

# Development of Tool Support within the Domain of Risk-Driven Security Testing

Vetle Volden-Freberg



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

Department of Informatics  
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2017



# **Development of Tool Support within the Domain of Risk-Driven Security Testing**

Vetle Volden-Freberg

1st August 2017

© 2017 Vetle Volden-Freberg

Development of Tool Support within the Domain of Risk-Driven Security  
Testing

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

Printed: Reprosentralen, University of Oslo

# Abstract

Today, there exists a wide range of services and applications across several platforms that are prone to attacks. Attackers find new ways to exploit malfunctions and vulnerabilities within these systems every day. There is an increase in cyber security risks and targeted attacks towards the public, industry and governments by the use of the web, social media, mobile devices, cloud services and so on. Therefore, security must be considered thoroughly in the software development life cycle, to minimise the risks represented by either an attacker, the intended user of the system, or other non-human causes that might lead to catastrophic damage to a system.

The security testing community has met these challenges by proposing an approach to security testing that is supported by security risk assessment. This approach is commonly referred to as risk-driven security testing and aims to focus testing on the most severe risk a system is exposed to. The field of risk-driven security testing is relatively new and immature. Thus, lacks formality, preciseness and dedicated tool support. As a response to this, the CORAL approach has been proposed. The CORAL approach is an approach that provides a domain-specific risk analysis language and a method to conduct risk-driven security testing, consequently, providing more formality and preciseness.

However, the approach needs to be supported by dedicated tool support in order to aid security testers further. This thesis investigates how the CORAL approach can be supported by a tool, in order to fulfil the overall aim of introducing proper tool support for the domain of *risk-driven security testing*. We propose a tool developed as a plug-in for the Eclipse Papyrus tool, which supports the CORAL approach. The risk analysis language in our tool adopts a textual notation as opposed to the graphical notation defined for the CORAL risk analysis language. Consequently, as part of the development and evaluation process of the tool, we conducted an empirical study to investigate whether the textual notation adopted by the tool had any impact on comprehensibility in comparison to graphical notation. The results of our empirical study indicate that there is no significant difference with respect to comprehensibility.

Moreover, our results show that the tool is appropriate for security testers in terms of carrying out risk-driven security testing following the CORAL approach, including defining security test cases.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Contribution . . . . .	3
1.2.1	Adaptation of CORAL as a UML Profile . . . . .	3
1.2.2	CORAL Plug-in – Tool Support for Risk-Driven Security Testing . . . . .	4
1.2.3	Empirical Study – Comparison of Textual and Graphical Notation . . . . .	4
1.3	Thesis Overview . . . . .	4
<b>2</b>	<b>Problem Characterisation</b>	<b>7</b>
2.1	Background and Conceptual Clarifications . . . . .	7
2.1.1	Modelling . . . . .	7
2.1.2	Modelling Languages and Tools . . . . .	8
2.1.3	UML Sequence Diagrams . . . . .	8
2.1.4	The UML Testing Profile . . . . .	9
2.1.5	The CORAL Approach . . . . .	9
2.1.6	State of the Art Risk-Driven Security Testing . . . . .	13
2.2	Problem Specification . . . . .	14
2.2.1	Success Criteria . . . . .	15
<b>3</b>	<b>Research Method</b>	<b>17</b>
3.1	Technology Research . . . . .	18
3.2	Evaluation Strategies . . . . .	20
3.3	Selection of Appropriate Evaluation Strategies . . . . .	22
3.4	Prototyping . . . . .	23
3.5	Empirical Study . . . . .	23
<b>4</b>	<b>Research-Based Design</b>	<b>25</b>
4.1	Artefact Design . . . . .	25
4.2	Eclipse, Tools and Frameworks . . . . .	26
4.2.1	Eclipse Modelling Framework . . . . .	28
4.2.2	Graphical Editing Framework . . . . .	28
4.2.3	Graphical Modeling Framework . . . . .	28
4.2.4	Eclipse Papyrus . . . . .	28
4.3	Options for Tool Design . . . . .	29
4.3.1	Plug-in or RCP application? . . . . .	30

4.4	Adaptation of CORAL as a UML Profile . . . . .	30
4.4.1	Data Types . . . . .	31
4.4.2	Lifelines . . . . .	34
4.4.3	Messages . . . . .	35
4.4.4	Risk-Measure Annotations . . . . .	38
4.4.5	CORAL Constraints . . . . .	38
4.4.6	Object Constraint Language . . . . .	39
4.4.7	CORAL Constraints in OCL . . . . .	40
4.5	Customisation . . . . .	41
4.5.1	Graphical Notation . . . . .	41
4.5.2	Palette . . . . .	42
4.6	Deploying the Profile as a Plug-in . . . . .	43
4.7	Researched-Based Design Summary . . . . .	44
<b>5</b>	<b>Evaluation - Empirical Study</b>	<b>45</b>
5.1	Characterisation of the Study . . . . .	46
5.1.1	Current State Analysis . . . . .	46
5.1.2	Topic of our Empirical Study . . . . .	47
5.2	Set Goals . . . . .	48
5.2.1	Formulate the Goal . . . . .	48
5.2.2	Formulate Research Questions . . . . .	49
5.3	Choose Process . . . . .	49
5.3.1	Formulate Hypothesis . . . . .	49
5.3.2	Determine Variables . . . . .	50
5.3.3	Identifying the Subjects of the Study . . . . .	51
5.3.4	Study Design . . . . .	51
5.3.5	Preparation of Experiment Material . . . . .	54
5.4	Execution . . . . .	61
5.4.1	Study Preparation . . . . .	61
5.4.2	Study Execution . . . . .	61
5.4.3	Data Validation . . . . .	63
5.5	Analysis of Results . . . . .	63
5.5.1	Data Visualisation . . . . .	64
5.5.2	Applying Descriptive Statistics . . . . .	67
5.5.3	Hypothesis Testing . . . . .	72
5.5.4	Findings Related to Efficiency . . . . .	73
5.5.5	Findings from the Post-Experiment Questionnaire . . . . .	74
5.5.6	Threats to Validity . . . . .	75
5.5.7	Analysis Summary . . . . .	78
<b>6</b>	<b>Discussion</b>	<b>81</b>
6.1	Success Criterion 1. . . . .	81
6.2	Success Criterion 2. . . . .	83
6.3	Success Criterion 3. . . . .	83
<b>7</b>	<b>Conclusion</b>	<b>85</b>
7.1	Directions for Future Work . . . . .	86



<b>Acronyms</b>	<b>89</b>
<b>Bibliography</b>	<b>91</b>
<b>Appendices</b>	<b>101</b>
<b>A Main Task Questionnaire – Group A</b>	<b>103</b>
<b>B Main Task Questionnaire – Group B</b>	<b>119</b>
<b>C Presentation – Group A</b>	<b>135</b>
<b>D Presentation – Group B</b>	<b>143</b>
<b>E Letter of Consent</b>	<b>151</b>
<b>F Task Scores from the Experiment</b>	<b>155</b>
<b>G SPSS – Statistics Calculations</b>	<b>159</b>
G.1 Total Score . . . . .	159
G.2 Total Score – Without Outlier . . . . .	165
G.3 Part 1 Score . . . . .	171
G.4 Part 1 Score – Without Outlier . . . . .	177
G.5 Part 2 Score . . . . .	183
G.6 Part 2 Score – Without Outlier . . . . .	189
<b>H Example of a CORAL Threat Model Developed using the CORAL Tool</b>	<b>195</b>



# List of Figures

2.1	The relationship between the CORAL risk analysis language and method depicting the language at the core of the approach. Illustration borrowed from Erdogan [33]. . . . .	10
2.2	The icons for the modelling constructs in the CORAL risk analysis language. Illustration borrowed from Erdogan [33].	11
2.3	The seven steps of the CORAL method, illustration borrowed from Erdogan [33]. . . . .	13
3.1	Method for technology research – main steps [109, p. 8] . . .	19
3.2	Evaluation strategies, adapted from McGrath [78, p. 32]. . .	21
4.1	The Eclipse project [121] consisting of five sub-projects: Platform [27], JDT [120], PDE [95], E4 [22] and Orion [26]. . .	27
4.2	CORAL adaptation in a UML profile. . . . .	31
4.3	The frequency data types. . . . .	33
4.4	The conditional ratio data types. . . . .	33
4.5	The enumerations for TimeUnits, Consequences and Likelihoods. . . . .	34
4.6	The CORAL lifelines, extending the UML Lifeline meta class.	35
4.7	The CoralMessages profile. . . . .	37
4.8	The CoralRiskMeasureAnnotations profile. . . . .	38
4.9	An example of a violated constraint with respect to the UnwantedIncident message. . . . .	41
4.10	Examples of violated constraints. . . . .	41
4.11	The lifeline compartment in Papyrus displayed on the left. On the right, examples of sub-compartments that would make CSS customisation for lifelines easier. . . . .	42
4.12	The CORAL palette. . . . .	43
5.1	The six steps in the "A Single Empirical Study" framework [99]. . . . .	45
5.2	The first step in the empirical study framework "A Single Empirical Study" [99]. . . . .	46
5.3	The second step in the empirical study framework "A Single Empirical Study" [99]. . . . .	48
5.4	The third step in the empirical study framework "A Single Empirical Study" [99]. . . . .	49

5.5	Independent variables, confounding factors and dependent variables. Figure adapted from [129, p. 14] . . . . .	50
5.6	The experiment process. Group A is given material with graphical notation, whilst Group B is given material with textual notation . . . . .	53
5.7	The fourth step in the empirical study framework "A Single Empirical Study" [99]. . . . .	61
5.8	The fifth step in the empirical study framework "A Single Empirical Study" [99]. . . . .	64
5.9	Box plot for the total score for Group A and B. . . . .	65
5.10	Box plot for the total score of Part 1 for Group A and B. . . . .	66
5.11	Box plot for the total score of Part 2 for Group A and B. . . . .	67
5.12	Experiment principles adapted from [122, 130]. . . . .	76
H.1	Example of CORAL threat model in the CORAL tool, depicting an XSS attack toward the feedback feature of a web application for shopping. . . . .	195

# List of Tables

4.1	The CORAL plug-in extension points. . . . .	43
5.1	Hypothesis for the empirical study . . . . .	50
5.2	Questions for demographic survey . . . . .	55
5.3	Questions for the main questionnaire . . . . .	58
5.4	Post-experiment questionnaire . . . . .	59
5.5	Table with all the task score categories. . . . .	60
5.6	Knowledge profiles . . . . .	62
5.7	The participants for Group A. WE = years of working experience, D = degree, B = bachelor's degree, M = master's degree. Knowledge in terms of Likert values: UML modelling, SD = sequence diagrams, R = risk assessment or analysis, UI-US = UI Design and usability. . . . .	62
5.8	The participants for Group B. WE = years of working experience, D = degree, B = bachelor's degree, M = master's degree. Knowledge in terms of Likert values: UML modelling, SD = sequence diagrams, R = risk assessment or analysis, UI-US = UI Design and usability. . . . .	63
5.9	Ranges for the accepted values for kurtosis that is approximately normally distributed, table adapted from [75]. . . . .	69
5.10	Descriptive statistics applied to the total score for Group A and Group B. The max score for the whole task set is 27. . . . .	70
5.11	Descriptive statistics applied to the Part 1 score for Group A and Group B. The max score for Part 1 is 12. . . . .	71
5.12	Descriptive statistics applied to the Part 2 score for Group A and Group B. The max score for Part 2 is 15. . . . .	71
5.13	Average time for the task set. $\bar{x}(t)$ is the average time for either Group A or Group B in seconds, $\Delta t = t_B - t_A$ . Furthermore, positive/negative values for $\Delta t$ and % indicate that Group B spent more/less time than Group A. . . . .	74
5.14	Post-experiment questionnaire answers . . . . .	75
F.1	Task scores for Group A. . . . .	156
F.2	Task scores for Group B. . . . .	157



# Acknowledgements

I would like to extend my utmost gratitude to my supervisors Ketil Stølen from the University of Oslo and Sintef, and Gencer Erdogan from Sintef, for giving me the opportunity to undertake this project, providing support, advice, encouragement and guidance to finish my thesis work.

I am indebted to my co-supervisor, Gencer Erdogan, for his guidance and encouragement during the last months of my thesis, who provided me with critical and constructive feedback on my thesis writing which in turn helped shape this master's thesis.

I would also like to thank Mwiza Kumwenda and Øystein Lytskjold Olavsén for reading and commenting on the thesis. Further, I would like to thank Magnus Åsrud and Olav Wegner Eide for companionship when we were all working towards the same goal of finishing our theses.

Thanks are also due to the people who participated in the empirical study that was carried out as part of this thesis. Their participation was crucial, and I am grateful that the process was executed so swiftly, providing me with valuable data and feedback.

I would like to thank my dear family, especially my parents, Lill Kristin Volden and Øivind Freberg, whose love, support, hard work and sacrifices to provide a safe environment for me and my siblings to learn and prosper, have helped me become the person I am today and pushed me toward academic endeavours.

Last but not least, I would like to thank my lovely wife, Rizkika Widya Tarandeli who is always there for me, for bringing encouragement, love and support in my life, in addition to providing advice and comments throughout the thesis process with regard to the writing.





# Chapter 1

## Introduction

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

### 1.1 Motivation

Every year the World Economic Forum releases an annual global risks report. The latest report shows that among the most likely risks to occur within the next 10 years, massive-incident of data fraud/theft, and large-scale cyber attacks come in on fifth and sixth place respectively [39]. At the same time, rising cyber dependency is identified as the fourth top trend that determines global developments over the next 10 years [39]. Meanwhile, there is an increase in cyber security risks and targeted attacks towards the public, industry and governments by the use of the web, social media, mobile devices, cloud services and so on. Furthermore, targeted attacks toward states as exemplified by the recent US elections shows a new trend where adversaries intended to influence public opinion and create an atmosphere of distrust [118]. Not to mention the emerging range of IoT services which further expands the attack surface of our technological infrastructure. With all of this mentioned, we clearly see the need for software security [12, 33, 96, 118]. Software security is the ability of software to resist, tolerate and recover from events that threaten dependability while maintaining confidentiality, accessibility and integrity of information [33, 70]. As the growth of new applications and information systems being developed is increasing, the need for software security grows accordingly. Today, there exists a wide range of services and applications across several platforms that are prone to attacks. Attackers find new ways to exploit malfunctions and vulnerabilities within these systems every day. Therefore, security must be considered thoroughly in the software development life cycle, to minimise the risks represented by either an attacker, the intended user of the system, or other non-human causes that might lead to catastrophic damage to a system.

Software security is achieved by use of a variety of software security practices in the software development life cycle. These are the result

of systematic research for the purpose to create secure software [33, 46]. One of the most important practices in order to ensure software security in the software development cycle is security testing [33]. According to the ISO/IEC/IEEE 29119 software testing standard, security testing is a "type of testing conducted to evaluate the degree to which a test item, and associated data and information, are protected so that unauthorized persons or systems cannot use, read, or modify them, and authorized persons or systems are not denied access to them" [62]. In this context, a test item is the object of testing, e.g. a system or parts/components of a system. Recently, the field of software testing has gone towards a model-based testing approach, and software security testing has followed [33]. Model-based testing focuses on deriving test cases based on explicit behavioural models of a system under test (SUT) and/or its environment [126].

In terms of testing in general, we note the following challenges. First, due to the fact that systems and software tend to be complex and divided into different components that require different inputs, it is impossible to exhaustively test every single aspect of a given system under test [62]. Kaner [65] argues that we cannot test every single aspect due to the fact that we cannot test every single input to a program. Nor can we test all combinations of inputs, we cannot determine every execution path of a given program, and we cannot test for all potential failures that come with the faulty design of a graphical user interface or requirements analysis. Second, when carrying out a test with respect to security- safety- and reliability-critical software, testers face the issue of determining the tests that can reveal faults, errors or failures that cause the most severe risks [33]. Third, the testing phase of development is usually limited by a strict budget and time constraints [40], thus underlining the importance of defining a scope when carrying out tests, and provide "good enough testing" [65]. To address these challenges, approaches that combine security risk assessment to aid security testing have been proposed. These approaches aim to determine which aspects of systems that are most exposed to risk, and use this information to guide the security testing [33]. This approach is often referred to as *risk-based* security testing. However, we will use the term *risk-driven* security testing (RST), as this term better reflects the fact that risks are the main driving factor to guide all phases of the test process [33, p. 4].

Security risk assessment is a process that involves identifying risks, estimating risks and evaluating risks [74]. In this context, a risk is the likelihood for an unwanted incident to occur and its consequence on a specific asset. An asset is something that has value to a party, and which requires protection [74]. Erdogan et al. point out that the field of risk-driven security testing is still immature and points out that the field requires more formality and preciseness, along with dedicated tool support [34]. Further, Erdogan introduces the CORAL approach, which is a stepwise method to combine security risk assessment and security testing. In turn, providing more formality and preciseness for risk-driven security testing. The CORAL approach consists of a risk analysis language and a method to conduct risk-driven security testing. This leads to the problem addressed in this thesis, which involves investigating the possibility of developing

tool support for the CORAL approach, in order to fulfil the overall goal of introducing proper tool support for the domain of *risk-driven security testing*.

The main objective of this thesis is to provide a tool that adopts the CORAL risk analysis language to support the CORAL method. To summarise, there is a need for proper tool support within risk-driven security testing due to the following:

- Security testing is a necessity in a world where software is exposed to new threats and attacks every day, which may impact our technological infrastructure.
- Security testing assisted by security risk assessment aids security testers to carry out risk-driven security testing in selecting and designing security tests that address the most severe risks.
- The testing phase of development is usually limited by strict budget and time constraints. Thus, being able to carry out risk-driven security testing with a tool may reduce costs in terms of money and time.

## 1.2 Contribution

This thesis presents three kinds of contributions. First, it presents an adaptation of the CORAL risk modelling language as a UML profile. The profile introduces a way to include the CORAL constructs that are extensions of UML constructs. In addition, the profile accounts for the constraints derived from the abstract syntax that describes the risk analysis language. Second, it provides a plug-in for the CORAL UML profile for the Eclipse Papyrus modelling tool. With the CORAL constructs being represented by a textual notation. Third, it provides an empirical study in terms of an experiment. In the experiment, we investigate how the difference between textual and graphical notation may affect the comprehensibility and efficiency in the interpretation of threat models represented in CORAL. In the following, we explain each of these contributions in more detail.

### 1.2.1 Adaptation of CORAL as a UML Profile

The CORAL UML profile defines a total of 6 nested profiles which describes specific categories of constructs within CORAL. These are: CoralDataTypes, CoralLifelines, CoralRiskMeasureAnnotations and CoralMessages. CoralMessages has, in turn, two nested profiles, namely CoralIntervalMessages and CoralExactMessages. Whilst the CoralLifelines and CoralMessages profiles define stereotypes for the CORAL constructs that extend UML constructs, CoralDatatypes and CoralRiskMeasureAnnotations define stereotypes for terms that are undefined in the abstract syntax. Among these are frequencies and conditional ratios which can be specified

as either an interval or an exact value, consequence, likelihood and time unit.

In addition to the CORAL constructs, the UML profile defines the constraints according to the CORAL abstract syntax specified in the object constraint language (OCL).

### **1.2.2 CORAL Plug-in – Tool Support for Risk-Driven Security Testing**

The CORAL plug-in is a plug-in of the Eclipse Papyrus modelling tool, which supports the creation of threat scenarios, or what we refer to as threat models. The plug-in adopts a textual notation for the CORAL constructs in terms of UML stereotype annotations. Moreover, by extending Papyrus, we have the advantage of having the UTP already defined, as it is provided by Papyrus. As a result, the plug-in supports the possibility of conducting steps of the CORAL approach which involves the design of test cases based on threat models.

### **1.2.3 Empirical Study – Comparison of Textual and Graphical Notation**

As a result of designing the CORAL plug-in with a textual notation for the CORAL constructs, an empirical study was conducted to compare textual and graphical notation. The overall goal of the study was to investigate whether there is a significant difference with respect to comprehensibility and efficiency when interpreting threat models with either a textual or graphical notation. The study was conducted by the means of an experiment in June 2017.

The findings indicate that there is no significant difference in comprehensibility by using either textual or graphical notation. However, they indicate that there is a difference with regard to efficiency. This is due to the participants subjected to graphical notation spent consistently less time in solving the tasks than those subjected to the textual notation.

## **1.3 Thesis Overview**

This thesis is organised in the seven chapters as follows.

**Chapter 1 – Introduction** is divided into the following sections: Section 1.1 provides the motivation for conducting the thesis. Section 1.2 gives an overview of the main contributions of this thesis. While Section 1.3 gives an overview of all the chapters that constitute the thesis.

**Chapter 2 – Problem Characterisation** is divided into the following sections: Section 2.1 introduces relevant background knowledge regarding this thesis. This includes: modelling in general, modelling languages and tools, UML sequence diagrams, the UML testing profile

(UTP), the CORAL approach and presents state of the art risk-driven security testing. Further, Section 2.2 specifies the problem addressed in this thesis along with success criteria in Section 2.2.1.

**Chapter 3 – Research Method** describes how the research in this thesis was conducted and is divided into the following sections: Section 3.1 presents the technology research aimed at improving or producing new artefacts. Moreover, Section 3.2 presents categories of evaluation strategies, what they are and what they 'measure'. Section 3.3 consists of a discussion of appropriate evaluation strategies for this thesis. Finally, Sections 3.5 and 3.4 describes the evaluation strategies applied in this thesis, empirical study and prototyping respectively.

**Chapter 4 – Research-Based Design** is divided into the following sections: Section 4.1 goes more into detail about the components needed to develop the tool and argues why one should base new applications on the Eclipse rich client platform. Then, in Section 4.2 we present the Eclipse Foundation and relevant Eclipse projects, before discussing our options of tools/frameworks in Eclipse. These being EMF, GEF, GMF and Eclipse Papyrus, presented in Sections 4.2.1, 4.2.2, 4.2.3 and 4.2.4 respectively. In Section 4.3 we discuss our options for tool design with respect to the aforementioned tools/frameworks, further specifying whether we should create an RCP application or a plug-in in Section 4.3.1. Next, Section 4.4 describes the process of adapting the CORAL risk analysis language as a UML profile. This includes a description of all the CORAL constructs along with the CORAL constraints expressed in OCL. Then, Section 4.6 describes how the CORAL profile is deployed as a Papyrus plug-in. Finally, Section 4.7 summarises Chapter 4.

**Chapter 5 – Evaluation - Empirical Study** provides the empirical study conducted by the means of an experiment. This empirical study aims to uncover whether there is a difference with respect to comprehensibility and efficiency between using either textual or graphical notation. In Section 5.1 we characterise our empirical study, this involves a current state analysis and a mapping study of similar studies, along with a description of the topic of our empirical study. Section 5.2 sets the goal for our empirical study along with research questions and what to measure. In Section 5.3 we choose the process for our empirical study, this includes formulating a hypothesis, determining variables, identifying the subjects for our empirical study and empirical study design. In Section 5.4 we describe the empirical study execution, herein preparation, execution and data validation of our empirical study. Finally, in Section 5.5 we analyse our experiment results by visualising the data, applying descriptive statistics and conducting a hypothesis test.

**Chapter 6 – Discussion** provides a discussion of our thesis by discussing our achievements with respect to our success criteria.

**Chapter 7 – Conclusion** concludes the thesis and provides directions for future work.

## Chapter 2

# Problem Characterisation

In this chapter, we first provide some background information relevant to our research topic, before specifying a baseline for our thesis accompanied by success criteria. To start, Section 2.1.1 goes through what the notion of modelling in computer science is, and present some history as well as the categories of modelling paradigms. Then, Section 2.1.2 introduces relevant languages and tools for modelling. Next, Section 2.1.3 presents UML sequence diagrams. Further, Section 2.1.4 presents the UML testing profile. We then go through the CORAL approach in Section 2.1.5 and state of the art risk-driven security testing in Section 2.1.6. Finally, in Section 2.2 we specify the baseline for our research topic with success criteria in Section 2.2.1.

## 2.1 Background and Conceptual Clarifications

### 2.1.1 Modelling

Modelling within computer science is a way of conceptualising and describing a computerised system or parts of a system at a high level of abstraction, either as a textual or graphical representation. One of the first approaches to modelling computer systems with a graphical representation dates back to 1958 when Young and Kent created an abstraction to describe a data processing problem [133]. In 1962 the CODASYL development committee proposed an information algebra as a framework for describing data processing problems [8]. The motivation behind both of these approaches was to achieve a machine-independent way of describing systems. These efforts may have helped pave the way for the development of the relational data model described by Edgar F. Codd in 1969 [13], and further, the entity-relationship model as proposed by Chen in 1976. In an effort to provide a "basis for a unified view of data" [11]. This model combined the advantages of the network model [102], relational model [13] and the entity set model [2]. The aforementioned publications and proposed approaches were just the beginning of a new way of thinking about computerised systems and the challenge of creating good ways of describing and representing

them. Today there exists many modelling paradigms. As cited by Erdogan et al. [33, p. 38], according to Lamsweerde [68] and Utting et al. [127], we can group modelling notations into seven modelling paradigms: state-based notations, transition-based notations, history-based notations, functional notations, operational notations, stochastic notations, and data-flow notations.

## 2.1.2 Modelling Languages and Tools

Today, there exists a number of standardised modelling languages. The most accepted standard however, is the unified modelling language (UML) standard developed by the object management group (OMG) in the early 1990s. The motivation behind the creation of UML was to create a more complete modelling language. By combining the advantages of the Booch method, OOSE and OMT, the first version of UML was published in 1997 [7], and has been an ongoing project since then. The latest UML version as of May 2016 is UML 2.5 [88]. UML has encouraged the creation of a large variety of specifications and extensions, suited for specific purposes. Examples of extensions are [89]: SysML used for modelling a wide range of systems engineering problems [87], SoAML used for modelling service oriented architectures [103], and IFML used to model interaction flow models to describe the principal dimensions of an application front-end [60]. For a full list of formally published UML specifications and extensions, refer to [89]. In this thesis, we will benefit from UML sequence diagrams (see Section 2.1.3) and the UML testing profile (see Section 2.1.4). There exists a wide range of tools that adopt UML or UML specifications/extensions. Examples of open-source tools are Eclipse Papyrus [91] and ArgoUML [1] that are published under the Eclipse public licence (EPL) [32], and UMLet [125] which is published under the GNU general public licence (GPL) [47] to name a few. Examples of proprietary UML modelling tools are IBM Rational [57] and MagicDraw [76].

## 2.1.3 UML Sequence Diagrams

In UML, a sequence diagram is a kind of interaction diagram, which focuses on interchanging messages between lifelines (objects/components) [88, p. 593]. Sequence diagrams can be used as a systematic way of representing interactions/events between components within a system. This allows a software developer to specify a computer system as a set of sequence diagrams. Since sequence diagrams are part of the UML 2.5 specification, it supports several operators such as: alt, opt, par, loop, neg, assert, strict, ignore, consider and critical [4]. An interaction operator specifies the semantics of a combined fragment and determines the usage of the interaction operands in the combined fragment [58]. A combined fragment is a logical grouping, which contains conditional structures (operands) that affect the flow of messages [14]. Sequence diagrams are



most widely used in the model-based testing (MBT) community [18, 35], for this reason, the CORAL method benefits from using sequence diagrams.

For a more elaborate explanation of the usage of UML 2.5 sequence diagrams and the semantics, refer to the UML 2.5 manual [88, p. 564-596].

#### 2.1.4 The UML Testing Profile

The UML testing profile (UTP) is a profile that extends and restricts the original (UML) language [4, p. 29] for testing. As the UTP 1.2 manual states, UTP is used for: "Designing, visualising, specifying, analysing, constructing, and documenting the artefacts commonly used in and required for various testing approaches, in particular, MBT approaches. Model-based test specifications expressed with the UML Testing Profile are independent to any methodology, domain, or type of system" [123]. UTP gives the tester the ability to specify abstract test models with respect to the system under test (SUT), that simplifies validation and the readability of test models [4]. UTP also provides the possibility to define default systems behaviour, making it easier to catch unwanted exceptions during test execution [4, 123]. As summarised by Baker et al. [4, p. 32], the UTP provides the following concepts to describe test behaviour:

- **Test objective** allowing the designer to express the intention of the test.
- **Test case** is an operation of a test context specifying how a set of cooperating components interact with the SUT to realise a test objective.
- **Default** is a concept for making the behaviour description more complete by specifying situations where the described sequence does not happen.
- **Verdict** is a predefined enumeration specifying possible test results, for example, pass, inconclusive, fail and error.
- **Validation action** is performed by the test component to indicate that the arbiter is informed of the test component's test result.
- **Timers** are used to manipulate and control test behaviour as well as to ensure the termination of test cases.
- **Time zones** are used to group components within a distributed system, thereby allowing the comparison of time events within the same time zone [4, p. 32].

UTP defines several stereotypes used to specify certain model elements, for a tabular summary see [123, p. 116].

#### 2.1.5 The CORAL Approach

The CORAL approach is an approach that combines a risk analysis language with a method for risk-driven security testing and was proposed

by Erdogan [33]. The description of the risk analysis language and method, i.e. the approach is documented by Erdogan, and this section introduces the approach based on the documentation.

The CORAL approach aims specifically at helping security testers, by providing an approach to systematically conduct risk-driven security testing. The risk analysis language resides in the core of the approach. This is illustrated in Figure 2.1, which shows the relationship between the language and method. We first present the CORAL risk analysis language, before describing the steps of the method.

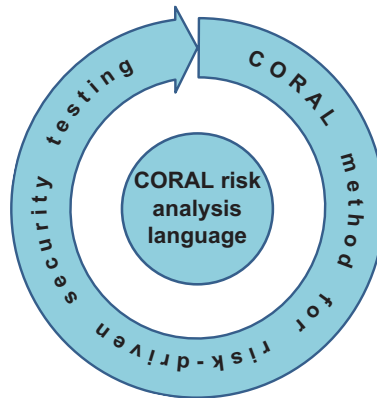


Figure 2.1: The relationship between the CORAL risk analysis language and method depicting the language at the core of the approach. Illustration borrowed from Erdogan [33].

### The CORAL Risk Analysis Language

The risk analysis language is based on UML interactions and consists of constructs that extend common UML constructs, such as the UML (asynchronous) message and lifeline. The interactions are expressed in UML sequence diagrams. The language provides a graphical notation to represent risk-related information directly in the diagram. This way, the security tester can apply risk analysis directly in the diagram, and based on that design security tests. Consequently, the security tester does not have to conduct risk analysis separately using another language. In summary, the CORAL risk analysis language consists of:

**A Graphical notation** that provides the necessary constructs for identifying, estimating and evaluating security risks [33]. The icons used to represent risk information is based on the corresponding graphical icons from the CORAS risk analysis language [74]. CORAS is an approach to risk analysis, supported by a language, method and a tool [74]. The use of CORAS icons is due to empirical studies have proven to be cognitively effective [108]. The graphical icons are grouped into five categories: diagram frame, lifelines, messages, risk-measure annotations and interaction operators [33]. Figure 2.2 shows all the different icons for the language.





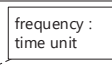
Diagram frame		Messages		Interaction operators	
Notation		Message type	Notation	Operator	Notation
Frame	sd Name	General message	message name →	Potential alternatives	alt
		New message	▶ message name →	Referred interaction	ref Name
		Altered message	◀ message name →	Parallel execution	par
		Deleted message	◀ message name →	Loop	loop
		Unwanted incident message	☀ message name →		
Lifelines					
Lifeline type	General lifeline	Deliberate threat lifeline	Accidental threat lifeline	Non-human threat lifeline	Asset lifeline
Notation	Name	 Name	 Name	 Name	 Name
Risk-measure annotations					
Annotation type	Notation				
Frequency					
Conditional ratio	conditional ratio				
Consequence	consequence				

Figure 2.2: The icons for the modelling constructs in the CORAL risk analysis language. Illustration borrowed from Erdogan [33].

As can be seen from the figure, there are five lifeline types: general lifeline, deliberate threat lifeline, accidental threat lifeline, non-human threat lifeline and asset lifeline. Moreover, there are five types of messages: general message, new message, altered message, deleted message and unwanted incident message. Finally, CORAL introduces three risk-measure annotations: frequency, conditional ratio and consequence.

**An abstract syntax** defined in extended Backus–Naur form [61]. The rules defined by the syntax specify what combinations of constructs that can be used to model syntactically correct interactions. Further, the grammar makes use of eight undefined terms: *identifier*, *asset lifeline*, *int*, *minint*, *maxint*, *exact*, *interval* and *time unit*. Refer to Erdogan [33, p. 63-65] for further reading regarding the abstract syntax.

**A Natural-language semantics** that provides security testers with a structured approach to generate the semantics of interactions produced by the CORAL language, in terms of English prose. This is to help security testers "clearly and consistently document, communicate and analyse risks" [33]. Refer to Erdogan [33, p. 65-68] for further reading.

## The CORAL Method

The CORAL method is a stepwise method with a total of seven steps as seen from Figure 2.3 to conduct risk-driven security testing [33, p. 68]. The method expects as input for Step 1, a description of the system under test. This may be in the form of "system diagrams, use case documentation, system manuals, source code, executable versions of the system and so on"

[33]. As indicated by the arrows, each step takes as input the output from the preceding step. The steps are conducted as follows [33]:

**Step 1:** Based on the description of the system under test, planning for the risk-driven security testing process can begin. This includes: preparation of a model from the system under test as a set of sequence diagrams, identification of security assets, and definitions of the frequency and consequence scales. Finally, a risk evaluation matrix is constructed from the frequency and consequence scales.

**Step 2:** From the models of the system under test and the identified security assets, security risks are identified, represented as unwanted incident messages. Then, for each risk, threat scenarios that may cause these are identified.

**Step 3:** For the messages in the threat scenarios identified in Step 2 that cause risk, frequencies and conditional ratios are estimated. From this, the frequencies for each unwanted incident (risk) is calculated, and consequence in terms of impact on assets defined. The frequencies, conditional ratios and consequences are modelled by using the CORAL risk-measure annotations.

**Step 4:** In this step, based on the annotated threat scenarios, we evaluate the risks according to the risk evaluation matrix. In addition, we specify a suspension criteria e.g. a threshold for risk values. Next, the risks that are of a similar nature are aggregated to figure out if their risk values should be increased. If they have, we assess whether they should be included in the testing. Finally, we select which risks to test based on the suspension criteria. The risks that are not covered by the suspension criteria are excluded from the testing.

**Step 5:** We now proceed to specify test cases for each risk selected for testing. First, a reference is made to the threat scenario for which the risk occurs. Second, a test objective is specified for each threat scenario. Third, we annotate the threat scenarios with stereotypes from the UML testing profile [123], thus, selecting the interactions that fulfil the test objective.

**Step 6:** We now carry out security testing with respect to the security tests designed in Step 5. The test cases may be "executed manually, semi automatically, or automatically" depending on whether the test has to be carried out manually, or is implementable in a tool e.g. as an executable model.

**Step 7:** A test incident report is written, based on the test results, we document each test incident. This results in a test case incident report.

For simplicity, we will refer to the CORAL risk analysis language as CORAL or the CORAL modelling language. The method will be referred to as the CORAL method, and the combined use of the language and method will be referred to as the CORAL approach, for the duration of

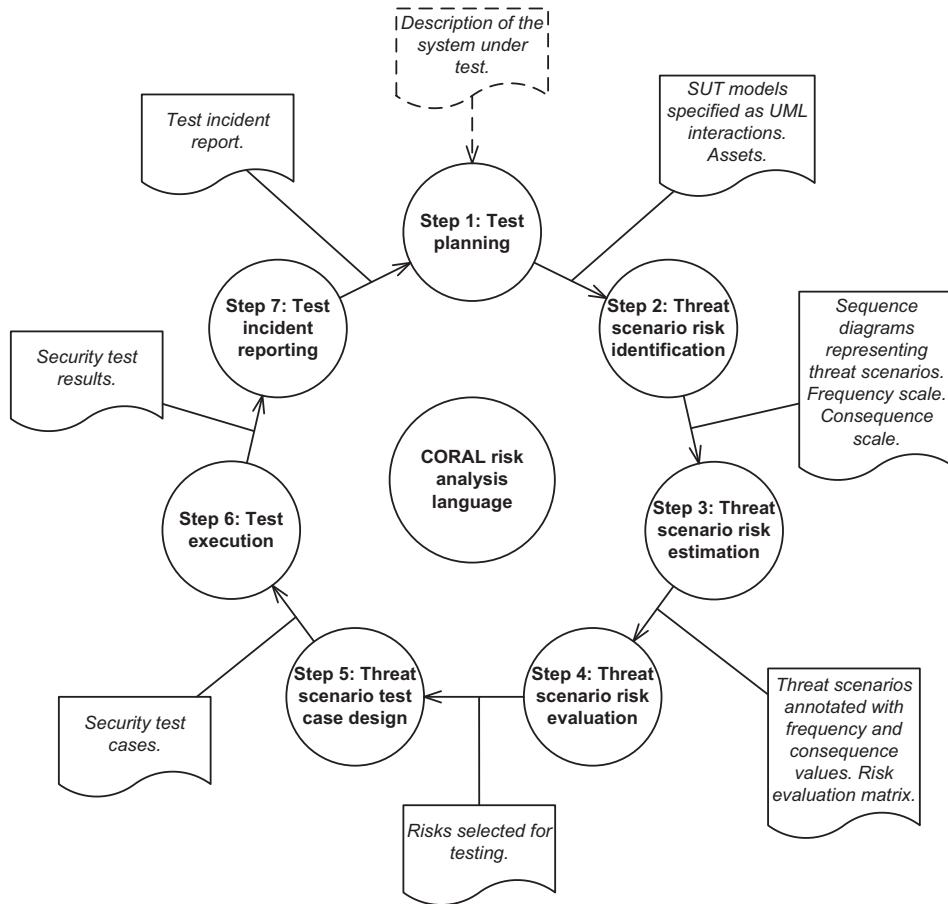


Figure 2.3: The seven steps of the CORAL method, illustration borrowed from Erdogan [33].

this thesis. Also, the diagrams that are developed using the CORAL tool will be referred to as threat models.

### 2.1.6 State of the Art Risk-Driven Security Testing

Security testing approaches that are supported by security risk assessment is commonly referred to as *risk-based* security testing. However, we will use the term *risk-driven* security testing, as this term better reflects the fact that risks are the main driving factor to guide all phases of the test process [33, p. 4]. This survey of existing approaches on the topic is heavily based on the work carried out in the systematic literature review by Erdogan et al. [34]. The literature review was systematically carried out in an effort to find relevant sources with regard to the topics *test-based risk analysis* (TR) and *risk-based testing* (RT) [34]. The search was conducted twice, and after sorting the findings based on author, they identified a total of 28 approaches. Three approaches were concerned with TR, whilst 25 were concerned with RT. However, the 25 approaches that are concerned with RT can be further divided into nine different categories [33, p. 47]. The

category that is most relevant to our proposed thesis is "Approaches with main focus on security; that is risk-driven security testing" [33, p. 47]. From this category, six approaches were found. Risk-driven security testing is a fairly new concept, as all the identified approaches were published within the last seven years, and more frequently as of late. Xu et al. proposed an approach that generates security tests based on formal threat models in the form of predicate/transition nets [132]. The approach has been applied in two realistic case studies and proved to be able to kill security mutants that were deliberately injected into the system. It is not supported by a tool. Murthy et al. proposed an approach that combines the advantages of NIST and OWASP to model threat scenarios and test cases [81]. The approach was applied to a gaming application and proved to save time, cost, and resource usage. Zech et al. use a model-based approach to risk-driven testing targeting cloud computing environments [134, 135]. The approach is mainly focused on finding systems deficiency, rather than the traditional focus on systems validity. The work shows no reference to any empirical evaluation. The approach has, however, dedicated tool support in the form of an Eclipse plug-in. The approaches suggested by Botella et al. [9], Großman et al. [48, 49] and Seehusen [101] makes use of the CORAS risk analysis language [74]. By the use of CORAS they identify security risks and create risk models. These models contain threat scenarios that are used to determine test procedures, which in turn are used to specify test cases. The guidelines of how to accomplish this was proposed by Seehusen [101]. Similar procedures have been applied by Botella et al. and Großman et al. Botella make use of UML class diagrams, object diagrams and state machines to instantiate the test pattern [9]. Großman uses a test design strategy [48, 49]. Botella makes use of Seehusen's CORAS tool for risk modelling and CertifyIt for test case design and execution. Großman et al. have incorporated CORAS in their own tool for risk modelling as well as test case models for test execution.

## 2.2 Problem Specification

The problem addressed in this thesis is concerned with creating dedicated tool support for the CORAL risk analysis language, which is the core of the CORAL approach. Consequently, providing tool support within the field of risk-driven security testing in general. The purpose is to provide sufficient tool support within the domain of testing, risk-driven testing, and security testing [33, p. 33]. As Erdogan states in Section 8.4 Directions for future work [33, p. 104-105]:

One obvious direction of future research is to develop a modelling tool for the CORAL approach. As pointed out in our literature review in Paper 1, the field of risk-driven testing needs more formality and proper tool support. The CORAL language is already formalised, and this opens for appropriate tool support for the CORAL approach. A supporting tool

would obviously increase the efficiency of risk modelling. Moreover, it could also support automatic test execution since the CORAL approach makes direct use of the risk models for test identification and test execution purposes [33, p. 104-105].

We gather that the tool has to support the CORAL risk analysis language. This includes the graphical notation, syntax and semantics (constraints). Furthermore, it should extend existing UML constructs for the CORAL constructs that are UML-extensible. In addition to this, we need to figure out how to include the constructs that are not supported by UML, e.g. risk-measure annotations. To cover the steps in the CORAL approach that is involved with testing, the tool should have support for UTP. This way, security testers can make use of UTP to specify test cases directly in the threat models.

To guide the development of the CORAL tool, we generalise two high-level components that will make up a baseline for the tool. These components are a *diagram editor* and a *test case design suite*:

- **The diagram editor** will provide the security tester with utilities to create an explicit behavioural description of a system under test. In order to sufficiently capture the system behaviour, CORAL makes use of UML sequence diagrams, as mentioned in Section 2.1.5. Consequently, the diagram editor is an editor for sequence diagrams. Moreover, the diagram editor should provide CORAL constructs to apply risk information directly in the model representing the system under test. For this purpose, the editor must implement common features found in diagram editors. Examples of common features are: a canvas, palette of modelling elements, a draw engine to draw model elements on the canvas, state information to provide undo/redo, key-bindings, and so on. Additionally, the diagram editor should support the CORAL graphical notation in order to represent risk information relevant to the system under test. The CORAL notation will be applied to stereotypes within the UML notation. Frequencies, conditional probabilities and consequences do not have corresponding UML constructs for sequence diagrams [33], so this will have to be implemented.
- **The test case design suite** will provide security testers with the possibility of using the UTP to specify test cases directly in the threat models. In this suite, the tester will be able to create and modify test cases by making direct use of threat models.

### 2.2.1 Success Criteria

To be able to fulfil the requirements as mentioned in the previous section, and reach the overall aim of the thesis, we need to identify and define a set of success criteria. We define the following success criteria:

**Success Criterion 1.** *The tool should support the creation of security tests based on the available risk picture*

It should be possible to accurately model the aspects of the risk assessment with the tool. Moreover, it should be possible to specify test cases with respect to the available risk picture.

**Success Criterion 2.** *The tool must sufficiently aid security testers in selecting and designing security tests with the help of security risk assessment.*

By this, we mean that the tool must provide the security testers with useful information based on the risk assessment. Such that this information will ensure that the selected and designed test cases are the ones that cause the most risk with respect to the suspension criteria. The suspension criteria serve as a threshold for the risk values we want to include in the security testing. Consequently, used in the method used to reflect the investable testing effort [35].

**Success Criterion 3.** *The tool must be appropriate and comprehensible for security testers.*

The main stakeholders that will benefit from the creation of this tool are security testers. Therefore, the tool must be appropriate and comprehensible to them. Hence, it is important that the features are properly expressed.



## Chapter 3

# Research Method

In Section 2.2, we described the main topic of this thesis and the challenges security testers face within the domain of risk-driven security testing. Based on this, we defined a set of success criteria that this thesis intends to fulfil in order to accomplish our objective. In this section, we discuss the steps required in order to conduct our research, namely the research method to be applied in the proposed thesis.

The word research comes from the middle French word "*reserche*" which translated to English means "*to go about seeking*" [20]. In the context of classical research, the term research has been defined in several ways [109, p. 3]. Borrowing the definition from Merriam-Webster, research is:

Investigation or experimentation aimed at the discovery and interpretation of facts, revision of accepted theories or laws in the light of new facts, or practical application of such new or revised theories or laws [20].

In short, what we seek is data that will either add knowledge or modify existing knowledge. When conducting research, a researcher must first formulate a question as a basis for the research and form a tentative explanation to answer this question, a hypothesis [109, p. 5]. Through observations and investigations, the researcher has to check whether the hypothesis is true in reality, in a process known as hypothesis testing. Hypothesis testing is also referred to as evaluation [109, p. 3]. It is common to make predictions regarding the outcome of the observations and investigations. Predictions are statements that are only proven true if the hypothesis is true [109, p. 5]. Thus, if an evaluation of a hypothesis confirms the predictions, the hypothesis is strengthened. However, if the predictions are proven false, it can cause a rejection of the hypothesis [109, p. 5]. This approach is commonly referred to as the "*scientific method*", or as Solheim and Stølen define it, classical research [109, p. 3]. A hypothesis can be strengthened through evaluation, although it can never be ultimately proven. However, there might arise new questions worth examining, thus the classical research is an iterative process [109, p. 6]. The essence of classical research as pointed out by Solheim and Stølen:

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

This research method is heavily rooted within what Solheim and Stølen refer to as basic research defined as follows "*Research for the purpose of obtaining new knowledge*", with the main steps defined as *problem analysis, innovation and evaluation* [109, p. 4, 6]. We will refer to the innovation step as *research-based design*, as this better reflects the meaning of the step nowadays. This thesis, however, is more concerned about asking questions regarding technology. It is about finding better ways of solving practical problems, and specifically how to aid the domain of risk-driven security testing. To this end, we will benefit from a research method called technology research [109, p. 3].

Technology research adopts the method established within the classical sciences with a more practical approach in order to create new and better artefacts. Technology research is similar to design science. An artefact in design science is defined as something created by people for some practical purpose, examples are algorithms, methods, notations, techniques, and even conceptual frameworks [128, p. 29]. This is similar to the definition of an artefact in technology research. According to Solheim and Stølen, an artefact is referred to as an object manufactured by a human being [109, p. 3], an object intended to be useful for human beings. In design science, the objective is to design artefacts to interact with a problem context in order to improve something in that context [128, p. 3]. Unlike classical research, which aims to understand reality, design science aims to develop artefacts that serve human purposes. The technology research method is an iterative method that consists of three main steps, *problem analysis, research-based design and evaluation*. These steps correspond to the three steps of the design cycle, *problem investigation, treatment design and treatment validation* [128, p. 27]. The design cycle, however, is part of a broader cycle, known as the engineering cycle, in which a designed and validated treatment is implemented in the problem context, and the implementation is evaluated [128, p. 33]. Technology research falls mainly within the category of applied research, which is "*Research seeking solutions to practical problems*" [109, p. 4]. Hence, the main difference between basic research and applied research is the former being aimed at discovery of new, general information about the real world that might not be directly applicable. Whilst the latter is specifically aimed at being directly applicable and solving practical issues. In the following sections, we will further explain the technology research method (Section 3.1), give an overview of evaluation strategies (Section 3.2) and an overview of the selected evaluation strategies for this thesis (Section 3.3).

### 3.1 Technology Research

The technology research method is a research method that is aimed at improving or creating new artefacts, e.g materials, automates, medicines,

oil production methods and computer programs [109]. In our case, we will create a computer program, that will assist security testers in testing software with a focus on security. The first thing a researcher has to do when conducting technology research is to collect requirements concerning the artefact (problem analysis). As opposed to the classical research method, instead of asking the question about *What is the real world like?*, we ask the question *How can we produce better or new artefacts that can benefit humanity in solving practical issues we face in the real world?* After the researcher has established the context of the task at hand, the second step is the research-based design step. The research-based design step is the process of making the new or better artefact under the assumption that it is feasible. When the artefact or a prototype of the artefact is ready, the researcher has to figure out whether the artefact satisfies the requirements established during the problem analysis. This process is referred to as the evaluation, see Section 3.2 for an overview of evaluation strategies. There are many evaluation strategies one can use to obtain the information needed to determine whether the artefact satisfies the requirements or not, and decide whether it needs further development. The technology research method follows the same basic steps as the classical research method; *problem analysis, research-based design and evaluation*. Also, the technology research method is an iterative method as illustrated in Figure 3.1. After one iteration of the method, it is likely that the researcher has to adjust the requirements and go through the research-based design step again to produce a new artefact that reflects the modified requirements. With regards to our thesis, this would involve modifying the thesis success criteria. This is very natural since there are usually elements one tend not to consider when carrying out the problem analysis. These elements can later be revealed once you start creating the artefact in the research-based design step, or during the evaluation step.

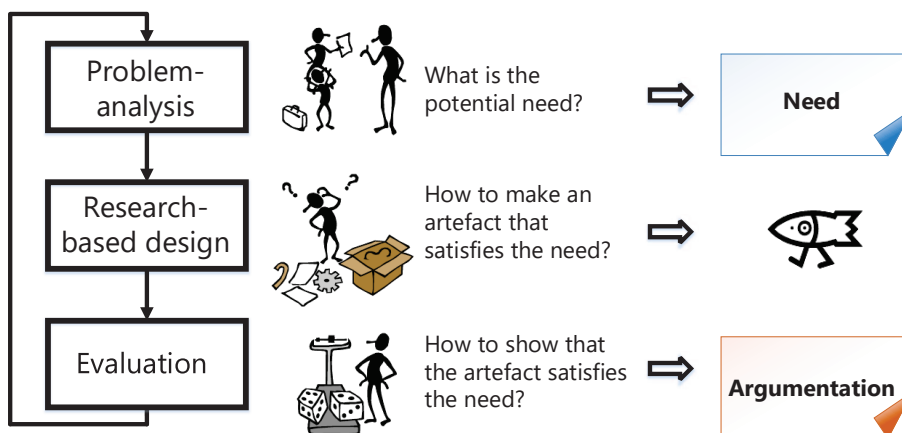


Figure 3.1: Method for technology research – main steps [109, p. 8]

## 3.2 Evaluation Strategies

As mentioned in Section 3.1, an evaluation strategy is a process aimed at providing information that can contribute to a decision regarding the artefact's fulfilment of requirements. There is a wide range of evaluation strategies available. The strategy one is to choose depends on several factors. With resource restrictions in mind there are, according to McGrath primarily three factors one would want to maximise [78, p. 31]:

(A) **Generality**

A measure of the validity of results across populations.

(B) **Precision**

The precision of measurement of the acquired results, and control of external variables that are not part of the study.

(C) **Realism**

To what degree the evaluation reflects realism (if it was performed in a realistic context).

Although one would want to maximise all these factors to achieve the best result possible, McGrath argues that this is not possible and that every research strategy is flawed – although different strategies have different flaws [78, p. 32]. It is therefore important that one chooses evaluation strategies that complement each other to attain acceptable values for each factor. Figure 3.2 illustrates that the spatial relationship between the common evaluation strategies emphasise the dilemma: "The very things that help increase one of the desired features –A, B and C –also reduce the other two" [78, p. 32].

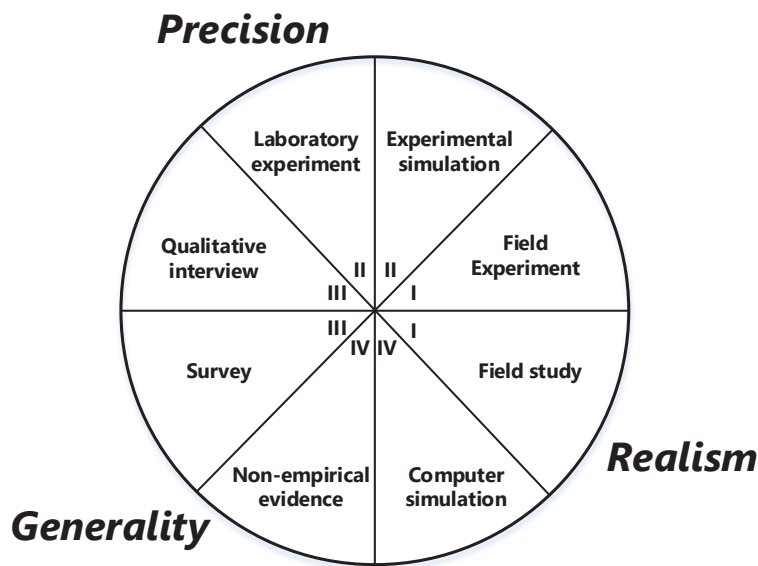


Figure 3.2: Evaluation strategies, adapted from McGrath [78, p. 32].

To cover all the common evaluation strategies in depth is beyond the scope of this problem analysis, however, we will give a brief description of each of the strategies depicted in Figure 3.2.

- **Field studies** are direct observations of "natural" systems with little or no interference by the researcher. Field studies are high on realism, but lack precision and generality, as they are hard to replicate.
- **Field experiments** like field studies are observations carried out in a natural environment, however with certain factors being deliberately manipulated for study.
- **Laboratory experiments** are attempts to recreate systems with a large degree of control and the possibility to isolate variables to be examined, Gains high precision at the cost of low generality and realism.
- An **experimental simulation** is a laboratory study in which we try to simulate relevant processes that occur in the real world.
- A **survey** is concerned with gathering information from a broad group of carefully selected informants. The information is usually gathered through a set of questions, either as questionnaires or interviews. Surveys gain generality at the loss of precision and realism.
- A **qualitative interview** is a collection of information from a few selected individuals. The answers are more precise than those of a survey, but cannot be generalised to the same degree.

- A **computer simulation** is operating on a model of a given system. Scores higher on realism than generality due to the fact that it is system specific.
- **Non-empirical evidence** is a theoretical approach based on argumentation with logical reasoning. Scores high on generality as this is the overall aim, but low on precision and realism as a result of not being empirical.

The eight strategies are further divided into four groups [109, p. 17]:

- I The evaluation is performed in a natural environment.
- II The evaluation is performed in an artificial environment.
- III The evaluation is independent of environment.
- IV The evaluation is independent of empirical measurements.

### 3.3 Selection of Appropriate Evaluation Strategies

To get a starting point from which we can select appropriate evaluation strategies, we re-examine the requirements or success criteria we established in Section 2.2.1:

1. *The tool should support the creation of security tests based on the available risk picture.*
2. *The tool must sufficiently aid security testers in selecting and designing security tests with the help of security risk assessment.*
3. *The tool must be appropriate and comprehensible for security testers.*

With respect to success criterion one, we need to assess whether the tool fully supports the creation of security tests by the use of the CORAL approach. This involves verifying that the tool fully supports the CORAL risk analysis language and UTP for designing test cases. To this end, we can benefit from an evaluation strategy called prototyping to gain a better understanding of the requirements of the artefact. Prototyping falls somewhere in between experimental simulation and field experiment, as we (the developer) would try to simulate the security testers' activity while controlling certain factors for study. See Section 3.4 for further explanation of prototyping.

Success criterion two and three are mostly concerned with the security testers' perception of the tool. To what extent the tool can sufficiently help the security tester in determining which tests address the most severe risks, is largely based on the quality of the risk assessment carried out by the security tester beforehand and his/her ability to produce inputs that reflect reality as precisely as possible. This is obviously the security testers' responsibility. Furthermore, appropriateness is also measured to what extent the CORAL tool benefits from using well-known paradigms

within software and security testing. To evaluate success criterion two and three one could conduct an empirical study to gain precision and realism. An empirical study is an evaluation strategy that is based on direct and indirect observation or experience. This approach, however, requires the availability of the individuals to participate in the empirical study. Finding appropriate participants can often times be difficult. Traditionally, students have been selected as participants for empirical studies. However, having industry professionals are often required to gain accurate insight and responses to research questions [105]. Empirical studies are described further in Section 3.5.

### 3.4 Prototyping

A prototype is an initial version of a system [111, p. 45], that represents the artefact created from requirements established initially in the problem analysis. The process of prototyping is concerned with writing programs for the purpose of learning about their optimal design and construction [5]. The method can help us figure out what are the strengths and weaknesses of our tool early in development, and discover new requirements or success criteria. As stated by Balzer et al. "*Given a proposed solution to a problem, prototyping is used to answer three types of question: Is this a method for achieving the solution; does the proposed implementation have acceptable performance, production cost, and reliability; and is it a good solution?*" [5]. Prototyping is an iterative approach and one can end up producing several prototypes to achieve a satisfactory understanding of the requirements.

### 3.5 Empirical Study

An empirical study is usually carried out by one of the three major strategies: *survey*, *case study* or *experiment* [130]. While surveys and case studies are both qualitative and quantitative, an experiment is a quantitative evaluation strategy [129]. The survey evaluation strategy was described in Section 3.2.

A case study often referred to as 'research-in-the-typical', is a study concerned with studying a real project, activities or assignments. Throughout the study, data is collected for statistical analysis. The case study aims to track specific attributes or relationships between attributes [130]. In our case, a case study may, for example, be aimed at conducting a security risk assessment to support security testing, i.e. performing the CORAL approach with the tool in a real project. Due to this, the control of a case study is lower than for an experiment. Since a case study is an observational study, while an experiment is a *controlled* experiment [130].

An experiment often referred to as 'research-in-the-small', is often conducted with a limited scope and in a laboratory setting [129]. Experiments are *controlled*, since one controls certain variables, and apply treatment to them for the experiment's control groups to observe an effect. The effect

is measured and data is gathered, which forms the basis of the statistical analysis. There are two types of experiments: randomised and quasi-experiments. The former being an experiment where treatment is assigned to participants at random, while in a quasi-experiment treatment is not randomly assigned [129].

The strategy used to carry out the empirical study for this thesis is further described in Chapter 5.



## Chapter 4

# Research-Based Design

In this chapter we go through the second step in the technology research method explained in Section 3.1, namely the research-based design step. In Section 4.1 we present a more in detail explanation of the components required to develop a modelling tool, and argue why it is beneficial to build the tool from existing non-proprietary software. Section 4.2 introduces the Eclipse environment and relevant tools and frameworks with respect to creating the CORAL tool. Section 4.3 introduces the options with regard to tool design, and argues why we should benefit from using the UML profile in Papyrus. Further, the section discusses whether to create an RCP application or a Papyrus plug-in. In Section 4.4 we present the process of adapting the CORAL abstract syntax as a UML profile. This includes the specification of the UML profile, customisation and deployment of the plug-in. Finally, we summarise the research-based design step and argue why our evaluation will focus on comparing textual and graphical notation.

### 4.1 Artefact Design

In Section 2.2 we discussed the three main components the CORAL tool should consist of. These were: diagram editor and test case design suite. These are high-level descriptions and do not cover all the required functionality by far. There are several components needed in order to have a fully functional modelling tool. First, we would have to develop a documentation scheme for how models would be represented in a file format. Second, we would need to create a parser to accurately parse the information from our file format. Third, you would need to build objects from the model elements stored in the file format, and represent these to the user. Furthermore, a user interface would be required to manage files, projects, and models. The UI should also have a palette with the modelling elements. The diagram editor would require a draw engine to draw the model elements. Moreover, a debugger would be required or some kind of model validity check to ensure that the models comply with the CORAL constraints.

With respect to maintainability, we would have to implement a

mechanism suitable to manually or automatically update the application and its components. In addition, the tool should implement the UML meta model. This is to facilitate the integration of other models with CORAL models, and since sequence diagrams are part of UML. Along with this, UTP would be required to include the stereotypes used for the last stage of the CORAL approach described in Section 2.1.5. It would also be beneficial to include a matrix editor for the risk evaluation matrix, likelihood scale and consequence scale.

Finally, the tool has to be optimised to the extent that it is scalable and able to handle complex models. With this mentioned, is it feasible to design all of these features from scratch? It is feasible to the extent that it is possible, although unlikely given the time constraints of this thesis.

However, often times in software development one has to rely on existing functionality. This can be in the form of an application programming interface (API), library or framework. By building on existing functionality one can significantly reduce the development cost required to develop new software. With this in mind, we identify useful resources for which we can build the CORAL tool to satisfy the aforementioned requirements. Moreover, to avoid having to buy a licence to extend existing software, it is imperative to find resources that are open-source and non-proprietary. In addition, allowing the CORAL tool to be open-source will open the possibility for other developers to provide maintenance updates or additional functionality in the future, and at the same time make it more accessible to end-users.

When creating new client-side applications, one can rely on what is called a rich client platform (RCP). An RCP is a platform that provides a minimal set of plug-ins from which new applications can be based. Serving as a tool to make it easier to integrate independent software components. There are several open-source platforms from which one can base an application. Examples are the Eclipse RCP [28] and the Netbeans RCP [82] to name a few. By extending an RCP one can reduce development cost considerably since much functionality is provided by the platform and only needs adjustment to the user-domain. Furthermore, this enables the developer to focus development efforts on the required functionality (user-domain), rather than other necessities such as e.g. appearance, resource management, navigation menus and so on. The Eclipse Project provides several technologies that are useful to the end of creating the CORAL tool. In relation to this, there are several options available with respect to creating an RCP application in the Eclipse environment. These will be discussed in the section that follows.

## 4.2 Eclipse, Tools and Frameworks

The Eclipse Foundation was created in 2001 as an independent non-profit organisation. The organisation's purpose is to serve as a community for individuals and organisations who want to collaborate on developing commercially-friendly open source software [119]. The Eclipse foundation

consists of several independent software projects, with the Eclipse Project serving as the core (top-level) project [121]. The Eclipse project provides core components divided into five sub-projects as can be seen from Figure 4.1 and constitutes the Eclipse software development kit (SDK). These, in turn, support the development of platforms, frameworks integrated development environments (IDE) and other applications.

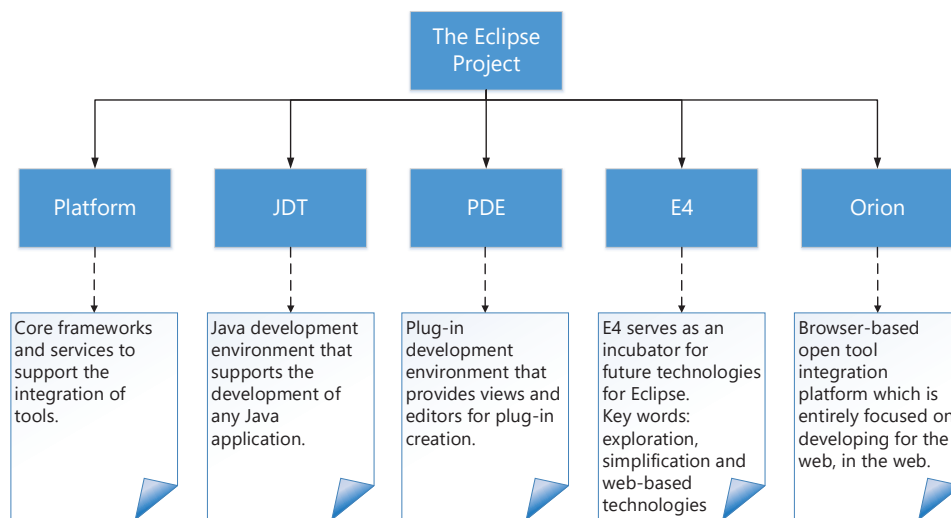


Figure 4.1: The Eclipse project [121] consisting of five sub-projects: Platform [27], JDT [120], PDE [95], E4 [22] and Orion [26].

All software produced by the Eclipse Foundation is made available under the Eclipse public licence (EPL) [32]. According to the EPL, permissions allow for using, modifying and redistributing the software for free. The Eclipse software can also be integrated with proprietary components as part of a commercial product [112].

To the end of creating a client-side application, one can rely on the Eclipse platform, which consists of the platform UI, the standard widget toolkit (SWT), resource management (file/folder management) and debugger. These features are enough in order to create any basic application. However, for our CORAL tool, we are in need of a sequence diagram editor, the UML 2.x specification, a palette of modelling elements and so on. All of these features are available in different plug-ins from a range of Eclipse projects. To develop the CORAL tool one could identify all the needed plug-ins and apply them along with modifications to end up with a modelling tool according to our requirements. However, even if many of the required plug-ins are available, the knowledge to assemble and modify them all in a complete modelling tool would be required. Luckily, to the end of creating editors for modelling languages, Eclipse provides several frameworks to accomplish this.

### 4.2.1 Eclipse Modelling Framework

The Eclipse modelling framework (EMF) is "a modelling framework and code generation facility for building tools based on a structured data model" [23]. Basically, in EMF you can create a model of an application by the means of an EMF-model or *ecore-model*. Ecore is a small and simplified subset of UML consisting of four classes: EClass, EAttribute, EReference and EDataType [112]. Based on the *ecore-model* of an application and a *gen-model*, EMF will generate the corresponding Java interfaces, a UML diagram and an XML schema. The *genmodel* is a decorator model for the *ecore-model* [31]. You can also supply EMF with a specification of the application as either Java interfaces, a UML diagram or an XML schema. EMF will then generate the *ecore-model* along with the other specifications [112]. The latest release of EMF is version 2.13.0, released on June 28th 2017 [23].

### 4.2.2 Graphical Editing Framework

The graphical editing framework (GEF) is a framework to build graphical applications that can be integrated into the Eclipse UI [24]. Furthermore, one can use GEF to create graphical editors for modelling languages [24]. The latest release build of GEF is version 5.0.0, released on June 13th 2017. The new version of GEF uses the JavaFX [63] rendering engine [41], as opposed to the legacy version, which used SWT [117]. See [25, 42] for further information of the components that make up GEF.

### 4.2.3 Graphical Modeling Framework

The graphical modelling framework (GMF) is a framework that combines EMF and GEF to provide a graphical modelling framework. This enables the developer to define meta-models in EMF, and rich graphical editors with GEF. Then, GMF puts all of this together and creates a graphical editor, which can optionally be implemented in an RCP application [43]. The GMF project is divided into three sub-projects: GMF Runtime, GMF Notation and GMF Tooling. The sub-projects follow their own release plan, with their latest releases on June 28th 2017, June 22nd 2016 and June 22nd 2016 respectively.

### 4.2.4 Eclipse Papyrus

Papyrus [91] is an Eclipse-based (RCP) application/tool, created with the intention of providing an "integrated, user-consumable environment for editing any kind of EMF-model" [93]. Moreover, Papyrus has support for UML2 which provides an EMF implementation of the UML 2.x OMG meta model [124]. It also supports domain specific modelling languages (DSMLs) such as SysML and MARTE, and the integration of GMF-based editors [93]. The Eclipse Papyrus project is a sub-project of the model development tools (MDT) project [30], which in turn is a sub-project of the

Eclipse modelling project [29]. The latest release of Papyrus is the 3.0.0 Oxygen release as of June 27th 2017.

### Functionality

As mentioned, Papyrus implements the UML 2.x specification. In addition to this, it provides diagram editors for all the UML diagram types [91]:

- Class diagram.
- Object diagram.
- Package diagram.
- Composite structure diagram.
- Component diagram.
- Deployment diagram.
- Profile diagram.
- Use case diagram.
- Activity diagram.
- State machine diagram.
- Communication diagram.
- Sequence diagram.
- Timing diagram.
- Interaction overview diagram.

Since Papyrus supports the creation of UML profiles, one can create a DSML specified as a UML profile for any of the UML diagram types. Also, Papyrus has support for the definition of model constraints, in terms of either OCL or Java. This will be discussed further in Section 4.4.6. Furthermore, Papyrus is customisable as one can create customised palettes, graphical/textual/tabular notations, customisation for a variety of views, i.e. the different windows inside the application along with creation wizards for your DSML [91]. The supported functionality is documented in the user's- and developer's guide [94]. However, the documentation is a bit lacking and there are several new features that have come after the Oxygen release that is not included in the documentation at the time of writing this thesis. In addition, some documentation is prone to being obsolete since some previously used plug-ins are deprecated. These plug-ins are usually listed in migration guides between release versions. Migration to Papyrus Oxygen is explained in a migration guide [90].

## 4.3 Options for Tool Design

Having looked at several frameworks/tools within Eclipse, we proceed to identify what option to benefit from when creating the CORAL tool.

One option would be to create the CORAL specification in EMF as an *ecore-model*. Then, generate all the classes needed to support the language from the *ecore-model* and *genmodel*. To the end of applying a graphical editor to the model, we could use GEF, then wrap it up in an RCP application. This approach was more common before the introduction of GMF.

Another option is to use GMF which is similar to the process for the first option, although simplified.

Finally, we can benefit from the UML profile provided by Papyrus. This way we first figure out how to adapt the CORAL abstract syntax in a UML profile, specify model constraints and customise according to the customisation options. Since CORAL extends several constructs from UML, using Papyrus to create a UML profile seems like an easier option to implement CORAL with respect to the other options. First, well-known DSMLs such as SysML and MARTE have been implemented as UML profiles in Papyrus. Second, UTP which is needed for the last step in the CORAL approach is present in Papyrus. Third, learning and applying EMF along with GEF would take a considerable amount of time to master in comparison to using UML profiles. Finally, Papyrus is actively updated. For these reasons, to the end of creating the CORAL tool, we will benefit from creating a UML profile for CORAL along with the customisation features of Papyrus.

#### **4.3.1 Plug-in or RCP application?**

Since Papyrus is part of Eclipse, an alternative is to create an RCP application based on Papyrus. This will allow for the creation of a stand-alone tool, branded as the CORAL tool. This is beneficial if we want to restrict the tool to only concern the CORAL DSML. However, if new updates are released for Papyrus, these would have to manually applied to our RCP application, in effect, leading to more maintenance work. The other alternative is to create a Papyrus plug-in with our DSML. The plug-in can use extension points and configurations to limit the Papyrus view when creating a CORAL diagram. Such that information that is irrelevant to CORAL is hidden from the user. Furthermore, a plug-in would allow the user to create models with other DSMLs or UML in the same tool. This is beneficial if one is working with different modelling languages in a project. Also, the input to the CORAL approach is a description of the SUT. The description can be in the form of "system diagrams, use case documentation, system manuals, source code, executable versions of the system, and so on" [33, p. 68]. For this purpose, the user can first use the UML sequence diagram editor to model the system. Then, when going through the steps of the CORAL approach, apply stereotypes with ease. For these reasons, we will deploy our CORAL DSML as a plug-in for Papyrus. In the section that follows, we explain how the CORAL UML profile was created.

### **4.4 Adaptation of CORAL as a UML Profile**

In this section, we describe the process of how CORAL was adapted as a UML profile. As mentioned in Section 3.3 on page 22, we argued why *prototyping* is an appropriate evaluation strategy to utilise during the tool development process. In order to implement the CORAL DSML we apply

the following steps. First, we identify the concepts and relations with respect to CORAL. Second, based on the identified concepts and relations, we specify UML stereotypes and/or data types. Third, we specify the constraints relevant to the CORAL language. Fourth, we customise our diagram editor according to the options available within Papyrus. Finally, we deploy our profile as a CORAL plug-in for Papyrus. We then get our prototype which in turn is evaluated by simulating security tester activities. The adaptation process is depicted in Figure 4.2. As can be seen from the figure, the process is iterative, indicated by the dashed arrow from the fifth step to the first. In the following subsections we will introduce

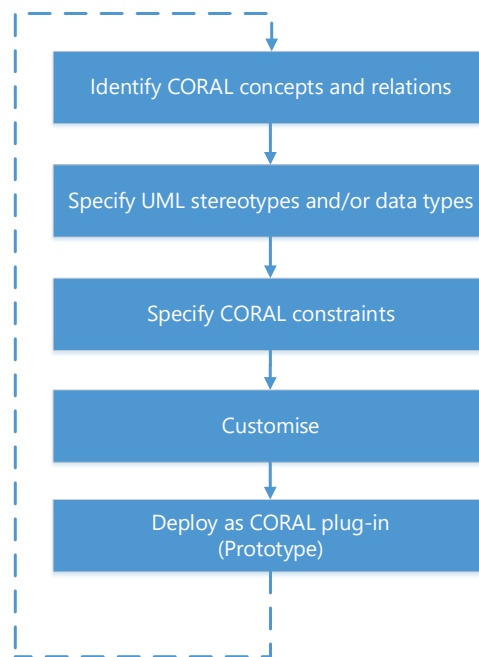


Figure 4.2: CORAL adaptation in a UML profile.

each of the CORAL constructs identified from the abstract syntax [33] and how to specify them as stereotypes. A stereotype is the primary extension construct in UML, that can extend existing UML-constructs and their properties. The version of the CORAL profile that is presented is the final version, after being refined through several iterations according to the prototyping strategy.

#### 4.4.1 Data Types

We review the abstract syntax for the CORAL risk-measure annotations. Since these constructs do not have corresponding UML constructs, we apply them as properties of messages instead of annotations. Such that they can be easily referenced. The abstract syntax is written in EBNF [61]. While non-terminals are written in font *math mode*, terminals are written in font Sans Serif. The terminals that are written in **Bold Sans Serif** represent the syntactical element type [33]:

<i>transmission frequency</i>	=	<i>frequency</i> ;
<i>reception frequency</i>	=	<i>frequency</i> ;
<i>frequency</i>	=	<b>f</b> ( <i>exact</i> , <i>timeunit</i> )   <b>f</b> ( <i>interval</i> , <i>timeunit</i> );
<i>conditional ratio</i>	=	<b>cr</b> ( <i>exact</i> )   <b>cr</b> ( <i>interval</i> );
<i>consequence</i>	=	<b>c</b> ( <i>identifier</i> );

From this we identify two data types we have to implement in our profile, these will be implemented using the class `DataType` which will be represented as an `EDataType` in EMF. The data types are: `Frequency` and `CondRatio` (conditional ratio). Furthermore, according to CORAL abstract syntax frequencies and conditional ratios can be represented either as an *exact* positive real number ( $exact \in \mathbb{R} \geq 0$ ) or an interval of real non-negative numbers including 0 [33]. The time units and consequence values will be represented as enumeration literals. The CORAL abstract syntax does not mention likelihood values [33], however, likelihood values are related to frequencies, as a frequency value will fall within the range of values defined by the likelihood scale. The corresponding likelihood values will also be represented as enumeration literals. Following are the data types and enumerations:

### Frequencies

A frequency can be represented as either an *exact* or an *interval* value. For this reason, we create an *abstract* `DataType` named `Frequency`. `Frequency` has two properties that are in common for both *exact* and *interval* frequencies, namely *time unit* and *likelihood*. Further, we create two data types that are generalisations of `Frequency`, `IntFrequency` and `ExactFrequency`. `IntFrequency` has two properties: *min:Real* and *max:Real*, while `ExactFrequency` has one property: *exact:Real*. The data types are represented in Figure 4.3 in a UML profile diagram.



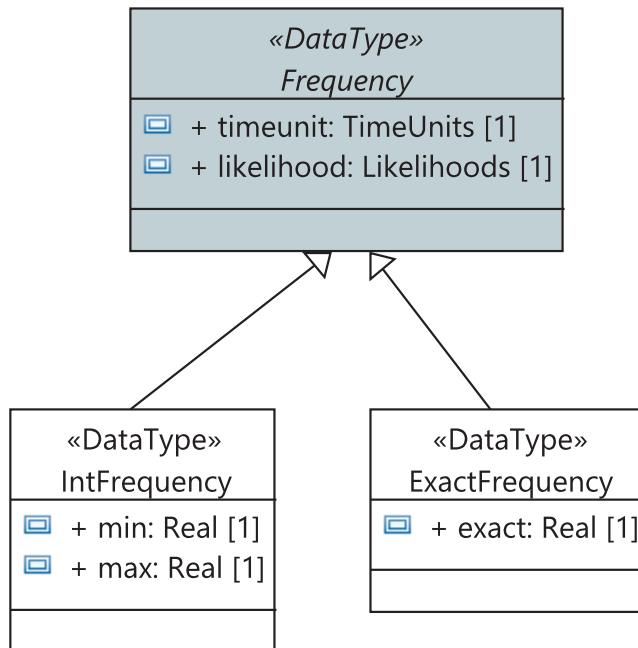


Figure 4.3: The frequency data types.

### Conditional Ratios

Similar to the frequency, a conditional ratio can either be an exact or an interval value. We specify IntCondRatio with two properties: *min:Real* and *max:real*, while ExactCondRatio has one property: *exact:Real*. These can be seen from Figure 4.4.

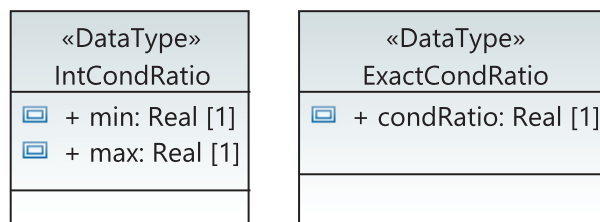


Figure 4.4: The conditional ratio data types.

### Enumerations

The undefined terms from the abstract syntax such as time units, consequences and likelihoods are implemented as enumerations: TimeUnits, Consequences and Likelihoods. The enumeration literals for TimeUnits are second(s), minute(s), hour(s), day(s), week(s), month(s) and year(s) [33]. Enumeration literals for Consequences are: insignificant, minor, moderate, major and catastrophic. Finally, enumeration literals for Likelihoods are: rare, unlikely, possible, likely and certain. These are shown in Figure 4.5.

Data types and enumerations are defined in a nested profile named CoralDataTypes.

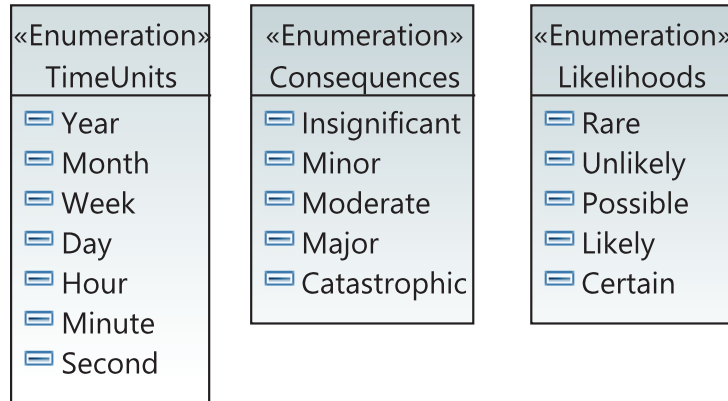


Figure 4.5: The enumerations for TimeUnits, Consequences and Likelihoods.

#### 4.4.2 Lifelines

CORAL defines four new lifeline constructs: asset, deliberate threat, accidental threat and non-human threat. These are semantically equivalent to the UML lifeline. Additionally, a general lifeline is referenced in the syntax. This is basically a UML lifeline and is used in CORAL to "model the system under test, as well as the environment interacting with the system under test" [33]. The identifiers defined in the CORAL syntax are equivalent to the naming conventions of lifelines in UML interactions [33]. Since we implement the CORAL lifelines as stereotypes (extension) of the UML lifeline, they inherit the *name* property. Thus, there is no need to implement another identifier property for CORAL lifelines, as specified in the abstract syntax. The CORAL lifelines are shown in Figure 4.6 in a UML profile diagram. As can be seen from the figure, the CORAL lifelines are defined in a profile named CoralLifelines. This profile is a nested profile within the CORAL profile.

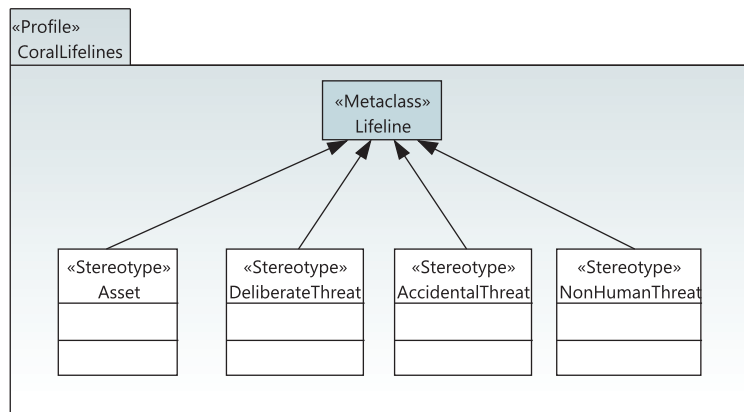


Figure 4.6: The CORAL lifelines, extending the UML Lifeline meta class.

### 4.4.3 Messages

The CORAL language defines five different kinds or categories of messages that belong to three collective terms. These being *risky message*, *deleted message* and *unwanted incident message*. The abstract syntax is written in EBNF [61]. While non-terminals are written in font *math mode*, terminals are written in font Sans Serif. The terminals that are written in **Bold Sans Serif** represent the syntactical element type [33]:

<i>message</i>	=	<i>riskymessage</i>   <i>unwantedincident message</i>   <i>deletedmessage</i> ;
<i>risky message</i>	=	<b>rm</b> ( <i>identifier</i> , <i>transmitterlifeline</i> , <i>receiverlifeline</i> , <i>riskymessage category</i> , <i>transmission frequency</i> , <i>conditionalratio</i> , <i>reception frequency</i> );
<i>unwanted incident message</i>	=	<b>uim</b> ( <i>identifier</i> , <i>transmitterlifeline</i> , <i>asset lifeline</i> , <i>transmission frequency</i> , <i>consequence</i> );
<i>deleted message</i>	=	<b>dm</b> ( <i>identifier</i> , <i>transmitterlifeline</i> , <i>receiverlifeline</i> );
<i>risky message category</i>	=	general   new   alter;

We see from the abstract syntax that some properties are equivalent to that of a message in UML. Such that when we create a stereotype for each message kind that will extend the UML meta class Message, these properties are inherited. These include *transmitter lifeline*, *receiver lifeline* and *identifier*. The identifier property is equivalent to the *name* property for a UML message. We define the profile CoralMessages nested within the Coral profile. Further, the transmission and reception frequencies can be of either the IntFrequency or the ExactFrequency data type. For this reason, we create two UML profiles for CORAL messages, namely CoralIntervalMessages and CoralExactMessages. These profiles are nested within the CoralMessages profile depicted in Figure 4.7 on page 37. As can be seen

from the figure, the collective terms RiskyMessage, DeletedMessage and UnwantedIncident extend the UML Message meta class. RiskyMessage represented in both the CoralIntervalMessages and CoralExactMessages profile as an abstract stereotype with the properties *transmissionFrequency*, *receptionFrequency* and *conditionalRatio*. Furthermore, GeneralMessage, NewMessage and AlteredMessage are generalisations of the RiskyMessage stereotype. The UnwantedIncident stereotypes in both CoralIntervalMessages and CoralExactMessages extend the UML Message meta class. We could have added another abstraction layer to UnwantedIncident messages on the same level as DeletedMessages, with an abstract UnwantedIncidentMessage stereotype. This stereotype would have had one generalisation each in the CoralIntervalMessages and CoralExactMessages profiles. However, this was deemed to be an unnecessary abstraction layer.

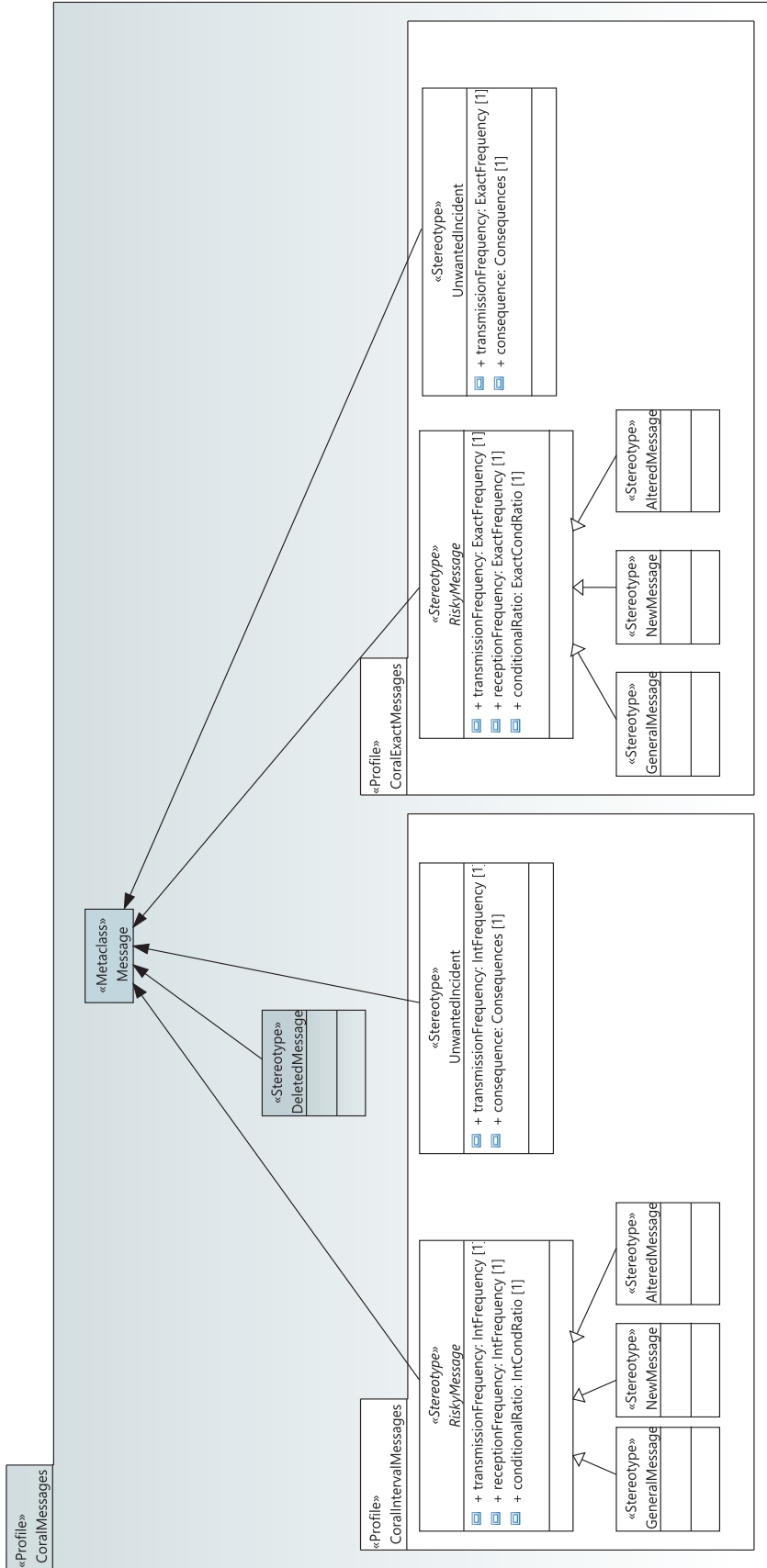


Figure 4.7: The CoralMessages profile.

#### 4.4.4 Risk-Measure Annotations

As the CORAL risk-measure annotations do not have corresponding UML constructs, they were implemented as properties of frequencies and messages. Consequently, we do not have any graphical constructs to represent them in the diagram. To solve this issue, one can use the UML Comment meta class. A Comment is a textual annotation which can be attached to a set of UML elements with an association end called the *context link* [88, p. 40]. By creating stereotypes that extend the Comment meta class, we get textual annotations with the «Stereotype» extension tag for each respective risk-measure annotation. These risk-measure annotations are not directly associated with the property values of the annotated elements, i.e. updating the annotations' values does not update the annotated elements' property values. The risk-measure annotations are implemented for visual expressiveness. The risk-measure annotation stereotypes are defined in the CoralRiskMeasureAnnotations profile seen from the UML profile diagram in Figure 4.8 below.

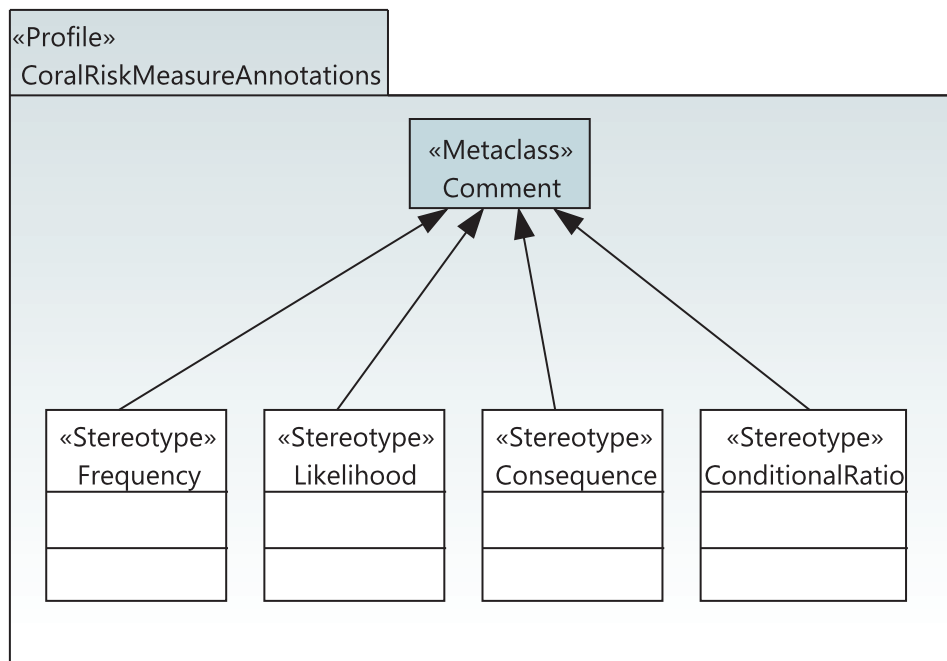


Figure 4.8: The CoralRiskMeasureAnnotations profile.

#### 4.4.5 CORAL Constraints

Erdogan [33, p. 64] describes a set of constraints that affect certain modelling constructs. With respect to risk-measure annotations, according to the abstract syntax Erdogan defines the following constraints:

1. Frequency annotations may not be attached to a deleted message, nor may it be attached on the receiving end of an unwanted incident message.

2. The conditional ratio may not be attached to a deleted message or an unwanted incident message.
3. The consequence annotation may only be attached to unwanted incident messages [33, p. 64].

In addition to these constraints, we identify a fourth constraint related to the unwanted incident message. As Erdogan states: "We see from the syntax that the lifeline receiving an unwanted incident message is an asset lifeline. The asset lifeline is modelled in an interaction to represent the asset that is harmed by an unwanted incident" [33, p. 64]. This yields our fourth constraint:

4. The receiver lifeline of an unwanted incident message may only be an asset lifeline.

Furthermore, Erdogan mentions constraints with respect to frequency values and conditional ratios [33]:

5. Frequencies and conditional ratios may only be positive real numbers, including 0.

Now that we have identified our constraints, we need to figure out how to apply them to our DSML. To this end, as mentioned in Section 4.2.4 on page 28, constraints are supported in Papyrus by expressing them in either Java or OCL. For our DSML we will benefit from OCL, which is described further in the next subsection.

#### 4.4.6 Object Constraint Language

The object constraint language [86] (OCL) is a language used to express "all kinds of (meta) model queries, manipulations and specification requirements" [10]. It was originally developed by IBM in 1995 and was used to specify constraints for UML. Although in 1997 it became integrated with UML and has become a key component in model-driven engineering [10]. Version 2.4 is the current version of OCL [86]. OCL has several applications, the following are examples of OCL expressions [10, 86]:

- Invariants, these are conditions that at any time must be satisfied in any instance of a model element for which they are defined. These are always boolean expressions, e.g. an invariant can be that the value for a frequency must be bigger or equal to zero.
- Initialisation of class properties.
- Derivation rules that describe how derived values are to be computed.
- Query operations. OCL is not a programming language, so it is not possible to specify any program logic or flow control. For this reason, OCL expressions are not directly executable. However, they can indicate the result of a query operation.

- Pre- and post-conditions associated with an Operation or other behavioural feature.

In the section that follows, we proceed to define constraints according to those discussed in Section 4.4.5, expressed in OCL.

#### 4.4.7 CORAL Constraints in OCL

We now consider how to define OCL constraints for the aforementioned CORAL constraints. Since risk-measures have been implemented as properties for the message types, constraints 1-3 become obsolete. The reason for this is that from the message stereotype definitions the affected messages do not have these properties. DeletedMessage does not have any of the constrained properties from constraints 1 and 2. The unwanted incident message only has transmissionFrequency and consequence properties, such that constraint 1 and 2 are satisfied. No other message has the consequence property, therefore, constraint 3 is also satisfied. The risk-measure annotations that are implemented for visual purposes can be attached to elements such that they violate these constraints. Constraints can be made for these constructs as well. However, they can become very complex and tedious to create accurately. Due to this, constraints for the risk-measure annotations that extend the Comment meta class are not prioritised. The following OCL invariants are defined for constraints 4-5:

##### Constraint 4.

The receiver lifeline of an unwanted incident message may only be an asset lifeline. This constraint affects UnwantedIncident in CoralIntervalMessages and CoralExactMessages.

##### OCL expression:

```
self.base_Message.receiveEvent.oclAsType(UML :: MessageOccurrenceSpecification).covered → collect(l|l) → collect(l|l.extension_Asset)
.oclIsUndefined() = Bag{false}.
```

This constraint became overly complex. This is due to a limitation in the EMF Java model, as the UML2 evaluation returns an EList for the InteractionFragment::covered. The *collect(l|l)* will then extract the element from the EList. Finally, the *extension\_Asset* is collected from *l*. The *oclIsUndefined()* will return true if the extension is not of the Asset stereotype. Therefore, in order for the constraint to be satisfied, *l.extension\_Asset* has to be defined for the *receiveEvent* of an unwanted incident message.

##### Constraint 5.

Frequencies and conditional ratios may only be positive real numbers, including 0. This constraint affects IntFrequency, ExactFrequency, IntCondRatio and ExactCondRatio.



OCLE expression for interval:

$self.min \geq 0.0 \text{ and } self.max \geq 0.0$ .

OCLE expression for exact:

$self.exact \geq 0.0$  for ExactFrequency,

$self.condRatio \geq 0.0$  for ExactCondRatio.

The constraints can be validated in the Papyrus tool by right clicking on a model  $\rightarrow$  validation  $\rightarrow$  validate model. Examples of output are given in Figure 4.9 and 4.10. Having defined our UML profile according to the CORAL abstract syntax, and also defining constraints, we proceed to customise our diagram editor.



Figure 4.9: An example of a violated constraint with respect to the UnwantedIncident message.

The screenshot shows the Model Validation window in Papyrus. It has three tabs: Properties, Model Validation (selected), and References. The window contains a table with two columns: Description and Element. There are three rows of violations, each with a red exclamation mark icon in the Description column.

Description	Element
The 'RecipientIsAsset' invariant is violated on '<<UnwantedIncident>>RootElement:Inter...	<<UnwantedIncident>> <<Message> M...
The 'FrequencyIsPositiveValue' invariant is violated on '<<UnwantedIncident>>RootElemen...	Int Frequency Year
The 'MaxGreaterThanMin' invariant is violated on '<<UnwantedIncident>>RootElement:Int...	Int Frequency Year

Figure 4.10: Examples of violated constraints.

## 4.5 Customisation

As mentioned in Section 4.2.4, Papyrus offers customisability with respect to several features in the tool. These customisation options are described in the tool smith guide which is part of the Papyrus guide which is available in the Papyrus tool by clicking help.

### 4.5.1 Graphical Notation

The customisation for graphical notation is supported by CSS rules. The CSS rules can be found in tool smith guide  $\rightarrow$  CSS style sheets  $\rightarrow$  supported properties. These rules support adding a custom icon to any model element with a stereotype applied. This customisation feature works well for many UML diagrams in Papyrus. However, it is not satisfactory for sequence diagrams. This is due to the way the Lifeline class is represented in the GMF-based diagram. In UML a lifeline is represented by a rectangle with a dashed line from the bottom middle of the rectangle, stretching to the bottom of the diagram as can be seen from Figure 4.6. On the left side of

the figure is a representation of the compartment for lifelines in Papyrus. Applying CSS rules enables you to position the icon in different positions in the compartment, e.g. top, left, right centre and so on. To position our icon according to the graphical notation we choose the top position. Furthermore, we would like to hide the rectangle and position the name property under the icon. To hide the rectangle, we would have to set the compartment visible = false. As you can see from the lifeline on the left however, this would hide the whole lifeline. In addition to this, there is currently no way to display the lifeline name outside the rectangle. If the compartments had been divided into two sub-compartments as in the lifeline to the right, the desired behaviour could have been possible. Although, it is important to note, that Papyrus is actively updated, and this may be corrected in an upcoming release. The tasks for the Photon release of Papyrus will be announced on the Papyrus wiki [92].

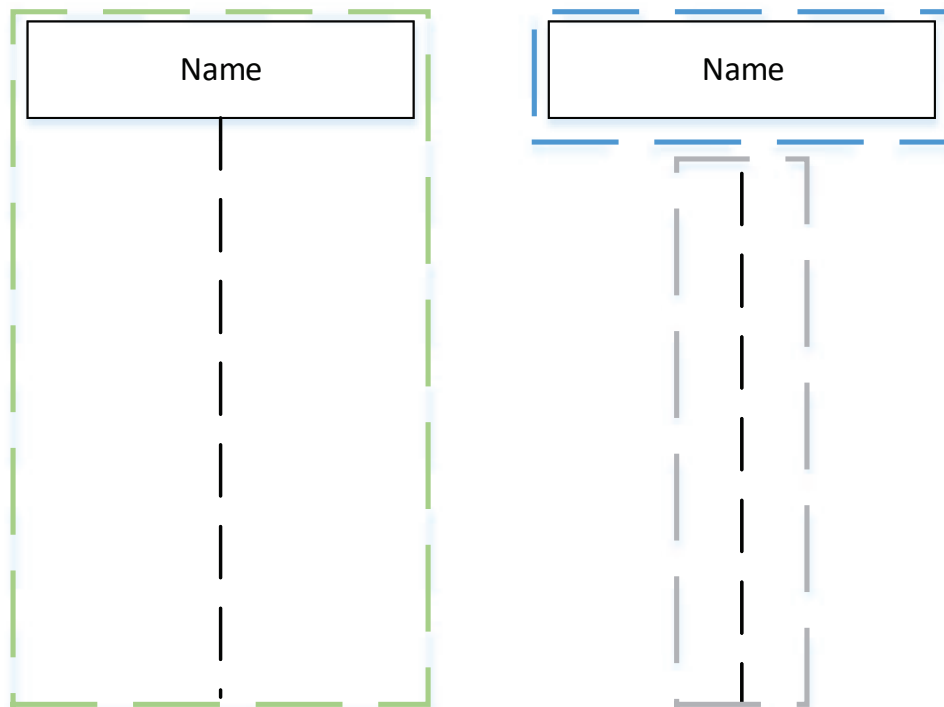


Figure 4.11: The lifeline compartment in Papyrus displayed on the left. On the right, examples of sub-compartments that would make CSS customisation for lifelines easier.

#### 4.5.2 Palette

The Papyrus palette customisation was previously achieved by the use of an XML schema, however, it is now replaced by the palette configuration file. This file can be created by the use of the palette customisation tool within Papyrus. One can specify the drawers (folders) for the palette, and assign them icons. Within these drawers, one can add all the UML elements and apply stereotypes to them e.g. our message types, lifelines etc. To

include the palette with the CORAL diagram, one can benefit from the customisation of the view. The CORAL palette can be seen from Figure 4.12 below.

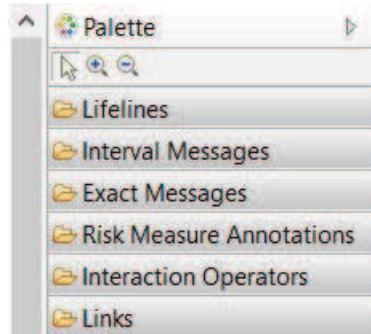


Figure 4.12: The CORAL palette.

## 4.6 Deploying the Profile as a Plug-in

The last step is to deploy the CORAL profile as a plug-in. To achieve this we have to generate a static profile, this is explained in the Papyrus guide→user guide→tasks→using UML profiles→generating static profiles. First, we apply the EPackage stereotype from the Ecore profile to our CORAL profile. Second, we create a gen-model from our UML file. In this step, it is important to set the runtime jar property of the root to *true*. Third, we generate our model code. This will generate the Java classes for our nested profiles along with classes for the model elements and their validators. Finally, we add the following extension points:

Table 4.1: The CORAL plug-in extension points.

Extension point	Description
org.eclipse.emf.ecore.generated_package	We specify an extension point for each nested profile.
org.eclipse.emf.ecore.uri_mapping	A URI mapping to the profile, both relative and absolute paths.
org.eclipse.uml2.uml.generated_package	Specifies the location of the XMI-id of the profile.
org.eclipse.papyrus.uml.extensionpoints.UMLProfile	Registration of the static profile name, path, description and provider properties.

We can then proceed to export our plug-in with our relevant resources specified in the build properties. The plug-in can be downloaded from the Bitbucket repository: <https://bitbucket.org/vetlevo/no.uio.ifi.coral.profile/>.

## 4.7 Researched-Based Design Summary

In this chapter, we argued why it is beneficial to rely on existing tools/frameworks as opposed to developing the tool from scratch. To attain our overall goal, tool support within the domain of risk-driven security testing. To that end, an adequate solution was to benefit from using the UML profile in Papyrus, a technique in which a modeller can tailor the UML meta model to represent a DSML. Moreover, Papyrus already has the UTP defined. This provides the opportunity of using the tool to carry out the steps of the CORAL approach related to testing as well.

After deciding to create the CORAL profile using Papyrus, we argued why the profile should be deployed as a plug-in for Papyrus, instead of creating an RCP application based on Papyrus. We then proceeded to create the CORAL profile based on the abstract syntax defined by Erdogan [33], before applying customisation and deploying it as a plug-in. During the UML-profile development in Papyrus, several UML profiles were created and refined through iterations. With the customisation options provided by Papyrus, it is possible to add a graphical notation to the domain specific modelling language constructs. Regretfully, for sequence diagrams, this customisation was not satisfiable. Graphical notation can be added to lifelines, however, the lifeline box will still be visible. Currently, there is no functionality (provided by CSS rules) that supports adding the CORAL icons and hiding the lifeline rectangles. Further, there is no option to add a graphical notation to edges/messages. For this reason, the CORAL tool currently supports the creation of threat models by the use of textual notation. This is in the form of textual stereotype annotations. Consequently, our empirical study will investigate whether this has any effect on how threat models are interpreted, this is further discussed in the chapter that follows.

## Chapter 5

# Evaluation - Empirical Study

In Chapter 4 we went through the process of developing the artefact, namely the CORAL tool. In this chapter, we outline and conduct an empirical study by the means of an experiment, intended to evaluate the differences in notations, graphical or textual.

As Sjøberg et al. [107] point out, empirical studies are needed to develop or improve processes, methods and tools for software development. In our case, the empirical study is particularly concerned with the development of a tool. Further, they point out that the term *experiment* is inconsistently used within the software engineering community and is often used synonymously with *empirical study*. However, an experiment is rather a means of conducting an empirical study, much like a survey or a case study. In this chapter the overlying process of evaluation will be referred to as an empirical study, adopting the six steps of the quality improvement paradigm (QIP framework) provided by Basili et al. [6]. QIP is a generic improvement cycle, which is widely accepted as a recommended way to work with the improvement of software development [99, 129]. However, it can also be used as a framework for conducting empirical studies [129]. Furthermore, this framework was adopted during the ESERNET project [15] to structure empirical studies. The ESERNET project was carried out to raise the awareness within the software engineering community to systematically conduct empirical studies [129]. Figure 5.1 illustrates the six steps of the QIP framework, referred to as "A Single Empirical Study" in the ESERNET project [99].

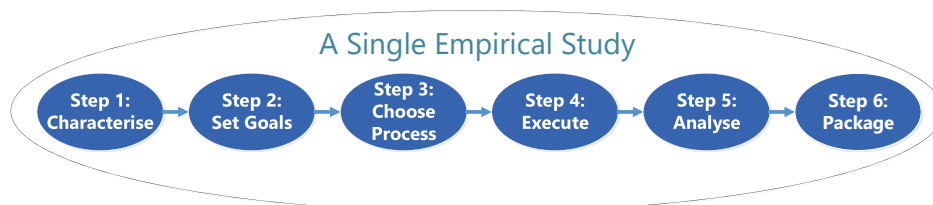


Figure 5.1: The six steps in the "A Single Empirical Study" framework [99].

Each step in this framework is broken into several substeps, which are further explained in each respective sub section of this chapter, in relation

to our study.

Steps one to five are described in sections 5.1-5.5. However, the sixth step, package, is left out. This is due to the fact that this step refers to the documentation produced as an output of the previous five steps, which is covered by this chapter.

## 5.1 Characterisation of the Study

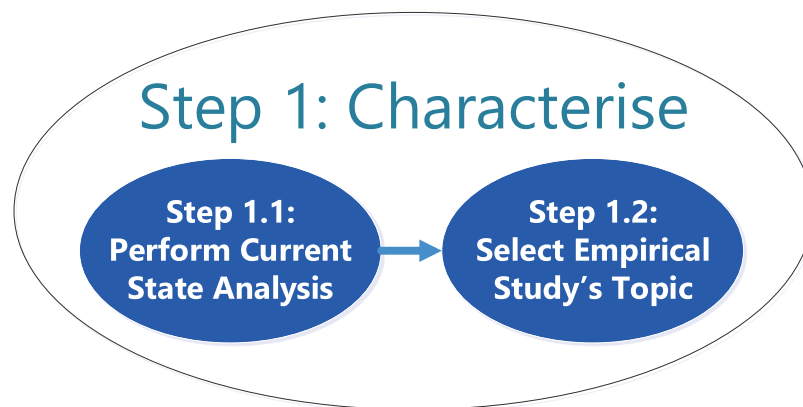


Figure 5.2: The first step in the empirical study framework "A Single Empirical Study" [99].

### 5.1.1 Current State Analysis

Due to the development issues summarised in Section 4.7, the evaluation of the tool will focus on the differences between the proposed tool, and the tool that has been developed. The main distinction between the two being textual and graphical notation. To establish a baseline for our study, we conduct a systematic mapping-study to get a high-level overview of similar studies within model-driven engineering (MDE). A systematic mapping study is a process that aims to gather evidence of a topic on a high level of granularity. This is an ideal strategy if it is likely that little evidence exists on the topic, or if the topic is very broad. A systematic mapping study is often conducted before a systematic literature review [66]. The focus of the mapping study involves (1) identification; (2) selection of primary studies; (3) study quality assessment [3]. The study was conducted using the major digital libraries like IEEE, ACM using Oria, IEEE Xplore and Google Scholar. The search terms were: Comparison of AND ("graphical and textual" OR "icons and textual" OR "graphical and stereotype" OR "icons and stereotype") AND ("notation" OR "representation") AND ("empirical study" OR "experiment" OR "controlled experiment" OR "quasi-experiment").

Following the results from the systematic mapping study, we identify several studies. While most of them are concerned with the comparison of

two different modelling approaches, some are concerned with specifically comparing textual and graphical notation.

### **Comparison of Different Modelling Approaches**

De Lucia et al. [17] conduct an experimental comparison of ER and UML class diagrams for data modelling, in three controlled experiments. The experiments were conducted in an educational setting with university students. The results demonstrated better comprehension levels with using UML class diagrams. Hadar et al. [51] investigate comprehensibility between Use Case and Tropos to express requirements models, also in three experiments. Results show that Tropos outperformed Use Case in terms of comprehensibility, but required more effort, such that the productivity, in the end, was similar. This study was also conducted with university students as participants.

Labunets et al. [67] studied a slightly more relevant topic in comparison with our own, as they compared a tabular versus a graphical notation. The compared modelling notations was that of NIST [113] and CORAS [74]. They found that tabular representations were in some cases more effective than the graphical notations in simple comprehension tasks, and likewise for some more complex tasks.

### **Comparison of Textual and Graphical Notations**

Hogganvik and Stølen [54] empirically investigated the differences between using UML notation with stereotype annotation and UML notation with stereotype annotation and graphical icons. This study involved both professionals and students. Their findings report that the participants using graphical notations were able to conclude faster, however, not reaching a higher correctness of interpreting the models.

Meliá et al. [79] compared graphical and textual notations for the maintainability of MDE domain models in a pilot study. The study was performed with students as participants. The study showed that the participants using textual notation performed better with regard to analysability coverage and modifiability efficiency.

#### **5.1.2 Topic of our Empirical Study**

The empirical study will try to uncover if a graphical representation in comparison to a textual representation of the modelling constructs aids the participants' comprehension of the threat models, and efficiency to solve the given tasks. Having described the topic of our empirical study, and also taking note of similar experiments, we formulate the goal for our study in the section that follows.

## 5.2 Set Goals

This section introduces the goal and research questions that constitute the empirical study. Referring to Figure 5.1 on page 45, we now focus on the second step. The substeps in the second step is illustrated in Figure 5.3 below. We benefit from Goal, Question and Metric GQM for establishing our goal and research questions [99, 110]. This involves considering the following questions:

- Object of study – what is being studied?
- Purpose – what is the intention of the study?
- Quality Focus – Which effect is being studied?
- Perspective – From which perspective the object is studied?
- Context – Where is the study conducted?

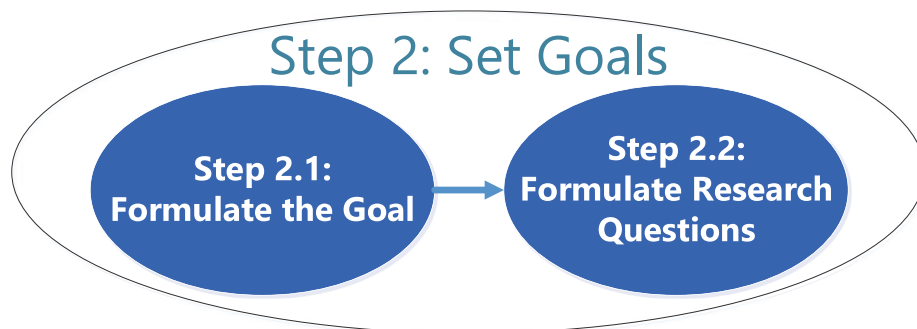


Figure 5.3: The second step in the empirical study framework "A Single Empirical Study" [99].

### 5.2.1 Formulate the Goal

To aid our research efforts, a goal must be established, and research questions along with a hypothesis devised. Having said that, the overall goal of this study is to evaluate two different ways to illustrate the modelling constructs in the CORAL modelling language. The two being either textual or graphical notation. When referring to textual notation we refer to the notion of using UML stereotype annotations to represent the CORAL modelling constructs. The graphical notation refers to the icons used for the CORAL modelling constructs as proposed by Erdogan [33]. The evaluation will focus on understanding and interpreting threat models with respect to *comprehensibility*, *efficiency*. Furthermore, we want to assess the participants' satisfaction with respect to the study.



## 5.2.2 Formulate Research Questions

The research questions addressed in this experiment are the following:

- RQ1:** Will the use of a DSML using either textual or graphical notation to represent threat models affect the objective performance of comprehensibility? That is, is there a measurable difference with respect to effectiveness (the degree to which something is successful in producing the desired result [19]) between the use of the two notations.
- RQ2:** Will the use of a DSML using either textual or graphical notation to represent threat models affect the participants' efficiency in solving the provided tasks?

## 5.3 Choose Process

In this section we go into the details of the study process, step three in Figure 5.1 on page 45. This step is divided into five substeps, before proceeding to step four, the execution of our process. The five steps are illustrated in Figure 5.4 below.

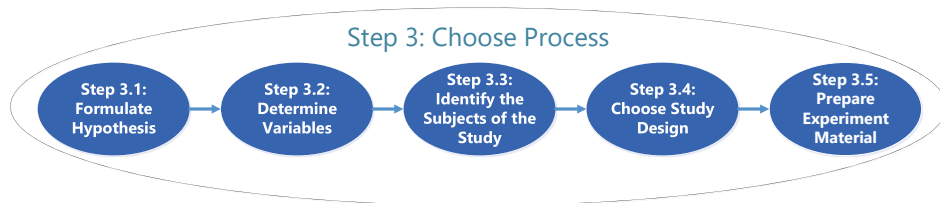


Figure 5.4: The third step in the empirical study framework "A Single Empirical Study" [99].

### 5.3.1 Formulate Hypothesis

Following the goal and research questions we established in Section 5.2, we need to devise a hypothesis. This leads to hypothesis  $H_0$  as listed in Table 5.1.

Considering that the CORAL modelling language is relatively new at the time of writing this thesis, there exists no empirical evidence to support that either notation is superior to the other. For this reason, alternative hypotheses  $H_1$  and  $H_2$  are formulated. These are also listed in Table 5.1.

Table 5.1: Hypothesis for the empirical study

Null Hypothesis	Alternative Hypothesis
$H_0$ Threat models with textual notation are equally comprehensible in comparison to threat models with graphical notation.	$H_1$ Threat models with textual notation are more comprehensible than threat models with graphical notation.
	$H_2$ Threat models with graphical notation are more comprehensible than threat models with textual notation.

### 5.3.2 Determine Variables

In order to assess and compare the two different notations with respect to the hypothesis established in Section 5.3.1, there is a need to identify the variables for our study. In Section 5.3.4, we argue that a controlled experiment is a suitable study design. Thus, we identify the independent and dependent variables for the experiment.

Independent variables are considered to be the input to an experiment. In effect, the motivation for an experiment is to investigate whether variations in the independent variables have an effect on the dependent variables (output of the experiment), see Figure 5.5. "Confounding factors are variables that may affect the dependent variables without the knowledge of the researcher" [129]. It is therefore important to identify these factors to attain higher validity for our study. This is related to the threats to validity, discussed further in Section 5.5.6.

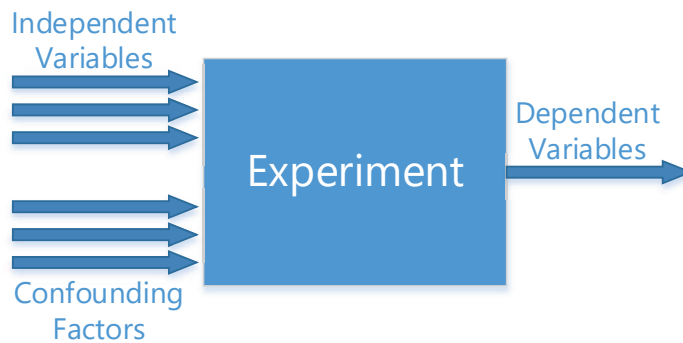


Figure 5.5: Independent variables, confounding factors and dependent variables. Figure adapted from [129, p. 14]

For this experiment, the independent variables is the threat model representation whose notation can hold two different values: textual or graphical notation. The dependent variables in the study are *comprehensibility*, *efficiency* and *satisfaction*.

*Comprehensibility* is measured by effectiveness – the participant is able to reach successful task accomplishment. In other words, the participant is able to develop and comprehend models [19, 100]. Furthermore, comprehensibility will be measured by taking into account the participants' task score.

*Efficiency* – the participant is able to develop and comprehend a model relatively quickly and correctly according to the syntax and semantics of the modelling language [100]. Having in mind that there currently is no empirical evidence to suggest what is a relatively quick comprehension of the CORAL threat models, a comparison between the two populations with respect to efficiency is required.

*Satisfaction* – the participant will state his/her subjective contentment gained from the experience [100]. Satisfaction, in this case, is mostly concerned with the experiment, and not with using the modelling language or the CORAL tool directly. For this reason, the questions regarding satisfaction will be related to the experiment.

### 5.3.3 Identifying the Subjects of the Study

The participants in this study consist mainly of graduate students and a number undergraduates within the field of computer science. Some of which are currently working or studying. A total of 16 participants of which 9 are graduate students, and 6 are undergraduates. The recruitment was done through the researcher's own network and selected based on specific characteristics such as being a student within computer science and having knowledge of programming. This type of sampling is also called purposive sampling [105]. For more information about the participants' demographics, see Section 5.4.2.

### 5.3.4 Study Design

As discussed in Section 5.2, the study aims to uncover differences between notations across a population. Thus, the study aims to achieve generality of the comprehensibility of different notations across a population. Moreover, as briefly described in Section 5.1, most of the identified articles design an experiment for their study. Likewise, we will design what is referred to as a controlled experiment [129]. Since we have identified a single independent variable that we change for each population (control group), by assigning it a value of either textual or graphical notation (referred to as a treatment of each control group). This study design is what Wohlin et al. [129] refer to as *standard design 1*. For this design, Wohlin et al. recommend using a t-test for the hypothesis testing. The reasoning for choosing the t-test is discussed further in Section 5.5.3 on page 72.

There are several factors to consider when designing the process of the experiment. Keeping in mind that the focus for the experiment is the difference in notations, the process obviously has to involve interpreting threat models in some form. Ideally, the evaluation with respect to the CORAL tool would involve the experiment being carried out by tool interaction. However, there are disadvantages with this approach. One disadvantage is that if the participant were to solve tasks on the participant's own computer. The participant would have to set up the environment correctly in Papyrus on the participant's own computer. For one, this requires more effort from the participant, such that the participant might lose interest in participating. Furthermore, it may lead to errors while setting up the environment. Consequently, the participant can become frustrated, and more dialogue with the researcher may be required to set it up correctly. The other alternative would be to set up a laboratory experiment. The environment would have to be set up on a number of computers in a lab, depending on the number of lab-sessions. This obviously requires more preparation and also resources as several machines would have to be requested for this purpose. Furthermore, To set up a laboratory experiment would require all participants to be available to participate in the same physical location within a reasonable time frame. Additionally, conducting the experiment using the CORAL tool would require that the participant would be given training in both the tool and the notation. Again, requiring more effort from the participants and more preparation.

To summarise, conducting the experiment with the tool would be ideal with respect to the evaluation of the tool. However, because of time restrictions and lack of resources, this is deemed unfeasible for this study.

If the experiment process does not require interaction with the tool, an alternative is to create a questionnaire with questions regarding threat models. This can be implemented in a survey tool and can avoid the situation where the participant has to set up the environment manually. Unfortunately this lacks realism with respect to the tool, but seems more feasible to implement in terms of time, resources and participants' willingness to participate. Moreover, to reach out to as many participants as possible, it is beneficial to organise the survey using an online survey tool. This way, the participants can answer the survey whenever they have time, wherever they may be. Among the articles identified in Section 5.1.1, two studies were conducted using online tools. Meliá et al. [79] used Qualtrics [97] for the questions regarding subjective performance. Labunets et al. [67] used SurveyGizmo [114] to conduct the study, however, it was conducted in a laboratory setting.

In conclusion, no study design is faultless. However, in order to gain insight about our study's topic, an appropriate experiment process has to be established with the restrictions discussed previously in mind. For this reason, the experiment will be conducted using questionnaires, by the use of an online survey tool. The selection of an adequate online survey tool is discussed later in this section. The chosen experiment process is represented by Figure 5.6.

First, we collect information regarding each participants' educational

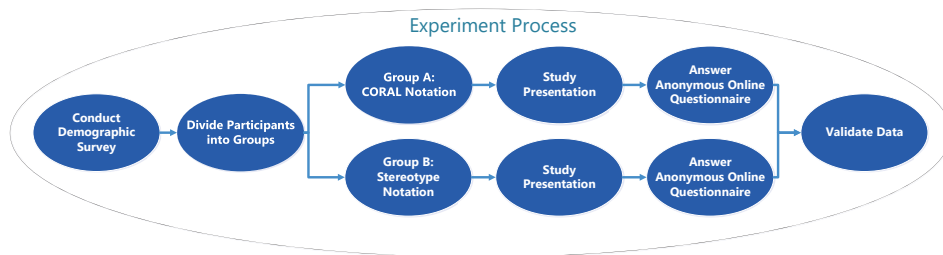


Figure 5.6: The experiment process. Group A is given material with graphical notation, whilst Group B is given material with textual notation

background and skill set. This way, we can divide the participants in two groups fairly based on competence. This is to avoid having a skewed division of the participants with respect to competence. Consequently, we can further specify our controlled experiment as a quasi-experiment. Since we do not randomise the participants among the groups [104]. Hadar et al. [51] also considered dividing into groups, but went for a randomised approach. This was due to time restrictions as a consequence of conducting the experiment in a single lab session. The participants are divided into Group A and Group B. Group A is given material with graphical notation, while Group B is given material with textual notation. After dividing the participants into groups, we provide a presentation, intended to give an introduction to the modelling constructs with their respective notation. The presentation is identical for both groups apart from the notation (see Appendix C and Appendix D). Along with the presentation, participants are given the main questionnaire with tasks to solve. Finally, each participant response is reviewed and the data is validated to ensure that it can be part of the statistical analysis.

### Ethical Considerations

When gathering research data while conducting an empirical study that concerns human participants, one has to consider the issue of privacy. Depending on where you conduct the study, you have to contact the data protection official for research in the country where the study is conducted. In Norway (where this study is conducted), the appropriate institution is the Norwegian centre for research data (NSD) [84]. In our experiment, to be able to contact the participants, a list of participants is needed with their contact information (e-mail addresses). In addition, the online survey tool utilised to conduct the demographic survey will have to store the e-mail address of the responses, such that the participant can be given a group for the main questionnaire. This data is considered to be personal data [59].

As a consequence of collecting personal data [16], the research project is subject to notification to the NSD. For this reason, a notification form was submitted on 25th of May 2017. Following NSD's evaluation process of the project, reviewing the notification form, along with the experiment

material. The project was approved on the 14th of June 2017 with the expected end date of 30th of June 2017. By then, the personal data has to be anonymised. This implies that all e-mail addresses and names have to be disassociated from the gathered data. Further, the personal data have to be deleted.

### **Survey Tool Selection**

There are several things to consider when choosing a survey tool. Survey logic, time stamps per question, timer functionality, to name a few. Does the tool provide all the needed features? Moreover, as discussed previously, the survey tool has to store personal data about the participant for the demographic survey. If one uses an external data processor, one would have to make a "Data processor agreement, which regulates and ensures the processing of personal data." [85]. To circumvent this issue, we can use the survey tool provided by the University of Oslo (Nettskjema). However, Nettskjema does not provide the timer functionality needed for the main questionnaire. Therefore, there is a need to find another survey tool that has timer functionality and can retain anonymity. Among the identified survey tools were:

- SurveyGizmo [114].
- SurveyMonkey [115].
- Zoho Survey [136].
- Eval&Go [36].
- Google Forms [44].
- SurveyPlanet [116].
- LimeSurvey [73].
- QuestionPro [98].

In the end, the choice led to the Eval&Go tool. This tool has all but one of the required features, time stamps. Which would be required to look at the individual time per question. However, it was the only tool that provided timer functionality, to enforce time limits on a question. In addition, it was the only tool that included all the features free for students. Furthermore, it has the ability to ensure anonymity. If the survey is set up correctly, one can avoid storing e-mail/IP addresses, browser information or cookies. Also, by providing all the participants with the same survey link, no response can be traced back to a particular participant.

### **5.3.5 Preparation of Experiment Material**

The participants were given (1) A letter of consent (see Appendix E); (2) A demographic survey; (3) A presentation; (4) The main questionnaire with tasks to solve.

## Demographic Survey

The demographic survey employs a Likert scale with five values (no knowledge, minor knowledge, some knowledge, good knowledge, expert). The questions for the demographic survey can be seen from Table 5.2, and are mostly related to the participants' skill set and working background. Most of the studies identified in Section 5.1.1 included a demographic survey in their experiment. In our demographic survey, information such as age, gender, and so on are left out. Such information is recognised as indirectly identifiable personal data by the Norwegian centre for research data (NSD) [59]. To make participation as anonymous as possible, we intend to limit these types of questions. The questions requesting indirectly identifiable personal data concerns educational and work-related background, as this increases the granularity of the sorting criteria for the next step in the experiment process.

Table 5.2: The Questions for the demographic survey. The logic column signifies what has to be true for that particular question to appear in the survey tool. The "-" symbol implies that this question does not implement logic.

Q#	Question	Answer(s)	Logic
Q1	What's your current occupation?	Student, working, other	-
Q2	Are you studying full-time or part-time?	Full-time OR part-time	If Q1 == student
Q3	Are you working full-time or part-time?	Full-time OR part-time	If Q1 == working
Q4	What's your job title?	Open question	If Q1 == working
Q5	Do you have any working experience within information technology or engineering?	No OR yes	-
Q6	How many years?	Open question	If Q5 == yes
Q7	What's your education level?	Bachelor's degree OR master's Degree OR ph.d OR other	-
Q8	Please specify other	Open question	If Q7 == other
Q9	Study Programme	Information technology OR other	-
Q10	Please specify other	Open question	If Q9 == other

Continued on next page

Table 5.2 – continued from previous page

Q#	Question	Answer(s)	Logic
Q11	Further Specify Programme	Design, use and interaction OR language and communication OR nanoelectronics and robotics OR programming and networks OR technical and scientific applications OR other	-
Q12	Please specify other	Open question	If Q11 == other
Q13	Knowledge of UML modelling	No knowledge OR Minor knowledge OR Some knowledge OR Good knowledge OR Expert	-
Q14	Knowledge of UML sequence diagrams	No knowledge OR Minor knowledge OR Some knowledge OR Good knowledge OR Expert	-
Q15	Do you have any work-related experience with model-driven development?	No OR yes	-
Q16	How many years?	Open Question	If Q15 == yes
Q17	Knowledge of risk assessment or risk analysis	No knowledge OR Minor knowledge OR Some knowledge OR Good knowledge OR Expert	-
Q18	Do you have any work-related experience with risk assessment or risk analysis?	No OR yes	If Q17 >= Minor knowledge
Q19	How many years?	Open question	If Q18 == yes
Continued on next page			



**Table 5.2 – continued from previous page**

Q#	Question	Answer(s)	Logic
Q20	Knowledge of UI design or usability	No knowledge OR Minor knowledge OR Some knowledge OR Good knowledge OR Expert	-
Q21	Do you have any work-related experience with UI design?	No OR yes	If Q20 >= Minor knowledge
Q22	How many years?	Open question	If Q21 == yes

## Tasks

For this experiment, the tasks need to address comprehensibility. To this end, the tasks will be concerned with model-reading, similar to one of the task categories utilised by Hadar et al. [51]. To be able to differentiate between the notations, it is important to have a mixture of easy and difficult tasks. If the tasks are either all easy or all difficult, there probably would not have been a noticeable difference between the control groups. Over the years, there has been done much research on the human brain's processing capability and complexity of tasks [52, 53]. There have also been proposed methods for calculating the complexity of tasks [52], for example, the method proposed by Wood [131]. Generally, the more concepts and variables a human is exposed to, the more complex the question becomes. The tasks developed for this experiment does not implement a method for complexity calculation. However, the difficulty of the tasks is determined by the aforementioned characteristics in mind. For example, to ease the learning curve for the participants, we can leave out the elements in the CORAL modelling language that are not different between the two notations. This includes risk-measures, such as conditional probabilities, transmission/reception frequency and consequence values as described in Section 2.1.5. However, it can be argued that if these elements are left out the experiment's realism is affected. Since the textual notation along with risk-measures might lead to cluttering of the threat models, i.e. too much text in the model.

When designing the tasks, it is also important to keep in mind that the questionnaires are answered online. Thus, we do not have any direct control of the environment where the participant answers the questionnaire. For that reason, it may be beneficial to enforce a time limit for each question. In an attempt to avoid the situation where the participant overestimates the amount of time required for a given task. Furthermore, if there had been no time restriction, there is a possibility that the easier tasks would be correctly answered by most of the participants. For this reason,

each task is given a time limit such that the participant has to comprehend the diagram within a reasonable amount of time.

The task set was created and reviewed by other researchers in short pre-studies. This enabled the tasks to be put under test and by gaining valuable feedback, the task descriptions and models were refined. In total there were seven iterations and seven task sets devised. The final task set is divided into two parts. Part 1 has less complicated tasks concerned with identifying different modelling constructs. Part 2 contains tasks of higher complexity compared to Part 1 and introduces some new concepts required to interpret the threat models. The tasks are related to several threat models based on well-known attacks towards a web application. See Table 5.3 for a list of all the tasks. Part 1 consists of tasks 1-6, while Part 2 consists of tasks 7-13. For the full task set along with corresponding threat models, see Appendix A and Appendix B. At the end of the main questionnaire there are post-experiment questions related to satisfaction with the experiment, see Table 5.4. In total there are 13 questions and a max score of 24 points. The total allotted time for the tasks is 1440 seconds or 24 minutes.

Table 5.3: Tasks for the main questionnaire. The task scores with an addition sign indicate that the score is determined by the number of correctly stated items as expected for the answer, e.g. one point per correct quantity of a given message type.

Task #	Description	Score	Time(s)
1	How many new messages are explicitly modelled in the threat model below?	1	60s
2	How many altered messages are explicitly modelled in the threat model below?	1	60s
3	How many deleted messages are explicitly modelled in the threat model below?	1	60s
4	How many messages are explicitly modelled between lifeline L1 and L3 in the threat model below, and what are their types?	1+1+1	60s
5	How many messages are explicitly modelled between lifeline L1 and L4 in the threat model below, and what are their types?	1+1	60s
6	How many messages are explicitly modelled in the threat model below, and what are their types?	1+1+1+1	60s
Continued on next page			

Table 5.3 – continued from previous page

Task #	Description	Score	Time(s)
<b>Total</b>	-	<b>12</b>	<b>360s/ 6min</b>
7	Some messages in this diagram are supposed to be deleted messages, can you spot which?	1+1	180s
8	In the threat model below, the asset SourceCode can potentially be harmed. Which statements are the most accurate with respect to the threat model?	1	300s
9	According to the model, describe how the hacker causes the unwanted incident to occur.	Cat. 1-4	
10	In the threat model below, the asset UserData can potentially be harmed. Which statements are the most accurate with respect to the threat model?	1	300s
11	According to the model, describe how the hacker causes the unwanted incident to occur.	Cat. 1-4	
12	According to the model, describe how the hacker causes the unwanted incident to occur.	Cat. 1-4	300s
13	According to the model, describe all possibilities of where the attack might fail.	1+1	
<b>Total</b>	-	<b>15</b>	<b>1080s/ 18min</b>

Table 5.4: Table containing all the questions for the post-experiment questionnaire. The questions that use the Likert scale have five values from strongly disagree to strongly agree.

Q#	Question	Answer
1	I had enough time to solve the given tasks.	Likert scale
2	The objectives of the empirical experiment were clear to me.	Likert scale
3	The task descriptions were clear, and I understood what to do.	Likert scale
4	I experienced no difficulties in understanding the threat models.	Likert scale
Continued on next page		

**Table 5.4 – continued from previous page**

<b>Q#</b>	<b>Question</b>	<b>Answer</b>
5	I experienced no difficulties in answering questions in Part 1.	Likert scale
6	I experienced no difficulties in answering questions in Part 2.	Likert scale
7	I think the presentation given to me before the questionnaire provided me with enough information to solve the tasks.	Likert scale
8	Do you have any comments with respect to the tasks given in the experiment?	Open
9	Do you have any further comments about the experiment?	Open

### **Task Scores**

The task scores for the task set is based on the number of correctly stated items. That is, for questions that require more than one item, a single point is given for each correctly stated item. For example, for questions that involve identifying a quantity of more than one particular message type, a point is given for each correctly identified quantity of a message type. In addition, open-ended questions are rated by categories, see Table 5.5.

**Table 5.5: Table with all the task score categories.**

<b>Score Categories</b>		
<b>Cat.#</b>	<b>Description</b>	<b>Score</b>
1	The participant is not able to describe the attack in any meaningful or definite way.	0
2	The participant is only able to describe the attack by category, e.g. SQL Injection XSS and so on.	1
3	The participant is able to vaguely describe how the hacker causes the unwanted incident to occur in plain text.	2
4	The participant is able to describe the sequence of events that lead up to the unwanted incident. I.e. the sequence of messages. Either by referencing the message names directly, or describing their occurrence in sequence.	3

## 5.4 Execution

So far, we have characterised our empirical study, set goals and formulated a hypothesis. Further, we determined our variables and designed a controlled experiment intended to provide statistical data for our hypothesis test. In this section, we go through Step 4, the execution of our experiment. This step can be seen from Figure 5.7 and contains three substeps.

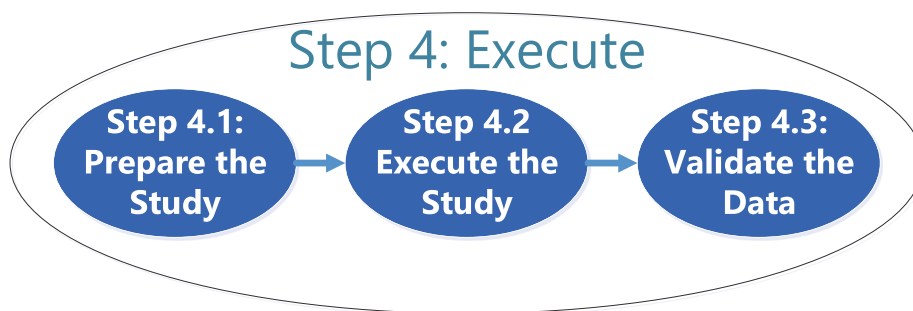


Figure 5.7: The fourth step in the empirical study framework "A Single Empirical Study" [99].

### 5.4.1 Study Preparation

The study preparation for this experiment consists mainly of setting up the questionnaire in the survey tool, namely Eval&Go. After setting up the survey, the form was tested and verified to be correct according to the task set and the time restrictions enforced for each task. Furthermore, options such as the *save* (allowing participants to finish a previously uncompleted response) feature and *back* feature (to change a previous answer) were disabled such that the tool does not store any IP addresses, browser information or cookies. Preparation also involved setting up the participant list along with e-mail invitations including the demographic survey and letter of consent.

### 5.4.2 Study Execution

#### Demographic Survey

On June 14th 2017, the invitations for the demographic survey were sent to all the participants by e-mail. By June 18th, all participants had submitted their answers. Following the gathered data, we identify four groups of participants based on the current occupation: five students, eight currently working, two who are both studying and working and one specified as other. Their knowledge profiles can be seen from Table 5.6. There was an average of 3.5 years working experience within information technology or engineering, with one participant having 20 years working experience.

There were no participants that had working experience with MDE. One participant had two years of working experience with risk assessment or analysis. Finally, four participants had working experience with UI design or usability, with one, two four and eight years each.

Table 5.6: Knowledge profiles

	None	Minor	Some	Good	Expert
<b>UML modelling</b>	0	2	11	3	0
<b>Sequence diagrams</b>	0	3	10	3	0
<b>Risk assessment or analysis</b>	2	6	6	2	0
<b>UI design or usability</b>	2	4	8	0	1

The criteria given most weight when dividing the participants into two groups are UML modelling, sequence diagrams and risk assessment or analysis knowledge. Moreover, the participants are equally distributed with respect to education level. The two groups can be seen from Table 5.7 and Table 5.8. In terms of working experience within information technology and engineering, Group B has five years on average, whilst Group A has two years. However, Group B has a participant in this data field with 20 years of working experience. UI design and usability knowledge and its respective working experience were not weighted heavily in the group assignment.

Table 5.7: The participants for Group A. WE = years of working experience, D = degree, B = bachelor's degree, M = master's degree. Knowledge in terms of Likert values: UML modelling, SD = sequence diagrams, R = risk assessment or analysis, UI-US = UI Design and usability.

<b>Group A</b>									
<b>P#</b>	<b>W</b>	<b>D</b>	<b>UML</b>	<b>SD</b>	<b>MDE _WE</b>	<b>R</b>	<b>R_WE</b>	<b>UI-US</b>	<b>UI-US _WE</b>
P1	0	B	2	2	0	2	0	2	0
P2	2	B	1	1	0	1	0	0	0
P3	1	B	3	3	0	2	0	0	0
P4	1	M	2	2	0	0	0	2	0
P5	0	M	2	2	0	1	0	2	0
P6	4	M	2	1	0	1	0	2	2
P7	8	M	2	2	0	2	0	4	8
P8	0	M	3	3	0	3	0	0	0
<b>Average</b>	<b>2</b>	<b>M</b>	<b>2.12</b>	<b>2</b>	<b>0</b>	<b>1.5</b>	<b>0</b>	<b>1.5</b>	<b>1.25</b>

Table 5.8: The participants for Group B. WE = years of working experience, D = degree, B = bachelor’s degree, M = master’s degree. Knowledge in terms of Likert values: UML modelling, SD = sequence diagrams, R = risk assessment or analysis, UI-US = UI Design and usability.

<b>Group B</b>									
<b>P#</b>	<b>WE</b>	<b>D</b>	<b>UML</b>	<b>SD</b>	<b>MDE _WE</b>	<b>R</b>	<b>R_WE</b>	<b>UI-US</b>	<b>UI-US _WE</b>
P9	1	B	2	2	0	0	0	1	1
P10	20	B	2	2	0	1	0	2	0
P11	5	B	2	2	0	1	0	1	0
P12	0	M	2	2	0	2	0	2	0
P13	5	M	1	1	0	1	0	2	0
P14	9	M	2	2	0	3	2	2	4
P15	0	M	2	2	0	2	0	1	0
P16	0	M	3	3	0	2	0	1	0
<b>Average</b>	<b>5</b>	<b>M</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>1.5</b>	<b>0.25</b>	<b>1.5</b>	<b>0.62</b>

## Main Questionnaire

The main questionnaire was distributed to the participants using an anonymous survey link on June 18th 2017 by e-mail. All answers were submitted by June 25th. The tasks scores in full can be seen from the tables in Appendix F.

### 5.4.3 Data Validation

Since the questionnaire is carried out using a survey tool, the task of data validation simply consists of exporting the answers via the tool and storing them in e.g. an Excel sheet. In addition, the data validation involves giving scores based on the data according to our score system discussed in Section 5.3.5.

## 5.5 Analysis of Results

Having designed and executed our controlled experiment, starting from Step 1 to Step 4, we now analyse the gathered data in Step 5. Step 5 can be seen from Figure 5.8 and involves the following substeps: Step 5.1 visualise the data, Step 5.2 use descriptive statistics, Step 5.3 accept or reject hypothesis.

In order to do a good analysis of experimental data gathered from an experiment, it is important to look at the data from different perspectives, with a variety of methods [105]. In our experiment, there were no assumptions about the differences in task scores for our main questionnaire between Part 1 and Part 2. Keeping in mind that Part 1 and Part 2 have a different level of difficulty, this difference in difficulty would ideally affect task scores correspondingly. However, it is not straightforward to justify a

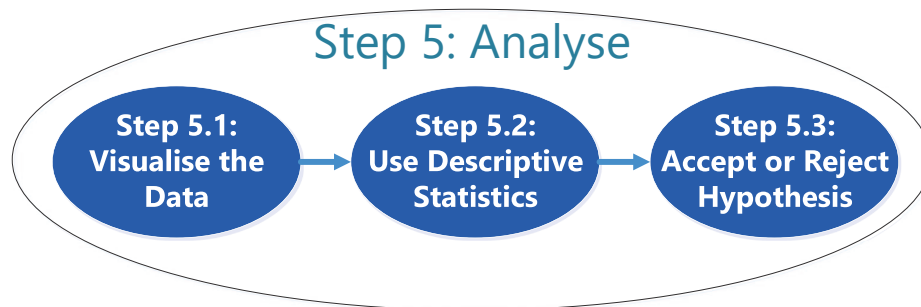


Figure 5.8: The fifth step in the empirical study framework "A Single Empirical Study" [99].

ratio between these task scores based on difficulty. For this reason, we can analyse the results from three different perspectives among our two control groups. First, we analyse the total score by combining the score from Part 1 and Part 2. Second, we analyse the score of Part 1. Third, we analyse the score of Part 2. Moreover, we need to put into our considerations the sample size for our control groups. Obviously with a sample size of eight in each group, and a total of 16 participants, the strength of our hypothesis testing will suffer. As larger sample sizes yield higher precision in statistical tests. Also, larger sample sizes can disprove data points interpreted as outliers in smaller sample sizes. For example, if the outlier was not due to pure chance. An outlier is a value that appears to deviate significantly from the other values in the data set. This can be due to an error in measurement, and if identified as such, be neglected from the analysis. The outlier may also be an extreme occurrence in the variability inherent in the data, and thus should be considered in the statistical analysis [50]. With this in mind, we proceed to visualise our gathered data.

### 5.5.1 Data Visualisation

When visualising our data, it is important to use a technique that appropriately conveys the data in a way that represents an accurate distribution of the data. As Madsen points out: "*Graphs (charts, plots, etc.) are suited to get a feel of patterns, structures, trends and relationships in data and thus are an invaluable supplement to a statistical analysis.*" [75]. There are several visual representations one can use to visualise data. Among those are bar charts, histograms, scatter plots, box plots to name a few. For our analysis, we benefit from box plots generated by IBM SPSS [56]. The box plot shows each quartile ( $Q_1$ ,  $Q_2$ ,  $Q_3$ ,  $Q_4$ ) of the data, the median, and both extreme values at each whisker [77]. Such that each box separated by a line shows the distribution of 25% of the data set starting from the bottom whisker, each line represents the start and end of a quartile. Additionally, the box plot shows outliers given by  $o^n$ , where  $n$  is the record in the data set identified as an outlier.

Figure 5.9 shows a box plot of the total score for the main questionnaire.



The box on the left represents the distribution of Group A, while the box on the right represents that of Group B. The plot for Group A reports an outlier of record 4, having a score of 5. This record has a low score due to the participant leaving several answer boxes empty. This might be because the participant either did not know how to answer the questions or models. It can also be because the participant was not interested in participating. For this reason, we can conduct an analysis for both situations. One where the outlier is included, and one where it is disregarded. If we read the plot of Group A with the outlier included, we can tell that the distribution is highly skewed to the minimum score. However, if this value is disregarded, it looks approximately normally distributed as shown in Figure 5.9. The box

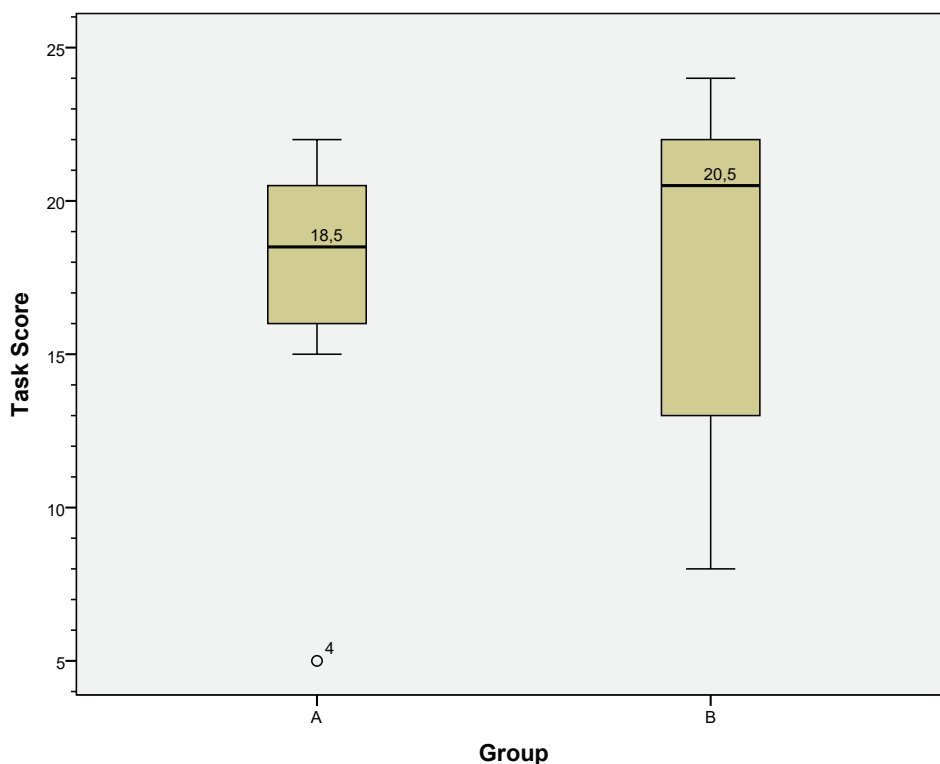


Figure 5.9: Box plot for the total score for Group A and B.

plot for Group B does not include any outliers. This is due to the fact that there is more than one value that deviates from the rest of the values in the set. These values are scores of 8 and 11. The values within  $Q_1$  and  $Q_2$  are widely spread compared to  $Q_3$  and  $Q_4$ , making the plot skewed towards smaller values. Finally, the plot for Group A has a median of 18.5, min and max of 5 and 22 respectively. Group B has a median of 20.5, min of 8 and max of 24. If we compare the two,  $Q_3$  and  $Q_4$  for Group B has considerably better scores than that of Group A. However Group A has less variation of scores in its  $Q_2$ .

From Figure 5.10 we see the box plots for the total scores of Part 1 in the main questionnaire. As can be seen from the plot in Figure 5.9, both Group A and Group B are highly skewed towards the left (if we include

the outlier for Group A). We also note that the difference between  $Q_4$  and  $Q_3$  for Group B is small. When we compare the plots of Group A and B in this instance, it seems that Group B has consistently higher scores than Group A. In the plot for Group A, however, the outlier from previously has not been identified as significant. This is due to another participant also having a low score for Part 1, but a high score for Part 2. Group A and B have medians of 10.5 and 11 respectively. Further, Group A has min and max of 3 and 12. While Group B has min and max of 6 and 12.

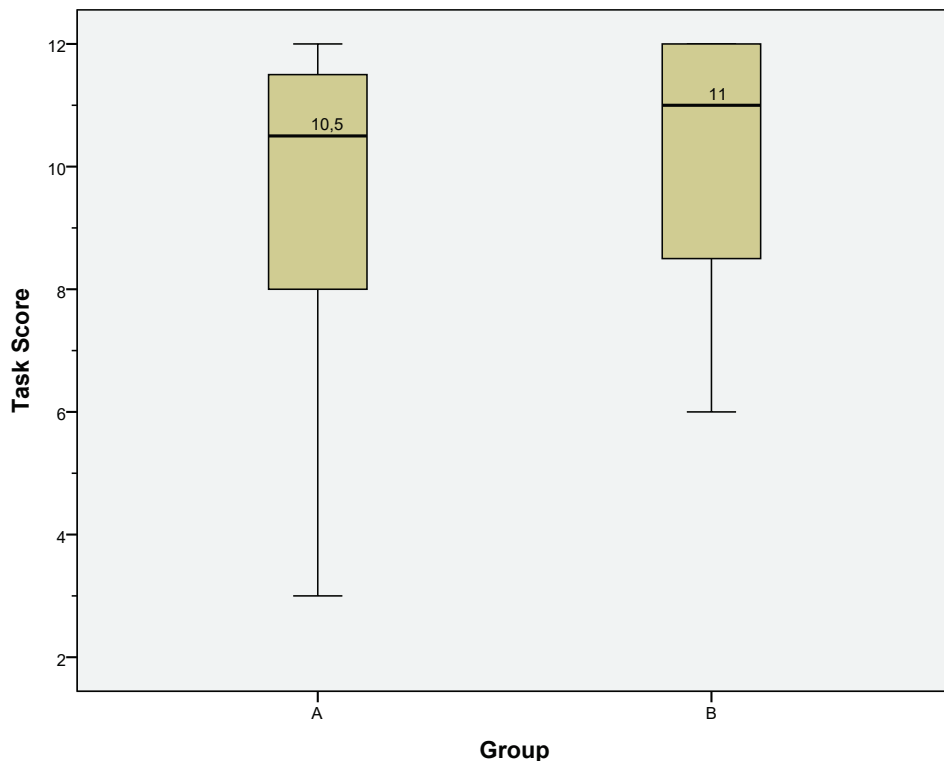


Figure 5.10: Box plot for the total score of Part 1 for Group A and B.

Figure 5.11 shows the last perspective of our analysis, the box plots for the scores of Part 2. The box plot for Group A seems to have a normal distribution, again, if we keep in mind that the outlier from before is included in  $Q_1$ . Group B looks to be quite variable in  $Q_2$ . Additionally, we observe that Group A has a median of 8.5 and Group B of 10. Group A's min and max are 2 and 12 respectively, while Group B has 3 and 12 respectively.

To summarise, the box plots of the three different perspectives do not give any clear indication if there is any difference between the two treatments. It may seem, however, that Group B has a slight improvement over Group A. If we exclude the outlier from Group A, their distributions look to be approximately normally distributed for the total score. However, if we look at Part 1 and 2 individually, the distribution is not as normally distributed. We proceed to apply more descriptive statistics before deciding on which hypothesis testing method to use.

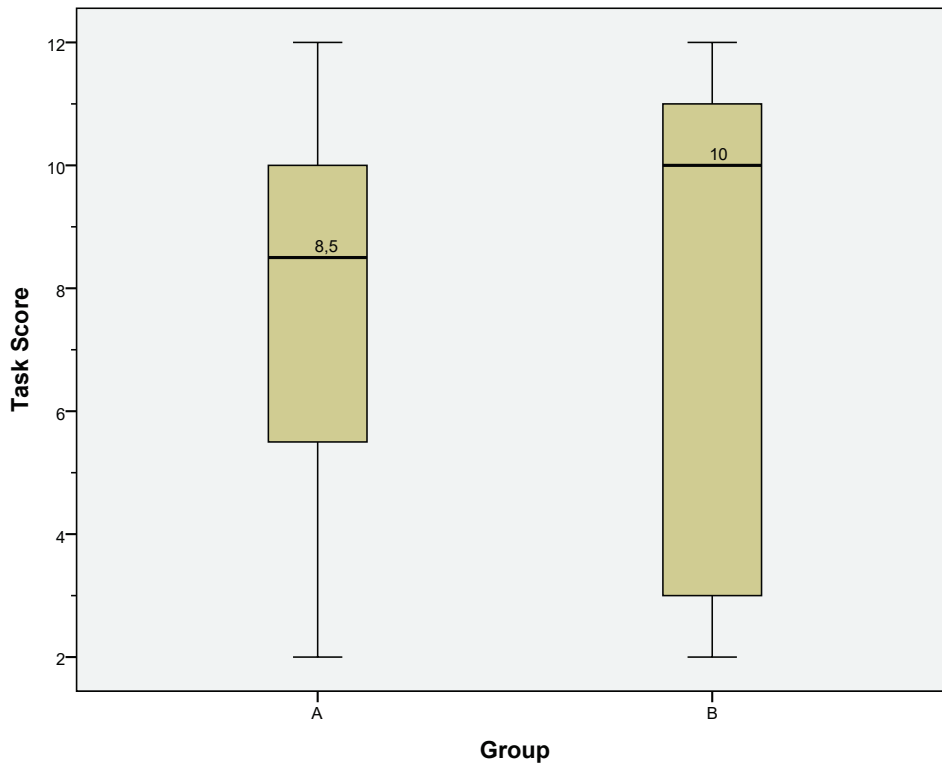


Figure 5.11: Box plot for the total score of Part 2 for Group A and B.

### 5.5.2 Applying Descriptive Statistics

Descriptive statistics is a general term for methods to summarise and describe data [37, 75]. This includes using tables, charts (as in Section 5.5.1), and statistical calculations such as averages, percentages, variances, standard deviation and so on. We proceed to describe the calculations we will use.

#### Arithmetic Mean

The arithmetic mean, often called the average of a data set is the summation of all values divided by the number of entries. The calculation is most often used to see the central tendency in the data set [37, 75], however the mean can often be misleading if there are extreme values included in the set. In statistics, the arithmetic mean is often referred to as either population mean or sample mean. For our experiment, we will refer to the mean as the sample mean. The sample mean  $\bar{x}$  is given by:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} = \frac{x_1 + x_2 + \dots + x_n}{n} \quad (5.1)$$

#### Median

The median is the value "in the middle" of ordered numbers, i.e. the value that splits the data set into two equally sized parts [75]. If the data set's size

is an odd number, the median is the middle value. If the data set's size is an even number, the median is the mean of the two central values [37].

### Standard Deviation

The standard deviation is among the most commonly used measure of spread [37, 75]. The calculated value is the mean distance from the values in the data set to the sample mean. In other words, we get a measure of how much the values in the data set deviates from the sample mean. The standard deviation is calculated by taking the square root of the variance. The standard deviation  $\sigma$  is given by:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (5.2)$$

### Variance

Variance is another commonly used measure of spread, its value is the average of the squared distances between the values in the data set and the sample mean [75]. Variance  $s^2$  is measured in square units, given by [37]:

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (5.3)$$

### Standard Error

The standard error, sometimes abbreviated SE, is the standard deviation of the sampling distribution of a statistic [37, 75]. For our calculations we find the standard deviation of the sample mean, this can be calculated by:

$$SE_{\bar{x}} = \frac{\sigma}{\sqrt{n}} \quad (5.4)$$

The size of  $SE_{\bar{x}}$  depends on the data set size. For larger data sets, the  $SE_{\bar{x}}$  will be smaller and vice versa.

### Skewness and Kurtosis

Skewness and kurtosis are measures one can use to check whether the data follow a normal distribution [75]. These measures are particularly useful if your data visualisation method does not accurately represent the distribution. The skewness value indicates how *skewed* the distribution is compared to a normal distribution [75]. While a value close to zero indicate a symmetrical distribution, positive and negative values indicate a right- or left-skewed distribution respectively.

The kurtosis gives an indication of how big the *tails* (distributions distant from the mean) of the distribution is [75]. A normal distribution has a value of 0, while positive and negative values indicate larger and smaller tails than a normal distribution respectively. A distribution with

a positive kurtosis implies a distribution that is more steep towards the top than a normal distribution. A negative kurtosis implies a distribution that is more flat towards the top than a normal distribution [75]. One may accept larger deviations from 0 for kurtosis than for skewness. However, the values for kurtosis that fall within an acceptable range to be classified as acceptable for a normal distribution depends on the sample size [75]. The ranges can be seen from Table 5.9. Note that for our sample sizes which are  $n = 8$  the ranges does not cover this size. For this reason, the kurtosis may be biased, however, it may give an indication whether the distribution is approximately normally distributed or not. The skewness

Table 5.9: Ranges for the accepted values for kurtosis that is approximately normally distributed, table adapted from [75].

$n$	Min. Kurtosis	Max. Kurtosis
25	-1.2	2.3
100	-0.7	1.1
400	-0.4	0.5

and kurtosis require comprehensive calculations. To this end, IBM SPSS [56] will be utilised to carry out the calculations. We proceed to apply the aforementioned calculations to our data set. The calculations as output from IBM SPSS can be found in Appendix G.

### Descriptive Statistics for the Total Score

Descriptive statistics for the total score of both Group A and Group B can be seen from Table 5.10. We see that the min and max are not very different between Group A and Group B. If the outlier in Group A is excluded, the min for Group A is bigger than the min of Group B with a score of 15 and 8 respectively. The means are similar between the groups. However, without considering the outlier, Group A has a mean of one point more than Group B. Medians differ by two points, however, as discussed previously they are similar if we exclude the outlier.

Furthermore, an interesting observation is the variance, where Group B has bigger variance than Group A. This is due to a slightly higher standard deviation. With the exclusion of the outlier, Group A's standard deviation is more than halved. Consequently having a noticeable lower variance compared to Group B. The standard deviation for both groups indicate that the values are spread. Moreover, the standard error of the mean indicates that our sample mean deviates with approximately two points for both groups. Both groups have negative skewness, indicating that the distributions are left-skewed. This confirms our observation from the box plots in Section 5.5.1. Group A with the exclusion of the outlier has a skewness value of  $-0.373$ , indicating that it is reasonably normally distributed. The kurtosis values for Group A and Group B differ from each other by some margin. While the kurtosis value for Group B is well within the acceptable range established earlier ( $n = 25$ ), Group A is

outside with a difference of 1.656. Group A also falls outside the range with the exclusion of the outlier with a difference of 0.1. To summarise, the

Table 5.10: Descriptive statistics applied to the total score for Group A and Group B. The max score for the whole task set is 27.

<b>Total Score</b>			
	<b>Group A</b>	<b>Group B</b>	<b>Group A (no outlier)</b>
<b>Minimum</b>	5	8	15
<b>Maximum</b>	22	24	22
<b>Mean (<math>\bar{x}</math>)</b>	17.13	17.88	18.86
<b>Median</b>	18.50	20.50	20.00
<b>Variance (<math>\sigma^2</math>)</b>	29.554	34.411	6.476
<b>Std. Deviation (<math>\sigma</math>)</b>	5.436	5.866	2.545
<b>Std. Error (<math>SE_{\bar{x}}</math>)</b>	1.922	2.074	.962
<b>Skewness</b>	-1.862	-.827	-.373
<b>Kurtosis</b>	3.956	-.812	-1.314

descriptive statistics tell us that the two groups are similar, with Group B having a slightly better score than Group A, being less skewed and more normally distributed as well. With the exclusion of the outlier however, Group A looks to have a better mean score with a lesser median than Group B. Further, this yields Group A with a higher precision of measurement than Group B, with a lower  $\sigma$  and  $SE_{\bar{x}}$ . Additionally, Group A gains a smaller skewness value, indicating an approximately normal distribution.

### **Descriptive Statistics for the Score of Part 1**

We now turn our attention towards the descriptive statistics for Part 1, see Table 5.11. The extreme values are similar with and without the outlier. With both groups having participants achieving a max score. Means are also similar, with a difference of 0.75 in favour of Group B before the exclusion of the outlier. After, the mean is in favour of Group A with a difference of 0.16. A high mean score suggests that most of the participants understood the tasks for Part 1 for both groups. The variance is higher for Group A than Group B, due to a higher standard deviation. However, as noticed before, the exclusion of the outlier yields Group A with lesser variance and higher precision of measurement. Skewness values indicate left-skewed distributions for both groups. However, the exclusion of the outlier yields Group A with a lesser left-skewed distribution. Kurtosis values for both groups indicate approximate normal distribution, with Group B having a kurtosis value very close to 0.

### **Descriptive Statistics for the Score of Part 2**

Finally, we take a look at the descriptive statistics for Part 2, seen from Table 5.12. The extreme values are equal for both groups, however, exclusion of the outlier yields Group A with a higher min. Further, the means

Table 5.11: Descriptive statistics applied to the Part 1 score for Group A and Group B. The max score for Part 1 is 12.

<b>Part 1 Score</b>			
	<b>Group A</b>	<b>Group B</b>	<b>Group A (no outlier)</b>
<b>Minimum</b>	3	6	8
<b>Maximum</b>	12	12	12
<b>Mean (<math>\bar{x}</math>)</b>	9.38	10.13	10.29
<b>Median</b>	10.50	11.00	11.00
<b>Variance (<math>\sigma^2</math>)</b>	9.125	4.982	2.905
<b>Std. Deviation (<math>\sigma</math>)</b>	3.021	2.232	1.704
<b>Std. Error (<math>SE_{\bar{x}}</math>)</b>	1.068	.789	.644
<b>Skewness</b>	-1.515	-1.029	-.618
<b>Kurtosis</b>	2.279	-.069	-1.396

are equal for both groups (7.75), with Group A having a lesser median than Group B. Taking into account that the max score for Part 2 is 15, a mean of 7.75 suggests that the participants struggled more with these tasks compared to Part 1, as expected. Group B has a higher variance than Group A, due to a higher standard deviation. The standard error is similar for both groups, with Group B having a higher value. Skewness values still indicate a left-skewed distribution for both groups. However, taking into account both the skewness and kurtosis values, the distributions are approximately normally distributed. Having looked at the descriptive

Table 5.12: Descriptive statistics applied to the Part 2 score for Group A and Group B. The max score for Part 2 is 15.

<b>Part 2 Score</b>			
	<b>Group A</b>	<b>Group B</b>	<b>Group A (no outlier)</b>
<b>Minimum</b>	2	2	4
<b>Maximum</b>	12	12	12
<b>Mean (<math>\bar{x}</math>)</b>	7.75	7.75	8.57
<b>Median</b>	8.50	10.00	9.00
<b>Variance (<math>\sigma^2</math>)</b>	11.357	18.214	6.952
<b>Std. Deviation (<math>\sigma</math>)</b>	3.370	4.268	2.637
<b>Std. Error (<math>SE_{\bar{x}}</math>)</b>	1.191	1.509	.997
<b>Skewness</b>	-.638	-.579	-.570
<b>Kurtosis</b>	-.291	-2.097	.547

statistics for both groups from three different perspectives. We argue that Group B has performed better than Group A. However, the exclusion of Group A's outlier suggests that Group A performed better than Group B. It is important to note, however, that the differences are small, and we cannot conclude whether there is a significant difference between the groups only by comparing their descriptive statistics for either of the three perspectives. For this reason, we proceed to use statistical methods to answer our null hypothesis.

### 5.5.3 Hypothesis Testing

In our experiment, we applied two experimental conditions with different participants being used for each condition. For this reason, an appropriate hypothesis testing method is the independent t-test, also called an unpaired t-test [38, 106]. We will use the independent samples t-test function provided by IBM SPSS [56]. There are two variants of this t-test, one assuming equal variances, and one assuming unequal variances. To determine which variants to use, we will use Levene's test for equality of variances [71] provided by IBM SPSS [69]. When assessing the output of Levene's test and other hypothesis tests, we consider the *p-value*. If the Levene's test yields  $p \leq .05$ , it suggests that the variances are unequal. However if yields  $p > .05$ , we can assume that the variances are roughly equal [38].

After determining whether the variances are equal or not, we proceed to apply the appropriate independent t-test for each perspective given by [38, 75]:

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \quad (5.5)$$

Where  $\mu_1 - \mu_2$  is the difference between the population means for the two groups. If the null hypothesis is true, the samples have been drawn from the same population, therefore  $\mu_1 = \mu_2$  and we discard it from our equation [38]. When we carry out this calculation, we get what is called the t-statistic. From this t-statistic, we can find our critical value, using a t distribution table, along with a significance level  $\alpha$ , also called p-value and degrees of freedom [38, 69]. For our t-test, we will use a 95% confidence interval (*p-value* = 0.05) and degrees of freedom (*df*) given by:

$$df = n_1 + n_2 - 2 \quad (5.6)$$

where  $n_1$  and  $n_2$  are the sizes for each of our two control groups [38, 69].

We evaluate the null hypothesis established in Section 5.3.1:

$H_0$  Threat models with textual notation are equally comprehensible in comparison to threat models with graphical notation.

The results from the independent t-tests with respect to  $H_0$  are the following:

- **Total Score:** The assumption of homogeneity was assessed using Levene's test and the assumption of equal variances was accepted with  $F(14) = .613$  and  $p = .447$ . The independent samples t-test assuming equal variances yielded values  $t(14) = -.265$ ,  $p = .795$ . These values indicate that there is no statistically significant effect between Group A and Group B. Thus from this perspective, we accept the null hypothesis  $H_0$ .
- **Total Score, excluding the Group A outlier:** The assumption of homogeneity was assessed using Levene's test and the assumption



of equal variances was rejected with  $F(14) = 6.712$ . and  $p = .022$ . The independent samples t-test assuming unequal variances yielded values  $t(14) = .430$ ,  $p = .677$ . These values indicate that there is no statistically significant effect between Group A and Group B. Consequently, the total score without the outlier also accepts  $H_0$ .

- **Part 1 Score:** The assumption of homogeneity was assessed using Levene's test and the assumption of equal variances was accepted with  $F(14) = .358$  and  $p = .559$ . The independent samples t-test assuming equal variances yielded values  $t(14) = -.565$ ,  $p = .581$ . There is no statistically significant effect between Group A and Group B,  $H_0$  is accepted.
- **Part 1 Score, excluding the Group A outlier:** The assumption of homogeneity was assessed using Levene's test and the assumption of equal variances was accepted with  $F(14) = .613$ . and  $p = .447$ . The independent samples t-test assuming equal variances yielded values  $t(14) = -.265$ ,  $p = .795$ . Consequently, there is no statistically significant effect between Group A and Group B,  $H_0$  is accepted.
- **Part 2 Score:** The assumption of homogeneity was assessed using Levene's test and the assumption of equal variances was accepted with  $F(14) = 2.291$ . and  $p = .152$ . The independent samples t-test assuming equal variances yielded values  $t(14) = .000$ ,  $p = 1.000$ . These values are definite, there is no statistically significant effect between Group A and Group B,  $H_0$  is accepted.
- **Part 2 Score, excluding the Group A outlier:** The assumption of homogeneity was assessed using Levene's test and the assumption of equal variances was rejected with  $F(14) = 6.408$ . and  $p = .025$ . The independent samples t-test assuming unequal variances yielded values  $t(14) = .454$ ,  $p = .658$ . There is no significant statistically effect between Group A and Group B,  $H_0$  is accepted.

After performing the appropriate t-test to each perspective based on Levene's test for equal variances we conclude that all tests report acceptance of our null hypothesis. This means that, according to these results, the comprehensibility of threat models with either graphical or textual notation with respect to the given task set is equally comprehensible.

#### 5.5.4 Findings Related to Efficiency

The reported time spent per task are averages, as individual time was not an available feature in the Eval&Go survey tool. For this reason, the time measurement is prone to outliers that cannot be identified. Moreover, we cannot perform t-tests to compare the means, as we cannot calculate the standard deviation. The average time spent on each task and the task set in total can be seen from Table 5.13. We examine the average time for each group and note that Group B spent considerably more time than Group A for the whole task set. Furthermore, most of the reported differences

Table 5.13: Average time for the task set.  $\bar{x}(t)$  is the average time for either Group A or Group B in seconds,  $\Delta t = t_B - t_A$ . Furthermore, positive/negative values for  $\Delta t$  and % indicate that Group B spent more/less time than Group A.

Task #	$\bar{x}(t_A)$	$\bar{x}(t_B)$	$\Delta t$	%
1	22	31	9	40.91%
2	22	24	2	9.09%
3	13	21	8	61.54%
4	49	46	-3	-6.12%
5	36	41	5	13.89%
6	44	51	7	15.91%
7	119	145	26	21.85%
8+9	156	233	77	49.36%
10+11	167	205	38	22.75%
12+13	232	232	0	0.00%
<b>Total</b>	<b>860</b>	<b>1029</b>	<b>169</b>	

seen from the fourth column in the table are in favour of Group A. On average, Group B spent 22.92% more time than Group A on a task. Keeping in mind that our hypothesis tests strengthened the hypothesis that textual and graphical notation are equally comprehensible. These results related to time spent per task imply that using a graphical representation aids the participant in efficient task solving. This claim is further substantiated by Moody [80]. Moody argues that visual representations are more effective than textual because they are processed in parallel by the visual system, while textual representations are processed serially by the auditory system [80]. This is because textual representations are one-dimensional (linear), while graphical are two-dimensional (spatial) [80]. In other words, visual representations are more cognitive effective than textual. Cognitive effectiveness is defined as the "speed, ease and accuracy with which a representation can be processed by the human mind" [80].

However, it is important to note, since we lack individual time, we cannot ascertain that there were participants who contributed heavily to the average time statistic. This statistic could, for example, be affected by a participant either having skipped many questions or spent all/most of the available time for a task. However, we note that there is a difference in time and that having a more precise measurement of individual time may be of interest in future experiments to further answer the question related to participants' efficiency.

### 5.5.5 Findings from the Post-Experiment Questionnaire

From the Likert values obtained from the post-experiment questionnaire, we have found the overall satisfaction from the participants for each group by using mode for all the values. The overall answers are shown in Table 5.14. We note that the majority of both groups felt they did not have

Table 5.14: Post-experiment questionnaire answers

Question	Group A	Group B
I had enough time to solve the given tasks.	Strongly Disagree	Disagree
The objectives of the empirical experiment were clear to me.	Agree	Not Certain
The task descriptions were clear, and I understood what to do.	Agree	Agree
I experienced no difficulties in understanding the threat models.	Disagree	Not Certain
I experienced no difficulties in answering questions in Part 1.	Agree	Agree
I experienced no difficulties in answering questions in Part 2.	Not Certain	Disagree
I think the presentation given to me before the questionnaire provided me with enough information to solve the tasks.	Agree	Disagree

enough time to solve the tasks, with Group A strongly disagreeing with the statement. The majority participants in Group A state that they understood the objectives of the study, while the majority of Group B is uncertain. Both Groups agree that the task descriptions were clear, and they knew what to do. Moreover, Group A state they had difficulty understanding the threat models, while Group B are not certain. This might be due to the fact that the presentation did not focus heavily on the different applications of the CORAL approach, but mostly the notation required to solve the tasks. Both groups agree that they experienced no difficulties in answering the questions in Part 1. For Part 2 however, the majority of Group A are not certain whether they experienced difficulty. The majority of Group B, on the other hand, perceived the tasks in Part 2 as difficult. Finally, Group A state that the presentation provided enough information to solve the tasks. Group B however, disagrees. This is interesting, as both groups received the same presentation, although with different notation. This may be related to the fact that information represented visually are more likely to be remembered, due to the picture superiority effect [45, 72] as cited by Moody [80].

### 5.5.6 Threats to Validity

In this subsection, we discuss validity concerns that may threaten our controlled experiment, namely the threats to validity. In our experiment, we drew conclusions about our hypothesis by means of observation. Furthermore, these observations may strengthen or weaken our hypothesis. When drawing conclusions there are generally four steps involved, each for which there is a threat to the validity of the results [129, 130]. These can be seen from Figure 5.12 which show the basic principles of an experiment.

The four categories of threats are:

1. *Conclusion validity*: concerned with the possibility to draw correct conclusions about the relationship between the treatments and the outcome. E.g. whether the statistical power of the hypothesis test is sound, and questions regarding the reliability of the measurements.
2. *Internal validity*: ensuring that the observed outcome is a result of the given treatment, and not caused by an uncontrolled factor without the knowledge of the researcher. These factors were previously referred to as confounding factors in Section 5.3.2
3. *Construct validity*: related to the relationship between theory and observation. E.g. we have to make sure that concepts are defined clearly before measurements are defined.
4. *External validity*: related to the ability to generalise the results. In other words, whether the problems the participants have been working on are representative and whether the participants are representative of the target population.

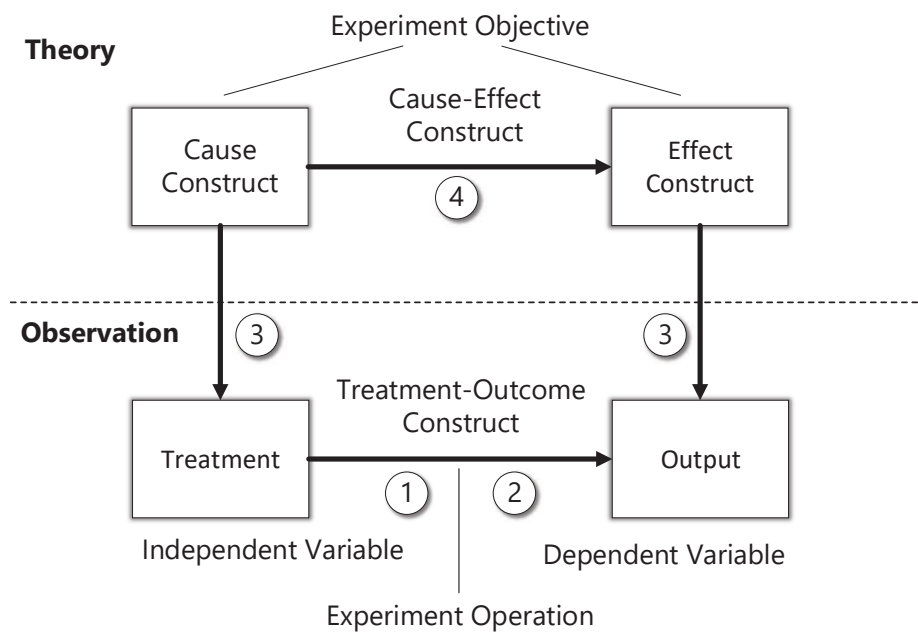


Figure 5.12: Experiment principles adapted from [122, 130].

### Conclusion Validity

For our hypothesis test, we used the parametric independent samples t-test, which assumes a normal distribution and independent control groups. This choice of method was motivated by our findings during the data

visualisation and use of descriptive statistics. In addition, the two control groups were completely independent, and each group were only subject to a single treatment. If the data was not normally distributed, we could have performed a non-parametric test such as the Mann-Whitney u-test [69]. Since Mann-Whitney does not assume a normal distribution. Although parametric tests if applicable, generally have higher power than non-parametric test, i.e. less data is needed to get significant results [129]. The t-tests were carried out from multiple perspectives in an attempt to mitigate arriving at a false conclusion when rejecting or accepting our null hypothesis. Moreover, we acknowledge that by having a bigger sample size, our conclusions would be more robust.

With respect to the findings related to efficiency, the time measurements were reported as averages. As mentioned in Section 5.5.4 we were unable to perform a t-test. For that reason, we cannot argue with the same statistical strength as we did for the hypothesis related to comprehensibility.

### **Internal Validity**

With respect to internal validity, a threat is introduced by not having randomised assignment of the treatment for our control groups. This is justified by trying to divide each group such that they have similar competence levels, in terms of knowledge, education level and working experience. However, the measurement of knowledge based on the Likert scale can be imprecise. Imprecision may be due to the Dunning–Kruger effect [21] as mentioned by Labunets et al. [67]. This is an effect wherein less competent people tend to overestimate their skills and knowledge, while more competent people tend to underestimate their skills and knowledge. Another threat to internal validity concerns the introductory material since the participants have to go through it on their own. As a consequence, we cannot control the degree to which the participant learns the given material. This uncertainty leads to two different situations. In which a participant either spends more or less time learning the material than others. We face another threat to internal validity with regard to participants' fatigue, e.g. if a participant decides to answer the questionnaire late in the evening or after work. Furthermore, we take note of the possibility of information exchange between participants. This was mitigated as the subjects were invited without knowledge of the other participants in the experiment. Finally, since we could not control the environment in which the participant answered the questionnaire, there was no way to ensure that the participant did not carry out internet searches to look for clues. In an attempt to mitigate this, the tasks had timers enforcing time restrictions.

### **Construct Validity**

A threat to construct validity is introduced by the theoretical constructs comprehensibility and efficiency by the manner in which they were measured. These constructs were measured by only one measurement type

each. These being task scores and average time. This threat is partially mitigated due to using measurement types similar to other studies [55, 79, 83]. The method used to measure these constructs were based on information-retrieval using questionnaires with closed-ended and open-ended questions, this method has also been used in similar empirical studies. Furthermore, all the experiment material was the same for both groups with the only difference being the notation to prevent bias. The introductory presentation introduced simple concepts in order to solve the given tasks. To ensure simplicity and reduce bias, the material was reviewed by other researchers as mentioned in Section 5.3.4. The threat models in the tasks exemplified well-known web application attacks, and correspond to realistic, although simplified situation. As mentioned in Section 5.5.5, the majority of participants understood the task descriptions, but faced difficulties understanding the threat models. Finally, another threat is introduced by enforcing time restrictions as this can cause time pressure. The time restrictions were evaluated through pre-studies. However, the majority of participants stated they did not have enough time to solve the tasks. It could have been beneficial to split the time-related post-experiment question into Part 1 and Part 2, due to the difference in difficulty to further pinpoint which tasks required more time.

### **External Validity**

Our sample of students of both undergraduates and graduates are not completely representative of our target population which are professionals within security testing. Who ultimately are the stakeholders likely to use the CORAL tool. The focus of the study however, was concerned with the comprehensibility and efficiency when interpreting predefined threat models with different notations. Thus, the study was not concerned with testing, and our sample was not of security testing professionals. The sample however, consists of developers at different levels, which is also relevant. Some of which were currently working or had been working for prolonged period. It can be argued, that developers are most familiar with the textual notation used in programming languages. Yet, all participants stated they had experience in using UML in some form. Furthermore, the experiment does not reflect a working environment and that the tasks are of a simplistic nature introduces a threat. The tasks are simplistic, but realistic with well-known web application attacks.

### **5.5.7 Analysis Summary**

To summarise, an empirical study was conducted by the means of a controlled experiment. Moreover, we formulated research questions along with null hypothesis and alternative hypotheses in case there was a statistically significant difference between the observations of applying different treatments. We consider the research questions:

**RQ1:** Will the use of a DSML using either textual or graphical notation to represent threat models affect the objective performance of compre-

hensibility? That is, is there a measurable difference with respect to effectiveness between the use of the two notations.

**RQ2:** Will the use of a DSML using either textual or graphical notation to represent threat models affect the participants' efficiency in solving the provided tasks?

In our analysis, the data was visualised, represented by descriptive statistics and finally tested using t-tests. All t-tests accepted our null hypothesis and which further indicates that CORAL with textual notation is equally comprehensible in comparison to CORAL with graphical notation. Textual notation being the UML stereotype annotations, and graphical being the icons introduced in the CORAL modelling language. Furthermore, this answers RQ1. The models we used in the experiment are documented in Appendix C (Group A – graphical notation) and Appendix D (Group B – textual notation).

With respect to RQ2, we compared the average time spent for each task, which indicates that there is a noticeable difference. Participants using graphical notation spent considerably less time than those using textual. As mentioned in Section 5.5.4, by measuring individual time we could have benefited from a t-test to further strengthen this claim. Finally, the post-experiment questionnaire provided useful information about the experiment design that can be used to improve future experiments.

Having discussed the threats to validity and research questions established for our empirical study, we proceed to discuss whether our artefact, the CORAL tool, fulfils the success criteria in the chapter that follows.





## Chapter 6

# Discussion

In Chapter 3 we presented the technology research method which is applied in this thesis. As part of the technology research method, when an artefact has been developed, the researcher must discuss to what end the artefact satisfies the requirements established during the problem analysis. In the context of our thesis project, we revisit our success criteria established in Chapter 2 on page 15. Our aim is to discuss whether our thesis work has fulfilled the success criteria, and to what extent. We established three success criteria for our thesis work, the development of the CORAL tool. In the following, we discuss each success criterion individually.

### 6.1 Success Criterion 1.

The first success criterion states: *The tool should support the creation of security tests based on the available risk picture*

As described in Section 2.1.5, the CORAL approach consists of a method that is supported by a domain specific modelling language to support risk-driven security testing. In order to support the creation security tests based on the available risk picture using the CORAL approach, one has to first apply the risk analysis language to assess the risks the system is exposed to. This activity corresponds to the first four steps of the CORAL method, in which the security tester creates threat models as a basis for the creation of security tests.

The CORAL tool supports the creation of threat models, by providing all the CORAL constructs required to capture risk according to the available risk picture. The CORAL tool has five different constructs to represent actors in a system or its environment, each represented as a lifeline with a rectangle annotated with corresponding stereotype names. To represent the different components/parts of a system, one can use the general lifeline, equivalent to a UML lifeline. The tool further has three kinds of lifelines representing threats, deliberate threat, accidental threat and non-human threat, represented by a rectangle annotated with «DeliberateThreat», «AccidentalThreat» and «NonHumanThreat» respectively. Finally, the

tool provides a lifeline to represent something of value to a party, e.g. confidentiality of client information, availability of financial records and so on. This is represented by the asset lifeline, represented by a rectangle annotated with «Asset».

To represent interactions, the tool provides the CORAL messages, which are stereotypes of the asynchronous UML message. All the messages are represented by an arrow with an open arrow head, annotated with the corresponding stereotype name. To represent expected behaviour, one can use the general message, which is equivalent to the asynchronous UML message. To represent behaviour that deviates from expected behaviour, one can use the altered message, represented by a message annotated with «AlteredMessage». To represent behaviour introduced by a threat one can use the new message, represented by a message annotated with «NewMessage». To represent behaviour that has been deleted due to interaction from a threat, one can use the deleted message, represented by a message annotated with «DeletedMessage». Finally, unwanted incidents, i.e. events that may harm assets are represented by a message annotated with «UnwantedIncident». In addition to specifying behaviour using messages, one can use the interaction operators to specify logic that describes the interactions further. These include: alt, ref, par and loop.

Further, the security tester can annotate the messages with risk information using the risk-measure annotations. These are implemented in the tool as properties of messages, but can also be visually represented in the diagram by the use of UML comment stereotypes. We provide the following stereotypes of comment: frequency, likelihood, consequence and conditional ratio. In Figure H.1 in Appendix H one can see a CORAL threat model annotated with these risk-measure annotations.

After creating threat models using the CORAL constructs, the security tester can validate the model according to CORAL constraints. Since the tool implements model validation, to ensure that the models that security tests are based upon are syntactically and semantically valid in both CORAL and UML. This further supports the creation of security tests, by ensuring the validity of the threat models.

Based on the above discussion, we see that the tool is fully capable of supporting the CORAL approach and that it supports the creation of security tests based on the available risk picture. As pointed out in Section 4.5.1 and Section 4.7, the tool does not currently support the graphical icons of the CORAL language due to the technical difficulties met during the development of the tool. As already pointed out, these technical difficulties were out of our control, and it was not possible to resolve these issues during the time of this thesis. However, we do not see the lack of graphical icons as a major disadvantage in the tool because, as shown by our empirical evaluation, there is no significant difference with respect to the comprehensibility of CORAL diagrams annotated with stereotypes versus CORAL diagrams annotated with graphical icons.

## 6.2 Success Criterion 2.

The second success criterion states: *The tool must sufficiently aid security testers in selecting and designing security tests with the help of security risk assessment.*

In terms of selecting security tests, ideally one would like to design tests to address the most severe risks. For this purpose, the CORAL tool implements the risk-measure annotations of the CORAL risk analysis language as properties of messages. This, in turn, allows for security testers to input risk information for each message part of a threat scenario. When the threat models of interest have been created, and their threat scenarios annotated with risk information, one can select which risks to test according to their risk levels. The frequency scale, consequence scale and risk evaluation matrix are not provided by the tool. Consequently, they need to be manually filled out and calculated on a sheet, e.g. Excel. It is possible, however, to open an Excel sheet in a new tab inside the tool by right clicking the excel file and selecting open with system editor.

To aid the design of test cases, one can use the UML testing profile (UTP) provided by Papyrus to annotate threat models with stereotypes from the UTP.

## 6.3 Success Criterion 3.

The third success criterion states: *The tool must be appropriate and comprehensible for security testers.*

To the end of ensuring comprehensibility, an empirical study was carried out to assess whether the textual notation adopted by the tool was equally comprehensible in comparison to the graphical notation in the CORAL approach, with respect to the interpretation of threat models. The study was not conducted with security testers specifically as participants, however, the test material was abstracted to such a level that the target group with a background in computer science and software development could participate. Participants were both undergrad and graduate students, some of whom possessed work experience in software development.

The collected data were analysed from six different perspectives all of which accepted the null hypothesis with a 95% confidence interval. Due to this, our study concludes that the textual notation is equally comprehensible to that of graphical notation. Further, the findings related to efficiency report that the participants which received the graphical notation treatment were more efficient in solving the tasks, spending 22% less time on average. These findings are similar to the findings of Hogganvik and Stølen [54], who found that the participants using graphical notations were able to conclude faster, however, not reaching a higher correctness of interpreting the models, compared to those using textual notation. Their study involved both professionals and students.

Our findings indicate that the textual notation adopted by the tool

is comprehensible to the target group of our empirical study, and these findings may relate to security testers as well. However, these findings are limited to our experiment and further empirical investigations are needed to further test our hypothesis from Section 5.3.1.

As for the appropriateness of the tool, we can argue that it is appropriate for security testers by including the UTP, which is often used within the model-based testing community [4]. Since the CORAL tool implements the UTP, security testers can define test cases and test outcomes by defining test objectives, verdicts, the system under test, and the test-components that constitute the test environment. To this end, the CORAL tool is appropriate for security testers.

To evaluate the claim of appropriateness with respect to the usability of the tool, further empirical evidence is needed, e.g. by the means of a usability study of the CORAL tool.

## Chapter 7

# Conclusion

In our introduction, we established that security testing is a necessity in a world where every day, software is exposed to new threats and attacks. Further, there is a need to efficiently and accurately carry out risk-driven security testing to reduce costs in terms of time and money. Efficiently in the sense that risk-driven security testing approach should be carried out in a reasonable amount of time, and accurately in the sense that the security testing address the most severe risks.

There are a number of different risk-driven security testing approaches that have been proposed to meet these challenges [34], however in this thesis, our focus was specifically on the CORAL approach [33]. Moreover, we pointed out that by applying the CORAL approach, one can systematically carry out risk-driven security testing, to more accurately target security testing at the most severe risks, as well as reduce the time needed to carry our security testing. However, to fully meet this, there is a need for dedicated tool support for the CORAL approach.

Thus, in this thesis, we propose a tool to support the approach by providing the following contributions:

- Adaptation of CORAL as a UML Profile.
- A CORAL plug-in for Eclipse Papyrus to provide tool support for risk-driven security testing.
- An empirical study to assess the comprehensibility of the textual notation adopted by the tool in comparison to the graphical notation introduced in the CORAL risk analysis language.

The CORAL tool consists of a UML profile that includes all the required CORAL constructs to capture risk according to the available risk picture as discussed in the previous chapter. These constructs are visually represented in CORAL threat models by a textual notation, i.e. the UML stereotype annotations. Threat models created in the CORAL tool supports the creation of security tests. In addition, the tool implements model validation to ensure that security tests are based on syntactically and semantically valid CORAL and UML models.

Furthermore, the tool includes the UML testing profile provided by Eclipse Papyrus, thus ensuring appropriateness for security testers and

aids them in the selection and design of security tests based on security risk assessment. The tool has been developed as a plug-in for the Eclipse Papyrus tool, and can be download from the Bitbucket repository:

<https://bitbucket.org/vetlevo/no.uio.ifi.coral.profile/>.

We conducted an empirical study to investigate whether the use of textual notation to represent CORAL threat models would affect the comprehensibility in comparison to the CORAL graphical notation. The empirical study was carried out by means of a controlled experiment with 16 participants. The gathered data were analysed from six different perspectives, and from all perspectives, the following hypothesis established in Section 5.3.1 was accepted:

$H_0$  Threat models with textual notation are equally comprehensible in comparison to threat models with graphical notation.

The acceptance of our hypothesis implies that textual notation is equally comprehensible to graphical notation. As a result of this, we argue that the threat models produced by the CORAL tool are comprehensible for security testers.

The empirical study points out that the use of graphical notation is more efficient in comparison to textual notation. With the participants receiving graphical notation spending on average 22% less time for each question than those receiving textual. These findings are substantiated by similar studies of Hogganvik and Stølen [54], who also reported equal comprehensibility with respect to textual and graphical notation, yet, the participants subjected to graphical notation reported as being more efficient in solving the tasks. Further, with respect to efficiency, Moody [80] argues that graphical notations are more cognitive effective than textual.

In the following section, we discuss directions for future work related to the CORAL tool.

## 7.1 Directions for Future Work

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

- Empirical investigations to determine the usability of the CORAL tool can be carried out. A usability study might uncover features that are not properly expressed to the intended user, or absence of behaviour required to fulfil the tool's potential. This can provide valuable insight toward how the tool can be refined to suit the needs of security testers.
- Implement a specification for the frequency and consequence scales, accompanied by an editor to input the scale values. Furthermore, we could provide a specification of an evaluation matrix whose values are derived/calculated based on the values defined in the frequency and consequence scales by the security tester.

- Implement a feature in which the values of the risk-measure annotations that extend the UML Comment class are derived based on the risk-measures defined as property values for their respective messages.
- Implement a path prioritisation algorithm to prioritise which paths to test based on a suspension criteria. We refer to a sequence of threat scenarios leading up to and including a risk as a path [35], while the suspension criteria serve as a threshold for the risk values we want to include in the security testing.
- Implement a feature to ease the communication of CORAL threat models between security testers, by providing automatically generated natural-language semantics in English prose for threat models as defined by the CORAL approach [33].
- Integrate JUnit [64] to automatically generate Java test cases based on the test cases defined by using the UTP.





# Acronyms

API	application programming interface p. 26
CSS	cascading style sheets pp. 41, 44
DSML	domain specific modelling language pp. 28–30, 39, 43, 44, 48, 49, 81
EBNF	extended Backus–Naur form pp. 11, 31, 35
EMF	Eclipse modelling framework pp. 5, 27–30, 32, 40
EPL	Eclipse public licence pp. 8, 27
ER	entity-relationship p. 47
GEF	graphical editing framework pp. 5, 28–30
GMF	graphical modelling framework pp. 5, 28, 29, 41
GPL	GNU general public licence p. 8
GQM	goal, question, metric p. 47
IDE	integrated development environment p. 26
MARTE	modelling and analysis of real time and embedded systems pp. 28, 30
MBT	model-based testing pp. 1, 8, 9, 84
MDE	model-driven engineering pp. 39, 46, 47
MDT	model development tools p. 28
OCL	object constraint language pp. 4, 5, 29, 39–41
OMG	object management group pp. 8, 28
QIP	quality improvement paradigm p. 45
RCP	rich client platform pp. 5, 25, 26, 28–30, 44
RST	risk-driven security testing pp. i, 2–4, 9–12, 43, 85
SDK	software development kit p. 26
SUT	system under test pp. 1, 2, 9, 11, 12, 15, 30, 84
SWT	standard widget toolkit pp. 27, 28
SysML	systems modelling language pp. 28, 30
UI	user interface pp. 25, 27
UML	unified modelling language pp. 3–5, 7–10, 15, 25, 27–32, 34, 35, 38–44, 47, 48, 62, 78, 79, 81, 82, 85, 86
URI	uniform resource identifier p. 43
UTP	UML testing profile pp. 4, 7–9, 12, 15, 22, 25, 30, 43, 83–85, 87
XMI	XML meta data interchange p. 43
XML	extensible markup language pp. 27, 42



# Bibliography

- [1] *ArgoUML*. <http://argouml.tigris.org/>. Accessed May 18, 2016.
- [2] C. W. Bachman. ‘Data Structure Diagrams’. In: *SIGMIS Database* 1.2 (1969), pp. 4–10.
- [3] J. Bailey, D. Budgen, M. Turner, B. Kitchenham, P. Brereton and S. Linkman. ‘Evidence relating to Object-Oriented software design: A survey’. In: *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. IEEE Computer Society, 2007, pp. 482–484.
- [4] P. Baker, Z. R. Dai, J. Grabowski, Ø. Haugen, I. Schieferdecker and C. Williams. *Model-driven testing: Using the UML testing profile*. Springer, 2007.
- [5] R. Balzer, F. Belz, R. Dewar, D. Fisher, R. Gabriel, J. Guttag, P. Hudak and M. Wand. ‘Prototyping’. In: *Annual Review of Computer Science* 4.1 (1990), pp. 453–465.
- [6] V. R. Basili, G. Caldiera and H. D. Rombach. ‘Experience Factory’. In: *Encyclopedia of Software Engineering*. John Wiley & Sons, 2002.
- [7] G. Booch, J. Rumbaugh and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.
- [8] R. Bosak, R. F. Clippinger, C. Dobbs, R. Goldfinger, R. B. Jasper, W. Keating, G. Kendrick and J. E. Sammet. ‘An Information Algebra: Phase 1 Report—Language Structure Group of the CODASYL Development Committee’. In: *Communications of the ACM* 5.4 (1962), pp. 190–204.
- [9] J. Botella, B. Legeard, F. Peureux and A. Vernotte. ‘Risk-Based Vulnerability Testing Using Security Test Patterns’. In: *Proc. 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA’14)*. Springer, 2014, pp. 337–352.
- [10] J. Cabot and M. Gogolla. ‘Object constraint language (OCL): a definitive guide’. In: *Formal methods for model-driven engineering*. Springer, 2012, pp. 58–90.
- [11] P. P. Chen. ‘The Entity-relationship Model—Toward a Unified View of Data’. In: *ACM Transactions on Database Systems* 1.1 (1976), pp. 9–36.

- [12] *Cisco 2017 Annual Security Report*. <http://b2me.cisco.com/en-us-annual-cybersecurity-report-2017?keycode1=001464153>. Accessed July 21, 2017.
- [13] E. F. Codd. *Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks*. Research Report RJ599. IBM, 1969.
- [14] *Combined fragments in sequence diagrams*. [https://www.ibm.com/support/knowledgecenter/en/SS4JE2\\_7.5.5/com.ibm.xtools.sequence.doc/topics/ccombfrag\\_v.html](https://www.ibm.com/support/knowledgecenter/en/SS4JE2_7.5.5/com.ibm.xtools.sequence.doc/topics/ccombfrag_v.html). Accessed July 30, 2017.
- [15] R. Conradi and A. I. Wang. *Empirical methods and studies in software engineering: Experiences from ESERNET*. Springer, 2003.
- [16] *Data Protection Official for Research*. <http://www.nsd.uib.no/nsd/english/pvo.html>. Accessed June 24, 2017.
- [17] A. De Lucia, C. Gravino, R. Oliveto and G. Tortora. 'An experimental comparison of ER and UML class diagrams for data modelling'. In: *Empirical Software Engineering* 15.5 (2010), pp. 455–492.
- [18] A. C. Dias Neto, R. Subramanyan, M. Vieira and G. H. Travassos. 'A Survey on Model-based Testing Approaches: A Systematic Review'. In: *Proc. 1st International Workshop on Empirical Assessment of Software Engineering Languages and Technologies*. ACM, 2007, pp. 31–36.
- [19] *Definition of effectiveness*. <https://en.oxforddictionaries.com/definition/effectiveness>. Accessed June 03, 2017.
- [20] *Definition of research*. <http://www.merriam-webster.com/dictionary/research>. Accessed April 12, 2016.
- [21] D. Dunning, K. Johnson, J. Ehrlinger and J. Kruger. 'Why people fail to recognize their own incompetence'. In: *Current directions in psychological science* 12.3 (2003), pp. 83–87.
- [22] *Eclipse E4*. <http://wiki.eclipse.org/E4>. Accessed July 13, 2017.
- [23] *Eclipse EMF*. <https://projects.eclipse.org/projects/modeling.emf.emf>. Accessed July 22, 2017.
- [24] *GEF/Developer FAQ*. [http://wiki.eclipse.org/GEF/Developer\\_FAQ](http://wiki.eclipse.org/GEF/Developer_FAQ). Accessed July 13, 2017.
- [25] *GEF (Graphical Editing Framework)*. <https://www.eclipse.org/gef/>. Accessed July 13, 2017.
- [26] *Eclipse Orion*. <http://wiki.eclipse.org/Orion>. Accessed July 13, 2017.
- [27] *The Eclipse Platform*. <http://wiki.eclipse.org/Platform>. Accessed July 13, 2017.
- [28] *The Eclipse RCP*. [https://wiki.eclipse.org/Rich\\_Client\\_Platform](https://wiki.eclipse.org/Rich_Client_Platform). Accessed July 11, 2017.
- [29] *Eclipse Modelling Project*. <https://projects.eclipse.org/projects/modeling>. Accessed July 14, 2017.
- [30] *Model Development Tools Project*. <https://projects.eclipse.org/projects/modeling.mdt>. Accessed July 14, 2017.

- [31] *EMF/FAQ*. <https://wiki.eclipse.org/EMF/FAQ>. Accessed July 20, 2017.
- [32] *Eclipse Public License - Version 1.0*. <https://www.eclipse.org/legal/epl-v10.html>. Accessed July 12, 2016.
- [33] G. Erdogan. ‘CORAL: A Model-Based Approach to Risk-Driven Security Testing’. PhD thesis. University of Oslo, 2016.
- [34] G. Erdogan, Y. Li, R.K. Runde, F. Seehusen and K. Stølen. ‘Approaches for the combined use of risk analysis and testing: a systematic literature review’. In: *International Journal on Software Tools for Technology Transfer* 16.5 (2014), pp. 627–642.
- [35] G. Erdogan, A. Refsdal and K. Stølen. ‘A Systematic Method for Risk-driven Test Case Design Using Annotated Sequence Diagrams’. In: *Proc. 1st International Workshop on Risk Assessment and Risk-driven Testing (RISK’13)*. Springer, 2014, pp. 93–108.
- [36] *Eval&Go*. <http://www.evalandgo.com/>. Accessed July 06, 2017.
- [37] B.S. Everitt and A. Skrondal. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010.
- [38] A. Field. *Discovering Statistics Using IBM SPSS Statistics*. SAGE Publications, 2013.
- [39] The World Economic Forum. *The Global Risks Report 2017 12th Edition*. Insight Report Ref: 050117. The World Economic Forum, 2017.
- [40] V. Garousi and J. Zhi. ‘A survey of software testing practices in Canada’. In: *Journal of Systems and Software* 86.5 (2013), pp. 1354–1376.
- [41] *Online Tutorial: Getting started with GEF 5.0*. <https://info.itemis.com/en/gef/tutorials/>. Accessed July 13, 2017.
- [42] *GEF Wiki*. <https://github.com/eclipse/gef/wiki>. Accessed July 13, 2017.
- [43] *Graphical Modeling Framework*. [http://wiki.eclipse.org/Graphical\\_Modeling\\_Framework](http://wiki.eclipse.org/Graphical_Modeling_Framework). Accessed July 13, 2017.
- [44] *Google Forms*. <https://www.google.com/forms/about/>. Accessed July 06, 2017.
- [45] P. Goolkasian. ‘Pictures, words, and sounds: From which format are we best able to reason?’ In: *Transactions on Software Engineering, IEEE* 127.4 (2000), pp. 439–459.
- [46] P. Goolkasian. ‘Software security’. In: *IEEE Security & Privacy* 2.2 (2004), pp. 80–83.
- [47] *The GNU General Public License v3.0 - GNU Project - Free Software Foundation*. <http://www.gnu.org/licenses/gpl-3.0.html>. Accessed May 18, 2016.

- [48] J. Großmann, M. Berger and J. Viehmann. ‘A Trace Management Platform for Risk-Based Security Testing’. In: *Proc. 1st International Workshop on Risk Assessment and Risk-driven Testing (RISK’13)*. Springer, 2014, pp. 120–135.
- [49] J. Großmann, M. Schneider, J. Viehmann and M.-F. Wendland. ‘Combining Risk Analysis and Security Testing’. In: *Proc. 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA’14)*. Springer, 2014, pp. 322–336.
- [50] F. E. Grubbs. ‘Procedures for detecting outlying observations in samples’. In: *Technometrics* 11.1 (1969), pp. 1–21.
- [51] I. Hadar, I. Reinhartz-Berger, T. Kuflik, A. Perini, F. Ricca and A. Susi. ‘Comparing the comprehensibility of requirements models expressed in Use Case and Tropos: Results from a family of experiments’. In: *Information and Software Technology* 55.10 (2013), pp. 1823–1843.
- [52] G. S. Halford, R. Baker, J. E. McCredde and J. D. Bain. ‘How many variables can humans process?’ In: *Psychological Science* 16.1 (2005), pp. 70–76.
- [53] G. S. Halford, W. H. Wilson and S. Phillips. ‘Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology’. In: *Behavioral and Brain Sciences* 21.6 (1998), pp. 803–831.
- [54] I. Hogganvik and K. Stølen. ‘A graphical approach to risk identification, motivated by empirical investigations’. In: *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2006, pp. 574–588.
- [55] I. Hogganvik and K. Stølen. *Empirical Investigations of the CORAS Language for Structured Brainstorming*. Technical Report A05041. SINTEF Information and Communication Technology, 2005.
- [56] IBM SPSS. <https://www.ibm.com/analytics/us/en/technology/spss/>. Accessed July 06, 2017.
- [57] IBM Rational Software. <https://www.ibm.com/software/uk/rational/>. Accessed July 23, 2017.
- [58] *Interaction operators in sequence diagrams*. [https://www.ibm.com/support/knowledgecenter/en/SS8PJ7\\_9.6.0/com.ibm.xtools.sequence.doc/topics/rinteracoperate.html](https://www.ibm.com/support/knowledgecenter/en/SS8PJ7_9.6.0/com.ibm.xtools.sequence.doc/topics/rinteracoperate.html). Accessed July 30, 2017.
- [59] *Indirectly Identifiable Personal Data*. <http://www.nsd.uib.no/personvernombud/en/help/vocabulary.html>. Accessed June 20, 2017.
- [60] *Interaction Flow Modeling Language, version 1.0*. OMG Document Number formal/2015-02-05. Object Management Group. 2015.
- [61] *ISO/IEC 14977:1996(E), Information technology – Syntactic metalanguage – Extended BNF, first edition*. International Organization for Standardization. 1996.

- [62] ISO/IEC/IEEE 29119-2:2013(E), *Software and system engineering - Software testing - Part 2: Test process*. International Organization for Standardization, 2013.
- [63] *JavaFX Overview*. <http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm{#}JFXST784>. Accessed July 13, 2017.
- [64] *JUnit*. <http://junit.org/junit4/>. Accessed July 31, 2017.
- [65] C. Kaner. *The Impossibility of Complete Testing*. Tech. rep. Accessed March 13, 2016. <http://www.kaner.com/pdfs/imposs.pdf>, 1997.
- [66] B. Kitchenham and S. Charters. *Guidelines for performing systematic literature reviews in software engineering*. Technical Report EBSE-2007-01. Keele University, and University of Durham, 2007.
- [67] K. Labunets, F. Massacci, F. Paci, S. Marczak and F. M. de Oliveira. 'Model comprehension for security risk assessment: an empirical comparison of tabular vs. graphical representations'. In: *Empirical Software Engineering* (2017), pp. 526–540.
- [68] A. Lamsweerde. 'Formal Specification: A Roadmap'. In: *Proc. Conference on The Future of Software Engineering*. ACM, 2000, pp. 147–159.
- [69] S. Landau and B.S. Everitt. *A Handbook of Statistical Analyses Using SPSS*. Taylor & Francis, 2004.
- [70] C.Y. Lester and F. Jamerson. 'Incorporating Software Security into an Undergraduate Software Engineering Course'. In: *Proc. 3rd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'09)*. IEEE Computer Society, 2009, pp. 161–166.
- [71] H. Levene. 'Robust Tests for Equality of Variances'. In: *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*. Springer, 1960, pp. 278–292.
- [72] W. Lidwell, J. Butler and K. Holden. *Universal principles of design: a cross disciplinary reference*. Rockport Publishers, 2003.
- [73] *Lime Survey*. <https://www.limesurvey.org/>. Accessed July 06, 2017.
- [74] M.S. Lund, B. Solhaug and K. Stølen. *Model-Driven Risk Analysis: The CORAS Approach*. Springer, 2011.
- [75] B.S. Madsen. *Statistics for Non-Statisticians*. Springer, 2016.
- [76] *MagicDraw*. <https://www.nomagic.com/products/magicdraw>. Accessed July 23, 2017.
- [77] R. McGill, J. W. Tukey and W. A. Larsen. 'Variations of box plots'. In: *The American Statistician* 32.1 (1978), pp. 12–16.
- [78] J.E. McGrath. *Groups: interaction and performance*. Prentice-Hall, 1984.
- [79] S. Meliá, C. Cachero, J. M. Hermida and E. Aparicio. 'Comparison of a textual versus a graphical notation for the maintainability of MDE domain models: an empirical pilot study'. In: *Software Quality Journal* 24.3 (2016), pp. 709–735.

- [80] D.L. Moody. ‘The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering’. In: *Transactions on Software Engineering, IEEE* 35.6 (2009), pp. 756–779.
- [81] K.K. Murthy, K.R. Thakkar and S. Laxminarayan. ‘Leveraging risk based testing in enterprise systems security validation’. In: *Proc. 1st International Conference on Emerging Network Intelligence (EMERGING’09)*. IEEE Computer Society, 2009, pp. 111–116.
- [82] *NetBeans Platform Learning Trail*. <https://netbeans.org/kb/trails/platform.html>. Accessed July 11, 2017.
- [83] E. G. Nilsson and K. Stølen. *The FLUIDE Framework for Specifying Emergency Response User Interfaces Employed to a Search and Rescue Case*. Technical Report A27575. SINTEF Information and Communication Technology, 2016.
- [84] *Norwegian Centre for Research Data*. <http://www.nsd.uib.no/nsd/english>. Accessed July 2, 2017.
- [85] *Online Surveys*. [http://www.nsd.uib.no/personvernombud/en/help/research\\_methods/online\\_surveys.html](http://www.nsd.uib.no/personvernombud/en/help/research_methods/online_surveys.html). Accessed June 24, 2017.
- [86] *Object Constraint Language, version 2.4*. OMG Document Number: formal/2014-02-03. Object Management Group. 2014.
- [87] *OMG Systems Modeling Language, version 1.5*. OMG Document Number formal/2017-05-01. Object Management Group. 2017.
- [88] *OMG Unified Modeling Language, version 2.5*. OMG Document Number formal/2015-03-01. Object Management Group. 2015.
- [89] *OMG Formal Specifications*. <http://www.omg.org/spec/>. Accessed May 18, 2016.
- [90] *Papyrus/Migration Guide/Oxygen*. [https://wiki.eclipse.org/Papyrus/Migration\\_Guide/Oxygen](https://wiki.eclipse.org/Papyrus/Migration_Guide/Oxygen). Accessed July 15, 2017.
- [91] *Papyrus Modeling Environment*. <https://eclipse.org/papyrus/>. Accessed May 18, 2016.
- [92] *Papyrus/Photon Work Description*. [https://wiki.eclipse.org/Papyrus/Photon\\_Work\\_Description](https://wiki.eclipse.org/Papyrus/Photon_Work_Description). Accessed July 18, 2017.
- [93] *Papyrus Project*. <https://projects.eclipse.org/projects/modeling.mdt.papyrus>. Accessed July 14, 2017.
- [94] *Papyrus Wiki*. <https://wiki.eclipse.org/Papyrus>. Accessed July 14, 2017.
- [95] *The Eclipse Plug-in Development Environment*. <http://wiki.eclipse.org/PDE>. Accessed July 13, 2017.
- [96] *The Global State of Information Security Survey 2017*. <https://www.pwc.com/gx/en/issues/cyber-security/information-security-survey.html>. Accessed July 21, 2017.
- [97] *Qualtrics*. <https://www.qualtrics.com/>. Accessed July 06, 2017.
- [98] *QuestionPro*. <https://www.questionpro.com/>. Accessed July 06, 2017.



- [99] T. Sandelin and M. Vierimaa. 'Empirical studies in esernet'. In: *Empirical Methods and Studies in Software Engineering*. Springer, 2003, pp. 39–54.
- [100] C. Schalles. *Usability evaluation of modeling languages*. Springer, 2012.
- [101] F. Seehusen. 'A Technique for Risk-Based Test Procedure Identification, Prioritization and Selection'. In: *Proc. 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'14)*. Springer, 2014, pp. 277–291.
- [102] M. E. Senko, E. B. Altman, M. M. Astrahan and P. L. Fehder. 'Data structures and accessing in data-base systems, I: Evolution of information systems'. In: *IBM Systems Journal* 12.1 (1973), pp. 30–44.
- [103] *Service oriented architecture Modeling Language (SoaML) Specification, version 1.0.1*. OMG Document Number formal/2012-05-10. Object Management Group. 2012.
- [104] W. R. Shadish, T. D. Cook and D. T. Campbell. *Experimental and quasi-experimental designs for generalized causal inference*. Wadsworth Cengage learning, 2002.
- [105] F. Shull, J. Singer and D.I.K. Sjøberg. *Guide to Advanced Empirical Software Engineering*. Springer, 2007.
- [106] K. Singh. *Quantitative Social Research Methods*. SAGE Publications, 2007.
- [107] D. I. K Sjøberg, J. E. Hannay, O. Hansen, V. By Kampenes, A. Karahasanovic, N. K. Liborg and A. C. Rekdal. 'A survey of controlled experiments in software engineering'. In: *IEEE transactions on software engineering* 31.9 (2005), pp. 733–753.
- [108] B. Solhaug and K. Stølen. 'The CORAS Language - Why it is designed the way it is'. In: *Proc. 11th International Conference on Structural Safety and Reliability (ICOSSAR'13)*. Taylor and Francis, 2013, pp. 3155–3162.
- [109] I. Solheim and K. Stølen. *Technology research explained*. Technical Report A313. SINTEF Information and Communication Technology, 2007.
- [110] R. van Solingen, V. Basili, G. Caldiera and H. D. Rombach. 'Goal Question Metric (GQM) Approach'. In: *Encyclopedia of Software Engineering*. John Wiley & Sons, 2002.
- [111] I. Sommerville. *Software Engineering*. Pearson, 2011.
- [112] D. Steinberg, F. Budinsky, E. Merks and M. Paternostro. *EMF: Eclipse Modeling Framework*. Pearson Education, 2008.
- [113] G. Stoneburner, A. Y. Goguen and A. Feringa. *SP 800-30. Risk Management Guide for Information Technology Systems*. Technical Report. National Institute of Standards & Technology, 2002.
- [114] *SurveyGizmo*. <https://www.surveygizmo.com/>. Accessed July 06, 2017.

- [115] *SurveyMonkey*. <https://www.surveymonkey.com/>. Accessed July 06, 2017.
- [116] *Survey Planet*. <https://surveyplanet.com/>. Accessed July 06, 2017.
- [117] *SWT: The Standard Widget Toolkit*. <https://www.eclipse.org/swt/>. Accessed July 13, 2017.
- [118] *Symantec Internet Security Threat Report, Volume 22, April 2017*. <https://www.symantec.com/security-center/threat-report>. Accessed July 22, 2017.
- [119] *About the Eclipse Foundation*. <http://www.eclipse.org/org/>. Accessed July 12, 2017.
- [120] *The Eclipse Java Development Tools*. <http://wiki.eclipse.org/JDT>. Accessed July 13, 2017.
- [121] *The Eclipse Project*. [http://wiki.eclipse.org/Eclipse\\_Project](http://wiki.eclipse.org/Eclipse_Project). Accessed July 13, 2017.
- [122] W.M.K Trochim. *The Research Methods Knowledge Base, 2nd edn*. Cornell Custom Publishing, 1999.
- [123] *UML Testing Profile (UTP), version 1.2*. OMG Document Number: formal/2013-04-03. Object Management Group. 2013.
- [124] *MDT/UML2*. <https://wiki.eclipse.org/MDT/UML2>. Accessed July 19, 2017.
- [125] *UMLet*. <http://www.umlet.com/>. Accessed May 18, 2016.
- [126] M. Utting, A. Pretschner and B. Legeard. 'A taxonomy of model-based testing approaches'. In: *Software Testing, Verification and Reliability 22.5* (2012), pp. 297–312.
- [127] M. Utting, A. Pretschner and B. Legeard. 'A taxonomy of model-based testing approaches'. In: *Software Testing, Verification and Reliability 22.5* (2012), pp. 297–312.
- [128] R. J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014.
- [129] C. Wohlin, M. Höst and K. Henningsson. 'Empirical research methods in software engineering'. In: *Empirical Methods and Studies in Software Engineering*. Springer, 2003, pp. 7–23.
- [130] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell and A. Wesslén. *Experimentation in Software Engineering*. Springer, 2012.
- [131] R. E. Wood. 'Task complexity: Definition of the construct'. In: *Organizational behavior and human decision processes 37.1* (1986), pp. 60–82.
- [132] D. Xu, M. Tu, M. Sandford, L. Thomas, D. Woodraska and W. Xu. 'Automated Security Test Generation with Formal Threat Models'. In: *IEEE Transactions on Dependable and Secure Computing 9.4* (2012), pp. 526–540.

- [133] J. W. Young and H. K. Kent. 'An abstract formulation of data processing problems'. In: *Proc. Preprints of papers presented at the 13th national meeting of the Association for Computing Machinery*. ACM, 1958, pp. 1–4.
- [134] P. Zech. 'Risk-Based Security Testing in Cloud Computing Environments'. In: *Proc. 4th International Conference on Software Testing*. IEEE Computer Society, 2011, pp. 411–414.
- [135] P. Zech, M. Felderer and R. Breu. 'Towards a Model Based Security Testing Approach of Cloud Computing Environments'. In: *Proc. 6th International Conference on Software Security and Reliability Companion (SERE-C'12)*. IEEE Computer Society, 2012, pp. 47–56.
- [136] *Zoho survey*. <https://www.zoho.eu/survey/>. Accessed July 06, 2017.



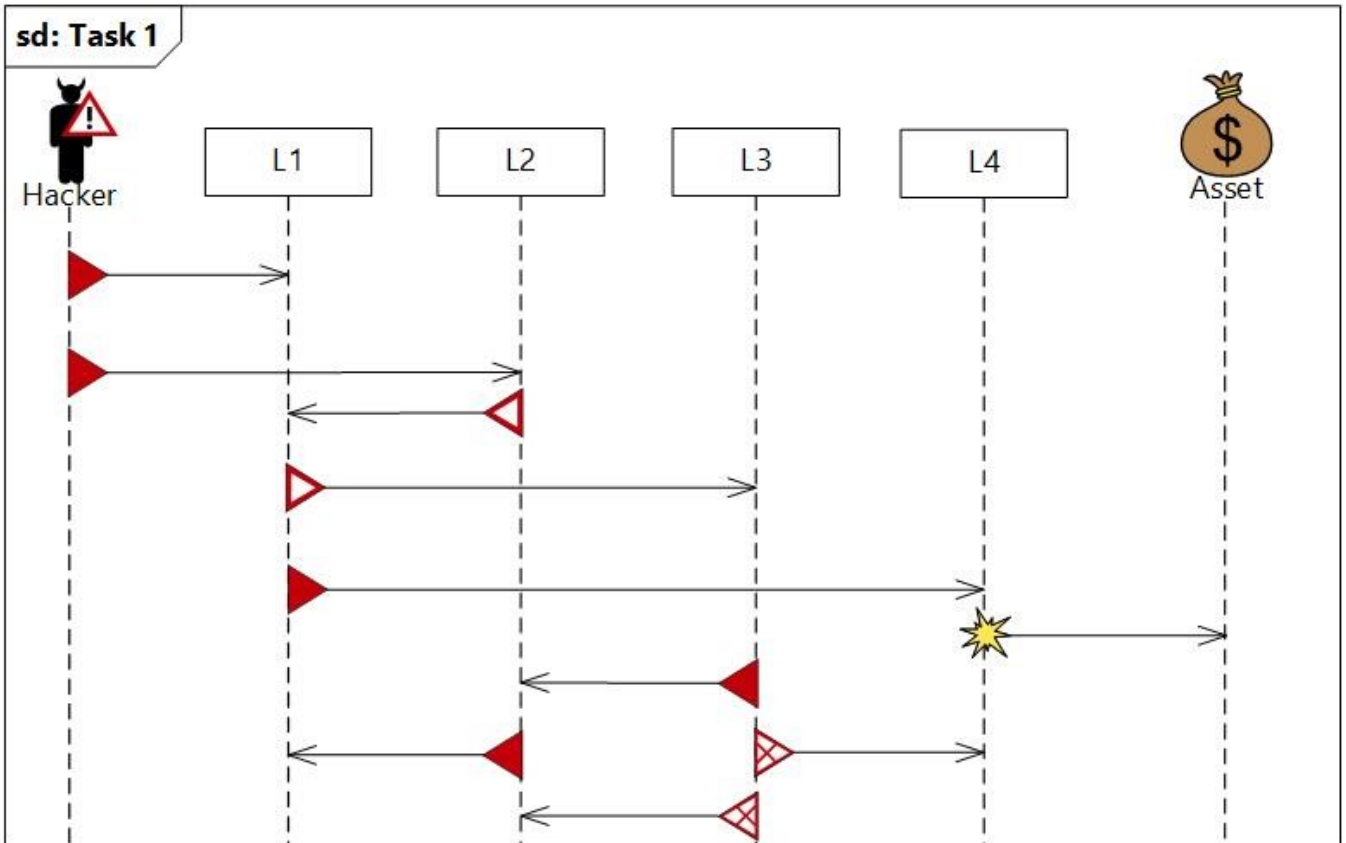
# Appendices



## **Appendix A**

# **Main Task Questionnaire – Group A**

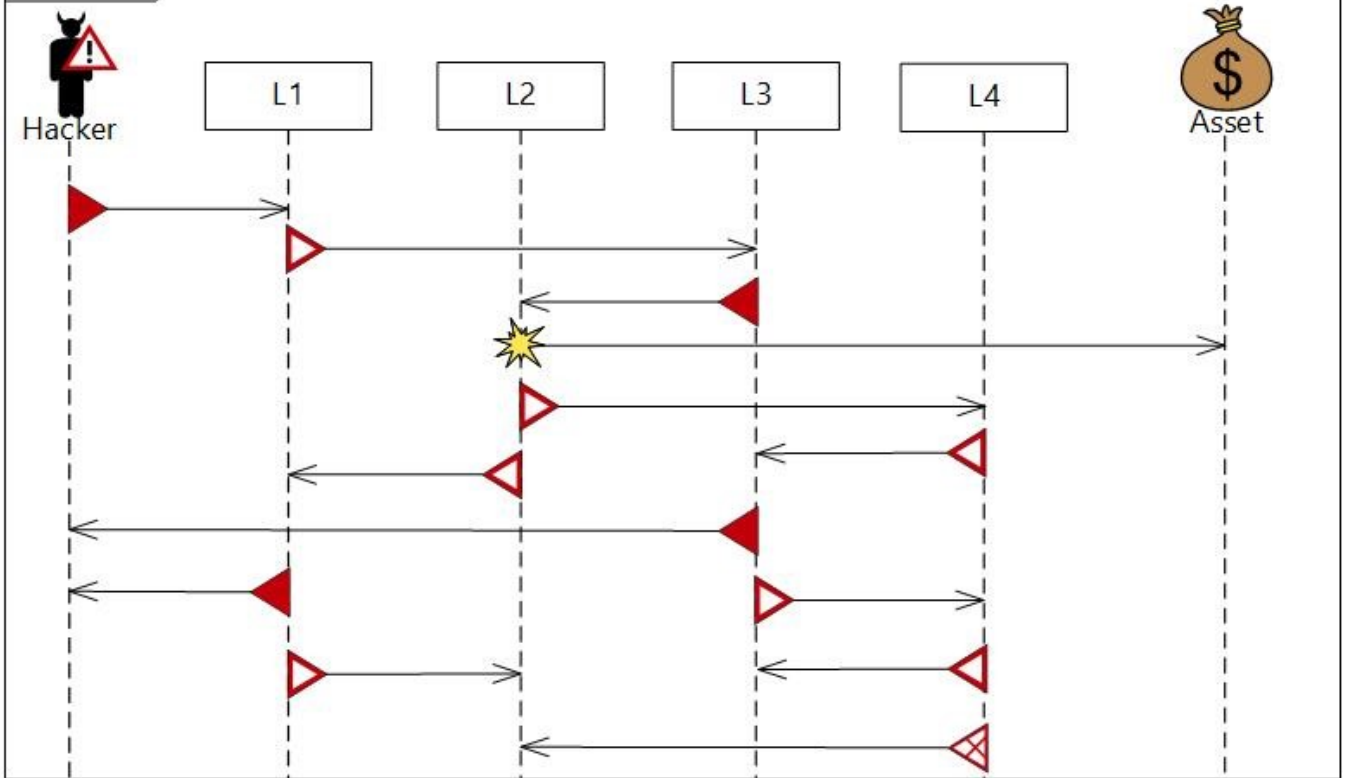
1. How many *new messages* are explicitly modelled in the threat model below?



2. How many *altered messages* are explicitly modelled in the threat model below?



sd: Task 2

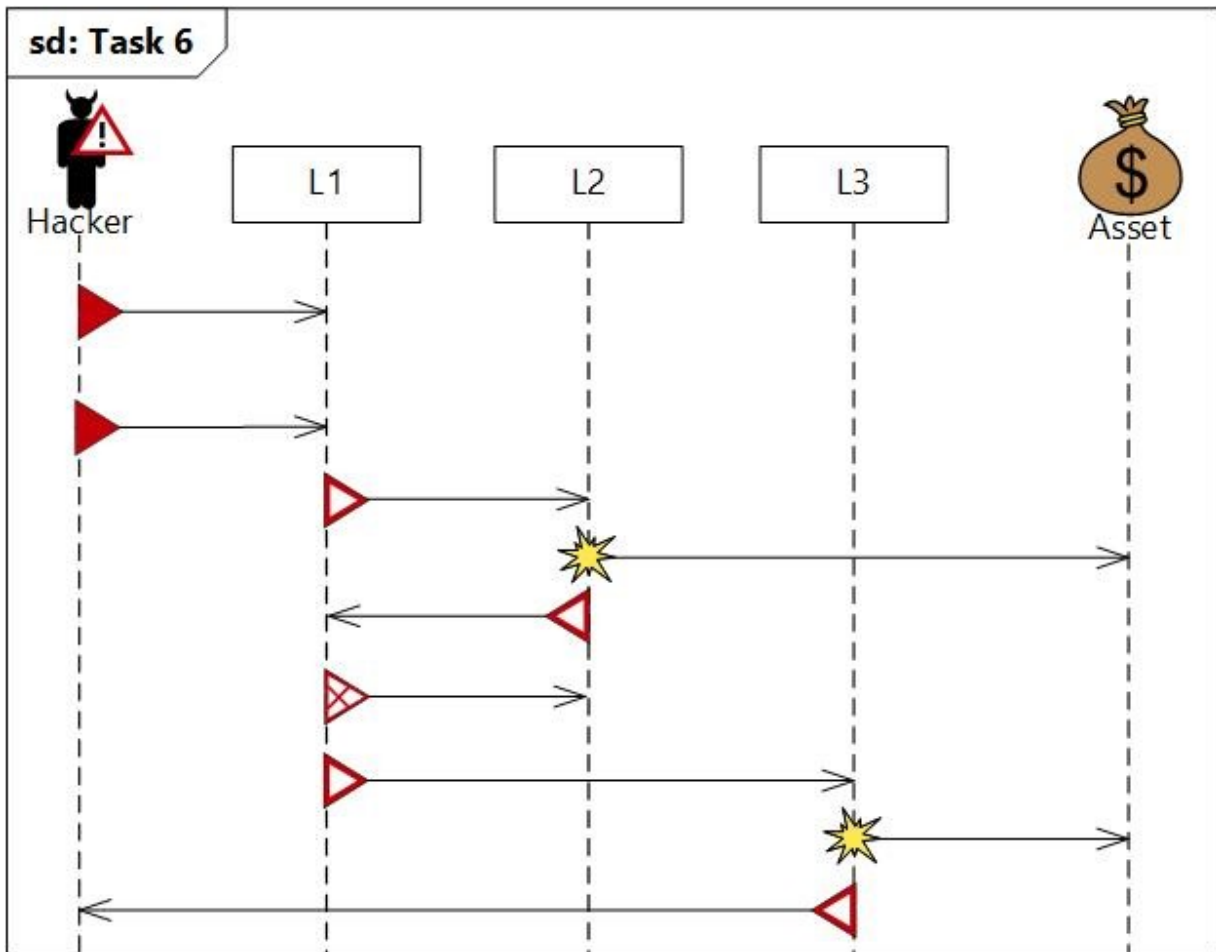


3. How many *deleted messages* are explicitly modelled in the threat model below?







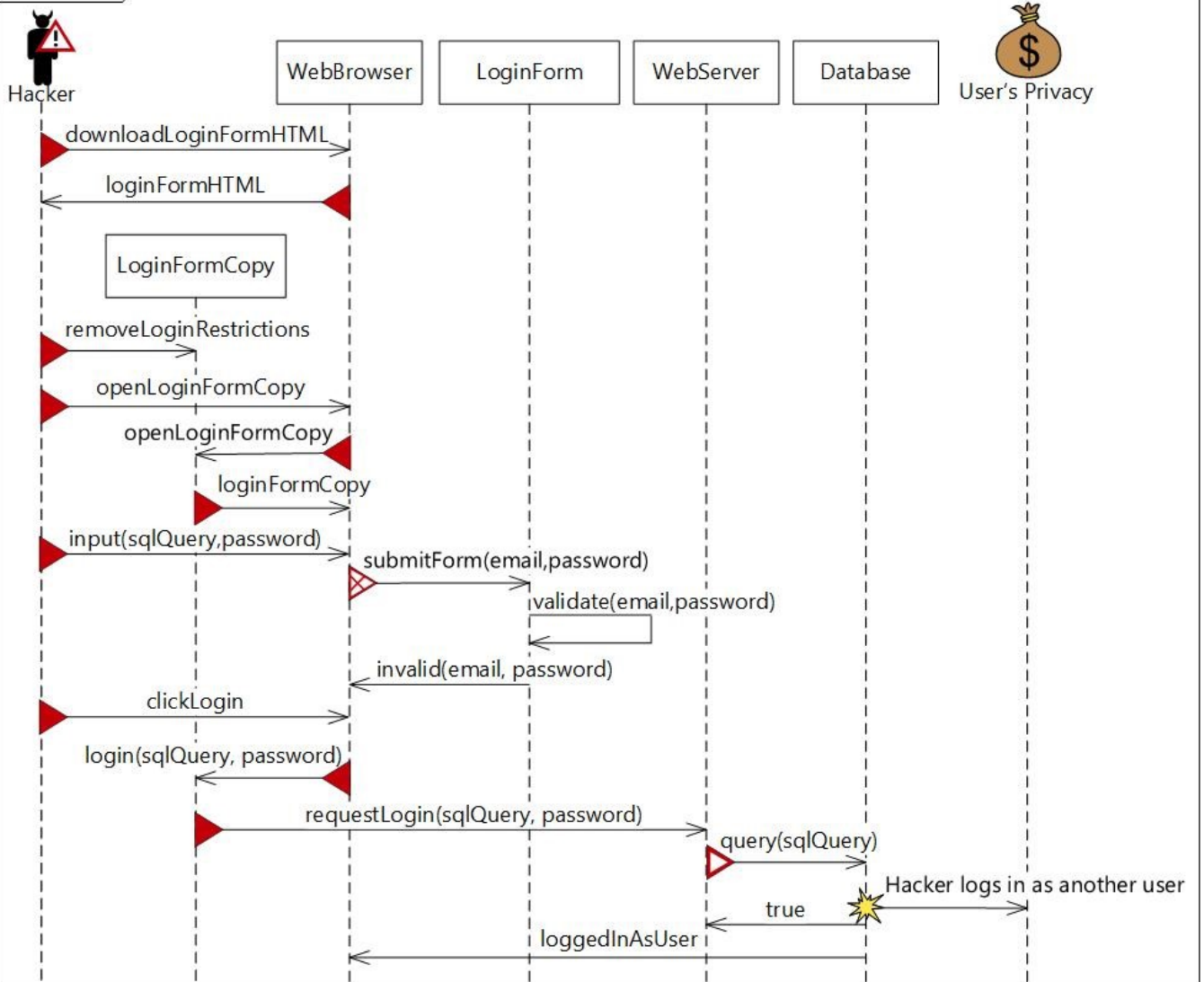


7. In the following diagram a hacker submits otherwise illegal input by downloading a local copy of the login form and removing input restrictions.

We focus on how messages can affect their succeeding messages.

Some messages in this diagram are supposed to be deleted messages, can you spot which?

sd: Task 7



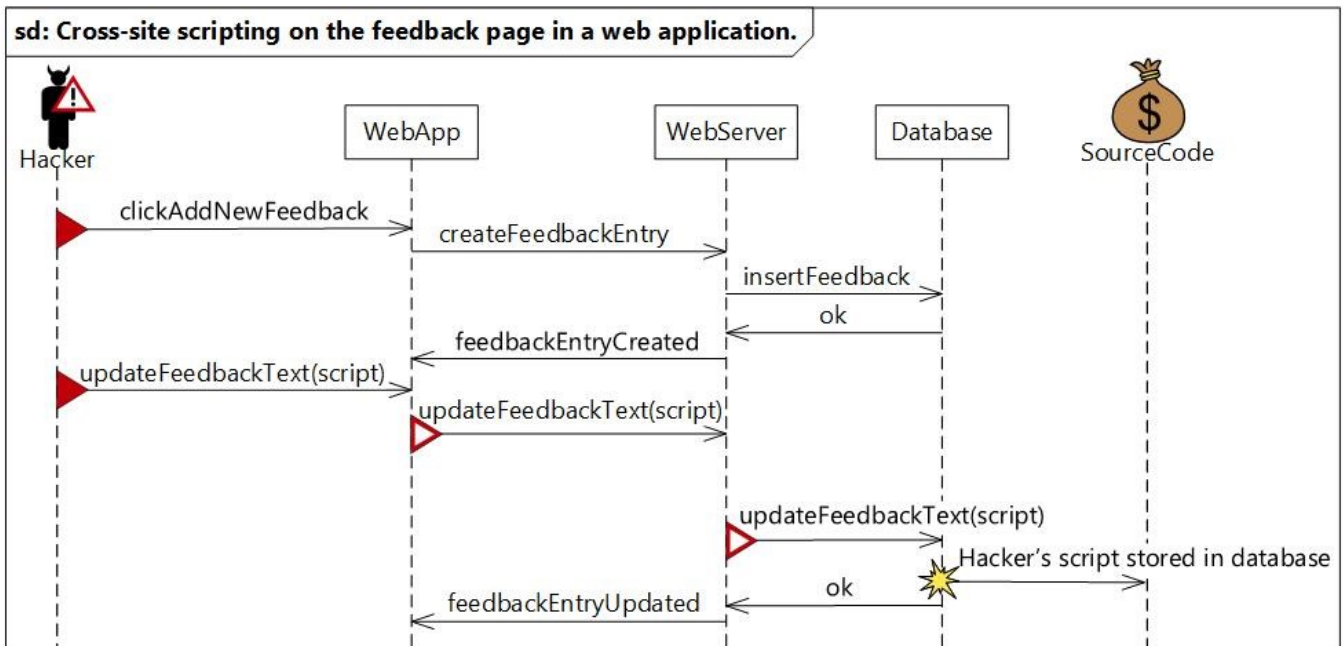
We use the following definitions of *Confidentiality*, *Availability* and *Integrity*:

- *Confidentiality*: Only authorised actors have access to information.
- *Availability*: Authorised actors have access to information they need when they need it.
- *Integrity*: Only authorised actors can change, create or delete information.

Recall that an unwanted incident is a message that can potentially harm or reduce an asset's value.

8. In the threat model below, the asset *SourceCode* can potentially be harmed. Which statements are the most accurate with respect to the threat model?
- The unwanted incident can potentially harm the *confidentiality* of the asset *SourceCode*.
  - The unwanted incident can potentially harm the *availability* of the asset *SourceCode*.
  - The unwanted incident can potentially harm the *integrity* of the asset *SourceCode*.

9. According to the model, describe how the hacker causes the unwanted incident to occur.



We use the following definitions of *Confidentiality*, *Availability* and *Integrity*:

- *Confidentiality*: Only authorised actors have access to information.
- *Availability*: Authorised actors have access to information they need when they need it.
- *Integrity*: Only authorised actors can change, create or delete information.

Recall that an unwanted incident is a message that can potentially harm or reduce an asset's value.

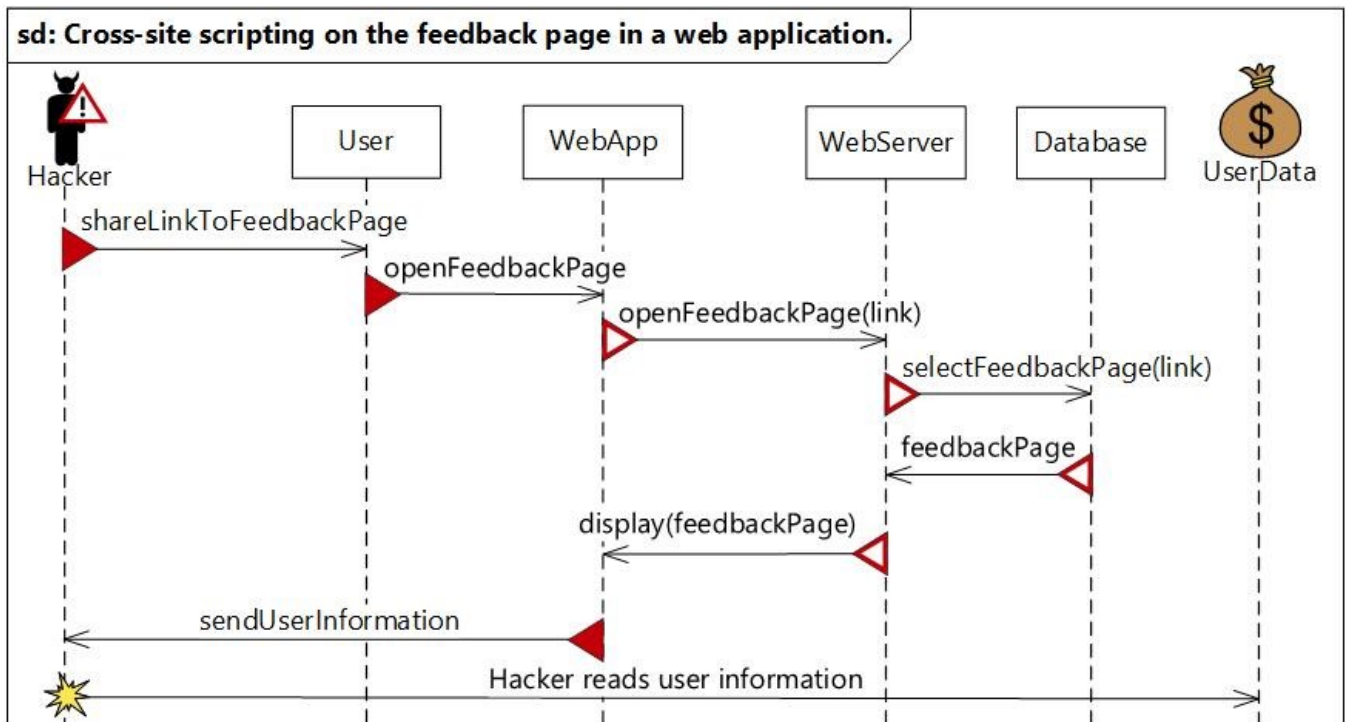
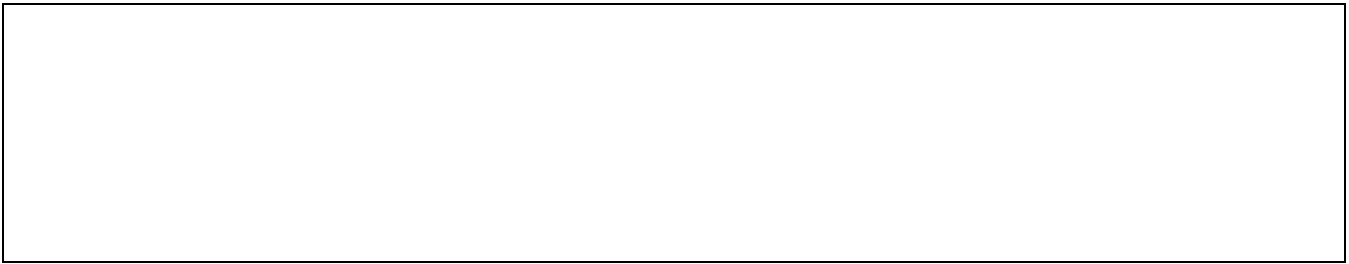
10. Now let's assume that the web app for privacy reasons hides users' real name and email address when someone views their feedback.  
Further let's assume that the hacker has stored a malicious script in the database.

In the threat model below, the asset *UserData* can potentially be harmed. Which statements are the most accurate with respect to the threat model?

- The unwanted incident can potentially harm the *confidentiality* of the asset *UserData*.
- The unwanted incident can potentially harm the *availability* of the asset *UserData*.
- The unwanted incident can potentially harm the *integrity* of the asset *UserData*.



11. According to the model, describe how the hacker causes the unwanted incident to occur.



Do not be startled by the alt operators in the threat model for these questions.

The guards in the alt (the text in brackets []) signifies what has to be true for the messages inside to occur.

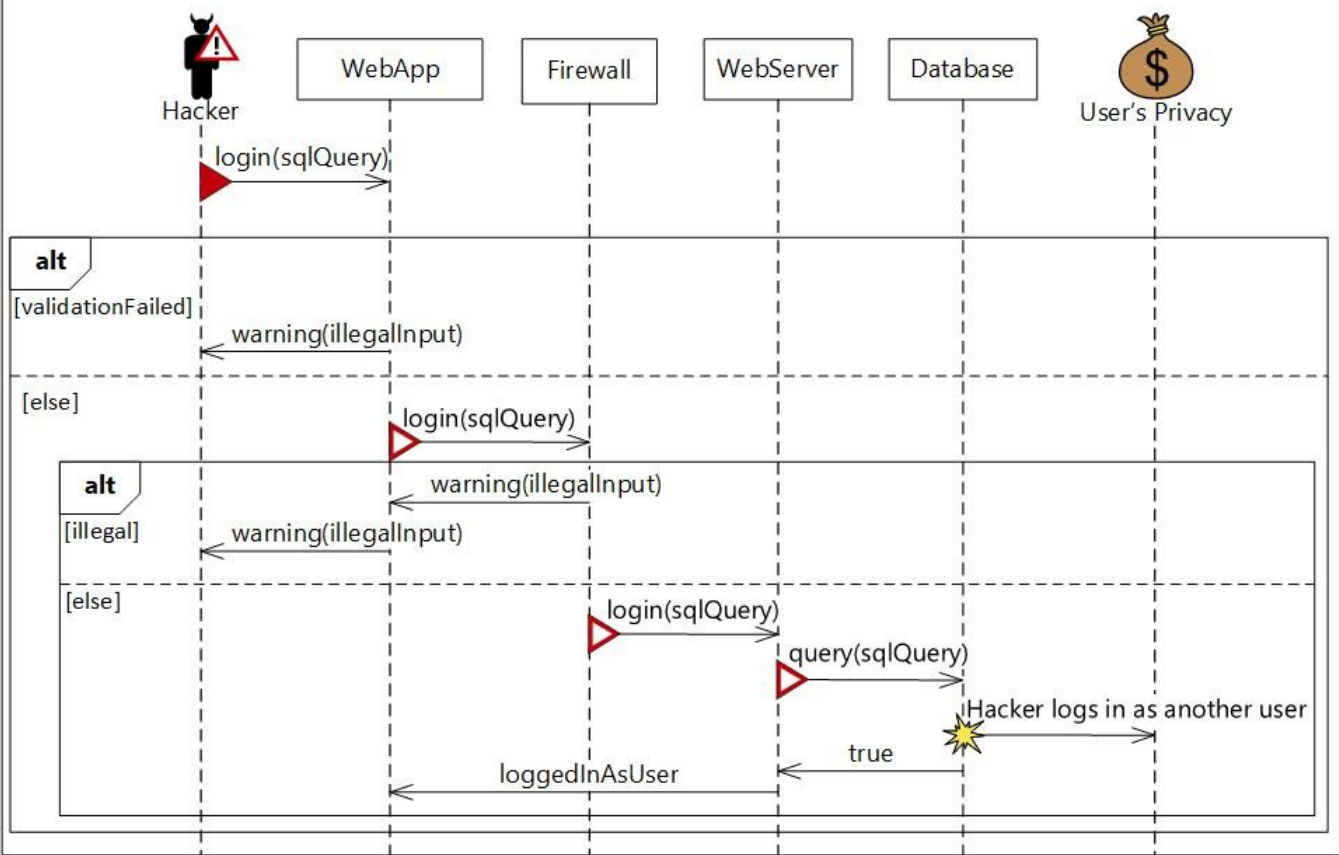
Basically the alts are read as, either this happens, or that happens, or nothing happens (if none of the conditions are met).

For this model for example, the input is either invalid or considered valid.

12. According to the model, describe how the hacker causes the unwanted incident to occur.

13. According to the model, describe all possibilities of where the attack might fail.

sd: SQL Injection on the login functionality of a web application



This is the last page of the questionnaire. Here we would like to get your feedback on the experiment.

14.

	Strongly Disagree	Disagree	Not Certain	Agree	Strongly Agree
I had enough time to solve the given tasks.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The objectives of the empirical experiment were clear to me.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The task descriptions were clear, and I understood what to do.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I experienced no difficulties in understanding the threat models.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I experienced no difficulties in answering questions in Part 1.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I experienced no difficulties in answering questions in Part 2.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I think the presentation given to me before the questionnaire provided me with enough information to solve the tasks.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

15. Do you have any comments with respect to the tasks given in the experiment?

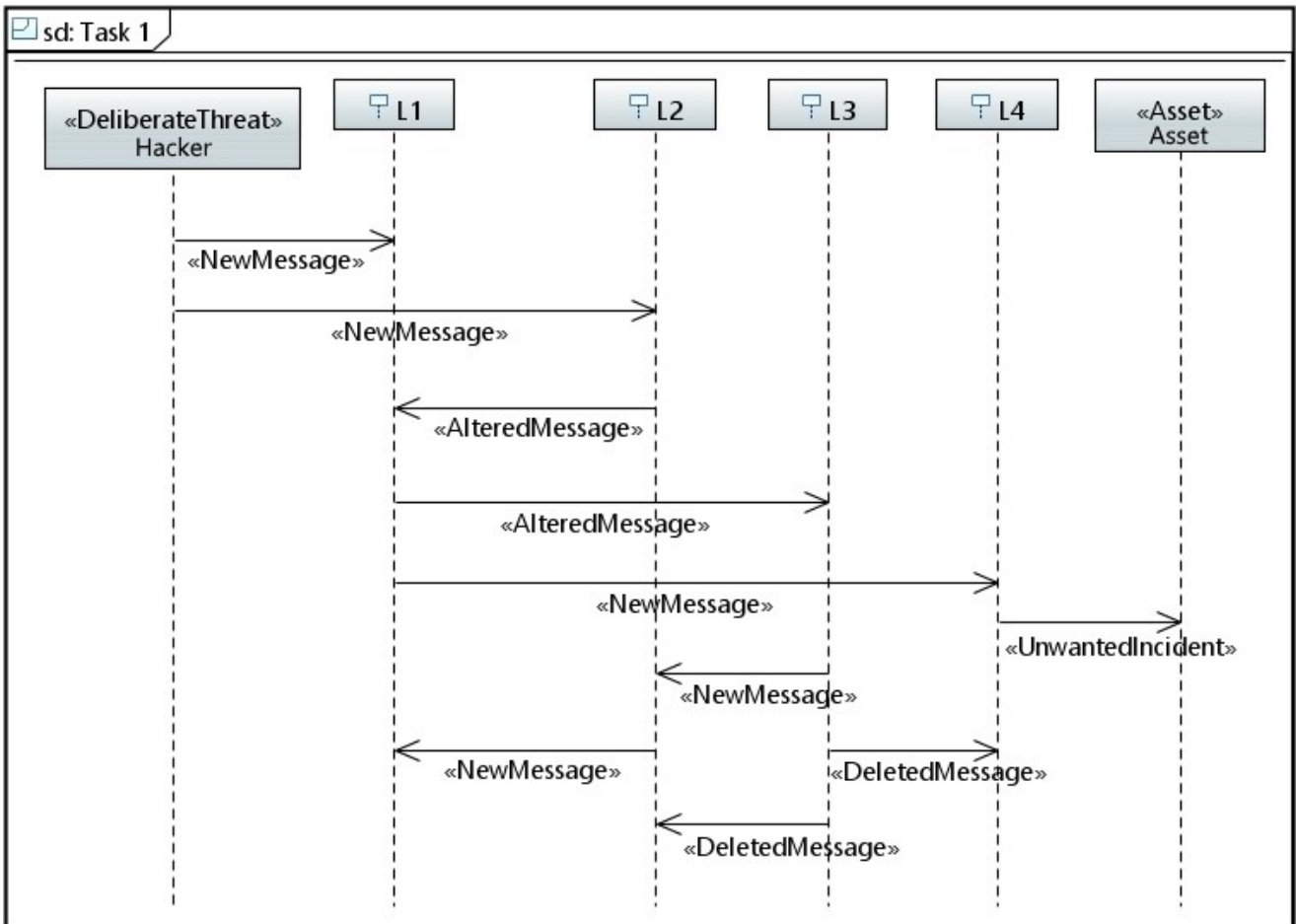
16. Do you have any further comments about the empirical experiment?



## **Appendix B**

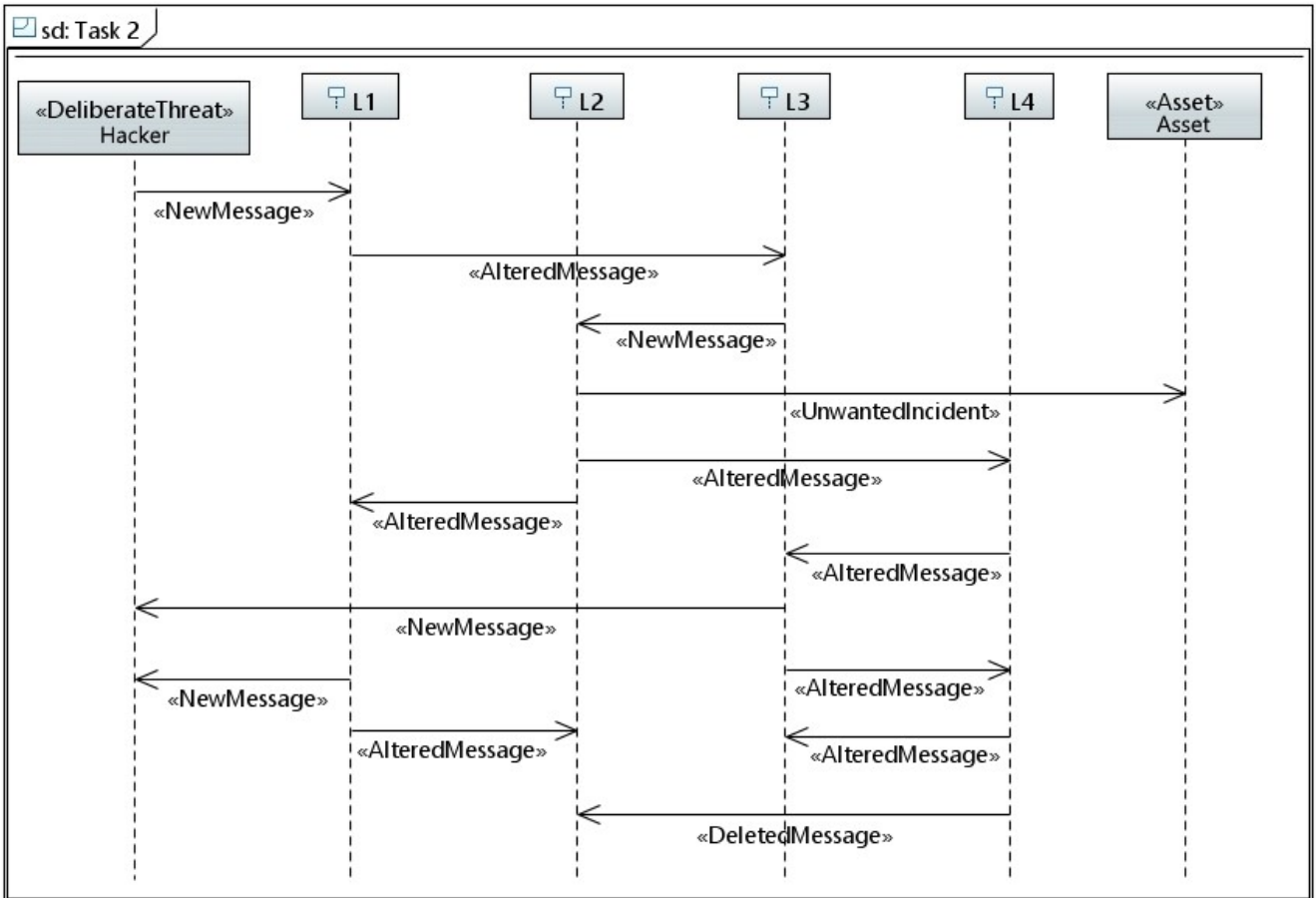
# **Main Task Questionnaire – Group B**

1. How many *new messages* are explicitly modelled in the threat model below?



2. How many *altered messages* are explicitly modelled in the threat model below?

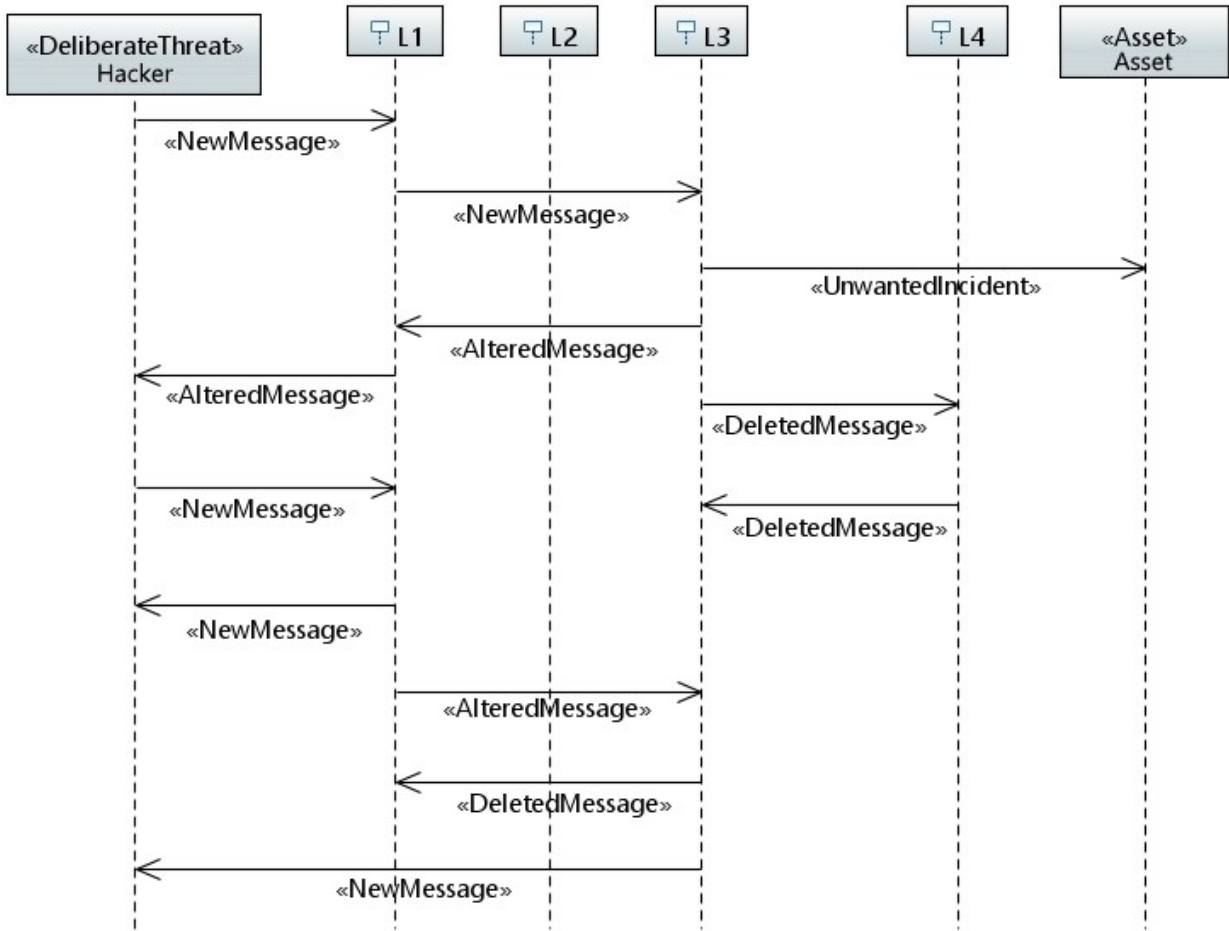




3. How many *deleted messages* are explicitly modelled in the threat model below?

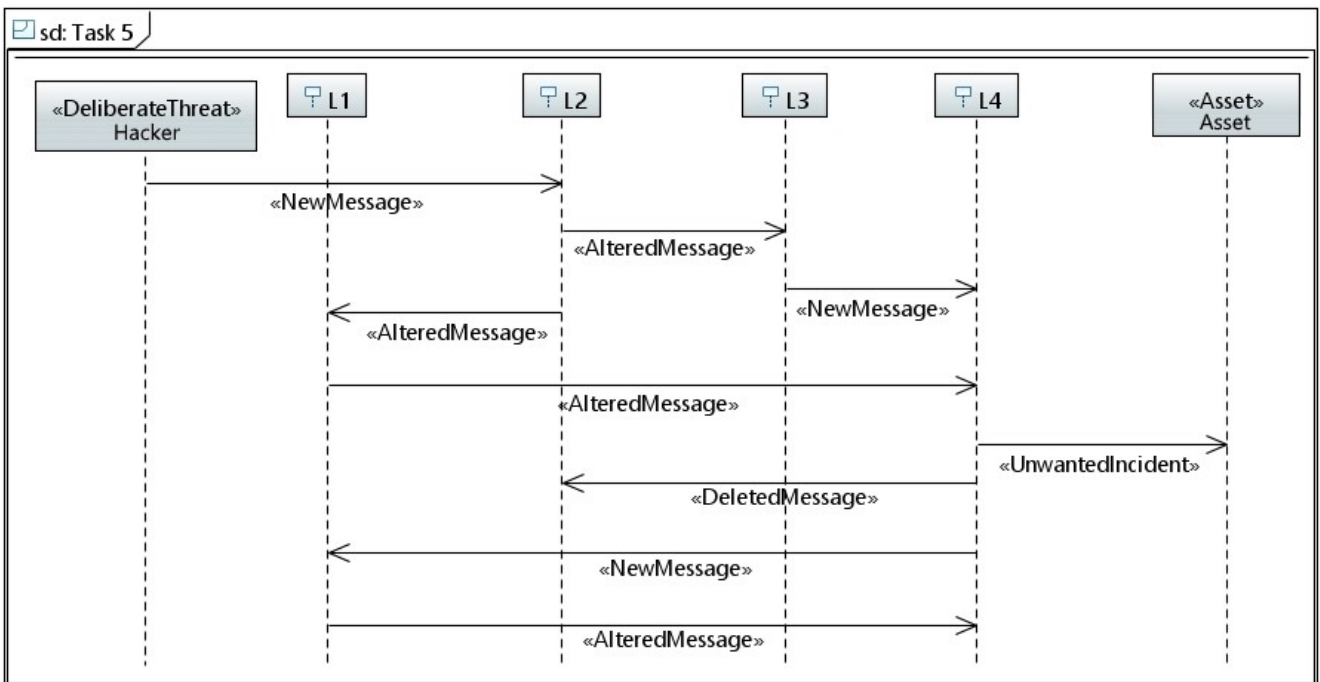


sc: Task 4



5. How many messages are explicitly modelled between lifeline L1 and L4 in the threat model below, and what are their types?

	0	1	2	3	4	5	6	7	8	9	10
General Message	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
New Message	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Altered Message	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Deleted Message	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Unwanted Incident	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



6. How many messages are explicitly modelled in the threat model below, and what are their types?

0 1 2 3 4 5 6 7 8 9 10

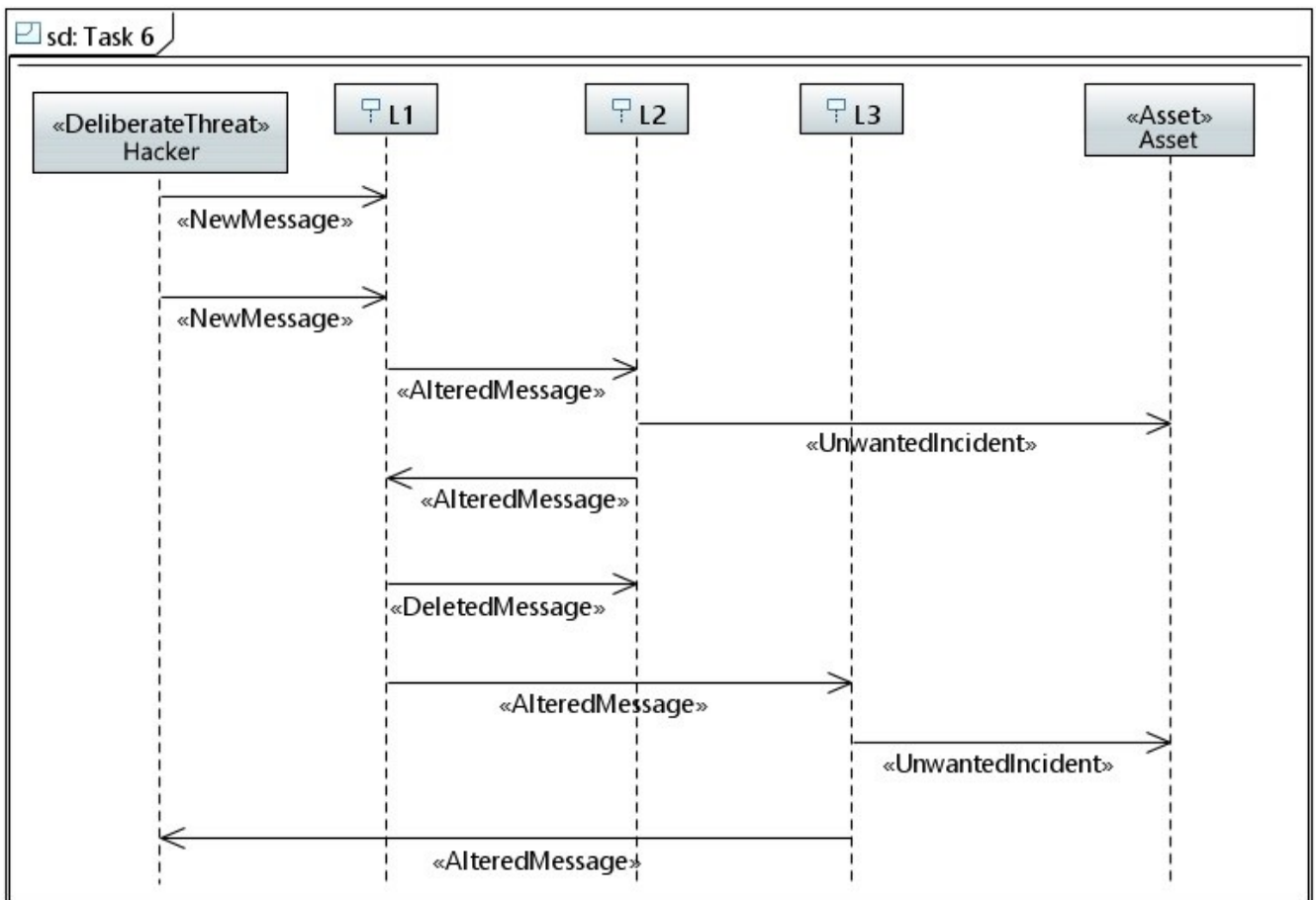
General Message

New Message

Altered Message

Deleted Message

Unwanted Incident



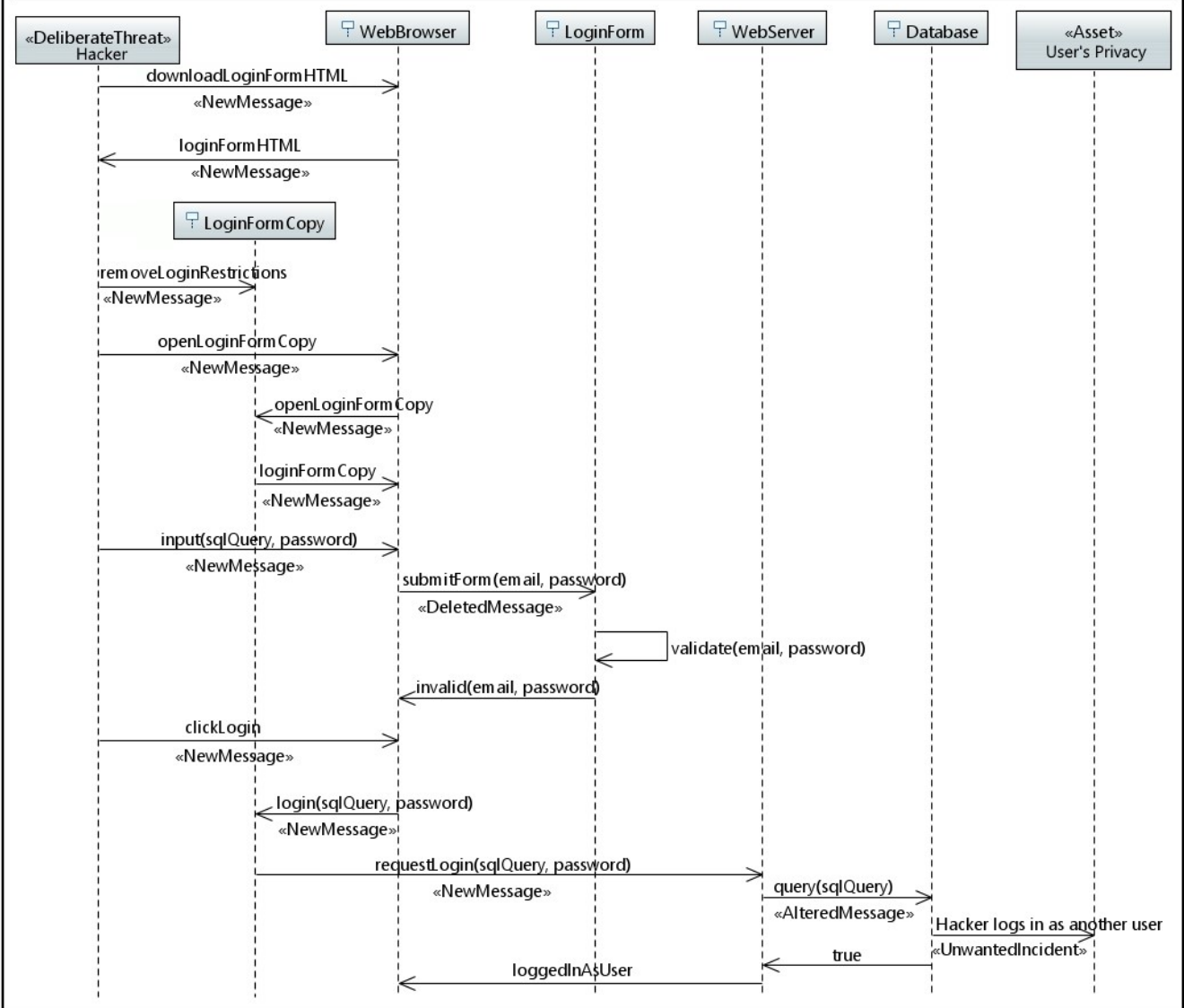
7. In the following diagram a hacker submits otherwise illegal input by downloading a local copy of the login form and removing input restrictions.

We focus on how messages can affect their succeeding messages.

Some messages in this diagram are supposed to be deleted messages, can you spot which?



sd: Task 7



We use the following definitions of *Confidentiality*, *Availability* and *Integrity*:

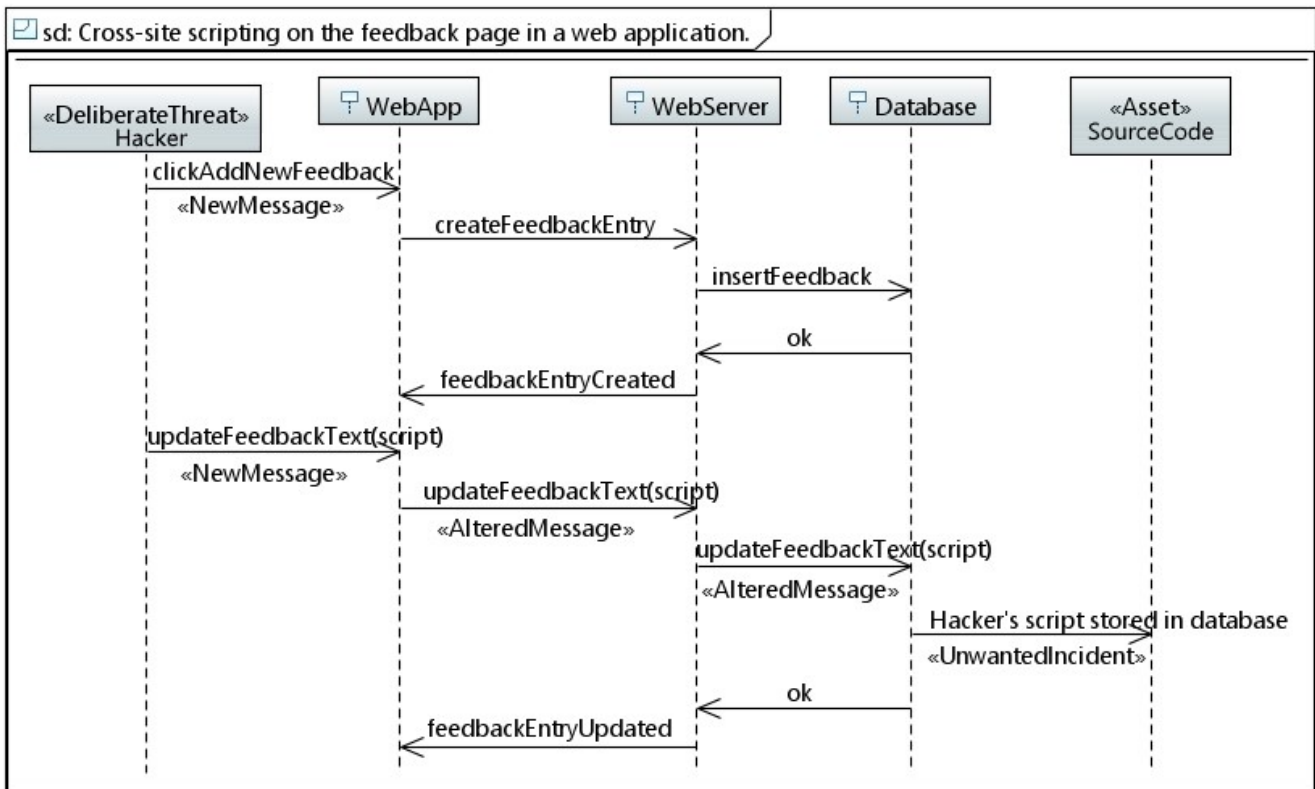
- *Confidentiality*: Only authorised actors have access to information.
- *Availability*: Authorised actors have access to information they need when they need it.
- *Integrity*: Only authorised actors can change, create or delete information.

Recall that an unwanted incident is a message that can potentially harm or reduce an asset's value.

8. In the threat model below, the asset *SourceCode* can potentially be harmed. Which statements are the most accurate with respect to the threat model?
- The unwanted incident can potentially harm the *confidentiality* of the asset *SourceCode*.
  - The unwanted incident can potentially harm the *availability* of the asset *SourceCode*.
  - The unwanted incident can potentially harm the *integrity* of the asset *SourceCode*.

9. According to the model, describe how the hacker causes the unwanted incident to occur.





We use the following definitions of *Confidentiality*, *Availability* and *Integrity*:

- *Confidentiality*: Only authorised actors have access to information.
- *Availability*: Authorised actors have access to information they need when they need it.
- *Integrity*: Only authorised actors can change, create or delete information.

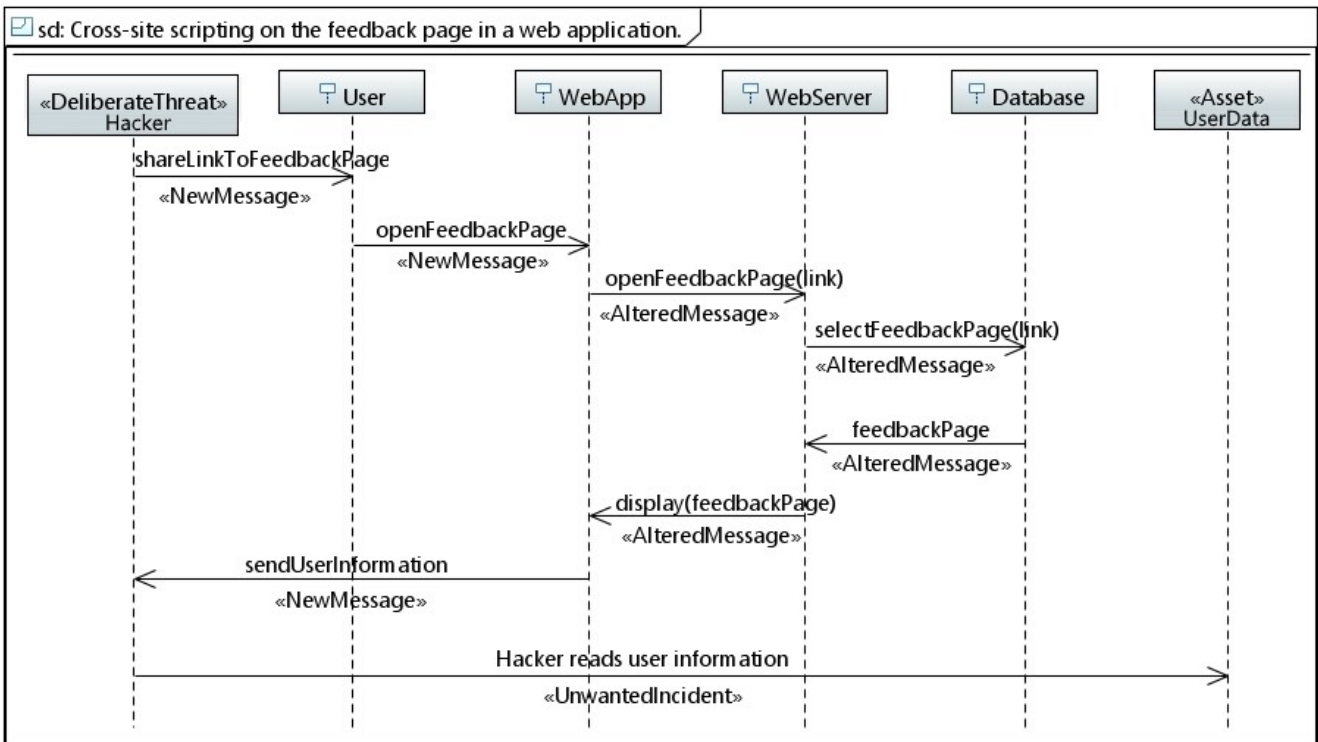
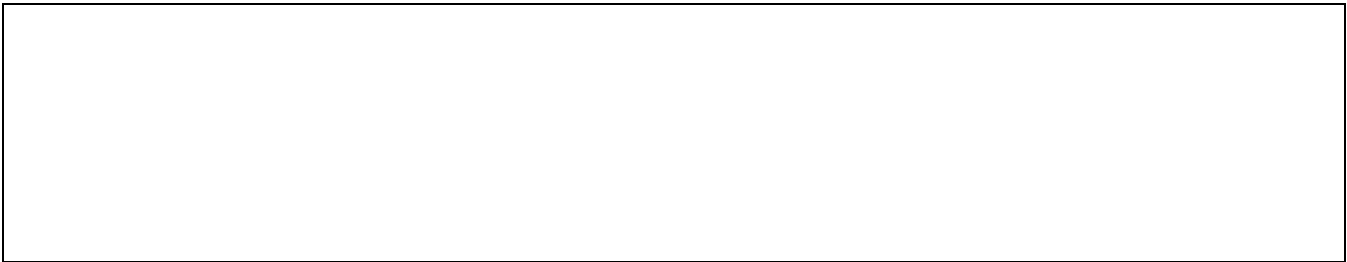
Recall that an unwanted incident is a message that can potentially harm or reduce an asset's value.

10. Now let's assume that the web app for privacy reasons hides users' real name and email address when someone views their feedback.  
Further let's assume that the hacker has stored a malicious script in the database.

In the threat model below, the asset *UserData* can potentially be harmed. Which statements are the most accurate with respect to the threat model?

- The unwanted incident can potentially harm the *confidentiality* of the asset *UserData*.
- The unwanted incident can potentially harm the *availability* of the asset *UserData*.
- The unwanted incident can potentially harm the *integrity* of the asset *UserData*.

11. According to the model, describe how the hacker causes the unwanted incident to occur.



Do not be startled by the alt operators in the threat model for these questions.

The guards in the alt (the text in brackets []) signifies what has to be true for the messages inside to occur.

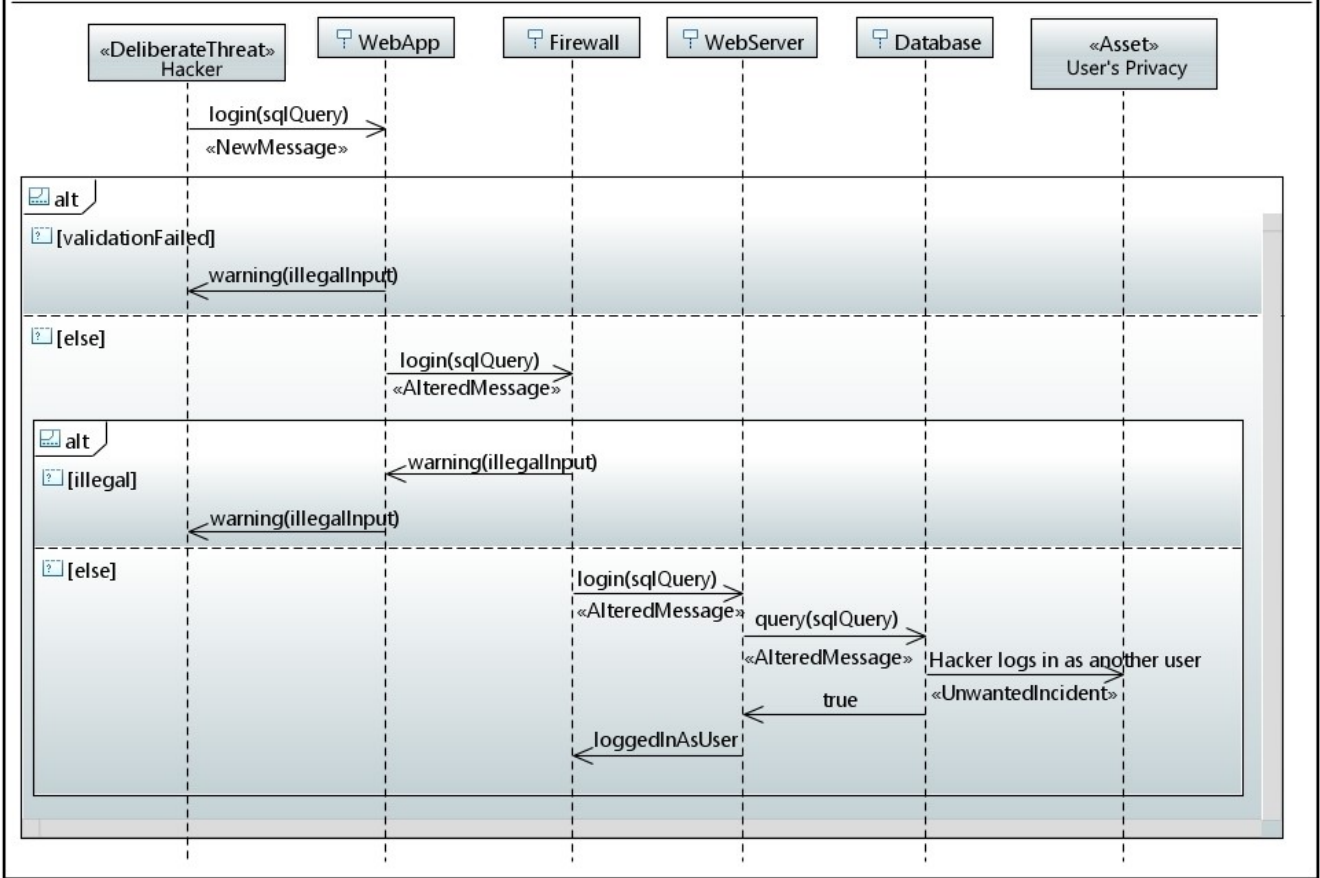
Basically the alt are read as, either this happens, or that happens.

For this model for example, the input is either invalid or considered valid.

12. According to the model, describe how the hacker causes the unwanted incident to occur.

13. According to the model, describe all possibilities of where the attack might fail.

sd: SQL Injection on the login functionality of a web application.



This is the last page of the questionnaire. Here we would like to get your feedback on the experiment.

14.

	Strongly Disagree	Disagree	Not Certain	Agree	Strongly Agree
I had enough time to solve the given tasks.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The objectives of the empirical experiment were clear to me.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The task descriptions were clear, and I understood what to do.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I experienced no difficulties in understanding the threat models.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I experienced no difficulties in answering questions in Part 1.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I experienced no difficulties in answering questions in Part 2.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I think the presentation given to me before the questionnaire provided me with enough information to solve the tasks.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>


15. Do you have any comments with respect to the tasks given in the experiment?

16. Do you have any further comments about the empirical experiment?

## **Appendix C**

### **Presentation – Group A**

# Experiment – Assessment of Graphical Notation

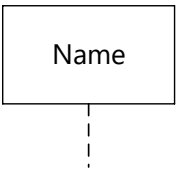

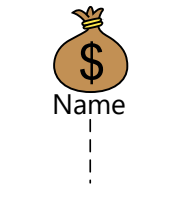


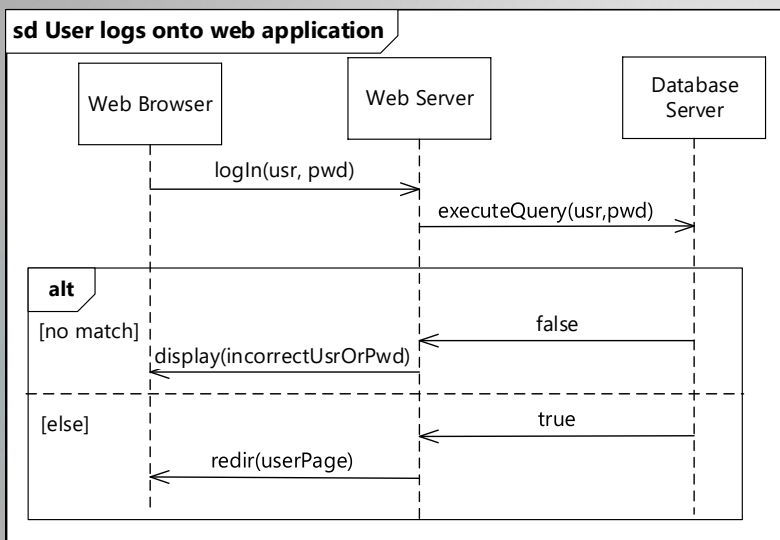
- In this experiment we need your help to assess a graphical notation used to model attacks in sequence diagrams.
- **Stage 1:**  
You will be given a survey to gather background information. This will be used to further divide participants into two groups fairly based on competence. Furthermore, the letter of consent must be signed and emailed back to [vettlevo@ifi.uio.no](mailto:vettlevo@ifi.uio.no)
- **Stage 2:**  
You will be given a survey with tasks to solve.  
There are a total of 13 tasks to solve, divided into two parts:
  - Part 1: 6 tasks given a 1 minute time restriction.
  - Part 2: 7 tasks given time restrictions > 1 minute.
  - At the end there is a short post-experiment questionnaire that will assess your experience with the experiment.



# Graphical Notation



Lifelines		
Lifeline type	Notation	Description
General lifeline		A general lifeline is a process/part of the system or an environment that interacts with the system.
Deliberate threat lifeline		A deliberate threat is a human threat that has malicious intents.
Asset lifeline		An asset lifeline is a security asset which we want to protect.

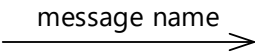
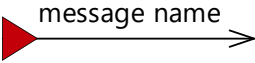
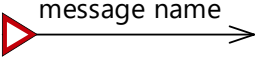
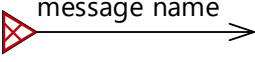
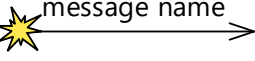


Simple example of a log in feature in a web application

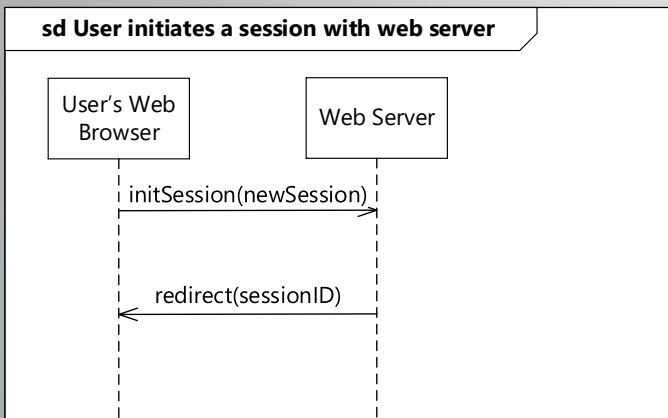
1. The scenario begins with the message `login(usr, pwd)`
2. The web server then executes a query on the database to verify the user credentials.
3. We then enter the `alt` (potential alternatives) with two guards (`[no match]`, `[else]`).
4. If the provided credentials are incorrect, the database will issue a message `false`, and the web server will notify the user that either the username or password is incorrect.
5. If the credentials provide a match, the web server redirects the web browser to the user page.

# Graphical Notation



<b>Messages</b>		
<b>Message type</b>	<b>Notation</b>	<b>Description</b>
General message		The general message is a message that has not been manipulated by a threat.
New message		The new message is a new message in the system with the intention of manipulating system behaviour.
Altered message		The altered message is a message that has been manipulated to deviate from expected behaviour.
Deleted message		A deleted message is a message that has been deleted as a result of previous manipulations.
Unwanted incident message		An unwanted incident represents an event that harms or reduces the value of an asset.

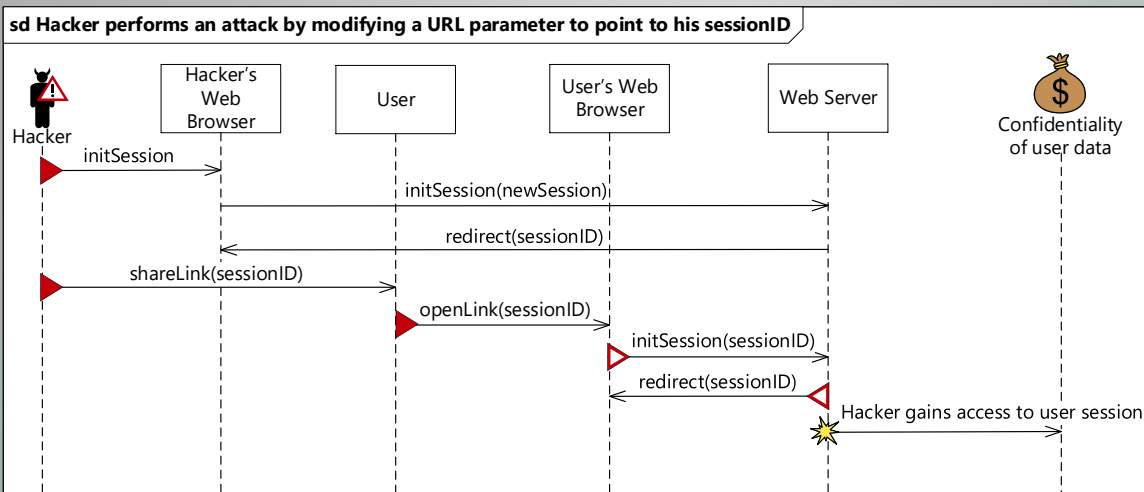
# Threat Model - Example



➤ Let's assume we have a very naive implementation for session management.

1. A user's web browser issues a request to the web server that it wants a new session.
2. The web server replies by redirecting the user's web browser to a web page with the new sessionID.

# Threat Model - Example



Messages	
Message type	Notation
General message	message name →
New message	▶ message name →
Altered message	◁ message name →
Deleted message	◁ message name →
Unwanted incident message	★ message name →


- Further, let's assume our deliberate threat is a hacker. The hacker performs an attack by first initiating a new session with the web site. Then, modifies a URL parameter to point to his sessionID.
- The hacker proceeds to share this link with another user of the web site through social engineering.
- The user opens the link and proceeds to use the web site with the same sessionID. If for example the user logs onto the web application using this session, the hacker will be logged in as well (as the user). (Assuming that the web server blindly accepts an existing sessionID for the `initSession()` function).



## **Appendix D**

### **Presentation – Group B**

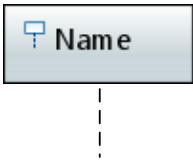
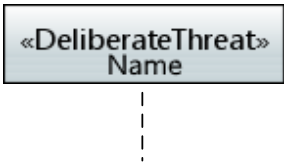

# Experiment – Assessment of Graphical Notation

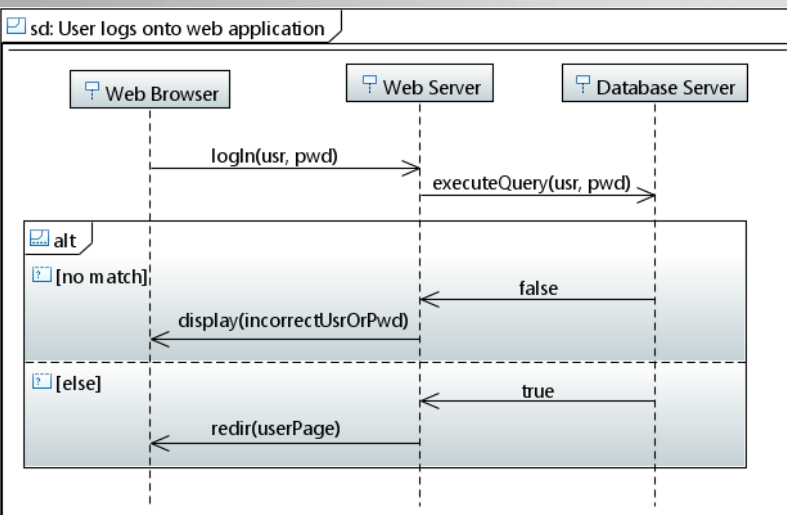


- In this experiment we need your help to assess a graphical notation used to model attacks in sequence diagrams.
- **Stage 1:**  
You will be given a survey to gather background information. This will be used to further divide participants into two groups fairly based on competence. Furthermore, the letter of consent must be signed and emailed back to [vetlevo@ifi.uio.no](mailto:vetlevo@ifi.uio.no)
- **Stage 2:**  
You will be given a survey with tasks to solve.  
There are a total of 13 tasks to solve, divided into two parts:
  - Part 1: 6 tasks given a 1 minute time restriction.
  - Part 2: 7 tasks given time restrictions > 1 minute.
  - At the end there is a short post-experiment questionnaire that will assess your experience with the experiment.



# Graphical Notation

Lifelines		
Lifeline type	Notation	Description
General lifeline		A general lifeline is a process/part of the system or an environment that interacts with the system.
Deliberate threat lifeline		A deliberate threat is a human threat that has malicious intents.
Asset lifeline		An asset lifeline is a security asset which we want to protect.



Simple example of a log in feature in a web application

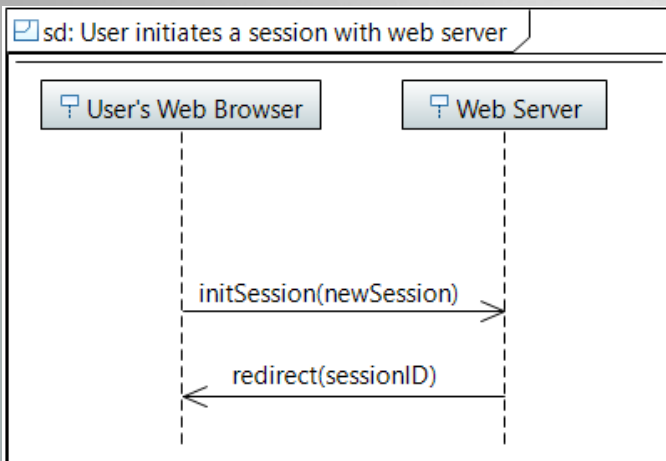
1. The scenario begins with the message `login(usr, pwd)`
2. The web server then executes a query on the database to verify the user credentials.
3. We then enter the alt (potential alternatives) with two guards (`[no match]`, `[else]`).
4. If the provided credentials are incorrect, the database will issue a message `false`, and the web server will notify the user that either the username or password is incorrect.
5. If the credentials provide a match, the web server redirects the web browser to the user page.

# Graphical Notation



<b>Messages</b>		
<b>Message type</b>	<b>Notation</b>	<b>Description</b>
General message	<u>message name</u> →	The general message is a message that has not been manipulated by a threat.
New message	<u>message name</u> «NewMessage» →	The new message is a new message in the system with the intention of manipulating system behaviour.
Altered message	<u>message name</u> «AlteredMessage» →	The altered message is a message that has been manipulated to deviate from expected behaviour.
Deleted message	<u>message name</u> «DeletedMessage» →	A deleted message is a message that has been deleted as a result of previous manipulations.
Unwanted incident message	<u>message name</u> «UnwantedIncident» →	An unwanted incident represents an event that harms or reduces the value of an asset.

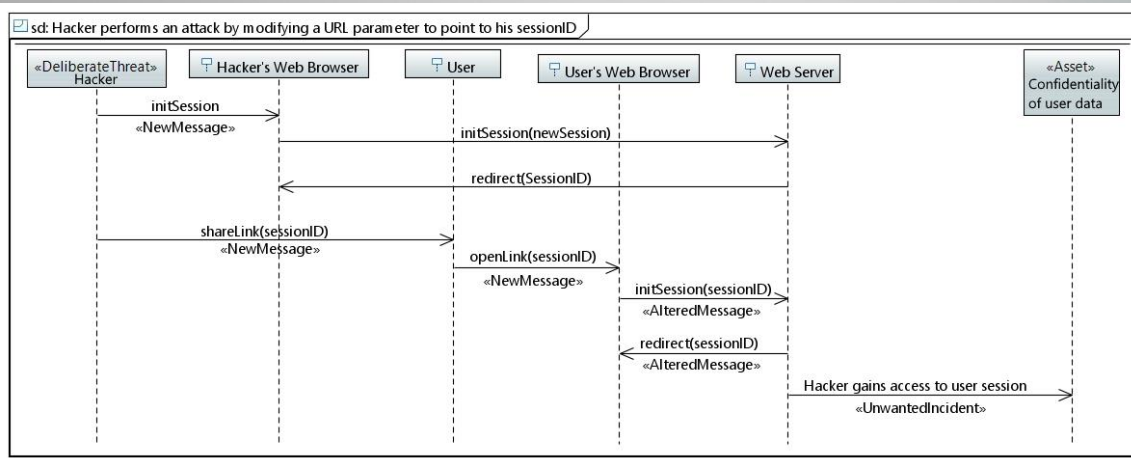
# Threat Model - Example



➤ Let's assume we have a very naive implementation for session management.

1. A user's web browser issues a request to the web server that it wants a new session.
2. The web server replies by redirecting the user's web browser to a web page with the new sessionID.

# Threat Model - Example



Messages	
Message type	Notation
General message	message name →
New message	message name «NewMessage» →
Altered message	message name «AlteredMessage» →
Deleted message	message name «DeletedMessage» →
Unwanted incident message	message name «UnwantedIncident» →

- Further, let's assume our deliberate threat is a hacker. The hacker performs an attack by first initiating a new session with the web site. Then, modifies a URL parameter to point to his sessionID.
- The hacker proceeds to share this link with another user of the web site through social engineering.
- The user opens the link and proceeds to use the web site with the same sessionID. If for example the user logs onto the web application using this session, the hacker will be logged in as well (as the user). (Assuming that the web server blindly accepts an existing sessionID for the initSession() function).



## **Appendix E**

# **Letter of Consent**

# Request to Participate in the Research Project

## Experiment - Assessment of Graphical Notation

### Background and Purpose

The research project is part of a master's thesis, and is conducted in order to assess graphical notation that is used to create models within the domain of risk-driven security testing. The master thesis project is affiliated with the University of Oslo and Sintef. The participants are contacted through the student's network.

### What Does Participation in the Study Involve?

The participation in the study is divided into two parts.

1. The participant is given the letter of consent through email along with a link to an online survey (*USIT's nettskjema*). This survey is a demographical survey. The demographical survey will be used to further divide participants into two groups fairly based on competence.
2. The participants are given a short presentation through email along with a link to an online questionnaire (*Eval&Go*). The presentation will briefly present the information needed in order to solve the tasks in the questionnaire. The questionnaire will include several tasks for the participants to solve. The tasks will be based on threat models with respect to attacks towards web applications. There are a total of 13 tasks to solve, divided into two parts:
  - Part 1: 6 tasks given a 1 minute time restriction.
  - Part 2: 7 tasks given time restrictions  $\geq$  1 minute.
  - At the end there is a short post-experiment questionnaire that will assess participants' experience with the experiment.

### What Happens With your Information?

The only direct personally identifiable information that is collected are email addresses. However, these are only used for the contact between the participant and the master's student responsible for the research project. The email addresses are not connected to any of the gathered data, and the list of email addresses will not be persisted after the experiment has been conducted. The project will according to plan be finished by late June depending on when all participants have finally submitted all the surveys. The only data that will be persisted



is indirect personally identifiable information like education level education programme and occupation.

## **Voluntary Participation**

It is voluntary to participate in the study, and you can at any time withdraw your consent without giving any reason. If you withdraw, all your information will be anonymous.

If you want to participate or have any questions about the study, please contact master's student Vetle Volden-Freberg +47 454 037 71 [vetlevo@ifi.uio.no](mailto:vetlevo@ifi.uio.no), or thesis supervisor Ketil Stølen +47 922 16 112 [Ketil.Stolen@Sintef.no](mailto:Ketil.Stolen@Sintef.no). The study has been reported to the Norwegian Centre for Research Data (NSD).

## **Consent for Participation in the Study**

I have received information about the study, and am willing to participate:

---

(Participant's signature / date)



## **Appendix F**

# **Task Scores from the Experiment**

Table F.1: Task scores for Group A.

Group A														
P#	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Total
P1	1	1	1	3	2	4	0	1	1	1	2	2	1	20
P2	1	1	1	1	0	4	2	1	3	1	0	3	2	20
P3	1	1	1	2	2	4	0	0	0	0	0	2	2	15
P4	0	0	0	0	1	2	0	1	0	1	0	0	0	5
P5	0	1	1	2	2	4	0	1	3	1	0	0	2	17
P6	1	1	1	3	2	4	2	1	2	1	0	1	2	21
P7	1	1	1	0	2	3	1	1	1	1	3	0	2	17
P8	1	1	1	2	2	4	0	0	3	1	3	2	2	22
<b>Avg.</b>	<b>0.75</b>	<b>0.875</b>	<b>0.875</b>	<b>1.625</b>	<b>1.625</b>	<b>3.625</b>	<b>0.625</b>	<b>0.75</b>	<b>1.625</b>	<b>0.875</b>	<b>1</b>	<b>1.25</b>	<b>1.625</b>	<b>17.125</b>

Table F.2: Task scores for Group B.

<b>Group B</b>														
<b>P#</b>	<b>Task 1</b>	<b>Task 2</b>	<b>Task 3</b>	<b>Task 4</b>	<b>Task 5</b>	<b>Task 6</b>	<b>Task 7</b>	<b>Task 8</b>	<b>Task 9</b>	<b>Task 10</b>	<b>Task 11</b>	<b>Task 12</b>	<b>Task 13</b>	<b>Total</b>
P9	1	1	1	3	2	4	0	1	2	1	2	3	2	23
P10	0	0	1	2	2	4	0	1	2	1	2	3	2	20
P11	1	1	1	0	1	4	0	1	0	1	0	1	0	11
P12	1	1	1	3	2	4	0	1	2	1	3	3	2	24
P13	1	1	1	3	2	3	0	1	2	0	3	2	2	21
P14	1	1	1	3	2	4	0	0	0	1	0	0	2	15
P15	0	0	0	1	1	4	0	1	0	1	0	0	0	8
P16	1	1	1	3	2	3	0	1	2	1	2	2	2	21
<b>Avg.</b>	<b>0.75</b>	<b>0.75</b>	<b>0.875</b>	<b>2.25</b>	<b>1.75</b>	<b>3.75</b>	<b>0</b>	<b>0.875</b>	<b>1.25</b>	<b>0.875</b>	<b>1.5</b>	<b>1.75</b>	<b>1.5</b>	<b>17.875</b>



## **Appendix G**

# **SPSS – Statistics Calculations**

### **G.1 Total Score**

```

EXAMINE VARIABLES=Total BY Group
/PLOT BOXPLOT HISTOGRAM
/COMPARE GROUPS
/STATISTICS DESCRIPTIVES
/CINTERVAL 95
/MISSING LISTWISE
/NOTOTAL.

```

## Explore

### Notes

Output Created		
Comments		
Input	Data	
	Active Dataset	DataSet1
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	16
Missing Value Handling	Definition of Missing	User-defined missing values for dependent variables are treated as missing.
	Cases Used	Statistics are based on cases with no missing values for any dependent variable or factor used.
Syntax		<pre> EXAMINE VARIABLES=Total BY Group /PLOT BOXPLOT HISTOGRAM /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL. </pre>
Resources	Processor Time	00:00:01,27
	Elapsed Time	00:00:00,43

## Group



**Case Processing Summary**

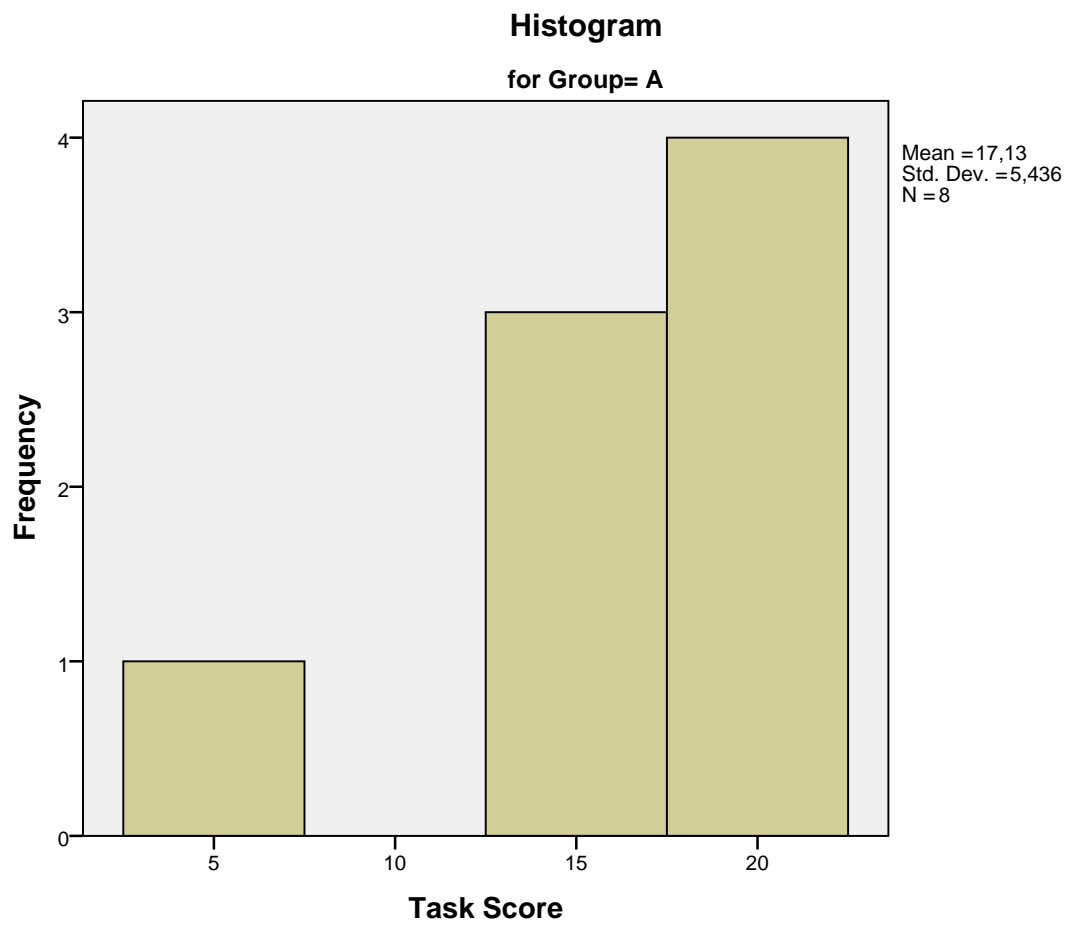
Group		Cases					
		Valid		Missing		Total	
		N	Percent	N	Percent	N	Percent
Task Score	A	8	100,0%	0	0,0%	8	100,0%
	B	8	100,0%	0	0,0%	8	100,0%

**Descriptives**

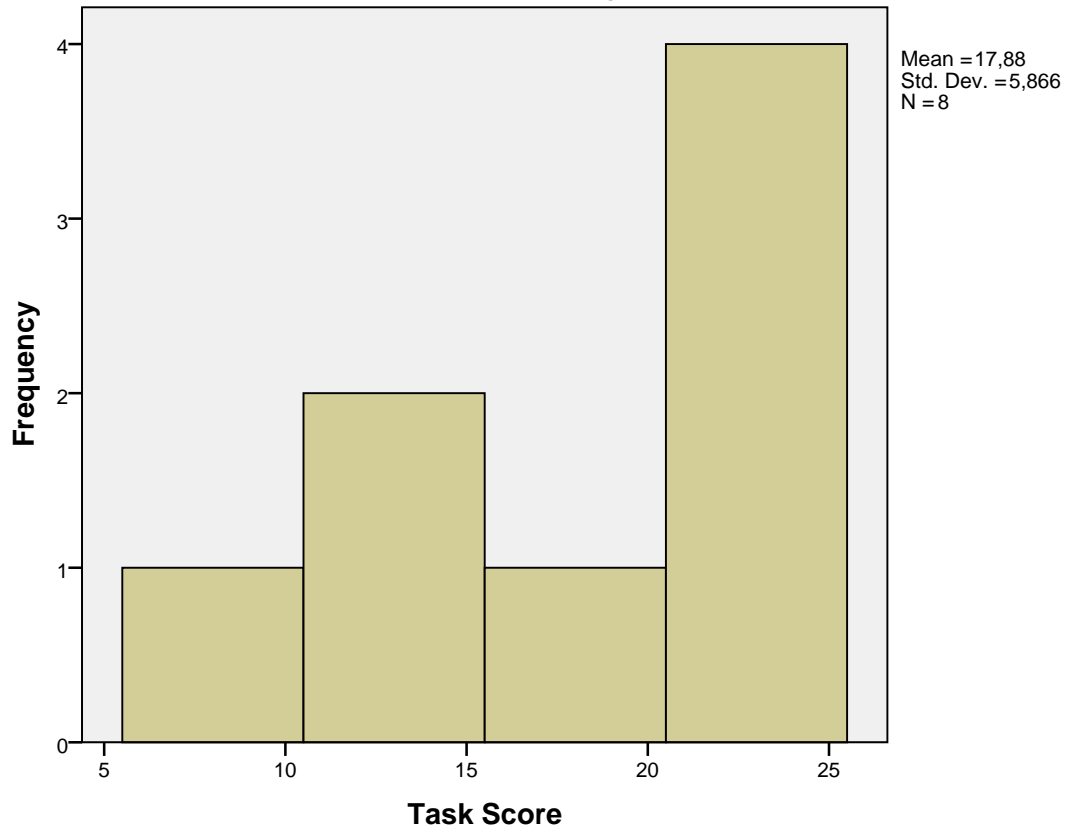
Group		Statistic	Std. Error		
Task Score	A	Mean	17,13	1,922	
		95% Confidence Interval for Mean	Lower Bound	12,58	
			Upper Bound	21,67	
		5% Trimmed Mean	17,53		
		Median	18,50		
		Variance	29,554		
		Std. Deviation	5,436		
		Minimum	5		
		Maximum	22		
		Range	17		
		Interquartile Range	5		
		Skewness	-1,862	,752	
		Kurtosis	3,956	1,481	
	B	Mean	17,88	2,074	
		95% Confidence Interval for Mean	Lower Bound	12,97	
			Upper Bound	22,78	
		5% Trimmed Mean	18,08		
		Median	20,50		
		Variance	34,411		
		Std. Deviation	5,866		
		Minimum	8		
		Maximum	24		
		Range	16		
		Interquartile Range	11		
		Skewness	-,827	,752	
		Kurtosis	-,812	1,481	

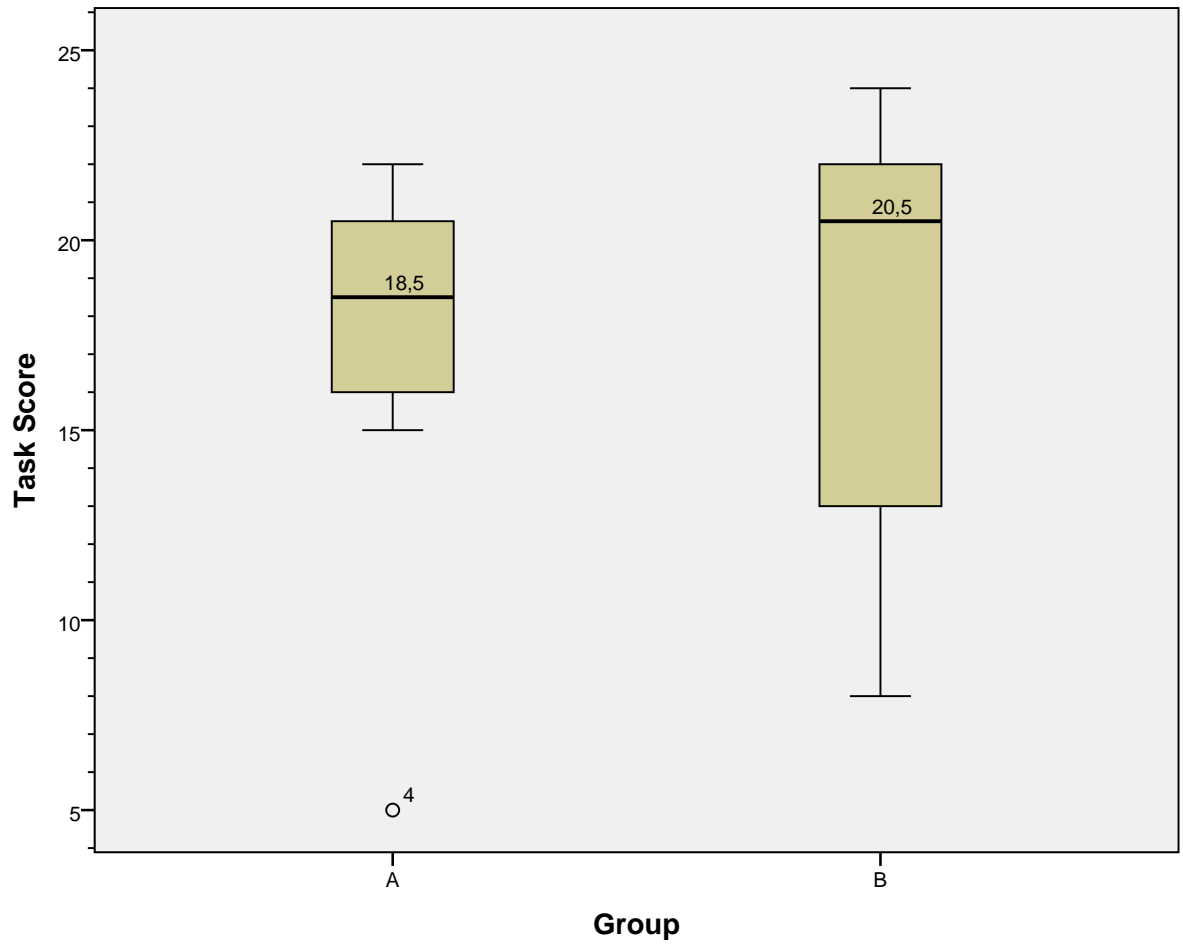
# Task Score

## Histograms



**Histogram**  
for Group= B





## **G.2 Total Score – Without Outlier**

```

EXAMINE VARIABLES=Total BY Group
/PLOT BOXPLOT HISTOGRAM
/COMPARE GROUPS
/STATISTICS DESCRIPTIVES
/CINTERVAL 95
/MISSING LISTWISE
/NOTOTAL.

```

## Explore

### Notes

Output Created		
Comments		
Input	Data	
	Active Dataset	DataSet2
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	15
Missing Value Handling	Definition of Missing	User-defined missing values for dependent variables are treated as missing.
	Cases Used	Statistics are based on cases with no missing values for any dependent variable or factor used.
Syntax		EXAMINE VARIABLES=Total BY Group /PLOT BOXPLOT HISTOGRAM /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL.
Resources	Processor Time	00:00:01,30
	Elapsed Time	00:00:00,48

## Group

**Case Processing Summary**

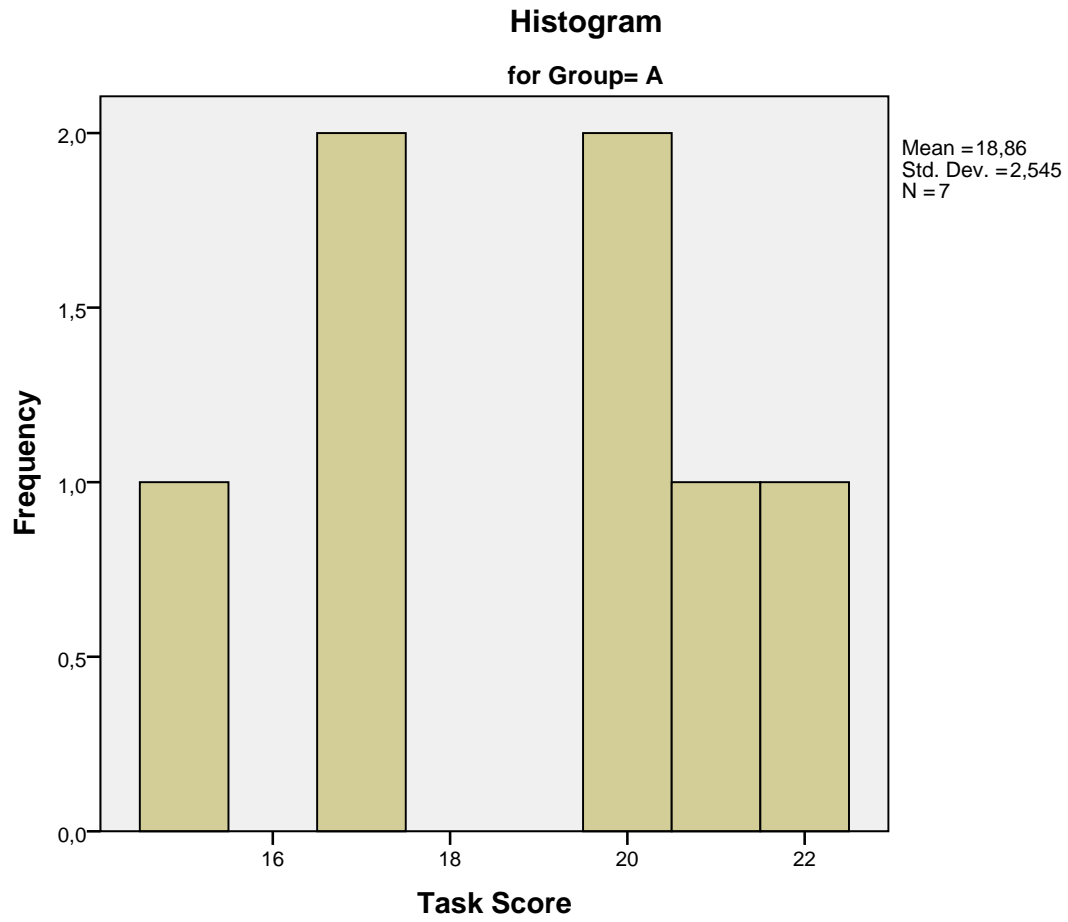
		Cases					
		Valid		Missing		Total	
		N	Percent	N	Percent	N	Percent
Task Score	A	7	100,0%	0	0,0%	7	100,0%
	B	8	100,0%	0	0,0%	8	100,0%

**Descriptives**

Group		Statistic	Std. Error		
Task Score	A	Mean	18,86	,962	
		95% Confidence Interval for Mean	Lower Bound	16,50	
			Upper Bound	21,21	
		5% Trimmed Mean	18,90		
		Median	20,00		
		Variance	6,476		
		Std. Deviation	2,545		
		Minimum	15		
		Maximum	22		
		Range	7		
		Interquartile Range	4		
		Skewness	-,373	,794	
		Kurtosis	-1,314	1,587	
B	B	Mean	17,88	2,074	
		95% Confidence Interval for Mean	Lower Bound	12,97	
			Upper Bound	22,78	
		5% Trimmed Mean	18,08		
		Median	20,50		
		Variance	34,411		
		Std. Deviation	5,866		
		Minimum	8		
		Maximum	24		
		Range	16		
		Interquartile Range	11		
		Skewness	-,827	,752	
		Kurtosis	-,812	1,481	

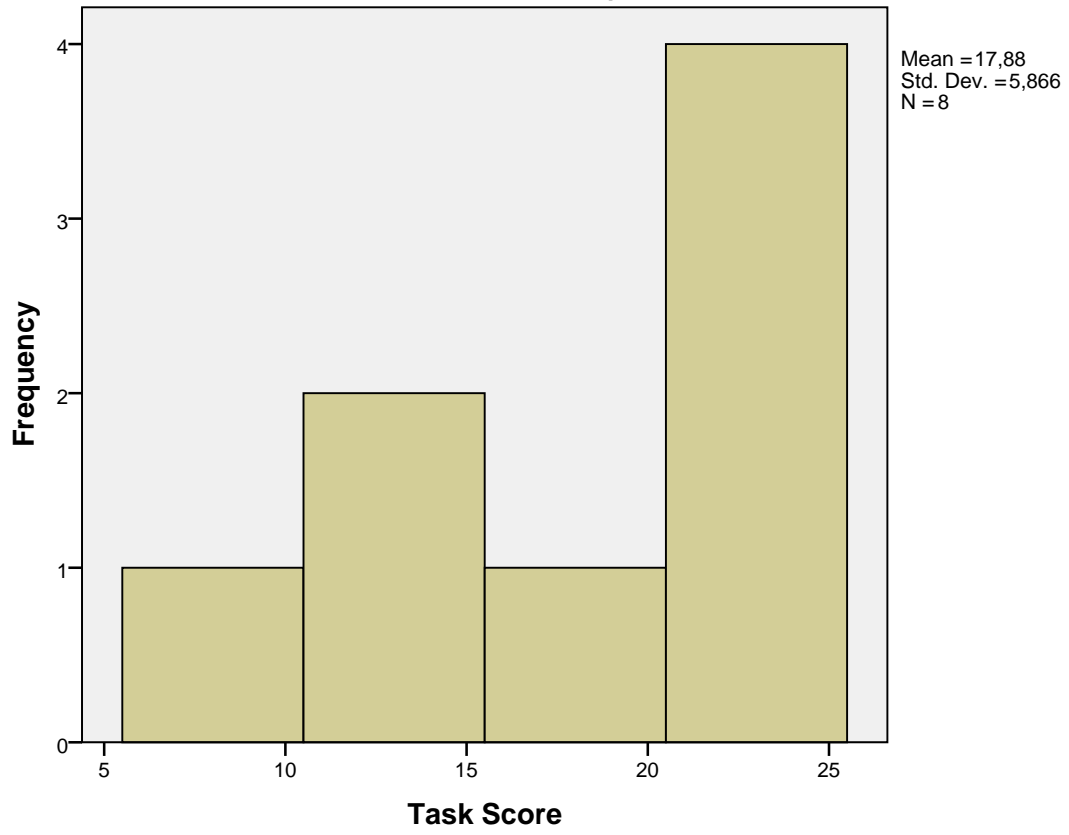
# Task Score

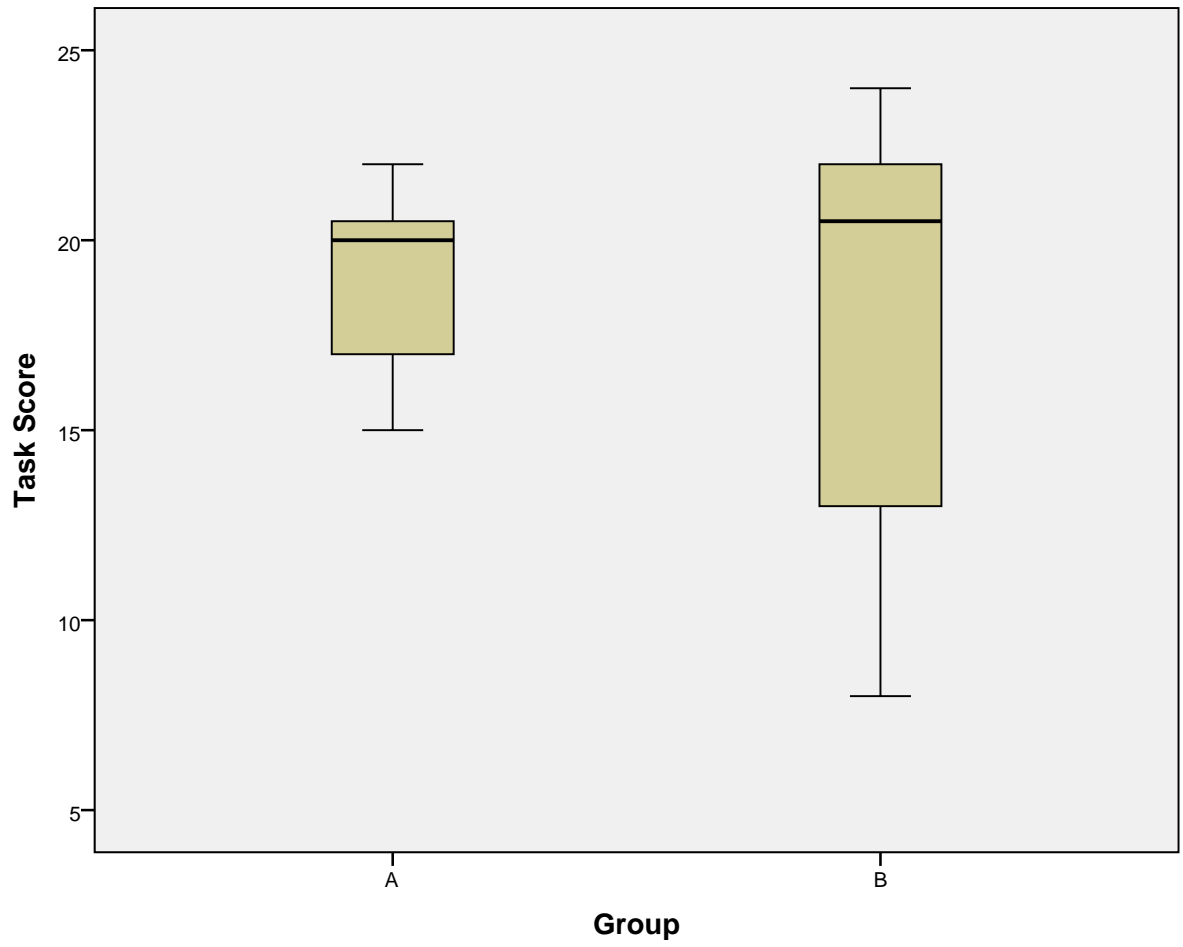
## Histograms





**Histogram**  
for Group= B





### **G.3 Part 1 Score**

```

/PLOT BOXPLOT HISTOGRAM
/COMPARE GROUPS
/STATISTICS DESCRIPTIVES
/CINTERVAL 95
/MISSING LISTWISE
/NOTOTAL.

```

## Explore

### Notes

Output Created		
Comments		
Input	Data	
	Active Dataset	DataSet1
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	16
Missing Value Handling	Definition of Missing	User-defined missing values for dependent variables are treated as missing.
	Cases Used	Statistics are based on cases with no missing values for any dependent variable or factor used.
Syntax		EXAMINE VARIABLES=Part1 BY Group /PLOT BOXPLOT HISTOGRAM /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL.
Resources	Processor Time	00:00:03,92
	Elapsed Time	00:00:00,87

[DataSet1]

## Group

### Case Processing Summary

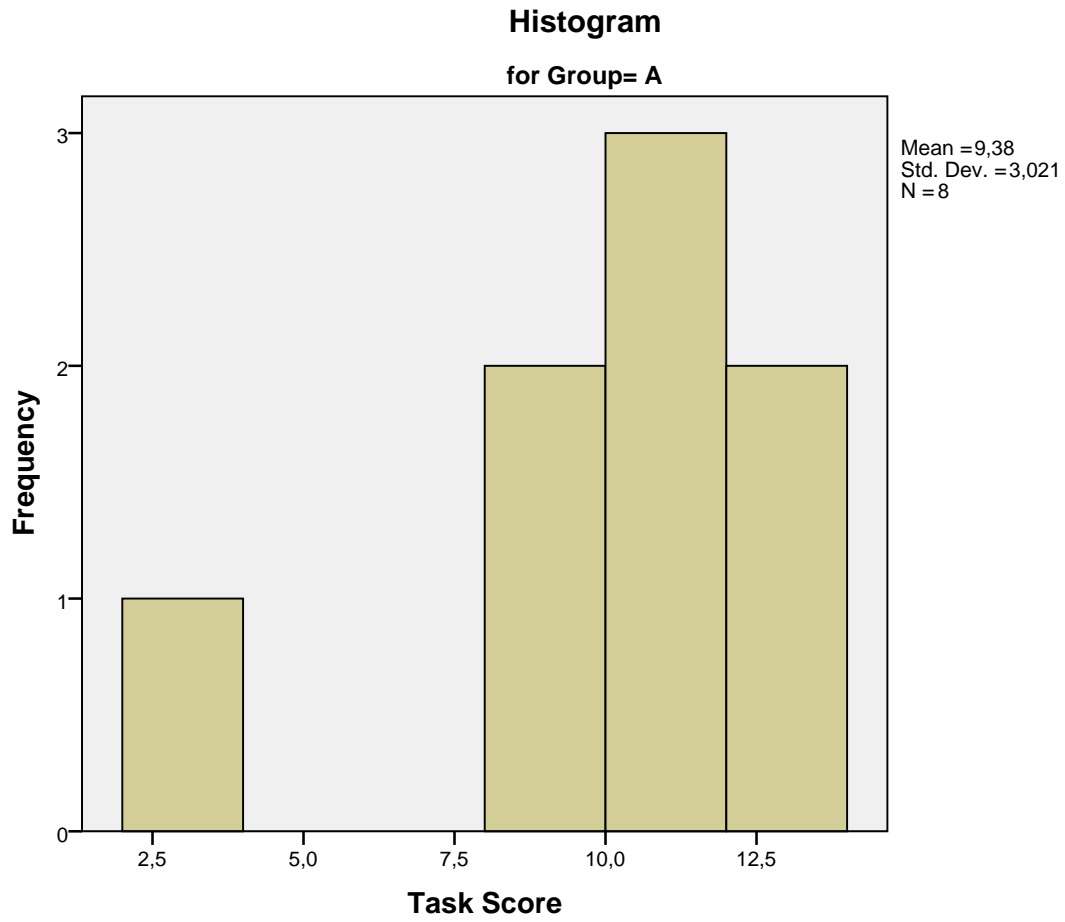
Group		Cases					
		Valid		Missing		Total	
		N	Percent	N	Percent	N	Percent
Task Score	A	8	100,0%	0	0,0%	8	100,0%
	B	8	100,0%	0	0,0%	8	100,0%

### Descriptives

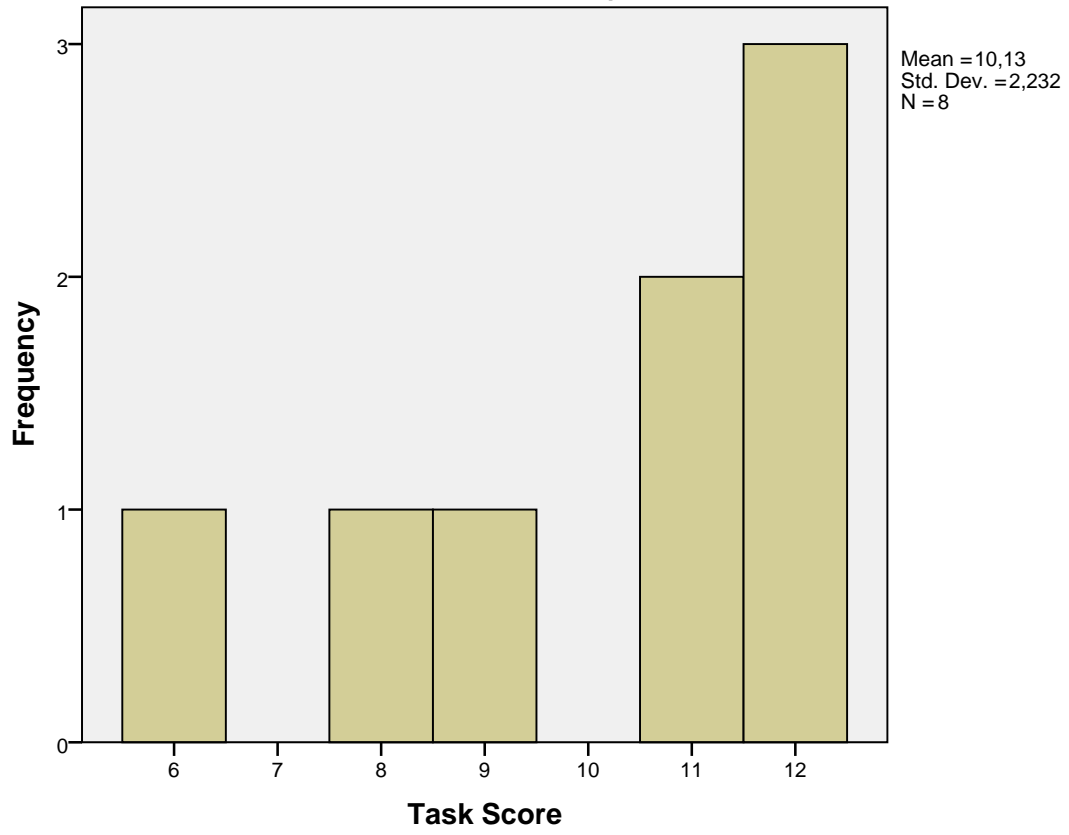
Group		Statistic	Std. Error		
Task Score	A	Mean	9,38	1,068	
		95% Confidence Interval for Mean	Lower Bound	6,85	
			Upper Bound	11,90	
		5% Trimmed Mean	9,58		
		Median	10,50		
		Variance	9,125		
		Std. Deviation	3,021		
		Minimum	3		
		Maximum	12		
		Range	9		
		Interquartile Range	4		
		Skewness	-1,515	,752	
		Kurtosis	2,379	1,481	
			B	Mean	10,13
95% Confidence Interval for Mean	Lower Bound			8,26	
	Upper Bound			11,99	
5% Trimmed Mean	10,25				
Median	11,00				
Variance	4,982				
Std. Deviation	2,232				
Minimum	6				
Maximum	12				
Range	6				
Interquartile Range	4				
Skewness	-1,029			,752	
Kurtosis	-,069			1,481	

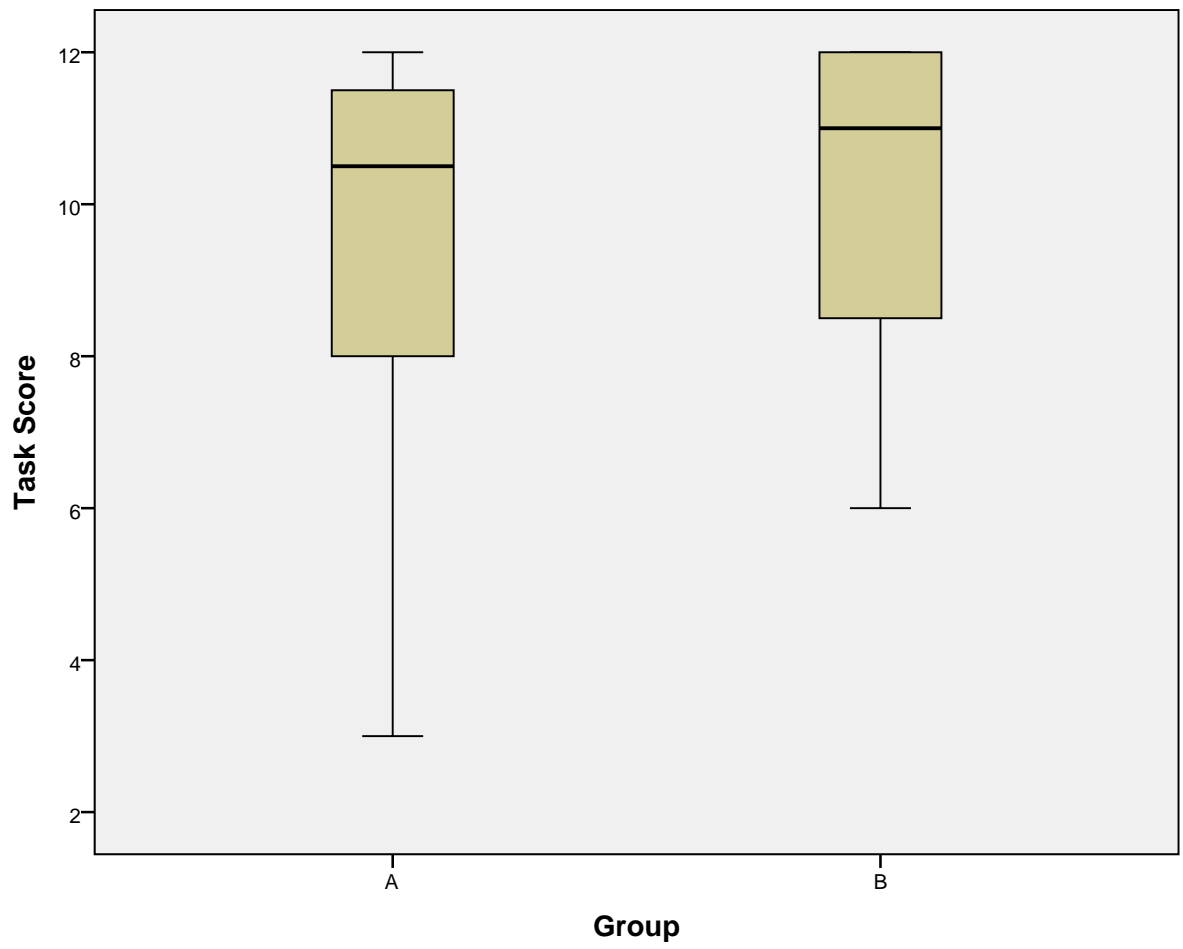
# Task Score

## Histograms



**Histogram**  
for Group= B







## **G.4 Part 1 Score – Without Outlier**

```

/PLOT BOXPLOT HISTOGRAM
/COMPARE GROUPS
/STATISTICS DESCRIPTIVES
/CINTERVAL 95
/MISSING LISTWISE
/NOTOTAL.

```

## Explore

### Notes

Output Created		
Comments		
Input	Data	
	Active Dataset	DataSet1
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	15
Missing Value Handling	Definition of Missing	User-defined missing values for dependent variables are treated as missing.
	Cases Used	Statistics are based on cases with no missing values for any dependent variable or factor used.
Syntax		EXAMINE VARIABLES=Part1 BY Group /PLOT BOXPLOT HISTOGRAM /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL.
Resources	Processor Time	00:00:03,84
	Elapsed Time	00:00:00,85

**Case Processing Summary**

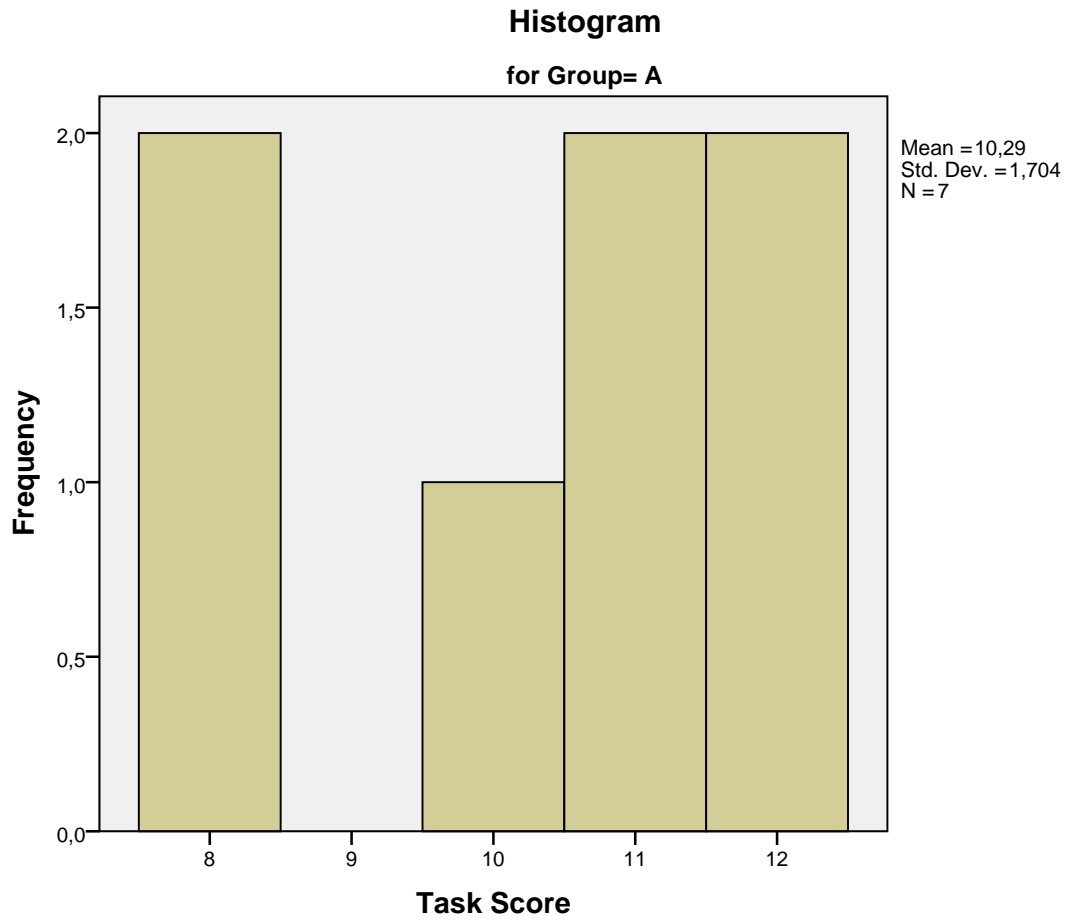
		Cases					
		Valid		Missing		Total	
		N	Percent	N	Percent	N	Percent
Task Score	A	7	100,0%	0	0,0%	7	100,0%
	B	8	100,0%	0	0,0%	8	100,0%

**Descriptives**

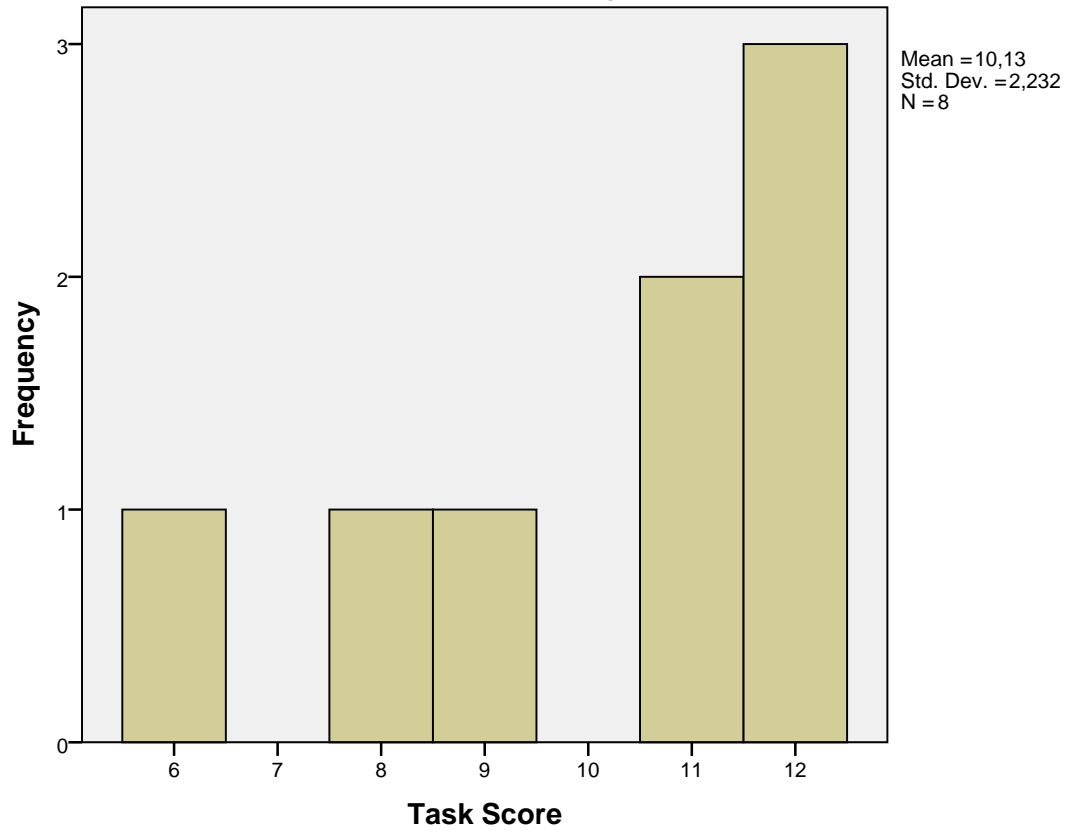
Group		Statistic	Std. Error		
Task Score	A	Mean	10,29	,644	
		95% Confidence Interval for Mean	Lower Bound	8,71	
			Upper Bound	11,86	
		5% Trimmed Mean	10,32		
		Median	11,00		
		Variance	2,905		
		Std. Deviation	1,704		
		Minimum	8		
		Maximum	12		
		Range	4		
		Interquartile Range	4		
		Skewness	-,618	,794	
		Kurtosis	-1,396	1,587	
		B	B	Mean	10,13
95% Confidence Interval for Mean	Lower Bound			8,26	
	Upper Bound			11,99	
5% Trimmed Mean	10,25				
Median	11,00				
Variance	4,982				
Std. Deviation	2,232				
Minimum	6				
Maximum	12				
Range	6				
Interquartile Range	4				
Skewness	-1,029			,752	
Kurtosis	-,069			1,481	

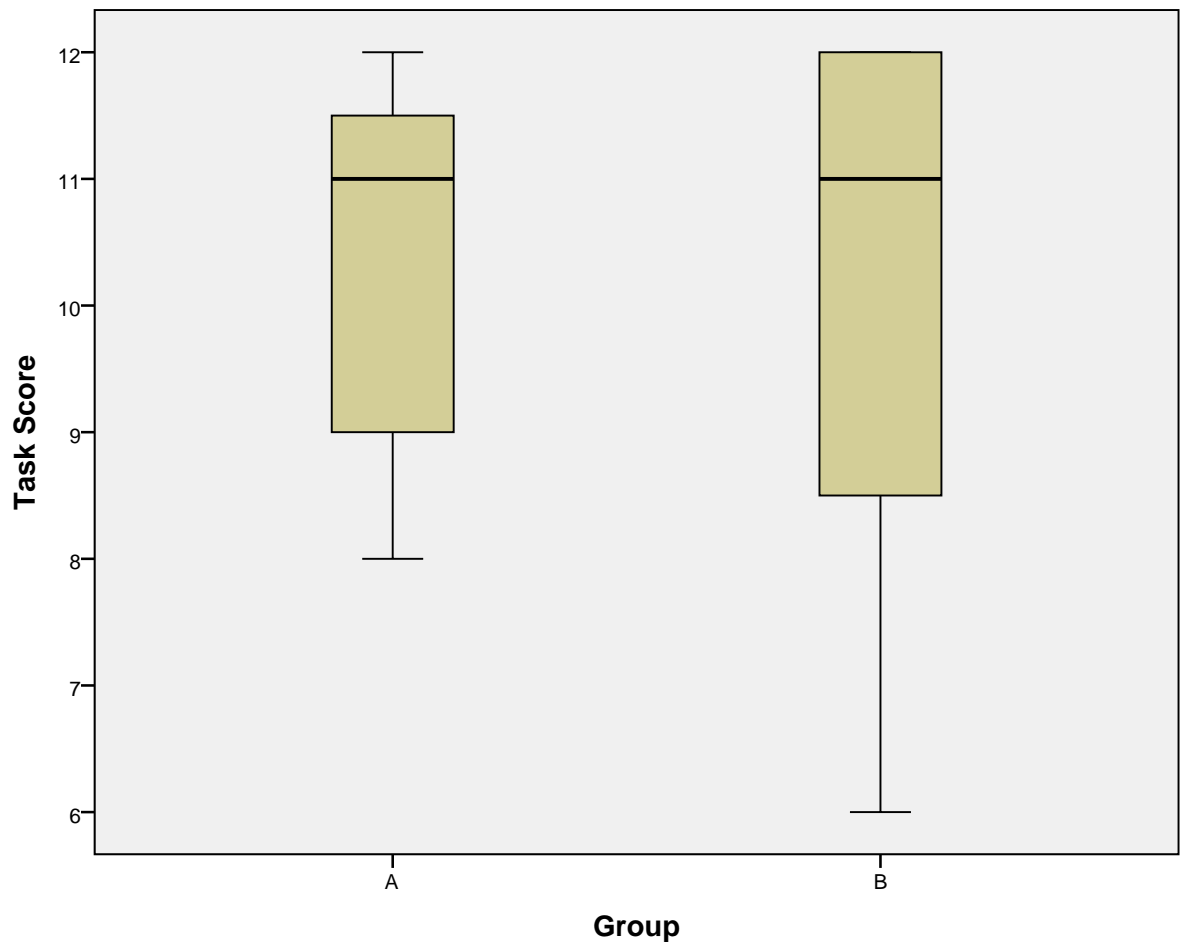
# Task Score

## Histograms



**Histogram**  
for Group= B





## **G.5 Part 2 Score**

```

EXAMINE VARIABLES=Part2 BY Group
/PLOT BOXPLOT HISTOGRAM
/COMPARE GROUPS
/STATISTICS DESCRIPTIVES
/CINTERVAL 95
/MISSING LISTWISE
/NOTOTAL.

```

## Explore

### Notes

Output Created		
Comments		
Input	Data	
	Active Dataset	DataSet1
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	16
Missing Value Handling	Definition of Missing	User-defined missing values for dependent variables are treated as missing.
	Cases Used	Statistics are based on cases with no missing values for any dependent variable or factor used.
Syntax		EXAMINE VARIABLES=Part2 BY Group /PLOT BOXPLOT HISTOGRAM /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL.
Resources	Processor Time	00:00:01,17
	Elapsed Time	00:00:00,36

## Group



**Case Processing Summary**

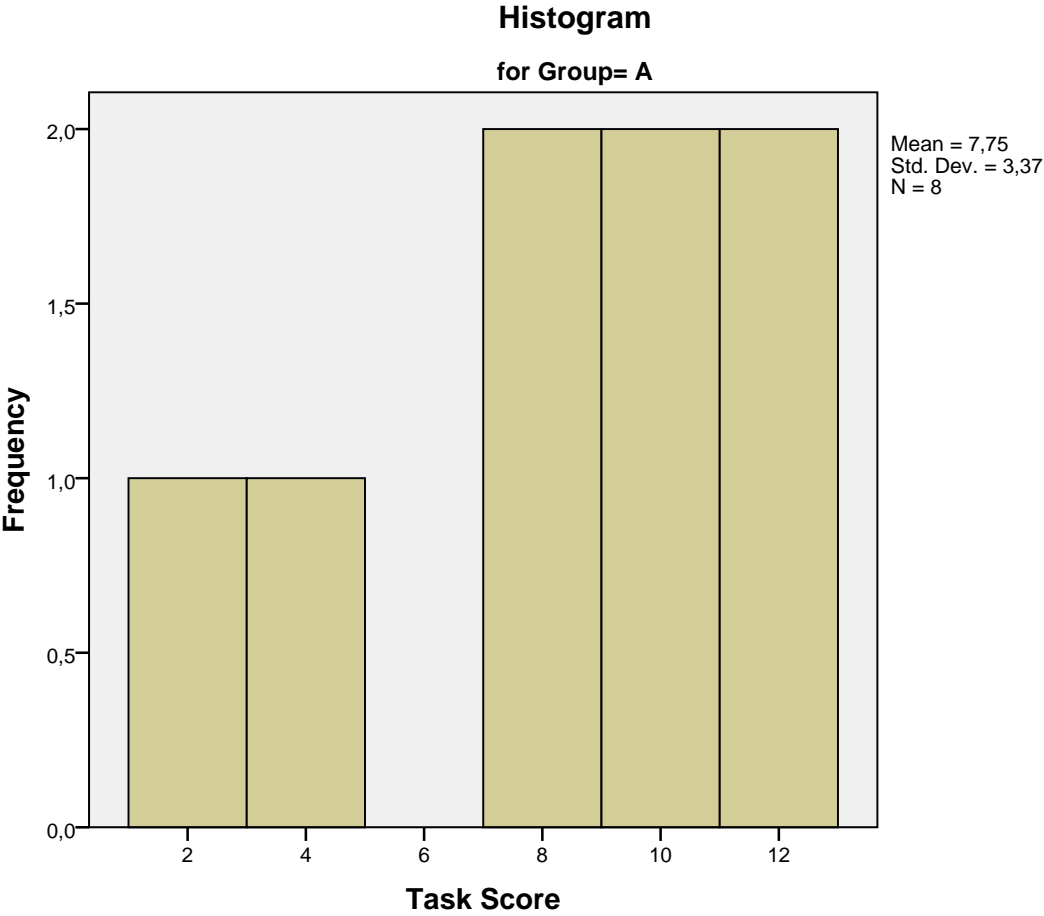
		Cases					
		Valid		Missing		Total	
		N	Percent	N	Percent	N	Percent
Task Score	A	8	100,0%	0	0,0%	8	100,0%
	B	8	100,0%	0	0,0%	8	100,0%

**Descriptives**

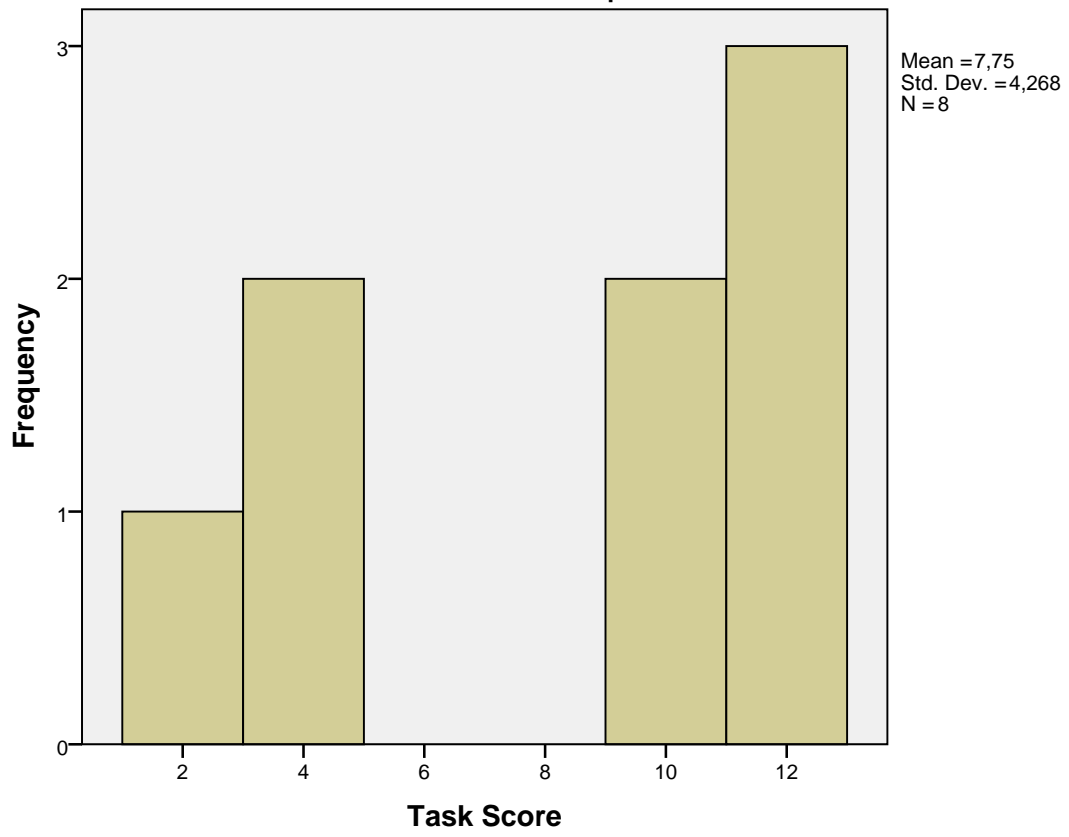
Group		Statistic	Std. Error		
Task Score	A	Mean	7,75	1,191	
		95% Confidence Interval for Mean	Lower Bound	4,93	
			Upper Bound	10,57	
		5% Trimmed Mean	7,83		
		Median	8,50		
		Variance	11,357		
		Std. Deviation	3,370		
		Minimum	2		
		Maximum	12		
		Range	10		
		Interquartile Range	6		
		Skewness	-,638	,752	
		Kurtosis	-,291	1,481	
	B	Mean	7,75	1,509	
		95% Confidence Interval for Mean	Lower Bound	4,18	
			Upper Bound	11,32	
		5% Trimmed Mean	7,83		
		Median	10,00		
		Variance	18,214		
		Std. Deviation	4,268		
		Minimum	2		
		Maximum	12		
		Range	10		
		Interquartile Range	8		
		Skewness	-,579	,752	
		Kurtosis	-2,097	1,481	

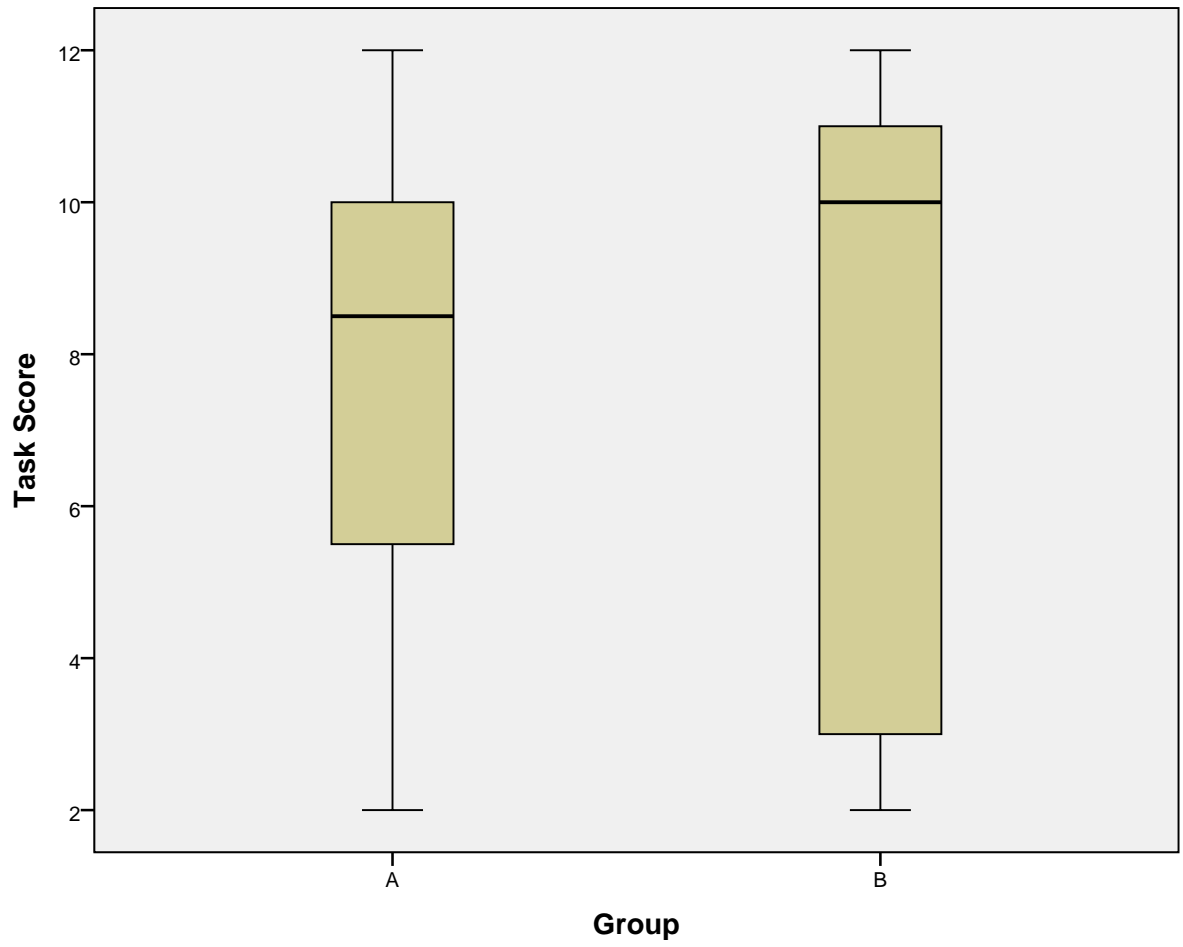
# Task Score

## Histograms



**Histogram**  
for Group= B





## **G.6 Part 2 Score – Without Outlier**

```

EXAMINE VARIABLES=Total BY Group
/PLOT BOXPLOT HISTOGRAM
/COMPARE GROUPS
/STATISTICS DESCRIPTIVES
/CINTERVAL 95
/MISSING LISTWISE
/NOTOTAL.

```

## Explore

### Notes

Output Created		
Comments		
Input	Data	
	Active Dataset	DataSet2
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	15
Missing Value Handling	Definition of Missing	User-defined missing values for dependent variables are treated as missing.
	Cases Used	Statistics are based on cases with no missing values for any dependent variable or factor used.
Syntax		EXAMINE VARIABLES=Total BY Group /PLOT BOXPLOT HISTOGRAM /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL.
Resources	Processor Time	00:00:01,30
	Elapsed Time	00:00:00,48

## Group

**Case Processing Summary**

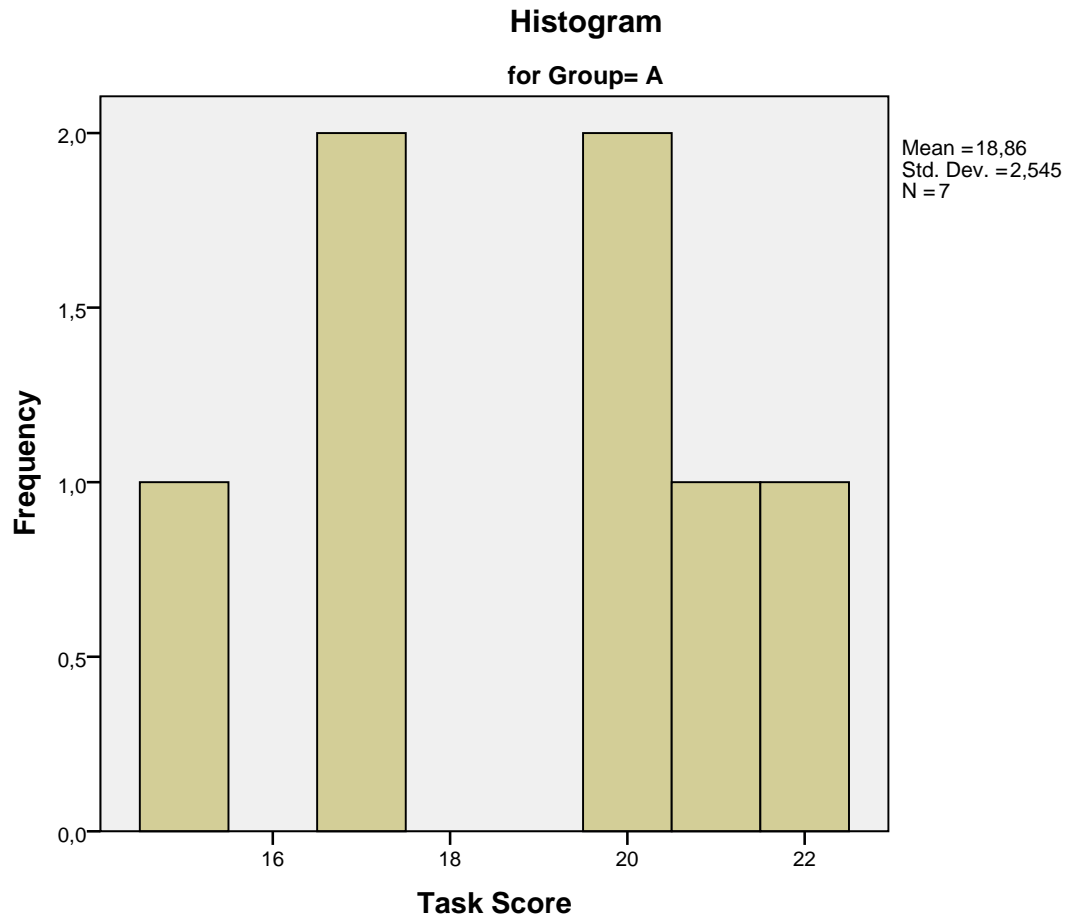
		Cases					
		Valid		Missing		Total	
		N	Percent	N	Percent	N	Percent
Task Score	A	7	100,0%	0	0,0%	7	100,0%
	B	8	100,0%	0	0,0%	8	100,0%

**Descriptives**

Group		Statistic	Std. Error		
Task Score	A	Mean	18,86	,962	
		95% Confidence Interval for Mean	Lower Bound	16,50	
			Upper Bound	21,21	
		5% Trimmed Mean	18,90		
		Median	20,00		
		Variance	6,476		
		Std. Deviation	2,545		
		Minimum	15		
		Maximum	22		
		Range	7		
		Interquartile Range	4		
		Skewness	-,373	,794	
		Kurtosis	-1,314	1,587	
B	B	Mean	17,88	2,074	
		95% Confidence Interval for Mean	Lower Bound	12,97	
			Upper Bound	22,78	
		5% Trimmed Mean	18,08		
		Median	20,50		
		Variance	34,411		
		Std. Deviation	5,866		
		Minimum	8		
		Maximum	24		
		Range	16		
		Interquartile Range	11		
		Skewness	-,827	,752	
		Kurtosis	-,812	1,481	

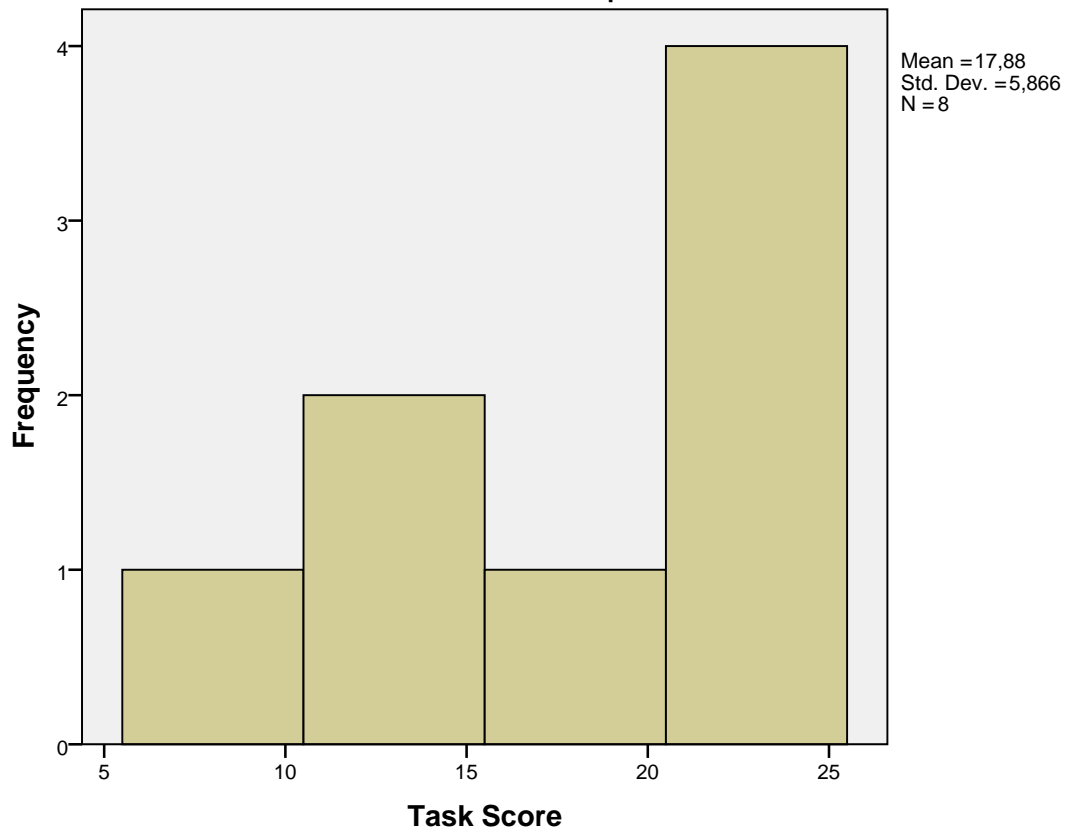
# Task Score

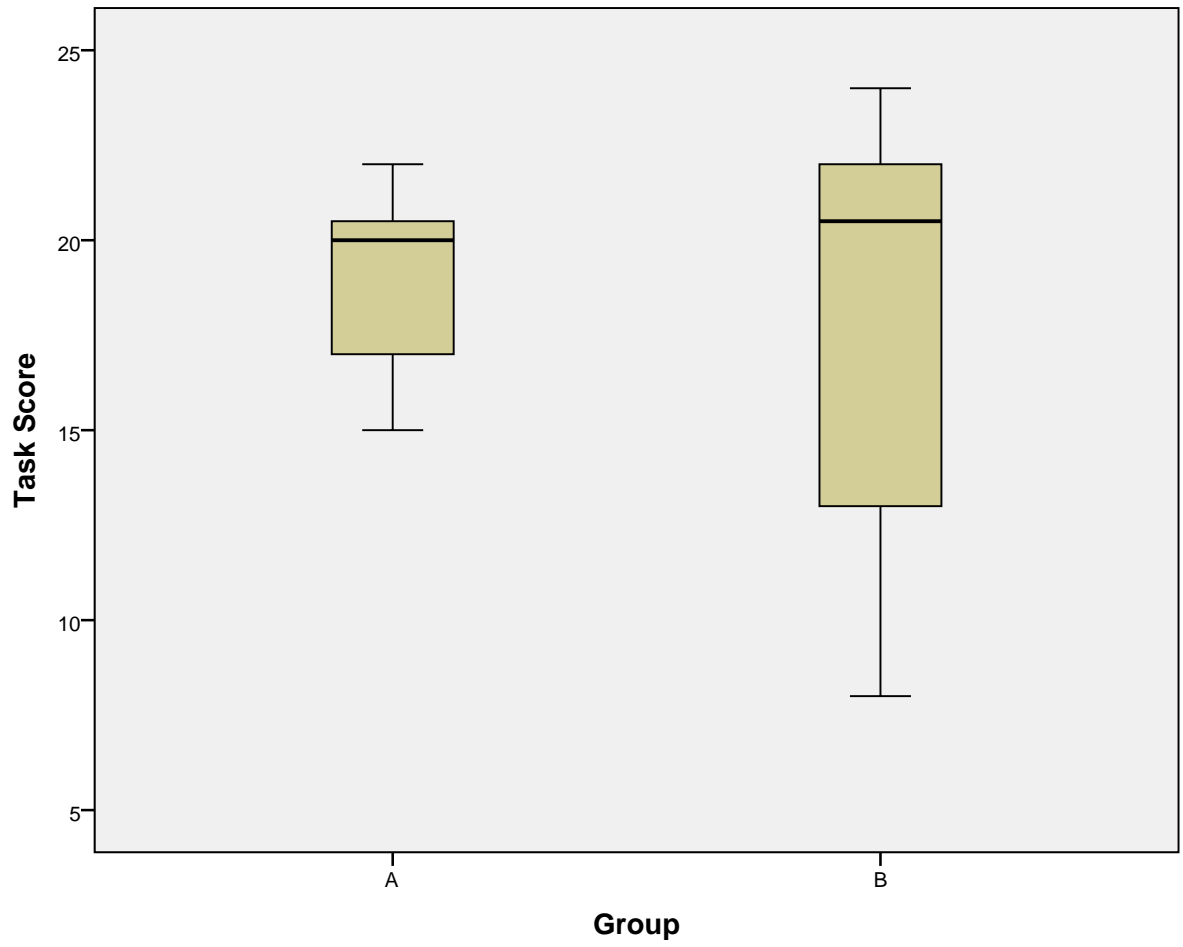
## Histograms





**Histogram**  
for Group= B





## Appendix H

# Example of a CORAL Threat Model Developed using the CORAL Tool

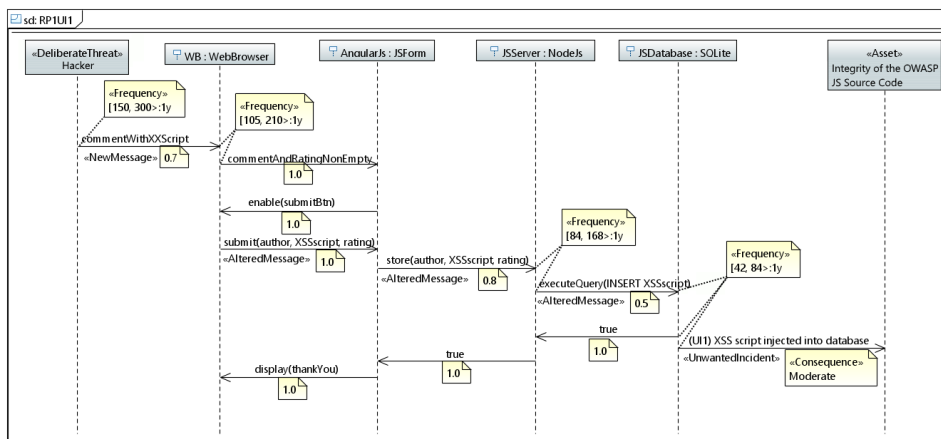


Figure H.1: Example of CORAL threat model in the CORAL tool, depicting an XSS attack toward the feedback feature of a web application for shopping.