, security holes, or routines [5]. Since the Swedish Armed Forces have systems that handle sensitive information or needs to
be extra secured for other reasons, they need to be protected against attacks. One way of securing these systems is according to the Swedish Armed Forces to disconnect them from all external networks [5]. If an attacker then wants to get access to them, the attacker must get physical access and this can be difficult if the systems are secured with physical security systems, for example locked doors. One way of getting this access is get hold of keys or access cards and McClure et al. gives examples on this and how to pick locks, copy, hack, or create access cards [6]. If this fails, another way is to bypass the physical security through the use of social engineering; this is explained in more detail in section 4.2.2.

According to McClure et al. an attacker often starts an attack by targeting known vulnerabilities, gets a limited access and tries to escalate this to get administrator or root access [6]. One problem that is described by both the Swedish Armed Forces and McClure et al. is that administrators who tries to prevent this type of attack must know and keep track of every security hole while those that attack only need to know those that are needed for the specific attack [5, 6]. Even if the administrators succeed in keeping the attackers out of their system, it is according to the Swedish Armed Forces still possible for the attacker to create disruptions by overloading the network or mail servers with traffic [5].

4.1.3 Malicious code

Malicious code is according to the Swedish Armed Forces, code in the form of add-ons, script or source code that were made with the purpose of attacking the system on which they are executed [5]. The goal can according to McClure et al. be to create back doors in the system, create logs of the user’s activities, or get hold of information [6]. Examples of malicious code can be worms, virus, or similar scripts that are transmitted between computers. The primary defence against malicious code is antivirus software, but these programs can only detect code that is known to the developers of the program and they need to be updated continuously.

4.1.4 Adding or modification of hardware

It is not only the software in a system that can be modified by an attacker; the hardware can be modified as well. This can according to the Swedish Armed Forces be done by adding extra storage units that saves copies of important information or by adding wireless network cards that gives an attacker a way into the network [5]. If this is combined with other factors like insiders, the security can according to the Swedish Armed Forces be drastically reduced [5]. McClure et al. describe how hardware can be modified by reverse engineering and how it is possible to analyse and get information from firmware [6]. Another possibility is according to the Swedish Armed Forces, that the hardware is modified on the way from the supplier to its final destination [5].

4.1.5 Exposure of source code or blueprints

According to the Swedish Armed Forces, many systems that handle sensitive information is developed specifically for this and are not available to the public [5]. But just because a system is unknown does not mean that it is secure. If the source code or blueprints for such a system is stolen, the attackers can according to the Swedish Armed Forces reverse engineer it to find out how it can be attacked [5]. McClure et al. describe similar techniques but is more focused toward hardware and how it can be reverse engineered in order to find or passwords built into devices [6].
4.1.6 Wireless communication

Wireless communication has according to the Swedish Armed Forces the properties that it is easy to wire-tap, interrupt and that it is easy to establish the position of the transmitters [5]. Due to this, they write that the information that is sent through wireless networks should always be encrypted [5]. McClure et al. also write that it is hard to discover if the traffic in the network was wire-tapped or not [6]. McClure et al. describe wireless communication further and mention several techniques on how to find and connect to wireless networks, so-called war driving [6]. This includes everything from different types of antennas to how an attacker can break the encryption of the network [6].

4.1.7 Wired communication

Communication that is transmitted in cable is harder to wire-tap than wireless communication as equipment in many cases must be attached directly on the cable, in switchboards, or switches [5]. One problem with wired communication is that all wired communication and electronic equipment, like computers, transmit electromagnetic radiation that can be detected. This is known in the Swedish Armed Forces as exposing signals [4]. For more information about exposing signals, see section 1.5. Another problem with wired communication is according to the Swedish Armed Forces that some parts of the network passes through servers abroad or in wireless networks and this makes wired communication as vulnerable to wire-tapping as wireless communication [5]. A weakness that McClure et al. describe is that many routers or other network equipment work directly from the box, often with weak or well-known standard passwords [6]. If these are not changed before the equipment is plugged in, it is easy for an attacker to take control over the equipment and transmit copies of all traffic to another location so they can be analysed later [6].

4.2 Social threats

Some types of threats are not aimed at the hardware or software but at what Sasse et al. and Mitnick et al. describe as the weakest link in the security chain: the users behind all security systems [45, 46]. In this section, two types of social threats are covered: insiders and social engineering.

4.2.1 Insiders

According to the Swedish Armed Forces, insiders are not a new problem but still a serious threat and can cause large amounts of damage [5]. Today there are more persons that have access to computer equipment than ever before and it is possible to store large amounts of information on removable hard drives and memory sticks. This in combination with the fact that information is easier to copy when it is digital makes, according to the Swedish Armed Forces that the consequences today can become more extensive than they were before [5].

The Swedish Armed Forces also describe that an insider can be recruited by free will or be forced to become one [5]. They write that it is easier to map a person’s behaviour when there is much information available through sources on the Internet [5], this information can be used to get a hold on someone for extortion or make threats towards their relatives.
4.2.2 Social engineering

McClure et al. describe social engineering as a technique where an attacker takes advantage of the people that exist behind all security systems in order to get past them, often by making use of a telephone or by email [6]. In social engineering, there exist many techniques for deceiving or persuading people. McClure et al. describe some of these, for example to call and pretend to be an employee that needs help, to pretend being someone from IT support or by fooling users with fake web pages, so called phishing [6]. Mitnick et al. describe different scenarios of how social engineering can be used in more detail and describe different techniques for deceiving or persuading people [46]. One technique they describe is how an attacker can start by looking for basic information, for example by looking at public information on the Internet or by looking through a company or organizations garbage [46]. By using this, they write that an attacker can pretend to be an employee calling for help and then get access to more information [46]. The information that the attacker wants does not need to be sensitive but can according to Mitnick et al. be combined with other information in order to help the attacker escalate the attack [46]. Besides this, they describe that it is important to build up trust with the person being manipulated to increase the chances of success; if this fails it is also possible to stimulate feelings of sympathy or shame [46]. Other ways to get past security is to help people and then ask them for a favour or target people low in the hierarchy and threaten them by pretending to be someone with authority [46].

A similar discussion is taken by Cialdini, which presents six different psychological tendencies which can be used to influence or manipulate people [47].

1. The first tendency is reciprocation, which according to Cialdini is to perform a service or give someone a gift [47]. The person who receives this will then feel they are in debt to the attacker and this makes it easier to make them do something in return.
2. The second tendency is consistency, which according to Cialdini is to make a person vow or express their opinion about something [47]. The next time, the person is often consistent and stands up for this opinion if the attacker asks for something which is related to what the person said the first time.
3. The third tendency is social validation, which according to Cialdini is that people often look for social validation for their behaviour [47]. If an attacker convince a person that many others have done the same thing before, it is easier to get the person to do what the attacker wants.
4. The fourth tendency is liking, which according to Cialdini is that people are more cooperative with someone they like [47]. Therefore it is easier to manipulate a person if they like their attacker.
5. The fifth tendency is authority, which according to Cialdini is that people often follow a leader with authority [47]. By acting that way or wearing symbols that represent authority, it is easier to make people obey.
6. The sixth tendency is scarcity, which according to Cialdini is that people want things that are rare [47]. By offering people things that are only available in a small edition or convincing them they are especially selected it is easier to make them accept or buy something.

Cialdini writes that the likely reason for why these tendencies work is that they are positive from an evolutionary perspective, especially when people are depending on a group that needs to be held together [47]. They can be used both for legitimate and illegitimate purposes, but if a person is aware of these ways to manipulate, it is easier to resist [47]. He also writes that the tendencies are well understood and often used for advertising [47].
4.3 Human-computer interaction and security

According to Dix et al. HCI or human-computer interaction has its roots from research in ergonomics and human factors but has then moved into areas of cognitive science, psychology, and computer science [48]. They describe human-computer interaction as focused on understanding how users think and reasons, the social factors that controls user’s behaviours and how computer systems should be developed in order to support their users [48].

There exist other threats than those that happens because of an external attacker and some of these are related to human-computer interaction. According to Besnard & Arief, it is in many cases regular users that consciously or unconsciously breaks the security regulations that exist and makes an attack possible [49]. This section describes this issue and presents some different views on the relation between human-computer interaction and security.

4.3.1 The human factor

According to Sasse et al. many that conduct research in security have realized the importance of the human factor in the last years, but those that attack computer systems have known and exploited this for a long time [45]. They write that it is better to focus on why the problems happen in the first place, instead of blaming the users and the human factor [45]. Sasse writes in another report that security systems that are hard to use can lower the security instead of raising it [50]. Smith has a similar opinion and claims that many problems with security comes from bad interaction between the users and the system [51].

Besides this, there are other problems related to the human factor that can influence the security. Besnard & Arief describe that humans do not use all conceivable information or their whole capacity to solve a task, but uses so much information or capacity to solve the task so that the result is good enough [49]. They write that this also applies to security and that users often choose a level of security that is lower than it could be [49]. They write that this can be to write down passwords to remember them or not allocating the time needed to update software and eliminating breaches in security [49]. They describe that this can also be a programmer that does not bother to eliminate problems with security in order to save time or money during the development of a program [49]. Another problem according to Besnard & Arief is that users tend to take larger risks over time, which leads to that things that was not accepted in the beginning becomes more accepted over time and that the security is deteriorating over time [49].

4.3.2 User centered security

Sasse et al. and Sasse write that many problems depends on that the security systems and the policies that exists are not created with the users in mind, and describe some causes to why users have problems following security policies [45, 50]. The primary ones are that the users are exposed to a high cognitive load when there are many passwords and procedures that must be memorized and that the goals with the security does not converge with the users goals [50]. According to Besnard & Arief and Sasse et al. the security should be focused around the users and not the activity, as these might be conflicting goals that are hard to combine [49, 45]. They write that the user’s goal is to do what they are employed for and to think about security or stick to security policies will conflict with that goal [49, 45].

It can according to Sasse et al. be a certain level of security that needs to be upheld, which takes a lot of time and conflicts with demands on the employees to be finished with a task to a specific deadline [45]. If the employee must prioritize between several goals, the security will according to Besnard & Arief often come in second hand [49]. Besnard & Arief write
that the security can be seen as an encumberment that makes it harder to fulfil the user’s
goal and therefore it needs to be enforced in order for them to follow it [49]. They write
that the security should be focused around the users so they do not need to think about it
and can focus on the task instead [49]. This will both simplify the user’s task and make it
easier to uphold the security.

4.3.3 Security policy and human behaviour

According to Besnard & Arief, policies do not rule human behaviour and humans have a
tendency to find new ways to do tasks that bends or breaks the rules that exist [49]. They do
not mean that policies and rules are without use but that the users are not necessarily going
to follow them and that the security does not automatically become better just because
there is a document that specifies how something should be done [49]. This is something
that those responsible for the security should take into account. According to Sasse, it
might be so that the users do not feel that there exists any real threat and therefore do
not follow the security policy that exists [50]. The reason for this is that people usually
want to perform a task as cost effective as possible. According to Besnard & Arief, if it is
possible to do something faster or more effectively by breaking the security policy, someone
will eventually do it, regardless of the type of security policy that is in use [49]. Sasse writes
that many have tried to enforce security by forcing the users to follow the security policy
by constant inspections [50]. She also writes that this probably works better within the
military sector as most of the security functions that are in use comes from there and are
adapted to their type of activity [50].

4.3.4 Human memory and passwords

There are several things that the human memory has problems with. According to Sasse
some of these are that memories fade over time, meaningless things are harder to remember
than things that can be related to something, it is hard to remember things without clues,
it is hard to forget things on demand, and memories can disturb each other [50]. The way
in which passwords are remembered demands according to Sasse, that the users do just
these things [50]. Two ways for users to solve this problem are according to Sasse to write
down the passwords or use the account of someone else in order to log on to a system if
they forget their password [50]. Both these solutions which enables the users to continue
their daily work, also bypasses the security system. A report from Sasse et al. describe
several studies around passwords, the problems that exist and the policies that rules how
they should be used [45]. Their result showed that by demanding that the passwords should
be changed regularly, the system was more exposed since the users were forced to write
them down in order to remember them [45]. By not demanding that the passwords should
be changed as often, it is according to Sasse et al. possible for users to remember stronger
passwords without writing them down as they used them for a longer period of time [45].
They also write that it is possible to use other methods to identify users, use techniques for
management of several passwords, use the same password for several systems, and motivate
the users to follow the policies instead of punishing them when they do not [45].
Chapter 5

Combination of models from the Swedish Armed Forces

This chapter describes an evaluation of how the Swedish Armed Forces risk management model can be used for the risk analysis in step P2 of the Swedish Armed Forces IT lifecycle model. After the evaluation, an example of how the risk management model can be extended for IT is presented. The extension is based on the result from the previous chapters and takes the combined model, other models and methods, and threats for IT into account.

5.1 Evaluation of how the risk management model can be combined with the IT lifecycle model

The risk management model and step P2 of the IT lifecycle model are similar in many ways, especially since the risk management model is general and can be used for many different risks. The main difference between them is the threat, risk, and vulnerability analyses.

In the IT lifecycle model, it was only mentioned that they should be done, not how they should done [2]. This was instead described in more detail in another manual from the Swedish Armed Forces, known as H Säk IT [4]. Here it was described that a threat analysis should be done first to identify the threats; this was then followed by risk and vulnerability analyses [4]. When the threat analysis was done, it was described in H Säk IT that the threats and risks were compared against the security demands that could be found in the KSF [4]. The threats or risks that the security demands had not succeeded to eliminate and only reduced or not affected at all were called vulnerabilities [4]. The notions of threat, risk, and vulnerability from H Säk IT were equivalent to the notations in the risk management model but there was no clear separation between the risk and vulnerability analyses. To compare threats with protections were described in H Säk IT as a risk analysis, but the result were vulnerabilities, not risks [4]. In addition to this, two ways to perform the risk and vulnerability analyses were described in H Säk IT [4]. In the first, a threat analysis was performed followed by a combined risk and vulnerability analysis, but in the second, a threat analysis was performed, then a risk analysis and a vulnerability analysis [4].

In the Swedish Armed Forces risk management model, a threat analysis was performed first to identify the threats that existed [1]. When this was done, the model described that the protections were identified and compared with the threats; a lack of protection was then defined as a vulnerability [1]. This way of performing a vulnerability analysis was similar
to the first model but here it was done before the risk analysis. Risk was then according to the model, a combination of the estimation of the probability and consequence of the threat based on the previous analyses [1]. The risk management model made a clear difference between the notations of threat, risk, and vulnerability in the same way as the older manual but made an even clearer difference between the different analyses and how the notations were used in the model. The risk management model also described the different steps in each analysis in much more detail, compared to how it was done in the IT lifecycle model.

Since the order of the analyses was different between the models, one of them had to be modified in order to combine them. When the threat, risk, and vulnerability analyses were compared, two things could be seen. The first was that the risk management model was more detailed than the analysis used in the IT lifecycle model and that the steps were defined in a way that made a clear separation between them. The risk management model was also developed after the manuals that described the IT lifecycle model [2, 4], was developed to be used within the whole Swedish Armed Forces and included detailed examples to how it should be used [1]. Based on this, it was the step P2 in the IT lifecycle model that should be modified. This was done by switching place on the risk and vulnerability analyses in the model so the risk management model could be used for the risk analysis. Those parts of the risk management model that were outside of step P2 were still used, but in other parts of the model. The basic values of the analysis and scale of consequences in the first step became part of step P1 and the fifth step became part of the second decision point. Some parts of step P2 were also outside of the risk management model, the most important were the security and constitutional analyses and the establishment of information security classes. Since step P2 contains more parts than the risk management model does, this was to be expected, but is still a weakness of the combined model as all parts could not be combined.

How the combined model looks can be seen in Figure 5.1. This was based on a similar layout as in Figure 3.5 that was adapted from a figure in the report by Bengtsson & Hallberg [16], but with the parts that were combined with the Swedish Armed Forces risk management model highlighted, and extended with those parts that were described in DIT04 [2] but not part of the report by Bengtsson & Hallberg. A larger version of the same diagram where the details are easier to see can be found in Figure B.5 in Appendix B.

Figure 5.1: A diagram that shows how the two models can be combined. The structure of the model is adapted from the report by Bengtsson & Hallberg [16].
5.2 Extending the risk management model for IT

The risk management model can be extended and used for many types of threats and risks. In order to use it to the risk analysis in step P2 of the IT lifecycle model, it needed to be extended for IT. This section presents the five steps in the risk management model and an example of how they can be extended, given the information that have been presented about other models and methods for threat, risk, and vulnerability analyses in chapter 3 and threats for IT in chapter 4. This extended model can then be used for these analyses in the modified step P2 of the IT lifecycle model.

5.2.1 Step 1 - Establish basic values for the analysis

The first that should be done is to establish the basic values for the analysis as described in the risk management model. When these are established, the assets that are worth to protect are identified. In step P2 in the IT lifecycle model, this was done through an analysis of the activity, a preliminary system description and by classifying the information. Stoneburner et al. described that the system could be defined by identifying the hardware, software, information, and users [19]. This was done with the help of forms, interviews, documentation, and analytic software. The system that was identified could be both implemented or on the planning stage. The U.S. Department of Energy, Office of Energy Assurance described how network architecture could be identified [33], which McClure et al. also described in detail [6]. The Legal, Financial and Administrative Services Agency described that it was important to estimate the difference between theory and practice and how an activity looked in reality [25]. Otherwise, the result of the analysis could be misleading. The information that was in or around the system should then be classified according to the classes established by the Swedish Armed Forces [4] and a scale of consequences in ten steps should be set.

5.2.2 Step 2 - Concretize and estimate the threats

When the activity and the system are defined, the threats should be identified and broken down to unwanted events, occurrences, or modus for attack. The threat level should be estimated on a scale of five and could according to Palm be estimated from intention, capacity and opportunity [7]. Stoneburner et al. wrote that the threats could be intentional, unintentional, natural, human or depend on the environment around the activity [19]. Based on the information in chapter 3 and chapter 4, the following threats were identified:

- Open information [31]
- The difference between theory and practice [25]
- Espionage, sabotage, criminality related to security, infiltration, and technology as a threat [31]
- Social engineering [11, 6, 46]
- Dependencies on other infrastructure [11]
- Unexpected interaction between system components [42]
- Systems that lack redundancy [44, 24]
- Security that is not created from the users goals and perspective [50, 49]
- Long and complicated passwords [50]
- Users that do not follow security policies [49]

Besides these, there were many other threats that were described in manuals from the Swedish Armed Forces [5, 4], by McClure et al. [6] and Stoneburner et al. [19]. Examples of
these were different weaknesses in software, exploits for attacking and gaining control over computers, and more traditional military threats. These threats do not make a complete list as threats change over time. To counter this, new analyses must be done continuously and the threats must be analysed based on the preconditions that exist in each case.

5.2.3 Step 3 - Identify protections and estimate vulnerabilities

When the threats are identified, the protections should be as well. One way to support this was to divide the protections into different classes and start the analysis from there. Stoneburner et al. classified protections as either technical, non technical, preventive, or detective [19]. The technical protections were according to Stoneburner et al. built into hardware or software in form of encryption, access controls or similar things and the non technical in policies or physical protections [19]. An example of preventive and detective protections could be firewalls and antivirus software. When the protections were identified, it was useful to identify what protection that could exist, not only those that existed. In order to identify these protections, the methods that were part of step P2 could be used. By doing a security and constitutional analysis and taking the security demands in the KSF into account, it was possible to find new security demands that the system should fulfil.

The vulnerabilities could according to Stoneburner et al. be identified by studying sources of vulnerabilities, performing system security testing, penetration testing, and using checklists [19]. Penetration testing was also described by the U.S. Department of Energy, Office of Energy Assurance [33] and described in great detail by McClure et al. [6]. When the vulnerabilities were identified, they should be estimated on a scale of five. They could according to Palm be estimated from security awareness, exposure and resources [7].

5.2.4 Step 4 - Estimate the risk

The risk is estimated through the probability and consequence for the threats that have been identified. This is done with the help of the analyses that have been performed in the previous steps. The probability is estimated on a static scale and the consequences on the scale that is established in the first step, this is then transferred to a risk matrix. According to Stoneburner et al. the probability was estimated from the motivation and capacity of the threat source, the nature of the vulnerability and if there existed protection against the threat [19]. They also wrote that the consequence was estimated through the security class that the information had and what would happen if there was a loss of integrity, loss of availability and loss of trust in the information [19]. In step P2 of the IT lifecycle, the estimation of consequences was done during the security analysis, earlier in the process than the risk analysis. However, this was only focused towards the consequences that were related to information and not all types of risks.

5.2.5 Step 5 - Risk management decision and plan for follow-up

In the last step, it is decided if the risks are acceptable, should be taken care of or if the decision should be sent up in the hierarchy. A plan for follow-up and analysis of the proposed measures are done as well. The Legal, Financial and Administrative Services Agency described how a follow-up might be done and presented several questions that should be answered [25]. If there were several measures that could be done, there was a high chance that all of them could not be performed due to limits in time or due to cost. Both Stoneburner et al. and the Swedish Road Administration described how to do a cost versus benefit analysis to decide which measures that should be performed [19, 32].
Chapter 6

Design for the new implementation

This chapter describes a design proposal for how the combined and extended model presented in chapter 5 can be implemented. This starts with a description of the program NTE and the two plugins SANTA and EASTER. Then some design guidelines are presented, this starts with more general guidelines and them moves on to more and more specific requirements. This is followed by the design proposal which describes what functionality that should be implemented and how this should be done. This is then expanded further as the implementation of the model is described in more detail. When the proposed design has been described, an evaluation if NTE and EASTER can support these changes are presented. The last part of the chapter is a list of the functionality that needs to be implemented based on the proposed design and the evaluation of NTE and EASTER.

6.1 Description of NTE

NTE stands for New Tool Environment and is a program that acts as a framework for plugins. It was constructed as part of a master’s thesis done at FOI by Bengtsson & Brinck [3] and replaced an earlier program called ROME2. ROME2 was constructed to implement a method for security assessment called MASS and was a part of an earlier master’s thesis done at FOI by Andersson [14]. MASS was later extended by the method XMASS by Hallberg et al. [13] and the new program NTE was then constructed to implement the changes in the method and other functionality related to file and database handling.

NTE works by creating an environment that can be used by plugins to the program. The plugins used must implement interfaces that are specified by NTE and deals with database handling, saving, loading, and exporting projects. In NTE it is possible to create a new project, open an existing one, save a project to the database in the project file, delete projects, and export them to a database in a new project. By implementing the interfaces, it is possible to create new plugins that uses NTE as a framework or to create new plugins that connects NTE to other databases. At present the only database in use is SQLite.

NTE also keeps track of a number of Requirement Collections that are a collection of security requirements that can be used by the systems created in the plugins. The structure of NTE can be seen in Figure 6.1 and a more complete description of the history and implementation of NTE can be found in the report by Bengtsson & Brinck [3].
XMASS is a method for security assessment that computes a numerical value that represents the security of a computer network. The network is represented as a graph with entities and relations between them. The entities and relations can be given profiles that represent different levels of security. These profiles then affect the calculations done in XMASS. For further information of XMASS, see the report by Hallberg et al. [13]. XMASS was then implemented as the plugin SANTA for NTE by Bengtsson & Brinck and allows the user to model a network and assess its security [3]. The plugin implements functionality to model a network by creating and modifying entities and profiles. These are then used in the evaluation phase, where it is possible to set different start and end states to evaluate how changes in the entities in the network affect security over a longer period of time.

EASTER is a new plugin for NTE that has a different approach than XMASS and SANTA. Where XMASS measures security as a numerical value, EASTER measures it as how many security requirements the system supports. The security requirements are stored as separate Requirement Collections and the system supports several different Requirement Collections. The plugin is based on SANTA and contains much of the functionality that was used there. Much of the code that were part of SANTA is still left in EASTER but is not used for anything. Aside from the user interface and the calculations, most of the modelling of the system is done in the same manner in EASTER as it was done in SANTA.

Two screenshots of EASTER can be seen in Figure 6.2 and Figure 6.3. These shows the main user interface and the forms where the user can fill in which security requirements that the system should fulfil. The first screenshot shows the main window of EASTER with a panel where the entities and the relations between them are constructed. On the right, a menu that shows data about the system can be seen. In this network, there is an entity that does not fulfil all requirements that it should. The second screenshot shows a dialogue window that is part of the properties of each entity. This lets the user select which security requirements that the entity should fulfil, create and set profiles for the entities.
6.2 Design guidelines

The design of the implementation of the combined and extended model was based on guidelines from several sources. These were the literature study in chapter 3, the sources the literature study was based on, literature about human-computer interaction, risk management, and requirements from the Swedish Armed Forces. The design was also heavily influenced by the combined and extended model that was presented in chapter 5, since it was this model that should be implemented. This section describes these sources in more detail and lists the guidelines for the design. The guidelines start with the more general ones from the literature on human-computer interaction and design, then move on to more specific
guidelines from the literature study and a set of guidelines for risk analysis. The last guidelines for the design are very specific and are a list of functionality that representatives from the Swedish Armed Forces and the Swedish Defence Materiel Administration would like to have implemented in the program.

6.2.1 General rules and guidelines for user-centered design

Dix et al. describe several guidelines for interface design that have been proposed by Schneiderman, Norman and Nielsen [48, p. 282]. These are all on a high level of abstraction and do not fit with every type of user interface. Dix et al. write that even though these are rather simple guidelines, by having them as a checklist for the design, the user interface will be better than if they would be ignored [48, p. 282]. These guidelines are included since Schneiderman, Norman and Nielsen are well known authorities in their field and their guidelines are discussed in many textbooks on design and user interfaces.

Schneiderman’s eight golden rules of interface design

This section describes eight rules proposed by Schneiderman regarding interface design. They are presented exactly as they are listed in the book by Dix et al. [48, p. 282] but are then given a short explanation.

1. Strive for consistency
   The layout of a program should remain the same all the time, the same terminology should be used in all parts, and objects or actions should behave the same throughout the whole program.

2. Enable frequent users to use shortcuts
   More advanced users should be able to use shortcuts when doing actions to increase speed. These can be made using keyboard combinations, scripts or macros.

3. Offer informative feedback
   All actions should give feedback to the user and larger or complex actions should have more detailed feedback.

4. Design dialogs to yield closure
   When an action is finished, the user should be informed about this in a clear way in order to understand that the action was finished.

5. Offer error prevention and simple error handling
   It is better to prevent errors from happening that to handle them afterwards. Since all errors cannot be avoided, they should be simple to recover from.

6. Permit easy reversal of actions
   To help the users to explore the program, it should be easy to reverse the actions that can be made in the program and get back to a previous state without any errors.

7. Support internal locus of control
   The users should feel that they are in control of the program and can influence the outcome of the actions taken.

8. Reduce short-term memory load
   The display should be kept simple so the information can be seen more clearly. If the same information is used on multiple pages, these could be combined or the information should be visible on both.
Norman’s seven principles for transforming difficult tasks into simple ones

This section describes seven principles proposed by Norman for transforming tasks. They are presented exactly as they are listed in the book by Dix et al. [48, p. 283] but are then given a short explanation.

1. **Use both knowledge in the world and knowledge in the head**
   Knowledge can be present both in the world and in the head. External knowledge can help to reduce the memory load of the users and internal knowledge can help experienced users to speed up familiar tasks.

2. **Simplify the structure of tasks**
   Tasks can be simplified in many ways, for example by the use of mental aids, automating parts of them or changing the nature of the task.

3. **Make things visible**
   The system should make clear what it can do and how. When the user does something the system should respond and show the user what happened and why.

4. **Get the mappings right**
   It should be easy to see what the controls of the system do and how much they change it when they are used.

5. **Exploit the power of constraints**
   The use of physical and artificial constraints can guide the users into doing the right actions and preventing them from doing things wrong.

6. **Design for error**
   Humans make errors; the system should be designed with that in mind and help the users recover from them.

7. **When all else fails, standardize**
   If a situation cannot be made natural, then it should be made a standard so the users only have to learn it once.

**Nielsen’s heuristic evaluation**

This section describes ten heuristics proposed by Nielsen to evaluate interface design. They are presented exactly as they are listed in the book by Dix et al. [48, p. 325] but are then given a short explanation.

1. **Visibility of system status**
   The user should be informed of what happens in the system, how long it should take and how much of the task that is complete.

2. **Match between system and real world**
   Instead of giving feedback to the user in terms of variables and other system specific details, it should be given in language that someone who does not have full understanding of how the system works can understand.

3. **User control and freedom**
   If a user make an unwanted action, it should be possible to go back to the previous state or redo the last action.

4. **Consistency and standards**
   Terminology should not change and have a new meaning in different parts of the program or system. The terminology should not be used in a way that is different from the standard way either.
5. Error prevention
   It should be hard for the users to do something wrong with the system. It is better
to prevent errors than to help the users recover from them.

6. Recognition rather than recall
   Things should be visible so the user does not have to remember data from one part of
the program when working in another.

7. Flexibility and efficiency of use
   It should be possible to change how actions are done. Experienced users should have
the possibility to do actions in a faster and more efficient way than inexperienced
users.

8. Aesthetic and minimalist design
   The amount of text in a dialog should be as short as possible so the relevant information
can be made more visible.

9. Help users recognize, diagnose and recover from errors
   When an error occurs, it should be easy to understand why it happened and how to
recover from it.

10. Help and documentation
    The system should have some sort of help function or documentation. This information
should be focused on the user’s point of view, allow searching and provide examples
on how to carry out an action.

6.2.2 Guidelines for risk analyses

The guidelines for risk analyses were collected from various sources in the literature study
and a paper about risk analysis from Ciechanowicz [52]. The literature study was the base
for the combined and extended model described in chapter 5 but also contained some guide-
lines that could be used for the design. The first guideline is from the Swedish Emergency
Management Agency and is that the assessment of threats, risks, vulnerabilities and protec-
tions should be followed by how reliable they are [12]. If the assessment is taken from data
coming from recent practice it should be highly reliable, if there is no data to support the
assessment, its reliability might be lower. The second guideline is from the Legal, Financial
and Administrative Services Agency and deals with the analysis of the difference between
theory and practice [25]. The third guideline is also from the Legal, Financial and Adminis-
trative Services Agency and is that risks should be sorted after priority and the threat they
make to the system [25]. The last guideline is how attack graphs can be used to analyse a
network to find unknown vulnerabilities as was described by several authors, for example
Schneier, Ramakrishnan & Sekar and Lippman & Ingols [36, 42, 35]. This could be used to
extend the analysis of the network and help find new and unknown vulnerabilities that result
from the interaction between different components or the abuse of several vulnerabilities.

Ciechanowicz lists a number of requirements and problems related to risk analyses [52].
The requirements deal with things that a risk analysis method should support, common
problems, and functionality that an implementation should have. These requirements can
be used as guidelines for the design and are grouped together into a list and presented below.

1. The method should be thorough enough to be able to identify all major risks. It should
also be able to identify the cost of the security measures, their downsides and be able
to use during the development of a system.

2. The method should be usable by different types of users, not only security experts. It
should fit the culture of an organization, it should be possible to change the terminol-
ogy, and it should be possible to update it when new technology is introduced.

3. The method should handle dependencies, should be repeatable and it should be possible to change the level of detail. The method should be fast and the results should be saved in a database so they can be combined with each other.

4. The method should support scenarios, statistics and it should be possible to identify the largest security problems.

5. The method should ensure that the users think through why they put in data in the method and understand how the result was calculated. Even a simple method can produce complex results and if the users are just filling in data without thought, the result can be unreliable.

6. It should be possible to reuse results from an earlier analysis and export them to different types of reports.

7. It should be possible to make changes and add more information to the database but data from an analysis should at the same time be protected so it is not changed, is available and can be trusted. It should be possible to freeze the results from an analysis so they are not changed after the analysis is finished.

8. The method should support that an organization sometimes need to make a decision that violates the recommendations presented by the method.

9. Even if the method recommends a counter measure to a problem it does not guarantee that the security becomes higher since there can be more unidentified problems.

### 6.2.3 Requirements from the Swedish Armed Forces

During a seminar in the spring 2009 with representatives from the Swedish Defence Research Agency, the Swedish Armed Forces and the Swedish Defence Materiel Administration some issues with NTE and the plugin EASTER were identified. The following list is a summary and explanation of these issues that the representative from the Swedish Armed Forces and the Swedish Defence Materiel Administration had with NTE and EASTER, as they were described by Jonas Hallberg [23].

1. The security demands that are implemented in EASTER should not be represented as a checklist that can just be ticked off. It should be possible to motivate why one or several demands are fulfilled like it is with the context profiles that can be created in EASTER. This is also more suitable and compatible with the new version of the KSF that are under development.

2. The program should help the user to decide if a security demand is fulfilled or not, which is hard to do at present.

3. The program should be a support when developing demands for a system.

4. The program should support evaluation and testing of an IT system. It should be possible to model the system in the program and then test how secure it is and what demands it fulfils.

5. The program should help validate how reliable a system is and validate a future security level. This means that the program should be able to simulate the difference between the level of security a system can have and the level of security it actually has.

6. The program should be able to identify weak points in a system and decide which components that have a critical role in it.

7. The program should explain the role of security for the architects of IT systems. This means that the program should help the architects and developers to think about the security from the start when designing a system.
6.3 Design proposal

The design was based around the concept of three views: one for text, one for the risk matrix, and one for the graph representations of the system. The combined and extended model described in section five should be implemented as a process divided into several steps that relate to the steps in the model and use all three views. The three views were based upon the three alternatives for presenting the result of the analysis in appendix eight of the Swedish Armed Forces risk management model [1]. The three views should be linked together and information about the analysis should be available in all three views. The amount and the type of information will be different though. The idea of linking the views together and presenting information from one view in the others were based upon the work done by the Swedish Road Administration by presenting the information of their risk analysis as a map where further information was linked to the risks presented on the map [20]. The supervisors for the master's thesis at FOI have also made comments that have helped to improve the design proposal.

6.3.1 Three views

This section describes the three views in more detail. The design of the text view was based upon the guidelines from Ciechanowicz, Norman, and the literature study in chapter 3. The matrix view was based upon the Swedish Armed Forces risk management model and the report by the Swedish Road Administration. The graph view was based upon the requirements from the Swedish Armed Forces, the guidelines from Schneiderman, Norman, and the literature study in chapter 3.

Text view

The information in the text view should be presented as a list divided into different categories. It should be possible to choose what to display and switch between threats, assets, security measures, and risks. The information in the text view should be linked together and it should be possible to view related information to what is currently viewed on the screen. For example, it should be possible to select a threat and then view the related assets, security measures and risks. The text view should also be able to represent what cannot be represented in the other views. This might include the basic values for the analysis, users, constitutional analysis, counter measures and follow-up. The user interface should be designed so it is easy to switch between different types of information. This can be done by using tabs and open new windows to present the related data. The risk analysis should be presented step by step to reduce the amount of information that is presented at once. It should also be possible to know where in the analysis the current step is and how many steps there are left to do. During the analysis it should be possible to see the result of the previous steps and information that has been submitted when performing other steps in the analysis. By keeping information available in the program, the user's task is simplified and can be made faster. More experienced users should be able to combine this support with knowledge of the risk analysis and the program to perform the task even more effectively.

If the result of the risk analysis is well documented, it is easier to repeat the analysis. To enforce this, all decisions made in the model should be motivated and the result of the analysis should be possible to export to a report. Information that is put into the program should be motivated so the users think through what they put in and why. To change the level of detail and make the analysis faster, it should be possible to skip some parts of the analysis or import them from a previous analysis. If a step in the analysis is skipped, the
decision should be motivated so it is clear later on why this was done. To ensure that no critical steps of the analysis are skipped, constraints can enforce that some steps must be completed before the analysis can continue or that the user is well aware of this. Sometimes, the organization might need to make a decision that violates the recommendations that have been presented in the analysis. This should be supported, but it needs to be motivated why it is done and why the scale of consequences does not support this specific decision.

For the risk analysis it could be useful to include scenarios when analysing the threats. For each threat there could be a scenario that describes how the threat could affect the system. The threats, vulnerabilities, and risks could also be sorted according to name, type or how severe they are. The ability to include how reliable an assessment of a threat, vulnerability, risk, or security measure is could also be added to the text view. This function together with the analysis of the difference between theory and practice could be an optional part of the analysis. These could be added by selecting them from a drop-down list with pre-made questions or structures to use in the analysis. It should be possible to add new questions to the text view or structure the data as the user see fit. It should then be possible to save these as structures to use them later in the analysis or export these changes to use as a template for another analysis.

Matrix

The matrix should represent the risks with probability on one axis and consequence on the other. The information in the matrix should be collected from the data that has been collected earlier. It should be possible to see if a threat affects several assets by showing this as separate objects in the matrix. It should also be possible to see which security measures that affects a risk and choose how much information that should be presented. When clicking on risks in the matrix, information about the risk should be shown. This could include the related asset, security measures, the threat value, vulnerabilities, and other information.

There are two possible ways to connect assets with threats in the risk analysis: the first is to use a matrix and the second is to use a drop-down list. The advantage of the matrix is that it is easy to add relations between assets and threats, especially if a threat affects many assets or many assets are affected by a single threat. The disadvantages are that if there are many threats and assets there might be hard to find the right element in the matrix and that it might be hard to see the connection between threats and assets as a matrix for an inexperienced user. Another disadvantage is that unless all connections between the threats and assets are done at the same time, there will be a lot of switching between text and the matrix view. When working with threats it might seem more natural to map them to assets one by one instead of do all the mappings at the same time. The advantages of a drop-down list are that it is possible to make these mappings when the threats are identified and that it is commonly used in user interfaces. A disadvantage is that it is hard to do a mapping between a single threat and many assets, since a new drop-down list has to be added dynamically and filled in as a separate action each time. As there exist advantages and disadvantages for both a matrix and a drop-down list, a solution would be to include support for both and let the user choose which one to use.

Graph

The graph view should be used for modelling the system and keep all the current functionality that exists in the plugin EASTER, but it should be made more clear how to add new entities and relations. In addition to that, the system that is modelled in the graph view should be connected to the rest of the analysis and it should be possible to view information
that is related to the entities in the graph. If a threat is modelled in the graph, it should be possible to view the related assets, protections, vulnerabilities and risks for this.

One type of functionality that should be expanded is how security demands are fulfilled. In the EASTER, they are presented as a list of checkboxes that the user selects from, but it would be better if the user had to motivate why each demand is fulfilled instead. Each security demand can be given a specific motivation why it is fulfilled or they can be gathered together in a context profile that represents a solution that motivates why certain demands are fulfilled. When the user motivates why a demand is fulfilled, a form of pre-defined demands, criterions, or free text could be used. To ensure that the user does not create a system that includes a breach in security, there can be a constraint in the program that forces the user to motivate why an entity or a relation that breaches security should be added. In addition to this, some alternatives to how this could be avoided should be presented.

To be able to present the requirements that shall be met by the system more clearly, there could exist a set of well defined requirements for each security requirement, but on a lower level of abstraction. These requirements can then show what needs to be done if the system should fulfil the security demands for a certain level of security, but on a more physical level. An example of this could be that in order to fulfil the demand physical security, the detailed demands could be security cameras, access cards, and security guards. This information could then be presented as a detailed list with measures to be taken for the system if it does not fulfil these requirements. The list could also be exported or printed to use as a checklist when carrying out these measures to improve the security of the system.

Another function that should be added is to represent threats and weaknesses as profiles that can be connected to entities, relations and be represented graphically in the model. This could be done by having context profiles that contain not only security demands but also threats or weaknesses that lowers the level of security in the modelled system. This could be used to simulate that the operational security of the system is lower than it should be or other events that affects the security, for example a weak physical security. This could be combined with a graphical representation of weak or critical points of the system that could be shown in the model and be used for validating the reliability of the system.

To support evaluation and testing, an existing system could be modelled in the program. This system would then be tested so any vulnerabilities or weaknesses can be identified. These weaknesses, where they exist and how they can be solved could then be listed. Any measures done to the system to solve these problems could then be added to the model and evaluated again. To improve the testing of systems in the program, it could be expanded with the functionality of attack graphs. This might lead to a better analysis of the weaknesses in the system but would require large changes to how the system is modelled and evaluated.

The way the system is modelled in EASTER at present is mostly to check if all entities fulfil the security demands that are specified in the Requirement Collection and that no networks of different security classes are connected to each other. The plugin SANTA has more functionality but to include support for attack graphs, the system must be able to model a more realistic network and not only a network on a higher level of abstraction.

A way to expand the system that does not require as much changes as an implementation of attack graphs is to model it more closely after the security demands. Instead of motivating why a demand is fulfilled as text, the entity itself could be changed to show this. Each of the specific demands for each security demand mentioned above could be modelled as add-ons to the entities instead of being shown as a list. For each demand, there could be one or more functions that could be added to the system in order to fulfil it. For example, if the demand states that the system needs to be secured against computer viruses, two functions could be added: an anti-virus program and scanning of files sent by mail. These could
be added to single computers or larger networks. New add-ons could also be created and connected to security demands when needed. When the system is tested, both the added functions and the motivation in text should be considered. When the system is expanded in this way, the result of the analysis will be a bit different. Instead of producing a list of the requirements that the system does not fulfil, it could make a list of more practical changes that needs to be done to the system. If this method of testing could be used by the developers of the program, they could see at an early stage what level of security the finished system will fulfil. This should help them to think more about security during the development and therefore develop a better system.

6.3.2 General design decisions

This section describes the more general design decisions that affect the whole program or include functionality that are part of more than one of the three views. These decisions were based on the guidelines from Schneiderman, Norman, Nielsen, and Ciechanowicz.

When the program or a new analysis is started the analysis should start automatically. If an analysis exists when the program is started, it should continue where it was last time and it should be clear to the users that the analysis is not finished and that they continue on a previous analysis. During the analysis it should be possible to check the result of the information that has been submitted so far and then continue with the analysis. When the project is opened the user should have the option to continue the analysis or choose to check the result so far in one of the three views. When the analysis is finished, the program should clearly state that it is finished and present the user with some options of what to do next.

It should also be possible to freeze the result of a finished analysis so it cannot be changed afterwards. If the result needs to be changed, it has to be exported to a new project. This is similar to how Requirement Collections are handled: when a collection is modified it cannot overwrite the existing one but must be saved as a copy. When the result is exported it should be possible to choose if it is the structure and the result of the analysis or only the structure that should be exported. It should also be possible to choose which parts of the analysis that should be exported. The result of the analysis should be possible to export in a format suitable for a report. This should support the document formats that are commonly used. If a project with a finished analysis is opened, the result should be presented automatically.

In order to support the many different types of threats, risks, and other parts of the analysis, the structure of the analysis itself must be flexible. It should be possible to add new questions and change some parts of the structure of the analysis when the program is running. It should also be possible to go back to an earlier stage to add or change the data without changing what has been done in later stages. Some parts of the analysis depends on that the earlier stages are finished and the structure should ensure that these parts are finished first. In some parts of the combined and extended model there exist parallel stages. These should be possible to do in any order but later stages might require that all of them are done before the analysis can continue by restricting the access to the later stages. It should also be clear to the user which stages of the analysis that have been finished, which stage that is being done at the moment and which stages that have not been started yet.

Feedback is an important feature and could be improved and implemented in several ways. One way to do it is to show the last command that was executed. This is done in EASTER but could be expanded for more commands and provide more details about them. Another way is to make sure that all dialogs and buttons in the program reflect what they do and what happens if the user presses them. A third way is that all errors should be explained in language that a user can understand and it should also be explained why an
error happened in the same way. In the program, issues should be explained in common language or in terms of the risk analysis instead of limits with the program or its functions.

NTE and EASTER does not permit reversal of actions that have been done. If an error was made to the modelled system or the analysis that the user wants to recover from, the only option would be to close the project without saving and reload an earlier copy from the database. Since errors will happen, this support should also be added to the system.

It should also be possible to get a short introduction to the risk management and the IT lifecycle model through the use of tool-tips or as a help function in the program. The help function should contain information about the models, what the different steps require, and how they are used in the program. It should also list what the program can do.

The buttons, windows, menus, or other parts of the user interface should look and behave the same in the whole program. If one window can be closed by clicking on a button in the top right corner, than all windows should have this functionality. Shortcuts are also useful, especially for the more experienced users. Functionality that could benefit from this is the three views in the program. If all else fails, the program should be made to look like other similar programs or problematic steps should be explained when they are used for the first time and then used with consistency.

6.4 Implementation of the combined and extended model

The combined and extended model consists of several steps for identifying and assessing threats, vulnerabilities, and risks. This section describes how the input of data, a way to analyse and to visualize the data can be implemented based on the three views and the general design issues identified.

6.4.1 Input of data

The input of data should be done as an implementation of the combined and extended model presented in chapter 5. The different steps in the model are described with the tasks that needs to be performed or the questions that needs to be answered at each step. Unless specifically described, the standard view for submitting data is the text view.

Step 1 - Basic values for the analysis

Define basic values for the analysis like:

- Who shall perform the analysis?
- Where shall it be done?
- When shall it be done?
- Why shall it be done?
- What is the maximum cost for the analysis?

When the basic values are answered it needs to be decided if the analysis should be done at all. If this question cannot be answered at this stage it should be left until it can be answered in a later stage. When this is done a scale for the consequences to the system should be defined. It shall have ten steps and both the scale and the steps in it should be defined and are used later in the analysis. The last part of this step is to identify which type of threats that exists for the system. These will then be analysed in greater depth later.
Step 2 - Assets that are worth to protect

In this step, the assets that are important should be identified and analysed. They should be described so it is clear which type of asset it is and how important they are for the organization. An example of assets could be personnel or trust.

Step 3 - Analysis of the activity

Both the existing and future activity should be identified and analysed. This includes describing how they work, and identifying both resources and functions. The existing activity should be identified as it can be used as the foundation for the analysis of the new activity. Like the existing activity, it should be described and the resources and functions should be identified. The demands that the activity should fulfil should be identified as well as any changes it might bring to the organization and work procedure.

Step 4 - Users

Since the users are a central part of the design proposal, an extra step should be added to the implementation. In this step, the users of the system should be identified, both the users at present and in the future. They should be described so their level of access, education and the goals they have with or through the system can be identified. At this stage personas could be developed to aid the analysis. A persona is a detailed but artificial user for the system that can be used for testing and evaluation. More information about personas can be found in the paper by Grudin & Pruitt [53].

Step 5 - Information and security analysis

In this step the information that is in or shall be in the system should be identified. The security class of the information and if there exists several classes of information in the system should be identified as well. For each type of information in the system, it should then be decided if the information should be classified as secret and what the consequences would be if it was revealed.

Step 6 - Preliminary system description

The new system and how it is supposed to work should be described in this step. The demands that have been identified should be analysed and those that cannot be realised should be removed. The parts of the system that cannot be described graphically should be described in text. These might include for example demands for usability, policies, operational demands, and education for the users. Those demands that can be described graphically are then modelled in the graph view.

Step 7 - Evaluate the preliminary system description

The preliminary system description that was constructed should be evaluated from several perspectives. Some of these are: personnel, economy, technology, activity, organization, infrastructure, and support.
Step 8 - Threat analysis

The type of threats that have been selected in the first step should be identified and analysed further. The threats that have been identified in the modelling of the system should also be included. These threats should be broken down so they can be described as an unwanted event, an occurrence or a modus for attack. These should then be assessed on a scale of five on the basis of intention, capacity, and occasion. A single threat can affect several assets and an asset can be affected by several threats. The mapping can be done with a matrix or a drop-down list. The matrix should have threats and assets on separate axis that can be connected by marking elements in the matrix.

Step 9 - Constitutional analysis

The laws and regulations that affect the system should be described. This should include which law or regulation it is, how and which part of the system they affect and what this leads to for the system.

Step 10 - Analysis of security measures

The security demands on the system should be formulated based on the analysis of the activity, the constitutional analysis and the demands from the KSF. The existing security measures should be identified and compared with the security demands.

Step 11 - Assess vulnerabilities

The security measures that have been identified in the previous step should be compared with the threats and several security measures can affect a single threat. For each threat it should be described how the security measures affects it, if it lessens or eliminates it. If there exists an older or similar system, it is possible to conduct penetration testing to evaluate the vulnerabilities in the system. The result of the comparison is the vulnerabilities of the system. These are assessed on a scale of five on the basis of security awareness, exposure, and resources.

Step 12 - Estimate the risk

The risks are a combination of probability and consequence for the threats that have been identified. The estimation of these should result in a value on a scale of ten and a motivation for this estimation should be presented. Threats that affects several assets should be treated as separate risks. The risks are then given a risk value depending on the probability, consequence, and their placement in the risk matrix.

Step 13 - Follow-up

It should be decided and motivated which risks that are acceptable and which needs to be taken care of. For the risks that have to be taken care of, counter measures have to be identified. The counter measures should then be analysed so they do not introduce further risks. The final part is to decide when a follow-up to the analysis should be made and if the results of the analysis were expected.
6.4.2 Analysis of data

The result of the analysis described above can be seen as an analysis of data but the analysis is mostly done by hand and the data is then submitted into the program. To analyse data in the program, the matrix or graph view can be used. Risks presented in the matrix can be analysed as patterns might show up in the matrix that point out that risks are related, that there are serious issues with the security, or that the scale of consequences is badly defined. Systems in the graph view can be analysed to a higher degree. It should be possible to add an existing system to the program and analyse if there is a breach in security. It should also be possible to choose different levels of security around the system to simulate different levels of security during operation or threats to the system. This should then point out the critical parts of the system and lead to a list of vulnerabilities to take care of. This type of test can be done both as a part of the larger analysis or on its own. If it is done on its own, it might miss many of the points found in the larger analysis but can be useful when there exists a system and an earlier analysis has been made. It could also be used if the system has changed or as a follow-up to the larger analysis.

6.4.3 Visualization of data

Data should be possible to visualize in several ways. The first is as a graph with entities and relations, the second as a matrix and the third as a number of fields with data. In all three of the views, it should be possible to view information that is present in the others. In the graph view it should be possible to get more information about the different assets, threats, protections, vulnerabilities, and risks. In the matrix view it should be possible to get information about the related threat, asset and security measures. It should also be possible to see which assets that are affected by a particular risk and see the related security measures in the matrix view. In the text view it should be possible to get more information to related data about threat, risks, vulnerabilities, assets, and security measures.

Of the three views, the matrix view is the one that is most focused on visualization. The primary goal of the matrix view is to present data about the risks in a way so it is easy to see how they relate to each other and which risks that are most important to take care of. All information cannot be presented in all three views as they are better at some types of information than others. The aim of the proposal is to enable so as much information as possible can be presented in all three views and it should be possible to export the result to external programs in forms that are suitable for reports. This will not be a complete report but the basic structure for one and ensures that all reports for the system are similar.

When the result of the analysis should be presented, the matrix view is useful to get a quick overview of the result of the analysis. The text view contains more data but it is harder to get a good overview of it and the graph view focuses on a narrow field of the analysis and does not present any risks. From the matrix view the risks are clearly shown and it should be possible to view more information related to each risk. If a more detailed presentation is needed, it should be possible to show the data in the text view.

6.5 Evaluation of NTE

NTE and EASTER contains several functions that are of use to the implementation. In this section, these are listed and an evaluation if NTE and EASTER should be used for the implementation is presented. The useful functions in NTE and EASTER were the modelling of a system through the EASTER plugin, the database and file handling, the
ability to export a project, the Requirement Collections and the ability to add new plugins. The modelling handled many of the features that were useful to the implementation of the combined and extended model described in chapter 5. The ability to save the analysis in a database and that NTE handled the functionality to save, load, export, and delete were all essential to the new implementation, as were the Requirement Collections. Since NTE allowed the use of plugins, it was possible to make a new plugin if the functions in EASTER would not be useful and still keep the functionality of NTE. The largest things that NTE and EASTER lacked were the ability to view the data in three views, the ability to export the analysis to external formats and a more detailed modelling of the system. The other types of functionality that had been presented in the design proposal were minor and were extensions of functionality in NTE and EASTER. Some of the functionality could also be implemented by extending and modifying parts of the plugin SANTA, as the code was unused but still part of EASTER. To expand it for an implementation of attack graphs was a larger issue and it was not clear if EASTER was suitable for this functionality, how it should be represented or implemented. To include it would be a huge benefit, but this issue required further analysis about the construction of attack graphs. To add security requirements as functions to the entities could be seen as a way to expand the graph view more towards the attack graph extension, but would require less work. In order to be able to implement this, the entities needed to be modelled closer to real computers with representations for firewalls, programs with known weaknesses, operating systems, open ports, and other details. In EASTER, the entities just fulfilled security requirements and the system checked for connections between networks of different security classes, but the plugin SANTA might be modified for this.

If a new program would be developed, most of the functions that were present in NTE would have to be implemented again. If the implementation of the model would not need the functionality of a database or use Requirement Collections, then another program might have been more suitable. However, the implementation of the model needs these components and the Requirement Collections that are implemented are based on those from the Swedish Armed Forces. Therefore NTE was suitable as a basis for the implementation since only minor parts of the program needed to be changed. These were some inconsistencies with the user interface and some issues with the stability. The plugin EASTER had much of the functionality for modelling graphs that were needed for the implementation. For most of the changes that have been suggested, there were only minor issues with EASTER. EASTER was therefore suitable for the implementation of the three views as all parts except the menu bar of the GUI in NTE could be modified by the plugin.

### 6.6 Functionality that should be implemented

From the design proposal, the proposed implementation, and the analysis of NTE, a list of functionality to implement was constructed. The list was then divided into two groups: the features that were suitable to implement as a part of the master’s thesis and those that were not. From the list of features that were suitable, a few was selected and implemented.

The first list of features were those functions that were possible to implement as part of a master’s thesis. The most central of these were the three views and the ability to do an analysis. Other features included that the analysis should be flexible and different ways to expand the graphical representation the system.

- Three views: a graphic representation of the model, a text based view for putting in and viewing information, and a matrix to represent risks. The three views should be connected and the information should be accessible from all three.
It should be possible to start an analysis that is taken step by step to a conclusion and uses all three views.

It should be possible to add new questions to the analysis and not only answer those that exist.

It should be possible to view the result before the analysis is complete and then continue the analysis.

Every requirement in the KSF that the system fulfil should be motivated. This can be done by filling out a pre-defined form or making a new motivation in free text.

It should be possible to represent threats as a physical relation profile for the entities and relations in the graph. It should be possible to see graphically which entities and relations that the threat affects. Several threats could be handled by separate colours. To avoid making the graph to crowded with information it should be possible to turn off this feature.

It should be possible to present the information from the analysis in a structured and easy way, suitable for presentations.

It should be possible to present the security requirements that are unfulfilled.

The second list of features were those functions that were not suitable for implementation as a part of a master’s thesis. Some of these were larger features, like the ability to export the data in the program to different formats for reports. Some were trivial but time consuming, for example the adding of tool-tips and the creation of a help function. Others were more complicated, like the extension of the graph view with the functionality of attack graphs.

It should be possible to save the structure of an analysis to use it later. The questions, criterions, checklists, level of detail and other things should all be saved.

It should be possible to see where the weak or critical points of a modelled system are.

It should be possible to model security demands as add-ons to the entities in the graph.

The result of the analysis should be possible to export to another analysis.

It should be possible to export the result of the analysis to an external program for the basis of a report.

There should be more detailed tool-tips in the program to give the users more support. These should give information about the program and the risk management model.

There should be a help function that gives more detailed help with the analysis and refer to the original documents for further reading. The help function should be searchable.

It would be good to extend the graph view with the functionality of attack graphs.

From these lists, a few features were selected for implementation. These were the three views, the ability to do an analysis, the ability to modify the structure of the analysis, and the ability to view the result. These features are explained in more detail in chapter 7. From the remaining features, two mockups of the new design were constructed, these can be seen in Figure 6.4 and Figure 6.5. These were constructed to show how things might look for some features that would not be implemented as a part of this master’s thesis. The first mockup shows a way to motivate each security demand that the entities in the graph view should fulfil. The second mockup shows how the critical points of the network and threats can be modelled in the graph view with the help of profiles.
Figure 6.4: A mockup of how the security demands can be motivated in the program by presenting the user with a form to fill in and submit.

Figure 6.5: A mockup of how critical point and threats can be represented in the graph view of the program.
Chapter 7

Implementation

The implementation presented in this chapter is based on the features that were selected in chapter 6 and is done as an extension to the plugin EASTER to the program NTE. At FOI, it was decided that the new implementation would be called ASCENSION. The features that were implemented were the three views, the ability to do an analysis, the ability to modify the structure of the analysis, and the ability to view the result of the analysis. The implementation was done in Visual Studio and in C# because the existing program that this implementation extended was developed in Visual Studio and C#. The implementation of NTE and EASTER were stored on a CVS server so the new one was developed as a branch to this. This had the benefit that it would be easy to return to the original code if needed and the new code could be re-introduced into the trunk at a later stage. The implementation EASTER can be seen in Figure 7.1 that shows how the different classes were connected. As can be seen in the figure, there were many classes that were unused. These were remains from the plugin SANTA and contains code used for calculations and different type of profiles.

![Diagram of EASTER implementation](image_url)

**Figure 7.1:** A diagram showing how the implementation of EASTER looks like and which classes that are part of it or are disabled.
7.1 General description of ASCENSION

This section describes the implementation from a general perspective and lists the different parts of it. The parts work as add-ons to the original code and were in most cases implemented as separate classes or as partial classes to existing ones. The exception from this were the GUI and the connection to the database plugin. These were implemented in the main class of EASTER and were coupled more tightly to the original implementation than the others. The new implementation ASCENSION can be seen in Figure 7.2 that shows how the extensions are part of the old implementation. Another view of this that focus more on the inheritance between the classes can be seen in Figure 7.3. An expanded version of this diagram can be found in Appendix B and lists the methods in each class as well.

![Diagram showing the implementation of ASCENSION](image)

Figure 7.2: A diagram showing how the new implementation is connected to the existing classes in EASTER, keeping the same layout as Figure 7.1.

![Class diagram of added classes](image)

Figure 7.3: This is a class diagram of the classes that have been added to the implementation of EASTER.
7.1. General description of ASCENSION

7.1.1 Data structure

In the program, data was stored in a data structure that was based around several generic Hashtables and ArrayLists, called Dictionaries and Lists in C#. In the Dictionaries, the data was stored as a pair of a unique key and a value. The key in this implementation was a Guid, a Globally Unique IDentifier. This ensured that all objects stored in the data structure were uniquely distinguishable from each other. The use of a Guid as the key also ensured that no key would be the same and kept track of this automatically. The value in each pair was an object of the class DataStructure that contained the name of the specific type of data, stored the relations to other objects in the data structure and contained another Dictionary. The relations to other objects were stored as a List of Guids for the objects that were related to the current object and the Dictionary was similar to the others with pairs of keys and values, where the key was a Guid but the value was an object of the class DataField. This class contained several variables and stored the actual information in the data structure as the class DataStructure was more of a container for these sub-objects. The reason for this structure was that the number of objects and sub-objects stored in the data structure varied over time and new objects could be created by the user during run-time.

The objects that were stored in the data structure could be of several types, for example a threat, a risk or an analysis of the users in the system. The sub-objects that were stored in the objects could also be of several types and more types could be added during the analysis. An example of the types of sub-objects that existed from the start of the analysis were the name of the object, a description of it or a value that represented it. The sub-objects were also more tightly connected to the GUI than the objects in the data structure. This was because many of the variables stored there were used to save information related to the presentation of the data in the sub-object in the GUI. This could be, for example the size of a text box, the text in a label that explained what this text box should contain, or if the text box should support multiple rows.

7.1.2 GUI structure

Since the number of components in the GUI changed over time and could be added or removed during run-time, they could not be static but had to be added dynamically. In the implementation, this was done through methods that added new components when called upon by events in the GUI. In order to keep track of and store these events a data structure for the components in the GUI was constructed. When Panels, Labels, Buttons, and other objects were created, they were added to this structure. The structure was a Dictionary with a Guid for the key and an object of the class GUI_Structure for the value in each pair. The Guid for each object was the same as for the corresponding object in the data structure to make it easy to access the object when it needed to be changed or saved. In this way, it was possible to modify the structure of the GUI by adding or removing components and keep the connection to the data structure as this was changed when the system was saved.

In the same way as the data structure, the structure of the GUI contained a Dictionary in each object in the Dictionary. For this Dictionary, the pair of keys and values were a Guid as the key and an object of the class InformationObject as the value. Each object of the class InformationObject was related to an object of the class DataField. The class GUI_Component kept track of all the sub-objects that made up an object and contained methods for creating components for the GUI depending on what data that was loaded from the data structure. For example, if the type of sub-object in the data structure was a String, the sub-object created in the GUI would contain a RichTextBox. These classes also kept track of the event handlers that were used to tie methods to the components in the GUI.
7.1.3 Structure of the analysis

The analysis was structured into 13 steps and each of them was connected to a specific dictionary in the data structure. To make it possible to specify parameters to the save and load methods and separate the type of data that was used in the program, each step and Dictionary were connected with a value in an enumeration; a strongly typed constant. This solution was not the most efficient and since they were separated by an enumeration more steps to the analysis could not be added during run-time. The advantages of this was that it was more efficient to iterate through all objects of a single type since they were separated into different dictionaries, instead of having to separate the different types of objects when working with the data structure. The reasons that the structure was made up from Dictionaries were that they were efficient to search in and that the data structure used unique identifiers to separate the objects. They were not as easy to work with as Lists when iterating through all objects but the search function was used to a great degree when saving data and removing relations.

7.1.4 Methods for reading and saving data

The methods for reading and saving data transformed it from the form it had in the data structure to the form it had in the GUI and then back again. Both methods were general and took a single parameter: an enumeration of the different types of data in the data structure. This was used to select the dictionary that was connected to this type of data.

When data was loaded to the GUI, the method that read the data checked if there existed any data at all. If no data existed, the method called on another method that loaded a pre-made empty structure to the GUI. This empty structure was different depending on which type from the enumeration that was sent to it and was also used when new objects were added. If there existed data, the method iterated through the whole dictionary and then created new objects that were added to the GUI. When the object was created, event handlers for the Buttons in it were created as well. For each object, all sub-objects that existed in it was then read in turn and added to the GUI. Depending on the type of data that existed in each sub-object, different types of components were created in the GUI.

The method that saved data to the data structure followed a similar procedure but in the opposite direction. It iterated through all objects in the GUI structure and then transferred the information from these to the data structure. This started with the objects and then all sub-objects that were part of each object. The method checked which type each sub-object belonged to and fetched the values from the corresponding component in the GUI. When the object and sub-objects were saved to the data structure there were three cases: both of them existed in the data structure, the object existed but not the sub-object, and neither of them existed. If both the object and the sub-objects existed, they were updated, if the object existed but not the sub-object, it was created, if neither existed, they were both created and filled with data. The method also checked if any of the sub-objects contained relations to another objects. If this was the case, this relation was updated as the method checked if there had been any changes, removed old relations and created new ones. This method is described in the section that follows. When the information had been added, the method that saved data also deleted the information in the data structure that had been deleted in the GUI. This was done by comparing the data structure with the GUI structure. If an object or sub-object existed in the data structure but not in the GUI structure, the Guid was saved in a List. The objects or sub-objects that had the same Guids as the the Guids in the List was then removed from the data structure.
7.1.5 Method for creating and removing relations

Each object in the data structure contained a List that kept track of the relations between
the object and other objects in the data structure by storing the Guid of all related objects.
These Lists were only relevant for a certain type of objects, the types that contained assets,
threats, protections, vulnerabilities, or risks. When these objects were created in the GUI,
the Lists were created as well. For the other objects in the data structure, the Lists served
no function and were always set to null. In some cases, these pre-defined relations were the
only relations that could be made. For example, each risk could only have a relation with
a single asset and a single threat. This was enforced as no further Lists could be added
to an object of the type risk. The relations were saved in the List in each object as both
objects needed to know that they were related to the other. This made it easier to see the
relations for a single object, but made the methods for creating and removing relations more
complicated as the information was spread out between two objects in the data structure.

In order to create and remove these relations three methods were needed: one to add a
relation between two objects, one to remove all relations to an object, and one to remove a
relation from two objects. The methods could create many types of relations between the
assets, threats, protections, vulnerabilities, and risks but the GUI limited which relations
that could be made. In this way, it is possible to create other types of relations if the need
arises by changing the GUI instead of the underlying method. How the GUI limited the
relations can be seen in Figure 7.4. This diagram only shows which relations that could be
made, not where in the program they could be made. For example, threats have a many to
many relation with protections as several threats can be affected by a single protection and
several protections can affect a single threat. However, it was not possible to create these
relations from the threats, only from the protections. Most objects also had a fixed relation
with assets, this meant that no further relations than the one that existed could be created
to a specific asset from a specific object.

![Diagram showing the relations between different objects in the data structure and the limitations imposed by the GUI.](image)

Figure 7.4: A figure that shows how the different objects in the data structure are be related
to each other and which limitations that have been made to the relations in the GUI.
The methods for creating and removing relations were not very complicated or very efficient as they contained many nested IF and ELSE-IF cases that connected the right Guid to the right object. In general the methods that created the relation checked if a relation existed and removed old relations that had been removed from one of the objects. If the relation had not changed, nothing happened but otherwise a new relation was created in both objects. For the risks, there was also a method that created secondary relations. Secondary relations only applied to objects of the type risk and made connections to the protections and vulnerabilities that were connected to the asset and the threat that was connected to the risk. There existed two methods for removing relations, one that removed the whole object and all relations to it and one that removed a relation between two objects. The method that removed an object searched through all the other Dictionaries for the objects that were related to the specified object and removed any relations to it. The other method started with the the Guid and type for both objects that were part of the relation, compared these with different combinations of possible relations and then removed the relation by calling on the right methods in the right objects.

7.1.6 Methods for presenting data

Data could be presented in several ways, the first was in the graph view where a system was presented in the form of a graph, the second as a matrix where risks were presented, the third was as text in the regular text view and the presentation view. The fourth was a dialogue window where relations between assets, threats, protections, vulnerabilities, and risks were presented. In all methods that presented text in some way, the layout and the number of components in the GUI was modified based on the number of objects in the data structure. This meant that most of these components were created through the code during run-time instead of being created in a graphical editor in Visual Studio before the program was compiled. This presented some difficulties as the layout, size and other properties were not known before the methods were used during run-time. The matrix was an exception to this as the panels were static and had a fixed position. The presentation of relations in an own dialogue window had the benefit that it could be added to other functions of the GUI. The relations could be added to all views and showed related information to what was being shown. This was not implemented for all views as only the text view and the matrix view had support for this. In the following two sections, the matrix and the text view are presented in more detail.

Matrix view

The matrix was made of 100 panels arranged in rows and columns with 10 panels in each. Each panel was coloured according to the corresponding risk value. White represented (1), green (2), yellow (3), orange (4) and red (5). Every object in the matrix was an object of the class MatrixPanel, which inherited from the class Panel but also contained variables that made it possible to know if one or more risks were associated with it and the Guid of those risks. As the panels themselves had no way of storing their relation to each other, they were saved in a matrix where the index of each object could be mapped to the risks. The index was mapped to the values of probability and consequence, which already existed in each object and made it very efficient to map from a risk object to a position in the matrix. Each risk in the matrix was represented by an image that was placed on the panel that the risk was connected to. When a panel was clicked on, an event was fired that first checked if there was a risk connected to the panel. If there was, a dialogue window was displayed that showed the related asset, threat, protections, and vulnerabilities for the selected risk. In a
menu to the right of the matrix there was a drop-down menu that let the user select which type of risk to show in the matrix. It were possible to view all risks at the same time or to view those that were connected to a specific asset. A screenshot of the matrix view can be seen in Figure B.2 that can be found in Appendix B.

Text view
The text view was split into two parts, the regular text view were data could be submitted and the presentation view. In the regular text view, the data was structured so that all objects in the Dictionary that were shown were placed in their own Panel. These Panels were then filled with smaller Panels for each sub-object in the objects. Each smaller panel then contained a component, for example a ComboBox and a Label. The component was a representation of the actual data in the sub-object and the Label was a description of it. The larger Panels also contained Buttons for adding or removing sub-objects, as well as a Button to remove the object itself. The regular text view also contained a menu that showed the number of the present step, which steps that had been completed, a Button for adding new objects and Buttons for changing steps in the analysis. A screenshot of the regular text view can be seen in Figure B.1 that can be found in Appendix B. The presentation view was structured in a similar way to the regular text view but each category of objects, the whole Dictionary, was placed in the same Panel and each object in the Dictionary was then placed in a smaller panel inside this one. As the sub-objects were only presented, they did not need their own Panel and were displayed as Labels in the same Panel as the object they belonged to. A screenshot of the presentation view can be seen in Figure B.3 that can be found in Appendix B.

7.2 Detailed description of some parts of ASCENSION

One of the more central aspects of the implementation was how the data in the program was organized and structured. This section describes this in more detail, explains why certain data types were chosen, describes some problems related to the structure of the data and how these problems were solved.

7.2.1 Data structure
The data structure was designed so that all sub-objects that were part of an object were equal. This simplified the methods for reading and saving data, but also meant that it was not possible to choose that certain objects should be possible to single out from the rest, as they were all stored in the same structure. This posed a problem when there was a need to access specific sub-objects like name or type and therefore these values were saved both in the sub-object and as variables in the object itself. The same problem happened with the relations, as the Guid of the relation was saved in both the List of relations and in the specific sub-object that kept track of this relation in the GUI. The sub-object needed to save this data as otherwise it was not possible to completely remove or change relations that had been added, as the object could not be singled out from the rest of the data structure.

The data in the data structure was accessed either by searching for it by the corresponding Guid or by the use of an enumerator to iterate through all objects in the data structure. When a single object was updated or accessed for other reasons, it was accessed by its Guid but when the whole structure was needed, for example in the methods for reading and saving data, it was accessed by the use of the enumerator. When enumerators were used, two
of them were needed, one for the objects and one for the sub-objects in the object. The
enumerator for the sub-objects were initialized again for each sub-object and were used in
a similar way as an inner for-loop when two or more for-loops are nested together.

The data structure consisted of two primary data types, Lists and Dictionaries. These
were chosen since they were generic and therefore limited the types of objects that could be
stored in them. If an object of the wrong type was stored in the data structure, this produced
an error during compilation instead of during run-time. When only Guids were saved, a
List was used since its size was dynamic and there was no need for the extra functionality
in a Dictionary. The Dictionary was chosen since it allowed the information to be saved
in pairs with a key and a value. The key was also unique and allowed for identification of
each object in the data structure and the Dictionary had efficient methods that retrieved
the data in them. Guids were used for the identification since they were globally unique
and the implementation ensured that all objects in the data structure had a unique key.

7.2.2 GUI structure

The structure of the GUI enabled that data could be filled in, saved to the right object
in the data structure and to connect different objects to each other. In order to be able
to add, modify and remove information from the objects, each object in the GUI was
connected to the data structure with a Guid. When new objects were created or when data
was read from the data structure, the methods checked the type of the data and created
the right components to display the data in. This ensured that a String was presented in a
RichTextBox, an Integer was presented in a NumericUpDown, a List of values was presented
in a ComboBox and a Boolean was presented in a CheckBox. These components allowed
both for presentation and manipulation of data which was important as data in the text
view was both presented and modified. When data was presented in dialog windows or in
the presentation view, it was placed in labels instead as there was no need to modify it.

Another function of the GUI was that all objects and sub-objects in the data structure
were created from methods there. This was done in two ways: as pre-defined objects that
were created when a new project was created and by the user through the user interface.
The pre-defined objects were created as an empty structure that had a number of fixed
sub-objects from the start. These were hard coded into the code but all code related to this
was placed in a single method so this could be changed in the future. One possible extension
to this was to be able to load pre-defined structures of an analysis in a similar way to how
Requirement Collections were handled in the program. New objects could be created in the
user interface by clicking on a button. This checked which type of objects that were present
in the GUI and created a new, empty object of the same type. The sub-objects in each
object were created by selecting a dialog window that let the user specify the size, the type,
and other properties of the sub-object. In this dialog window, there were some restrictions
to the properties and the program only created a sub-object if all values in it were valid.

For both the objects and the sub-objects, there were event handlers that ensured that
buttons in the objects and sub-objects triggered actions when they were clicked by the user.
Since the number of objects was not known and could change over time, these needed to be
created and added dynamically. When the objects and sub-objects were created, these event
handlers were connected to methods in the class EASTER_Functions, which was the main
part of the new implementation. In order to remove objects and sub-objects, the events
needed to send the Guid of the object or sub-object that should be removed as a parameter
to the method that removed them. This was solved by the use of delegates that made it
possible to access methods in the class where the object or sub-object were created. This
might not be the best solution to this problem and a re-design of the structure of the GUI
could make this unnecessary. However, the present solution worked and made it possible to
remove objects and sub-objects that had been added during run-time.

7.3 Evaluation

The implementation that was made did not contain all the identified features that were
presented in chapter 6, instead some features had been prioritized over others. There were
also issues with both the existing implementation and limitations with new one. This section
describes this and presents a proposal for how the implementation that was made could be
improved.

7.3.1 Features that were implemented

The new implementation contained several new features. Most of these were from the list
in section 6.6 but some were on a lower level of abstraction and more tightly connected to
the actual implementation. This section presents a list of those features that have been
implemented.

1. ASCENSION contains three views, both a text view and a matrix view in addition to
   the already existing graph view. The text view is used to view and submit data to
   the program while the matrix view is used to view risks in a risk matrix.
2. It is possible to do an analysis that consists of 13 steps. The information submitted
   can then be presented and used in the matrix view.
3. It is possible to see the risks that exists as a risk matrix that shows how they relate
to each other and which risk level the risks belong to.
4. It is possible to show different types of risks in the risk matrix based on which asset
   they are related to, or all risks at once. It is also possible to view the relation that
   the risk has with an asset, a threat, protections and vulnerabilities in a dialog window
   from the matrix view.
5. The data in ASCENSION can be presented in a structured way in the dialog windows
   for relations, the risk matrix and the presentation view.
6. It is possible to switch between the different views during the analysis to watch the
   result so far, see the risks in the risk matrix or model a system in the graph view.
7. It is possible to create relations between objects and then view these relations in a
   separate window.
8. ASCENSION contains a dynamic GUI where new components can be created and
   modified during run-time. The components that are created can also be removed
during run-time.
9. The GUI adapts to the created components and adjusts the size of panels after how
   many components a panel contains.
10. The data structure of ASCENSION is connected to the database of the project. The
    data that is submitted can be saved to this database and loaded from it when the
    project is opened.
11. A small addition has been made to the interface NTEToolPlugin that specifies that
each plugin must implement the method SaveSystem. This method is called upon by
    NTE before the system is saved to the database and lets the plugin save any data to
    its internal data structure so it can be saved to the database. In SANTA this is an
empty method but in ASCENSION this calls upon the method saveData that saves the data in the current step of the analysis.

12. It is possible to skip the analysis and go directly to the graph view.

### 7.3.2 Features that were not implemented

Besides the features that was implemented there were many features that were not implemented. These were from the different guidelines and requirements in chapter 6 as well as the list of features that were suitable for implementation in section 6.6.

1. The graph view has not been changed at all as the features that have been implemented are focused on the other views.
2. There is no direct coupling between the new parts in ASCENSION and the graph view that existed in EASTER.
3. It is not required to motivate why a requirement is fulfilled in the graph view or why a step in the analysis is finished.
4. It is not possible to motivate why a step in the analysis should be skipped or to skip a step other than leaving it blank.
5. It is not possible to show threats as a physical relation profiles in the graph view.
6. The matrix view is not used for creating relations between different objects. This was done with a drop-down list instead.
7. The presentation view is not suitable for presentations. This means that the information is not structured in a way that is suitable for giving presentations for a larger audience, for example with the help of a projector and a tool like PowerPoint.
8. The security demands that the system does not fulfill are not shown as a separate list in the graph view.
9. The analysis does not start from where it was last time when a project is loaded from the database and always starts from the beginning, but all data is filled in as it should.
10. There does not exist an end of the analysis that gives the user some choices on how to proceed, for example by starting a presentation or exporting the analysis.
11. The result of the analysis cannot be frozen and can be modified even after the analysis has been completed.
12. The parallel steps in the combined and extended model are not present in the implementation. All steps are done in linear fashion.
13. There are no restrictions that stops the user from going to the next step in the analysis without having finished the current one.
14. There is no restriction that stops the user from creating several risks that are connected to the same asset and threat.
15. Feedback from all actions taken in the program are not shown on the menu bar at the bottom of NTE.
16. The messages that are shown when something goes wrong in the program are not as informative for a regular user as they could be.
17. The system does not contain a function that lets the user undo the last actions.
18. No requirement from the Swedish Armed Forces is completely fulfilled. Requirement 1, 2, 5, and 6 are not fulfilled at all.
19. None of the guidelines from Ciechanowicz are completely fulfilled.
20. All guidelines from the literature study are not fulfilled, but can be by adding more questions to the empty structures that are created in the GUI or by adding more
questions to the analysis by hand during run-time.

21. The implementation has taken the guidelines from Schneiderman, Norman and Nielsen into account but focused on the implementation of the program and not the user interface. Many of these guidelines are unfulfilled.

### 7.3.3 Issues with NTE and EASTER

Some issues with NTE and EASTER have come up during the implementation of the new features in ASCENSION. This section describes the problems, limitations and bugs that have been identified.

#### Problems with the database plugin

During the work with the implementation some problems have been discovered with the database plugin. Most of these were problems that were related to a lack of documentation of the properties that objects had to fulfil in order to be saved in the database while other were problems with the database plugin itself. The database plugin was part of the original implementation and transfers objects into fields in the database when data is saved. This is then reversed when data is loaded from the database. All classes except primitive ones that should be saved had to implement an interface but besides this there were more properties that needed to be fulfilled. The first of these was that all variables in the class needed to be declared in public so the database plugin could read them. This was a problem since it is a principle of object oriented programming that variables should be declared as restrictive as possible to enforce modularization and create weak couplings between classes. The second property was that there needed to be a constructor in the class that took all of these variables as parameters when the object was reconstructed. This constructor needed to take all variables and in the exact order as they were written in the code, otherwise a run-time error occurred. The database plugin could not handle all types of objects and when new types are introduced, it might need to be modified. The database plugin could on the other hand handle all types of objects in the plugin ASCENSION. Another problem was that the plugin could not handle variables that were named \textit{index}, this seemed to create problems in the saving of objects as this name was used by the plugin in some way.

#### Limitations and bugs

The existing implementation of NTE and EASTER had some limitations and bugs, these are described in more detail in this section.

1. The first limitation was that there existed old unused code from SANTA in EASTER. This made it hard to read the code, hard to get an overview of which functions that were used in each class and to know which classes that were used in the project.

2. The second limitation was that it was hard to know what to do the first time a project was started as the user was presented with an empty drawing area. It would be better if it was more clear how to create new entities and relation between them.

3. The third limitation was that the time it took to save the system was much longer when there were entities in the graph view. In ASCENSION a huge difference could be observed when saving a system with or without entities in the graph view. Having a lot of information in the other views did not seem to slow down the save process but only a few entities in the graph view made it much slower.
4. The fourth limitation was a random crash that occurred when saving a project, closing it and then opening it again. This caused the order of which the objects from the database was read to be scrambled up and caused an exception as the database plugin tried to cast them to the wrong type. This error did not happen every time and it was possible to do the same sequence of action 2, 20, or perhaps 40 times before it happened. The cause of this error is unknown.

5. The fifth limitation was a limitation in the way the plugins were handled, as if the plugin was changed, the program would crash if an older version was loaded from a project file.

6. The sixth limitation was a bug, as Visual Studio created code for the GUI automatically and this conflicted with some of the changes that had been made manually. This had to do with that all parts of the existing implementation were part of the same namespace and this name was added to the code when something was changed in the graphical editor. This then caused a syntax error.

7.3.4 Limitations of ASCENSION

The new implementation ASCENSION had several limitations as well, some were simply features that were not implemented and others were more serious and might require larger changes to the program. These limitations were different from the list in the previous section as they were tightly connected to the implementation of ASCENSION, were not part of the list of features in section 6.6 and were in general on a lower level of abstraction.

One Dictionary per page in the analysis

The first limitation was that only one Dictionary could be used for each page in the text view when data was submitted. All the objects that were shown had to be of the same type as the methods for saving and loading could only handle one type at a time. When the objects were only presented, they were static and were not affected by this limitation. This meant that multiple Dictionaries could not be used for one page in the analysis. This limitation did have the benefit that it simplified the process of saving and loading as the methods only needed to handle a single Dictionary.

Hard coded steps in the analysis

The second limitation was that the type of data connected to a specific Dictionary that was shown for each step in the analysis was hard coded. This was closely related to how the number of steps in the analysis was defined. The number of steps in the analysis was hard coded as they could not be changed during run-time. This was done by specifying the number of Dictionaries by hand and by enumerating the different types of data. It would be possible to create new Dictionaries, add them to a List and connect them to a step in the analysis during run-time. This might be a step in the right way but then there would be a collection inside a collection inside a collection and the real limitation was the enumeration. This could not be changed and the only way to get around this would be to separate the different types of data in another way. The data types that were used in the analysis were also hard coded by the use of an enumeration. This made it easy to separate them and ensure that all types were of those that were pre-defined. This also created the limitation that new types of data could not be added to the analysis other than by modifying the source code.
Data stored in several places

The third limitation was related to the second one and was that the code related to the number of data types in the analysis was stored on several places: both in a class for class definitions, in the method for saving, in the method for loading and in the method for creating components in the GUI. If the number of data types was changed this had to be changed in all four places and if it not, the program could crash if an object was casted to the wrong type as the program did not cancel the method if something went wrong. This error only happened when the source code had been modified and could not happen during regular use of the program. This crash could be avoided if the method for saving data would store the object that should be saved in a temporary variable and validate it before it was saved in the data structure. At present the object is saved piece by piece and an error when saving one of the sub-objects could make the whole object invalid.

No inheritance in the data structure

The fourth limitation was that when a sub-object in the data structure was created there were many parameters that only had uses for certain types of sub-objects. In this way the objects contained more information than necessary as some parameters were unused. One way to solve this would be to create new classes that inherits from the class InformationObject and only uses the parameters that they need. Another related limitation with the objects was that all sub-objects were created equal, even though certain types of them, like the name and the type of the object should be easier to distinguish. In the way the data structure was implemented these objects could only be separated by looking through all objects and comparing a label with a hard coded String. This could introduce errors and was not a very efficient solution. The same situation happened when a certain object needed to be identified, like the values for the probability and consequence in a risk. In that case, the method iterated through all sub-objects in the risk object until it found the values by comparing the name of the label. If the label would be changed or a new sub-object with the same name would be added, the method could retrieve the wrong value or none at all.

Structure of the GUI

The fifth limitation was how the components were stored and presented in the GUI. The most visible one was that when components were removed, the whole GUI was saved, erased and created anew. If a scrollbar existed, this was especially visible as the whole view was moved to the top of it. If the GUI was not erased when components were removed, there would be holes between the remaining components as they would not be moved to the position that was held by the one that was removed. The best would be to be able to remove components and replace the existing ones without erasing the whole GUI every time. This should not be a new problem and might be solved by creating a new way to store the components and using methods that exist in C# or .Net to be able to make better updates of the GUI.

Components of the GUI

The sixth limitation was that there were also some problems with the components that were used to represent the risk value for the risk objects and the number of steps that were completed in the analysis. The risk value was shown in a NumericUpDown in the text view, but the user should not able to change it directly, as it is set depending on the values for probability and consequence. The reason for using a NumericUpDown was that the risk value could not be separated from other Integers in the data structure. The
value of this Integer was set to the correct value when the risks were saved but could be modified in the GUI when it should not even be seen there. A slightly similar problem was to show the number of steps that had been completed in the analysis. This was shown as 13 checkboxes in the menu in the text view and was checked when all fields in a step had values. If the checkbox should be shown in color it had to be enabled, but then it could also be manipulated by the user. This did not change any value in the object but should not be possible and could be confusing for the user. The Checkbox could be set to disabled, but then it was not shown in color and it was harder to see which steps that were completed as both the checkbox and the background were gray.

Size of components cannot be changed
The seventh limitation was that the size of the components in the GUI could not be changed once they were created. If a lot of information needed to be presented in a single RichTextBox, it might be good to be able to increase the size of it when needed. Nothing in the data structure or save function prevented this from happening but there was no functionality in the GUI that allowed the size to be changed.

Limitation when submitting data
The eighth limitation was related to that when new sub-objects were created, the TextBoxes that were used for submitting the size of them only accepted numeric values. This was done with a regular expression and ensured that no other characters were submitted to this field, but also limited the use of other functional keys, like backspace. It was still possible to remove values by selecting them and press delete, but it would be better to create a more extensive regular expression than the one that exist to support this functionality.

Relations
The ninth limitation was that the relations were saved in two places in each object, both in the List of relations and in the specific sub-object that handled the connection to a list of possible relations. In the sub-object, both the Guid of the related object and the index in the list was saved. The index was needed when the list was recreated when the object was loaded to the GUI. Without the index, the relation would be reset to the first value in the list. The Guid was needed to reset the index if the object that was related to it was removed from the list. If this was not done, the index would point to either the wrong value or cause an exception if the index was higher than the total number of objects in the list.

Automatic generation of source code
The tenth limitation was a bug created by Visual Studio as code was generated automatically when the graphical editor was used and this created conflicts with the code that was added manually. This was similar to the problems with NTE and EASTER but for the new one, there was a method related to an event handler that got duplicated every time something was changed in the graphical editor.

7.3.5 Design problems for a re-design of ASCENSION
Based on the evaluation of ASCENSION, a re-design and some problems that this could focus on was proposed. The goal with this was to identify ways to change the implementation
in order to get around the limitations that exists. This section focus on two main areas, the data structure of the program and how information was presented in the GUI. For each of these, a number of problems that should be solved and ideas to how this can be done are presented. These two areas are also described on a higher level of abstraction than the limitations in the previous section.

**The data structure**

For the data structure there were both simple and more complex problems that needs to be solved. There were two smaller problem and six more complex ones.

- **Simple problems**
  The simple problems were related to the save method and can be solved relatively easy. The first of these was that the object was saved piece by piece. This made it possible to start saving an object and then fail if one of the sub-objects contained an error. There were methods that checked if these errors existed but if these should miss an error, there was no recovery. One way to solve this would be to save the entire object in a temporary variable that is verified before it is saved to the data structure. This ensures that there are two separate ways to check that no errors are inserted into the data structure and introduces a redundancy in the system. The second smaller problem was that objects were updated even if no changes had been made to them. This made the update slower and demanded more resources than needed. This could be solved by checking if the object is different from the existing one. If the object is different, it is updated, otherwise it is not.

- **New steps in the analysis**
  The first of the complex problems was to ensure that new steps could be added to the analysis without changing the source code. This involved to change how the Dictionaries were saved in the program, how the method for switching page was constructed and to find an alternative to the enumeration of the different types of data in the analysis. The first two of these are rather simple as the Dictionaries could be saved in List or similar structure which enables new Dictionaries to be added. The method for switching page could then be connected to each position in the List instead of being hard coded. However, the best way would be to find a more flat data structure that still would be efficient to search and iterate through and be flexible enough to allow more data structures to be added during run-time. When the data structure is re-designed, the real problem is to find a way to separate the different types of data. The strength with enumeration is that all types that are specified are constants and have symbolic named that are easy to work with. The alternative to enumerations should have all benefits they have but be flexible so new types could be added during run-time if needed.

- **Inheritance in the data structure**
  The second of the complex problems was to re-design the data structure to support inheritance. In a previous version, the objects used this and inherited from the class DataStructure, but this was removed as the benefit for this was limited. Using inheritance for the sub-objects might be better as they contain several values that only have use for certain types of sub-objects. They could then be structured after type, for example String, Integer, List, or Boolean. The largest tasks with this is to ensure that there exists weak coupling between the classes, that this only affects the data structure and that the rest of the code can be left as it is. This might not be possible but the solution should aim for this goal.
Separation of objects
The third of the complex problems was to be able to distinguish certain sub-objects in the data structure. Some sub-objects contain information that is more important than others and they should be easier to find in the structure. This includes many different types of information, for example the name of the object or the value of the probability and consequence of a risk. In ASCENSION, these sub-objects are found by iterating through all sub-objects in an object and comparing the label with a predefined String. If this String is different, for example if the developer spells the name wrong, the correct sub-object is never found. The method for distinguishing the sub-objects should be able to give each object a unique identifier and new types of these special objects should be possible to create during run-time by the user.

Centralize casting and converting of objects
The fourth of the complex problems was to centralize the casting and converting of objects between the data structure and the GUI in one place. This was spread out in four places in the code: in a class for class definitions, in the method for saving, in the method for loading and in the method for creating components in the GUI. If one of these were inconsistent with the others, the program could crash. The benefits of having all code related to this in one place, if it is possible, is to simplify the creation of new objects and avoid problems when the source code is updated.

Save the relations in one location
The fifth of the complex problems was to re-design the relations in order to save them in one place. The relation was saved in both objects that are related and in two places in each object. The reason for saving it in two places in each object was to be able to maintain a connection to the GUI and the index of the ComboBox in which the list of possible relations was presented. The new design must be able to solve these issues in order to simplify and make the process more robust.

Simplify the data structure
The sixth of the complex problems was to simplify the structure of the data. There existed several places in the code where data was stored in multiple locations. One of these were the Dictionaries where the GUID was saved both as the key and in some cases, like the structure of the GUI, in each object as well. A new structure should be both more simple to use, only save each object in one place and be possible to use without breaking the structure.

The GUI structure
The problems with the GUI could be broken down in four problems.

Flicker in the GUI
The first problem was that there was some flicker in the GUI when objects were removed. This might be solved by using more of the functionality that exist for the graphical components in C# instead of defining methods in ASCENSION for this. If the objects are saved directly in the GUI instead of in a Dictionary, it might be possible to both simplify the structure of the data and reduce this flicker. Another solution would be to save the position that are if a scrollbar is used, in order to go to the same location again after a panel has been removed.

Size of components
The second problem was that the size of some components in the GUI could not be changed once they were created. The most obvious of these were the textboxes in the text view. One way of doing this would be to allow the textboxes to be resized by
7.3. Evaluation

clicking on them and dragging them to a new size with the mouse pointer. This then needs to be restricted so they cannot be larger than the panel they are contained in.

- **Representation of objects in the GUI**
  The third problem was that both the risk value and the finished steps were represented with components that were unsuited for these objects. The risk value was set when the risks were saved but could be changed in the GUI since it was represented like an Integer. One way of solving this would be to introduce a new type of data that only shows the values but does not allow them to be changed. The checkboxes in the menu that shows the finished steps can also be changed by the user and this could be solved by using pictures instead.

- **Structure of the GUI**
  The fourth problem was to re-design the whole structure in which the components in the GUI is stored. One way to do this would be to create new classes that inherits from the regular components like Panel or TextBox, similar to how the matrix was constructed. In these new classes, methods and variables could be added that contains the functionality that exists in the implementation at present. This should simplify the structure but could introduce new problems with finding the right object when saving and updating or how this structure should be connected to the data structure.

7.3.6 Future potential of ASCENSION

There are several ways in which ASCENSION could be extended or improved which could increase its future potential and use. This section lists some of these.

1. One way to use the program was mentioned by Jonas Hallberg during a presentation of ASCENSION at FOI. He described that it could be used to present an idea to the Swedish Armed Forces and therefore make it easier to find out if this idea is similar to how the risk management process is carried out at present. Even though there exist a model for how it should be done, reality can be a bit different.

2. ASCENSION could be used to simplify the risk management process during the development of IT systems.

3. ASCENSION could be used as the foundation for the development of IT systems. This requires that it is extended to include support for the entire IT lifecycle model.

4. ASCENSION could be used to create the foundation for reports about risk management. This would ensure that all reports have a similar look and contain at least the same number of basic steps. The information that has been submitted into the program could be exported and converted to one of several common document formats, for example for Microsoft Office or OpenOffice.

5. ASCENSION could be used for presentations. This would mean that the same program that is used for the analysis could be used to present it and eliminate the extra work of exporting data and creating presentations.

6. ASCENSION could be used to create a basic structure for an analysis that could be reused for more analyses, in a similar way to how Requirement Collections are used.

7. ASCENSION could be made more dynamic and be useful for other types of risk management. New steps could be added to the analysis and other parts of the program could be more modular and connected if needed.

8. ASCENSION could be used to evaluate existing systems. The graphical representation of the systems could be extended with more functions to model sub-components in each entity and evaluate the system from other criterions than the security demands.
Chapter 8

Conclusion

This report described the work that was done in order to evaluate and combine two models from the Swedish Armed Forces: the risk management model and the IT lifecycle model. An example was then presented of how the risk management model could be extended for IT and the combined and extended model was implemented as a prototype in an existing software. The report started with a literature study that listed and compared different models and methods for threat, risk, and vulnerability analyses. Then different threats for IT, how the two models from the Swedish Armed Forces could be combined and then an example of how one of them could be extended for IT was described. Then a design proposal was developed, of which some parts were implemented as ASCENSION, an extension to the program NTE and the plugin EASTER. This chapter describes the different achievements, the limitations that were made and presents some proposals for future work in this area.

8.1 Achievements

During the work with the master’s thesis that was described in this report, several goals was achieved. This section lists and explains these achievements.

1. The first achievement was to do a literature study. This served as a foundation for the rest of the report, different types of models were identified and then compared with the Swedish Armed Forces risk management model. From this, several ideas was developed that were used during the combination of models and in the design proposal.

2. The second achievement was the identification of several threats for IT. These were divided into three categories and were used to extend the Swedish Armed Forces risk management model for IT.

3. The third achievement was the combination of two models from the Swedish Armed Forces: the risk management model and the IT lifecycle model. The risk management model was combined with step P2 in the IT lifecycle model which changed the order of the risk and vulnerability analyses in this step as well as gave a clear definition to how they should be used.

4. The fourth achievement was the example of how the Swedish Armed Forces risk management model could be extended for IT. This was done by combining the knowledge gained from the literature study about different models, of which some was specialized for IT, and the different threats that was identified in chapter 4.
5. The fifth achievement was the generation of a design proposal for how to implement the combined and extended model from chapter 5. This was done through design guidelines on different levels and requirements from the Swedish Armed Forces. From this design proposal, a proposal for how to implement the combined and extended model in an existing software was developed.

6. The sixth achievement was to evaluate if the program NTE and the plugin EASTER were suitable as the foundation for the implementation of the combined and extended model. The evaluation showed that they were suitable for this and why.

7. The seventh achievement was to break down this design proposal into several clearly defined points. These were then divided into two groups: those that were suitable to implement as a part of a master’s thesis and those that were not suitable to implement as a part of a master’s thesis.

8. The eighth achievement was to implement some of the points that were suitable to implement as a part of a master’s thesis as the plugin ASCENSION to the program NTE. The largest points that were implemented were the three views, the ability to do an analysis, the ability to modify the structure of the analysis, and the ability to view the result.

9. The ninth achievement was to evaluate ASCENSION and come up with points that should be changed and present design problems and ideas to how these could be solved.

10. The tenth achievement was to present how ASCENSION could be extended and show what potential it might have in the future.

8.2 Limitations

There were several limitations to the work that have been done. This section presents those limitations sorted into five categories.

1. The first category of limitations were related to the literature study. The sources for the literature study were mostly government organizations and many were related to the Swedish government. This limited the study and a bigger picture that could have been seen if more private organizations had been included is missing. Many of the models were also similar in layout and structure. Most of them were also found among the semi-quantitative models and there were few models or methods that were clearly qualitative or quantitative. The last limitation with the literature study was that the models were only compared against the Swedish Armed Forces risk management model and not with each other.

2. The second category of limitations were related to the threats against IT. Many of the threats have been identified from a few sources. These sources described many different types of threats, but had this part been based on more sources, even more threats would have been identified. Another limitation was that the threats were described on a rather high level of abstraction and that this section did not describe them in detail.

3. The third category of limitations were related to the combinations of models. The primary limitation of this part was that it was only based the manuals from the Swedish Armed Forces and no detailed information from interviews or contacts with those who had developed the models. There also existed some limitations with the combination of the models as all parts could not be combined. The example for an extension of the risk management model for IT was only an example and not a
complete extension, as this would had taken far to much time and work to be able to complete as a part of a master’s thesis. It is also unknown what the Swedish Armed Forces opinion is about the changes that have been made.

4. The fourth category of limitations were related to the design proposal. The report described many guidelines for the design but all of these were not used in the design proposal. Those that were used were the more specific guidelines and many of the more general were not part of the proposal. This was mainly because this part served more as a checklist that could be used for improving the user interface and including them all in the design proposal would have been difficult. Another limitation with the design proposal was that some of the three views were described in more detail than others. The matrix view was much less detailed and contained less functionality than the other two views. The parts about analysis and visualization of data were also much less detailed than the part about input of data. This was mainly because most of the analysis is done by hand in the original model and than filled into the tool. The visualization was also static compared to the input of data.

5. The fifth category of limitations were related to the implementation. These were described in detail in section 7.3.2 and Section 7.3.4 but can be summarized into four points. The first point is that all features that could be implemented as a part of a master’s thesis have not been implemented. The goal was never to implement all of these but some of them were prioritized before the rest. The second point is that the data structure in ASCENSION and the casting of objects were more complex than than they needed to be. A re-design of this part could make it easier to understand and reduce the chance that errors occur. The third point is that the data structure contains some limitations and that some temporary solutions have been made to get around this. To solve this, the data structure needs to be re-designed. The fourth point is that the user interface is more rigid than it could be and some parts of it is hard coded into the program. By allowing the users more freedom in the program, the analysis could become better and suit the needs of the organization to a higher degree.

8.3 Future work

There is much work that can be done to improve the solution and continue the work on the task that was described in this report. This section lists some possible ways in which work with this can be done in the future.

1. The first way to continue the work is to look into attack graphs. By studying this area further, it could be decided if the implementation should be extended for this type of functionality, if a new program should be developed or if this should be skipped. Two interesting areas of this have been identified in this report: the problems that exists for attack graphs and if they can be combined with intrusion detection systems. The problems limit the use of attack graphs and more work can be done in this area. If attack graphs could be combined with intrusion detection systems, it might be possible to take data from these systems, interpret it and propose what the next action of an attacker might be. This could then be used to adapt the security of a system so the level of security is heightened if an attack is likely to happen and the system administrators could be informed what has happened and what the next step in the likely attack might be.
2. The second way to continue the work is to look into the connection between human-computer interaction and security. By studying this further, the connection between usability and security can be identified in more detail and studies could be made to see how security systems should be constructed so they become more user friendly. If the system is easy to use and does not cause distractions or problems for the users, it is more likely that it will be used like it was supposed to be.

3. The third way to continue the work is to extend the risk management model for IT. The example presented in this report could be used as a way to start as the literature study and the threats for IT presented in this report includes many sources for further information.

4. The fourth way to continue the work is to implement more functionality into the plugin ASCENSION. This could be to implement more of the features identified in this report, for example to implement more functionality in the graph view and make the user interface more dynamic. It can also be to implement some of the features found in the second list in Section 6.6, where some of the features were more complex or more time consuming.

5. The fifth way to continue the work is to re-design the plugin ASCENSION that was described in this report. This can be based around the evaluation found in Section 7.3 and produce a detailed specification for how to improve the current functionality.

6. The sixth way to continue the work is to evaluate and re-design the user interface of NTE and ASCENSION with focus on human-computer interaction. This could start with the guidelines presented in this report, which could be used for a prototype of the improved user interface. This could then be evaluated with the help of surveys, user studies and interviews. This evaluation could then serve as the foundation for the development of a new user interface.
Chapter 9

Thanks

Thanks to Jonas Hallberg & Johan Bengtsson, my supervisors at FOI, Swedish Defence Research Agency and Jerry Eriksson, my supervisor at the Department of Computing Science, Umeå university.
References


[31] Säkerhetspolisen (Swedish Security Service), Rikspolisstyrelsen, Box 8304, Stockholm. En vägledning till säkerhetsanalys, 2005. The guidance is in swedish.


Appendix A

Translations

The translations of names, terminology and abbreviations in the report.

- **Accreditation decision** - Ackrediteringsbeslut
- **Assessment** - Värdering
- **Certification decision** - Auktorisationsbeslut
- **County Administrative Board of Stockholm** - Länsstyrelsen i Stockholms län
- **Demands for approved security functions** - Krav på godkända säkerhetsfunktioner
- **Estimation** - Bedömning
- **Exposing signals** - Röjande strålning
- **Government Offices** - Regeringskansliet
- **IBERO** - Instrument för beredskapsvärdering av områdesansvar
- **Information security class** - Informationssäkerhetsklass
- **Legal, Financial and Administrative Services Agency** - Kammarkollegiet
- **Manual for the Swedish Armed Forces Intelligence Service, information technology** - Handbok för Försvarsmaktens Säkerhetstjänst, informationsteknologi
- **Manual for the Swedish Armed Forces Intelligence Service, threat estimation** - Handbok för Försvarsmaktens Säkerhetstjänst, hotbedömning
- **Swedish Defence Material Administration** - Försvarets Materiellverk
- **Swedish Defence Research Agency** - Totalförsvarets Forskningsinstitut
- **Swedish Emergency Management Agency** - Krisberedskapsmyndigheten
- **Swedish Rescue Services Agency** - Räddningsverket
- **Swedish Road Administration** - Vägverket
- **Swedish Security Service** - Säkerhetspolisen
Appendix B

Figures

These screenshots and diagrams present how the implementation of ASCENSION looks like and other information that was not possible to include in the main part of the report. Figure B.1 shows how the text view looks. Figure B.2 shows how the matrix view looks like. Figure B.3 shows how the presentation looks like. The graph view has not changed and looks exactly like it did before. Figure B.4 shows an expanded class diagram of the classes in ASCENSION. Figure B.5 shows how the two models from the Swedish Armed Forces can be combined, but this version is enlarged so the details are easier to see.

Figure B.1: This is how the text view of the prototype looks like.
Figure B.2: This is how the matrix view of the prototype looks like.
Figure B.3: This is how the presentation of data in the prototype looks like.
Figure B.4: This is a diagram that shows the expanded classes in the class diagram for ASCENSION. In addition to the information presented earlier, this also shows the methods in each class.
Figure B.5: A diagram that shows how the two models can be combined. The structure of the model is adapted from the report by Bengtsson & Hallberg [16]. This version is enlarged so the details are easier to see.