TU WIEN Informatics

Towards Concepts and Solutions for Testing High-Security Software Architectures

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Elena Nuiding, BSc
Matrikelnummer 11925876

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Thomas Grechenig

Wien, 20. September 2023

Unterschrift Verfasserin

Unterschrift Betreuung

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

Towards Concepts and Solutions for Testing High-Security Software Architectures

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Software Engineering and Internet Computing

by

Elena Nuiding, BSc
Registration Number 11925876

to the Faculty of Informatics

at the TU Wien

Advisor: Thomas Grechenig

Vienna, 20th September, 2023

Signature Author

Signature Advisor

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Towards Concepts and Solutions for Testing High-Security Software Architectures

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Elena Nuiding, BSc
Matrikelnummer 11925876

ausgeführt am
Institut für Information Systems Engineering
Forschungsbereich Business Informatics
Forschungsgruppe Industrielle Software
der Fakultät für Informatik der Technischen Universität Wien

Betreuung: Thomas Grechenig

Wien, 20. September 2023

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Elena Nuiding, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 20. September 2023

Elena Nuiding

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Besonders möchte ich meinen Eltern, Birgit und Benedikt, danken, die mich während der gesamten Zeit meines Studiums durchgehend unterstützt und bestärkt haben. Ohne euch wäre das nicht möglich gewesen. Einen großen Dank auch an meine Freunde, die mein Leben in Wien so besonders machen. Außerdem möchte ich mich bei allen Interviewpartnern bedanken, die sich Zeit für die Interviews eingeräumt haben. Zu guter Letzt möchte ich mich bei meinen Betreuern für deren stetigen Rat und ihr Feedback während des gesamten Prozesses der Diplomarbeit bedanken.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I especially want to thank my parents, Birgit and Benedikt, who supported and encouraged me throughout my studies. Without you, this would not have been possible. A big thank you also to my friends who make my life in Vienna so special. I would also like to thank all the interviewees who participated in the interviews. Last but not least, I would like to thank my supervisors for their excellent advice and feedback throughout the thesis process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Durch die zunehmende Digitalisierung und einer damit einhergehenden Vernetzung verschiedener Softwaresysteme steigt die Relevanz des Schutzes sensibler Daten immer stärker an, besonders im E-Health-Bereich. Gleichzeitig nimmt die Anzahl an Cybersicherheitsangriffen zu. Daher werden Softwarearchitekturen in einem solchen Bereich durch zahlreiche Sicherheitsmechanismen abgesichert. Entsprechende Architekturen lassen sich auch als High-Security Software Architectures (HSSA) bezeichnen. Es handelt sich dabei um komplexe Softwarearchitekturen mit einer großen Anzahl unterschiedlicher integrierter Komponenten und Services, die über viele unterschiedliche Sicherheitsmechanismen abgesichert sind. Da die Architektur von HSSA sich von der von klassischen Softwarearchitekturen unterscheidet, müssen sie beim Testen anders adressiert werden. Die vorliegende Arbeit untersucht, welche Herausforderungen beim Testen von HSSA bestehen und wie diese in der Praxis adressiert werden.

Zur Identifikation der Herausforderungen beim Testen von HSSA werden semistrukturierte Experteninterviews durchgeführt. Um im Anschluss daran zu eruieren, wie die identifizierten Herausforderungen in der Praxis mit Konzepten adressiert werden, wird eine Case Study mit zwei Fällen mit HSSA aus dem E-Health-Bereich durchgeführt. Abschließend wird die Gültigkeit der identifizierten Konzepte in Bezug auf Übertragbarkeit, Qualität und Validität über semistrukturierte Experteninterviews evaluiert.

Die Ergebnisse der Evaluation zeigen, dass ein Großteil der identifizierten Konzepte beim Testen von HSSA für weitere ähnliche Projekte im HSSA-Umfeld einen Mehrwert darstellen und valide Ansätze bieten, um bestehende Herausforderungen beim Testen von HSSA zu adressieren. Die Bereitstellung der Konzepte unterstützt Softwareunternehmen beim Testen ihrer HSSA. Dies erweist sich beispielsweise bei initialer Etablierung des Testprozesses oder beim Auftreten ähnlicher Herausforderungen zu denen, welche in der Arbeit identifiziert wurden und für die somit bereits geeignete Lösungen vorhanden sind, als vorteilhaft. Die Verwendung der sich in der Praxis als erfolgreich bewährten Konzepte ermöglicht Unternehmen Zeit- und Kosteneinsparungen. Daher sind diese Konzepte geeignet, Unternehmen darin zu unterstützen, ihre HSSA trotz ihrer speziellen Softwarearchitektur testen zu können und somit die Qualität und Sicherheit ihrer Software zu verbessern.

Keywords: *High-Security Software Architectures, Software Testing, Case Study, Experteninterviews, Security*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Due to the increasing interconnectivity of software systems and digitization of data, the need to protect sensitive data is growing, especially in the eHealth sector. At the same time, the number of cyber security attacks is increasing. Therefore, software architectures in such areas are secured by numerous security mechanisms. These architectures can also be referred to as High-Security Software Architectures (HSSA). They are complex software architectures with many different integrated components and services that are secured through various security mechanisms. Because their architecture differs from that of classical software architectures, they must also be addressed differently during testing. Therefore, this thesis investigates which challenges exist in testing HSSA and how these are addressed with concepts in practice.

For this purpose, semi-structured expert interviews are conducted to determine the challenges in testing HSSA. Subsequently, a case study with two cases of HSSA from the eHealth domain is performed to determine how the identified challenges are addressed with concepts in practice. Finally, further semi-structured expert interviews are conducted to evaluate the validity of the identified concepts in terms of their transferability, quality, and their mapping to the respective challenge categories in testing HSSA.

The results of this evaluation have shown that the majority of the identified concepts in testing HSSA also represent added value for similar projects in the HSSA environment and provide valid approaches to address existing challenges in testing HSSA. The knowledge gained can support software companies that develop HSSA by providing them with ideas on how to test their HSSA. This is particularly useful when they start testing their HSSA for the first time or when they encounter similar problems to those identified in the thesis and thus can address them with appropriate concepts. Companies can save both time and costs by utilizing existing concepts rather than developing them from scratch. Moreover, the concepts have already proven successful in existing projects. For this reason, these concepts are suitable to support companies in testing HSSA despite the specific architecture of these systems and, therefore, increasing their software's quality and security.

Keywords: *High-Security Software Architectures, Software Testing, Case Study, Expert Interviews, Security*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

The need to protect personal data has gained significant importance over the past decade, especially in the eHealth sector [1]. Software companies in this sector process, save, and transmit these particularly sensitive digital data [2], [3]. Considerable investments have been made in digitization to achieve better medical diagnoses and improve the accessibility to and quality of health care [4]. At the same time, cyber security attacks have increased sharply in recent years [5]. Healthcare data are particularly attractive to attackers because they contain sensitive individuals' data [6]. When conducting cyber security attacks, attackers attempt to manipulate software or steal information [7]. Such manipulations and information thefts have devastating effects [6]. These pieces of information can be misused, for example, for fraudulent billing, resulting in significant financial losses [5]. However, cyber security attacks are harmful not only to individuals but also to companies. Significant monetary damages often result from extortion, in which the attackers demand ransom to prevent the publication of data or to regain access to software [8]. To address these problems, software architectures must be as secure as possible.

There are various approaches to achieving security in software architectures. Popular approaches are *layering*, *network segmentation and segregation*, *spatial separation from the hardware*, *encryption and security certificates*, *secure communication protocols*, and the usage of *authorization roles*. In the context of this thesis, software architectures that employ a combination of such security mechanisms are referred to as High-Security Software Architectures (HSSA). Furthermore, such architectures are characterized by a high degree of complexity resulting from many integrated services and components.

1.1 Problem Statement and Motivation

Due to the increasing interconnectivity of software systems, digitization of data, and a drastic increase in cyber security attacks at the same time, the importance of HSSA

is growing strongly. Meanwhile, software testing is becoming increasingly important in that context for verifying the correct functionality of software and detecting security vulnerabilities. The architecture of HSSA differs fundamentally from that of traditional software architectures in their complex composition of components and services and the security mechanisms, which are not found in such an extensive form within traditional software architectures [9]. The problem, however, is that general testing approaches such as [10] or [11] do not consider these HSSA-specific characteristics and are therefore only suitable to a limited extent for testing HSSA. The identification of suitable concepts for testing HSSA has not yet been thoroughly investigated in the literature. However, software companies would benefit from such concepts in order to rely on successfully proven approaches to overcome the challenges of testing HSSA.

Before suitable concepts for testing HSSA can be identified, it is necessary to develop a deeper understanding of the specific challenges in testing HSSA. Challenges arise when HSSA are to be tested because high security is contrary to high testability. The characteristics of a software architecture that lead to security lead to poorer testability, and better testability characteristics likewise lead to lower security. Particularly, the aspects of *controllability*, *observability*, *complexity*, *isolability and dependency* are thereby in contention.

It would be advantageous if software companies, for example, from the eHealth sector, could draw upon an existing set of concepts for testing their HSSA without spending time and financial resources developing appropriate concepts. The validity of these concepts would be strengthened if they had already proven successful in practice, indicating that they are applicable in real-world situations and not just in theory.

Therefore, this thesis investigates the challenges of testing HSSA and how they can be addressed with concepts in practice. The main focus is on concepts that address the technical aspects of testing but not the manual or organizational ones. In particular, concepts in testing HSSA from the eHealth area are considered because HSSA from this area require an exceptionally high degree of software testing due to the sensitivity of this kind of data.

1.2 Expected Results

The objective of this thesis is to identify and categorize existing challenges in testing HSSA, both in literature and in practice. Moreover, it aims to investigate how the identified challenge categories are addressed with concepts in practice. In addition, it further intends to evaluate the identified concepts and mapped challenge categories and to derive implications for testing HSSA.

The research questions to be answered are the following:

RQ1: What are the challenges in testing in the context of HSSA and into which categories can the identified challenges be divided?

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

RQ2: How are the identified challenge categories addressed in practice?

RQ3: Which implications for testing HSSA can be derived from the empirical results of the case study?

1.3 Methodology

The methodological procedures for answering the research questions are explained below.

1.3.1 Approach for Answering RQ1

The approach to answering the first research question comprises a theory portion and an interview part. The first step is to deduce a deductive category system for the expert interviews. Therefore, the category system's main categories are derived from two approaches. First, a literature review is conducted to identify known challenges in testing HSSA in theory. Relevant papers are examined, and the most important challenges are extracted from them. Second, those security mechanisms that are in conflict with testability are added as main categories to the category system because security mechanisms limit testability, and therefore, it must be investigated whether individual security mechanisms create challenges when testing HSSA. Then, semi-structured expert interviews are conducted to obtain the experts' experiences of challenges in practice in the previously derived categories. To structure the interview results in a more fine-grained manner, a multilevel deductive-inductive category formation is employed, with the main categories already being present from the previously derived deductive category system and the subcategories being formed from the interview text material by applying Kuckartz's *Content Structuring Content Analysis* [12]. The result from this phase reveals the challenges of testing HSSA, which are classified into different main and subcategories. The final step of this phase is to select the challenge categories that focus on technical aspects and that are deemed the most challenging by the experts, resulting in a filtered deductive category system for the second phase.

1.3.2 Approach for Answering RQ2

The next step is identifying concepts for the challenge categories selected for closer examination. Therefore, a qualitative embedded multiple-case study with two cases is performed to determine how the identified challenge categories of the previous steps are addressed with concepts in practice. The two cases are industrial projects from the eHealth sector. Data triangulation, which is the use of several different data sources, is a suitable method for data collection. In the context of this thesis, informal interviews, code repositories, documents, test case specifications, and confluence articles are used as data sources. The following data analysis involves a within-case analysis of each case separately and a cross-case analysis to identify commonalities and differences between the units of analysis [13]. The result of this phase is a set of concepts that are used in practice to address the challenges of HSSA.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

1.3.3 Approach for Answering RQ3

The next step is to evaluate the identified concepts and mapped challenge categories to be able to provide implications for testing HSSA in general. Therefore, semi-structured expert interviews are conducted. The most relevant generified concepts and mapped challenge categories are presented to the experts. The experts are requested to evaluate the overall quality and transferability of each concept. In addition, they are to assess whether the concepts represent valid approaches for the mapped challenge categories. The questions are both qualitative and quantitative. The qualitative questions are evaluated using Mayring's qualitative content analysis form of *Structuring*, and the quantitative questions are assessed by statistical calculations [14]. The result of this phase is recommendations on concepts for testing HSSA.

1.4 Structure

This thesis analyzes challenges in and concepts for testing HSSA. A theoretical framework is presented to create an understanding of the foundations of software testing and HSSA in Chapter 2. Chapter 3 provides an overview of related work and explains which challenge areas are examined in the expert interviews. Chapter 4 describes challenges in the context of HSSA and the corresponding expert interviews and presents the design of the deductive category system for the case studies. While Chapter 5 addresses the identification of concepts for testing HSSA from the analysis of two case studies, Chapter 6 approaches the generified concepts. Chapter 7 explains the evaluation of the concepts and the mapped challenge categories, and Chapter 8 answers the research questions, describes the limits of the thesis, and proposes future work. The thesis concludes with a summary in Chapter 9.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Foundations

This chapter provides a theoretical framework for understanding the foundations of the areas of *software testing* and *HSSA*.

2.1 Software Testing

This section introduces *software testing* and explains the essential concepts of this subject. It first defines *software testing* in more detail and highlights its goals and purpose. It then presents the *Software Testing Life Cycle (STLC)* and explains the most common classifications of tests according to *test techniques*, *test levels*, and *test types*, followed by an overview of *test automation (TA)*, including the test pyramid. Finally, this section discusses the definition of *testability* and the characteristics of a software architecture that influence testability.

2.1.1 Definition, Goals and Purpose of Software Testing

The International Software Testing Qualifications Board (ISTQB) [15] defines software testing as the following:

"The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation, and evaluation of a component or system and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects."

The Institute of Electrical and Electronics Engineers (IEEE) [16] instead defines software testing in the following way:

"The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component."

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Software testing aims to evaluate the requirements of software. By performing tests, specified software requirements are compared with the actual implementation of the software, and the completeness of the implementation of the expected requirements is determined [17].

The overall goal of software testing is to assess the quality of a test object, which can be a single unit of software, combined units, or an entire system. According to the norm International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 25000 [18], the quality of software is measurable by the following quality attributes: functionality, reliability, portability, security, maintainability, performance efficiency, usability, and compatibility.

An advantage of testing is that it minimizes the risk of software failures that could trigger significant financial damage and harm a company's reputation. In the short term, testing is time-consuming and costly. In the long term, however, it saves money, as fixing bugs at later stages is significantly more expensive. Therefore, tests should be integrated into the development process at an early stage. The earlier testing begins, the sooner erroneous decisions or errors regarding software architecture, scalability, functionality, and security can be discovered. Regarding security, software testing enables the detection of vulnerabilities that could otherwise be potentially exploited. Fewer vulnerabilities increase the probability of protecting sensitive information such as financial or health-related data. Ensuring the security of the customer's data also relates directly to increased customer trust in the product. However, testing can only increase the probability of error detection and cannot prove the absence of errors [19].

Nonetheless, when considering a software test, the costs must always be weighed against the benefits. Exhaustive testing might not be possible due to a lack of time and cost resources [20]. Therefore, the most important test cases must be selected and prioritized so that the most important functionalities can be tested despite the cost, resources, and time constraints [21].

2.1.2 Software Testing Life Cycle (STLC)

Software testing consists of various activities that can be grouped into specific phases, and these phases form the so-called STLC. The STLC consists of seven activities: *test planning*, *test control*, *test analysis*, *test design*, *test implementation*, *test execution*, and *test closure* (shown in Figure 2.1) [19]. Depending on a project's needs, these phases can also be iterated through multiple cycles in a recurring manner. The following description summarizes the individual phases based on [17], [19] and [22].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

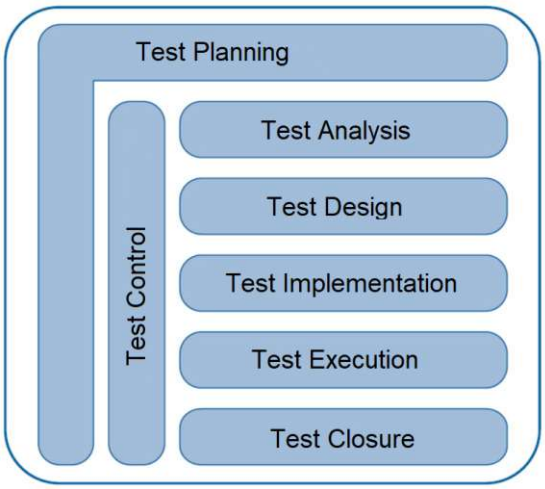

Figure 2.1: Software Testing Life Cycle [19]

Test Planning Test planning is conducted during the entire STLC. During test planning, the required test documents are created. The test activities, levels, and types of tests to be used are also defined in this phase.

Test Control Like test planning, test control must be performed throughout the entire STLC. This activity checks whether the test plan and the associated activities are being followed and allows for tracking and monitoring of the tests' progress. Deviations from the plan are recorded, and measures are determined to reestablish conformity with the plan as soon as they occur.

Test Analysis During test analysis, the test basis is analyzed in detail. The test basis comprises all the artifacts from which the requirements for the software to be tested can be determined. In this phase, the test conditions are identified, and decisions are made regarding which components should be tested and in what order of priority. Bidirectional traceability between requirements and tests must be ensured.

Test Design Both abstract and concrete test cases are designed during the test design process based on the results of the test analysis. The tests should always consider both the positive and the negative cases. Test data, preconditions, and expected results must be defined for each test case. Moreover, the testing infrastructure must be provided.

Test Implementation During test implementation, the final preparation for testing occurs so that the tests can be performed in the subsequent phase. The testing infrastructure and the test cases must now be realized in detail.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Test Execution During test execution, the actual execution of the test cases occurs either manually or automatically. For each test case, the actual result is compared with the expected result. Deviations from the expected result must be recorded and reported in some form so that the development team receives feedback regarding issues to be resolved. Once the developers have made the appropriate changes, the test cases must be repeated to verify that the issues have been corrected.

Test Closure During the test closure phase, a report is prepared in the form of a summary of the test activities that were performed and the test results. The test environment is also archived. Furthermore, a retrospective is performed so that the test process may be improved in the future. Finally, responsibility is transferred to the support and maintenance team.

2.1.3 Test Techniques

One aspect by which software tests can be classified is by test techniques, which can be divided into *static* and *dynamic tests* (shown in Figure 2.2).

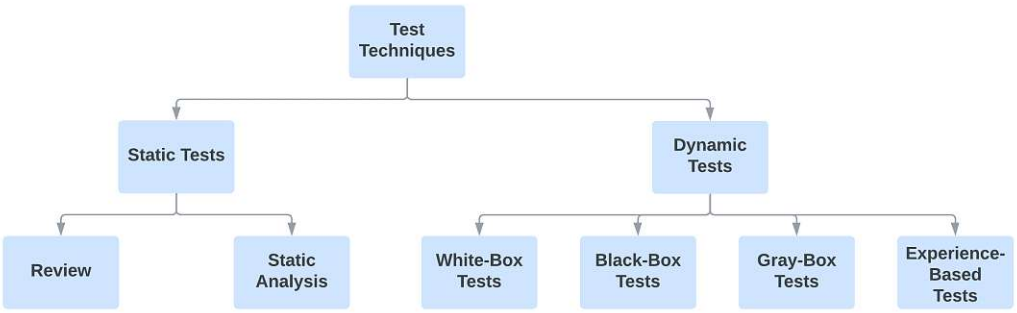

Figure 2.2: Test techniques (based on [19], [23], [24])

Static Testing Static testing is characterized primarily by the fact that it does not involve the execution of code. Rather, the aim is to check work results, which are typically documents or code, from different software development phases [24]. Overall, a distinction is made between reviews and static analysis. While reviews are usually performed manually, static analysis is performed using automated tools. The intention of reviews is to detect errors in documents. Reviews can be divided into four types: informal, walkthrough, peer review, and inspection. Static analysis aims to identify irregularities in static artifacts such as documents or code [25]. It focuses on analyzing the data flow or control flow of code, among other aspects [15].

Dynamic Testing Unlike static testing, dynamic testing involves the execution of code [24]. Dynamic design techniques can be split into the following categories:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- *White-box testing*: This technique tests the internal structure of the test object. Accordingly, the code and the architecture of the test object must be known [23]. The goal is that each portion of the code is executed at least once, resulting in complete coverage of all code portions during the white-box tests [19]. This type of testing is mainly used for unit or regression testing [23]. White-box tests can reveal hidden errors in the code of individual software components, but they do not test the overall completeness of the software [26]. Typical methods of white-box testing are coverage-based methods (statement/ branch/ path condition coverage) [19].

- *Black-box testing*: This technique tests the externally visible behavior of the software [23]. Its focus is on testing the fundamental aspects of the software, such as whether the input is accepted correctly and whether the output is produced correctly [27]. Black-box tests are derived from the product specifications. This test method is mainly used for integration and system testing [28]. Typical methods of black-box testing are equivalence partitioning, boundary value analysis, state transition testing, decision table testing, combinatorial testing, and use case testing [19].

- *Gray-box testing*: This technique combines black-box and white-box testing. When using this technique, part of the code is known, but the respective internal code structure is not known [23]. Some methods of gray-box tests include regression, orthogonal array, pattern, and matrix tests [24].

- *Experience-based testing*: This technique is based on a tester's experience and ability, which grows with work experience. Typical methods of experience-based tests are error guessing, checklist-based testing, attack testing, and exploratory testing [24].

2.1.4 Test Levels

According to the ISTQB glossary [15], a test level is a *"group of test activities that are organized and managed together."* The size of the unit to be tested differs depending on the test level. Distinctions can be made between four levels: *unit*, *integration*, *system*, and *acceptance testing* (shown in Figure 2.3).

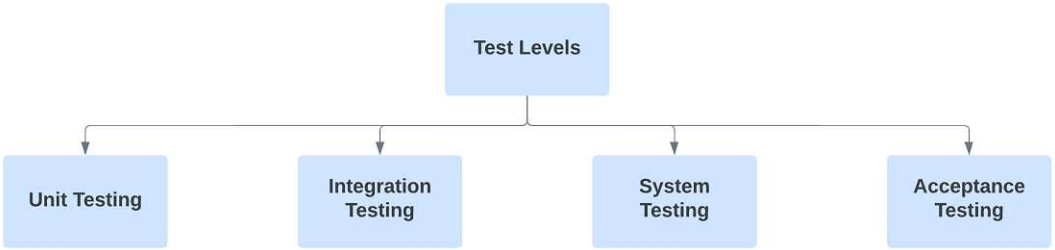

Figure 2.3: Test levels (based on [19], [23], [28])

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Unit Testing Unit testing is defined as *"testing of individual hardware or software units or groups of related units"* [16]. Its goal is to verify that the individual functions of the components function as intended [23]. The respective test object is tested in isolation from the rest of the code. This type of testing can and should be included in the development of the software from the beginning [28].

Integration Testing Integration testing is defined as *"testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them"* [16]. As the name implies, tests at this level focus on the integration and testing of interfaces between components rather than the internal functionality of an individual component [19]. A basic distinction can be made between two different forms of integration testing, namely incremental and big-bang integration testing. The incremental approach involves different numbers of integration stages (depending on the complexity of the software), during which increasing numbers of components are integrated and successively tested in interaction with one another. In the big-bang approach, all components are integrated in a single step and tested as a single unit [28].

System Testing System testing is defined as *"testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements"* [16]. In contrast to unit and integration tests, tests at this level consider the customer's point of view rather than that of the developers' [19]. System tests are normally implemented as black-box tests and are usually performed in a test environment simulating the production environment as realistically as possible [23].

Acceptance Testing Acceptance testing is defined as *"formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the user, customer or other authorized entity to determine whether or not to accept a system"* [16]. This type of testing focuses on the satisfaction of the customers and end users of the software, and these groups are typically involved in the tests. Acceptance tests determine whether the product can be delivered. Thus, they involve testing whether the product complies with the contract and whether the users accept the product as it is [19]. Particular focus is placed on whether the needs and requirements of the end users are met [23]. Acceptance tests, however, should no longer reveal errors as such because these issues should already have been resolved during tests at the previous levels [28].

2.1.5 Test Types

The ISTQB glossary [15] defines a test type as *"a group of test activities based on specific test objectives aimed at specific characteristics of a component or system."* Test types can be classified into *functional*, *non-functional*, *structural*, and *change-related tests* (shown in Figure 2.4) and can be applied to the different test levels discussed above [19].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

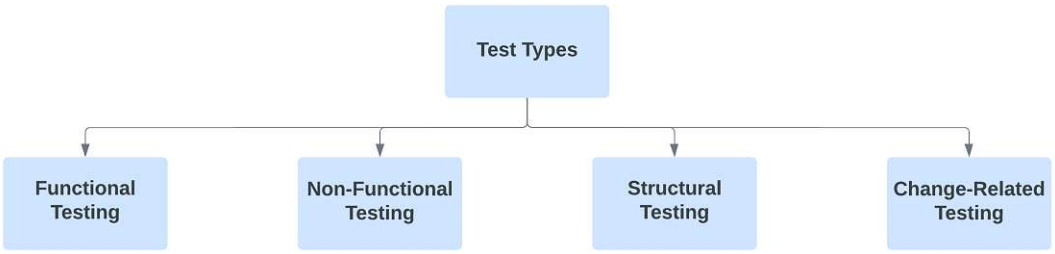

Figure 2.4: Test types (based on [19])

Functional Testing Functional testing focuses on validating the functional requirements of the software being tested. Each function of the software is to be tested by providing input to the function and verifying its output. Black-box tests are most commonly used in this case [19]. Examples of functional tests are technical functional tests, which determine the completeness of the specified functionality, and interoperability tests, which test the interfaces between the individual components of the software [29].

Non-Functional Testing Non-functional testing, as the name implies, tests the non-functional aspects of the software. The focus of this testing is on checking the quality of the software [19], ensuring that non-functional requirements such as reliability, efficiency, usability, modifiability, and transferability are met [29]. Examples of non-functional tests are performance tests, stress tests, and load tests [19].

Structural Testing Structural tests are usually designed as white-box tests, testing the internal structure of the various components or systems. These types of tests ensure that each program instruction performs its intended function. They are mainly used on the lower two test levels (unit and integration levels) as they are usually too time-consuming to perform on the higher test levels [19].

Change-Related Testing Change-related tests are used to verify changes made to the software. They determine whether the changes have been implemented correctly and have not led to any new errors or undesired behavior. Usually, this type of testing is implemented as a confirmation or regression test. A confirmation test is performed after fixing a bug to verify the problem has been eliminated in the new version of the software [20]. Regression tests are used to verify that no new errors have been included in the changed version of a previously functioning piece of software [30]. Regression tests are conducted at all test levels [19]. Because running entire test suites in regression testing is time-consuming and costly, a combination of minimization of redundant tests and appropriate test case selection and prioritization must be used [30]. Confirmation tests are usually executed before regression tests. While confirmation tests primarily test the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

part of the software in which an error has previously occurred, regression tests focus on testing the overall functionality of the software [20].

2.1.6 Test Automation

In this section, TA and its *advantages*, *disadvantages*, and *preconditions* are briefly explained. Moreover, the concept of *Continuous Integration (CI)* and the *test pyramid* are described.

Definition TA is the automation of the test activities to be performed, including the automatic comparison of the expected results with the actual results [17]. In manual testing, the tester usually assumes the role of the user, performing activities that a user typically performs when using the system, verifying them, and uncovering any errors. Automated tests, on the other hand, involve arranging code or scripts for testing and executing them without human intervention [31].

Advantages Automating tests may save time, as the tests are no longer executed manually. This savings is especially evident when running regression tests [17]. In addition, TA may reduce testing costs and increase test reusability [31]. Furthermore, defects may be detected more quickly and frequently, improving product quality. Moreover, TA may reduce execution times, leading to higher test coverage [32].

Disadvantages The introduction of TA may be associated with considerable costs, and there is a risk that the benefits of TA may not outweigh the costs it incurs. Moreover, depending on the project, TA may be too complex to implement based on the knowledge of the testers. In addition, depending on the time frame, the time and resources required to automate the tests may be too scarce [33].

Preconditions Successful introduction of TA requires establishing an automation strategy, including precise planning and a detailed definition of related activities. A further prerequisite for successful implementation is a certain period of time and sufficient financial and human resources [33]. However, not every activity can be automated. Therefore, manual tests are sometimes necessary [32].

Continuous Integration CI is a concept in software development used to increase software quality. Its principles are applied to the daily work of software development teams. To integrate newly developed or modified code into existing software, certain quality standards must be met. For this purpose, pipeline-supported tools are used to examine this quality and permit code integration only with a successful pass [34], [35].

A CI pipeline consists of several stages, for example, build, test, and package stages. The pipeline is executed every time a developer pushes code to the repository. In the test stage, automated tests rather than manual tests are integrated to accelerate the testing process. Overall, TA supports the CI process by ensuring that code is properly tested

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

and that problems are identified at an early stage [36]. Examples of tools that enable the integration of TA into the CI process and allow the creation of pipelines are Jenkins[1], GitLab CI[2], and Bitbucket[3] [36].

Test Pyramid A key concept of TA is the test pyramid introduced by Mike Cohn that shows where and in which order automation efforts should be invested, indicating the types of tests and the frequency with which they are to be performed. The test pyramid consists of the following three layers [37] (shown in Figure 2.5):

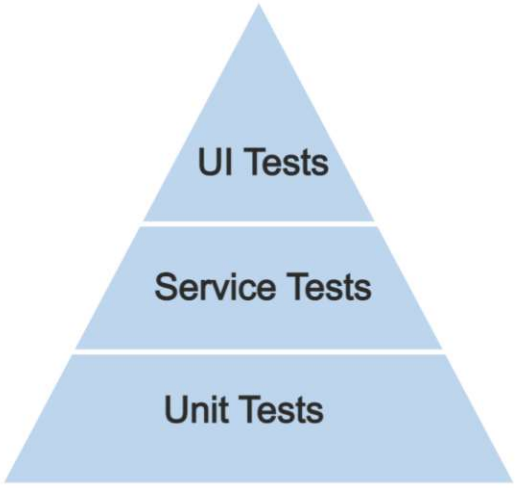

Figure 2.5: Test pyramid (based on [37])

- *Unit tests*: show in detail exactly where the software contains an error.
- *Service tests*: combine Application Interface (API) and integration testing.
- *User Interface (UI) tests*: focus on the end-to-end testing of the software.

The pyramid shows that a large number of unit tests should first be executed in TA. Subsequently, a medium number of service tests should be carried out. Finally, automated UI testing should be performed with the smallest number of tests. The reason for this order is that it only makes sense to perform service tests once the individual functions are working correctly, and UI tests are only rational once the backend of the underlying software has been sufficiently tested [38].

[1] `https://www.jenkins.io/`, Accessed: 22.08.23
[2] `https://docs.gitlab.com/ee/ci/`, Accessed: 22.08.23
[3] `https://bitbucket.org/product/de/features/pipelines/`, Accessed: 22.08.23

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.1.7 Testing Infrastructure

This section defines the term *testing infrastructure*, introduces *static and on-demand test environments*, and explains the concepts of *virtualization*, *containerization*, and *container orchestration*.

Definition The ISTQB glossary [15] defines testing infrastructure as *"the organizational artifacts needed to perform testing, consisting of test environments, test tools, office environment and procedures."* Test environments are the environments in which tests are conducted, containing both software and hardware components [15]. Their purpose is to avoid damage in the production environment when testing. The test environment should closely emulate the production environment but, at the same time, should contain simplifying mechanisms so that testing is as easy as possible [23]. Test environments contain test data, which are generated either manually or automated with test data generation tools [39]. Test tools are designed to assist humans in the execution of tests, thereby achieving consistency, especially in test execution and test results [19].

Static Test Environment Static environments are traditional environments with permanent infrastructure[4]. This approach is mainly used in traditional quality assurance and is used for testing after merging the code[5]. All software is deployed at the end of a release cycle, and all tests are run simultaneously in the same environment. The advantage of permanent test environments is that they only need to be created and configured once. A problem with the permanent infrastructure is that resources are permanently occupied, and the number or size of possible environments is, therefore, limited. If the test environment is damaged, the other testers are also blocked, as all teams test in the same environment[4].

On-Demand Test Environment An on-demand test environment is an independent, fully functional test environment that exists temporarily, can be created quickly, and allows isolated testing[6]. This approach is mainly used when testing is to be performed before the code is merged[5]. The main advantage of this approach is that the test environment can be quickly started and stopped so that resources are only temporarily blocked. The challenge in this case, especially in a highly integrative system with many software components, is to ensure that all components can be dynamically ramped in an encapsulated manner[4]. On-demand test environments can be established by employing concepts like virtualization, containerization and orchestration using technologies such as

[4] https://www.qovery.com/blog/from-static-to-dynamic-environments-why-and-how, Accessed: 22.08.23

[5] https://www.uffizzi.com/blog/test-environments, Accessed: 22.08.23

[6] https://testguild.com/on-demand-environments/, Accessed: 11.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Docker or Kubernetes[7,8].

Virtualization Virtualization is the execution of a virtual instance of a computer system [40]. Using this concept, the hardware resources of a physical machine are partitioned and allocated to Virtual Machines (VMs) [41]. VMs are virtual, isolated instances of a computer system [42]. Each VM possesses its own guest operating system and resources [43]. Between the physical resources and the VMs themselves lies the abstracted intermediate layer of the hypervisor, which is installed on the hardware layer [42]. Hypervisors are used to create, manage, and execute VMs [44].

Containerization Containers are isolated environments that incorporate all necessary elements to run an application, including all dependencies. Therefore, containers are platform-independent and can run on any infrastructure [40]. In contrast to VMs, containerization uses the resources of the same operating system [45]. An example of container-based virtualization of software applications is Docker[9,10](shown in Figure 2.6).

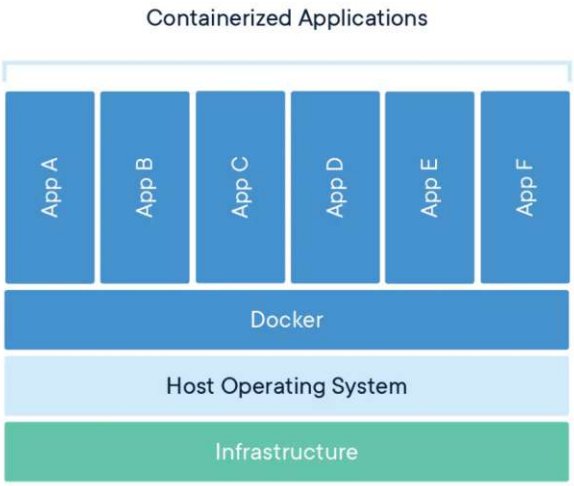

Figure 2.6: Docker[10]

Container Orchestration The term container orchestration refers to the tool-based management and coordination of containers in container-based applications. Processes such as deployment of containers and resource allocation are automated. One such tool is Kubernetes[11], in which the containers run in a cluster [46].

[7] https://www.digitalocean.com/community/tutorials/how-to-configure-a-continuous-integration-testing-environment-with-docker-and-docker-compose-on-ubuntu-14-04, Accessed: 22.08.2023

[8] https://blog.getambassador.io/kubernetes-development-environments-from-local-to-remote-4e33131147c6, Accessed: 15.08.2023

[9] https://www.docker.com/, Accessed: 21.08.2023

[10] https://www.docker.com/resources/what-container/, Accessed: 22.08.2023

[11] https://kubernetes.io/, Accessed: 21.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.1.8 Software Testability

This section defines the term *testability* and explains the characteristics that affect the testability of software.

Definition Numerous different definitions of software testability exist. ISO/ IEC 25010 [47] defines software testability as *"degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met."* Garousi et al. [48], in comparison, define software testability as *"the degree to which a software system or component facilitates the establishment of test criteria, generation, and execution of test cases, and also the probability that a piece of software will fail during testing if it includes a fault."* Overall, all definitions have in common that testability is the extent to which the characteristics of a software allow for testing. A high degree of testability indicates the possibility of testing the system quickly and easily. On the other hand, a low degree of testability means that testing is more complex, which in turn leads to higher time and cost expenditures [49]. As a result, software companies aim for a high degree of testability to save time and money [48].

Characteristics According to Garousi et al. [48], among the attributes of software that most influence software testability are the following (in descending order):

- *Observability*: Observability is the degree to which the behavior of a program can be observed [48]. For example, should an error occur during testing, observability allows the state of the test object to be examined more closely [50].
- *Controllability*: Controllability refers to the extent to which a component can be brought to the desired state for testing. A distinction must be made between the controllability of the test environment on the one hand and the controllability of the state of the test object on the other [48].
- *Complexity (and therefore also simplicity)*: Complexity refers to the intricacies of software in terms of its functionality, structures, and code. The likelihood of errors and the associated difficulty of testing increase as complexity increases [50].
- *Dependency (cohesion, coupling)*: Cohesion indicates the extent to which the software components are interdependent, while coupling refers to the extent to which the modules of a software product are dependent on each other. With high cohesion, classes and their methods are usually more compact, reducing their complexity and resulting in higher testability. In contrast, high coupling leads to lower testability because of the high dependency between components, making it difficult to test individual components in isolation [48].
- *Understandability*: Understandability refers to the extent to which the software (component) to be tested is self-explanatory. Depending on the level of understanding, tests can be easier or more difficult created [48].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- *Inheritance*: Inheritance indicates that new classes are created based on existing classes in the production code [51]. More complex inheritance hierarchies thus lead to higher dependency and lower testability [48].
- *Availability*: Availability indicates that the software is in a constant state for a longer period of time so that it can perform operations without failures to a certain degree. Testing the software at runtime can limit its availability. Therefore, software with high availability requirements can only be tested at runtime to a limited extent [48].
- *Flexibility*: Flexibility refers to the ability of software to adapt to changes [47]. With highly adaptive, flexible software, the tests are usually also flexible and adaptable [52].
- *Maintainability*: This term refers to the degree of simplicity with which a software component can be modified [47]. High modifiability is usually associated with high testability because the tests for highly modifiable software are often also highly adaptable [53].

Lal and Kumar [50] have identified observability, controllability, complexity, and understandability as essential factors influencing testability.

According to Poston [54], factors that influence the testability of software are the following:

- *Isolability*: Isolability indicates that the object under test can be tested separately from the rest of the software.
- *TA*: TA refers to the possibility of automating a test. A high degree of TA is associated with high testability.

In summary, the above concepts indicate that the highest degree of testability can be achieved when the software is well-controlled, observable, simple, understandable, automatable, and isolable.

2.2 High-Security Software Architectures

This section defines the term *High-Security Software Architectures (HSSA)* and explains the mechanisms that are used for the implementation of security.

2.2.1 Definition

The ISO/IEC/IEEE 42010 standard [55] defines software architectures as the following:

"Fundamental concepts or properties of an entity in its environment and governing principles for the realization and evolution of this entity and its related life cycle processes."

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

A software architecture can be considered as a blueprint that provides instructions regarding how the software is to be implemented in concrete terms [56]. It is composed of various software components and describes their relationships to one another [57].

Forms of software architectures include, for example, Microservices Architecture-Based Applications (MSA) and Internet of Things (IoT) architectures [58], [59].

However, this work focuses on secure software architectures, referred to as HSSA in the context of the thesis. While literature does not offer a precise definition of this term, it can be deduced from the following definitions. Reza and Mazumder [60] describe secure software architectures as software systems that employ a combination of security mechanisms at the architectural level to implement security. Pirker and Lechner [61] mention that security in software architectures is essential in healthcare software. Furthermore, they emphasize that such a secure architecture of health software has a high degree of complexity, often involves many other software systems, and communicates with many different other components.

In the context of this thesis, HSSA is understood as complex software architectures containing a large number of different integrated hardware and software components that combine different security mechanisms at the architecture level to produce the most secure software possible. Such security mechanisms can include, for example, layering, network segmentation and segregation, spatial separation from the hardware, encryption and security certificates, secure communication protocols, and the use of authorization roles [62]–[68]. The various components and services mainly communicate through technical interfaces.

2.2.2 Security Mechanisms

Various mechanisms are used individually or in combination to implement security in software architectures. The security mechanisms may be code-related, as well as hardware-, network- and environment-related. The following section explains the concepts of *layering*, *network segmentation and segregation*, *spatial separation from the hardware*, *encryption and security certificates*, *secure communication protocols*, and *authorization roles* (shown in Figure 2.7).

Layering Layering, also called an n-tier architecture, is a classic software architecture pattern. Each layer is responsible for a certain functionality, a concept also called separation of concerns, as each layer only addresses the functionality of its layer. The software can have many different layers, but the four typical ones that are generally present are the presentation, business, persistence, and database layers. The presentation layer is the UI that allows interaction with the application. The business layer performs the business and domain logic. The persistence layer is responsible for data access, and the database layer is responsible for data storage. Each layer only interacts with the layer directly above or below it, ensuring separation of functionality and responsibility.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

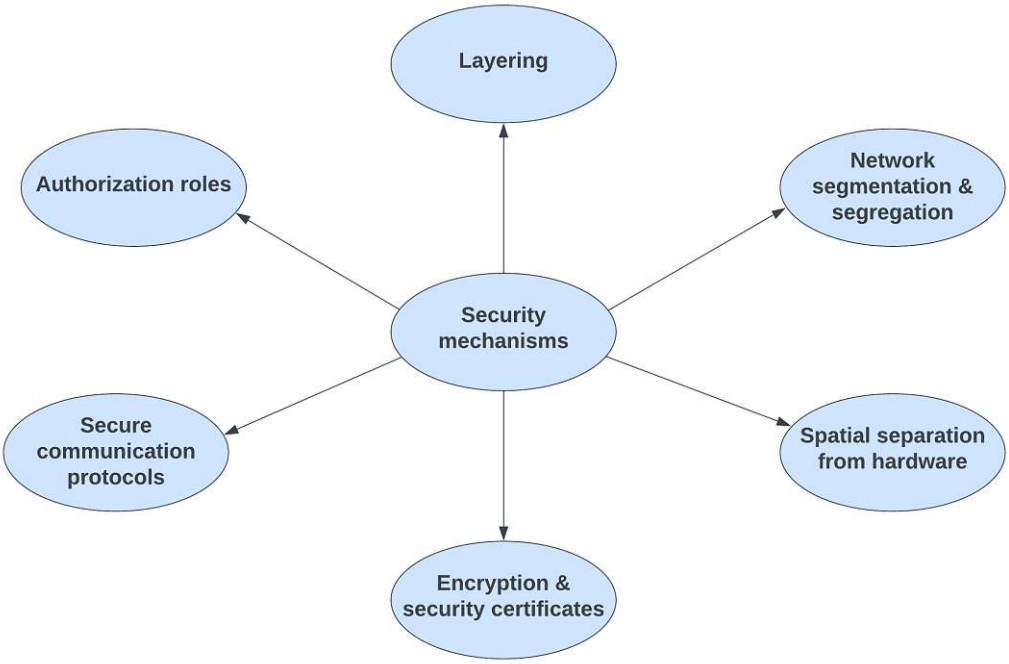

Figure 2.7: Security mechanisms (based on [62]–[68])

The separation of areas of responsibility and the fact that each layer may access only the data in its own area of responsibility ensure stronger security in the software [62].

Network Segmentation and Network Segregation Network segmentation divides a network into different subnets, each operating as a separate network, independent of the other subnets. Network segmentation allows easier monitoring of the data traffic between the individual subnets because the subnets in which the data flow occurs can be precisely localized. Closely related to network segmentation is network segregation, which means that additional rules are added to network segmentation. For example, these rules may take the form of individual user profiles and security policies. Such detailed guidelines allow for precise control of data traffic. In general, critical network elements are separated and compartmentalized from less sensitive networks. This separation can be accomplished by isolation of sensitive resources from an external network, for example. Network segregation is typically achieved through the separation of hardware and data, the use of virtual Local Area Network (LAN)s and private Virtual Local Area Network (VLAN)s, network access control, or the presence of different firewalls (network, host-based, or application firewalls).

In the case of attacks, network segmentation and network segregation ensure that the spread of the attack within the IT system is restricted. For example, malware cannot spread across all subsystems easily in that case. The attack surface is fundamentally smaller due to the smaller segments. Moreover, the attacks can be isolated because

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the individual segments are additionally shielded from one another by firewalls and segregation rules, which make it even more challenging to penetrate the other subnets. Furthermore, network segregation allows for the protection of vulnerable endpoints. Network segmentation and segregation can prevent malicious traffic from reaching vulnerable devices [63].

Spatial Separation from the Hardware As soon as hardware becomes part of an IT system, security can be increased by physically separating it from the software. This separation can be performed, for example, by building data centers with devices that are used for data transmission or storage and placing them in specially guarded rooms to which only certain groups of personnel have access. This access restriction can be implemented by means of a key, for example, thus restricting access to the hardware [64].

Encryption and Security Certificates Encryption is used to protect the visibility of data and messages so that unauthorized persons cannot view them. Data can be encrypted symmetrically, asymmetrically, or through hybrid methods [65], [69]:

- *Symmetric encryption*: A secret key is used for encryption and decryption. This key is known only to the sender and the recipient and must be exchanged secretly [65].
- *Asymmetric encryption*: Each communication partner has a public and a private key. The public key is publicly known and is used to encrypt a message. Only the recipient is aware of the private key which is used to decrypt the message. By using asymmetric encryption, no secret keys need to be exchanged through secret communication [70].
- *Hybrid methods*: These techniques combine symmetric and asymmetric encryption methods [69]. With this approach, communication is usually encrypted with a shared secret key, and the secret key is exchanged through asymmetric encryption. Further communication occurs via the previously exchanged shared secret key [71].

The advantage of asymmetric encryption over symmetric encryption is that the former is significantly more secure. At the same time, however, considerably higher computing effort is required to encrypt messages using asymmetric encryption [65]. Hybrid encryption uses the advantages of both methods and is, therefore, usually more efficient than symmetric and asymmetric methods [69].

Encryption is often linked with digital certificates, which can be seen as a form of digital ID card. A digital certificate typically contains information such as the issuer, the validity period, the issue date, and the public key and is only valid for a limited period of time [72], [73]. The most commonly used standard for issuing digital certificates is the X.509 standard [73]. A digital certificate can be used to verify the identity of a person [74]. Furthermore, the functionality of digital certificates includes the creation of digital

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

signatures and the encryption of data transfers [75]. Digital certificates aim to protect sensitive data or content and to prevent data misuse and identity theft [74], [75].

A key concept related to digital certificates is a Public Key Infrastructure (PKI). Through this infrastructure, it is possible to issue and manage digital certificates [76]. A central component in this system is the Certificate Authority (CA), which is responsible for issuing, verifying, and signing the certificates [73]. The validity status of a certificate can be queried from a responder via protocols such as the Online Certificate Status Protocol (OCSP) [77].

Digital certificates can be used virtually. An example of virtual use involves, for example, Transport Layer Security (TLS) certificates, which are used for secure data transfer between two systems. Encryption algorithms ensure that no attacker can read or modify the data during transmission [66].

Secure Communication Protocols Communication protocols are rules for data transmission between two or more participants [67]. Common protocols of the application layer of the ISO/OSI layer model include Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), Post Office Protocol Version 3 (POP3), File Transfer Protocol (FTP) [78]. HTTP is mainly used to transmit Hypertext Markup Language (HTML) pages on the Internet. It is a connectionless protocol and works by means of a request-response scheme [67]. SMTP is primarily applied for sending mails between mail servers [78]. The client sends commands and information to the server to execute transactions. The server then executes the corresponding transactions and informs the client regarding the success or failure of the transaction. POP3 is a communication protocol that is used to fetch messages from a mail server [79]. As in SMTP, the client communicates with the server via commands. FTP is mainly used for uploading and downloading files. Two channels are established for an FTP connection: the control channel, which is used on the client side to send commands or receive the corresponding status codes, and the data channel, over which the data files are transferred. To make protocols such as SMTP, POP3, FTP, and HTTP secure, encryption techniques must be used. TLS certificates are most commonly employed [67].

Authorization Roles The role-based access control model was originally introduced in 1992 by Ferraiolo and Kuhn [68]. The fundamental idea behind role-based access control is to manage and control access to files and work processes based on different authorization roles. Each role has different permissions to view, modify, or delete specific data and to perform work processes. However, a role is not bound to a person. One person can assume different authorization roles [80]. An example division is an administrator role versus a basic user role [81].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

State of the Art

This chapter presents related work that addresses issues and challenges in testing in related areas such as IoT and MSA. Based on the findings obtained, areas are identified for which more detailed challenges in testing HSSA are to be investigated in Section 4.

3.1 Related Work

This section summarizes challenges in testing software architectures similar to HSSA.

3.1.1 Paper 1: Internet of Things: Current Challenges in the Quality Assurance and Testing Methods

Because IoT systems differ from traditional software architectures due to their high number of different components, various communication protocols, and physical nature, typical software testing approaches are unsuitable for quality assurance. Therefore, Bures et al. [82] conducted literature research to identify aspects that influence testing techniques and a survey of 10 IT solution providers to determine which aspects of quality assurance were perceived as particularly challenging. Based on the problems identified, the authors were able to derive requirements for testing IoT systems.

Some IoT aspects that significantly influence testing techniques, according to these authors' literature research, are described in the following paragraph. The installation of IoT devices in public areas is considered problematic. The main problem is that devices that collect data are often easily accessible to attackers and can be manipulated or exploited by them. Furthermore, the possibilities for updating devices are often limited, leading to security gaps that cannot be closed by updates and, thus, potential exploitation. However, if the performance of updates is possible, many different software versions are present on the devices. The high number of different devices with different versions means that an exponential variety of configurations is required. Testing such a

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

setup results in many tests, high associated testing effort, and high testing costs. Another aspect is that certain physical layers, such as network protocols, are no longer tested by the suppliers as in other systems but must be tested by oneself. Moreover, many different communication protocols are used due to the heterogeneity of IoT systems, leading to more tests.

Part of the study was to determine which aspects of quality assurance in IoT solutions are perceived as particularly challenging by industry participants. One of the challenging aspects is testing the interoperability between different components, as hardly any standardization exists for IoT architectures. Furthermore, the number of platform configuration options is perceived as difficult to test due to the heterogeneity of systems. In addition, the integration and associated testing of the interaction of many individual components are seen as challenging aspects.

Based on the previous findings, the authors identified the following requirements for testing. There is a fundamental need to automate tests, especially integration tests. Furthermore, it must be possible to generate test data efficiently. However, implementing these needs is considered problematic due to their complexity and time-consuming nature.

3.1.2 Paper 2: Challenges of Testing Complex IoT Devices and Systems

Gomez and Bajaj [83] investigated which types of tests are used for IoT systems and which challenges are involved when testing such complex systems. Only the challenges, rather than the test types, are summarized in the following since only the former are relevant to this thesis.

According to the authors, the challenges in testing such systems arise mainly from the heterogeneity of the system, standardization issues, security and privacy issues, interoperability complexity, and the test environment. The heterogeneity of the system's composition of hardware and software components leads to a tremendous amount of testing that is needed. One reason for the high number of tests required in this context is the version diversity of each component and the large number of possible combinations of components. Standardization problems exist, for example, with the testing of communication protocols and the associated verification of whether the standards have been maintained. The problem of testing security and privacy arises primarily from the fact that IoT systems often only implement lightweight security mechanisms instead of all-encompassing ones. These minor security functions are implemented to keep battery consumption low. Therefore, this setting must be tested with different approaches from those for traditional IT systems. Regarding interoperability, the challenge arises primarily from the fact that many different layers and combinations of components must be tested. It is also considered difficult to simulate interoperability as well as the setup of the test environment. Due to the complexity of the systems and the high degree of interoperability between individual components and layers, it is challenging to build a testbed or a virtualized environment.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.1.3 Paper 3: Testing Microservices Architecture-Based Applications: A Systematic Mapping Study

Waseem et al. [84] conducted a systematic mapping study of the use of testing practices in MSA by analyzing peer-reviewed literature. This study aimed to determine which test-related challenges occurred in MSA in recent years.

In MSA, each component is an independent modular unit, possibly with its own release cycle. Therefore, multiple versions of each service exist. Due to the agile nature of microservices, the interface of each component often changes from version to version, resulting in the need to adapt the corresponding tests frequently. The more services are part of the application, the more challenging it becomes to test the interaction between them and, therefore, to find an efficient test solution. Another challenge in testing the communication between the services is to verify that the protocol standards are followed. In addition, dependencies occurring between the services are considered demanding, for example, if only two services are actually to be tested as part of an integration test, but the start of a third service is necessary to be able to perform this test. Among the identified challenges in testing is TA. The challenge of TA increases proportionally with the complexity of the system to be tested. Problems arise primarily due to the heterogeneous configuration of the individual microservices. Moreover, there is the risk of deciding on an unsuitable tool for conducting TA in MSA, leading to undesired side effects or even the impossibility of applying TA. Overall, the authors found TA and intercommunication testing to be the most significant challenges in this area.

3.1.4 Further Related Work

Fadhil and Sarhan [85] describe challenges in testing IoT systems, such as standardization problems in testing and the complexity of interoperability tests. In their opinion, testing and identifying security problems are difficult. Therefore, it is necessary to identify solutions for testing the different layers of an IoT system with the added security aspects. Furthermore, they see a challenge in the test environment and testing tools.

Söylemez et al. [86] see challenges in testing MSA with regression testing, as all testing activities must be handled agilely due to the agile nature of MSA. Furthermore, they identify acceptance testing as a challenge due to the maintenance expenses arising from the agile nature of MSA. Moreover, it is difficult to form a test framework due to the difficulty of validating interfaces and integrations between services. They also consider TA to be a fundamental challenge.

3.2 Deductive Category System

The purpose of this section is to deduce a deductive category system for examination of challenges in testing HSSA in the expert interviews. The main categories of the category system are derived from two approaches. First, the most essential challenge categories of the previously analyzed papers are extracted. Second, main categories of

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

security mechanisms are added because of the trade-off between security and testability. Therefore, this section first explains the categories derived from the papers, second the extent to which there are conflicts between security and testability, and third the additional categories added because of these conflicts.

3.2.1 Challenge Categories Derived from Related Work

In the previous section, challenges in testing software architectures such as IoT or MSA were identified from the literature. These areas could also be challenging when testing HSSA as some of its characteristics are similar to those of IoT and MSA. Such similarities are, for example, that the software consists of many individual components with their own release cycles (in HSSA to separate the functionalities more strongly to increase security), that many different communication protocols are present (in HSSA these are additionally secured protocols), and that both hardware and software components are involved that may be located at different locations. The categories to be investigated in terms of challenges in testing HSSA are *spatial separation from the hardware*, *many different secure communication protocols*, *TA*, *interoperability testing*, *testing infrastructure*, and *test data*.

Spatial Separation from the Hardware IoT systems have proven that easy access to hardware presents an easy possibility for attackers to access or modify data. Therefore, hardware should be protected by security measures such as storage in rooms to which only authorized persons have access. Hence, the absence of hardware access must be considered during testing. This raises the question of the resulting challenges.

Many Different Secure Communication Protocols The presence of many different communication protocols in MSA and IoT systems has led to challenges in testing, including increased testing effort, a lack of standardization, and an inability to use tools. These problems could also exist with HSSA. Additional problems could arise because the protocols are secured by mechanisms such as TLS. Therefore, it is necessary to investigate challenges that arise during testing of HSSA due to the large number of different protocols and, moreover, challenges that are added due to the limited visibility resulting from the use of secure instead of non-secure protocols.

Test Automation In MSA and IoT systems, it has been shown that TA is needed to reduce costs and time. At the same time, however, this area can be challenging because, for example, not every aspect can always be automated, and suitable approaches must be selected in these cases. Therefore, the approach to TA in the area of HSSA and the resulting challenges should be investigated.

Interoperability Testing In the case of MSA and IoT systems, it has been shown that interoperability testing between components is crucial to ensure that the interactions between the individual components function as intended. Studies have shown, however,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

that interoperability testing becomes increasingly challenging, especially as the number of components increases, partly because of the many different versions of the interfaces. This raises the question of what challenges are encountered in interoperability testing of HSSA.

Testing Infrastructure In IoT systems, it has been shown that establishing the test environment is a challenge, especially concerning the question of whether all resources can be virtualized. In HSSA, as in IoT, the architecture is more specialized due to the security mechanisms and complexity of integrated services. Therefore, questions arise regarding the possibility of creating a purely virtualized test environment and what other challenges exist when creating the test environment. Because the test environment is part of the testing infrastructure, it is worth approaching the topic from a higher level and directly examining the challenges that affect the testing infrastructure.

Test Data To be able to execute tests, test data are required. The papers that have been presented have shown that it can be challenging to generate test data efficiently. Similarly, due to all of the security mechanisms applied in HSSA, test data may not be provided as easily as usual. More specific configurations and operations may be required for generation and deployment.

3.2.2 Trade-Off between Security and Testability

To make software as secure as possible, security mechanisms such as those mentioned in Section 2.2.2 are used. At the same time, software testing is important for verifying the software's functionality and identifying security gaps. However, security and testability may conflict with one another. The following discusses those characteristics mentioned in Section 2.1.8 that exhibit a potential conflict.

- *Controllability*: In HSSA, only selected authorized persons should be allowed to control the state of the software or the state of the software environment. In testing, however, it is precisely this controllability that should be ensured for every tester. The tester should be able to easily change the status of the test object at any time to be able to conduct the tests.
- *Observability*: Data should not be observable within secure software architectures. In testing, however, the observability and visibility of the data must be ensured because input and output must be controllable during testing.
- *Complexity*: Another point of trade-off is the complexity of software. Security mechanisms add an additional level of complexity to a piece of software. At the same time, this complexity makes testing more time-consuming and difficult.
- *Isolability and dependency*: An important point in this context is the possibility of isolating a part of the software. If a component is to be tested, then it is necessary

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

to perform this process independently from the remaining system. At the same time, however, security mechanisms often create dependencies on trustworthy third-party components, meaning that some parts of the software are not isolable.

Because testing is an indispensable factor in producing high-quality software, ways must be identified to test HSSA despite its security mechanisms.

3.2.3 Challenge Categories Derived from Security Mechanisms

Due to the trade-offs between security and testability, further areas should be examined to determine testing challenges that may arise due to the use of certain security mechanisms. The additional areas included for investigation in the following chapter are *key material and security certificates*, *logical network separation*, and *testing with different authorization roles*.

Key Material and Security Certificates Key material and security certificates change the setup of software by mainly restricting the visibility of data. A resulting issue is to what extent this affects testing if specific data can no longer be viewed. Indeed, many tests check the software for internal functions and structures and require specific data to be visible. In this respect, challenges concerning the tests and the execution of certain test levels could arise. This category addresses the trade-off in the fields of *observability*, *complexity*, *isolability*, and *dependency*.

Logical Network Separation Logical network separation allows data traffic between and within subnets to be closely monitored based on defined rules. The fundamental difference from conditions of no logical network separation is the division into several subnets and these additional rules. Therefore, it should be investigated whether the division into subnets leads to increased testing effort, whether testing different user profiles leads to challenges, and whether challenges arise if an element is to be tested in a subnet containing especially sensitive resources. This category addresses the trade-off in the fields of *observability* and *complexity*.

Testing with Different Authorization Roles Software systems often include many different user roles. Therefore, it is necessary that each role has different access and action rights in order to protect sensitive data. A question, therefore, arises as to how these different authorization roles affect testing and what challenges arise in the process. This category addresses the trade-off in the field of *controllability*.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 4

Conceptual Design

Because previous literature has not thoroughly examined challenges in testing HSSA, the work described in this chapter investigates this context by conducting semi-structured expert interviews. The categories to be examined for challenges in testing HSSA have been specified in the previous chapter. This chapter presents the methodology and the design of the interviews, explains the results of the data analysis (demographic, quantitative, and qualitative), summarizes the resulting challenges, and presents the emerging filtered deductive category system containing main categories and corresponding subcategories of challenges.

4.1 Methodology

This section presents the theory of the methodical procedure by explaining the difference between quantitative and qualitative research, the data collection form *semi-structured expert interviews*, and the data analysis method of *content structuring content analysis*.

4.1.1 Quantitative versus Qualitative Research

In research, a distinction is made between quantitative and qualitative research methods. Table 4.1 shows the main differences between both research approaches. The decision to use one of the two forms depends on the type of research question. Quantitative methods primarily aim at making statistically evaluable and generalizable statements based on the aggregation of numerical data. In contrast, qualitative research aims to explore unknown phenomena and create new theories focusing on individual cases [87].

In the context of this thesis, a combined research approach is chosen. Given the insufficient literature available regarding the challenges of testing HSSA, a qualitative research approach is necessary to answer the first research question RQ1. The open and flexible nature of qualitative data collection allows for capturing new knowledge in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

	Quantitative Research	Qualitative Research
Hypothesis	Testing of hypotheses	Generation of hypotheses
Sample size	Large number of cases	Single individual cases
Measurement size	Variable oriented	Case oriented
Data type	Numeric data	Data requiring interpretation
Goal	Causal explanation	Description, understanding
Analysis	Statistical analyses	Category formation

Table 4.1: Comparison between quantitative and qualitative research (based on [87])

an all-encompassing way. However, to assess the degree of challenge of the individual challenging categories, quantitative questions are employed as well.

4.1.2 Data Collection

A frequently used method to collect qualitative data is through the use of *guideline-based expert interviews* [88]. In this form of semi-structured interview, a guideline with questions is developed in advance. However, the sequence of questions is flexible, and the questions are open [87]. Expert interviews are interviews with specialists in the matter under investigation who have acquired this specialized knowledge through practice and experience. Expert interviews aim mainly to acquire new knowledge [89].

For this thesis, guideline-based expert interviews are deemed suitable to answer the first research question RQ1, related to the challenges in testing HSSA, for the following reasons:

1. *Knowledge generation*: As studies in the literature have hardly investigated the challenges of testing HSSA, new knowledge must be generated, and expert interviews can enable this. Moreover, interviewing experts allows for rapidly generating knowledge that would otherwise require time-consuming observations.
2. *Openness and flexibility*: The openness and flexibility of guideline-based expert interviews can overcome the lack of clarity regarding potential challenges by allowing the investigation of new fields of knowledge that only become apparent during the interviews.
3. *Practical orientation*: Experts have considerable expertise in a particular area due to their employment in that field. They often work in industry and, therefore, have a practical orientation to their field of expertise. Hence, they are suitable as interviewees and for answering the first research question RQ1.

4.1.3 Data Analysis

After conducting the interviews, analysis methods are applied to the experts' answers to structure the textual material and generate new knowledge from it in a targeted manner.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Expert	Abbr.	Role	Interview date	Medium
Expert 1	E1	Software tester with focus on test automation and manual testing	25.10.2022	Jitsi Meet
Expert 2	E2	Technical lead, Software architect	08.11.2022	Jitsi Meet
Expert 3	E3	Software developer with focus on testing, IT infrastructure manager	14.11.2022	Jitsi Meet
Expert 4	E4	Software developer with focus on test automation	23.12.2022	Jitsi Meet

Table 4.2: Overview of experts (challenge interviews)

One possible analysis method is *content structuring content analysis* by Kuckartz, a category-guided text analysis. It identifies topics and systematizes them by applying a multilevel deductive-inductive category-building process. In inductive category formation, the category formation is established from the interview material after conducting the interviews. In deductive category formation, the category system is already in place before the interviews are conducted and is derived in advance based on literature or existing studies [12]. In the present thesis, this form of analysis is suitable because the main categories have already been derived from the theory framework, but due to the large volume of interview material, the main categories need to be more finely differentiated into subcategories.

In the first coding process, the text material is assigned to the existing main categories. During the second coding process, the entire set of textual material is reprocessed by coding it into subcategories that are created inductively by differentiating the main categories into more finely granular subcategories. Once all text passages are finally coded, the data are further analyzed. The *category-based analysis along the main categories* by Kuckartz is chosen for this thesis. This form of analysis is a descriptive analysis, mainly used to answer the question of what is said about a topic in terms of content. For each main category, the contents of the subcategories should be described. Interpretations may also be made to better explain the relationships within each subcategory [12].

4.2 Interview Design

This section presents the selected interviewees and a summary of the interview guideline.

4.2.1 Selection of Experts

The experts were selected from a large IT company whose projects are in the area of HSSA. All four experts are involved in different projects for the company, and their primary work activity involves testing. Table 4.2 shows their exact roles. The experts were selected in such a manner to include those who are directly involved in test execution and test implementation, as well as experts who are more involved in test management. In this way, the challenges of testing HSSA can be addressed from different perspectives.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The interviews were all conducted via the video messenger Jitsi, and the planned duration for each interview was one hour.

4.2.2 Interview Guideline

An interview guideline was created to ensure the comparability of the expert interviews' results and to structure the content thematically. The guideline questions address the categories of the deductive category system presented in Section 3.2. The questions include demographic questions and content questions. Each content question about the challenges in testing HSSA contains two parts - a quantitative assessment of how challenging the expert considers the area to be - and a qualitative question concerning the explanation of the actual challenges. The complete guideline can be found in Appendix C. A pilot interview was conducted in advance to ensure that all questions were understandable, and the questions were subsequently adapted.

4.3 Demographic Data

The following section presents the results of the demographic questions. Half of the experts are between 25 and 34 years old, and the other half are between 35 and 44. All experts are male. One has a high school diploma, two have a B.Sc., and one has a PhD. All experts work in the industrial area (with half of them working for 5 - 10 years and the other half working for 11 - 20 years). Half of the experts work additionally in the scientific area. The average working experience of the experts in general software development is 15.25 years, in software testing 9.75 years, and in the area of HSSA 4.25 years. Three of the four experts actively work in software testing. On average, they rate their experience concerning TA at 1.75 on a scale of one to four, with one being very experienced and four being not experienced. The individual results of the demographic questions per expert can be found in Appendix B.

4.4 Quantitative Data Analysis

This section presents the results of the quantitative data analysis. Figure 4.1 shows how challenging the experts rate each area concerning testing HSSA on a scale of one to four, with one being very challenging and four not challenging.

Figure 4.2 shows the distribution of the quantitative assessment, including the median values as a boxplot. Statistical evaluation of the median values shows that in the area of HSSA, TA is rated as very challenging. Testing interoperability between different components is rated as challenging to very challenging. Testing when using key material and security certificates, when logical network separation and many different secure communication protocols are present, is rated as challenging. The setup of testing infrastructure, handling of test data, and testing when there is spatial separation from the hardware is rated as rather not challenging to challenging. Testing with different authorization roles is estimated to be rather not challenging in the area of HSSA.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

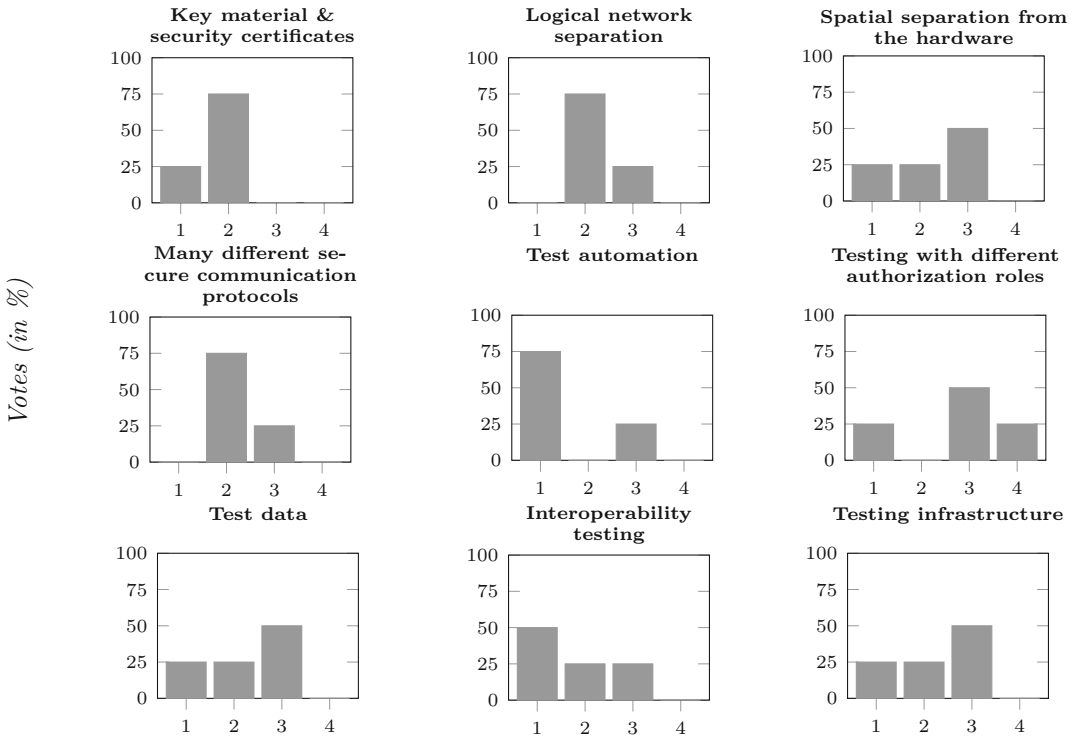

Figure 4.1: Quantitative estimation of the challenge level in terms of testing HSSA

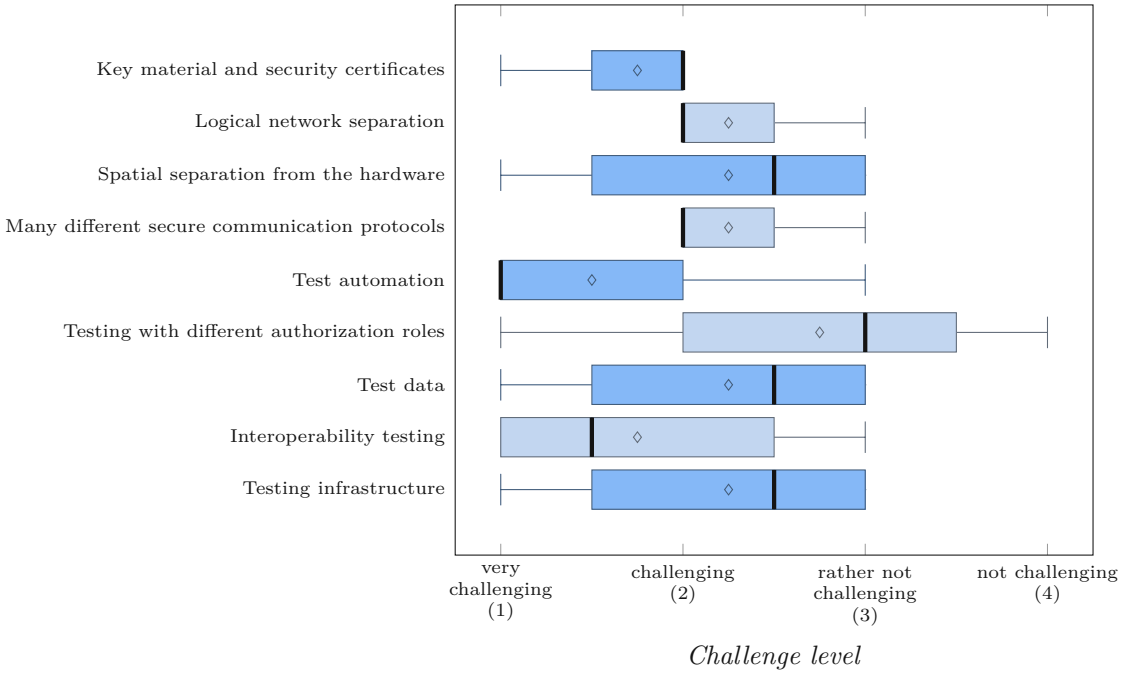

Figure 4.2: Quantitative estimation of the challenge level in testing HSSA boxplot

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

4.5 Qualitative Data Analysis

This section presents the results of the qualitative data analysis of the expert interviews. The deductive main categories of challenges were derived in Chapter 3.2. The corresponding subcategories were formed using the deductive-inductive approach of Kuckartz's *content structuring content analysis*. Subsequent analysis of the coded text segments was performed using the *category-based analysis along the main categories* (presented in Section 4.1.3). Different subcategories could be developed inductively based on the deductively formed main categories. Table 4.3 shows the deductively-inductively developed category system. All categories refer to challenges concerning the testing of HSSA.

4.5.1 Key Material and Security Certificates

The main category of *key material and security certificates* can be divided into the subcategories of *dependency on certificate authority*, *creation and manipulation of certificates*, *PKI*, and *validation of encrypted and decrypted data*.

Dependency on Certificate Authority A challenge arising from using security certificates is the dependency from the Certificate Authority (CA). To be able to perform tests, a CA is required to issue the certificates. This dependency becomes problematic when the test instance of the CA cannot be reached reliably. Such outages block the testing process as the tests cannot be performed without valid certificates [E1].

Creation and Manipulation of Certificates All four experts consider creating and manipulating key material and security certificates for testing HSSA to be challenging.

- *Customer requirements*: The main challenge in self-generated key material and certificates for the tests is that customers often have requirements for creation that contradict the recommendations of recognized organizations such as the National Institute of Standards and Technology. As a result, libraries often do not support these specifications. Thus, the test code must first be extended to support the corresponding generation requirements [E4].

- *Manipulation*: For testing, it is often necessary to manipulate officially obtained certificates [E1], [E3]. The challenge is to implement their use in such a way that the CA still recognizes manipulated certificates as valid [E1].

- *Temporal validity*: Certificates are only valid for a certain period. These temporal aspects are potential sources of error concerning the stability of automated tests. Test pipelines may fail because the certificates have not been exchanged in time or the validity period has been calculated as too short, resulting in maintenance effort [E3].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Main Category	Subcategory
Key material and security certificates	Dependency on certificate authority
	Creation and manipulation of certificates
	Public key infrastructure
	Validation of encrypted and decrypted data
Logical network separation	Dependency on the data center
	Configuration effort
Spatial separation from the hardware	Manual interaction
	Setting up a test station with hardware
Many different secure communication protocols	Test tools and libraries
	Higher complexity arising from transport encryption
Test automation	Initial setup
	Test code architecture
	State controllability
	Resources
	Difference local vs. virtual execution of TA
	Simulation of human interaction
Testing with different authorization roles	Increased number and complexity of tests
	Generalization of test data
Test data	Temporal validity
	Test data management
Interoperability testing	Third-party systems
	Complex environment management
	Increased testing effort
Testing infrastructure	Resource problems
	Hardware
	Configuration complexity
Further challenges	Restricted rights and complicated debugging
	Secure execution environment and storage management
	Library version updates
	Trade-off development and testing status

Table 4.3: Category system with main categories and corresponding subcategories

Public Key Infrastructure If the production code includes an entire PKI, a PKI for testing must also be set up. The challenge in establishing the PKI for the tests is that all aspects, including root certificates, TLS, and the content security policy, must be duplicated from the production code. Even after the initial setup, managing the infrastructure remains challenging, as keys must be issued and revoked continually [E2].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Validation of Encrypted and Decrypted Data Validating encrypted or decrypted data during testing can be challenging. The main challenge is that validation or verification requires to have control over the encryption and decryption functionality and the possession of the key to decrypt or encrypt the data. However, this key is often not possessed [E4].

4.5.2 Logical Network Separation

The main category of *logical network separation* can be divided into the subcategories of *dependency on the data center* and *configuration effort*.

Dependency on the Data Center Logical network separation creates a dependency on the data center during testing when specific tests require obtaining access rights to individual network segments. The data center is responsible for this assignment of rights. Because the assignment of rights is often unclear, this condition frequently leads to delays [E3].

Configuration Effort Some projects require routers for tests to access secure networks (VPN), resulting in a high configuration effort for the computer and hardware [E1]. Both must be configured to interact correctly, as must the software's individual and external components, which are located in other network segments [E3]. However, only two of four experts ([E1] and [E3]) consider the configuration effort to be a challenge in this context.

4.5.3 Spatial Separation from the Hardware

The main category of *spatial separation from the hardware* can be categorized into the subcategories of *manual interaction* and *setting up a test station with the hardware*.

Manual Interaction For tests involving hardware, a challenge with spatial separation is that manual interaction with the hardware is usually necessary at some point [E1], [E2], [E4]. However, manual interaction is not easily possible because the hardware is spatially isolated. There are three options in this case. First, testers can communicate with authorized persons who have access to the hardware and can perform the necessary actions on the devices. However, this option requires planning and time and eventually blocks further test execution [E1]. Second, the hardware can be automated with a second piece of hardware that performs the previously manually performed interactions, but this is usually too complex. Third, all devices can be fully virtualized, which is too time-consuming [E4]. Overall, three of the four experts see a challenge in testing in the manual interaction with the devices.

Setting up a Test Station with Hardware If a test station is set up with the hardware at one location, the testers at other locations must still be able to perform their tests that involve the hardware. The challenges in that context are to make it possible

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

to perform those tests without direct access for the testers but also to set up the test station in such a way that the devices are separated from each other and individually reachable [E3].

4.5.4 Many Different Secure Communication Protocols

The main category of *many different secure communication protocols* contains the subcategories *test tools and libraries*, and *higher complexity arising from transport encryption*.

Test Tools and Libraries One challenge is that using standardized test tools or libraries is challenging due to the heterogeneous setup resulting from many different protocols [E2]. If no library may be used or no suitable library is available to test a specific protocol, the code must be implemented manually [E4]. The degree of difficulty differs depending on the protocol. Especially proprietary protocols or dedicated protocols such as SMTP or POP3 are much more complicated to test than standard protocols like HTTP [E2], [E4]. Adding security to the protocols (e.g., through TLS) increases the challenge because the libraries do not necessarily provide adequate support for this function, or if they do, the behavior of a library differs from protocol to protocol [E4]. This condition implies that implementing the tests for the protocols can be very time-consuming [E4].

Higher Complexity Arising from Transport Encryption Due to the presence of transport encryption, it is more difficult to test the functionality of a communication protocol, including its correct data transmission [E2]. The presence of TLS requires the possession and correct handling of the TLS keys and certificates, which involves additional configuration effort in the test environment [E4].

4.5.5 Test Automation

The main category of *TA* can be divided into the subcategories of *initial setup*, *test code architecture*, *controllability*, *resources*, *difference local vs. virtual execution of TA* and *simulation of human interaction*.

Initial Setup The initial introduction of TA in a project requires configuring the entire setup, including the virtual resources on which the TA runs remotely. The main challenge is to find the proper settings and the configuration of special cases for which the solution is not immediately obvious [E3].

Test Code Architecture HSSA consist of many different components and services. Due to the large number of components and the overall complexity of such software systems, the architecture of the TA becomes particularly complex. Implementing TA can, therefore, involve a large amount of necessary test code, especially because there are also many different test data involved [E1].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

State Controllability Testing HSSA involves testing many different scenarios, especially due to the complexity and numerous security mechanisms of HSSA. Therefore, it is crucial to be able to control the state of the SUT or the components with which it interacts. Thus, it must be possible to produce not only standard states but also especially error and invalid states. This requirement is especially challenging due to the complexity of HSSA [E4].

Resources There are challenges regarding physical and virtual resources.

- *Physical resource problem*: Once hardware is part of a HSSA, full TA requires preparing a large number of devices. In that case, each device contains a different state to be tested. This situation leads to high costs and a spatial problem for the setup [E1], [E3].
- *Virtual resource problem*: This problem arises when only a certain contingent of virtual resources is available due to high resource costs. If too few virtual resources for TA are available, the tests cannot all run in parallel, and therefore the throughput time is very long. Moreover, testers are delayed in their testing process if only one can run the remote pipeline at a time. With HSSA, resources are primarily scarce because testing them involves starting many different components, including resource-intensive processes and simulators [E3].

Difference Local vs. Virtual Execution of TA In some projects, testers can perform automated tests locally in their specific setup and remotely in virtual pipelines. Virtual execution involves steps such as starting servers, runners, and virtual instances, which are not part of the local execution. If errors that did not occur in the local setup occur virtually, it is difficult to reproduce them locally, making it more difficult to understand the cause of the error. In the context of HSSA, common differences between running TA locally versus virtually involve using different certificates or starting resources differently. Another challenge is determining the appropriate settings so that the entire system continues to function correctly after a switch from local TA execution to a virtual form [E3].

Simulation of Human Interaction Simulation of human interaction in TA is challenging because human interactions are complex, and the simulation usually differs significantly from reality in the implementation [E1].

4.5.6 Testing with Different Authorization Roles

The challenges that arise when *testing with different authorization roles* can be categorized into the subcategories of *increased number and complexity of tests* and *generalization of test data*.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Increased Number and Complexity of Tests Each role has different authorization rights within an application. Therefore, tests must be performed from the perspective of each role. Compared to an application in which each role has the same permissions, this condition results in significantly more test cases and thus more test code [E1], [E2], [E4]. Complexity is added in tests with a security focus, which requires verification that the expected role image has been enforced [E2]. While [E1] considers testing with different authorization roles to be very complex, both [E2] and [E4] disagree on that.

Generalization of Test Data Test data should be generalized such that they can be applied to all roles in the same way and under comparable conditions. However, different roles usually have different rights, and therefore, different test data are needed. Hence, it is challenging to generalize the test data as much as possible so that they are valid for as many roles as possible [E3].

4.5.7 Test Data

The main category of *test data* can be categorized into the subcategories of *temporal validity* and *test data management*.

Temporal Validity One challenge with test data in the area of HSSA is temporal validity. For example, if tests are not executed for a long period, the temporal test data can expire, causing the entire TA to fail. The maintenance overhead mostly results from the fact that temporal test data must often be entered manually into the SUT [E1].

Test Data Management Test data management is challenging regarding versioning, modification, generation, import, and clean-up of test data.

- *Versioning*: Test data management involves frequent test data modification or the generation of new test data. However, older software versions might still require old test data. The challenge is to achieve proper versioning so that only the test data that are still necessary remains without creating a massive data repository [E3].
- *Modification*: A challenge in test data management arises when the TA has modified the test data during the test in such a way that a manual maintenance effort is required afterward to restore the original test data [E1].
- *Generation of test data*: Test data can usually be generated automatically. A prerequisite for this generation is that all third-party systems from which the data are obtained are fully virtualized. In some cases, this condition can not be implemented, posing the challenge of manual and time-consuming test data generation [E4].
- *Import and clean-up*: Another challenge is the import of test data and the corresponding environment configuration during test preparation and the clean-up after

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

testing. The clean-up is essential to ensure that no test data remains in the system after the execution of the tests so that further tests can be performed smoothly without any blocking processes [E2].

4.5.8 Interoperability Testing

The main category of *interoperability testing* can be divided into the subcategories of *third-party systems*, *complex environment management*, and *increased testing effort*. While three of the four experts see this category as a challenge, one ([E4]) does not.

Third-Party Systems Testing the interoperability of the SUT with third-party systems creates a dependency that leads to blocking of the test process in cases of outages of the third-party systems. Moreover, there is usually no access to the source code of the third-party systems, making troubleshooting more difficult [E1]

Complex Environment Management When testing interoperability between components of an HSSA, it is necessary to start a large number of different components and mocks [E2], [E3]. The challenge is to retain an overview despite this complexity on the one hand and, to choose the right setup (local or remote), and to determine the required resources on the other [E2]. Moreover, it is challenging that interoperability testing often requires at the network level that components can communicate with each other that usually do not interact [E3].

Increased Testing Effort Components in an HSSA often have different release cycles [E4]. Thus, the interfaces of the individual components change regularly [E3]. If their interoperability is to be tested, a multiplicity of different versions of the components must be tested, resulting in many more combination variants and, thus, a higher number of necessary test cases and test executions [E2], [E4]. This situation has the negative side effect of creating a resource problem because insufficient resources are usually available to test the entire combinatorics of the interoperabilities [E2]. To overcome this overhead, tests must be prioritized and selected to test the most critical interactions [E2], [E3].

4.5.9 Testing Infrastructure

The main category of *testing infrastructure* can be divided into the subcategories of *resource problems*, *hardware*, and *configuration complexity*.

Resource Problems The interaction of the software and hardware components and the provision and maintenance of the testing infrastructure becomes increasingly complex with increasing size and accordingly devours resources. Virtualizing all components is not a solution because it only changes the physical resource problem to a virtual one. Especially complex environments, such as HSSA, require multiple test environments utilizing further resources [E2].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Hardware The testing infrastructure includes the hardware that is required to execute the tests. The challenge is to create a place for the hardware and establish it there in such a way that all functions are reliably accessible and reachable for testing [E1].

Configuration Complexity Difficulties in configuring the testing infrastructure arise when specific non-standard cases must be configured [E3]. Because HSSA involve many different components, many areas exhibit incorrect configurations [E2]. The configuration complexity increases if the SUT requires connections to several external systems. In this case, the testing infrastructure must be configured accordingly [E4].

4.5.10 Further Challenges

The main category of *further challenges* consists of the subcategories of *restricted rights and complicated debugging*, *secure execution environment and storage management*, *library version updates*, and *trade-off development and testing status*.

Restricted Rights and Complicated Debugging The security mechanisms in an HSSA lead to the following restrictions:

- *Restricted log data and more difficult debugging*: Due to security measures, an HSSA is only allowed to release limited information to the outside world, and therefore log output is severely limited. As a result, fault localization and debugging are challenging [E3].
- *Log analysis*: HSSA involves many diverse components. Therefore, it is challenging to analyze and evaluate the log entries of these numerous components because there are so many log files in different places [E3].
- *Restricted rights*: Testers are restricted in testing because they cannot arbitrarily create states and perform actions due to action rights restricted by security mechanisms [E2].

Secure Execution Environment and Storage Management Another challenge in testing HSSA is testing with a secure execution environment and secure storage management. The secure execution context must be encrypted and booted individually for each execution. After each test execution, the execution context must be cleaned up accordingly. To verify that no sensitive information has been left in the main memory after execution, tests must be implemented. The main challenges are how to use, evaluate, and search main memory snapshots of the relevant areas after each test and how to verify that no sensitive data remains in the main memory snapshot [E2].

Library Version Updates A fundamental problem with testing in the area of HSSA concerns library versions. Overall, as many libraries as possible are used to facilitate test code implementation, however, the problem is that these libraries must always

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

be updated to the most recent versions to ensure that the library code complies with the latest security standards. The challenge is to maintain the test code when a new library contains many changes, with many functionalities that no longer work as they did previously. Frequent updates to library dependencies often result in failing test pipelines. In these cases, extensive troubleshooting is necessary to conclude that the library updates have caused the failure [E3].

Trade-Off Development and Testing Status Another challenge is the trade-off between development and testing status. To ensure security, a feature should only be merged when the tests for it are ready. However, because the number of testers is usually less than the number of developers, the testers are often behind schedule [E3].

4.6 Resulting Challenges

Data analysis of the interviews showed that numerous challenges exist when testing HSSA due to the various implemented security mechanisms and the complexity of integrated services and components. The following summarizes the resulting challenges.

- *Key material and security certificates*: Most challenging are the dependency on the CA and the need to successfully obtain and manipulate the certificates, as testing cannot occur without this availability. Moreover, the duplication of a PKI for the tests, as well as the validation of encrypted and decrypted data during testing, is challenging.
- *Logical network separation*: If the data center is not instantly available to assign rights to different subnets, the tests requiring those rights are blocked. In addition, a high level of configuration effort is necessary to access individual subnets during the tests.
- *Spatial separation from the hardware*: If the hardware is spatially separated during the tests, manual interaction is necessary at some point. Moreover, setting up a test station poses the challenge of fully separating the hardware devices from each other.
- *Many different secure communication protocols*: Challenging in this context is that using standardized test tools and libraries is difficult due to the heterogeneous nature of HSSA. Therefore, libraries often must be extended manually. Another challenge is that adding transport encryption to the protocols increases the complexity of testing them.
- *Test automation*: The initial introduction of TA is particularly challenging. Furthermore, creating the desired error and invalid states for the tests is demanding because the controllability of the entire system is required for this process. In addition, resource scarcity, both virtual and physical, poses a challenge. Apart

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

from these factors, the difference between the local and virtual test environments is challenging, as both behave differently, and the test conditions, therefore, differ. Moreover, simulating human actions poses a challenge.

- *Testing with different authorization roles*: Testing from the perspective of each role results in a higher number and complexity of tests. To be able to reuse test data, these test data must be generalized for the different roles. However, this task is demanding due to the different rights that each role has been assigned.

- *Test data*: A further challenge is the generation of temporally valid test data. The challenges in test data management are the achievement of proper versioning of the test data, the generation of test data, importation of test data, and subsequent clean-up after test execution.

- *Interoperability testing*: Testing the interoperability of the SUT with third-party systems creates a potential blocking of the test process in case of outages of the third-party systems. Complex environment management is a particular challenge during interoperability tests of an HSSA. Due to the many different versions of the individual components, higher testing effort is necessary.

- *Testing infrastructure*: Testing with a testing infrastructure involves resource scarcity. Moreover, it is complex to configure the testing infrastructure as HSSA involve many non-standard cases.

- *Further challenges*: Other challenges arise concerning the limited rights of the testers, limited log data, and complex log analysis due to the complex setup. A secure execution environment and secure storage management have also proven challenging during testing. Furthermore, library updates and related changes in the software can lead to failed tests. In addition, there is often a trade-off between the development and testing status,

4.7 Filtered Deductive Category System

Based on the results of the interviews regarding the challenges in testing HSSA, the main and corresponding subcategories of investigation are selected, which are to be investigated in the subsequent case studies. The selection of these categories is performed in two stages. In the first stage, filtering is performed in technical areas, and in the second stage, filtering is performed based on the experts' votes regarding how challenging a main category should be classified.

- *First step*: Because the case studies are intended to address technical areas of testing, the first step is to highlight all subcategories involving technical challenges in testing HSSA (these subcategories are printed in bold in Table 4.4).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- *Second step*: The second step is to ascertain how challenging the experts rated the different main categories on a scale of one to four, with one being the most challenging and four being the least challenging. All main categories that were assessed by at least 50 percent of the experts as rather less challenging (3) or not challenging (4) are excluded from further examination. All main categories that were not deleted are highlighted in gray in Table 4.4.

Table 4.4 shows the resulting filtered deductive category system. The subcategories to be examined in the case study for concepts in testing HSSA have been highlighted in gray and bold.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Main Category	Subcategory
Key material and security certificates	**Dependency on certificate authority** **Creation and manipulation of certificates** **Public key infrastructure** **Validation of encrypted and decrypted data**
Logical network separation	Dependency on the data center
	Configuration effort
Spatial separation from the hardware	Manual interaction
	Setting up a test station with hardware
Many different secure communication protocols	**Test tools and libraries** **Higher complexity arising from transport encryption**
Test automation	**Initial setup** **Test code architecture** **State controllability** **Resources** **Difference local vs. virtual execution of test automation** **Simulation of human interaction**
Testing with different authorization roles	**Increased number and complexity of tests**
	Generalization of test data
Test data	**Temporal validity**
	Test data management
Interoperability testing	**Third-party systems** **Complex environment management** **Increased testing effort**
Testing infrastructure	**Resource problems**
	Hardware
	Configuration complexity
Further challenges	**Restricted rights and complicated debugging** **Secure execution environment and storage management** **Library version updates** Trade-off development and testing status

Table 4.4: Filtered category system

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Case Study

A case study is conducted to determine which concepts are used in testing HSSA in practice. Therefore, the methodology used in the process and the selection of the two studied cases are presented below. Subsequently, the results of the within-case analyses and the subsequent cross-case analysis are explained.

5.1 Methodology

This section presents and explains case study as research methodology, the data collection form, and the data analysis form.

5.1.1 Case Study as a Scientific Method

Yin et al. define a case study as the following [13]:

"A case study is an empirical method that investigates a contemporary phenomenon (the 'case') in depth and within its real-world context, especially when the boundaries between phenomenon and context may not be clearly evident."

Case studies can be conducted either qualitatively or quantitatively, with the qualitative approach being chosen more frequently [90]. According to Yin, case studies can be divided into four different design types. A distinction is made between single- and multiple-case designs, on the one hand, and between a unit of analysis (holistic approach) and multiple units of analysis (embedded approach) on the other. Yin has further distinguished among explanatory, exploratory, and descriptive case studies [13].

In this thesis, a descriptive embedded-multiple case study is deemed suitable to answer the second research question RQ2, how the identified challenge categories are addressed in practice, for the following reasons:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

1. *Type of research question*: The type of research question strongly influences the choice of research method. Case studies are particularly suitable for answering "how" or "why" research questions [13].

2. *Comparability*: The deductive category system allows comparison of the different cases in a structured way in the individual units, making it possible to identify common and differing concepts and thus to better explain interrelationships.

3. *Meaningfulness*: By analyzing several cases, the validity of the concepts is increased, especially if they occur in multiple cases.

5.1.2 Data Collection

Data triangulation, or the use of several different data sources, is a suitable method for data collection [13]. In the context of this thesis, informal interviews, code repositories, confluence articles, test case specifications, and documents are used as data sources. Through data triangulation, a higher validity of the research results can be achieved because the object of study is examined from different angles, thus providing a broader, more objective view [91].

5.1.3 Data Analysis

Figure 5.1 presents the procedure for conducting the case study research and the subsequent data analysis. The methodology suggested by Yin for conducting multiple case studies is as follows. First, a within-case analysis, or the independent detailed investigation, of each case is conducted. A cross-case analysis is conducted next to uncover cross-case differences and similarities. The cross-case analysis thus increases the generalizability of the concepts found [13]. Both within-case and cross-case analysis work with a deductive category system with main categories and subcategories, which has been described in Chapter 4.7.

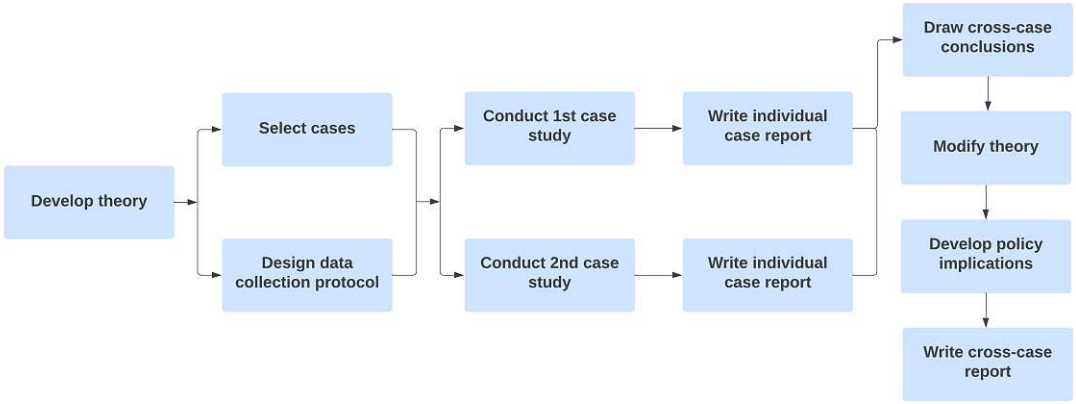

Figure 5.1: Multiple-case study procedure (based on [13])

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5.2 Case Selection

This section presents the motivations behind the selection of the two case studies chosen for the investigation to determine which concepts are used in practice when testing HSSA and summarizes the essential information related to the cases, including technical background information.

Both projects originate in the eHealth sector. Software projects from the eHealth area are particularly suitable for the case study for the following reasons:

- *Data sensitivity*: Health data are particularly worthy of protection due to their sensitivity. This protection serves to ensure the privacy of personal data [92].
- *Data protection laws*: Due to the sensitivity of health data, the security requirements for such data are exceptionally high. Strict legal regulations and laws, such as the General Data Protection Regulation, have been introduced throughout Europe to ensure compliance [92].
- *Requirements compliance*: To ensure that security standards are met, the functional and security-specific requirements that software must fulfill are specified in detail in eHealth projects (such as BSI TR-03161 [93] for example). To guarantee compliance with these requirements, a high degree of software testing is necessary[12].
- *High degree of integration*: Since eHealth systems usually have a high degree of integration with other services as well as other eHealth systems both on a national and international level, they have a high degree of complexity [94].

The reasons that both cases are suitable for investigation are as follows:

- *The projects involve HSSA*: Due to the previously mentioned reasons, numerous security mechanisms are used to make these software architectures secure.
- *Special focus on testing*: In both projects, particular focus is placed on testing to ensure the security of the infrastructure so that the handling of sensitive data is secure. Therefore, the techniques and strategies used to test HSSA despite its security characteristics and complex composition can be investigated.

Table 5.1 provides an overview of the case-specific characteristics of both cases.

5.2.1 Case Summary – Case 1 (C1)

This section describes the first case and provides technical background information.

[12] https://www.qamadness.com/fundamentals-of-ehealth-software-testing/, Accessed: 21.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Case Description The first case is a middle-scale software project from the eHealth sector. It consists of several hardware and software components and focuses on a device that can be used to read sensitive health data and transmit them to other components via a secure infrastructure. The data transfer and communication of the device to most other components runs through one interface. The focus of the TA is on testing the correct implementation of this interface in the software running on the device.

Technical Background Information In this project, Jira's test management tool Zephyr Scale[13] is used to specify test cases, their preconditions, and test execution steps. Tests are run only locally and not remotely. Each tester, therefore, requires an own test device to be able to test the software on it. To be able to execute the automated tests, a tester needs to enter the device-specific data into the TA code. Depending on which state of the software the tester wish to test with the automated tests, they install the relevant software on the device.

	Case 1	**Case 2**
Communication protocols	HTTP (secured via TLS), SSH	HTTP, SMTP, POP3, LDAP, SOAP, CETP ⇒ all secured via TLS
Number of components/ services	13 different components and services	19 different components and services
UI and technical interfaces	UI for configuring the setup and display of status information available; otherwise primarily technical interfaces	UI for configuring the setup available; most of the other services only have technical interfaces
Security mechanisms (relevant for conduction of case study)	Secure communication protocols, PKI, different authorization roles, certificates, key material	Secure communication protocols, PKI, VPN, spatial separation to the hardware, certificates, key material
Examples of used technologies	Java[16], Python[14], Docker[17], OpenSSH[15], GitLab[20], Maven[21]	Java[16], Docker[17], Kubernetes[18], Minikube[19], GitLab[20], Maven[21], Postgres[22]

Table 5.1: Case specific characteristics

[13] https://smartbear.com/test-management/zephyr-scale/, Accessed: 21.08.2023
[14] https://www.python.org/, Accessed: 25.08.2023
[15] https://www.openssh.com/, Accessed: 25.08.2023
[16] https://www.java.com/de/, Accessed: 25.08.2023
[17] https://www.docker.com/, Accessed: 25.08.2023
[18] https://kubernetes.io/, Accessed: 25.08.2023
[19] https://minikube.sigs.k8s.io/docs/, Accessed: 25.08.2023
[20] https://about.gitlab.com/, Accessed: 25.08.2023
[21] https://maven.apache.org/, Accessed: 25.08.2023
[22] https://www.postgresql.org/, Accessed: 25.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.2.2 Case Summary – Case 2 (C2)

This section describes the second case and provides its technical background information.

Case Description The second case is an industrial project from the eHealth sector. It is a large-scale project that is deployed throughout Germany. The project's software is a complex system composed of many independent services and components, each containing a unique functionality. The individual components communicate with each other via different secure communication protocols. The system is deployed in a secure infrastructure in which sensitive health data can be transmitted securely.

Technical Background Information Because this project involves sensitive data, the customer demands evidence of compliance with precisely specified requirements. To comply with this demand, each requirement must be covered by test cases. Therefore, the aim is to achieve a high degree of test automation. Each test implemented in the TA is defined with preconditions and detailed test execution steps in the Zephyr Scale[13] test management tool before implementation. The TA is implemented in a separate GitLab[23] repository, and GitLab CI is used to implement CI. The CI pipelines run in a Kubernetes[24] cluster environment. To be able to run the automated tests not only remotely but also locally, testing is performed locally with Minikube[25].

5.3 Within-Case Analysis – Case 1

This section presents the within-case analysis of the first case.

5.3.1 Key Material and Security Certificates

This section presents concepts identified in testing this HSSA concerning *key material and security certificates*.

Dependency on Certificate Authority During testing, certificates are partially manually obtained once (static approach). The CA must, therefore, be accessible during certificate obtaining but not during test execution. More details are presented below.

Creation and Manipulation of Certificates To obtain valid certificates, two approaches, static and dynamic, are implemented in this case.

- *Static approach*: In the static approach, certificates are obtained once. The requests from the SUT to the CA to fetch the certificates are intercepted via a mock server (for more details regarding the mock server, cf. Section 5.3.3). The tester sends the

[23] `https://about.gitlab.com/de-de/`, Accessed: 21.08.2023
[24] `https://kubernetes.io/`, Accessed: 21.08.2023
[25] `https://minikube.sigs.k8s.io/docs/`, Accessed: 21.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

intercepted request manually via Postman[26] to be able to subsequently read the certificate from the response. The certificate is stored in the TA's project resources and loaded into the system when used in the tests.

- *Dynamic approach*: In the dynamic approach, certificates are obtained from a CA each time a test is performed. Device-specific data are required to send a request to the CA to retrieve a certificate. Therefore, the device is first accessed via SSH to retrieve these data. The request is then completed with this data and sent to the CA, and the response is intercepted via the mock server to be able to read the certificate. This process is fully automated via implemented methods in the TA.

For some tests, the manipulation of the certificates is necessary. Manipulation of the certificates in this context indicates that they become either textually or temporally invalid. Textual manipulation in this context involves adding random bytes to the end of a certificate. For temporal invalidity, the expiration date is manipulated. In both approaches, the procedure is to obtain a valid certificate and then manipulate it. In the static approach, this manipulation is performed manually; in the dynamic approach, it is performed automatically via the test code.

Public Key Infrastructure No concept is identified for this subcategory. Although this aspect is present in the case, it is not tested.

Validation of Encrypted and Decrypted Data No concept is identified for this subcategory. Although this aspect is present in the case, it is not tested.

5.3.2 Many Different Secure Communication Protocols

This section presents concepts identified in testing this HSSA concerning *many different secure communication protocols*.

Test Tools and Libraries In this case, the tests for testing the secure communication protocols are implemented manually, but a supporting library is employed. The used library allows to send requests to the SUT and to receive its responses. However, the requests must be built manually since the library does not provide any supporting functions to that. However, like that, the library also allows to send requests containing errors. In this respect, the protocol can also be tested under misbehavior.

Higher Complexity Arising from Transport Encryption No concept is identified for this subcategory. Although this aspect is present in the case, it is not tested.

5.3.3 Test Automation

This section presents concepts identified in testing this HSSA concerning *test automation*.

[26] `https://www.postman.com/`, Accessed: 27.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Initial Setup In this case, test frameworks with built-in configuration capabilities are used to manage test environments, test data, and other dependencies. The two test frameworks employed are the following:

- *Selenium*: The Selenium[27] framework is used to test the web GUI of the software. The configuration options used are, for example, the logging configurations at which level of detail is logged, as well as prebuilt settings to configure the web drivers for different browsers.

- *JUnit*: The JUnit[28] test framework is employed. This framework offers prebuilt configuration options, thus, the code for this does not need to be implemented manually. The prebuilt configuration options used are, for example, annotations with which specific actions can be executed before and after each test or the use of test suites to combine several test classes that are to be executed together.

Test Code Architecture Parameterized tests are used in the TA. Abstract test scripts are employed, and test data providers are used to control which data are used to execute a test. The test is executed once for each test data set. The library used for this is JUnit[28]. Parameterized tests are used particularly frequently in error scenarios in which numerous different error data sets are available.

State Controllability As already mentioned, the project's central component communicates with most other components via an interface. In the TA, a mock server simulates this interface and the components behind it. The following special features are present:

- *Saving the executed requests*: The mock server (unlike the real component) has an endpoint that stores all executed requests. This endpoint can be queried via the TA (or manually) if necessary to check whether the sent requests, including their values, have arrived at the mock server exactly as desired.

- *Variable configuration of the mock server*: The mock server provides another endpoint that can be used to configure the exact values to be returned for a desired endpoint. When the endpoint is called, the desired response code, return values, and the endpoint to which these values should be supplied are returned. Thus, it is possible to create desired error scenarios and states and normal cases.

- *Encapsulation*: All mock server connections and configuration details are encapsulated in their management classes.

[27] `https://www.selenium.dev/`, Accessed: 22.08.2023
[28] `https://junit.org/junit5/`, Accessed: 21.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Resources A semi-automatic test approach is combined with test groups when testing the hardware. The tests are divided into different groups, whereby the tests of a test group test the device in exactly one state. The states are recorded in tags, and each test is annotated with the state required for the test. The concrete procedure for the actual execution is the manual preparation of the desired state on the hardware, the subsequent selection of a test grouping for which the tests are to be started, and the subsequent automatic execution of the tests. In this approach, the device does not need to be available in large numbers.

Difference Local vs. Virtual Execution of Test Automation No concept is addressed in this context, as the tests are only executed locally and not remotely.

Simulation of Human Interaction No concept is identified for this subcategory. Although this aspect is present in the case, it is not tested.

5.3.4 Interoperability Testing

No concepts for this main category are addressed in this context because no interoperability tests are present.

5.3.5 Further Challenges

This section presents concepts identified in testing this HSSA in relation to *further challenges*.

Restricted Rights and Complicated Debugging

- *Log reporting library*: In this case, a log reporting library is used to examine the logs of the individual test execution steps. This approach is combined with a central listener that allows logging of the specific configurations made for the test execution at the beginning of each test. This is relevant because many different states need to be configured for the device under test.

- *Testing with debug and prod build*: Testing is performed using different software versions. In addition to the product version of the software (prod build), a test version of the software (debug build) is set up to make testing easier for the testers. With the debug build, the tester can more easily perform desired actions or create required states. For example, in the debug state of the software, it is possible to log sensitive data or read out logs in general. SSH access to the hardware is also available so that the device can be manipulated directly to create initial states for the tests without accessing the UI.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Secure Execution Environment and Storage Management Test data are stored in a configuration file. These data are specific to each test device and, therefore, not pushed to the remote repository. Instead, they are entered manually by each tester locally. During test execution, data from the configuration file are loaded into the system and deleted after test execution.

Library Version Updates No concept is identified for this subcategory. Although this aspect is present in the case, it is not tested.

5.4 Within-Case Analysis – Case 2

This section presents the within-case analysis of the second case.

5.4.1 Key Material and Security Certificates

This section presents concepts identified in testing this HSSA concerning *key material and security certificates*.

Dependency on Certificate Authority In this case, static certificates are obtained once from a CA and used for testing. Further details are presented directly below.

Creation and Manipulation of Certificates Valid certificates are obtained once from an authentic CA. To obtain them, a selected person runs a script, as not all testers have access to the root certificate; therefore, most testers cannot create the certificates themselves. The certificates are created with the longest possible validity period so that they do not need to be frequently replaced. Once created, they are stored in a keystore with password protection and only loaded into the system when the individual tests are executed. Depending on the requirements, certificates or private keys are loaded into the test environment. The certificates are then either used as test data, for example, to read data from the certificate, or for encryption, decryption, or generation of signatures.

For some test cases, it is necessary to work with manipulated certificates. Certificates for which the validity period is to be manipulated to a non-valid period are obtained directly from the CA as temporally invalid certificates when they are created and stored in the project resources. Other content manipulations are performed dynamically at runtime by loading the valid certificates and then manipulating them in the test so that certain content becomes invalid.

Public Key Infrastructure OCSP and TLS simulators are used, which are not standard simulator software but explicitly implemented for this case study. While an actual OCSP responder is used in the production environment, a simulator is used for testing. The OCSP simulator allows checking of whether a certificate status is authentic and valid or has been revoked. The simulator checks the general validity of the certificate

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

and also checks the project-specific certificate validity requirements. Moreover, the OCSP simulator can also be used to configure which status is to be returned for a certificate. This function is used, for example, to test how the SUT behaves when the certificate status is invalid or unknown. If a tester does not wish to configure the OCSP simulator for each certificate individually, the status (also valid) can as well be set for several certificates simultaneously.

The TLS simulator is a TLS proxy that uniformly handles TLS termination for all other simulators in the TA. It acts as an intermediate component for connections between the SUT and the simulators. It accepts all HTTP(S), TLS connections on both sides, forwards the requests from the SUT to the simulator or in the reverse direction and performs their encryption or decryption.

Validation of Encrypted and Decrypted Data In this case, encryption and decryption methods are implemented in the TA. These methods prepare or manipulate data and send calls to encryption or decryption services, which are usually called automatically in the production system when data are sent.

To check the decryption functionality of the SUT, it is necessary to produce encrypted data in the TA. Therefore, the implemented encryption methods are used to produce encrypted data in the TA and send them to the SUT. Thus, encryption can be performed with different encryption ciphers, and it can be determined whether the SUT can process all inputs or if it rejects certain encryption ciphers. It can also be used to check particular special cases, such as those in which the encryption is present but the signature is not valid. The prerequisite for using these methods is that one requires control over encryption, signatures, and respective keys in the TA.

The decryption methods implemented in the TA are used to test the encryption functionality of the SUT. Specifically, unencrypted data are sent to the SUT, which encrypts the data and sends it to a component mocked in the TA via a simulator. The simulator functionality allows the encrypted data to be intercepted. The encrypted data are then decrypted using the decryption methods implemented in the TA. In this way, it is possible to check whether the original data are identical to the decrypted data and, thus, whether the SUT has performed the encryption correctly.

5.4.2 Many Different Secure Communication Protocols

This section presents concepts identified in testing this HSSA concerning *many different secure communication protocols*.

Test Tools and Libraries In this case, the code for the tests is manually implemented. However, libraries with supporting functions are employed. The supporting mechanisms of the libraries help to verify whether the SUT has implemented the protocol correctly according to specification. The libraries are used in the tests to send commands or requests to the SUT. Partially, functions of the library are used, and partially custom

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

commands are sent. Moreover, they have functionality that allows the response of the SUT to be stored and queried by the library to assess the correctness of the responses of the SUT. The original libraries do not cover some special cases, so they are extended with additional code, or certain functions are overwritten. In connection with the secure communication protocols, a central logger is used. This library automatically logs all commands and requests sent to a server and their responses. This central logger is registered as a listener with the respective library.

Higher Complexity Arising from Transport Encryption Testing is performed in multiple layers. When testing in multiple layers, tests are first performed without transport encryption to verify the correct transmission of the data and the functionality of the protocols as simply as possible without requiring a focus on complicating mechanisms. Only at a later stage is a complexity level added, and the tests are performed with transport encryption from that point forward.

5.4.3 Test Automation

This section presents concepts identified in testing this HSSA concerning *test automation*.

Initial Setup The JUnit[29] test framework with its prebuilt configuration options is employed. Prebuild configuration options used are, for example, annotations with which specific actions can be executed before and after each test or the use of test suites to combine several test classes that are to be executed together.

Test Code Architecture The following different techniques are used:

- *Parameterized tests*: Test data providers and abstract test scripts are used to decouple the test data from the test code.
- *Preconditions*: Preconditions, reusable encapsulated code, are used to prepare a condition before the actual test execution, for example, by retrieving interfaces to fetch certificates or tokens. If a test case is annotated with a specific precondition, it is executed before the test code execution.
- *Modularization*: Subtests are also used in the TA. These are reusable tests that do not represent completely independent standalone tests by themselves but can be integrated and combined as desired.

State Controllability Two concepts, simulators, and unique test data, are used in this category:

[29] `https://junit.org/junit5/`, Accessed: 21.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- *Simulators*: The simulators used in this case are complex structures that are completely self-implemented. Each simulator is a separate, independent Java[30] or Kotlin[31] project. They are used to simulate real actions or components.
  - *Configurability from TA*: The simulators are fully dynamically configurable from within the TA for each test case individually. Configurability in this context indicates that the simulators respond to requests exactly as desired. A simulator is configured for the subsequent call, a call of a specific path, or via a call that uses a unique identifier.
  - *Creating error states*: To be able to test the system during fault behavior, it is required that the SUT can be placed in every desired state. In the present project, the simulators allow the creation of error states. By default, the simulators are configured to respond to good cases. However, if an error state is to be created, a simulator is configured to respond with the desired error code or error state. Error codes are response codes, while error states can be complex structures such as invalid files or certificates.
  - *Verifiability of incoming and outgoing values*: The implementation of the simulators allows that from the TA, it is possible to query what information is arriving at or being sent from the simulator. Thereby, it can be checked whether requests on certain paths have (not) been sent and whether the content of a request sent by the simulator or a request arriving at the simulator corresponds to the expected value.
  - *Encapsulation*: Simulator connections are managed in their own manager classes.
- *Unique test data*: In the tests, unique test data are used. Name identifiers, or id identifiers, are created uniquely for each execution of a test case using random generators and UUIDs. For example, unique test data are used when configuring the simulators, allowing the isolation of a test and each of its executions.

Resources

- *On-demand setup of a test environment*: A test environment can be established automatically on demand. The following special features are available:
  - *Flexibility of setup*: The on-demand test environment can be started locally and remotely, enabling local and remote testing. The technology used locally is Minikube[32], while Kubernetes[33] is used remotely in the Gitlab[34] CI pipeline. The individual services are integrated and configured as Docker[35] containers.

[30] https://www.java.com/de/, Accessed: 25.08.2023
[31] https://kotlinlang.org/, Accessed: 25.08.2023
[32] https://minikube.sigs.k8s.io/docs/, Accessed: 21.08.2023
[33] https://kubernetes.io/, Accessed: 21.08.23
[34] https://about.gitlab.com/, Accessed: 25.08.2023
[35] https://www.docker.com/, Accessed: 25.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

  - *Scalability*: The test environment is launched depending on the number of components and services required and the resources involved.
  - *Fully automated start and stop*: The test environment can be started or shut down fully automatically by executing a script. A console script is executed locally, and a CI configuration script is executed remotely.
  - *SUT version*: The on-demand test environment can be started with exactly the version of the SUT that is needed. By default, the script extracts the image of the version to be tested from the name of the test branch and starts the test environment with this version. Otherwise, it is also possible to specify a different version via a console parameter.

- *Long and short-running tests*: There is a division into long-running and short-running tests. Long-running tests are marked via annotations and explicitly excluded from the GitLab[36] pipelines executed on every push to the repository. They are only run in nightly pipelines to avoid blocking resources during the day.

Difference Local vs. Virtual Execution of Test Automation Two concepts are applied in this regard:

- *Connection to the remote cluster*: If errors occur only in the remote environment but not in the local environment, then a local connection to the remote cluster is used. In doing so, the tester configures the GitLab[36] configuration so that Minikube[37] uses GitLab's[36] remote pipelines rather than a local Kubernetes[38] environment as in standard local testing. When a test is executed, it is executed in the remote cluster rather than the local cluster. This condition allows testers to see the logs in the local setup and better investigate, for instance, why a bug has occurred in the remote cluster but not in the local cluster.

- *Possible configurable test setup*: By making the test setup as configurable as possible, it is possible to quickly switch from local to remote testing. The configuration data are not hardcoded in the test code but are stored in external configuration files. Depending on the test environment (local or remote), a different configuration file is used in each case, thus allowing a rapid switch.

Simulation of Human Interaction In this case, simple human interactions are simulated with hardware simulators. Methods simulating human interactions are implemented on the simulator. These methods consist primarily of various HTTP calls, which would be executed automatically in certain real-world interactions. The required simulator methods can be called from TA.

[36] `https://about.gitlab.com/de-de/`, Accessed: 21.08.2023
[37] `https://minikube.sigs.k8s.io/docs/`, Accessed: 21.08.2023
[38] `https://kubernetes.io/`, Accessed: 21.08.23

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.4.4 Interoperability Testing

This section presents concepts identified in testing this HSSA concerning *interoperability testing*.

Third-Party Systems No concept is identified for this subcategory as this aspect is not tested in the case.

Complex Environment Management The environment and its instances for the interoperability tests are controlled automatically via scripts. Command line parameters are used to configure how many instances of each component can run and how many components attempt to access and interact with the SUT. This condition is particularly relevant for the interoperability tests performed in this case because the SUT is to be tested under different environmental conditions. This requirement is facilitated by the condition that no complex instances need to be manually started.

Increased Testing Effort No concept is identified for this subcategory as this aspect is not represented in the case.

5.4.5 Further Challenges

This section presents concepts identified in testing this HSSA in relation to *further challenges*.

Restricted Rights and Complicated Debugging

- *Real time log analysis*: With this mechanism, the log data of the individual Kubernetes[39] containers are accessed from the TA. Thus, it is checked whether individual components have carried out certain actions or have not carried them out. In addition, it can be checked whether sensitive data is logged or not logged. The procedure for this is as follows. In the TA, a checkpoint is initiated, from which point in time the logs of a container are tracked. For this process, the user provides the name of the container whose logs should be tracked. If the logs are now required for verification, they can be read out, and assertions can be made on them.
- *Log reporting and log data aggregation*: Another mechanism in this context is log reporting. This library is self-implemented. When this mechanism is enabled in the TA, a test execution log is created. In this case, each test execution step is provided with a log that describes in detail which configuration (for simulators, for example) has been established in the test step, which action has been executed, or which assertions have been made. The library writes the logs to a test execution

[39] `https://kubernetes.io/`, Accessed: 21.08.23

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

protocol. Like that, log data aggregation is achieved. This mechanism can be used in the local setup as well as in the GitLab[40] pipeline.

- *Testing in multiple environments*: Testing is performed in several different environments. In addition to the production environment, different test and integration environments are employed. The rights of a tester differ by stage. In purely internal test environments, for example, testers have full rights and can perform any desired action that is not possible in the production environment. In addition, simulators are also used in these environments, and it is also possible to log more sensitive data to better track errors.

Secure Execution Environment and Storage Management The focus in this context is on two concepts: containerization and virtualization, as well as secure storage.

- *Containerization and virtualization*: As explained in previous sections, isolated in-house test environments are primarily used for the tests. There, each simulator and test object runs in its own Docker[41] container. Therefore, each container can be reinitialized at the runtime of a test with the desired version number. Because each container is a standalone compartmentalized environment, each test running in a container is completely isolated from the other tests.

- *Secure storage*: Sensitive test data are stored in encrypted files, loaded into the system individually for each test run, and deleted again afterward.

Library Version Updates One way to monitor that deployed software versions still function correctly despite library version updates is to rely on automated smoke tests. At certain scheduled time intervals, automated smoke tests are run against the various test, integration, and production environments in GitLab[40] pipelines, and their results are reported in the testers' communication platform. If library version updates occurred in the meantime, it can be checked whether the software's basic functionality has been maintained despite the library updates.

5.5 Cross-Case Analysis

This section presents a cross-case analysis to specify the similarities and differences between the concepts of the respective subcategories. Table 5.2 shows for which subcategories concepts are found. Because comparisons are only possible if a concept is available for a subcategory in both cases, only those subcategories for which a test concept is available for both are compared. The reasons for the similarities and differences are discussed to conclude the cross-case analysis.

[40] `https://about.gitlab.com/`, Accessed: 27.08.2023
[41] `https://www.docker.com/`, Accessed: 27.07.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Main Category	Subcategory	Case 1	Case 2
Key material and security certificates	Dependency on certificate authority	✓	✓
	Creation and manipulation of certificates	✓	✓
	Public key infrastructure	×	✓
	Validation of encrypted and decrypted data	×	✓
Many different secure communication protocols	Test tools and libraries	✓	✓
	Higher complexity arising from transport encryption	×	✓
Test automation	Initial setup	✓	✓
	Test code architecture	✓	✓
	State controllability	✓	✓
	Resources	✓	✓
	Difference local vs. virtual execution of TA	×	✓
	Simulation of human interaction	×	✓
Interoperability testing	Third-party systems	×	×
	Complex environment management	×	✓
	Increased testing effort	×	×
Further challenges	Restricted rights and complicated debugging	✓	✓
	Secure execution environment and storage management	✓	✓
	Library version updates	×	✓

Table 5.2: Cross-case comparison of concepts

5.5.1 Key Material and Security Certificates

This section presents, across cases, the similarities and differences between the case-specific concepts identified in testing HSSA concerning *key material and security certificates*.

Dependency on Certificate Authority In both cases, static certificates are used for testing. To obtain valid certificates they are obtained once from an authentic CA. While the certificates are stored in a keystore for additional security in C2, they are stored in the project resources in C1.

Creation and Manipulation of Certificates As mentioned in the previous sections, valid certificates are obtained via a static approach in both cases. In the case of C1, a dynamic approach, in which certificates are obtained from an CA each time they are used in the tests, is also implemented for some certificates.

For C1, manipulation occurs manually for the static certificates, while this process is performed at runtime via the test code in the dynamic approach. For C2, the manipulation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

occurs via a script.

5.5.2 Many Different Secure Communication Protocols

This section presents, across cases, the similarities and differences between the case-specific concepts identified in testing HSSA concerning *many different secure communication protocols*.

Test Tools and Libraries The tests for testing the secure communication protocols are implemented manually. However, both cases use libraries with supporting functions to send requests to the SUT and receive requests from the SUT. While in C2, libraries have pre-implemented functionality for some requests, in C1, all requests are built manually. In C2, the libraries are extended with additional code; in C1 not.

5.5.3 Test Automation

This section presents, across cases, the similarities and differences between the case-specific concepts identified in testing HSSA concerning *test automation*.

Initial Setup Both cases rely on the use of test frameworks. JUnit[42] is used in both cases with its prebuilt configuration options like annotations and test groupings. The use of Selenium[43] for GUI testing only occurs in C1.

Test Code Architecture

- *Parameterized tests*: Both cases use parameterized tests to decouple test data and test code.
- *Preconditions*: Preconditions are only applied in C2 to prepare a certain state before test execution.
- *Modularization*: Subtests are only applied in C2 to modularize the code.

State Controllability The concept of *simulators* and *mock servers* are used in both cases.

- *Configurability*: In both cases, the simulators or mock servers are configurable from within TA. In C1, a request is sent to an endpoint of the mock server, which sets the values for the desired endpoint. However, in this case, no default values can be used. Instead, the values must be configured for each call to a desired endpoint. In C2, however, fixed default values are set in the simulator. The configuration to

[42] `https://junit.org/junit5/`, Accessed: 27.08.2023
[43] `https://www.selenium.dev/`, Accessed: 27.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

different values is set only for a subsequent call but is not permanent. Thus, no configuration must occur for default values. In addition, in C2, the endpoint does not need to be unique and can also target the next request with a suitable subpath.

- *Creation of error states*: In both C1 and C2, error and invalid states can be created arbitrarily by the simulators and mock servers.
- *Verifiability*: In both cases, it is possible to check in TA which values have arrived at the simulator. In C1, the verifiability is provided by a special endpoint that stores all sent requests. In C2, the verifiability is done via asynchronous methods that are configured to match paths, and once it has matched, the result of the request is saved.
- *Encapsulation*: In both cases, simulators and mock servers, as well as their connections, are encapsulated in and managed through their own management classes.
- *Degree of complexity*: The simulators used in C2 have a significantly more complex and expanded functionality than those used in C1. For example, the simulators used by C2 can handle all TLS termination for all other simulators, unlike those used by C1.

Resources

- *Short and long-running pipelines*: The concept of dividing tests based on their throughput time is only found in C2 and not in C1.
- *Complete start of an on-demand test environment via script*: The use of an on-demand, automatable startup test environment is only present in C2, and not in C1.
- *Semi-automatic testing with test groups*: This approach to saving physical resources is only present in C1 and not in C2.

5.5.4 Further Challenges

This section presents, across cases, the similarities and differences between the case-specific concepts identified in testing HSSA with respect to *further challenges*.

Restricted Rights and Complicated Debugging

- *Establishment of sufficient rights for testing*: Both cases have adopted the concept that an alternative version to the production software should be created to provide testers with additional rights so that they can more easily create states needed for testing. In both cases, however, the concrete implementation differs. In C1, different software images are used for testing (debug and prod build), whereas, in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

C2, multiple test and production environments are used for testing. The basic difference lies in how more rights are established for the testers.

- *Log reporting and log data aggregation*: This approach is found in both cases.
- *Real time log analysis*: This approach is only found in C2 and not in C1.

Secure Execution Environment and Secure Storage Management In C2, sensitive test data are stored in encrypted files and loaded into the system individually for each test run. In C1, on the other hand, each tester stores test data locally in a configuration file, which is then loaded into the system as needed.

While every test object and simulator runs in its own container in C2, this approach cannot be found in C1.

5.5.5 Reasons for Differences in Concepts in Cases

There are numerous reasons that so many concepts are only applied in one of the two cases. One fundamental difference between the two cases is that C1 has a hardware device at its center, while C2 depends on hardware but does not require it for much of the testing process. Another difference is that the number of software components is significantly higher for C2 than for C1. The reason that considerably more concepts were identified in C2 in comparison to C1 is probably because the project scope of C2 is remarkably larger than that of C1, therefore requiring more testing. In addition, the test team for C2 is also significantly larger than that for C1. To avoid a one-sided evaluation of the test concepts, not only the few test concepts common to both but also selected test concepts from both are evaluated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

Resulting Concepts

The following chapter summarizes the concepts identified from the case studies and generifies them. Those concepts that provide added value to testing HSSA, but are not considered specific to HSSA only, but represent general concepts for good testing are presented first. Second, those concepts that are specific for testing HSSA are presented in more detail.

6.1 General Concepts

The concepts listed below can be used to address the complexity of HSSA.

- *Test frameworks*: Test frameworks such as JUnit[44] are particularly useful due to their prebuilt configuration capabilities to simplify the initial setup of TA.
- *Parameterization*: Parameterized tests are used in TA to decouple test data and test code. This process results in abstract reusable test scripts.
- *Preconditions*: Preconditions are used to prepare a state before the actual execution of an automated test.
- *Modularization*: Subtests are used to quickly assemble tests with reusable blocks.
- *Unique test data*: Generation of unique and independent test data leads to the complete isolation of tests and thus enables test parallelization. This process allows each test to run independently with its own data set, providing isolation through the test data.

[44] `https://junit.org/junit5/`, Accessed: 25.08.2023

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- *Configurability of test setup*: Encapsulating configuration parameters in separate files and thus obtaining separation from the test code allows a rapid switch between different test environments, such as local and remote testing.
- *Short and long-running pipelines*: Splitting tests based on their throughput time addresses virtual resource scarcity.
- *Connection to the remote cluster*: Connecting the local setup to the remote cluster when errors occur only remotely but not locally allows better traceability of errors.
- *Environment management via scripts*: This concept focuses on using executable scripts to automatically control how many and which components of a system are started to perform interoperability tests to different environmental conditions.
- *Saving data to configuration files*: This concept deals with storing test data in configuration files to separate it from the test code.
- *Automated smoke tests*: Automated smoke tests allow the functionality of deployed software states, including states after library version updates, to be monitored to ensure that functionality remains after an update.

6.2 HSSA-Specific Concepts

This section explains the generified test concepts that are presented to the experts for evaluation.

6.2.1 Certificate Creation and Manipulation - Static and Dynamic Approaches

Many certificates are involved in the testing of HSSA. Valid certificates can be obtained from a CA in two ways:

- *Static approach*: In this approach, certificates are obtained once from a real instance of a CA before test execution and stored in the project resources. The validity period should be as long as possible so that they do not have to be replaced until the latest possible time. The certificates are only loaded into the system during test execution.
- *Dynamic approach*: This approach assumes that the instance of CA is available during testing. With this approach, certificates are obtained from the CA with each test execution.

Manipulation of the certificates can be performed manually with the static approach or dynamically in the test code during test execution.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

While the dynamic approach offers the advantage that certificates do not have to be exchanged manually, the static approach offers the advantage that there is no dependency on a CA during the tests.

6.2.2 PKI Simulators

The idea behind this concept is that individual components of a PKI are simulated so that no entire PKI has to be duplicated for the tests. The simulators should be configurable, which certificates they return and what the validity status of a certificate is. Furthermore, one component should handle the TLS termination for all other components so that this is implemented centrally, and not each component does have to implement this part.

6.2.3 Implementation of Encryption and Decryption Methods in Test Automation

To be able to check that the SUT correctly implements the encryption and decryption functionality, there should be the possibility of encrypting and decrypting data in the TA. This process allows sending encrypted data to the SUT or receiving encrypted data from the SUT. When producing encrypted data in the TA, it is possible to test whether the SUT can process data that is encrypted and signed in different ways, including invalid signatures or encryption. When receiving encrypted data from the SUT, it is possible to check if the SUT implemented the encryption correctly by using the implemented decryption methods in the TA.

6.2.4 Usage of Libraries when Testing Secure Communication Protocols

When testing secure communication protocols, if the tests are implemented manually, libraries should always be used whenever possible. Before applying the library, it should be checked whether the library allows library code to be extended or overwritten. This is particularly relevant if functionality that deviates from the standard case is to be tested. Furthermore, it should be checked whether the library allows custom commands to be sent. This is relevant when testing how the SUT reacts to misbehavior. Especially in the case of HSSA, these two aspects are to be considered since the SUT is to be tested often with special cases.

6.2.5 Testing in Several Layers (With and Without Transport Encryption)

Testing in several layers is employed to simplify the validation of a communication protocol's functionality and its correct data transmission. The idea behind this concept is to first perform tests without transport encryption to check the correct transmission of data and the functionality of the protocols as simply as possible. A complexity level is

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

added at a later stage, and the tests are performed with transport encryption from that point forward.

6.2.6 Simulators

The complexity of HSSA, including its numerous security mechanisms, requires that numerous complex scenarios can be created during the tests to ensure that the SUT is tested in every form. Simulators are particularly suitable for implementing this process. With this approach, the problem of state control can be solved.

When using simulators to test HSSA, the following should be considered:

- *Desired state creation*: It should be possible to create any desired state, such as standard scenarios, invalid scenarios, and error scenarios, using the simulators. The default values for the good cases should be implemented on the simulators themselves so that only invalid or error scenarios require additional configuration.
- *Verifiability*: It should also be possible to check whether the expected values have arrived at the simulator or whether certain calls have actually been made.
- *Configuration*: It should be possible to configure a simulator dynamically in each test case.
- *Encapsulation*: Simulator connections should be encapsulated in their own management classes.

6.2.7 Semi-Automatic Testing with Test Groups

The approach of semi-automatic testing with test groups is particularly applicable to HSSA in which different complex hardware states are present. To reach a high test coverage, it is necessary to conduct tests for as many hardware states as possible. The idea of this concept is to divide tests into logical groupings, with each grouping corresponding to one state of the device. As soon as a device has been prepared in exactly one state, the tests of that test group can be executed automatically. In this approach, a test device is only needed once, saving physical resources (number of hardware devices).

6.2.8 On-Demand Setup of a Test Environment

In HSSA, many different components are involved, which can lead to virtual resource scarcity. One approach to resolving this problem is an on-demand test environment. It is recommended that the isolated test environment can be started fully automated without a need for manual interaction. This condition can be implemented using a script, for example. The script should be applicable for a local test environment as well as a remote test environment so that testers are flexible in their test setup. In addition, the test environment should be able to be started against any desired state and version of the SUT.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

6.2.9 Establishment of Sufficient Rights for Testing

Due to the security mechanisms of HSSA, testers are severely restricted in their options for action. To be able to test more effectively, alternatives in which testers are unrestricted in their possibilities of action must be created.

- *Testing in different environments*: In this approach, test environments in which security mechanisms have been inactivated to give testers more rights are created. In dedicated test environments, it is, therefore, possible for the testers to use test certificates and simulators.
- *Testing with debug and prod build*: In this approach, a software version alternative to the production software is deployed for testing. The software debug build contains additional software features, but these features are used exclusively for testing and are not intended to enter the production version. With the additional features, testers can create states more directly without having to perform all actions end to end.

6.2.10 Log Reporting and Log Data Aggregation

In TA tests, each test execution step should be documented with a log that should contain information regarding which specific configurations are made for the test case, what occurs in detail in a test step, and what exactly is checked. Subsequently, these logs should be stored and aggregated in a test execution log protocol by automatically rewriting the log outputs.

The detailed test execution logs help to trace what caused a test execution to fail even though no sensitive content may be logged due to HSSA security restrictions.

6.2.11 Real Time Log Analysis

The idea of this approach is to be able to analyze the logs of many different components. Therefore, a mechanism must be implemented in the TA that allows easy access to these logs. This allows checking for specific log contents in different components and verifying if certain actions occurred or did not occur or if sensitive data was logged or not. This approach is needed to deal with the high number of different log files due to the various components of a HSSA.

6.2.12 Containerization and Virtualization

Because highly sensitive data are involved in HSSA, the tests should also be executed in an environment as secure and isolated as possible. One way to increase isolation is through containerization and virtualization. In TA, each simulator and test object should be executed in its own container. Thus, a higher degree of isolation can be achieved. This approach creates the possibility that a container can be restarted quickly and easily before each test execution to create the same initial state for all tests.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 7

Evaluation

By conducting a case study, concepts for testing HSSA were identified and assigned to the different challenge subcategories of the deductive category system (cf. Chapter 5). Subsequently, these concepts were generified (cf. Chapter 6). The current chapter presents the evaluation of the HSSA-specific generified concepts (cf. Section 6.2) and their mapped challenge categories. For this purpose, semi-structured interviews are conducted with three experts.

7.1 Methodology

This section presents the chosen data collection form and the data analysis method.

7.1.1 Data Collection

Semi-structured expert interviews are conducted to evaluate the identified concepts and the mapped challenge categories. A mixed-methods approach is employed. This data collection form is deemed suitable due to the following reasons:

1. *Expertise*: Through their many years of practical experience, the experts have acquired specialist knowledge and are therefore qualified to evaluate the identified concepts in terms of their quality, transferability, and validity.
2. *Validation of results*: By validating the results of the case study through expert interviews, a higher objectivity and generalizability of the results can be generated.

7.1.2 Data Analysis

Quantitative results are evaluated by calculating the average value ($\varnothing$), and the qualitative results are evaluated using Mayring's [14] qualitative content analysis of *Structuring*.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Expert	Abbr.	Role	Interview date	Medium
Expert 5	E5	Software tester with focus on test automation	17.07.23	Jitsi Meet
Expert 6	E6	Software developer	18.07.23	Jitsi Meet
Expert 7	E7	Software tester with focus on test automation	25.07.23	Jitsi Meet

Table 7.1: Overview of experts (evaluation interviews)

The structuring approach is suitable for evaluating the identified concepts and mapped challenging subcategories, as content-related aspects of the interview material are of interest. The experts' answers are classified into the deductive category system. Each category corresponds to one of the concepts presented in Section 6.2.

7.2 Interview Design

This section presents the selection of experts and the interview guideline.

Selection of Experts The experts were selected from a large IT company whose projects lie in the area of HSSA. Table 7.1 shows their exact roles. The interviews were all conducted via the video messenger Jitsi, and the planned duration for each interview was 45 minutes.

Interview Guideline An interview guide was created to ensure comparability of the expert responses. The interview questions refer to the concepts generified in Section 6.2. The experts were presented with each concept and the corresponding mapped challenging category, along with two optional qualitative and quantitative questions, each with two sub-questions about their assessment of the quality and transferability of the respective concept and its validity concerning the challenging category. The entire guide can be found in Appendix D.

7.3 Data Analysis

This section explains the quantitative and qualitative assessments of the experts. The quantitative assessment of the transferability and quality, or validity of a concept is based on a scale of one to four, with one being very high or highly valid and four being very low or not at all valid. Figure 7.1 shows the average values of the quantitative estimation regarding the transferability, quality, and validity of each concept. Table 7.2 shows the ranking of all concepts in descending order.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

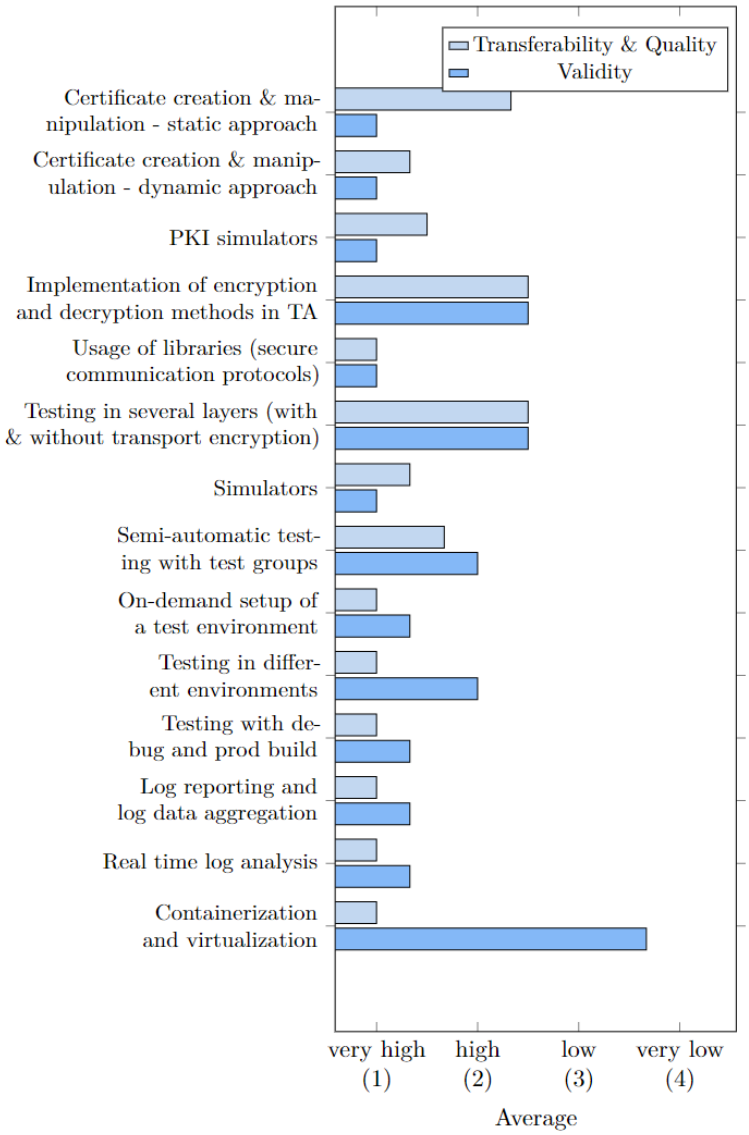

Figure 7.1: Quantitative evaluation results

7.3.1 Certificate Creation and Manipulation - Static and Dynamic Approaches

The overall quality and transferability of the static approach are rated as high (⌀2.33) and that of the dynamic approach as very high (⌀1.33).

- *Static approach*: This approach has the advantage that there is no dependence on a CA during testing, and no such high implementation effort is required in contrast to the dynamic approach [E5].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- *Dynamic approach*: This approach has the advantage of being future-proof as certificates do not need to be manually replaced within certain time periods [E5]. Regarding certificate manipulation, this approach is more suitable because the manipulation can be performed dynamically and flexibly, depending on the need for a particular test case, and it is especially preferable in non-standard cases [E6].

Both the static approach (⌀1.0) and the dynamic approach (⌀1.0) are considered to be highly valid concepts for creating and manipulating certificates.

All experts prefer the dynamic approach, but at the same time, they state that the preferable approach strongly depends on the specific project setting for testing.

7.3.2 PKI Simulators

One expert abstained from voting regarding the quality and transferability of this concept as the transferability of the concept is too dependent on the specific project context for him to be able to assess it. The remaining experts rate its quality and transferability as high to very high (⌀1.5). An advantage of this approach is that it facilitates configuring simulators to issue desired certificates [E6].

The experts rate the concept as a highly valid approach to avoid having to duplicate an entire productive PKI for testing and instead simulating single components (⌀1.0). However, the actual applicability depends on which aspects of the PKI are to be used for the tests and whether it is accordingly, first of all at all, possible to simulate the corresponding PKI component or whether it is also reasonable to do so [E6].

7.3.3 Implementation of Encryption and Decryption Methods in Test Automation

One expert abstained completely from this concept due to a lack of experience regarding this concept. The remaining experts rate the quality and transferability of this concept as moderate (⌀2.5). Prebuild services and libraries should be used wherever possible when implementing encryption and decryption methods in the TA. Complete self-implementation without their help is highly error-prone [E5].

The concept's validity in checking the validity of encrypted or decrypted data during testing is also rated as moderate (⌀2.5). It is not a desirable but absolutely necessary concept. To verify the decryption functionality of the SUT, it is necessary to produce encrypted data in the TA and send it to the SUT. This ability is made possible by the presented solution concept [E7].

7.3.4 Usage of Libraries when Testing Secure Communication Protocols

The experts rate the overall quality and transferability of this concept as very high (⌀1.0). The procedure mentioned above for using libraries is a generally valid approach that

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

applies not only when testing secure communication protocols but also generally when using libraries [E5], [E6], [E7]. A problem with this approach is that libraries are often initially classified as applicable, and non-standard cases that are not covered may emerge at a later time. Therefore, before using the libraries, it should be determined as precisely as possible which potential application scenarios will be necessary. Because libraries are expected to support correct behavior, they often permit no error cases at all or correct them automatically [E7].

In the opinion of the experts, the above approach is a highly valid approach for accommodating the lack of supported functions in libraries (⌀1.0). It is especially valid to try to extend libraries with additional code in case the library does not cover the corresponding behavior. It only has to be considered that some libraries might not allow for extending or overriding library functions [E6].

7.3.5 Testing in Several Layers (With and Without Transport Encryption)

One expert abstained from both votes regarding this concept. Overall, the quality and transferability of the concept are rated as moderate (⌀2.5), although the remaining experts rated this point very differently. [E6] considers the quality and transferability to be very high because, in his opinion, testing without transport encryption allows a level of complexity to be removed from the tests without encryption. [E7] considers the quality and transferability to be very low because, in his opinion, it offers no added value from a testing perspective to omit encryption because the system must be tested with transport encryption in the final state in any case.

The experts' opinions differ widely regarding whether this concept is a valid approach to accommodating the increased complexity of testing communication protocols due to transport encryption. The overall rate was moderate (⌀2.5). One expert ([E6]) considers the validity to be very high, as he states that it is advantageous to test without transport encryption during initial functional tests because this method places a stronger focus on actual functionality, making verification of the correct transmission of data more straightforward and also simplifies troubleshooting in the event of errors. However, [E7] states that this approach is primarily dictated by development, as only the already implemented functionality of the software, whether with or without transport encryption, can be tested.

7.3.6 Simulators

Overall, the quality and transferability of the concept of simulators in testing HSSA is considered to be very high (⌀1.33). This concept is particularly appealing in that it does not involve simple mock interfaces but rather a complex entity with a high level of detail, which allows the simulators to be controlled very precisely [E6]. Moreover, simulators can often be used in practice for multiple projects rather than just one because they are implemented as a separate product. Due to this transferability across projects, the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

implementation effort can be reduced significantly [E7]. The transferability of the concept has also been shown in practice in which different teams have independently chosen the same approach [E5].

The concept's validity is rated as very high (⌀1.0). Simulators can be used to address the problem of controllability during testing because any error scenarios can be created using them [E5], [E7].

7.3.7 Semi-Automatic Testing with Test Groups

The overall quality and transferability of semi-automatic testing with test groups is rated as high (⌀1.67). However, it should be noted that this concept is only transferable to projects involving hardware in which the state is statically preparable [E5], [E6]. For example, if human interaction is required during testing, state preparation is impossible [E5]. Full instead of semi-automation of tests would be desirable, but this approach provides an attractive optimization opportunity for projects in which this is not possible [E7].

According to the experts, the approach of semi-automatic testing with test groups is a valid approach (⌀2.0) to minimize the number of hardware needed to be tested. Through this approach, hardware is no longer needed in large numbers but only one device, saving many physical resources [E7]. However, this approach is only a short-term and not a long-term solution [E5].

7.3.8 On-Demand Setup of a Test Environment

Overall, the experts rate the quality and transferability of this concept as very high (⌀1.0). This approach can be transferred very well to software-only projects. However, if hardware is involved, it is project-specific if this concept is applicable [E5]. This approach is particularly appealing because the environment can be started in a completely automatable way with every desired version of the SUT. The test environment can also be easily reset to the status quo at any time through automation [E6].

According to the experts, the approach of using an on-demand test environment is highly valid (⌀1.33) in addressing the virtual resource scarcity that occurs during HSSA testing. With this approach, resources are not permanently blocked, as would be the case with a fixed stage [E5], [E6].

7.3.9 Establishment of Sufficient Rights for Testing

For this category, two concepts are evaluated because different approaches could be identified in how the testers get sufficient rights in both case studies.

Testing in Different Environments This concept's overall quality and transferability are rated as very high (⌀1.0). The advantage of this approach is that it can be used to

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

test whether the deployed software also functions as intended with the varying conditions that are present in the different environments (such as differences in databases) [E6].

The concept is also designated as a valid approach to establishing additional rights for the testers (⌀2.0). However, testing in different environments is standard and represents a consideration not only in HSSA to address limited rights due to security mechanisms but also in any other software architecture. In addition, the test environments are usually located in the company's internal area of control, and any desired rights can be established there at any time [E7].

Testing with Debug and Prod Build The quality and transferability of this concept are rated as very high (⌀1.0). However, one risk in this area is that this approach presents many more vulnerabilities and sources of error in comparison to the previously mentioned approach because the various debug feature flags could potentially also be activated in production [E5].

According to the experts, testing with debug and prod build is a highly valid approach (⌀1.33) to establish sufficient rights for the testers. This approach allows testers to view certain data that cannot be seen in production, take certain actions, and create states more directly [E7].

Preferred Use of which Concept Both [E5] and [E6] prefer the approach of testing in different environments, as this approach is much less error-prone than that of testing with debug and prod build. [E7] would prefer a third approach, in which the additional features needed for testing are packaged into additional artifacts and thus separated from the production code.

7.3.10 Log Reporting and Log Data Aggregation

According to the experts, the quality and transferability of the concept of log reporting and log data aggregation are very high (⌀1.0). This concept is attractive in that it creates a test execution step log protocol that can be given to the development team or even to the customer as proof that requirements have been met [E6], [E7]. In addition, this strategy can be used to compare with the test specification to determine whether all steps have indeed been executed and that the functionality has thus been fulfilled [E7].

The experts consider this concept to be a highly valid solution (⌀1.33) for addressing restricted log output. Although sensitive content is not logged, log reporting clarifies during which action errors occur and what causes these errors as every test execution step is provided with a detailed description of what is happening [E5], [E6]. In cases in which log interceptors are used to log the various protocol requests and responses, events occurring in the network at that moment are clear. Especially in non-standard cases, logging the description of the configuration also increases traceability, making it unnecessary to log sensitive data because traceability is still ensured [E7].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.3.11 Real Time Log Analysis

The experts rate the quality and transferability of this concept as very high (⌀1.0). This concept is particularly useful for HSSA because so many different components are present and so many different log files, therefore, must be considered when debugging during testing [E6]. However, it must be noted that if the real-time logs are used to examine conditions out of the TA, these logs may be delayed, and they are then read out too early, which means that certain actions have not been logged yet, although they have occurred and therefore an action is incorrectly asserted as not done [E7].

Furthermore, according to the experts, this concept is a highly valid approach to dealing with the high number of different log files of various components [⌀1.33]. This approach makes it easy to access the log outputs of individual components [E6].

7.3.12 Containerization and Virtualization

Overall, the experts rate the quality and transferability of this concept as very high (⌀1.0). An advantage of this concept is that containerization makes resetting the test system or individual components of the test system rapid and straightforward [E6]. In addition, it is easier to create certain states for tests that require isolation, such as network isolation [E5].

However, in their opinion, this concept is not at all a valid solution (⌀3.67) for creating a secure execution environment. Containerization creates a higher degree of isolation but does not provide a more secure execution environment. In addition, whether a more secure execution environment is even necessary for testing must be considered [E5], [E7].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Transferability and Quality	Validity
• Usage of libraries when testing secure communication protocols (⌀1.0) • On-demand setup of a test environment (⌀1.0) • Testing in different environments (⌀1.0) • Testing with debug and prod build (⌀1.0) • Log reporting and log data aggregation (⌀1.0) • Real time log analysis (⌀1.0) • Containerization and virtualization (⌀1.0) • Certificate creation and manipulation - dynamic approach (⌀1.33) • Simulators (⌀1.33) • PKI simulators (⌀1.5) • Semi-automatic testing with test groups (⌀1.67) • Certificate creation and manipulation - static approach (⌀2.33) • Implementation of encryption and decryption methods in test automation (⌀2.5) • Testing in several layers (with and without transport encryption) (⌀2.5)	• Certificate creation and manipulation - static approach (⌀1.0) • Certificate creation and manipulation - dynamic approach (⌀1.0) • PKI simulators (⌀1.0) • Usage of libraries when testing secure communication protocols (⌀1.0) • Simulators (⌀1.0) • On-demand setup of a test environment (⌀1.33) • Testing with debug and prod build (⌀1.33) • Log reporting and log data aggregation (⌀1.33) • Real time log analysis (⌀1.33) • Semi-automatic testing with test groups (⌀2.0) • Testing in different environments (⌀2.0) • Implementation of encryption and decryption methods in test automation (⌀2.5) • Testing in several layers (with and without transport encryption) (⌀2.5) • Containerization and virtualization (⌀3.67)

Table 7.2: Ranking of the concepts according to expert evaluation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 8

Discussion

This chapter discusses and interprets the results of this work in light of the research questions defined in Section 1.2. Furthermore, it discusses the threats to validity and mentions potential future research.

8.1 Answering the Research Questions

The following section answers the research questions, interprets the results, and explains lessons that have been learned.

8.1.1 Answering RQ1

This section summarizes the results for research question RQ1 ("What are the challenges in testing in the context of HSSA and into which categories can the identified challenges be divided?"), interprets and discusses them. The details for answering this research question can be found in Chapter 4.

Summary The challenges identified for testing HSSA can be divided into the main categories of:

- *Key material and security certificates*
- *Logical network separation*
- *Spatial separation from the hardware*
- *Many different secure communication protocols*
- *Test automation*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- *Testing with different authorization roles*
- *Test data*
- *Interoperability testing*
- *Testing infrastructure*
- *Further challenges*

Based on their average votes, the experts considered the main categories of *key material and security certificates*, *test automation* and *interoperability testing* to be the three most challenging categories.

Interpretation When comparing the actual results of this analysis with the challenge categories identified in Chapter 3, it is surprising that the categories *test data*, *spatial separation from the hardware* and *testing infrastructure* were considered by at least 50% of the experts to be rather not challenging even though these aspects were identified as challenges in the papers investigated in Chapter 3. One reason for this result may be that the literature review for challenges in testing was based on systems similar to HSSA but not on HSSA themselves because no papers that would have covered this aspect were available.

It would also have been expected that the security mechanism categories (*key material and security certificates*, *logical network separation*, *testing with different authorization roles*) added to the deductive category system for the expert interviews would be considered challenging in testing HSSA, because security limits testability. However, the main categories of *key material and security certificates* and *logical network separation* were indeed considered challenging, but *testing with different authorization roles* was considered rather not challenging. The reason that *testing with different authorization roles* was rated as rather not challenging is presumably that the challenges of the subcategories are rather general. Therefore, they might also occur with other software architectures.

Overall, expectations were mainly met in that the experts named challenges in testing HSSA for each category identified by the literature review that had been conducted in advance.

8.1.2 Answering RQ2

This section summarizes the results for research question RQ2 ("How are the identified challenge categories addressed in practice?"), interprets and discusses them. The details for answering this research question can be found in Chapter 5 and Chapter 6.

Summary Based on the case studies, the concepts that were identified as the most relevant for testing HSSA were the concepts of:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- *Certificate creation and manipulation - static and dynamic approaches*
- *PKI simulators*
- *Implementation of encryption and decryption methods in test automation*
- *Usage of libraries when testing secure communication protocols*
- *Testing in several layers (with and without transport encryption)*
- *Simulators*
- *Semi-automatic testing with test groups*
- *On-demand setup of a test environment*
- *Establishment of sufficient rights for testing*
- *Log reporting and log data aggregation*
- *Real time log analysis*
- *Containerization and virtualization*

Interpretation Overall, it is surprising that far fewer test concepts could be identified in C1 than in C2. One probable reason for this result is the smaller project and test team size of C1. In addition, the testers in C1 do not have as much professional experience as those in C2. For the *interoperability testing* area, no concepts at all could be found in C1 because no interoperability testing has been performed in C1. Another reason that no concepts were identified for C1 in some areas is that the environment for testing is only local and not remote.

One particularly crucial concept is that of *simulators*, which are extensively used in both projects in slightly different ways but with fundamentally similar ideas. The most significant feature of this concept is that invalid and error states can be optimally produced due to the complexity of the implementation. This effect makes it possible to test HSSA under error conditions exceptionally well.

Overall, the concepts that have been identified assist in testing HSSA because they provide approaches to testing this architecture despite its complexity and additional security mechanisms.

8.1.3 Answering RQ3

This section summarizes the results for research question RQ3 ("Which implications for testing HSSA can be derived from the empirical results of the case study?"), interprets and discusses them. The details for answering this research question can be found in Chapter 7.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Summary The experts rated seven out of 14 concepts as very transferable and qualitative, and highly valid approaches for the respective mapped challenge subcategory. Four of the 14 concepts were rated at least high to very high in transferability, quality, and validity concerning the mapped challenge subcategory. Only three of the 14 concepts were rated as rather low in transferability, quality, or validity concerning the respective challenge subcategory.

Interpretation Overall, most of the concepts generified in Chapter 6 are of high quality and transferable to other projects and can address numerous challenges. Thus, the result corresponds to the expectations.

Implications and recommendations for concepts that were rated as the least transferable and with the least quality, the least valid, or both are the following:

- *Implementation of encryption and decryption methods in test automation*: Although this concept was classified as only moderately transferable, qualitative, and valid overall, it is probably necessary to employ it because otherwise, the ability of the SUT to correctly encrypt and decrypt data including non-standard cases, cannot be checked. However, it is recommended to use libraries with supporting functionality if possible.
- *Testing in several layers (with and without transport encryption)*: This concept was classified as moderately transferable, qualitative, and valid. Because the development team and not the test team determine which features are ready for a test, it should instead not be used.
- *Containerization and virtualization*: This concept was seen as very transferable but not as a solution for the secure execution environment and storage management challenge. However, it can be used to achieve higher isolation between tests and to be able to reset individual containers to the status quo at any time.

Implications and recommendations for concepts that were rated as very transferable, of high quality, and highly valid are the following:

- *Certificate creation and manipulation - dynamic approach*: To obtain certificates during testing, the dynamic approach is generally preferable to the static one.
- *Usage of libraries when testing secure communication protocols*: Libraries should always be used when possible. Before using them, however, the scenarios that the library is to cover should be known.
- *Simulators*: The use of simulators is highly recommended, especially when testing invalid or error scenarios. They should also be used when tests should be performed independently of other components.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- *On-demand setup of a test environment*: The use of this concept is beneficial for pure software projects and creates a high degree of flexibility for the testers, as testers can start up the test environment in the desired state automatically.

- *Testing with debug and prod build*: When using this approach, tests should not only be conducted with the debug build but also with the prod build in the initial phases so that errors in the production version of the software can be detected at an early stage.

- *Log reporting and log data aggregation*: This approach is beneficial for error analysis when no sensitive data is logged as the aggregated logs nevertheless allow the discovery of specific patterns that point to the reason for the errors.

- *Real time log analysis*: This concept is particularly useful when many different components with different log outputs are present.

It should be particularly emphasized that the majority of the generified concepts (79 percent) have a high to very high quality and transferability and are classified as valid to very valid solutions and can, therefore, be employed as recommended to address the mapped challenge subcategories in practice. It should be noted that the concepts could also represent solutions for other challenges. This high percentage of well-rated concepts shows the contribution to generating new knowledge in testing HSSA.

8.2 Threats to Validity

This section explains the threats to validity.

Interviewee Selection and Number of Interviewees The seven experts interviewed regarding the challenges of testing HSSA on the one hand and the evaluation of the identified concepts and the mapped challenges on the other hand all worked in the same IT company. Although this IT company is vast and the experts were involved in a wide variety of projects, it would have been ideal to interview experts from different companies to increase the objectivity of the results. Moreover, it would have been more desirable to interview a larger number of experts to improve the generalizability of their arguments.

Number of Investigated Cases The case study of this thesis only investigated two cases to identify concepts in testing HSSA. This number may be too small to generalize the concepts that were found, as this would usually require a more significant number of cases to be investigated. Moreover, both cases originated from the eHealth sector. Cases should also be examined from other sectors, such as the banking or government sectors, to better generalize the results.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Concepts Investigated in the Case Study It would have been ideal if all concepts, rather than only those perceived as HSSA-specific, had been examined in the final evaluation. However, the limited time span of an interview was the limiting factor in this case. In addition, the assessment of the HSSA specificity of the concepts was subjective. It would have been preferable to briefly present all concepts to a panel of experts and then allow them to select which ones were relevant to testing HSSA and deserve closer examination in the evaluation.

8.3 Future Research

In the future, the findings of this thesis could be extended by evaluating the concepts that were not presented to the experts for evaluation. In addition, further case studies in areas other than the eHealth sector, such as the financial sector, could be investigated. In addition, the challenge categories that were excluded based on the experts' votes on the degree of difficulty could also be examined.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 9

Conclusion

Previous literature has not thoroughly examined concepts for testing HSSA. However, this is becoming increasingly important due to the increasing digitization and interconnectivity of software systems as sensitive data in digitized form becomes ever more significant in everyday life. To continue to protect them, it is important to be able to test such HSSA despite their complex architecture. The thesis offers a scientific contribution by proposing concepts for testing HSSA, which will support software companies in the future to test their HSSA and thus improve the quality and security of their software.

The first step was to deduce a deductive category system for the expert interviews conducted in the second step. The main categories were derived from two approaches.

- *Literature review*: A literature review was conducted to identify challenges in testing HSSA in theory. Since challenges of testing HSSA itself have hardly been studied in the literature, challenges of software architectures similar to HSSA were investigated. Based on this, challenge main categories could be deduced.
- *Security mechanisms*: Security mechanisms limit testability. Therefore, to determine whether individual security mechanisms create challenges when testing HSSA, selected security mechanisms were added as main categories to the category system.

The second step was to conduct semi-structured expert interviews to determine the existing challenges in testing of HSSA (RQ1). The four experts interviewed all work in an IT company in the field of testing HSSA. Based on the deductive category system deduced in the first step, the experts were asked how challenging they consider each main category to be and which challenges arise concerning each main category when testing HSSA. During data analysis, challenge subcategories were inductively added to the existing main categories. The data analysis results showed that, based on the average values, the three most challenging main categories are: *key material and security*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

certificates, *test automation* and *interoperability testing* in the context of testing HSSA. Next, the categories to be examined more thoroughly during the case study were chosen. From the most challenging ones, those categories with a technical focus were selected.

The third step aimed to determine which concepts are used to address the identified challenge categories in practice. Therefore, a multiple-embedded case study with two cases was conducted investigating two projects with HSSA from the eHealth field. Due to the sensitivity of health data, there is a particular focus on testing in this area. During the case study, the individual concepts identified were assigned to the individual subcategories of the deductive category system. The data analysis has shown that both cases feature numerous different concepts for testing HSSA.

The fourth step was to evaluate the concepts' quality, transferability, and validity concerning the mapped challenge subcategories. Therefore, semi-structured expert interviews with qualitative and quantitative questions were conducted with three experts. The concepts that were rated as high to very high in both quality and transferability, as well as a valid approach to the respective mapped challenge subcategory when testing HSSA, were the concepts of *certificate creation and manipulation - dynamic approach*, *PKI simulators*, *usage of libraries when testing secure communication protocols*, *simulators*, *semi-automatic testing with test groups*, *on-demand setup of a test environment*, *testing in different environments*, *testing with debug and prod build*, *log reporting and log data aggregation*, and *real time log analysis*.

Upon examining the methodology, it can be concluded that interviewing experts was reasonable because their practical experience allowed for determining challenges in real-world projects. Moreover, the chosen methodology of the case study was beneficial, as its practical reference enabled the identification of concepts that had already proven successful in practice. Evaluation by experts increased the objectivity and meaningfulness of the concepts and the mapped challenge categories.

The results of the thesis contribute to building new knowledge about the challenges of testing HSSA and concepts that are used in practice to address them, especially since a very high proportion of concepts were deemed to be beneficial regarding their transferability, quality, and validity by the subsequent evaluation of the experts. The knowledge gained can primarily support software companies that develop HSSA by providing them with ideas on how to test their HSSA, allowing them to save time and cost as they can draw upon existing solutions. This is particularly useful when they start testing their HSSA for the first time or when they encounter similar problems to those identified in the thesis and thus can address them with appropriate concepts.

Both cases studied originated from the eHealth industry. Therefore, it could be interesting to investigate further cases from other areas with sensitive data, such as the financial sector, to add additional perspectives. Moreover, a higher number of cases should be investigated to enhance generalizability. In the future, a wide range of possible concepts for testing HSSA could be generated and help companies to test their HSSA in detail and thus improve the quality and security of their software.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

API Application Interface. 13

C1 Case 1. 49, 50, 62–65, 85

C2 Case 2. 50, 51, 62–65, 85

CA Certificate Authority. 21, 34, 42, 51, 52, 55, 62, 68, 69, 75

CI Continuous Integration. 12, 13, 51, 59

FTP File Transfer Protocol. 21

GUI Graphical User Interface. 53, 63

HSSA High-Security Software Architectures. xiii, xv, 1–5, 17, 18, 23, 25–34, 37–44, 47, 49, 51, 52, 54–57, 60, 62–64, 67–71, 73, 74, 77–80, 83–85, 87–91, 108, 110, 112–114

HTML Hypertext Markup Language. 21

HTTP Hypertext Transfer Protocol. 21, 37, 50, 56, 59

IEC International Electrotechnical Commission. 6, 16, 17

IEEE Institute of Electrical and Electronics Engineers. 5, 17

IoT Internet of Things. 18, 23–27

ISO International Organization for Standardization. 6, 16, 17

ISTQB International Software Testing Qualifications Board. 5, 9, 10, 14

LAN Local Area Network. 19

MSA Microservices Architecture-Based Applications. 18, 23, 25, 26

OCSP Online Certificate Status Protocol. 21, 55, 56

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

PKI Public Key Infrastructure. 21, 34, 35, 42, 50, 69, 76

POP3 Post Office Protocol Version 3. 21, 37, 50

SMTP Simple Mail Transfer Protocol. 21, 37, 50

SSH Secure Socket Shell. 50, 52, 54

STLC Software Testing Life Cycle. 5–7, 91

SUT System Under Test. 38–41, 43, 51, 52, 56–60, 63, 69, 70, 76, 78, 86

TA test automation. 5, 12, 13, 17, 25, 26, 32, 35, 37–39, 42, 50–53, 56–60, 62–64, 67, 69, 71, 76, 80

TLS Transport Layer Security. 21, 26, 35, 37, 50, 55, 56, 64, 69

UI User Interface. 13, 18, 50, 54

VLAN Virtual Local Area Network. 19

VM Virtual Machine. 15

VPN Virtual Private Network. 36, 50

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

Scientific References

[1] S. Chenthara, K. Ahmed, H. Wang, and F. Whittaker, „Security and privacy-preserving challenges of e-health solutions in cloud computing“, *IEEE Access*, vol. 7, pp. 74 361–74 382, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2919982.

[2] E.-Y. Daraghmi, Y.-A. Daraghmi, and S.-M. Yuan, „Medchain: A design of blockchain-based system for medical records access and permissions management“, *IEEE Access*, vol. 7, pp. 164 595–164 613, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2952942.

[3] E. Yaacoub, K. Abualsaud, T. Khattab, M. Guizani, and A. Chehab, „Secure mhealth iot data transfer from the patient to the hospital: A three-tier approach“, *IEEE Wireless Communications*, vol. 26, no. 5, pp. 70–76, Oct. 2019, ISSN: 1536-1284. DOI: 10.1109/MWC.2019.1800590.

[4] M. Z. Chowdhury, M. T. Hossan, M. Shahjalal, M. K. Hasan, and Y. M. Jang, „A new 5g ehealth architecture based on optical camera communication: An overview, prospects, and applications“, *IEEE Consumer Electronics Magazine*, vol. 9, no. 6, pp. 23–33, Nov. 2020, ISSN: 2162-2248. DOI: 10.1109/MCE.2020.2990383.

[5] J. D. Miranda-Calle, V. Reddy, P. Dhawan, and P. Churi, „Exploratory data analysis for cybersecurity“, *World Journal of Engineering*, vol. 18, no. 5, pp. 734–749, Feb. 2021, ISSN: 1708-5284. DOI: 10.1108/WJE-11-2020-0560.

[6] W. Firmansyah, T. Mantoro, and P. D. Persadha, „Regulatory support to prevent health data breaches“, in *2022 IEEE 8th International Conference on Computing, Engineering and Design (ICCED)*, IEEE, Jul. 2022, pp. 1–4, ISBN: 978-1-6654-5389-9. DOI: 10.1109/ICCED56140.2022.10010539.

[7] A. F. Altwairqi, M. A. AlZain, B. Soh, M. Masud, and J. Al-Amri, „Four most famous cyber attacks for financial gains“, *International Journal of Engineering and Advanced Technology*, vol. 9, no. 2, pp. 2131–2139, Dec. 2019, ISSN: 2249-8958. DOI: 10.35940/ijeat.B3601.129219.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[8] Y. Li and Q. Liu, „A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments“, *Energy Reports*, vol. 7, pp. 8176–8186, Nov. 2021, ISSN: 2352-4847. DOI: 10.1016/j.egyr.2021.08.126.

[9] G. Murad, A. Badarneh, A. Qusef, and F. Almasalha, „Software testing techniques in iot“, in *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, IEEE, Jul. 2018, pp. 17–21, ISBN: 978-1-5386-4152-1. DOI: 10.1109/CSIT.2018.8486149.

[16] „Ieee standard glossary of software engineering terminology“, *IEEE Std 610.12-1990*, pp. 1–84, Dec. 1990. DOI: 10.1109/IEEESTD.1990.101064.

[17] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, „Software testing techniques: A literature review“, in *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, IEEE, Nov. 2016, pp. 177–182, ISBN: 978-1-5090-4521-1. DOI: 10.1109/ICT4M.2016.045.

[18] „Systems and software engineering—systems and software quality requirements and evaluation (square)—guide to square“, *ISO/IEC 25000:2014(E)*, pp. 1–27, Mar. 2014.

[21] R. Mukherjee and K. S. Patnaik, „A survey on different approaches for software test case prioritization“, *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 9, pp. 1041–1054, Nov. 2021, ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2018.09.005.

[22] M. Shaw, „Prospects for an engineering discipline of software“, *IEEE Software*, vol. 7, no. 6, pp. 15–24, Nov. 1990, ISSN: 0740-7459. DOI: 10.1109/52.60586.

[23] P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyandé, and J. Klein, „Automated testing of android apps: A systematic literature review“, *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 45–66, Mar. 2019, ISSN: 0018-9529. DOI: 10.1109/TR.2018.2865733.

[24] M. A. Umar and C. Zhanfang, „A comparative study of dynamic software testing techniques“, *International Journal of Advanced Networking and Applications*, vol. 12, no. 03, pp. 4575–4584, 2020, ISSN: 0975-0290. DOI: 10.35444/IJANA.2020.12301.

[25] „Iso/iec/ieee international standard - software and systems engineering –software testing –part 1:general concepts“, *ISO/IEC/IEEE 29119-1:2022(E)*, pp. 1–60, Jan. 2022. DOI: 10.1109/IEEESTD.2022.9698145.

[26] S. L. Jurj, R. Rotar, F. Opritoiu, and M. Vladutiu, „White-box testing strategy for a solar tracking device using nodemcu lua esp8266 wi-fi network development board module“, in *2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, IEEE, Oct. 2018, pp. 53–60, ISBN: 978-1-5386-5577-1. DOI: 10.1109/SIITME.2018.8599250.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[27] Z. Kaprocki, V. Pekovic, and G. Velikic, „Combined testing approach: Increased efficiency of black box testing“, in *2015 IEEE 1st International Workshop on Consumer Electronics (CE WS)*, IEEE, Mar. 2015, pp. 76–78, ISBN: 978-1-5090-4268-5. DOI: 10.1109/CEWS.2015.7867160.

[30] S Yoo and M Harman, „Regression testing minimization, selection and prioritization: A survey“, *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, Mar. 2012, ISSN: 0960-0833. DOI: 10.1002/stvr.430.

[31] Y. Amannejad, V. Garousi, R. Irving, and Z. Sahaf, „A search-based approach for cost-effective software test automation decision support and an industrial case study“, in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, IEEE, Mar. 2014, pp. 302–311, ISBN: 978-1-4799-5790-3. DOI: 10.1109/ICSTW.2014.34.

[32] D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä, „Benefits and limitations of automated software testing: Systematic literature review and practitioner survey“, in *2012 7th International Workshop on Automation of Software Test (AST)*, IEEE, Jun. 2012, pp. 36–42, ISBN: 978-1-4673-1821-1. DOI: 10.1109/IWAST.2012.6228988.

[33] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, „Impediments for software test automation: A systematic literature review“, *Software Testing, Verification and Reliability*, vol. 27, no. 8, e1639, Dec. 2017, ISSN: 0960-0833. DOI: 10.1002/stvr.1639.

[34] R. Jongeling, J. Carlson, and A. Cicchetti, „Impediments to introducing continuous integration for model-based development in industry“, in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Aug. 2019, pp. 434–441, ISBN: 978-1-7281-3421-5. DOI: 10.1109/SEAA.2019.00071.

[35] M. Meyer, „Continuous integration and its tools“, *IEEE Software*, vol. 31, no. 3, pp. 14–16, May 2014, ISSN: 0740-7459. DOI: 10.1109/MS.2014.58.

[36] A. Deshpande, S. Veenadevi, and S. Aleti, „Test automation and continuous integration using jenkins for smart card os“, in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jul. 2021, pp. 01–05, ISBN: 978-1-7281-8595-8. DOI: 10.1109/ICCCNT51525.2021.9580021.

[38] V. Mukhin, Y. Kornaga, Y. Bazaka, *et al.*, „The testing mechanism for software and services based on mike cohn's testing pyramid modification“, in *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, IEEE, Sep. 2021, pp. 589–595, ISBN: 978-1-6654-4209-1. DOI: 10.1109/IDAACS53288.2021.9660999.

[39] R. Romli, S. Sulaiman, and K. Z. Zamli, „Automatic programming assessment and test data generation a review on its approaches“, in *2010 International Symposium on Information Technology*, IEEE, Jun. 2010, pp. 1186–1192, ISBN: 978-1-4244-6715-0. DOI: 10.1109/ITSIM.2010.5561488.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[40] A. M, A. Dinkar, S. C. Mouli, S. B, and A. A. Deshpande, „Comparison of containerization and virtualization in cloud architectures“, in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, Jul. 2021, pp. 1–5, ISBN: 978-1-6654-2849-1. DOI: 10.1109/CONECCT52877.2021.9622668.

[41] Y. Li, D. Ou, C. Jiang, *et al.*, „Virtual machine performance analysis and prediction“, in *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*, IEEE, Nov. 2020, pp. 1–5, ISBN: 978-1-7281-7315-3. DOI: 10.1109/CCCI49893.2020.9256518.

[42] A. Abuabdo and Z. A. Al-Sharif, „Virtualization vs. containerization: Towards a multithreaded performance evaluation approach“, in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, IEEE, Nov. 2019, pp. 1–6, ISBN: 978-1-7281-5052-9. DOI: 10.1109/AICCSA47632.2019.9035233.

[43] R. Dua, A. R. Raja, and D. Kakadia, „Virtualization vs containerization to support paas“, in *2014 IEEE International Conference on Cloud Engineering*, IEEE, Mar. 2014, pp. 610–614, ISBN: 978-1-4799-3766-0. DOI: 10.1109/IC2E.2014.41.

[44] A. M, A. Dinkar, S. C. Mouli, S. B, and A. A. Deshpande, „Comparison of containerization and virtualization in cloud architectures“, in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, Jul. 2021, pp. 1–5, ISBN: 978-1-6654-2849-1. DOI: 10.1109/CONECCT52877.2021.9622668.

[45] P. P. W. Pathirathna, V. A. I. Ayesha, W. A. T. Imihira, W. M. J. C. Wasala, N. Kodagoda, and E. A. T. D. Edirisinghe, „Security testing as a service with docker containerization“, in *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, IEEE, Dec. 2017, pp. 1–7, ISBN: 978-1-5386-4602-1. DOI: 10.1109/SKIMA.2017.8294109.

[46] S. Miller, T. Siems, and V. Debroy, „Kubernetes for cloud container orchestration versus containers as a service (caas): Practical insights“, in *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, Oct. 2021, pp. 407–408, ISBN: 978-1-6654-2603-9. DOI: 10.1109/ISSREW53611.2021.00110.

[47] „Systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models“, *ISO/IEC 25010:2011*, pp. 1–34, Mar. 2011.

[48] V. Garousi, M. Felderer, and F. N. Kılıçaslan, „A survey on software testability“, *Information and Software Technology*, vol. 108, pp. 35–64, Apr. 2019, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2018.12.003.

[49] J. Voas and K. Miller, „Software testability: The new verification“, *IEEE Software*, vol. 12, no. 3, pp. 17–28, May 1995, ISSN: 0740-7459. DOI: 10.1109/52.382180.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[50] A. Lal and G. Kumar, „Intelligent testing in software industry“, in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jul. 2021, pp. 01–06, ISBN: 978-1-7281-8595-8. DOI: 10.1109/ICCCNT51525.2021.9580012.

[52] R. Srivastava, M. Khan, and Abdullah, „Flexibility: A key factor to testability“, *International Journal of Advanced Information Science and Technology (IJAIST)*, vol. 6, no. 1, pp. 89–99, Jan. 2015, ISSN: 0976-2221. DOI: 10.5121/ijsea.2015.6108.

[53] R. Srivastava, M. Khan, and Abdullah, „Modifiability: A key factor to testability“, *International Journal of Advanced Information Science and Technology (IJAIST)*, vol. 3, no. 6, pp. 85–94, Jun. 2014, ISSN: 2319-2682. DOI: 10.15693/ijaist/2014.v3i6.81-84.

[54] R. Poston, J. Patel, and J. Dhaliwal, „A software testing assessment to manage project testability“, in *ECIS 2012 Proceedings*, Association for Information Systems, May 2012, 219—, ISBN: 978-84-88971-54-8.

[55] „Ieee/iso/iec international standard for software, systems and enterprise–architecture description“, *ISO/IEC/IEEE 42010:2022(E)*, pp. 1–74, Nov. 2022. DOI: 10.1109/IEEESTD.2022.9938446.

[58] A. Razzaq, „A systematic review on software architectures for iot systems and future direction to the adoption of microservices architecture“, *SN Computer Scienc*, vol. 1, no. 6, p. 350, Oct. 2020, ISSN: 2662-995X. DOI: 10.1007/s42979-020-00359-w.

[59] T. Cerny, A. S. Abdelfattah, V. Bushong, A. Al Maruf, and D. Taibi, „Microservice architecture reconstruction and visualization techniques: A review“, in *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE, Aug. 2022, pp. 39–48, ISBN: 978-1-6654-7534-1. DOI: 10.1109/SOSE55356.2022.00011.

[60] H. Reza and N. Mazumder, „A secure software architecture for mobile computing“, in *2012 Ninth International Conference on Information Technology - New Generations*, IEEE, Apr. 2012, pp. 566–571, ISBN: 978-0-7695-4654-4. DOI: 10.1109/ITNG.2012.122.

[61] A. Pirker and N. H. Lechner, „Designing secure architecture of health software using agile practices“, in *30th Central European Conference on Information and Intelligent Systems*, University of Zagreb, Faculty of Organization and Informatics Varaždin, Oct. 2019, pp. 269–280.

[63] J Arnaud and J. Wright, „Network segregation in the digital substation“, in *13th International Conference on Development in Power System Protection 2016 (DPSP)*, IET, 2016, p. 4, ISBN: 978-1-78561-138-4. DOI: 10.1049/cp.2016.0056.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[65] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, „A comparative survey of symmetric and asymmetric key cryptography“, in *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, IEEE, Nov. 2014, pp. 83–93, ISBN: 978-1-4799-5748-4. DOI: 10.1109/ICECCE.2014.7086640.

[66] P. Szalachowski, „Evla: Extended-validation certificates with location assurance“, in *Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure*, Association for Computing Machinery, Jul. 2019, 73—79, ISBN: 9781450367868. DOI: 10.1145/3327960.3332379.

[68] K. R. Ferraiolo D., „Role-based access controls“, in *15th National Computer Security Conference*, National Institute of Standards and Technology, Oct. 1992, pp. 554–563.

[69] Q. Zhang, „An overview and analysis of hybrid encryption: The combination of symmetric encryption and asymmetric encryption“, in *2021 2nd International Conference on Computing and Data Science (CDS)*, IEEE, Jan. 2021, pp. 616–622, ISBN: 978-1-6654-0428-0. DOI: 10.1109/CDS52072.2021.00111.

[70] Z. Meng and Y. Wang, „Asymmetric encryption algorithms: Primitives and applications“, in *2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)*, IEEE, May 2022, pp. 876–881, ISBN: 978-1-7281-8115-8. DOI: 10.1109/ICETCI55101.2022.9832032.

[71] Y. Guo, P. Wu, W. Huang, Y. Zhang, and J. Meng, „A secure and efficient hybrid encryption scheme for power regulation and control business“, in *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, IEEE, Dec. 2022, pp. 149–154, ISBN: 978-1-6654-7968-4. DOI: 10.1109/IMCEC55388.2022.10020036.

[72] A. S. Wazan, R. Laborde, D. W. Chadwick, *et al.*, „On the validation of web x.509 certificates by tls interception products“, *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 227–242, Jan. 2022, ISSN: 1545-5971. DOI: 10.1109/TDSC.2020.3000595.

[73] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza, „Pki4iot: Towards public key infrastructure for the internet of things“, *Computers Security*, vol. 89, p. 101 658, Feb. 2020, ISSN: 0167-4048. DOI: 10.1016/j.cose.2019.101658.

[74] J. Zhu, C. Wan, P. Nie, Y. Chen, and Z. Su, „Guided, deep testing of x.509 certificate validation via coverage transfer graphs“, in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, Sep. 2020, pp. 243–254, ISBN: 978-1-7281-5619-4. DOI: 10.1109/ICSME46990.2020.00032.

[75] M. Zulfiqar, M. U. Janjua, M. Hassan, T. Ahmad, T. Saleem, and J. W. Stokes, „Tracking adoption of revocation and cryptographic features in x.509 certificates“, *International Journal of Information Security*, vol. 21, no. 3, pp. 653–668, Jun. 2022, ISSN: 1615-5262. DOI: 10.1007/s10207-021-00572-5.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[76] B. Zou, G. Zhao, H. Tang, R. Nie, R. Huang, and J. Tang, „Archiveschain:distributed pki archives system“, in *2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, IEEE, Mar. 2021, pp. 1009–1013, ISBN: 978-1-6654-1596-5. DOI: 10.1109/AEMCSE51986.2021.00206.

[77] J. Graefe, L. Leonardon, and M. Esmael, „Using trusted responders in constrained aviation environments to reduce authentication overhead“, in *2023 Integrated Communication, Navigation and Surveillance Conference (ICNS)*, IEEE, Apr. 2023, pp. 1–4, ISBN: 979-8-3503-3362-6. DOI: 10.1109/ICNS58246.2023.10124308.

[78] H. P. Shitole and P. Divekar, „Secure email software using e-smtp“, *International Research Journal of Engineering and Technology (IRJET)*, vol. 6, no. 3, pp. 3967–3971, Mar. 2019, ISSN: 2395-0072.

[82] M. Bures, T. Cerny, and B. S. Ahmed, „Internet of things: Current challenges in the quality assurance and testing methods“, in *International Conference on Information Science and Applications*, Springer Science and Business Media LLC, Jul. 2018, pp. 625–634, ISBN: 978-981-13-1056-0. DOI: 10.1007/978-981-13-1056-0_61.

[83] A. K. Gomez and S. Bajaj, „Challenges of testing complex internet of things (iot) devices and systems“, in *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, IEEE, Oct. 2019, pp. 1–4, ISBN: 978-1-7281-3003-3. DOI: 10.1109/KSE.2019.8919324.

[84] M. Waseem, P. Liang, G. Márquez, and A. D. Salle, „Testing microservices architecture-based applications: A systematic mapping study“, in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, Dec. 2020, pp. 119–128, ISBN: 978-1-7281-9553-7. DOI: 10.1109/APSEC51365.2020.00020.

[85] J. A. Fadhil and Q. I. Sarhan, „A survey on internet of things (iot) testing“, in *2022 International Conference on Computer Science and Software Engineering (CSASE)*, IEEE, Mar. 2022, pp. 77–83, ISBN: 978-1-6654-2632-9. DOI: 10.1109/CSASE51777.2022.9759705.

[86] M. Söylemez, B. Tekinerdogan, and A. K. Tarhan, „Challenges and solution directions of microservice architectures: A systematic literature review“, *Applied Sciences*, vol. 12, no. 11, p. 5507, Jun. 2022, ISSN: 2076-3417. DOI: 10.3390/app12115507.

[90] J. Seawright and J. Gerring, „Case selection techniques in case study research: A menu of qualitative and quantitative options“, *Political Research Quarterly*, vol. 61, no. 2, pp. 294–308, Jun. 2008, ISSN: 1065-9129. DOI: 10.1177/1065912907313077.

[91] P. Runeson and M. Höst, „Guidelines for conducting and reporting case study research in software engineering“, *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Apr. 2009, ISSN: 1382-3256. DOI: 10.1007/s10664-008-9102-8.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[92] I. M. Lopes and P. Oliveira, „Implementation of the general data protection regulation: A survey in health clinics“, in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, Jun. 2018, pp. 1–6, ISBN: 978-989-98434-8-6. DOI: 10.23919/CISTI.2018.8399156.

[94] T. Hyla and J. Pejaś, „Ehealth integrity model based on a permissioned blockchain“, in *2019 Cybersecurity and Cyberforensics Conference (CCC)*, IEEE, May 2019, pp. 172–177, ISBN: 978-1-7281-2600-5. DOI: 10.1109/CCC.2019.00013.

Book References

[10] P. E. Teichreber, *Praktische Software-Qualitätssicherung: Leitfaden für Testorganisation und -dokumentation*, 1. Auflage. Düsseldorf, Germany: Symposion, 2008, ISBN: 3939707406.

[11] J. Gao, H.-S. Tsao, and Y. Wu, *Testing and quality assurance for component-based software*. Norwood, MA: Artech House, 2003, ISBN: 1-58053-480-5.

[12] U. Kuckartz and S. Rädiker, *Qualitative Inhaltsanalyse. Methoden, Praxis, Computerunterstützung* (Grundlagentexte Methoden), 5. Auflage. Weinheim Basel: Beltz Juventa, 2022, ISBN: 978-3-7799-6231-1.

[13] R. K. Yin, *Case Study Research and Applications: Design and Methods*, 6th edition. Los Angeles et al.: SAGE Publications, 2018, ISBN: 1506336167.

[14] P. Mayring, *Qualitative Inhaltsanalyse: Grundlagen und Techniken*, 13., überarbeitete Auflage. Weinheim: Beltz, 2022, ISBN: 978-3-407-25898-4.

[19] A. Spillner and T. Linz, *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard* (iSQI-Reihe), 6. überarbeitete und aktualisierte Auflage. Heidelberg: dpunkt.verlag, 2019, ISBN: 3864905834.

[20] P. Morgan and B. Hambling, *Software testing: an ISTQB-BCS certified tester foundation guide*, 4th edition. Swindon, United Kingdom: BCS Learning and Development, 2019, ISBN: 1523148268.

[28] F. Witte, *Testmanagement und Softwaretest: Theoretische Grundlagen und praktische Umsetzung*, 2. erweiterte Auflage. Wiesbaden: Springer Fachmedien Wiesbaden Imprint: Springer Vieweg, 2019, ISBN: 978-3-658-25086-7. DOI: 10.1007/978-3-658-25087-4.

[29] T. Cleff, *Basiswissen Testen von Software: Vorbereitung zum Certified Tester (Foundation Level) nach ISTQB-Standard* (Informatik). Herdecke [u.a.]: W3L-Verl., 2010, ISBN: 3868340122.

[37] M. Cohn, *Succeeding with Agile: software development using Scrum* (The Addison-Wesley signature series: a Mike Cohn signature book), 2010th ed. Upper Saddle River, NJ [u.a.]: Addison-Wesley, 2010, ISBN: 0321579364.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[51] O. Musch, *Design Patterns with Java: An Introduction*, 1st ed. 2023. Wiesbaden: Springer Fachmedien Wiesbaden Imprint: Springer Vieweg, 2023, ISBN: 3658398299. DOI: 10.1007/978-3-658-39829-3.

[56] G. Starke, *Effektive Softwarearchitekturen: ein praktischer Leitfaden*, 9., überarbeitete Auflage. München: Hanser, 2020, ISBN: 3446465898. DOI: 10.3139/9783446465893.

[57] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice* (SEI Series in Software Engineering), 4. edition. Addison Wesley, 2021, ISBN: 0136886094.

[62] M. Richards, *Software architecture patterns*, 2nd edition. Sebastopol, CA: O'Reilly Media, Inc., 2022, ISBN: 1098134281.

[64] H. Geng, *Data Center Handbook: Plan, Design, Build, and Operations of a Smart Data Center*. Newark: Wiley, 2021, ISBN: 9781119597551. DOI: 10.1002/9781119597537.

[67] D. Hercog, *Communication Protocols: Principles, Methods and Specifications*. Cham: Springer Science and Business Media LLC, 2020, ISBN: 978-3-030-50404-5. DOI: 10.1007/978-3-030-50405-2.

[79] X.-Z. Gao, S. Tiwari, M. C. Trivedi, and K. K. Mishra, *Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2019*, 1st ed. 2021. Singapore: Springer Singapore Imprint: Springer, 2021, ISBN: 9811512752. DOI: 10.1007/978-981-15-1275-9.

[80] M. Cross, J. A. Martin, T. A. Walls, M. Grasdal, D. L. Shinder, and T. W. Shinder, *MMCSE Planning, Implementing, and Maintaining a Microsoft Windows Server 2003 Active Directory Infrastructure (Exam 70-294): Study Guide and DVD Training System*. Rockland: Syngress Publishing, 2003, ISBN: 9781931836944.

[81] J. F. Eric Conrad Seth Misenar, *CISSP Study Guide*, 4th edition. Rockland, MA: Syngress, 2023, ISBN: 978-0-443-18734-6. DOI: 10.1016/C2022-0-00490-6.

[87] S. Misoch, *Qualitative Interviews*, 2., erweiterte und aktualisierte Auflage. Berlin: De Gruyter Oldenbourg, 2019, ISBN: 3110545861. DOI: 10.1515/9783110545982.

[88] U. Froschauer and M. Lueger, *Das qualitative Interview: zur Praxis interpretativer Analyse sozialer Systeme* (UTB 2418 Soziologie, Wirtschaftswissenschaften), 2., vollständig überarbeitete und erweiterte Auflage. Wien: facultas, 2020, ISBN: 9783825252809. DOI: 10.36198/9783838552804.

[89] A. Bogner, B. Littig, and W. Menz, *Interviews mit Experten: eine praxisorientierte Einführung* (Qualitative Sozialforschung). Wiesbaden: Springer VS, 2014, ISBN: 3531194151. DOI: 10.1007/978-3-531-19416-5.

Online References

[15] ISTQB. „Istqb glossary". (N.A.), [Online]. Available: https://glossary.istqb.org/ (visited on 08/11/2023).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[93] B. Group. „Bsi tr-03161 security requirements for ehealth applications“. (2020), [Online]. Available: https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03161/tr-03161.html (visited on 08/23/2023).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Appendix

Appendix A – Deductive Category System with Main Categories

Main category	Definition
Key material and security certificates	All text passages that mention challenges concerning testing HSSA that arise from using key material and security certificates. They refer to virtual and physical certificates.
Logical network separation	All text passages that mention challenges concerning testing HSSA that result from logical network separation (combination of network segmentation and segregation).
Spatial separation from the hardware	All text passages that mention challenges concerning testing HSSA that arise due to a spatial separation from the hardware.
Many different secure communication protocols	All text passages that mention challenges concerning testing HSSA that arise from using many different secure communication protocols. Examples of secure communication protocols are HTTPS, FTPS, and SMTPS.
Testing with different authorization roles	All text passages that mention challenges concerning testing HSSA that arise from using different authorization roles. A typical distinction is between an admin and a not-admin role.
Test automation	All text passages that indicate challenges that arise in test automation in the area of HSSA.
Interoperability testing	All text passages that point out challenges that arise when testing the interoperability of different components in the area of HSSA.
Testing infrastructure	All text passages that refer to challenges that arise when setting up the testing infrastructure in the area of HSSA. The testing infrastructure includes the test environment, tools, and other components.
Test data	All text passages that point out challenges that arise when dealing with test data in the area of HSSA.

Deductive category system with main categories

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Appendix B – Demographic Data Challenges Expert Interviews

Variable	Expert 1	Expert 2	Expert 3	Expert 4
Age	25-34	35-44	35-44	25-34
Gender	Male	Male	Male	Male
Highest degree or level of education	B.Sc.	PhD	High school diploma	B.Sc.
Working in industrial area (in years)	5 - 10	11 - 20	5 - 10	11 - 20
Working in scientific area (in years)	< 5	11 - 20	not working in scientific area	not working in scientific area
Experience in software development (in years)	7	17	20	17
Experience in software testing (in years)	5	17	5	12
Working in area of HSSA (in years)	5	3	5	4
Current role	Software tester with a focus on test automation and manual testing	Technical lead, software architect	Software developer with a focus on testing, IT infrastructure manager	Software developer with a focus on test automation
Experience in test automation	2 (experienced)	2 (experienced)	2 (experienced)	1 (very experienced)

Demographic data challenges expert interviews

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Appendix C – Challenges Interview Guideline

Process

- *Thanking for agreeing to be interviewed*
- *Presenting myself*
- *Explain questionnaire structure*
- *Obtain consent for audio recording*
- *Ask demographic questions*
- *Mention that all questions regarding challenges relate to the practical experience of the experts on the field of HSSA*
- *Ask content questions*
- *Thanking for interview*

Demographic questions

1. *I consent to screen/ audio recordings being made of this interview session for evaluation purposes.*
    - Yes
    - No
2. *How old are you?*
    - 18 - 24
    - 25 - 34
    - 35 - 44
    - 45 - 54
    - 55 - 64
    - Over 64
3. *What is your gender?*
    - Female
    - Male
    - Other
4. *What is the highest degree or level of education you have completed?*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- High school degree
- Bachelor's degree
- Master's degree
- Ph.D. or higher

5. *Are you currently working in the industrial area?*
   - Yes
   - No
6. *If yes, how long have you been working in the industrial area?*
   - Less than 5 Years
   - 5 - 10 Years
   - 11 - 20 Years
   - More than 20 Years
7. *Are you currently working in the scientific area?*
   - Yes
   - No
8. *If yes, how long have you been working in the scientific area?*
   - Less than 5 Years
   - 5 - 10 Years
   - 11 - 20 Years
   - More than 20 Years
9. *What is your experience in (general) software development in years?*
10. *What is your experience in software testing in years?*
11. *How would you rate your experience with test automation on a scale from 1 (one) to 4 (four) with 1 being very experienced and 4 not experienced?*

Very experienced			**Not experienced**
1	2	3	4

12. *How long have you been working in the area of High-Security Software Architectures in years?*
13. *What is/ are your role(s) in this area?*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Content questions

1. *Which mechanisms make software architectures secure in this area?*
2. *How challenging do you consider the use of key material and security certificates in terms of testing HSSA?*

Very challenging			**Not challenging**
1	2	3	4

3. *Which challenges in terms of testing HSSA arise from the use of key material and security certificates?*
4. *How challenging do you consider a logical network separation in terms of testing HSSA?*

Very challenging			**Not challenging**
1	2	3	4

5. *Which challenges in terms of testing HSSA arise from a logical network separation?*
6. *How challenging do you consider a spatial separation from the hardware in terms of testing HSSA?*

Very challenging			**Not challenging**
1	2	3	4

7. *Which challenges in terms of testing HSSA arise from a spatial separation from the hardware?*
8. *How challenging do you consider the use of many different secure communication protocols in terms of testing HSSA?*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Very challenging			**Not challenging**
1	2	3	4

9. *Which challenges in terms of testing HSSA arise from the use of many different secure communication protocols?*

10. *How challenging do you consider testing HSSA with different authorization roles?*

Very challenging			**Not challenging**
1	2	3	4

11. *Which challenges in terms of testing HSSA arise from the use of different authorization roles?*

12. *How challenging do you consider test automation in the area of HSSA?*

Very challenging			**Not challenging**
1	2	3	4

13. *Which challenges arise with respect to test automation in the area of HSSA?*

14. *How challenging do you consider testing the interoperability between components of a HSSA?*

Very challenging			**Not challenging**
1	2	3	4

15. *Which challenges arise with respect to testing the interoperability between components of a HSSA?*

16. *How challenging do you consider the testing infrastructure of HSSA?*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Very challenging			**Not challenging**
1	2	3	4

17. *Which challenges arise with respect to the testing infrastructure of HSSA?*

18. *How challenging do you consider the handling of test data when testing HSSA?*

Very challenging			**Not challenging**
1	2	3	4

19. *Which challenges arise with respect to the handling of test data when testing HSSA?*

20. *Which other challenges not yet mentioned do you see in testing HSSA?*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Appendix D – Evaluation Interview Guideline

Process

- *Thanking for agreeing to be interviewed*
- *Presenting myself*
- *Explain questionnaire structure*
- *Obtain consent for audio recording*
- *Ask quantitative and qualitative questions*
- *Thanking for interview*

Content questions

Each concept and the corresponding mapped challenging category is explained to the experts.

- *Certificate creation and manipulation- static approach*
- *Certificate creation and manipulation - dynamic approach*
- *PKI simulators*
- *Implementation of encryption and decryption methods in test automation*
- *Usage of libraries when testing secure communication protocols*
- *Testing in several layers (with and without transport encryption)*
- *Simulators*
- *Semi-automatic testing with test groups*
- *On-demand setup of a test environment*
- *Testing in different environments*
- *Log reporting and log data aggregation*
- *Real time log analysis*
- *Containerization and virtualization*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

For each presented concept, the following questions are asked:

Very high **Very low**

1	2	3	4

1. *On a scale of 1 to 4, with 1 being very high and 4 being very low, how would you rate the overall quality and transferability of the concept? (optional)*
2. *What do you consider to be good in the concept? (optional)*
3. *What do you consider to be lacking in the concept? (optional)*
4. *What do you consider to be not transferable in the concept? (optional)*

Highly valid **Not valid**

1	2	3	4

5. *On a scale of 1 to 4, with 1 being highly valid and 4 being not valid, do you consider the presented concept as a valid solution to the mapped challenge category? (optional)*
6. *Why do you consider the concept (not) to be a valid solution to the mapped challenge category? (optional)*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.