

Project References

PROJECT ACRONYM	XIVT		
PROJECT TITLE	EXCELLENCE IN VARIANT TESTING		
PROJECT NUMBER	17039		
PROJECT START DATE	NOVEMBER 1, 2018	PROJECT DURATION	36 MONTHS
PROJECT MANAGER	GUNNAR WIDFORSS, BOMBARDIER TRANSPORTATION AN ALSTOM GROUP COMPANY, SWEDEN		
WEBSITE	HTTPS://WWW.XIVT.ORG/		

Document References

WORK PACKAGE	WP2: KNOWLEDGE-BASED TEST OPTIMIZATION
WORK PACKAGE LEADER	MEHRDAD SAADATMAND - RISE RESEARCH INSTITUTES OF SWEDEN
DELIVERABLE	D2.5: TEST OPTIMIZATION METHODOLOGY FOR PRODUCT VARIANTS: FINAL VERSION
DELIVERABLE TYPE	REPORT DOCUMENT (DOC)
DISSEMINATION LEVEL	PUBLIC

DATE & VERSION	FEB 28, 2022 - FINAL
MAPPED TASKS	<p>T2.1: KNOWLEDGE-BASED TESTING FOR PRODUCT VARIANTS</p> <p>T2.2: TEST OPTIMIZATION CRITERIA FOR VARIANT-INTENSIVE PRODUCTS</p> <p>T2.3: TEST OBJECT- AND FEATURE- BASED OPTIMIZATION</p> <p>T2.4: TEST PRIORITIZATION AND SELECTION FOR VARIANT-INTENSIVE SYSTEMS</p>
CONTRIBUTING PARTNERS	<p>RISE (SWE), PERCEPIO AB (SWE), IFAK (DEU), FCUL (PRT), MOBILELIVE(CAN), EXPLEO (DEU), WINTRUST (PRT), FRAUNHOFER FOKUS (DEU), ADDIVA AB (SWE), ARD GROUP (TUR), EKS INTEC (DEU), QA CONSULTANTS (CAN), MÄLARDALEN UNIVERSITY (SWE), UNIT INFORMATION TECH (TUR), ONTARIO TECH UNIVERSITY (CAN), ABB AB (SWE), ARCELIK (TUR), BOMBARDIER (SWE), FFT (DEU), MES (DEU), TURKCELL (TUR)</p>

Version History

DATE	TITLE	VERSION	DISSEMINATION LEVEL
FEB 28, 2022	TEST OPTIMIZATION METHODOLOGY FOR PRODUCT VARIANTS - FINAL VERSION	FINAL	PUBLIC
OCT 31, 2020	TEST OPTIMIZATION METHODOLOGY FOR PRODUCT VARIANTS - INTERMEDIATE VERSION	INTERMEDIATE	CONFIDENTIAL
Nov 30, 2019	TEST OPTIMIZATION METHODOLOGY FOR PRODUCT VARIANTS - INITIAL VERSION	INITIAL	CONFIDENTIAL

Table of Contents

Executive Summary	4
1. Knowledge-based testing - scope and definition	5
2. State of Test Optimization in XIVT	5
2.1. State of Practice in Test Optimization in XIVT Domains.....	5
2.2. State of solutions for Test Optimizations.....	9
2.3. Limitations on product variant systems and improvements	10
3. XIVT Test optimization methodology for product variants.....	14
3.1. Test optimization criteria for variant-intensive products.....	17
3.2. Test object- and feature-based optimization	20
4. XIVT-Test Optimization and supporting Methodologies	23
4.1. Variability-Aware Reuse Analysis method by RISE.....	24
4.2. NLP-based Design Specification Recommendation System for Requirements by FCUL	25
4.3. NLP-based Requirements Formalization for Automatic Test Case Generation approach by ifak.....	27
4.4. Similarity-based Requirement Formalization using Product Design Specifications by ifak	29
4.5. Combinatorial Interaction Testing methods by MDU	30
4.6. Clone detection in model-based design by Addiva, MDH and RISE.....	34
4.7. Budget-based Test Selection and Prioritization approach by FOKUS	35
4.8. Risk-based test scoring method by QAC	38
4.9. Test Selection and Prioritization at Turkcell	42
5. Knowledge-Based Testing: State of the art and gap analysis.....	44
5.1. Change Impact Analysis (CIA).....	44
5.1.1. CIA Based on Requirements.....	45
5.1.2. CIA Based on Use Case Models	46
5.1.3. CIA Based on Source Code	47
5.1.4. CIA Based on UML Models	48
5.1.5. CIA Based on Feature Models	49
5.1.6. CIA Based on Architectural Models.....	50
5.1.7. Risk estimation based on CIA	50
5.2. Test process optimization	52
5.2.1. Cursory Survey of Search based Test Case Optimization Techniques.....	52
5.3. Product Line Testing.....	54
5.4. Machine Learning and Knowledge-Based Testing.....	55

5.5. Model-Based Testing.....	56
5.6. Combinatorial Testing.....	57
5.7. Variant Modeling.....	57
5.8. Software Security Testing.....	58
5.9. Risk assessment of critical product variant systems	59
5.9.1. Safety risk assessment	60
5.9.1.1. Automotive domain.....	60
5.9.1.2. Rail domain	62
5.9.1.3. Industrial domain	65
5.9.1.4. Telecom domain	66
5.9.2. Security risk assessment.....	66
5.9.2.1. Automotive domain.....	66
5.9.2.2. Rail domain	68
5.9.2.3. Industrial domain	70
5.9.2.4. Telecom domain	71
5.9.3. Summary of safety and security risk assessment.....	73
6. Conclusions	74
References	75

Executive Summary

There are particular challenges and questions that need to be addressed when it comes to testing of variant-intensive systems. For instance, when a new product variant is instantiated from the baseline, it may not be trivial to decide whether the common features between the product variant and the baseline that were tested previously as part of the baseline testing process need to be re-tested or not to ensure the quality and correctness of the variant. Similarly, it is important to be able to determine that when a bug is detected or fixed in a product variant, how far back into the baseline and also which of the other variants we should choose to re-test to ensure that the bug does not propagate in other variants and is correctly fixed. Re-executing all the test cases in such scenarios can lead to suboptimal use of the testing resources.

Work-Package 2 (WP2) in XIVT focuses on answering such questions that are critical for optimal use of testing resources by providing tailored solutions for optimization of the test process of variant-intensive systems. In particular, WP2 provides solutions for test case selection and prioritization to enable optimization of the testing process in such systems by taking into account the criteria that are relevant for product lines and variants, such as feature coverage. It is also important to know the current state-of-the-art and state-of-practice in test optimization, particularly to identify coverage criteria per XIVT domains. This document also summarizes the results of an internal survey on state of test optimization in XIVT use case/tools providers.

The activities in WP2 are done as part of four tasks contributing to the development of the overall XIVT Test Optimization Methodology: Knowledge-based testing for product variants (T2.1), Test optimization criteria for variant-intensive products (T2.2), Test object- and feature- based optimization (T2.3), Test prioritization and selection for variant-intensive systems (T2.4). The latest results of these tasks at each reporting period serve as and constitute the inputs to WP2 deliverables.

Deliverables D2.1, D2.3, and D2.5 in WP2 report on the results and development of the aforementioned solutions in the project that constitute the overall XIVT Test Optimization Methodology (TOM) throughout the project. This is done in increments where this current deliverable, D2.5, contains the final version, by building on the previous results reported in D2.1 and D2.3. The software products and solutions in WP2 are delivered and reported as separate deliverables (D2.2, D2.4, D2.6).

1. Knowledge-based testing - scope and definition

In the context of the XIVT project and this deliverable, we use the term knowledge-based testing in a broad sense to refer to different testing techniques, methods, and approaches that rely on and require (large-scale) processing of information from sources of various types (e.g., textual requirements specifications along with code, etc.). This is particularly considered important since variability and different versions of a product could be created and brought about due to, for example, having a modified and different set of requirements (e.g., for different regions), different standards applied on a common baseline resulting in different product instances, re-using features and components from the baseline then extending with new features. Testing in such scenarios requires understanding the source and causes of variability, hence looking into and processing different artifacts and sources of information.

2. State of Test Optimization in XIVT¹

In this section, we present the results of a questionnaire-based survey conducted with XIVT partners. The results from the survey are divided into two parts, as follows:

- The practices of use-case owners regarding their test optimization are presented.
- The tools in the XIVT consortium are presented from the lens of test optimization.

2.1. State of Practice in Test Optimization in XIVT Domains

This sub-section summarizes the questionnaire results from the perspective of use case providers.

Nature of System-Under Test (SUT).

We gathered data about two aspects of the SUTs in XIVT, as follows. We asked if the SUT is a safety or security-critical SUT. This is essential input since many safety-critical systems have to provide evidence for compliance and thus might be required to execute the entire test suite. This can help the readers understand why sub-set selection from the test suite is not made.

In Table 1, we summarized the response of use case providers and the relevant standards to which the testing process must comply.

SUT Domain	Safety-Critical	Security-Critical	Standard to follow
Embedded software	Indirectly	No	N/A
Telecom	No	No	N/A
TV	No	No	N/A
Automotive	Yes	Yes	
Railway control	Yes	No	EN50657
Automotive (Expleo)	Not currently	No	

Table 1 : Domain and Nature of XIVT's SUTs

¹ The content of this section is based on the processed results of a survey conducted with the XIVT use-case, tools, and knowledge providers.

Test suite size.

When asked if the use-case has an existing test suite and the size, we got varying responses. All the use case providers in the XIVT project have a test suite at the product level with the test suite's varying size. One partner has a test suite of 1500 test scenarios in the telecom domain, where each test scenario might contain multiple test cases. In the automotive domain, the test suite has to capture different parts of the road map data, and thus the test suite of one of our partners in XIVT captures around 220000 miles of road. In addition, radar and stereo camera data are also used for testing. Our TV domain use-cases have a test suite of 101K test scenarios, out of which 100K test scenarios require manual execution. There are 20 test scenarios in the railway domain at the integration level, while around 600 tests for the components.

Motivation for test optimization.

Not all the tests are the same and might need ranking, prioritization, and selection. In addition, the execution of all test suite might not be possible. For example, for testing all the possible variants of Percepio's use case, 1 billion test cases may be needed. This is due to the varying configurations that could be enabled at compile time. *"A single test program, where 10-20 sanity checks are applied on the output. This, however, needs to be executed on every combination of 30-50 static build flags that affect the code in undocumented ways, meaning at least $2^{30} = 1$ billion tests if using a "brute force" approach with no test case prioritization."* In such cases, different test criteria are used to select a sub-set of from the test suite.

Test Ordering and Selection.

To gather data about the test execution order and sub-set selection, we asked the XIVT use-case partners if they rank and select a subset from their test suite. In the rail domain, the test cases are independent and thus do not require any ordering. However, test selection is performed to target the changes in the code. Nevertheless, due to the safety-critical nature of the product, the entire test suite is once again executed before the release of the software. Our automotive partners do not order test cases for execution, but one partner does select sub-sets based on the size of the test suite. For example, they might select a subset of map data from the large test suite due to time constraints. In the TV domain, test cases might depend on each other, and thus ordering is required. However, it is preferred to execute the entire test suite, but it is not possible at times, and the sub-set is selected to meet time constraints. In the telecom domain, test cases are ordered, but no test selection is made. Engineers at Percepio only test the critical sub-set of build configurations and select a critical sub-set from the test suite. Figure 1 shows that 50% (blue color) of the XIVT use case providers have to order the test cases for execution, and 57% of the partners also have some test selection in place. The current criteria for test selection are shown in Table 2.

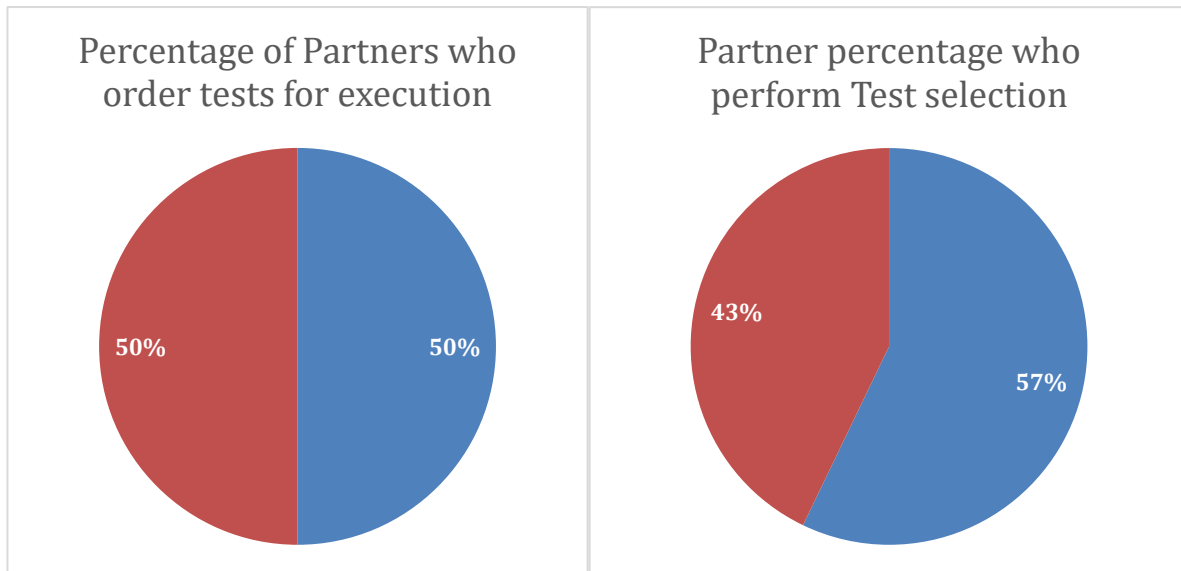


Figure 1: Test ordering for execution

S. No.	Criteria	Domain
1	Tests that target Changes	Railway
2	Tests that target critical build configurations	Percepio
3	Tests that would execute in limited time and with limited test resources	TV, Automotive

Table 2: Test selection criteria in different domains

Test categorization/classification.

In order to apply test selection effectively, it might require classifying the test cases into different categories. For example, one classification would be whether a test case is testing a safety-critical function or not. Such categorizations and classifications improve the test selection process, and thus we asked our partners if they categorize test cases. Figure 2 shows the percentage of partners who categorize (blue), do not have categorization in place but plan to have it (green), and have no categorization in place (red).

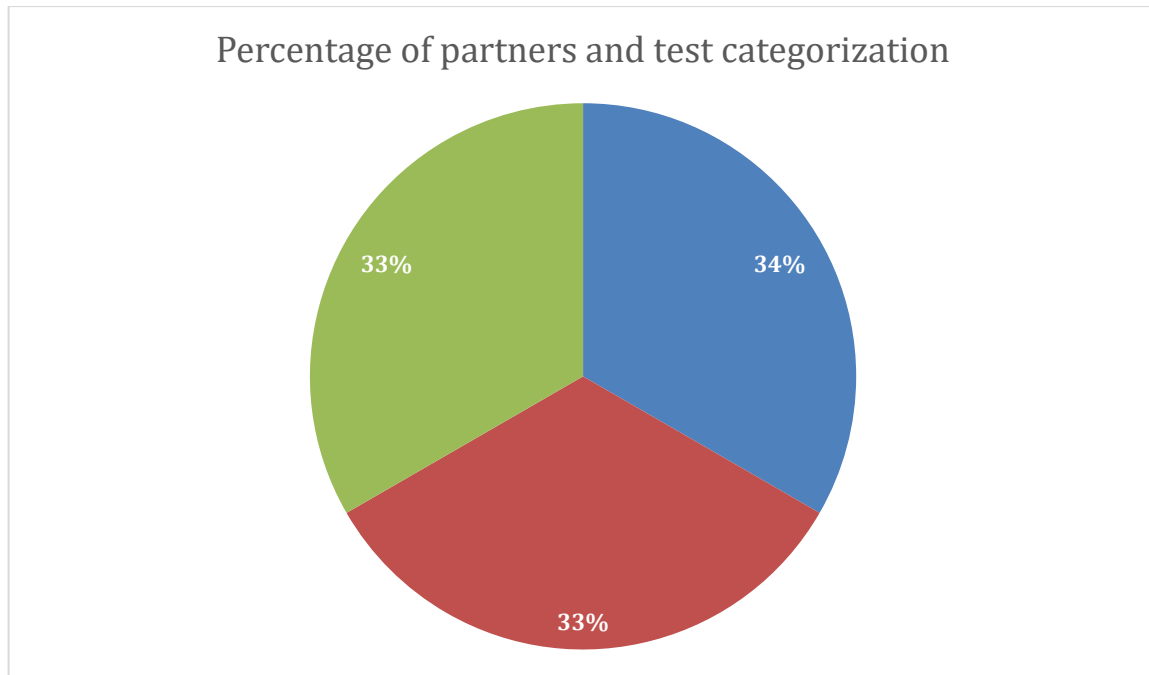


Figure 2: Statistics of partners regarding test case categorization

Tests can be categorized using different classification schemes. Table 3 shows the different categorization schemes XIVT partners are following or are planning to follow.

Domain	Scheme
Percepio	Functional and Performance
Telecom	Performance, Security, Regression, Functional, Unit, Acceptance
TV	Not known but in place
Automotive (Expleo)	Test difficulty level

Table 3 : Categorization schemes followed in XIVT's use cases for test categorization

Test Classification for Product-Line and Variants.

To report the maturity of the product-line testing process of the XIVT use case providers, we asked if the test cases could be traced to the standard and product variants. In addition, we also asked if there are separate test suites for the variants and the standard product.

In the railway and automotive domains of XIVT, they create separate test suites per variant. In all other domains, there is mainly one test suite. In almost all cases (except Expleo), the test cases can be traced to the variants using either IDs or naming conventions.

Areas of Improvement in the Current Testing Process.

To identify the possible areas of improvement, we asked the use case providers if the current testing process could be improved. Besides, we also asked if the process from the perspective of product-line testing could be improved. We performed coding on the answers to these questions and identified several improvement areas that the XIVT solution/tools providers could focus on. Table 4 shows possible areas of improvement in the current testing processes of the XIVT use-

case providers. The themes column represents the codes' central nature, and the codes represent the specific problems that need to be addressed to improve the testing process.

Themes	Codes
1. What configurations (not) to test?	1.1. Detection of product configuration that results in the same code of an already tested configuration 1.2. Analyzing the effect of configuration parameter on resultant code, that would identify similar code and thus can be skipped for testing 1.3. Detection of redundant configuration parameters 1.4. Testing approaches for composed features
2. Impact Analysis	2.1. Flagging changes that would result in test breakage 2.2. Flagging changes that would most likely result in a test failure
3. Variant Testing	3.1. Automated test script derivation for modified variants (test evolution) 3.2. Test selection for modified part of the variant 3.3. Test categorization based on the criticality 3.4. Sub-set selection from the test suite for the variants 3.5. Identification of critical scenarios
4. Test Execution	4.1. Simulation of the hardware part of the system can optimize the testing process in some XIVT use-cases
5. Coverage Criteria	5.1. Dead code detection, which effects coverage 5.2. Linking component-level coverage results to system's features

Table 4 : Areas of improvements in testing processes in XIVT use cases

2.2. State of solutions for Test Optimizations

This section summarizes the different tools in XIVT from the test optimization point of view. We asked the owner of tools in XIVT if their tools support test prioritization & selection and, if so, based on what criteria the prioritization & selection is made. In addition, we asked if the tools could support handling test cases for the product line and product variants.

Test prioritization and selection.

When we asked the XIVT partners, “do your tools support the prioritization and selection of the test cases?” we found that only tools from two partners have the functionality to select test cases. One partner is planning an extension to their tool. The criteria on which tests are selected in XIVT tools are presented in Table 5.

Tool Provider	Supported Criteria
MDH	Combinatorial Coverage Based Selection Code Coverage based Selection
Expleo	Feature Coverage Based Selection Pairwise Feature Coverage Based Selection Path Coverage Based Selection
RISE (Planned)	Delta Coverage Based Selection

Table 5 : Test case selection criteria already used in XIVT tools

Test case prioritization is supported by tools from three partners. Table 6 presents the different test prioritization criteria per partner.

Tool Provider	Supported Criteria
QAC	Risk-Based Test Prioritization
Expleo	Feature Coverage Based Selection Pair-wise Feature Coverage Based Selection Path Coverage Based Selection
RISE	Coverage to extra-functional properties

Table 6 : Test case prioritization criteria already used in XIVT tools

Test differentiation & Traceability between product line and variants.

The current XIVT tools have no functionality to differentiate between test cases coming from the product lines and variants. Besides, the tools also do not allow the mapping of test cases to the product line and variants.

2.3. Limitations on product variant systems and improvements

This section has as objective gives an overview of the limitations of the tools when they test a product baseline and its instances. To obtain the limitations, a questionnaire was made to XIVT’s use case providers as they are the ones that produce product variant systems and hence need to test them. On the other hand, the section also intends to give insights on how such tools can be improved face to the limitations found. For that, another questionnaire-based study was conducted for the solution providers, i.e., the XIVT’s partners that develop solutions for testing product variants, who apply different testing techniques and tools.

Limitations on product variant systems

We asked to the XIVT use case providers to enumerate the limitations they found in the tools they use for testing a product baseline and its instances, for example, in terms of optimizing the testing process for product variants and prioritization and selection of tests. Also, we asked them what they would like to see implemented on the tools to address such limitations (e.g., in terms of test prioritization, test/variant selection, and test optimization in general), and, in their opinion, what features are missing to make the tools useful/optimal/appropriate for testing product variants efficiently and effectively. Both questions were responded to by two use-case providers out of seven, denoting that only these two use tools for testing their products, i.e., they test their products. The next two tables show the results of these two questions.

Partner	Limitations
Bombardier Transportation on an Alstom Group company	<p>All the tools used for testing variants needs to connect to Simulink and DOORS, but they:</p> <ul style="list-style-type: none"> – are not “plug and play”, meaning that the user needs to know how they work internally to try to connect them to the others; – lack of a user guide that explains the process in a step by step process for the users working with them properly; – are not clear how they are related to the Safety work flow and if they are certified for that purpose; – are not provided with any demo showing that they are capable for using Simulink or DOORS. Currently, only the VARA tool and MDH evaluation for auto test case generation in Simulink Test.
An Automotive Manufacturer	<ul style="list-style-type: none"> – Testing speed. Current database represents 220,000 miles Lane-level information. If we test that on one (mapping) module in real-time with maximum speed that is allowed, it will take 200 days to test it. However, one way to get faster tests is to increase the number of controllers used. – Testing cost, i.e., the hardware equipment that is need to purchase for testing is quite expensive; – Store and analyze data, i.e., the hardware setup with six controllers (used currently) outputs 30GB of data daily which is stored for further analysis. Keeping this data for long-term will be a problem in a short time. Also, analyzing such amount of data brings another problem, since we will need tools and hardware able to process it and handle it, respectively.

Partner	Missing features
Bombardier Transportation on an Alstom Group company	<p>A lot of good tools but our user cases mostly see issues with how they align with our process and our toolchain. The tool provider should:</p> <ul style="list-style-type: none"> – highlight which part of the existing software verification process they are modifying; – know how to connect to our existing toolchain. – know what process must be followed with their toolchain to maintain consistent results;

	<ul style="list-style-type: none"> – be able to show clearly how they will improve the existing workflow and toolchain. E.g. if an automatic test case generation tool is created, it should first be checked against Simulink automatic test case generation tool and highlighting where this tool is more efficient. <p>As we are following a SIL2 workflow in a team of engineers focusing on control algorithms for hardware, not software development and hence all tools need to be very clear, suitable GUI should be created with the tools.</p>
An Automotive Manufacturer	<ul style="list-style-type: none"> – An optimized route to take. Test cases are generated by running a very simple algorithm to create them based on the longest drivable route. Many connections between routes are not being tested. For example, if we enter an interchange, we want to be able to make sure we can test all the variation of it (e.g. testing left turn, right turn); – The current dataset contains only highways while at some point later, it will include non-highways (e.g. Local routes). Therefore, we definitely want to be able to test various permutations and the different routes that a user can take. For example, what is the best way to test the entire route.

Insights for improving software testing tools

In order to find out what are the capabilities of solutions/tools that have been developed within XIVT and how they are measured, and what are the limitations they have and how they can be addressed, we asked to the XIVT solution providers the following questions, where the first two questions are related to the tool's capabilities, and the other two questions related to the limitations of the tool:

1. What are the capabilities of your tools/solutions in (optimizing) the testing process of variant-intensive systems in general? How is optimization achieved (e.g., test case prioritization, test case selection, generation of an optimized set of test cases, ...)?
2. Which metrics and criteria do you use to measure how much optimization and improvement is achieved in the testing process of a system using your tools?
3. What limitations do your tools/solutions have in (further) optimizing the testing process of variant systems? What is missing?
4. How can you provide tailored solutions able to address the limitations in the testing of variant-intensive systems? What extensions are needed?

The next two tables show the results of these two pairs of questions.

	Solution/tool by Solution Partner Provider					
Capability	MES	QAC	MDH	Expleo	RISE	FCUL
Generate requirement-based test cases	. MES Test Manager		. Complete test . SEAFOX	. Meran		

Generate mutation input-based test cases						. DeltaFuzzer
Test case selection			. Complete test . SEAFOX		. VARA*	. DeltaFuzzer
Test case Prioritization based on scoring		QAC's Risk - based Test Scoring				. DeltaFuzzer
Feature/pairwise prioritization				. VarSel		
Test case Prioritization based on requirements					. MBRP . VARA*	
Model variability				. Testona . Modica		
Metrics and criteria						
Model coverage	. MES Test Manager					
Code coverage	. MES Test Manager		. Complete test . SEAFOX			. DeltaFuzzer
Combinatorial coverage			. Complete test . SEAFOX			
Path coverage				. Modica		. DeltaFuzzer
Feature/pairwise coverage				. VarSel . Testona		
Requirements coverage	. MES Test Manager				. MBRP	

Partner	Limitations	Tailored solutions
MES	not target variant intensive systems yet	<ul style="list-style-type: none"> - formal requirements for safety critical real time systems - requirements based test case generation+ test instantiation and execution in the MATLAB/Simulink Ecosystem - providing structured reports for test results with detailed tracing between test artifacts
QAC	- score test cases in terms of security risks	In order to enable risk-based scoring:

	- procedure to score different security risks	- a knowledge database is required for evaluating cybersecurity risks - analysis of test cases to determine safety risks
MDH	- optimizations in test execution and checking of results	- combinatorial coverage should be extended with behavioral information (e.g., SysML models) for test selection
Expleo	- VarSel: Restricted to feature models of variants. Relevant coverages such as requirement coverage are not implemented yet. - Testona and Modica: Variant-overarching test cases/doublings of test cases between variants are not implemented yet. This would enable to generate test suites for sets of variants instead of variant instances. - Meran: Variant handling should be more intuitive w.r.t. GUI.	- VarSel: error detecting capabilities would be interesting to analyze in coordination with the fault injection tool developed by FOKUS - Testona and Modica: generation of test suites for sets of variants (instead of variant instances) based on test cases between variants - Meran: In general, the variant handling of the tools still has to be synchronized in order to obtain a unified process
RISE	not target variant intensive systems directly	- VARA: rank test cases based on requirements similarities
FCUL	- code coverage - prioritization of test cases based on score	- DeltaFuzzer: extending it with specific target for better covering the code and prioritize the interesting test cases

3. XIVT Test optimization methodology for product variants²

The XIVT Test Optimization Methodology for product variants (XIVT-TOM) is developed, targeting all the XIVT industrial domains. Given input artifacts (feature model, test cases, requirements, optimization criteria etc.), XIVT-TOM is designed to automatically determine the domain (e.g., railway, telecom etc.) and apply the best-suited optimization strategy to optimize the test suite. Figure 3 shows an excerpt of the initial design of the XIVT-TOM in which Software Product Line (SPL) artifacts and TOM criteria can be used to obtain an optimized test suite.

² The content of this section is first and foremost from the results of T2.4, and then secondly and to some degree also based on the results of the other WP2 tasks.

When using an SPL artifact, application engineers make different choices about how to use the variability and feature capabilities to points of variability in their system. This is also referred to as an instance of the SPL artifact in Figure 3. In this process, they must consider whether the combination of features and the variants they have chosen has ever appeared in an instance of the SPL artifact. If this is the case, XIVT-TOM is used to give confidence that interactions among the features have been exercised by the selected test suite.

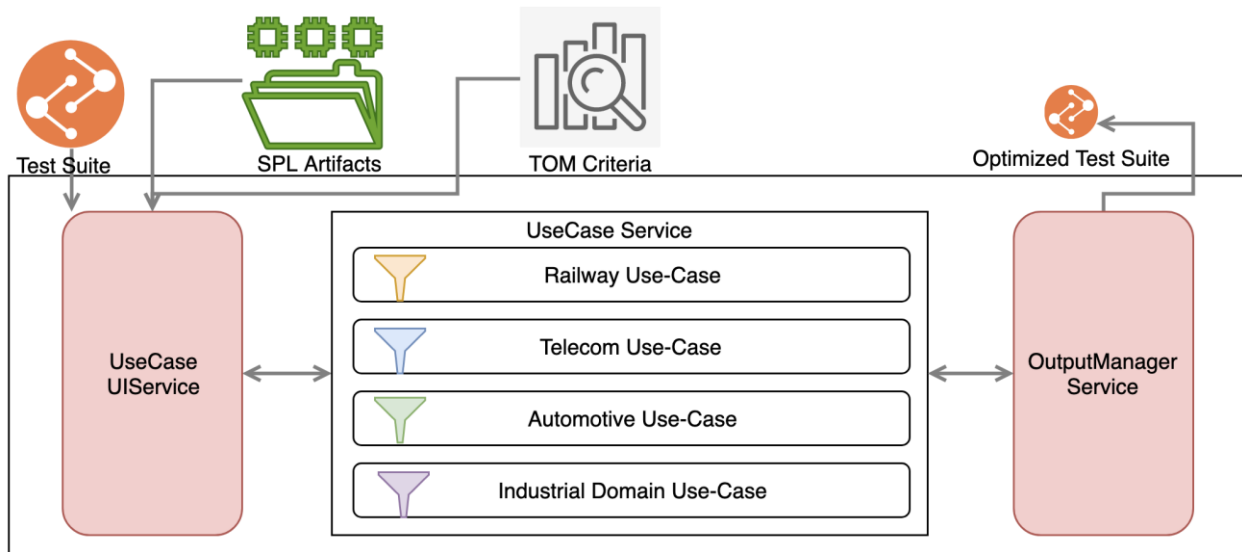


Figure 3: Excerpt of XIVT test optimization methodology.

The idea is to optimize testing to reduce the uncertainty, given that faults can arise due to such interactions. Ideally, the engineers would validate all possible combinations of features and points of variability that can be realized in an SPL instance. This would allow users to confidently instantiate the SPL to produce new systems. Unfortunately, the space of possible combinations in a realistic SPL (such as the artifacts from the Use Case Service) is enormous and exhaustive test execution of those combinations is intractable.

Each of the domains is addressed by its own use-case service. The use-case service is capable of taking a pre-processed input from the input service and then derive and optimize the test suite based on predefined criteria. Each domain-specific use-case service can take a set of input such as requirements, the models derived from requirements, or an existing test suite (if any). The service then uses natural language and machine learning-based approaches to compute similarities among the requirements. On the other hand, in parallel, the models or code are given input to the change-impact analyzer (a sub-service) to analyze commonalities and variabilities in the models/code. Extracting similarity information is essential for a high degree of reuse. The use-case service can then use commonality- and variability information to derive a test suite for the product line. The test suite, commonalities, and the requirements-level similarities are then passed

to the Criteria Evaluator to optimize the test suite. The Criteria Evaluator is used to model the variable and common parts of the SPL obtained from Natural Language Similarity Engine, which is mapped onto a TOM model. This TOM model serves as a semantical basis for defining coverage criteria for the SPL under test. Furthermore, combinatorial interaction testing (CIT) can be used to achieve the desired level of coverage and select test cases. In the use case service, the change-impact analyzer can be used to select products that should be tested systematically. Figure 3 shows an excerpt of the use case service.

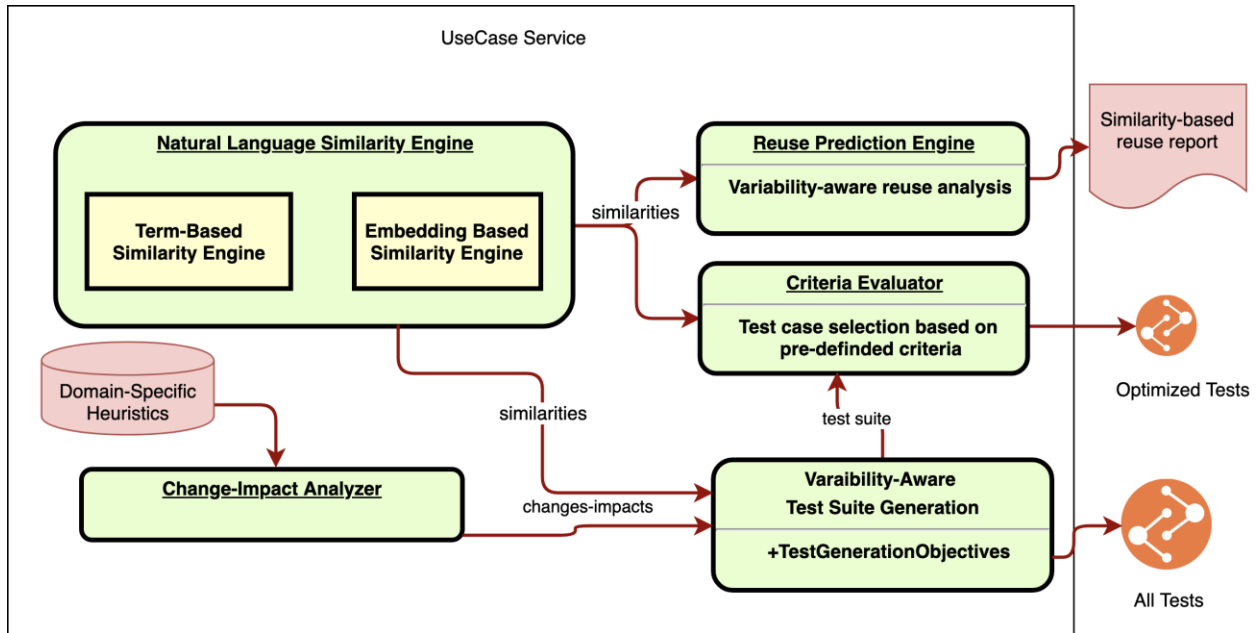


Figure 4: Excerpt of XIVT-TOM Use-Case Service.

In Figure 5, we show an example of the use of the XIVT-TOM Criteria Evaluator Service for the case of test optimization. Different artifacts representing similarities serve as input for this service. They are the results of a model-to-model transformation from abstracted representations (e.g., Feature Diagram (FD), SPL behavior, formula and/or coverage criteria used by the QA activities). The XIVT-TOM framework supports abstraction and composition mechanisms such that one can for example, use State Diagram Variability Analysis (SDVA) based on UML state machines to model the behavior of the SPL. Once the input models are transformed and used on the TOM criteria, they can be used to perform model-testing and/or other testing activities (e.g., test case execution, optimization, generation). By using a common representation of the TOM criteria, the XIVT-TOM methodology can benefit each use case and is the first step in this direction by combining coverage criteria techniques with variant testing in order to prioritize, select or minimize test suites.

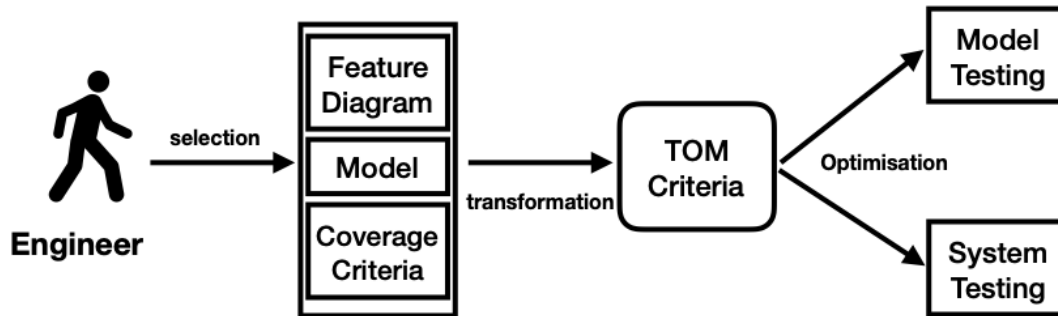


Figure 5: An example of the use of the XIVT-TOM Criteria Evaluator Service.

Note that each domain service can follow their specific implementation of the use-case service shown in Figure 4. For example, as a first step for the railway use-case, XIVT partners have developed Natural Language Similarity Engine and Reuse Prediction Engine using vectorization and clustering. This might not be the same as other XIVT domains and should be investigated further. Another example is the Change-Impact Analyzer that might have to work with models in the train industry but might have to work with source code in other XIVT domains.

3.1. Test optimization criteria for variant-intensive products³

In the XIVT methodology, we address the problem of optimizing a test suite given different SPL artifacts by defining and using families of coverage criteria (i.e., TOM Criteria). These TOM criteria allow tradeoffs to be made between the extent to which feature and configuration combinations are covered and the cost of test execution. TOM criteria are related to the XIVT-TOM methodology in such a way that optimization of SPLs can be exploited in the Criteria Evaluator to incrementally improve validation across the lifetime of an SPL artifact.

With regard to combinatorial testing in TOM, the selection of a certain degree of input parameter combinations (e.g., pairwise, three-wise) is a suitable coverage criterion. Here are some examples that are used in the XIVT-TOM methodology for variant combinatorial coverage:

- Each-value-used: 100% each-used coverage is defined for each feature of every test parameter in the test suite at least once.
- Pairwise: 100 % pairwise coverage is defined for every pair of values of any two tested features in the test suite.
- Base choice: 100 % Base choice coverage is achieved by combining the most frequently used values of each configuration or features tested within a test suite, where the rest of the values are base values as well.

³ Current results of T2.2 are the main input to this section.

The TOM Criteria focus on SPLs regardless of the source of variability in the SPL. As an example, for one category of such criteria (i.e., combinatorial coverage), we focus first on measuring the existing coverage levels achieved by existing test suites and how these can be improved using TOM combinatorial coverage criteria. The XIVT-TOM methodology uses these heuristics to estimate the quality of test suites. Test engineers for each use case in XIVT can use the TOM coverage criteria to assess whether the test activities are sufficient or not. It is important to emphasize that the idea of using XIVT-TOM is focused on using variant coverage criteria for the validation of the system and SPL artifact itself and not on specific instances of the SPL.

Given that a criterion in XIVT-TOM is chosen, reuse techniques can be used to reduce the effort in testing. Even so, each product has to be tested individually. In those XIVT use cases that show that the testing techniques lack traceability between a variation model, use cases, and test specifications in different instances, XIVT-TOM needs to be adapted. In the end, XIVT-TOM is used in suitable use cases for evaluating a subset of products to approximate a complete SPL-test according to various coverage criteria.

From running a questionnaire with all use providers, we identified different challenges on using coverage criteria for variant testing:

Coverage Criteria	<p>Difficult to apply variant coverage criteria to use during continuous development.</p> <p>Dead code detection, which effects coverage in different variants.</p> <p>Linking component-level coverage results to system's features in each variant.</p>
-------------------	---

Use case providers are using, as a basis for creating test cases for each variant, different testing techniques: specification-based testing, design-based test design and program-based test design, as well as exploratory testing. Examples of test goals and coverage criteria used are:

- ❖ equivalence partitioning coverage, boundary value analysis
- ❖ experience and risk-based cases
- ❖ statement, branch and MCDC coverage
- ❖ combinatorial coverage

In addition, we have collected data on what kind of test coverage criteria for variant testing are proposed by different technology providers:

Supported Criteria in XIVT
Combinatorial Coverage Based Selection
Code Coverage based Selection
Feature Coverage Based Selection
Pairwise Feature Coverage Based Selection
Path Coverage Based Selection
Delta Coverage Based Selection
Risk Based Test Prioritization
Coverage to extra-functional properties

For example, for specification-based testing, partners are proposing tools that support different coverage criteria based on the variant specification described, including variants and requirement coverage. When test cases are measured in terms of source code, the source code of the product line or that of a particular product is taken into account.

In order to account for the variability in the different use cases with respect to processes, existing tools and data formats, the XIVT toolchain is actually designed as a product line itself and takes these coverage criteria as an option for test generation, selection and prioritization. The following features of the XIVT toolchain are included:

1. Coverage Analysis: Determining the degree in to which a test suite covers, i.e. verifies, a given set of coverage items, e.g., requirements, features or portions of code.
2. Requirements Coverage Analysis: Determining the degree in to which a test suite covers a set of requirements.
3. Feature Coverage Analysis: Determining the degree in to which a test suite covers a set of feature.
4. Code Coverage Analysis: Determining the degree in to which a test suite covers a code base.

For example, the VarSel: Service for Variant Selection tool is used in the following way. The functionality of VarSel comprises the generation of test cases from a variability model of the SUT. The user first chooses a coverage level. The tool then generates test cases and chooses variants by means of the methods developed in XIVT, in particular in D3.2.

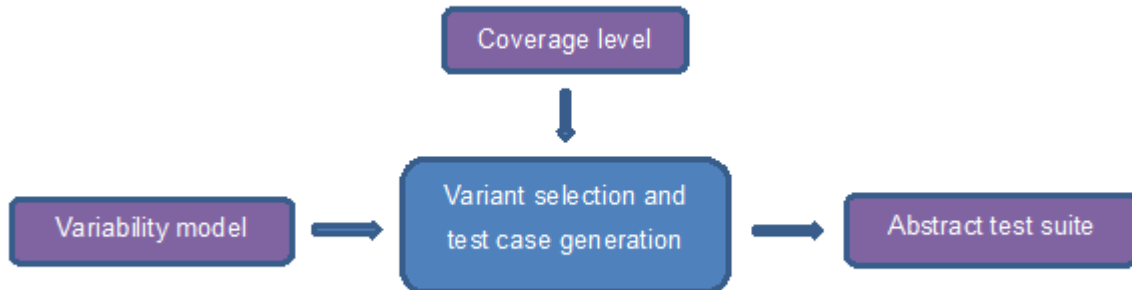


Figure 6: Architecture description showing the coverage level used as input to variant selection.

A thorough description of how XIVT TOM coverage criteria presented in this section are implemented in the XIVT tool chain can be found in D4.2.

3.2. Test object- and feature-based optimization⁴

One main goal of the XIVT TOM is to optimize the use of testing resources in variant-intensive products by providing decision support to determine which features to choose to test as test objects. For example, if a fault is detected in a component in the baseline architecture, we need to decide if it is necessary to (re-)execute the test cases for each product variant, for a subset of them or for the baseline only. Industrial requirements from the use-case owners can be taken as input, which can also serve to define the criteria for prioritization and selection. In the framework of model-based testing to software product lines, these requirements are transformed into feature models in order to specify test cases.

The tools developed for decision support to determine the scope of test to optimize the use of testing resources are applied to the XIVT use cases in order to evaluate the developed methodology. Finally, these tools are integrated as microservices into the platform architecture of this project.

With a strongly growing number of product variants, systematic reuse of artifacts across products becomes increasingly important. Feature orientation helps to organize and structure the whole product-line process as well as all software artifacts involved in terms of features. The main reason for using this approach is to have an easy way to trace the requirements of a customer to the software artifacts that provide the corresponding functionality. In this way, features become explicit in requirements, models, code, and test cases, i.e., across the entire life cycle [1].

⁴ Current results of T2.3 are the main input to this section.

A feature of a system can be defined as an optional or incremental unit of functionality. This is a rather intrinsic definition as features could also be defined for a single product being unaware of different variants of this product. More extrinsically, one can define features as portions that distinguish the product variants from each other or from the baseline. Regardless of the point of view, a feature is a logical unit of behavior that can be specified by a set of functional and non-functional requirements. The main objective of introducing features is to use them as a means to communicate between the different stakeholders of a product line (customers, end-users, managers, developers, test engineers), in order to distinguish product variants. Therefore, feature modeling is central to variability [1].

A specific product is identified by a subset of features, called a feature selection or resolution. A common approach of feature modeling is to express variability in terms of common and optional features. When instantiating a product variant from a baseline, four categories of features can be considered:

- 1) features common between the variant and its baseline
- 2) features added and specific to the variant yet not present in the baseline
- 3) features in the baseline that are not included in the variant
- 4) features present both in the variant and its baseline which are customized in the variant

A constraint on the feature selection is called a feature dependency. Features may depend on each other or exclude each other. That is, a feature change may affect many other features, which can become arbitrarily complex and is a major challenge in the development of software-intensive systems.

Feature models can be visualized in feature diagrams. The most popular form is the graphical representation as a feature tree; see, for example, the use case of Expleo in Figure 7. This is a graph-based representation with a hierarchical structure that visually depicts the features in groups of increasing levels of detail. An open-source framework to model, develop, and analyze feature-oriented software product lines is FeatureIDE which also has the functionality to visualize feature models as a feature tree.

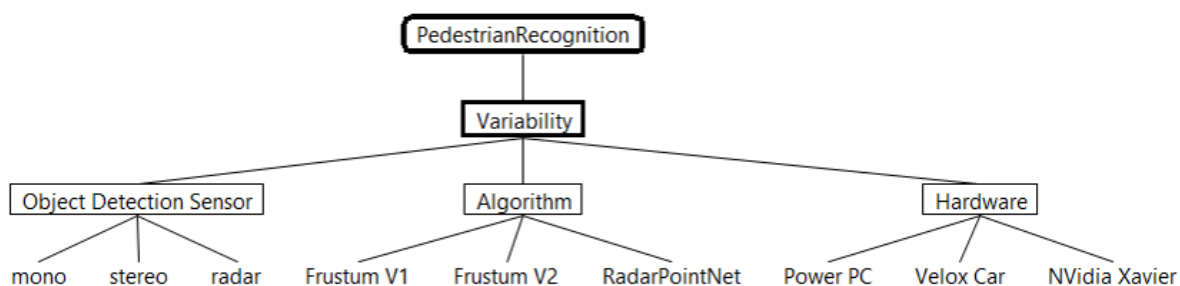


Figure 7: Example of a feature tree from Use Case of Expleo

In XIVT, there are many different use cases from different branches. Therefore, each industry partner and use case owner in XIVT might use the notion of features in another manner. By thorough surveys and interviews to all use case providers, we could find out what kind of variability and features are used in the industrial environment. The aggregated results are shown in the following Table 7. A graphical representation of these results is shown as a word cloud in Figure 8. As one can observe, on the one hand, features are considered as software artifacts such as types of algorithms, library extensions, or different UIs. On the other hand, features are also related to hardware components such as types of sensors, number of motors, fans, robots, or different monitor resolutions.

Use Case	Software Features	Hardware Features
Expleo: Pedestrian Detection	Classification Algorithms	Object Detection Sensors Hardware Platform
QAC/Automotive Manufacturer: Raw Map Data		Sensor Data
Bombardier Transportation an Alstom Group company n: Propulsion Control		Number of Motors Types of Fans Number of Contactors Types of Contactors Infrastructure (AC/DC)
ABB: Process Control System	Library Extensions Graphic Extensions System Extensions Number of Tags Number of Servers/Clients Depth of Models Number of Properties Plug-Ins	
FFT: Flexible Production Unit		Type of Robots Number of Robots Type of applications Welding Gun
Arcelik: TV User Interface	UI Web Browser Availability Picture Settings Audio Settings	Resolution Panel Types Tuner Multimedia Interface
mobileLIVE: Android Devices	OS Android Applications	Resolution Camera Chipset
Turkcell: Web Application	Vacation Management Customer Complaints Firewall Requests	
Percepio: Embedded Software	Build Flags Settings	

Table 7: Different types of features of XIVT Use Cases

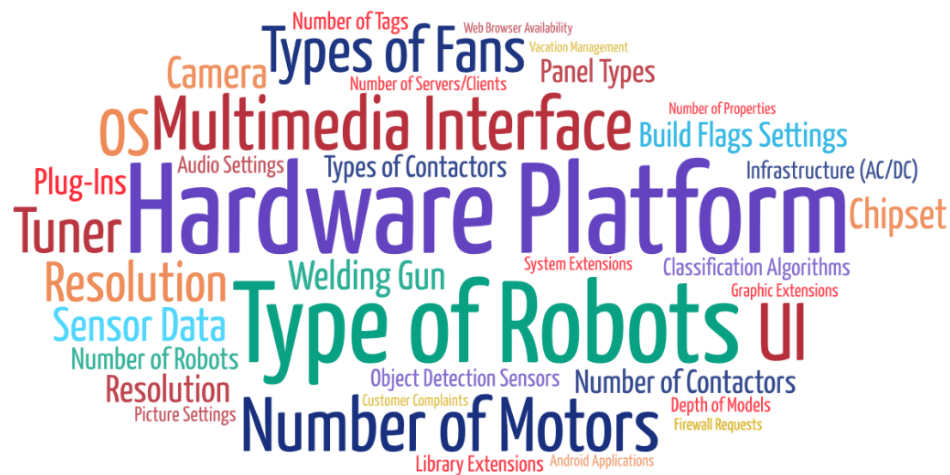


Figure 8 : The bunch of types of features of XIVT Use Cases⁵

The different methodologies developed in XIVT, including feature analysis, are described in the following chapter. Another strongly related topic to feature modeling is the change impact analysis (CIA). This is helpful for variant traceability and dependency with the aim of test optimization. Therefore, this document reports on state of the art in various CIA methods.

4. XIVT-Test Optimization and supporting Methodologies⁶

This section presents the different supporting techniques and approaches— proposed by XIVT partners — that support the overall XIVT-TOM methodology. Below, we present each method in detail.

⁵ created with wordart.com

⁶ The content of this section is first and foremost from the results of T2.3, and then secondly and to some degree also based on the results of the other WP2 and WP3 tasks.

4.1. Variability-Aware Reuse Analysis method by RISE

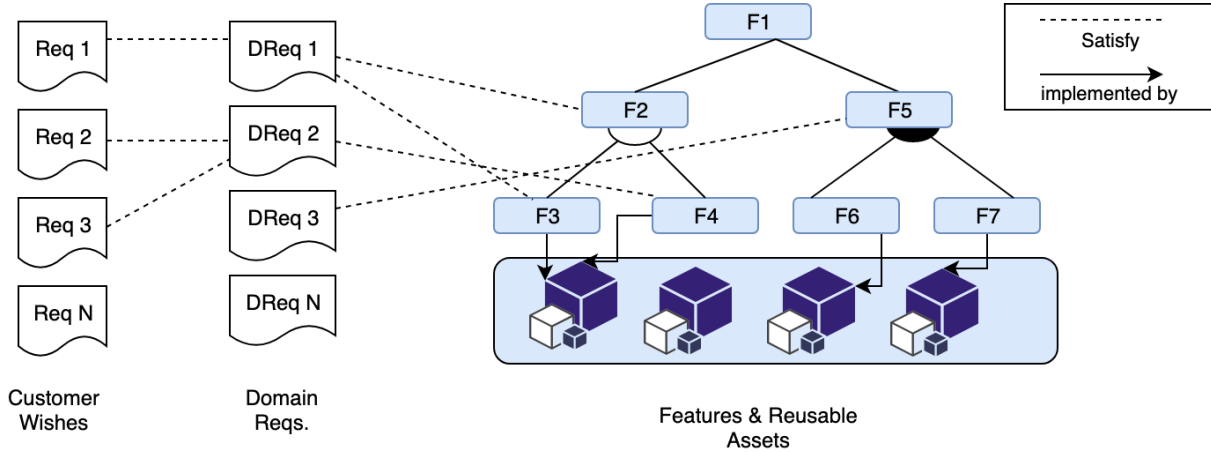


Figure 9: Requirements and Feature mapping

Context and the problem. As shown in Figure 9, typically, a product line implements a series of domain requirements in a particular market segment. The domain requirements satisfy a variety of customer requirements. Reusable assets realize the product line itself. When a new product is to be delivered to a new customer, companies often receive a customer requirements document. Engineers have to analyze the customer requirements to find reuse opportunities for existing product-line assets. The result of this analysis identifies the customer requirements that are already realized by the product-line assets. This, in turn, helps in avoiding redundant development efforts and in reducing the lead time. *However, identifying the reuse opportunities for existing product-line assets may become impractical as the number of existing requirements and assets grow (Scenario 1 of Bombardier Transportation, an Alstom Group company's Use-case in XIVT).*

The Assumption. In XIVT, we developed a tool called VARA [2] to support engineers in identifying reuse opportunities for existing product-line assets. VARA (Variability-Aware requirements reuse Analysis) aims to automate the requirements reuse analysis by recommending reuse of software based on customer requirements similarity and thus helps teams achieve quick and quality delivery of software systems. Like other recommender systems in software engineering, VARA is also based on the typical assumption that “customer requirements can be used as a proxy to identify software reuse opportunities”. However, this typical assumption was not extensively studied in the literature. Therefore, we investigated the extent to which requirements could be used as a proxy for software to identify reuse opportunities in the context of retrieval by means of correlation analysis between requirements similarity and software similarity [3, 4]. We found that a moderate positive correlation between requirements similarity and software similarity. Furthermore, we found that in our case, in around 60% of the cases, most requirement-similarity-based approaches would recommend software with good relevance.

How it works? VARA takes existing customer requirements and their links to product line assets as input and uses state-of-the-art natural language processing and machine learning algorithms to map these requirements into the vector space model (VSM). Existing customer requirements are mapped to representation vectors (numerical vectors) and are clustered based on similarity. When new customer requirements are received, VARA infers vectors for the new requirements and, based on similarity it identifies the most similar existing customer requirements. To support the engineers in reuse opportunities identification for existing assets, VARA retrieves the assets realizing those existing similar requirements as recommendations for reuse.

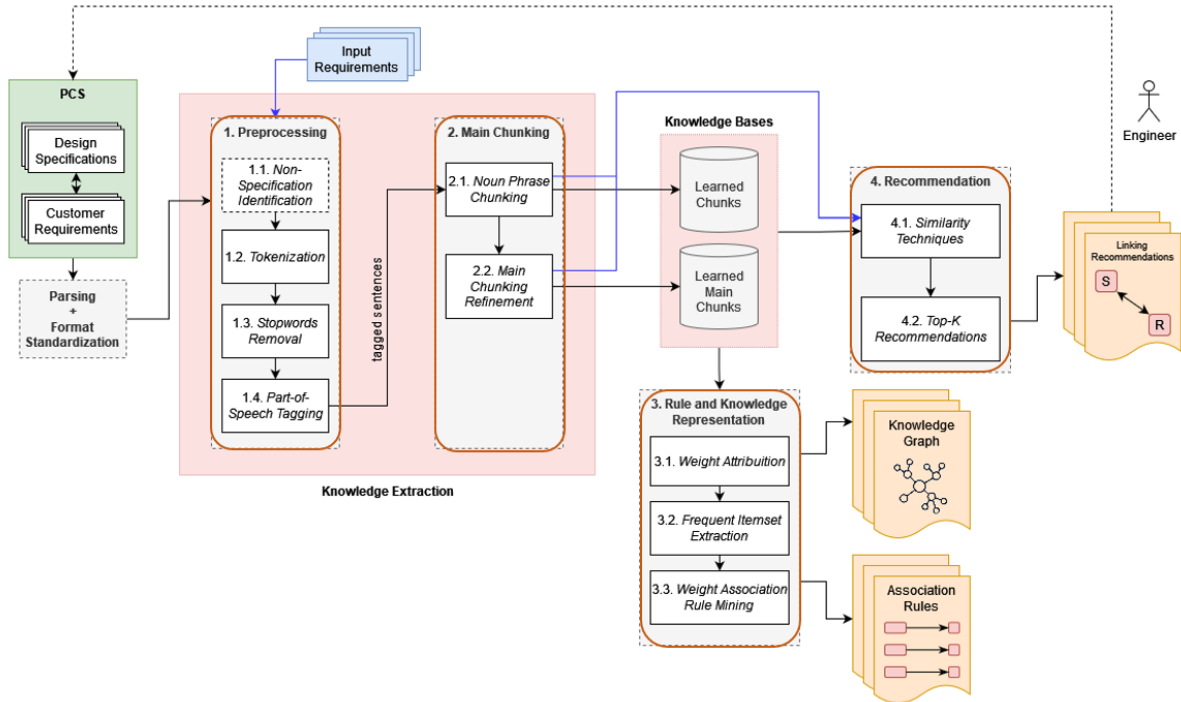
Evaluation. As per our evaluation in the railway domain, VARA was able to identify reuse opportunities for exiting product line assets with an average accuracy of 74% (later improved to 82%). Qualitative evaluation of VARA shows that engineers found the recommendations valuable and insightful.

4.2. NLP-based Design Specification Recommendation System for Requirements by FCUL

The ARRINA (*Association and Recommendation for Requirements in Natural Language*) [5, 6] system processes both software design specifications and customer requirements for identification and extraction of relevant components, features and information. The ARRINA model establishes a pipeline of Natural Language Processing and Information Retrieval methods that allows deriving products from a Software Product Line, offering an analysis for the integration of new requirements and which specifications to be associated with those.

ARRINA was developed in collaboration with Bombardier Transportation, an Alstom group company, and is tested within railway Propulsion and Control Subsystems (PCS). A PCS is a safety-critical system consisting of *software design specifications* (PCS standard product feature) and *customer requirements* (revision of a product feature) provided by various stakeholders. Specifications and requirements are both written by engineers in natural language and belong to two different but related domains. Since new configurations can be derived from these features and require considerable time for testing them properly, it is essential to prioritize requirements to test them later and swiftly. Therefore, our objective is to create a recommendation framework that links new requirements of new configurations to the subsystems, i.e., to specifications.

Below, is the diagram that defines the structure of ARRINA, mainly divided into four phases: 1. Preprocessing; 2. Main Chunking; 3. Rule and Knowledge Representation; 4. Recommendation.



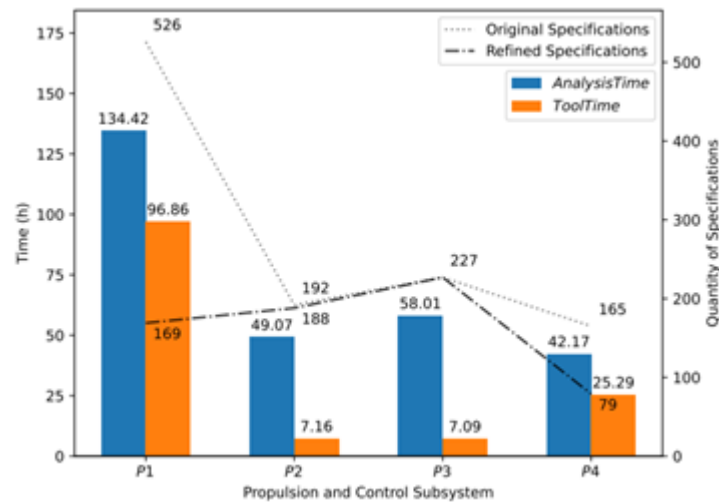
First, for each natural language textual description contained in a specification or requirement, we employ a **Preprocessing** task divided into four steps: *non-specification identification* (where the model identifies descriptions not corresponding to a specification, through a set of filters), *tokenization* (every word is defined as a token), *stopwords removal* (where stopwords are removed), and *part-of-speech tagging* (where every token is associated with its semantical tag). The **Main Chunking** phase receives a set of software design specifications, customer requirements and input requirements, with their sentences already tagged, performs noun phrase chunking techniques over each sentence, and then refines the results to obtain the most relevant chunks, the *main chunks*.

The **Rule and Knowledge Representation** phase receives as input the *main chunks* and their *derived chunks* of a subsystem (i.e., the software design specifications and customer requirements that have been processed) and outputs the association rules discovered between the main chunks and the knowledge graph representative of them. The phase employs a *Weight Association Rule Mining* (WARM) process, which evolves the standard Association Rule Mining methodology with the weight of each chunk that occurs in the dataset. The WARM process is boosted by the *Weight Attribution* and *Frequent Itemset Extraction* tasks that, respectively, calculate the weight of participation for each chunk and extract itemsets where they occur within the dataset. Afterward, the rules are extracted by WARM.

The **Recommendation** task is focused on providing top-k software design specifications as recommendations for a set of input requirements through similarity techniques between the chunks (both main and derived) contained in the design *specifications* and *input requirements*.

ARRINA is proven to extract main chunks considered relevant to engineers through simple grammar rules that can be adapted to other domains. The association rules and graph representations of components also can help the engineers better understand the relations occurring inside a subsystem for better component matching.

The model also has a great performance on recommending new and never associated input requirements. ARRINA has been tested with four subsystems, estimating a reduction of testing time, in some cases, of 85%. Recommendations also have accuracy close to 80% in some subsystems involved in testing.



4.3. NLP-based Requirements Formalization for Automatic Test Case Generation approach by ifak

Problem and Motivation. Due to the growing complexity and diversity of variants of software systems, assuring their quality is becoming increasingly laborious. Especially for safety-critical systems, extensive testing based on requirements is necessary. Methods of model-based test automation in agile software development offer the possibility to overcome these difficulties. However, a major effort is still required to create models from a large set of functional requirements provided in natural language. Many authors restrict their NLP approaches to a specific domain or a prescribed format. A major drawback of using controlled natural language or templates is that it forces the requirements engineer or designer not only to concentrate on the content but also on the syntax of the requirement. Furthermore, those algorithms are in general not applicable to existing requirements.

Recent advances in natural language processing show promising results to organize and identify desired information from raw text. As a result, NLP techniques show a growing interest in automating various software development activities like test case generation. Several NLP approaches and tools have been investigated in recent years, aiming to generate test cases from preliminary requirements documents [7, 8, 9].

Aim. In the tool ReForm [10], developed within XIVT, a new semi-automated technique for requirements-based model generation is considered, that reduces human effort and supports frequent requirements changes and extensions. The aim of this approach is to develop a method that can handle an extended range of domains and formats of requirements and provides enhanced but easily interpretable intermediate results. Requirement engineers do not need to write requirements with an exhaustive list of pre-conditions, post-conditions, etc., as expected by most of the approaches mentioned above, and they may not even conform to a particular template. This tool can potentially consider behavioral requirements not particularly aligning with a specified template or domain, and it does not rely on the outcome of coding practices for the generation of test cases but rather the formal specifications.

Methodology. We utilize an NLP parser and develop a rule-based algorithm to perform the transformation from requirements written in natural language into requirement models. Our rule set tries to conceive all relevant rules that could satisfactorily parse the input behavioral requirement and extract its semantic content. After decomposing the requirement into separate clauses, we identify syntactic and semantic entities from which we finally generate the requirement models.

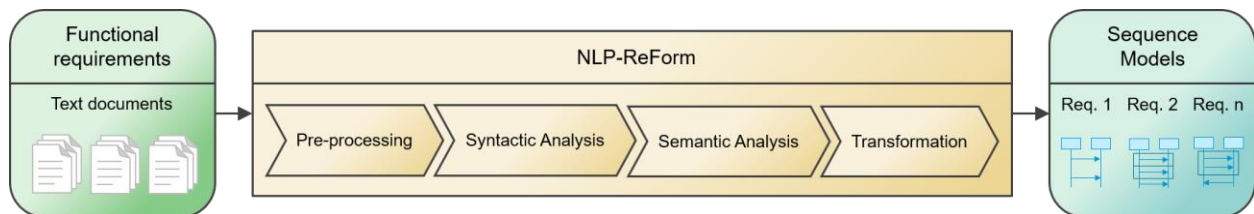


Figure 10: Overview of the NLP steps for requirement formalization

Evaluation. The toolchain for requirements-based model and test case generation was applied to an industrial use case from the e-mobility domain. The use case describes a system for charging approval of an electric vehicle in interaction with a charging station. The industrial use case was defined by AKKA and aims to provide a typical basic scenario and development workflow in software development for an automotive electronic control unit (ECU). We showed that the algorithm could produce complete, correct and consistent artifacts to a high degree. The evaluation results show the correctness of the algorithm of 75% on average with a 78% completeness and 94% consistency. We have also shown how these artifacts are then used to

create sequence diagrams for each requirement and transformed into a state machine for the entire specification model to generate abstract test cases finally.

4.4. Similarity-based Requirement Formalization using Product Design Specifications by ifak

Problem and Motivation. The rapid and simultaneously high-quality development of industrial software products demands an increasingly effective test process. Especially for safety-critical systems, such as in the automotive and railway domains, extensive testing based on requirements is necessary. However, any manual processing, such as requirement verification and test generation, from textual requirements is time-consuming and error-prone and also requires a lot of expert knowledge (*Scenario 4 of Bombardier Transportation, an Alstom group company's Use-case in XIVT*).

Most of the existing NL-based modeling approaches lack information about the architecture and design of the system. This is not only about mapping abstract to real entities such as components, signals and parameters. There are usually many specific details in the system design from which the requirement formalization process would benefit. The system architecture describes the structure and includes architectural design decisions and is therefore closely related to the requirements.

Aim. In XIVT, we developed the tool ReForm (version 2) [11] that shall support requirements engineers by providing automatic recommendations of requirements models from textual requirements and design specifications. We want to take into consideration product design specifications for a much more precise requirements formalization process. It shall generate human- and machine-readable models. The goal is to provide automated support for the manual requirements engineering process and thus reduce costs and effort.

Methodology. Our new approach utilizes Natural Language Processing (NLP) techniques to automatically generate requirement models from natural language requirements and design specifications. We perform textual similarity and contradiction analysis between the requirements and entity descriptions using classical to modern NLP algorithms. In this way, information from the system architecture such as signal and parameter descriptions is compared with the requirements texts. The generated models are represented in a simple machine- and human-readable language.

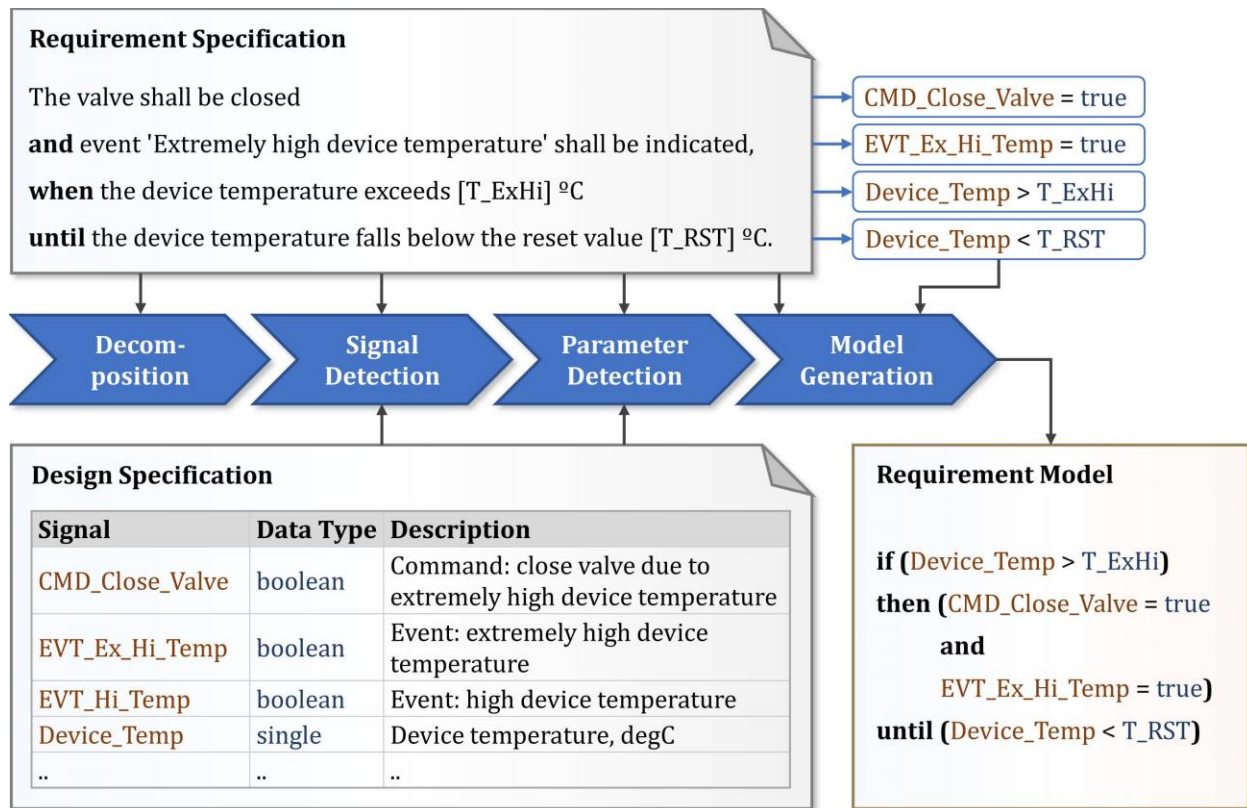


Figure 11: Exemplary requirement with an overview of the NLP pipeline

Evaluation. We consider a use case from the rail industry. We use the data of the Propulsion Control (PPC) system in Bombardier Transportation, an Alstom Group Company. The PPC is part of a large, complex, safety-critical system. It handles the control of the entire propulsion system, including both control software as well as electrical functions. The proposed requirement models for the considered propulsion system show an average accuracy of more than 90% and an exact match in about 55% of the cases. These results show that our approach can highly automate the process of requirements formalization, which can support the requirements engineer e.g., in requirements verification, consistency checking, and test case generation.

4.5. Combinatorial Interaction Testing methods by MDU

Problem and Motivation. During the software development process, testing is one of the most important parts to ensure the high quality of the final developed product. Software testing can be a time-consuming and expensive activity, especially for an SPL, so it is of interest to make it as efficient as possible. One method for effective fault detection in the program under test is a combinatorial testing.

Aim. In XIVT, we developed the tool called SEAFOX for feature interaction testing using a combinatorial approach. It has been shown that interactions of only a few parameters cause many failures. Combinatorial testing focuses on finding the combinations that produce those failures using different combinatorial techniques, t-way (e.g., pairwise). Our aim with this tool is to take a PLCopen XML file as an input that contains information about the program to be tested. Then the tool parses the file by extracting the needed information that will be used to generate test cases, such as the name of the input parameters and their respective data types. Another option is to manually provide the parameters and data types in the tool.

Further, SEAFOX tool creates test cases by generating the inputs using one of the available algorithms implemented in the tool: Random, Base choice, or Pairwise algorithm. These input values can then be exported as comma-separated values in a CSV file where each line is a new set of input parameters representing a certain test case. SEAFOX has a graphical user interface (GUI), as shown in Figure 12, where the user starts by loading a folder with the desired program or function block stored as an XML file. Alternatively, the parameter names and data types can be added manually. A limitation in the tool when importing an xml-file is that the file only can contain one function block with input parameters. If it contains two or more function blocks with input parameters, the tool will assume that all parameters can be combined in a test case.

After choosing the file to import, input variables from the file are transferred to the tool and visualized in the middle container of Figure 12. Below this container, the user chooses which combinatorial algorithm to use – Random choice, Base choice, or Pairwise. Then, a range for each parameter, as well as other additional information depending on the chosen algorithm, needs to be set before the test cases can be generated. The ranges specify the values each parameter can take and is either a single value, a closed interval, or a combination of both. Examples of ranges and the notations used in the tool:

- 7=a single value of only 7
- 1_3 = a closed interval (1 and 3 are included)
- 1_3;7 = a combination of both

Once all this is set, test cases can be generated and displayed in the right container. Lastly, the user can export the test cases as a CSV-file by clicking the Save test button. An example of test cases generated using Base choice is depicted in Figure 12.

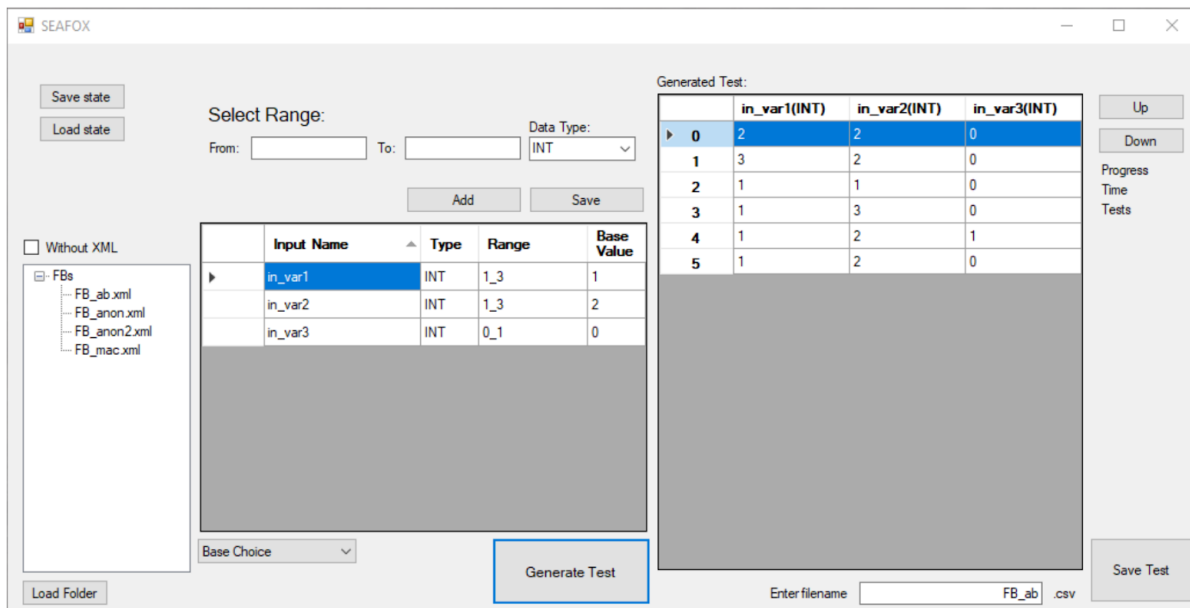


Figure 12. Base choice test suite generated in SEAFOX. The test cases have three parameters of integers of which each has a range and base choice-value set as shown in the middle container in the image. The generated test cases are located in the box to the right.

When the SEAFOX tool has generated test case input values, they are saved in a csv-file. To be able to use these values for automated test case creation, SEAFOX is integrated with CODESYS by taking the content of the csv-file into the respective test cases that would be created based on a template from the previously exported PLCopen XML file from CODESYS. Test cases are executed using CUnit tool.

Methodology. The usage scenario of this tool is explained in the form of steps as shown in Figure 13. The first step after selecting programs and specifying input ranges is to load these programs (written in FBD and ST programming languages) into CODESYS IDE and then create the necessary test suites using CUnit for the FB that were to be tested (step 3). Each test suite is able to hold several test cases, where the input parameters for those test cases were generated through the SEAFOX tool (step 4) and the expected outputs were provided manually. In the next step, the exported csv-file from SEAFOX and the file with the test suite from CODESYS are processed within the script to create a new test suite containing test cases with all test case inputs from the csv-file as a PLCopen XML file (step 5). This file was then imported back to the CODESYS programming environment (step 6) where tests were executed with the selected testing tool CUnit (step 7).

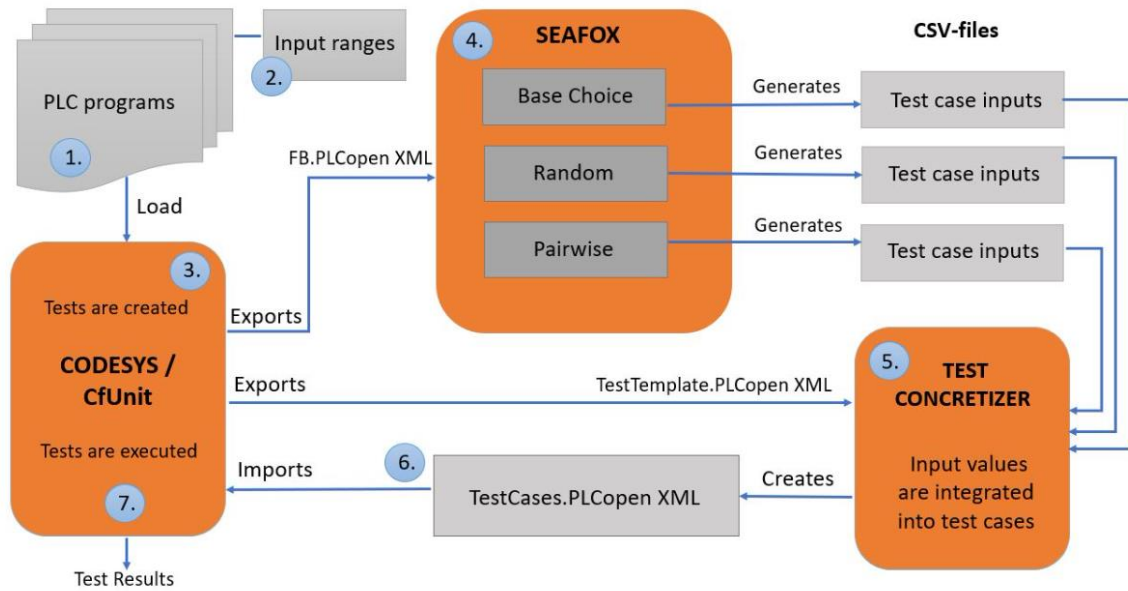
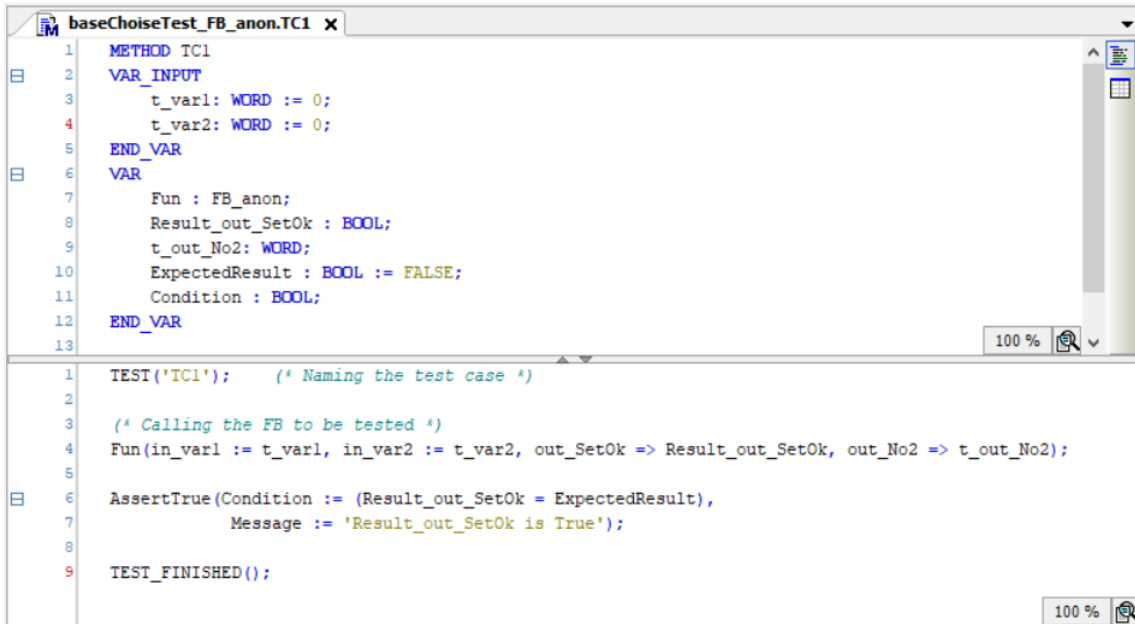


Figure 13: An overview of the integrated methodology for combinatorial test modelling and test execution in CODESYS for IEC 61131-3 and its usage scenarios.

Test cases in CODESYS IDE were written following the instructions by CfUnit. CfUnit provides an assertion method that checks if the condition in the assertion is True. If it is False, an assertion error is created. One test case per test suite was created as a template that was later used with the script and the exported file from SEAFOX to generate all the test cases for the test suite. All input variables in the test case are initialized with a dummy value for enabling the script to set the test case inputs (see Figure 14).



```

1  METHOD TC1
2  VAR_INPUT
3      t_var1: WORD := 0;
4      t_var2: WORD := 0;
5  END_VAR
6  VAR
7      Fun : FB_anon;
8      Result_out_SetOk : BOOL;
9      t_out_No2: WORD;
10     ExpectedResult : BOOL := FALSE;
11     Condition : BOOL;
12 END_VAR
13
14 TEST('TC1');    (* Naming the test case *)
15
16 (* Calling the FB to be tested *)
17 Fun(in_var1 := t_var1, in_var2 := t_var2, out_SetOk => Result_out_SetOk, out_No2 => t_out_No2);
18
19 AssertTrue(Condition := (Result_out_SetOk = ExpectedResult),
20            Message := 'Result_out_SetOk is True');
21
22 TEST_FINISHED();
  
```

Figure 14: A test case in CODESYS IDE. The input variables are initialized with a dummy value that will later be replaced with a proper test case input through the execution of the script. An assertion that compares the actual and expected result is used to measure the decision coverage.

SEAFOX CODESYS Edition is an open source tool that supports PLCOpen XML and *.EXP files to be imported from CODESYS. The supported algorithms are Random, base-choice and pairwise.

Evaluation. Our results on SEAFOX are related to terms of fault detection, code coverage, and the number of tests. The results of our studies show that pairwise testing, while useful for achieving high code coverage and fault detection for the majority of the programs, is almost as effective in terms of fault detection as manual testing. The results also suggest that pairwise testing is just as good as manual testing at fault detection for 64% of the programs. The results of our follow-up experiment show that test cases generated by this methodology achieved on average 90% decision coverage or higher regardless of the combinatorial technique used, with the overall average level being 94%. Interestingly enough, Random testing scored higher than both Pairwise and Base choice testing.

4.6. Clone detection in model-based design by Addiva, MDH and RISE

Problem and Motivation. Software reuse by copying and modifying components to fit new systems is common in industrial settings. However, it can lead to multiple variants that complicate testing and maintenance. Therefore, it is beneficial to detect the variants in existing codebases to

document or incorporate them into a systematic reuse process. For this purpose, model-based clone detection and variability management can be used. Unfortunately, current tools have too high computational complexity to process multiple Simulink models while finding commonalities and differences between them.

Aim. We explore a novel approach called MatAdd that aims to enable large-scale industrial codebases to be processed. The primary objective is to process large-scale industrial Simulink codebases to detect the commonalities and differences between the models.

Methodology. The work was conducted in collaboration with Addiva and Alstom to detect variants in Alstom's codebase of Simulink models. Alstom has specific modeling guidelines and conventions that the developers follow. Therefore, we used an exploratory case study to change the research direction depending on Alstom's considerations. More details on the methodology and its evaluation can be found in the report of Parkkila [12].

Evaluation. The results show that MatAdd can process large-scale industrial Simulink codebases and detect the commonalities and differences between its models. MatAdd processed Alstom's codebase that contained 157 Simulink models with 7820 blocks and 9627 lines in approximately 90 seconds and returned some type-1, type-2, and type3 clones. However, current limitations cause some signals to be missed, and a more thorough evaluation is needed to assess its future potential. MatAdd's current state assists developers in finding clones to manually encapsulate into reusable library components or find variants to document to facilitate maintenance.

4.7. Budget-based Test Selection and Prioritization approach by FOKUS

Budget based prioritization and distribution of responsibilities. The real issue when it comes to testing is probably not a technical problem but an economic problem: Due to limited resources, only small fractions of complex systems can indeed be tested. For variant-rich systems, the total number of possible variants is eventually already so large that testing all of them at least rudimentarily becomes way too expensive.

If the budget is the hardest limit for testing, then it makes perfect sense to try to optimize the utilization of the available resources. This includes prioritizing test cases based on their criticality assessment and actually selecting a subset of possible test cases according to some coverage criteria.

However, just looking at the potential benefit of the test cases is not enough. For solving the optimal testing budget utilization challenge, the costs of test cases have to be considered, too. Once some test case is executed for the first time, it is possible to measure the execution costs. Costs for fully automated pure software test execution will correlate to the execution time, the used calculation power, and memory consumption, which are easy to measure. If the parts of the

system under test or of the test environment are subject to significant wear, for instance, it might be more difficult to get precise values for the costs of individual test case executions.

It is definitely desirable to have at least some estimate for the order of magnitude of costs as early as possible – prior to test case execution, if possible even prior to test case implementation.

Mathematically, the optimal testing budget utilization problem can be modelled as a variant of the knapsack problem, which is NP-complete [13]. More precisely, it can be modelled as a multi-objective knapsack problem by interpreting the available budget as the weight limit of the knapsack and the costs of each test case as the weight of a piece while treating prioritizations like criticality values and at least one coverage criteria as multiple objectives. Coverage criteria evaluation is fundamentally different from other objectives because its value depends on the combination of all chosen test cases. Evaluating such an objective is itself an NP-hard coverage problem. That makes it especially difficult to develop good algorithms to approximate optimal testing budget utilization problem solutions with polynomial effort: For instance, iteratively identifying just the next best individual item to add to the knapsack until reaching the capacity limit will probably only be possible with estimating what could be a good candidate for the coverage objectives. Hence there will be multiple approximations, eventually leading to substantial deviations from perfect solutions. However, if there is only a single coverage criterion, then it is also possible to model the optimal testing budget utilization problem as a weighted cover problem and to find a good solution with a greedy algorithm [14], which avoids having a cover problem nested in a knapsack problem.

Budget based scheduling. The costs of executing a test case are not fixed: Some test cases might require preparation with some effort (e.g., create a certain configuration and reach a specific state) and once the test is finished, additional postprocessing might be required to tidy up. One could try to execute multiple different test cases having the same pre-conditions without spending the identical preparation effort again and again. Similarly, spending postprocessing efforts could be reduced if there are test cases that may be executed without it. There might even be test cases that can be executed in parallel, which could allow reducing costs in comparison to run the tests sequentially by better using resources like calculation power, memory and time, for example.

Clearly the order in which test cases are executed and the level of concurrency can have significant impact on the total testing costs. Therefore, it is a good idea to solve the test schedule optimization problem for the selected test cases prior to executing any of them. This requires a clear specification of preparation and postprocessing from the actual test execution for each test case – which is a good idea, anyway. Additionally, information for concurrency is desirable: which resources are needed exclusively for executing this test case, which resources may be shared efficiently? Parallel test execution of multiple test cases guaranteeing flawless execution by blocking used resources might result in even worse execution time compared to sequential execution if the multiple parallel executed test cases have to wait for one another most of the time and if locking is expensive.

For really selecting the test cases to use the budget in the best possible way, it would be necessary to take the optimal scheduled costs for the test cases into account in the selection process already. But the scheduled costs depend on the selection of test cases. So, there would be another nested

optimization problem to solve. In praxis, it is probably best to do the budget-based test case selection iteratively with calculating the costs for each new test case candidate, which should eventually be added next using optimal scheduling considering all the test cases selected so far.

Sharing test responsibilities and test efforts for variant rich systems. Even highly optimized test budget utilization will probably not make continuous testing of variant-rich systems with regularly updated components affordable for the producers. It is most likely not possible for them to test all variants for each change that might affect them. In the real world, they will not even know which variants are actively used. Eventually, the users can configure their own variants (i.e., configurations) by themselves. Testing only some standard variants might be insufficient and putting users of other variants at risk.

One idea to overcome this problem is to provide users with the test suite so that they can test their very own, potentially unique variant by themselves. Test execution might use the actual systems while they are not in productive use (e.g., in a maintenance pause), or tests could also be executed against a dedicated test version of the variant (e.g., testing a simulation), potentially in a cloud environment.

In order to prevent double-spending efforts, besides providing a variant flexible test suite to customers it is also required to have a common exchange platform (Figure 15) coordinating test efforts and for sharing test results already generated. If the system under test contains completely separated components, then potentially, some variants will differ only in a few of these components while others are identical. For instance, there could be a complete physical separation between safety-critical subsystems and comfort subsystems. Many variants will then probably share the same safety-critical core and have only diverse comfort subsystems. In such cases with complete isolation, tests for common parts do not have to be executed for all variants. Once generated, test results may be shared and used for all variants that are identical in that aspect.

The coordination of test efforts, i.e., the distribution of testing tasks, should try to distribute test case execution costs equally among users having identical variants. The test cases distribution should be done with optimal scheduling for each user. Distributed test case execution typically allows a highly concurrent test execution, so the solution of this cost-oriented coverage optimization problem will depend on the number of candidates for testing the same variant.

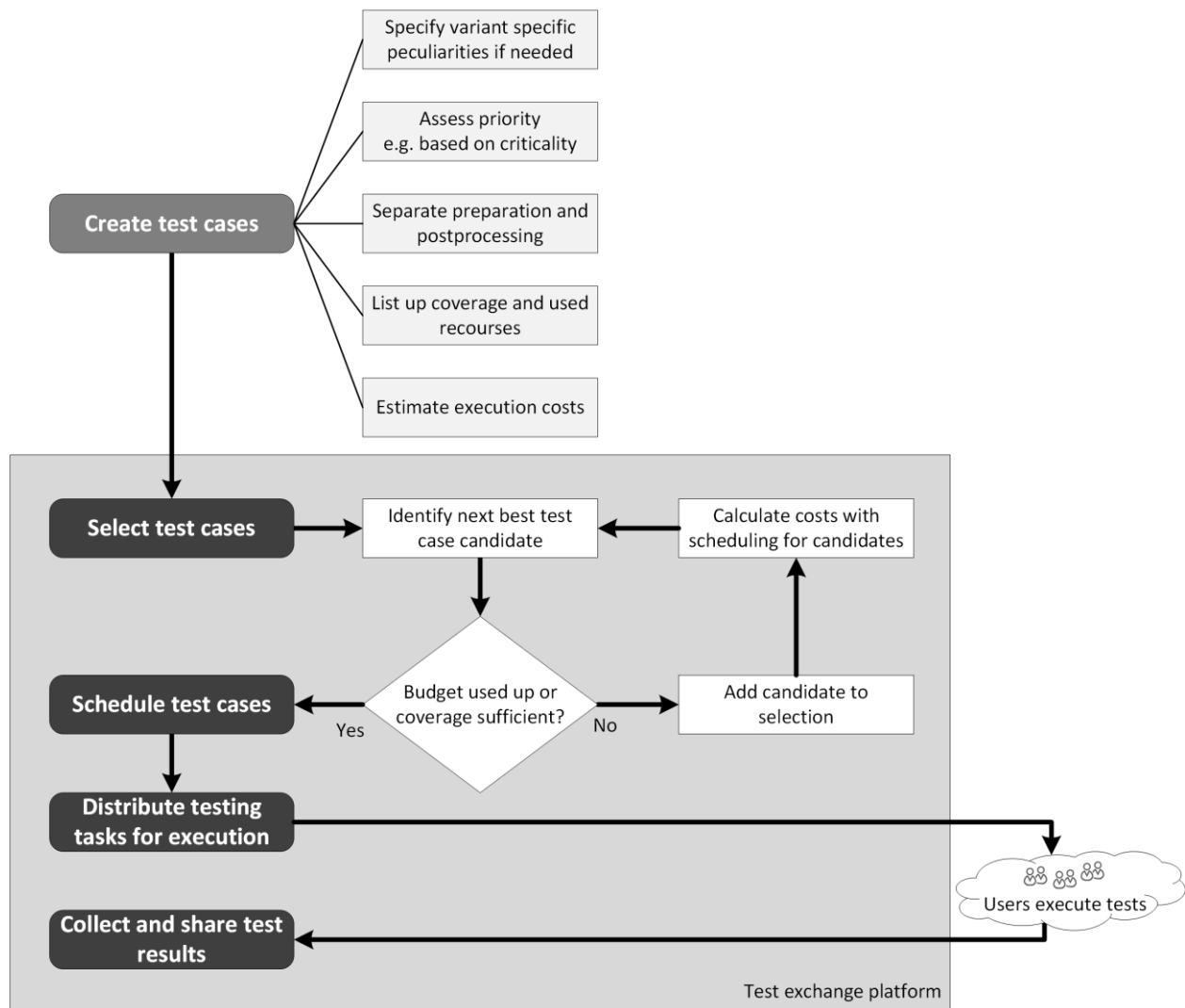


Figure 15 : Test exchange platform for budget optimized testing

4.8. Risk-based test scoring method by QAC

Introduction

Safety and security risks are two critical aspects for optimal test case prioritization. In this section, the technique being used for developing a tool for test optimization based on risk criteria is explained and a case study is also provided, as depicted in Figure 16. The inputs are the “abstract test suite” and a domain “specific Knowledge base” to a knowledge-based system, and the output are a set of scored test suites based on a risk which could be a safety and security-related risk.

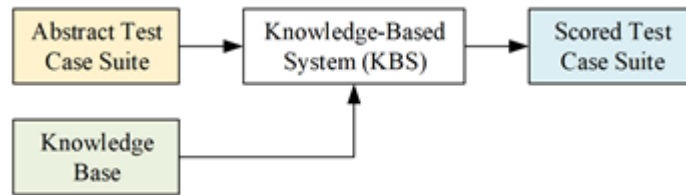


Figure 16. Architecture diagram of the proposed technique.

Methodology

The detailed architecture is shown in Figure 17. In this architecture, the selected abstract test cases within the test suites are characterized by three main parts:

- A set of components in a pre-execution condition (Pre-condition Set)
- A set of components acting as the input values (Input Set)
- A set of components that have the expected condition (Impacted Set)

The Impacted Set is scored based on two factors. The number of components relative to the pre-condition set. In other words, the number of components in the impacted set indicates the number of possible faults that a particular test case can identify. The second factor is the priority level of the components in this set. In fact, high-risk components should have a higher priority which can affect the overall score of the test cases.

We identify the operational situation from the domain-specific requirements, which consists of two parts, System states, and Environment conditions. Each part has several attributes, and each attribute may include a number of states. For example, for the automotive domain, we have “Operational mode” as part one of the operational situation, and an example of the attribute could be “Vehicle speed,” where the states could be Very slow, slow, etc.

The Operational situation (OpSit) is described by the test scenario and the corresponding testing environment where the system is supposed to act safely. It consists of the system’s state (e.g. operational mode) and environment conditions (e.g. weather conditions). So, for each state under OpSit, we assign a risk score.

The risk associated with the potential hazard with each attribute’s associated states is assessed in terms of severity, probability of exposure, and controllability. This is based on historical data and expert knowledge. Finally, the test cases are scored, and for each test case within the test suites, we will have a set of OpSit where we select the one with the highest score.

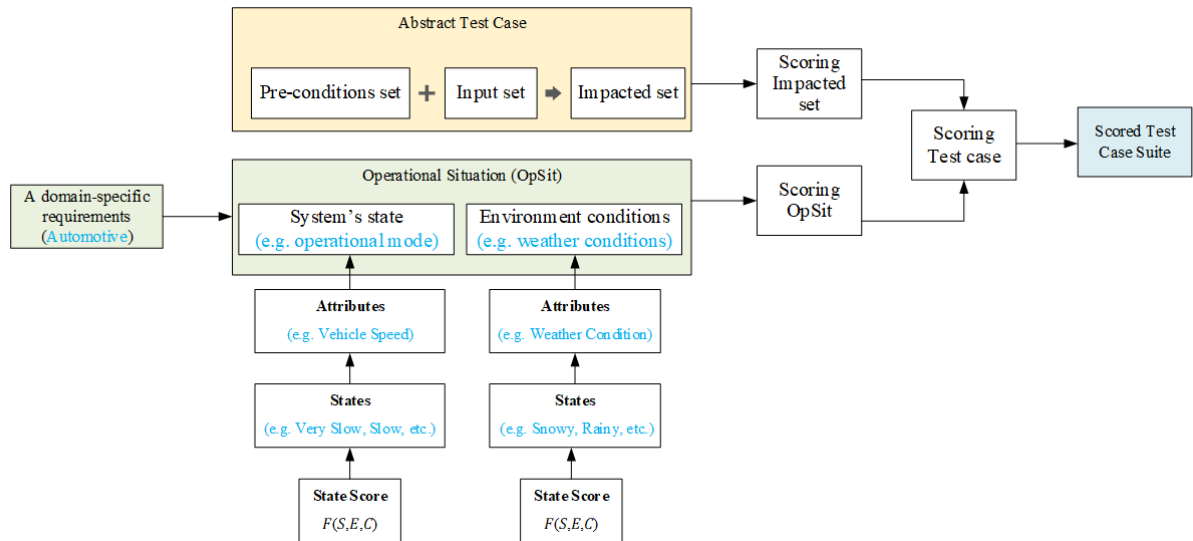


Figure 17. Detailed architecture diagram of the proposed technique.

Case Study

A case study in the automotive domain is selected here where we perform system integration testing, which is a subset of functional testing. In this example, we have selected a set of 200 test cases covering some functionality aspects of the system and want to score the test cases.

The system states and environment conditions are considered to have 12 attributes, and each attribute has a number of associated states. By considering all states for this case study, we will have 5,242 different OpSit for each test case.

The table here shows the selected 12 attributes (each is listed as one column here), and each row represents a combination of the various attributes that are considered as a single OpSit. A small portion is presented in this table.

OpSit No	Vehicle_speed	vehicle_direction	Road_linearity	Road_slope	Road_layout	Road_obstacle	Road_traffic	Road_pedestrians	Env_surface	Env_visibility	Env_temperature	Env_momentum	Oppt_Score
0	0.89	0.11	0.87	0.11	0.79	0.11	0.01	0.01	0.04	0.60	0.23	0.89	0.39
1	0.53	0.87	0.11	0.79	0.11	0.01	0.01	0.04	0.60	0.23	0.89	0.85	0.42
2	0.87	0.11	0.79	0.11	0.01	0.01	0.04	0.60	0.23	0.29	0.11	0.87	0.34
3	0.11	0.79	0.11	0.01	0.01	0.04	0.60	0.23	0.29	0.53	0.87	0.11	0.31
4	0.79	0.11	0.01	0.01	0.04	0.60	0.23	0.29	0.85	0.87	0.11	0.79	0.39
5	0.11	0.01	0.01	0.04	0.60	0.23	0.49	0.11	0.87	0.11	0.79	0.11	0.29
6	0.01	0.01	0.04	0.60	0.23	0.49	0.53	0.87	0.11	0.79	0.11	0.01	0.32
7	0.01	0.04	0.60	0.23	0.49	0.85	0.87	0.11	0.79	0.11	0.01	0.01	0.34
8	0.04	0.60	0.23	0.70	0.11	0.87	0.11	0.79	0.11	0.01	0.01	0.04	0.30
9	0.60	0.23	0.70	0.53	0.87	0.11	0.79	0.11	0.01	0.01	0.04	0.60	0.38
10	0.23	0.70	0.85	0.87	0.11	0.79	0.11	0.01	0.01	0.04	0.60	0.23	0.38
***	***	***	***	***	***	***	***	***	***	***	***	***	***

Table 8: List of selected attribute for this case study.

By considering all combinations of the test cases and OpSit, we will end up having over 1 million combinations that we will score. The histogram presentation here shows the resolution of the developed measure for calculating the risk score.

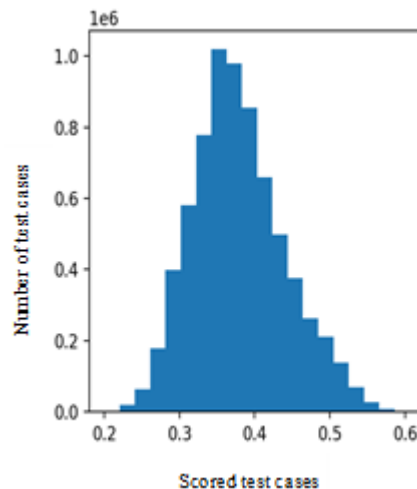


Figure 18: The histogram presentation of the generated test cases.

Eventually, we will create a subset of the scored test case. So, among those 5,242 different OpSit, for each test case, we only select the ones with the highest risk score for each test case. That's how we score the original 200 test cases.

4.9. Test Selection and Prioritization at Turkcell

Introduction

There are more than 90 development teams in Turkcell, each giving one or more modules of software products that are either in commercial use or in internal use. While so many teams and modules are present, there is only one test team in the company that performs all test tasks issued by the development teams. This has been creating an obvious need for some selection/prioritization mechanism. However, before the XIVT project, there was no facility to automatically prioritize these test tasks. This does not imply the tests are run in random order, but they were manually prioritized according to urgent needs, availability of scenarios, and some similar criteria, which may not be quite rational. Our use case in the XIVT project was the first application of automatic test prioritization trials in Turkcell.

Methodology

The prioritization methodology was developed by ARD and given for the use of Turkcell as a service. Since the methodology requires sufficient training before use, a number of scenarios that were already created and manually prioritized have been supplied to ARD. Those data were used for the training of the prioritization model. In our use case, initial inputs are voice-based. Hence, we apply an STT (Speech-to-Text) operation in order to obtain BDD test scenarios in text. In that respect, some converted scenarios were also supplied to ARD. After training the model in this way, a test prioritization service was ready to prioritize new tasks.

We are given web service and API-key to be able to submit our new prioritization tasks to the tool. Then, we send test data in pre-agreed format as seen in Figure 19 below. The most important fields in the request data are the test no and the test description fields, which have a significant effect on the determination of the prioritization result.

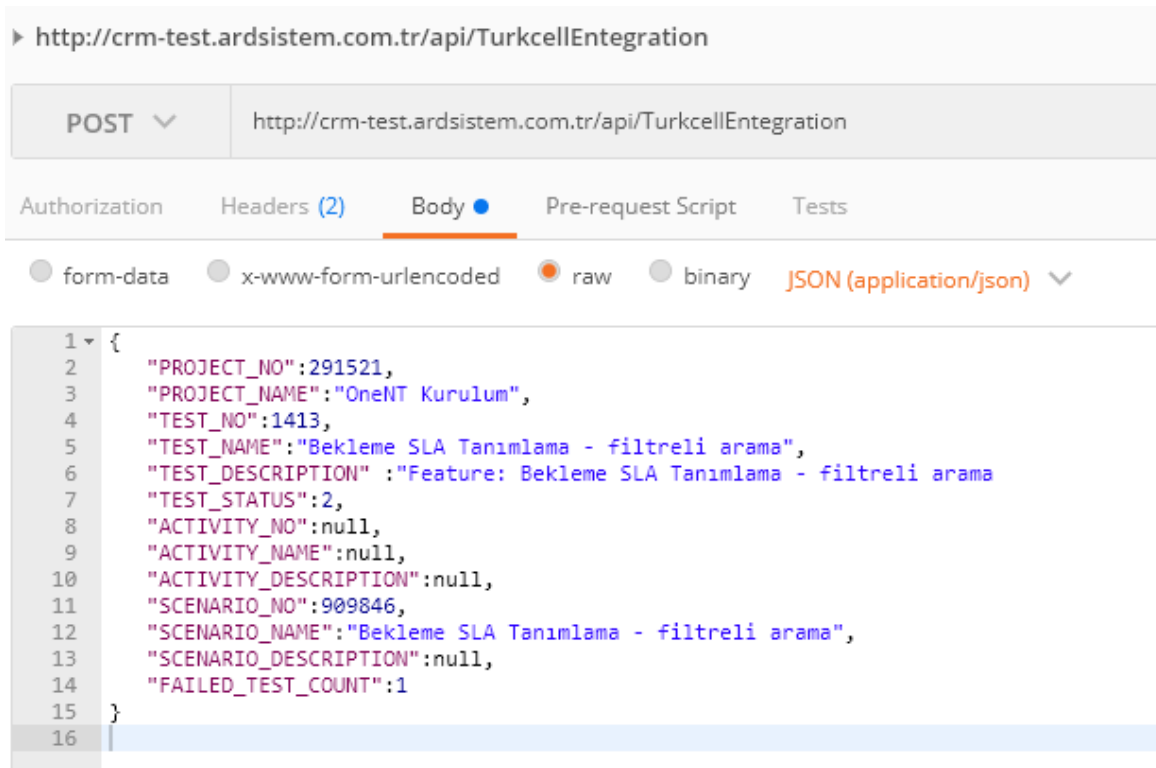


Figure 19: A screenshot of the prioritization service request.

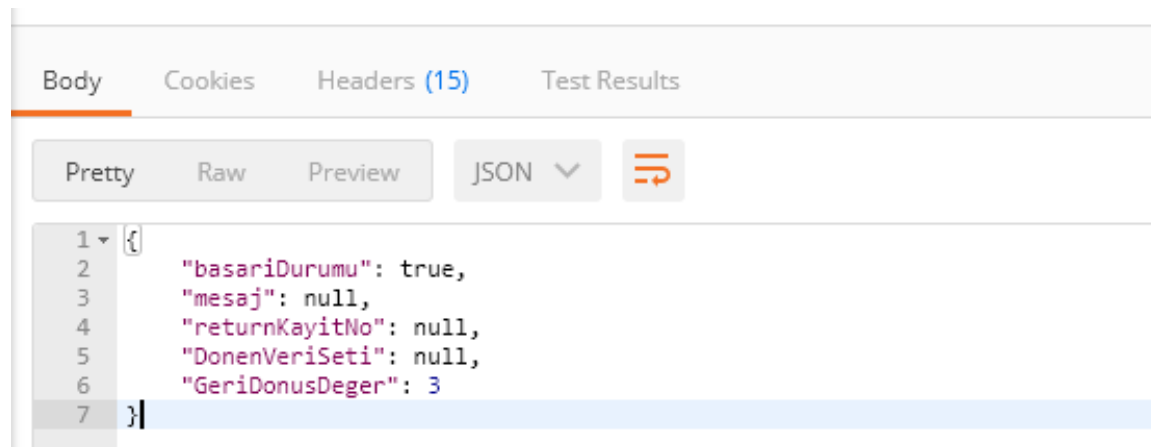


Figure 20: The response from the prioritization service.

Case study

In our case study, we prepared the most appropriate test group that has largest pre-prioritization data in order for the model to be trained better. However, our study showed us that even these

data are not adequate to be able to satisfactorily train the model. This was clear on the responses of the service, where every input resulted in the same priority. During discussions, this issue was reasoned on the insufficiency of the training data set. Training with larger datasets was ongoing during the final version of this document. However, this is only a matter of training and does not affect the prioritization methodology or the application set-up in our case study.

5. Knowledge-Based Testing: State of the art and gap analysis⁷

As described previously in the context of the XIVT project and this deliverable, we use the term knowledge-based testing in a broad sense to refer to different testing techniques, methods, and approaches that rely on and require (large-scale) processing of information from sources of various types (e.g., textual requirements specifications along with code, etc.). This section presents the current state of the art of knowledge-based testing and supporting topics for software product lines (SPL) on product variants systems, specifically on domains of automotive, railway, industrial production, and telecommunication software. We reviewed the role of machine learning, natural language processing, and evolutionary algorithms in test case generation and optimization. We also reviewed the literature on variability modeling, combinatorial testing, and model-based testing as a basis for test case optimization. Also, this section describes the risk assessment methods applied in such systems to aid the fields of security and safety. In addition, it discusses the limitations found in testing methods for these domains and gives a vision on how they can be addressed, but at the same time vision on how the current testing techniques and methods for SPL can be improved.

5.1. Change Impact Analysis (CIA)

Change impact analysis can aid the development and knowledge-based testing activities of variant-intensive systems. Therefore, in this section, we provide a brief overview of the state-of-the-art on change impact analysis.

CIA is defined as “identifying the potential consequences of a change in a system or estimating what needs to be modified to accomplish a change”. Adding a part of modification or new function to the current software implementation, one should extract test cases from the only scope of the impacted implementation for optimization of testing cost and software quality [15]. In addition, to analyze the cost of a change in terms of the estimated effort required to handle the change, CIA is important to examine the value added by a change.

CIA approaches can be classified in different ways. One classification divides the approaches into 1) Traceability approaches, 2) Evolution models, and 3) SPL assessment approaches [16]. Traceability approaches use traceability links and dependencies between SPL artifacts, evolution

⁷ This section is based on the current results of T2.1, the change impact analysis sub-section is based on results of T2.3.

models propose evolution models for SPL, whereas SPL assessment approaches assess the stability of a SPL. Another common subdivision of CIA methods refers to the artifacts that serve as the basis for the development and testing process [17], which we also follow below.

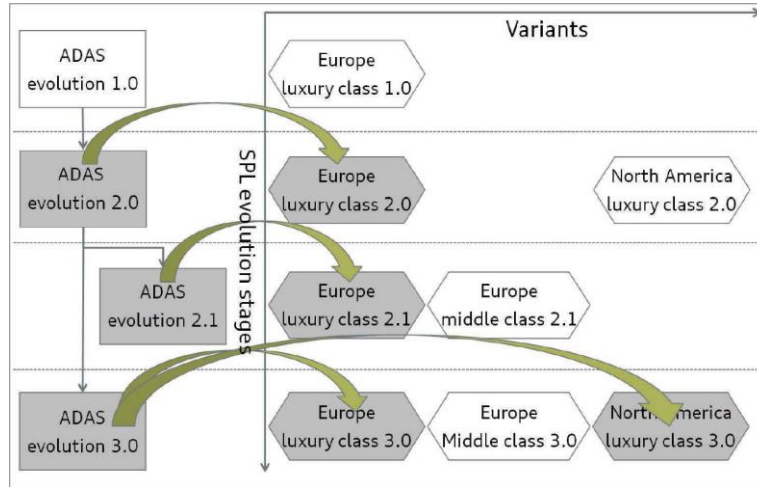


Figure 21: Example for affected variants by a feature change [18]

5.1.1. CIA Based on Requirements

Software systems often evolve due to changing requirements or due to refinements (feature requests, bug fixes) to the systems under development or maintenance. Finding the impact of specific changes in requirements on other related artifacts can assist engineers in evolution and in change management. Manually detecting the impact of a change on other related artifacts could soon become impractical as the number of artifacts could be quite large (particularly in a Software Product Line). Therefore, automated approaches are proposed to assist engineers in CIA. The approaches in this area could be categorized into two categories, as follows.

Requirements to requirements CIA. Goknil et al. [19] proposed a meta-modeling-driven approach to inter-requirements change impact analysis. The approach relies on manually identified dependencies among requirements. Particularly, relationships such as *requires*, *(partially)/refines*, *contains*, and *conflicts* are required to be identified among requirements. Furthermore, the approach also utilizes a change classification with classes such as *create*, *update* and *delete* for requirements, their properties and constraints, and their relationships. The requirement relationships and change classification information are used to identify the impact of a particular change (provided by the requirements engineer) on other requirements, and the approach also checks for consistency of the change. The approach is demonstrated on a benchmark case and is not yet empirically evaluated.

Arora et al. [20] proposed an approach for inter-requirement impact analysis that favors phrase-level structure and relatedness to identify other impacted requirements. The approach computes

the impact of changes under phrase-level propagation conditions provided by the engineer. In particular, first, the approach automatically identifies phrases within the requirements and computes word-level similarity across the phrases. When new changes are proposed, the approach maps them to related phrases in requirements and re-computes the similarities for the updated phrases. Then in an interactive manner, the approach allows the specification of propagation conditions on the phrases of the requirements. Finally, a sorted list of impacted requirements is generated and displayed to the engineer. The approach is evaluated on 14 change scenarios from two industrial cases.

Alkaf et al. [21] proposed a forward slicing-based approach to detect the impact of certain changes on other User Requirements Notation (URN) model elements. The approach supports impact computation for both Goal-oriented and Use-case map languages. The approach is illustrated on a benchmark case, and a user study is conducted for evaluating the accuracy and reduction of user-perceived difficulty.

Requirements to other artifacts to requirements CIA. Díaz et al. [22] focused on change impact analysis for product line architectures. The approach relies on product line architecture models and knowledge models to determine the impact of certain product line architecture model changes on other parts of the product line architecture model. Particularly, the approach combines traceability-based impact analysis with a rule-based approach to perform the change impact analysis. The approach is demonstrated on an example case.

Gethers et al. [23] proposed an approach for analyzing the impact of textual change requests on the software's source code. In cases where only the textual change report is provided, the approach uses information retrieval approaches to generate the impact set of source code using the textual change request. In addition, the approach can also utilize an already identified impacted set element as a base for impact set identification. Furthermore, run-time information (such as methods executed) could also be provided for better identification of the impact set. The approach is evaluated on four open-source cases.

5.1.2. CIA Based on Use Case Models

In many business contexts, use cases are the main artifacts for communicating requirements among stakeholders. In such contexts, Product Line (PL) use cases capture variable and common requirements, while use case-driven configuration generates product-specific (PS) use cases for each new customer in a product family.

Hajri et al. [24] proposed, applied, and assessed a change impact analysis approach for evolving configuration decisions in PL use case models. It automatically identifies the impact of decision changes on other decisions in PL use case models and incrementally reconfigures PS use case diagrams and specifications for evolving decisions. It is aimed to improve the decision-making process by informing the analyst about the impact of decision changes and to minimize manual traceability effort by automatically but incrementally reconfiguring the PS use case models, that is to only modify the affected model parts given a decision change and thus preserve as many traceability links as possible to other artifacts. A case study in the context of the automotive domain was performed. Further, they conducted structured interviews and a questionnaire study with

experienced engineers, suggesting that their approach is practical and beneficial to analyze the impact of decision changes and incrementally reconfigure PS models in industrial settings.

5.1.3. CIA Based on Source Code

The reuse of software artifacts (e.g., source code and test cases) enables the construction of SPL products that are driven by features and configuration modules that the final products should provide. Although modules hide internal implementation details, they are not independent and frequently communicate with each other, which means that changes in one module often affect other modules. Therefore, it is important to consider such dependencies and the source code changes, e.g., when determining the set of impacted artifacts after modifying source code [25].

Cloning software increases the number of clones, and thus their management and maintenance can be a difficult task. To reduce this, an extractive SPL approach would be beneficial. Lima et al. [26] therefore propose a product line architecture (PLA)-based approach to capture the structure of a set of existing projects and create a PLA that can serve as a basis for the creation of new SPLs by supporting the developers' activities and providing adequate support to meet the PLA requirements. As a result, PLA will provide a set of consolidated UML diagrams (packages and classes) and Design Structure Matrices. The approach reuses and enhances existing techniques and tools developed for architecture recovery of single systems to support variability identification from source code and variability documentation at the architectural level [26, 27]. On the other hand, software systems need to be highly modular to cope with challenges such as size and complexity, multilingual development, and variability of requirements. Hence, understanding the modularity of large-scale systems is essential for program analysis [25]. In addition, feature implementations can be changed to adapt SPLE context. The change may impact other features that are not affected by the change as the feature's implementation spans multiple code elements and shares code elements with other features [28]. All of these three aspects are relevant to Software Change Impact Analysis (CIA).

Software engineering has not yet been able to propose a clear solution to the question of whether or not the effects of a partial modification of the system affect the entire system or up to what range. One of the causes is that many existing methods focus on the I/O pattern of the system, but the behavior of the actual system depends on the status of the external system and any flag information. Hence, there is a need to take into account the interactions between function and data; hence, a method that focuses on functionality and data dependencies, i.e., Impact Data All Used (IDAU). Takeda et al. [15] propose a test case extraction method and test architecture using an improved IDAU method that focuses on function and data dependency. The approach is an improved IDAU method for internal specifications or implementation source codes. Graph mining techniques are also used to efficiently reduce the number of test cases.

At the feature level, the CIA is far from a trivial task when dealing with a large number of features. Manually tracing feature implementations to determine which features are affected is time-consuming, error-prone, and tedious. CIA is seldom considered at the feature level for changes at the source code level. Most existing work performs CIA at the source code level, while little work is performed at the requirements and design levels. Eyal-Salman et al. [28] propose a technique

based on Formal Concept Analysis (FCA) to study CIA at the feature level. This technique takes as input a changeset composed of classes to be changed and computes as output a ranked list of affected features. Each feature in this list has a probability of being affected, representing the feature's priority to be checked by maintainers. Also, two metrics were proposed to measure the extent to which a particular feature implementation is impacted, i.e., the extent to which a particular feature may be affected (IDM, *Impact Probability Metric*), and the changeability of features for determining the percentage of features that are affected by a particular change (CAM, *Changeability Assessment Metric*).

Angerer et al. [25, 29] present an approach that exploits the modularity of large-scale systems to first perform program analysis for individual modules and then compile the previously computed analysis results. However, partitioning a system dependency graph is not straightforward as it contains presence conditions representing variability. Hence, the authors propose the use of placeholders that are resolved when composing the precomputed System Dependence Graph (SDG) modules during configuration-aware program analysis. The approach is particularly useful in the context of SPL when product variants are derived by composing modules depending on specific customer requirements.

5.1.4. CIA Based on UML Models

Model-driven Engineering (MDE) and Model-based Testing (MBT) are methodologies for developing and testing systems based on domain models, serving as formalized descriptions of domain knowledge, requirements, design decisions and other information influencing the development. The Unified Modeling Language (UML) is a domain-independent and commonly used modeling formalism for structural and behavioral modeling of systems of any kind. A central objective of MDE and MBT is increasing efficiency and quality through automated derivation of development artifacts, such as code, documentation and intermediary models, through model transformations. Apparently, changes to the models will result in changes to the derived artifacts and change impact analysis (CIA) is used for determining the exact nature of this impact.

A common application of CIA is regression testing, in which a new version, i.e., an incremental variant, of an existing product is tested. For reasons of efficiency, it may be desirable to reuse test artifacts between different variants or even to reuse test results, provided them relating to only unchanged properties of the system under test can be ensured. A similar approach can be applied to testing software product lines (SPL), where instead of incremental variants, different product variants need to be tested. In the following, we present two approaches to this challenge.

Lity et al. [30] transfer the concept of delta modeling, commonly used for capturing the variability of an SPL, to describing the differences between versions of an SPL. They refer to this twofold application of delta modeling as “higher-order delta modeling”. In this context, a “delta” is an explicit and formal description of the changes required to be made to one variant in order to create another. Based on an analysis of the change operations comprising a delta, i.e., addition, removal or modification of model elements, deltas can be used for determining those product variants that

are affected by a change in the product line. In the context of regression testing, this information can be used to decide which variants need to be retested.

Lity et al. [31] suggest a method for determining test cases instead of variants that should be retested, based on incremental model slicing. Model-slicing is a technique in which a model is reduced to a sub-model ("slice") containing only those elements related to a specific slicing criterion. As the slicing criterion, the authors use the individual transitions of a state machine as representatives of test goals. Again, they use delta modeling for describing the differences between variants and apply the deltas not only to the core and variant models but also to the individual slices. This way, changes in slices can be used to determine whether a test case targeting a specific test goal has become obsolete, can be reused, or must be adapted.

5.1.5. CIA Based on Feature Models

Features are more structured and coarse-grained than requirements and therefore facilitate the understanding and traceability of an SPL evolution. Managing SPL evolution involves first managing changes at the problem space level (i.e., the feature model) and then tracing these changes to the solution space (i.e., shared assets like source code, design and test artifacts). The two main issues are 1) maintaining consistency between the feature model and the remaining assets, particularly the design, and 2) measuring the effort required to manage each change impact.

Traceability information between the SPL feature model and its assets is the cornerstone of SPL change impact analysis. In addition to traceability, SPL evolution must also address the following issues, which are closely related to change impact analysis: consistency of the adopted evolution, effort estimation in model evolution, and risks involved in model evolution. Evolution consistency means that the evolved SPL models are internally (the feature model and each asset separately) inter-consistent. For effort estimation in model evolution, a set of metrics for software evolution obtained from software change records is used. Finally, risk management aims to reduce potential risks and provide opportunities for positive performance improvement [32].

Maâzoun et al. [32] proposed an automated method that offers a set of recommendations to ensure the consistency between the SPL feature model and its design. It identifies the set of changes required to propagate each type of change from the feature model to the design. The automation is provided by a precise definition of every change operation in terms of its pre-condition, post-condition and impact on both the feature model and the design. Propagation is ensured by a UML profile for SPLs that explicitly links the SPL feature model to its design.

Using multiple feature models, each containing a reduced number of features relevant only to specific artifacts, has been identified as a possible solution to address the high complexity of evolving large-scale variable systems. Composition rules or cross-feature model constraints can be defined at a feature model level to specify how models should be recombined for analysis. However, these techniques require recomposing the models before validating the various

configurations, which is difficult to automate. Instead, Dintzner et al. [33] use partial information about constraints between feature models. Their approach identifies the configurations that can no longer be derived in each individual feature model, taking into account the impact propagation of feature changes across feature models. Furthermore, they developed a supporting tool “FMDiff” that proposes a feature change classification that details changes in features, their attributes, and attribute values. They apply their approach to the Linux kernel feature model and extract feature changes occurring in sixteen official releases [34].

Kahraman et al. [35] exploit the explicit representation of relationships between multiple feature models. They propose a modeling language that can be non-intrusively overlaid on existing multiple feature models to express the relationships between feature models. The language uses the concept of views to represent different aspects of the product line variability model, where each view corresponding to a feature model represents variability information in one or more development artifacts of the product line. A metamodel for relationships between feature model views and the semantics of the relationships is provided. It is shown how the proposed relationship types can be used to analyze the impact of a change on the entire variability model.

5.1.6. CIA Based on Architectural Models

Regarding the change impact analysis of architectural models, it is very rare that architectural documentation for industrial systems conveys sufficient information to be effective in erosion minimization and prevention needed in an SPL [36]. Research literature distinguishes three main activities involved in architectural model repair: recovery [37], discovery [38], and reconciliation [39]. In other words, mechanisms are used to elicit architectural details from other development artifacts (e.g., requirements, code, documentation). Based on these details, consistency problems can be analyzed and possibly fixed. There exists extensive research on exploiting code and its evolution history for recovery activities; moreover, implementation and run-time information can be used together with requirements and use-cases as support for discovery tasks. Once the necessary architectural information is recovered into models, it is possible to enact reconciliation mechanisms to manage evolutions.

5.1.7. Risk estimation based on CIA

Inherent risks produced by obsolete components need to be minimized. A model-based systems engineering approach is proposed by [40], which uses obsolescence considerations to allow the identification of components potentially at risk of obsolescence. The likelihood is rated by Technology Readiness Levels (TRLs), while the potential impact of the risk on the whole system was represented through several interfaces. The results revealed that avoiding the use of high-risk critical components can prevent unnecessary and unplanned expenses due to re-designing and reworking further down the life cycle. They suggested evaluating the impact of early risk analysis. They recommended evaluation of the impact of early risk analysis of the design on the life cycle of a system.

Implementation of the risk management tools is crucial for assessing the maturity of the technology and the respective risks that affect deliveries. A tool for managing the risks was proposed by [41] for projects that include the development of technological innovation and new products. The process integrates three concepts: Risk Matrix, the TRL, and the IRL (Integration Readiness Level). The TRL helped with analyzing the technological maturity at the component level, while IRL assesses the integration of the involved technologies and their interaction within the project environment.

Risk estimation plays a crucial role in the development of product and technology integration programs. The probability of a technical event and its corresponding cost is considered a technical risk. A technical risk assessment technique was proposed in [42] as an attempt to estimate the probability of change propagation. It provides an abstract view about the potential costs, and it is a conceptual model to estimate technical risk regarding the risks associated with modifying software elements within their changing systems. Through the development of a software tool (called TRE), it enables organizations to ground their development plans that the system has undergone both in the structural nature of their source code and the past development history.

There are many decisions support tools available in the literature that enables identification and mitigation of the risks in a project. However, only a few explicitly consider the effects of architecture on risk. For example, a risk estimation framework is proposed in [43] that includes considerations of the system architecture. They define risk as a combination of likelihood and impact should a change be required. The Technology Readiness Levels (TRL) were used as a measure for likelihood, and the impact was estimated through connectivity-related measures. By applying the framework to an industry example, they showed that the estimated risks were in line with the experience of engineers at the company.

Another decision support tool that considered architecture on risk assessment is presented in [44]. It is a risk-based change-impact analysis that is proposed to identify system variants relevant for retesting after an Over-the-Air-Update received by the road vehicle. Due to combinatorial explosion, testing all variants of such highly configurable systems is infeasible. Thus, they combined different concepts from product sampling, risk-based testing as well as configuration prioritization and implemented them to the automotive architectures. Through validation of their approach, they showed a reduction in testing effort by identifying and prioritizing incompatible variants w.r.t. the system update. However, their approach has not been validated through implementation on a large real-world system yet and knowing which factors affect the risk-based configuration prioritization is still an ongoing study.

5.2. Test process optimization

In software system development, testing can take considerable resources, and there are numerous examples in the literature of how to improve the testing process. Moreover, the improvement of the software testing process plays a vital role in software development life cycle improvement. Test case optimization techniques that are employed to decrease the number of test cases to be tested can be categorized into two general focus areas: 1) generating test cases optimally (i.e., optimized test case generation) and 2) optimization of the generated test cases and existing test suites.

To get optimized test suites, models may need to be employed, or else automation tools are used to generate the test cases. In both cases, machine learning-based techniques can be used to select the input data so that the generated test suite (number of test cases) should be optimized. In the last two decades, there has been a gradual increase in the literature related to optimized test case generation. Optimization of the generated test cases can be further grouped into the following three (03) sub-categories:

1. Test Case Prioritization
2. Test Case Minimization or Reduction
3. Test Case Selection

Test case prioritization ranks the test cases according to the performance criteria set normally prior to the test execution phase. Mostly, prioritization of the test cases needs a quantitative criterion based either on some sort of the coverage requirements or for the fault detection target, or in some cases, empirical data governs the prioritization effort. Initially, J. Harrold et al. defined the problem of test suite reduction in [45], but later it was named test suite minimization. Test case reduction aims at the removal of the test cases having redundant components as testing targets. The major difference between test case selection and test case minimization/reduction is that test case selection focuses on changes in the system while test case reduction focuses on redundancy in the single version.

Test case selection is a varied form of test case prioritization. In test case prioritization, tests are prioritized over others in execution, so we can get the maximum benefits in the least possible effort, while in test case selection, test cases are selected and included in the test suite with the aim that only this suite would be executed, and the rest of the test cases are not executed. Test case selection selects a few or all test cases from the test suite in a subset by a predefined criterion depending upon the requirements.

5.2.1. cursory Survey of Search based Test Case Optimization Techniques

Over 200 studies have been published in the last ten years aiming at the utilization of search-based techniques in test process optimization. Most of these search-based test studies focus on

the optimized generation of test data, while test case prioritization has remained popular over time too. Genetic algorithm, reinforcement learning, cuckoo algorithm, particle swarm optimization, memetic algorithm, ant colony optimization, genetic strategy and many customized heuristics have been employed by different researchers for optimized generation of the test data. Some of these studies have opted for a single algorithm, while many others combine one heuristic in a whole test suite generation approach with another one. Authors in [46] have proposed to rank test cases generated according to their likelihood to reveal a fault. The test case generation problem has been analyzed as a multi-objective optimization problem as well; authors in [47] have considered coverage targets as their objectives. In comparison, study [48] uses a multi-objective GA optimization and aims to produce a smallest test suite with the highest possible coverage for the given class. Authors in [49] have used GA along with Incremental Genetic Algorithm (IGA) and CFGs to generate the optimal test data for branch coverage. In another study, optimal test data generation was considered as a multi-objective problem addressing the path coverage and fault detection objectives [50]. A customized multi-objective search-based technique called Android Genetic Ripping has been designed for Android applications to meet the objectives of efficiency and effectiveness of generated test data [51]. In Study [52], GA has been used to apply search-based mutant selection to enhance the quality of test suites efficiently. A genetic strategy has been applied in [53] to decrease the test suite size and increase the speed of test suite generation.

Many of the search-based heuristics have been used to prioritize test cases and test data as well. Like the optimal test data generation, these techniques for prioritization have been based on popular evolutionary algorithms like GA, PSO and ACO, etc. In one such Study, a metaheuristic algorithm named "Proportion-Oriented Randomized Algorithm (PORA)" has been used to measure the distance among the different test case clusters. The algorithms aim to use the prefix evolution and efficient approximation for this purpose [54]. NSGA-II has been used to prioritize the test cases to achieve maximum coverage in a video conferencing system of CISCO [55]. The proposed algorithm, STIPI, has been compared on 211 different test cases taken from CISCO and has been compared based on five approaches in different time budgets. Another study focusing on prioritization of test cases for integration testing in software product lines has used the concepts of delta modeling [56]. This technique has used the differences between product variants in the specification. Similarly, a parallel genetic algorithm has been employed in Software product lines with smaller arrays [57]. This parallel GA has been used to acquire an acceptable performance using a Parallel Prioritized product line Genetic Solver (PPGS). A greedy algorithm-based solution has been employed in cyber-physical systems product lines as well. This solution prioritizes the test cases based on the reduced fault detection time and requirements covering time [58].

Most of the search-based techniques employed for Test Suite Reduction/minimization are derived from the famous evolutionary algorithms like genetic algorithms, Particle swarm optimization and simulated annealing, etc. These algorithms have been modified to meet the requirements of different stakeholders. In [59], GA and SA have been integrated to reduce the size of a test suite. Authors have proposed a GNA_SA strategy, where the target meta-heuristic is augmented by the host meta-heuristics. While exploring the tradeoffs among the different objectives of the test suite reduction, authors in [60] have proposed a technique that evaluates test-suite reductions based

on killed mutants rather than a traditional coverage. In [61], the authors have used a hybrid algorithm (self-designed greedy algorithm) to provide the larger set of solutions for test case selection based on good quality and diversity. Multi-objective optimization based on different algorithms has been used for the selection of test cases as well, e.g., using MOPSO for test case selection in [62], Pareto Based Multi-Objective Harmony Search for test case selection in [63], and a binary multi-objective PSO with Crowding Distance in [64].

5.3. Product Line Testing

According to the popular definition from CMU SEI, “a software product line (SPL) is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way”. Regarding the widespread use of product lines in almost all industrial domains, quality assurance of variants is a big challenge that has not been adequately dealt with before. There are many scientific papers dealing with different aspects of the problem (see [65, 66] for surveys of the literature). However, it is still highly difficult to guarantee that all products are working well and as expected since the following constraints will hinder the test process.

- The number of combined products grows exponentially with increase in the number of modules or features.
- There is always a limited number of resources in terms of time and computation power to run tests on a specific product.
- Each product derivation for test will cause additional costs.

Therefore, testing a product line is highly crucial because a fault in a (base-line) module or component could spread in hundreds of products. In general, two types of strategies can be used for product-line testing: Product-based (product-by-product) and domain-based (or family-based) testing. Product-by-product testing focuses on testing the products individually. In this strategy, products are generated and tested based on the testing methods for a single product. Product-based testing is called “optimized” if it uses a subset of all products; this approach is also known as sample-based testing. Otherwise, it is “unoptimized” when it analyzes all the generated products in an exhaustive or brute-force fashion [67].

One of the primary advantages of product-based testing is that the existing testing techniques and tools can be easily applied to the scope of product-line testing. Moreover, product-based testing can easily deal with the changes of product lines since only products changed need to be tested again. Therefore, product-based analysis is widely used in industries. However, product-based testing strategies reach their limits when the number of variants is huge, or when the quality assurance must include products that might never be ordered.

On the other hand, a family-based testing strategy operates on domain artifacts and uses the domain knowledge about the valid component/feature combinations. It often uses a representation of a virtual product that contains all the components/features rather than considering any particular product [67]. Family-based strategies reduce the computation redundancy since they avoid

duplicate analysis among the products. The performance of this strategy is mainly independent of the number of derived products, while the required sources/effort for product-based strategy increases with the number of new products. Whereas in a product-based strategy existing testing methods and tools can be used, they are not applicable directly as they are to family-based strategies. In addition, effective testing should consider the effects of the hardware and execution environment, i.e., the interplay between software and hardware in each product variant.

Sampling is an essential phase of optimized product-based product-line testing. Several approaches have been proposed for selecting a subset of generated products. One of the common approaches is Combinatorial Interaction and in particular T-wise combinatorial interaction, which systematically reduces the number of products under test. Overall, T-wise interaction coverage operates based on this observation that most of the faults occur during the interactions between at most T features/modules of the product. A number of different methods, including CASA, ICPL, IPOG, and IncLing have been proposed to perform the T-wise interaction sampling. The output of these algorithms is the minimal set of products covering all combinations of features/components. A number of machine learning algorithms can be applied to automate the sampling step in product-line testing.

5.4. Machine Learning and Knowledge-Based Testing

Machine learning is an analytical technique that does the same process which comes naturally in the mind of humans during the learning process. It “learns” new information from data without any a priori knowledge or mathematical model. In general, the performance of machine learning algorithms improves adaptively with an increase in the number of training samples.

Nowadays, machine-learning algorithms are widely used for knowledge discovery within data sets in many application areas like automotive and aerospace industries, finance, image processing, energy management and demand response. The learning algorithms try to find patterns and extract knowledge from data for prediction purposes. In some cases, learn the optimal state-action policy according to the received reward from the environment. Generally, three main types of learning techniques namely supervised learning, unsupervised learning, and reinforcement learning, are considered as categorizations of machine learning techniques. Supervised learning finds a model on a training data set (including known input and output). The extracted model is used for prediction purposes. The supervised learning algorithms work based on classification or regression. Unsupervised learning explores data to find hidden patterns/structures. It can be useful for reducing the dimensions of data. Cluster analysis techniques are the most common algorithms in the category of unsupervised learning. Clustering techniques can be a form of hard clustering or soft clustering. *Reinforcement learning* is a learning approach based on interactions with the environment. A system using this approach learns an optimal policy to reach a goal via sensing the environment, taking a possible action and receiving a reward signal from the environment. In other words, it tries to learn from experiences.

Machine learning provides very powerful tools which can be used in many areas within software testing.

5.5. Model-Based Testing

Model-based design (MBD) is a particular form of software development, where a system model is continuously used as the central artifact throughout the whole engineering process. Initially, requirements are captured in an abstract model, e.g., in SysML, representing the system specification. This abstract model is refined and transformed into a concrete implementation model, e.g., in UML or Simulink®. From this implementation model, executable code is generated automatically by a suitable model compiler.

Model-based testing (MBT) is a technique within the MBD paradigm, where test cases are generated from models describing some relevant aspect of the system under test. MBT has been proposed as a suitable way of replacing time-consuming and costly manual processes with model-based tools for the creation, execution and evaluation of tests. MBT promises considerable improvements in productivity, quality and cost savings. The basic idea [68, 69] is to provide more cost-effective testing of complex systems. Instead of manually writing a large set of test cases, a smaller set of test models are built to describe generally the behavior of the system and how it should be tested. A test generator tool is then used to generate test cases from these models automatically. There are several benefits [70], including easier test maintenance due to fewer artifacts to update, higher test coverage from the generated test cases, and documenting the behavior in higher-level models which helps in sharing the information and understanding the system. The UML (Unified Modeling Language) testing profile, UTP, is a standard for specifying model-based testing.

Model Testing. MBT techniques can be used in model testing [71, 72] as a way to raise the level of abstraction of testing by executing tests on the model instead of the implemented system. In this context, models represent any relevant information regarding the software behavior, structural information, environment, or other extra-functional properties. The goal of model testing is to help practitioners in the design of models by using the advances in software testing [73] in the generation, execution, and fault detection of the models. Other research areas related to model testing are model checking, model simulation and model-in-the-loop testing.

Test Generation. A wide variety of MBT tools exist, each with its own set of features and test generation algorithms. A far from a complete list of MBT tools can be found at [74]. MBT test generation can be divided into two categories based on how the generated tests are executed: offline testing and online testing [75]. In off-line testing, the tests are first generated in their entirety, and the resulting test cases are then executed in a separate step. This approach fits well into both traditional and agile development processes, with model-based testing simply replacing manual test creation. In contrast, online testing executes the tests as they are being generated. The advantage is that test generation can react to unexpected events in execution, making testing of nondeterministic systems easier.

5.6. Combinatorial Testing

Combinatorial testing has been suggested as an effective method of creating test cases at a lower cost. However, industrially applicable tools for modeling and combinatorial test generation are still scarce. As a direct effect, combinatorial testing has only seen a limited uptake in the industry that calls into question its practical usefulness. This lack of evidence is especially troublesome if we consider the use of combinatorial test generation for industrial safety-critical control software, such as are found in trains, airplanes, and power plants.

Combination test generation techniques are test generation methods where tests are created by combining the input values of the software based on a certain combinatorial strategy [76]. For example, base-choice [77] is a combinatorial technique used to generate tests by varying the values of one input parameter at a time while keeping the values of the other input parameters fixed to a base choice until all of the remaining combinations have been generated. However, this strategy might not be as effective when used on CPS because of its inherent limitation of just generating the right choice of values and not also considering the timing of the input parameters as well as the behavioral states. Work has been done to extend the base-choice coverage criterion by including timing [78] for testing CPS. The results show that timed base-choice generated test suites achieve better code coverage and fault detection compared to other combination test generation techniques. The use of combinatorial testing techniques needs to be further studied in variability testing and consider the implications of using multiple base and time choices as well as other extra-functional aspects. SEAFOX is the only available combinatorial test suite generation tool for industrial CPS software and can be used in XIVT for testing CPS. SEAFOX is open source software and is available. SEAFOX supports the generation of test suites using pairwise, base choice and random strategies. For pairwise generation, SEAFOX uses the IPOG algorithm as well as a first pick tiebreaker. A developer using SEAFOX can automatically generate test suites needed for a given program after manually providing the input parameter range information based on the defined behavior written in the specification.

5.7. Variant Modeling

In order to employ model-based techniques to highly configurable systems, it is necessary to model the variability in the intended product line. Although there have been several suggestions, no common industrial standard has been reached yet. There are several ongoing projects, also within ITEA, to deal with this question. Several of the present partners collaborated in the ARTEMIS VARIES project on the specification of variability in safety-critical embedded systems. The project contributed a consistent, integrated and continuous variability management method and tools for the entire product life cycle, as well as effective variability architectures and approaches for safety-critical embedded systems. A failed standardization attempt was the OMG common variability language, CVL, in 2012. Apart from legal issues, this approach was not specifically directed at testing.

The monograph [79] presents the results of the VAMOS project on variant modeling with SysML. In the book “Advanced Model-Based Engineering of Embedded Systems” [80], members of the current consortium report on the outcome of the German SPES_XT project, where they extend the SPES meta-model to capture the orthogonal concern of variability. Moreover, building blocks for realizing variant management and planned reuse, as well as for the assessment of variable artifacts, are presented. A result of the SPES project highly relevant for XIVT is the Variability Exchange Language (VEL) for the export and import of variability information between different tools. VEL allows creating an Entity-Relationship model, which defines features as being entities and relationships between them, with cardinalities and obligatoriness of entities.

5.8. Software Security Testing

Software testing plays an important role in the life cycle software development. One of its main goals is test software for discovering vulnerabilities or faults. This means that security testing validates software system requirements related to security properties, such as confidentiality, integrity, availability, authentication, authorization and non-repudiation. For that, security test models provide guidance for the systematic and effective specification and documentation of security test objectives and security test cases, as well as for their automated generation and evaluation [81]. To archive this, a set of test cases are generated and executed over the software under test (SUT) such that it reveals as many faults as possible by satisfying specific criteria [82], such as path coverage, code coverage, feature coverage, and so violate some security property. Furthermore, depending on the criteria we want to cover, different techniques to generate or select test cases and to generate test data can be applied.

Orso et al. [73] conducted a study on such techniques based on model testing for software testing in the generation, execution, and fault detection. However, for software security testing, generating test data is one of the most used and important methodologies, being symbolic execution and fuzzing the two techniques applied for that, having as coverage criteria path and code coverages.

Another important feature in security testing is the result of the test cases. Having in mind that a security fault is manifested by its exploitation on SUT, i.e., by the behavior of SUT when test cases are executed, therefore, from a set of generated test cases, those that their results do not reveal any failure or bug/vulnerability are considered useless [83]. However, such tests can be interesting if they discover new execution paths or uncover code, meaning that new test data can be generated in order to exercise these paths and discover new faults. Therefore, generating test data that can expose faults or exploit vulnerabilities presented in SUT is crucial. Also, since the security risk assessment of software is measured by faults, it contains and their severity, testing software adequacy and intensively is very important.

Zhang et al. [50] proposed a method of generating test data for multiple path coverage with faults detection. First, the method builds a mathematical model capable of representing a multi-objective optimization problem with constraints for covering multiple paths and detecting faults. Afterward,

the model is solved based on a weighted genetic algorithm. The method generates test data that, besides traversing the target path, detects faults in the path.

In order to maximize the failure detections within time and resource constraints, Matinnejad et al. [46] proposed a test prioritization algorithm to rank test cases automatically generated by its test generation algorithm according to their likelihood to reveal a fault. Unlike traditional existing test prioritization techniques, which mainly rely on dynamic test coverage information to prioritize test cases, i.e., test cases that get higher program structure coverage are prioritized higher, the approach proposed by the authors to rank test cases leverages from a combination of test coverage and fault-revealing probabilities of test cases. As a result of their approach, engineers can select, according to their test budget, a subset of the most highly ranked test cases.

Rebert et al. [84] presented a way to optimize test case selection in order to increase coverage of the software under test. They show that current test input selection strategies found in Peach do not have better results than randomly picking the test cases. They also show ways to improve the test input selection strategies to maximize the total number of vulnerabilities found during fuzzing. The results of the experiments showed an increase in vulnerability detected when compared to Peach and a reduced testing set size—allowing more bugs to be found with fewer testing cases.

Felderer et al. [81] based on model-based testing (MBT) and security properties, proposed a model-based security testing (MBST) taxonomy. MBT derives of variant testing based on models that encode information on the SUT and/or its environment. On the other hand, MBST is MBT of security requirements. The authors realized that MBT is insufficient to cover all specific security aspects since MBT focuses on functional testing. Hence, they defined a classification criterion for MBST based on filter criteria (i.e., model of system security, security model of the environment and explicit test selection criteria) and evidence criteria (i.e., maturity of evaluated system, evidence measures and evidence level). Such defined criteria complement existing classification schemes for MBT by security-specific aspects.

5.9. Risk assessment of critical product variant systems

It is not always possible to execute all the test cases in regression testing for the product variants due to limited time and resources. Therefore, prioritization and selection of the requirements for testing not only plays a crucial role in making efficient use of testing resources (which is also a part of CHALLENGE-2 of the XIVT project), but also it could be used to guide towards an efficient test case generation process. The most important factors in test prioritization and selection of variant-intensive products are achieving safety and security requirements. This can be done by the overall process of identifying, analyzing and evaluating risks, known as risk assessment.

Different types of hazards exist, such as biological, chemical, physical, and safety. The safety risk assessment includes three steps: hazard identification, risk analysis and evaluation, and risk control. The first step is identifying hazards and risk factors that have the potential to cause harm.

Analyzing and evaluating the risk associated with that hazard is the next step. Finally, elimination or controlling the hazard is conducted in the last step. It is an inherent part of risk management which is defined as a decision-making strategy that uses the quantitative values obtained from risk assessment models together with the insight, experience, and professional judgment. On the other hand, security risk assessment is the process of identifying and analyzing threats to implement adequate security measures. The major difference between safety and security modeling is the classification of hazards (safety) versus threats (security).

Security risks appear in different shapes, sizes, attack vectors, and potency levels in cyberspace. An attack vector is a technique to help hackers to gain unauthorized access to a device or a network. Some examples of attack vectors are mobile apps, servers, Wi-Fi, sensors, USB port, Bluetooth and broadband networks (e.g., cellular). Security can be categorized into physical security and logical security (also known as cybersecurity). However, they are not entirely independent of each other. Physical security aims at preventing unauthorized access to facilities, equipment and resources to protect personnel and property from damage or harm. Cybersecurity, on the other hand, is the protection of internet-connected systems, including hardware, software and data, from cyberattacks. Since our focus is on cybersecurity, the term 'security' will mainly refer to 'cybersecurity' throughout this section. In the following, a state-of-the-art literature review is conducted on safety and security risk assessment methods for product variants systems on the automotive, railway, industrial, and telecom domains. Besides, the standards/guidelines which are the basis for commonly used approaches for both safety and security risk assessments are introduced. Finally, a summary at the end of the section briefly lists the standards/guidelines along with their scoring methods.

5.9.1. Safety risk assessment

In this subsection, we review the safety risks associated with automotive, rail, industrial and telecom domains in the scope of the XIVT project to specify the current state of the art in methodology and standards/guidebooks selection.

5.9.1.1. Automotive domain

One of the most significant quality attributes of a vehicle through all stages of its life is safety. It requires particular care and attention. There are various aspects of the vehicle's overall safety, such as passive safety, active safety, and functional safety [85]. Active safety features are those that assist in avoiding or mitigating road crashes. They are employed for either crash prevention or severity reduction of an unavoidable crash. Some available features are collision avoidance systems, anti-lock brakes, and lane departure warning systems. Passive safety features are those that help to avoid further injury of the vehicle occupants in a car accident, including features such as airbags and seat belts. They do not perform any function until called to action.

The term functional safety refers to the absence of unreasonable risk introduced by systematic failures and random hardware failures [86]. Many functional safety standards have been developed for safety-critical systems [87, 88, 89]. A generic functional safety standard for electrical and electronic (E/E) systems within the scope of road vehicles is introduced in 2011, known as ISO 26262 standard [90]. It covers both hardware and software. In the automotive domain, the ISO 26262 standard corresponds to the state-of-the-art in functional safety of road vehicles at the date of this document. To achieve compliance with ISO 26262 a huge amount of manual work is required which makes it very costly and time-consuming [91]. Several works dealing with functional safety requirements (based on ISO 26262) for automotive systems have been proposed in the literature [92, 93, 94].

An overview of the current technological challenges in onboard and networked automotive systems is presented by Bello et al., [95], which is focused on functional safety analysis concerning SW/HW. An approach is proposed by Gharib et al. [96], which integrates both the E/E systems and the driver's behavior.

A review of recent advances in automotive functional safety design methodologies is conducted by Xie et al. [97]. It includes automotive E/E architecture, standards, etc. The paper summarizes recent progress in four categories functional safety analysis, functional safety guarantee, safety-aware cost optimization, and safety-critical multi-functional scheduling.

An initial study by Henriksson et al. [98] suggests the changes required to be implemented, allowing safety-critical machine learning development in the automotive context. A survey has been conducted on the safety of machine learning (ML)-based systems [99]. They have provided a structured, certification-oriented overview of the existing methods to support the safety of the ML-based systems and identify current open challenges for safety argumentation. These systems require their particular approach depending on the properties of the ML algorithm. A theoretical framework for safety verification of ML-based systems, such as automotive, is proposed by Kaindl et al. [100], where probability calculations are done for determining a SIL or ASIL. They claimed that it results in better safety verification for such systems compared to current approaches applicable to traditional software.

A comprehensive hazard analysis approach is proposed in [101] based on different safety hazards analysis techniques such as Preliminary Hazard Analysis (PHA), Reliability Block Diagrams (RBD), Fault Tree Analysis (FTA), Failure Modes Effects Analysis (FMEA), and Failure Modes Effects and Criticality Analysis (FMECA). A qualitative comparison of the FMEA and system theoretic process analysis (STPA) safety analysis methods are conducted in [102], which both deliver similar analysis results. A risk analysis FMEA approach is developed in [103] based on the Fuzzy Set Theory to provide the estimated risk priority number (RPN) more precisely. Through sensitivity analysis, they showed that expert traits have less impact on fuzzy-RPN estimation compared to a detectability index. This makes the proposed Fuzzy-FMEA model a credible alternative to risk analysis. Another fuzzy-based FMEA approach is proposed by Qin, et al. [104]

to deal with uncertainties which are more efficiently compared to the traditional FMEA approach. It is an evidential reasoning method under an interval type-2 fuzzy environment that considers the weight of three risk factors in the construction of RPN parameter.

Following the ISO 26262 standard, Hazard Analysis and Risk Assessment (HARA) is required to determine the criticality of the system under consideration and ensuring functional safety. A safety analysis method for autonomous vehicles (AVs) for a public transportation case study based on HARA is described in [105]. An example of a HARA for an automated unmanned protective vehicle without human supervision is proposed in [106]. They argued that current conventional HARA approaches might not be suitable for future applications with a wider range of functions. In HARA, different test engineers usually provide different classifications for the same safety goal [107]. An automatic and systematic HARA analysis method for Advanced Driver Assistance Systems (ADAS) is proposed by Sini et al. [108]. It utilizes a simulation-based approach that enables a more objective and repeatable analysis. Description of operational situations, high-level functional as well as classification rules in terms of severity and controllability are required to adopt this approach.

Since the current automotive safety standard ISO 26262 is not intended for fully automated driving vehicles, a systematic approach based on STPA to drive operational safety requirements and develop operational safety concepts is provided by Abdulkhaleq et al., [109]. However, compliance with ISO 26262 at different architecture levels was not investigated and needs to be considered. It should be noted that the standard ISO 26262 only deals with avoiding malfunctions focusing on electrical and electronic (E/E) systems. There are several examples of vehicle failures that exist in spite of software and hardware compliance with ISO 26262 [110]. Recently, ISO PAS 21448:2019, commonly referred to as SOTIF, standing for ‘Safety of the Intended Functionality’ has been proposed (the first version published in 2018) [111]. SOTIF is intended to complement ISO 26262 by considering issues relating to electronic systems that govern the safe operation of a vehicle. The ISO 21448 standard is expected to be released by the end of 2021. An integrated safety assessment approach for hazard analysis and risk assessment (HARA) in the automotive domain is proposed by Neurohr et al., [112], which covers both functional safety (ISO 26262) and safety of the intended functionality (PAS 21448- SOTIF). It identifies and quantifies the hazardous scenarios and complements these standards where necessary.

5.9.1.2. Rail domain

The functional safety in the rail industry is assessed according to EN 50126, EN 50128, EN 50129, EN 50159 and EN 50657 standards [113]. European train operators require safety standards to be followed, which enforce specific safety methodologies and process to be followed by the train manufacturer. Deviations from these pre-defined safety methods and processes are met with significant resistance by assessors which must ensure that safety standards are followed. There are several traditional approaches available in the literature to assess the risk in the railway domain, such as Fault Tree Analysis (FTA) [114], Bayesian analysis [115], Failure Mode and

Effects Analysis (FMEA) [116], event tree analysis (ETA) [117], Hazard and Operability Study (HAZOP) [118] and Analytic Hierarchy Process (AHP) [119]. FTA, as a probabilistic technique which is mostly used in system reliability and safety, provides a single defined adverse effect through identifying combinations of conditions and component failures, while FMEA identifies the range of all single component failures effects on the system. ETA, as a logical evaluative process, involves tracing forward in time or through a causal chain, in contrast to FTA, which is a deductive process [118]. HAZOP is a qualitative technique used as a systematic examination of a planned or existing process. It applies a set of guide words (descriptors) to several parameters.

In the literature, there are several methods developed for risk assessment associated with railway systems. Due to the growing number of passengers and amount of goods on a rail infrastructure network, it is essential to assess the infrastructure-related risks. An overview of the works is proposed on the assessment of risk at the network and object level as well as on the development of intervention programs at both the network and object-level [120]. They concluded that a systematic risk assessment process for railway networks to consider all objects in the network and the effect of object failures (on the level of service of the network) is required. A risk model is designed for estimation and evaluation of the risk in railway infrastructure to identify the most urgent needs for maintenance works [121]. Therefore, it can save money and minimize the associated risks.

In recent years, there has been an increase in utilizing railway transportation, in particular, railway stations. A common issue is overcrowding in stations. Thus, risk management for safety in railway stations is crucial. To deal with uncertainties in risk variables in risk management for overcrowding in railway stations, an intelligent system for managing risks is developed based on the Adaptive Neuro-Fuzzy Inference System (ANFIS) [122]. The construction of a new subway station is another issue that requires a safety risk assessment. A new safety risk assessment model for subway close-attached under crossing construction is proposed in [123]. They obtained the weight and correlation degree of each defined index by using the Analytic Hierarchy Process (AHP) and Fuzzy Matter Element Method (FMEM), respectively, and determined the corresponding risk grade.

One of the most important parts of the rail transportation system is the signal system, as it is directly related to operation safety, operation efficiency, and service quality. An extension of traditional FTA, called Timed Fault Tree analysis (TFTs), is proposed, which integrates temporal events and fault characteristics in the context of signal system failures [118]. It helps to determine when and which faults need to be eliminated to prevent accidents.

Train collisions, train derailments, train fires, level crossings, and other railway (traffic/safety) casualty accidents are common scenarios for hazard analysis and risk assessment in the railway sector. There are several approaches developed to predict, prevent and mitigate accidents in the railway domain. A risk assessment model can be based on the analytic hierarchy process (AHP) and fuzzy comprehensive evaluation, which integrate qualitative and quantitative analysis [124].

The risk factors in the rail transit system are analyzed, and then risks are divided into two stages of construction and operation risks. They sort the decision factors by defining the importance level of the factors in the hierarchy and integrating the experts' grading. A hazard analysis and risk assessment procedure is developed for the railway systems in [125]. It is a quantitative risk assessment model which utilizes various accident reports and information and several workshops with railway safety experts. It proposes a generic model that allows identification of areas of railway operation that need further risk controls, which allows sensitivity analyses to be carried out to determine the risk reduction from the introduction of new control measures. The authors of [125] have used the FTA technique, Event Tree Analysis (ETA) technique and other safety techniques to develop their model. An overview of a framework based on NLP and machine learning is proposed in [126], which provides an insight into railway accident reports. It helps risk analysis experts to study the causal relationship between causes and failures in the context of overall system safety by using text analysis. It aimed to develop a suite of big data (BD) risk assessment as well as development tools for reducing the safety risk. A more recent contribution of artificial intelligence (AI) methods and techniques to risk assessment of railway accidents is proposed by Hadj-Mabrouk [127] to ensure the safety of the design architecture of the transportation system for the users or the environment. One of the main challenges here is determining abnormal scenarios that might lead to a particular potential accident. The authors developed several approaches and tools which enhance modeling and assessment of knowledge about safety concerns.

The adoption of the Internet of Things (IoT) in the railway industry has increased, which leads to an increase in analytical needs for handling data. A dynamic and predictive risk management strategy is proposed by Alawad et al., [128], which utilizes artificial intelligence (AI). The authors have integrated big data analysis (BDA) into the risk assessment process by building upon a Bow Tie (BT) framework model, which aids the development of safer railway stations to ensure the safety of passengers.

Safety evaluation includes the safety files provided by manufacturer and safety studies such as PHA, the functional safety analysis (FSA), the analysis of failure modes, their effects, and of their criticality (AFMEC) or Software Error Effect Analysis (SEEA). In the context of risk assessment of the critical software used in railway transport, an approach is developed by Hadj-Mabrouk, [129] as part of the SEEA analysis. It is based on a hybrid method built around a classification algorithm, a rule-based automatic learning system (RBML), and a system based on knowledge (KBS) which systematizes acquisition tasks and knowledge transfer in the railway safety domain. It used artificial intelligence techniques and involved the development of several approaches and tools to assist in the modeling, storage, and assessment of knowledge about safety. However, these tools are at the mock-up stage at the time of this document.

Operational risk assessment is a challenging task, in particular for high-speed trains, which usually have a complex electromechanical system of thousands of interconnected components. Thus, advanced study and analysis must be carried out to identify the risk factors of the train and get the

risk ranking components in the system, to finally control the associated risks [130]. The existence of qualitative and quantitative information in operational risk assessment of high-speed train introduce uncertainty in the evaluation process, which leads to an impact on the results. Thus, a risk assessment approach is established based on the triangular fuzzy number intuitionistic fuzzy set (TFNIFS) and the dynamic VIKOR method [131]. They showed that in comparison with a static operational risk assessment result, the proposed approach provides better results.

5.9.1.3. Industrial domain

An industrial automation system must meet one or more safety standards such as IEC 61508 (generic) [132], IEC 62061 (machinery) [133] and IEC 61511 (process industry). IEC61511 (2016) depends on the intended application. IEC 61508 (generic) is defined as a functional safety standard for generic electric electronic and programmable electronic systems, which has four safety integrity levels (SILs) and contains a set of safety recommendations and practices for each SIL. It provides quantitative (such as reliability block diagrams and Markov analysis) and qualitative (simulation and formal verification) safety assessment techniques [134].

IEC 61508 does not provide an adequate analysis when both hardware and software aspects of the system are involved. Therefore, a functional safety assessment methodology is proposed by [135] by introducing a network of functional blocks for industrial automation systems. It is a model-based safety framework based on IEC 61499 (generic model for distributed systems), which combines Markov analysis and model checking [136] for quantified risk estimation. However, this safety assessment is restricted to permanent failure and discrete-time models. Following IEC 61508, an improved safety assessment framework of control systems is proposed by Suyama et al., [137], which compares various fault tolerances applied through fault-tolerant controller design. It is difficult to assess the safety function in a controller quantitatively due to the nature of transient responses of a device failure/restoration. A safety assessment framework is proposed to reduce events frequency concerning normal operating range in a control system [138]. This framework can supplement ordinary safety-related systems (SRSs) in risk reduction, according to IEC 61508. However, the proposed safety function cannot handle nonlinear control systems. The reliability of a single element or assembly of elements is often expressed as either its Safety Integrity Level (SIL) or Performance Level (PL). The former is determined on the basis of EN 62061 [139] or IEC 61508 to check whether the control system satisfies the desired safety requirements, while the latter refers to expressing the reliability of safety-related parts of a control system quantitatively. A comparative analysis of safety assessment methods based on PL and SIL concepts for industrial automation systems is proposed by Ciucias et al., [140].

The term industrial control system (ICS) is a generic term applied to a broad class of automation systems having control and monitoring functions such as supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), Programmable Logic Controllers (PLC), etc. In other words, it involves hardware, firmware, communications, and software used to implement monitoring and processes controlling of physical systems and includes SCADA

systems, DCS, PLC, etc. A FMEA based assessment approach is proposed for Safety-Critical SCADA Systems to evaluate, prioritize and correct the systems' failure modes based on pattern recognition [141]. The authors evaluated the rating values of failure modes and recognized action priorities for failure modes and further for applied corrective actions.

5.9.1.4. Telecom domain

To the best of our knowledge, there is no specific (functional) safety standard or guideline for Android devices. Therefore, the IEC 61508, the generic functional safety standard, can be used, which considers the whole lifecycle of electrical, electronic, or programmable electronic systems and products.

5.9.2. Security risk assessment

Similar to the previously presented safety subsection, we review the security risks associated with automotive, rail, industrial and telecom domains here to present the current state of the art in methodology and standards/guidebooks selection.

5.9.2.1. Automotive domain

Security is one of the biggest challenges in the automotive domain. The growing connected and autonomous vehicle (CAV) industry is exposed to a number of emerging cybersecurity threats as a result of an increase in both physical and digital touchpoints [142]. A recent study [143] explored cybersecurity threats in the auto industry by collecting and analyzing primary evidence from cybersecurity experts in this industry. They concluded that the current level of knowledge-sharing is inadequate. A state-of-the-art report on attacks and threats in vehicular communications is carried out [144]. They proposed a three-layer framework consisting of sensing, communication, and control to explain automotive security threats better. While the sensing layer is comprised of vehicle dynamics and environmental sensors, the communication layer is constructed from both in-vehicle and V2X communications. The autonomous vehicular functionality, such as vehicle speed, is controlled by the control layer.

For the functional safety aspect, we have classification based on ISO 26262 ASIL levels while there is no equivalent standard available for the security threats yet [145]. The ISO 26262 standard does not address security issues in automotive systems. A risk assessment framework aligned with ISO 26262 is proposed [146], which provides a security level estimation to formulate high-level security requirements. They analyzed the threat to identify the assets and the corresponding threats and then determined estimation of the threat level and impact level.

A security framework is proposed by Kukkala et al., [147] in an attempt to determine security requirements for tasks and messages in automotive systems. This framework is based on the ISO 26262 standard and applied to the FlexRay protocol. However, they stated that it applies to other

time-triggered protocols. To address the issues related to malicious external intervention in the automotive domain, an integrity level framework is developed which is compatible with classical integrity level architecture. It is built on the combination of automotive safety and security aspects utilizing the structure of Safety Integrity Levels (SIL) [148]. The cybersecurity and safety standardization and ongoing development related to autonomous vehicles as well as new automotive challenges are addressed by Schoitsch et al. [149].

A survey of existing threat and risk analysis methods is provided [150]. They developed a systematic threat analysis and risk assessment framework (SARA) targeting fully AVs by introducing a new threat model. A review of different commonly used threat assessment models is provided, and their characteristics are compared [151]. Besides, they proposed a fuzzy-based risk assessment model to improve the efficiency and accuracy of the assessment process.

In the course of security analysis, threat identification is crucial and requires having a sufficient threat modeling technique [152]. A recent and comprehensive literature review in threat modeling is conducted by Xiong et al., [153], which highlights the state of art in this field. They also found that threat modeling is a diverse field lacking common ground, and the definitions are being used in various ways. A model-based approach based on the identification of single threats for automated risk identification is proposed by Schmittner et al. [154]. One individual approach may not always identify all the system threats and may result in inadequate mitigation mechanisms. Therefore, a hybrid vehicular threat model is developed [155] that utilizes different threat modeling approaches to identify the potential threats and vulnerabilities. In this model, threat agents and security requirements are defined, and different assets (along with their corresponding threats) are classified. A cyberattack analysis method is proposed to analyze cyber threats on automotive vehicles and build a vulnerability-analysis system [156]. This method can identify the characteristics of the attack stages of existing car-hacking techniques and provide essential defense measures which need to be in place. In this study, 13 major hacking cases are analyzed based on the proposed approach.

To address the vehicle systems security needs, the SAE J3061 guidebook [157] proposed the “Threat Analysis and Risk Assessment (TARA)” approach to identify potential cybersecurity threats and assess and rate the risk associated with the threats. It is the state-of-the-art framework for cybersecurity analysis at the date of this document. However, the described security lifecycle is very similar to the safety lifecycle suggested by ISO 26262, which makes it insufficient for AV cases. The SAE J3061 recommends several TARA approaches such as EVITA and HEAVENS. In EVITA, the security threats are classified based on separate views (operational, safety, privacy and financial) to provide a risk level. In contrast, HEAVENS analyzes threats based on Microsoft’s STRIDE (an acronym for Spoofing - Tampering - Repudiation - Information Disclosure - Denial of Service - Escalation of Privilege) model [158]. It uses three elements for risk assessment are Threat Level (TL), Impact Level (IL) and Security Level (SL), where SL is the combination of TL and IL.

A new framework based on TARA is proposed by Bolovinou [159], called “TARA+” which combines the strengths of the SAE and ISO standards for Automated Driving (AD) Systems. A review based on the four most applicable TARA methods (EVITA method, HEAVENS, SAHARA, and BRA) is conducted by Gleirscher [160] for early development phase analysis of the system. They illustrated that the current standards and guidelines such as IEC 62443 (Security for industrial automation and control systems) and SAE J3061 are either incomplete or not directly applicable in practice.

Security issues do not necessarily generate new safety hazards in automotive systems’ design; however, they change the likelihoods of the existing hazards. Therefore, there is a need to address both functional safety and security considerations during development as recognized by the recent update of the ISO 26262 [161] and a reason for the ongoing SAE and ISO joint project. Some works are based on the integration of both safety and security aspects. Utilizing a Six-Step Model [162], an approach proposed by Sabaliauskaite et al., [163], which integrates AV safety, and security processes. It is compliant with the international standards SAE J3016, SAE J3061, and ISO 26262 and considers driving automation levels. Currently, Security-aware Hazard Analysis and Risk Assessment (SAHARA) and FMVEA (Failure Mode Vulnerability and Effect Analysis) are state of the art for co-analysis safety and security in this domain. In order to consider the security knowledge within the safety life cycle, a systematic pattern-based and ISO 26262-oriented approach is proposed [164], which can guide non-expert engineers toward the implementation of the best practices. To show the application of the approach, an automotive case having different scenarios is used. In another study [165], a process for co-engineering safety and security is proposed by systematizing commonalities as well as variabilities that exist in the requirements of ISO 26262 for functional safety and requirements of SAE J3061 in cybersecurity. A comprehensive survey on the safety and cybersecurity co-engineering of cyber-physical systems is conducted [166], which explores current open issues and research challenges. They defined a multi-attribute taxonomy of cybersecurity and safety to analyze different approaches.

Recently in the context of connected vehicles, the British Standards Institute (BSI) has published a recommendation for cyber security for connected and automated vehicles, PAS 1885:2018 (note that A “PAS” is a Publicly Available Specification, this is a pre-standardization document). It is intended to form the future definitive standard in Automotive Cyber-Security. Also, there is a joint standardization project by ISO and SAE on cybersecurity for all the systems and devices throughout the entire automotive domain named ISO/SAE 21434 started in 2016. It is currently in development and attempts to specify the Cybersecurity Assurance Level (CAL), which indicates the required level of cybersecurity process rigor [167]. However, it is based on the released SAE J3061 that introduced the method TARA.

5.9.2.2. Rail domain

There is a study with an emphasis on the physical security measures that affect the personal security of passengers based on their perception [168]. In this work, they proposed three

recommendations as installing X-ray equipment as detective assistance for security management, strictly checking passengers when entering the area, and paying more attention to the lighting system. Control functions on interlocking devices (e.g., light signals, railway switches, etc.) are performed by lineside shelter protection systems, which are small buildings located on railway track sides. They might cause critical security issues; thus, appropriate protections must be employed. In [169], they have addressed the limits of model-driven approaches when modeling physical security aspects in cyber-physical systems and, more specifically, in railway applications. They highlighted some issues in separated cyber and physical security modeling and proposed a possible solution by using two modeling approaches based on Unified Modeling Language (UML) (CIP_VAM and SecAM).

As the railway is becoming more intelligent and connected systems, it introduces new attackers and cyber-criminals. Threat modeling plays a crucial role in identifying new potential threats to the system, which makes it a preliminary step in the process of risk assessment. In other words, to identify what threats might exist. In the literature, several approaches have been published to assess the risks associated with these threats. A risk assessment approach is developed by Bloomfield et al., [170] for the European Railway Traffic Management System (ERTMS) from a cybersecurity perspective. Incorporation of security analysis in a safety context is conducted by Winther et al. [171]. They proposed an adaption of hazard and operability (HAZOP) studies which take into account security threats in a train leader telephone system to evaluate the system against potential threats through formulating a set of generic expressions.

One of the standards that have been utilized for security assessment in the railway domain is the IEC 62443 series “Security for industrial automation and control systems” [172] which defines five security assurance levels SL 0 (lowest) to SL 4 (highest) and uses three types of security levels (SL) as SL-T (Desired level of security), SL-C (Level of security achievable) and SL-A (Achieved level of security). This approach avoids the uncertainty or infeasibility of credible likelihood estimation and concentrates on the attacker’s capability as stipulated by the security levels (SL) definition. However, this approach focuses too much on the attacker capability and does not exploit all attack scenarios or security aspects into account. To solve this issue, an approach proposed by Braband [173] focused on railway automation applications. By introducing some adoptions (such as SL allocation), they showed that this approach matches well with the planned ISA/IEC 62443 series of standards.

In the context of cyber-attacks and security threats, the majority of threats target integrity and availability of the External Door control (EDC) system, which can lead to severe consequences. A security risk assessment of the EDC system of the Train Control and Monitoring System (TCMS) is conducted in [174]. They analyzed the characteristics of the railway threat landscape and investigated the impacts of the identified potential threats and their consequences on the whole system and evaluated associated risks. In the end, a set of countermeasures was proposed to strengthen the security of TCMS against identified potential threats. Another source of security risk is the terrorist attacks. A risk assessment framework proposed by Kaewunruen et al. [175]

based on probabilistic risk assessment to alleviate the risk of terrorist attacks on transport infrastructure. A probabilistic risk analysis approach has been adapted to analyze the security systems, which claimed that the framework implementation would guarantee a certain level of security via constantly improving the modeling techniques in parallel with the increasing complexity of the systems.

A Probabilistic risk assessment is proposed to analyze safety and security in a Communication Based Train Control (CBTC) [176]. It provides a threat modeling approach and uses security events as additional nodes in the fault tree. However, weighing identified events by the probability of their occurrence make it a challenging task. A cybersecurity risk assessment based on IEC 62243 is proposed by Schmittner et al. [177]. They developed a systematic threat modeling approach for identifying threats in the safety-critical railway domain, which is currently state of the art in this domain.

5.9.2.3. Industrial domain

A SCADA system tends to be prone to attacks of various forms, including physical attacks by a human. It can harm the system or use the system's resources. A review of several documented standards in SCADA from security aspects is provided [178], and solutions such as Password protection, Smart cards, Encryption are proposed. In order to reduce risk of failure and exposure of cyber-attacks on ICS networks and SCADA systems, IEC 62443 is the primary standard choice which is a multi-industry cybersecurity standard. A technical report is proposed in [179], which addresses the inter-networked building automation and control systems (BAS or BCS) using the BACnet protocol. It provides threat analysis and possible countermeasures that deal with threats from physical threats (such as physical access threat by disgruntled employees to perform unauthorized actions) to the building automation equipment and attached computers.

Modern ICSs are moving towards more integration with Internet-of-Things and cloud computing technologies [180] which introduces a broad range of security vulnerabilities [181]. There are several risk assessment frameworks and techniques that exist in academia and industry applicable to the ICS through workshops and standards and guidance materials [182]. A quantitative risk security assessment method for ICSs is proposed on the basis of the Bayesian network [183]. It utilizes an online data-driven parameter learning strategy to improve the accuracy of the real-time dynamic assessment result. They showed that it could improve the accuracy and fit the dynamic changes of the system. In another study, an approach based on the fuzzy analytic hierarchy process is conducted to assess the information security risks associated with ICS [184]. In order to enhance the lack of fuzziness in the evaluation result of the traditional analytic hierarchy process, they introduced a fuzzy consistent matrix and entropy method.

A joint risk analysis approach covering both safety and security aspects for ICSs is conducted [185]. It uses the Bow Tie method to provide the level of risks through a qualitative likelihood analysis methodology. Since it was not able to handle input data uncertainty, a fuzzy semi-

quantitative approach is proposed [186] to reduce risk underestimation. In Critical National Infrastructure (CNI) plants, the cyber-attacks through the ICS can be regarded as a dangerous external action that requires great attention. In [187], a cybersecurity risk evaluation approach is proposed for using SCADA systems in the ICS of CNI plants. This approach considers not only information security methods but all safety, security and reliability measures. It also estimates damage values from feasible cyber-attacks.

In the design of security of information systems, FMEA is an approach used extensively as it helps to prioritize all possible vulnerable areas (failure modes) of the system. However, it has some inherent problems and lacks application of FMEA for the security aspect of SCADAs [179]. A study on the Information Security Risk Management of SCADA Systems is carried out by Lin [188]. They proposed an evaluation model which addresses those inherent problems and is suitable for semi-quantitative analysis of a secure SCADA's failure modes.

A recent comprehensive review of security assessment methodologies in industrial control systems conducted by Qassim et al. [189]. They reviewed several ICS security assessment methodologies and investigated existing methodologies to see how they meet the security need of electrical power control systems, and realized that among all security assessment methodologies, vulnerability identification and prioritization techniques could fulfill all North American Electric Reliability Corporation-Critical Infrastructure Protection (NERC-CIP) security requirements. In addition, they showed that the security assessment techniques in ICS and IT systems are very similar as they both are on the basis of conducting vulnerability analysis and risk management techniques.

5.9.2.4. Telecom domain

Android is an extensively used operating system for smartphones and mobile devices, which makes it an interesting option for attackers who are looking to get access to sensitive data. The number and sophistication of mobile malware (such as Viruses, worms, Trojans, and bots) attacks on Android-based devices are fast growing as a result of malicious software variants. Thus, assessing the risk associated with malicious applications is a typical way to protect device users.

A survey report based on Android security threats and existing enforcements is conducted by Rashidi et al. [190]. They attempted to classify the works conducted between 2010-2015 and performed strengths and weaknesses analysis of the suggested solutions. A risk assessment approach compatible with established guidelines on risk assessment [191] for smartphones is proposed in ISO/IEC27005 [192]. They have divided the device into sub-assets, which assess the threats specific to smartphones and consider the characteristics of a smartphone security model. A framework was proposed [193] for quantitative security risk assessment of Android devices. They improved the efficiency of the Android permission system by informing the user regarding Android permissions and apps risks. However, the impact levels of Android permissions were not specified in this study. A risk assessment framework for Android resource usage, called XDroid,

is proposed, which aimed at providing a quantitative estimation on how likely resource access from an app would harm the users [194]. It utilizes a hidden Markov model and trains the model with malicious app dataset and tests with different test datasets using the Viterbi algorithm.

The commonly used (information) security standards are ISO/IEC 27000 standards (such as 27001, 27032, etc.) and the National Institute of Standards and Technologies Special Publication (NIST) 800 Series. An approach proposed by Lederm et al. [195] enables mobile device users to assess the level of risk associated with the handset behavior, no matter what skill level and knowledge they have. They have used several methods to minimize the effort required from the end-user and considered different security requirements of various services. An automated security assessment framework for mobile applications, data communication and the back-end server proposed by Zheng et al. [196] which was based on static and dynamic analysis. This analysis was conducted on local mobile applications, data communication, and server-side, respectively.

A recent survey has been conducted on Android malicious apps [197]. They explored the various techniques used for identifying Android malicious apps and provided a comparative study of various approaches. A risk assessment approach was proposed by Abdullah et al. [198] to evaluate the risk level of Android applications. It involves considering confidentiality (privacy), integrity (financial) and availability (system) factors and is able to detect Android botnet. Using a mathematical analysis of an app, it returns a risk level (i.e., Very Low, Low, Moderate, High, and Very High) which is easy to understand. Some approaches utilize machine learning techniques to assess the security threat to mobile devices. A continuous and automated risk assessment framework, called RiskMon, proposed by Jing et al., [199] which utilizes machine-learned ranking (MLR) technique to assess risks associated with mobile applications by providing a risk assessment baseline assigning a risk score for every attempt to access sensitive information. A risk assessment model for android devices based on applications permissions is proposed by Padrithi [200], which provides a risk score on a scale of 0-100 based on probability estimates of classifiers for each app. Installation of a malicious application is a typical security concern for smartphones. The evaluation of threats corresponding to the applications installed on the Android OS can be performed in conjunction with machine learning techniques as proposed by Park et al. [201]. The introduced situational awareness model consists of three levels: Malware Awareness, Threat Awareness and Decision-Making Awareness.

Ignoring security configurations by Android users might lead to security and privacy threats. An approach for characterizing these risks is proposed by Vecchiato et al. [202]. Under a successful attack situation, they analyzed a set of 41 security configuration recommendations concerning the likelihood of being exploited and the associated impact. The security posture of the Android smartphone is determined by considering both security configuration level and sensitive data risk assessment [203], which can enhance the security level of the device about sensitive data leakage. A method for assessing the privacy risk of Android users is proposed by Mylonas et al. [204]. They used a risk assessment process called Privacy Impact Assessment (PIA) which is

based on potential privacy risk and is associated with the collection, use or disclosure of personal information. They showed their method through a case study having two hypothetical users and actual app data. A dynamic privacy-scoring model proposed by Hamed et al. [205] for Android applications aims at assessing the risk to the users' privacy associated with those applications that require a set of permissions.

While there are many works focusing only on the security behavior of mobile apps [206, 207, 208], some researchers considered both the security and privacy behavior of mobile apps [209]. A smartphone framework called Enterprise Smartphone Apps Risk Assessment (ESARA) is proposed to analyze the potential privacy and security risks of using smartphone apps [209]. This risk assessment approach is based on the implementation of natural language processing (NLP) and machine learning techniques, which helps enterprises to protect their data against adversaries and unauthorized accesses. However, this approach lacks implementation in a real company environment.

5.9.3. Summary of safety and security risk assessment

The suggested standard(s) and guidebook for safety and (cyber) security risk assessment of automotive, rail, industrial and telecom domains are proposed along with the corresponding score approach, summarized in the following table.

Summary of standards and scoring approach for safety and security risk assessment.

Domains		(Functional) Safety	(Cyber) Security
Automotive	Standard(s)	ISO 26262 , ISO PAS 21448	SAE J3061 (Guidebook)
	Scoring approach	Automotive Safety Integrity Level ASIL A (lowest) to ASIL D (highest)	For HEAVENS approach: Security Level (SL) matrix (combination of threat level and impact level)
Railway	Standard(s)	EN 50126 – RAMS EN 50128 – Software safety EN 50657 – Software safety (from 2017) EN 50129 –Electronic Systems EN 50159 –Safety communication	IEC 62443
	Scoring approach	Safety Integrity Level (SIL) SIL 0 (lowest) to SIL 4 (highest)	Security Level (SL) SL 0 (lowest) to SL 4 (highest)

Industrial	Standard(s)	IEC 61508	IEC 62443
	Scoring approach	Safety integrity levels (SILs) SIL 1 (lowest) to SIL 4 (highest)	Security Level (SL) SL 0 (lowest) to SL 4 (highest)
Telecom	Standard(s)	IEC 61508	ISO/IEC 27000 and NIST 800 Series
	Scoring approach	Safety integrity levels (SILs) SIL 1 (lowest) to SIL 4 (highest)	No specific approach (Risk level: Low, Medium, High)

6. Conclusions

The main objective of WP2 in the XIVT project [210] has been to develop tailored test optimization solutions to efficiently address the peculiarities of variant-intensive systems. In order to determine whether, for example, a change or bug fix in one product instance requires re-testing of other instances and variants (instantiated from the same baseline) or not, different sources of information need to be carefully analyzed. To answer such optimization questions, appropriate knowledge processing and learning techniques, particularly based on Natural Language Processing, to deal with numerous and versatile sources of variability have been developed in the scope of this WP. Moreover, XIVT WP2 also offers automated solutions for performing similarity analysis on requirements across a series of product versions to identify common features, and offer recommendations for reuse contributing to the overall optimization goal.

In this deliverable, we reported on the highlights of XIVT achievements and solutions for test optimization of variant-intensive software systems. While the main focus has been on the industrial domains of the project, the developed solutions can easily be applied and extended for application in other domains and types of systems as well. Moreover, in this deliverable we also provided a brief state-of-the-art analysis where gaps with respect to the optimization of testing process of variant-intensive systems were identified and reported

References

1. Apel, S., Batory, D., Kästner, C., & Saake, G. *Feature-Oriented Software Product Lines*. Springer, Berlin, 2013.
2. Abbas, M., Saadatmand, M., Enoiu, E., Sundamark, D., & Lindskog, C. (2020, December). Automated reuse recommendation of product line assets based on natural language requirements. In *International Conference on Software and Software Reuse* (pp. 173-189). Springer, Cham.
3. Abbas, M., Ferrari, A., Shatnawi, A., Enoiu, E. P., & Saadatmand, M. (2021). Is Requirements Similarity a Good Proxy for Software Similarity? An Empirical Investigation in Industry. In *REFSQ* (pp. 3-18).
4. Abbas, M., Ferrari, A., Shatnawi, A., Enoiu, E. P., Saadatmand, M., & Sundmark, D. "On the relationship between similar requirements and similar software." In *Requirements Engineering* (2022): 1-25.
5. Vasco Leitão, Ibéria Medeiros. *SRXCRM: Discovering Association Rules Between System Requirements and Product Specifications*. In *Proceedings of the International Workshop on Natural Language Processing for Requirements Engineering (NLP4RE)*, Essen, Germany, April 2021.
6. Vasco Leitão. *Prioritization of Software and System Requirements through Natural Language Processing for Testing Software*, Master Thesis of Master in Data Sciences, Department of Informatics, Faculty of Sciences of University of Lisbon, November 2021.
7. Escalona, M. J. et al. "An overview on test generation from functional requirements," *Journal of Systems and Software*, vol. 84, no. 8, pp. 1379–1393, 2011.
8. Ahsan, I. et al. "A comprehensive investigation of natural language processing techniques and tools to generate automated test cases," in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC '17*. New York: ACM Press, 2017, pp. 1–10.
9. Garousi, V. et al. "NLP-assisted software testing: A systematic mapping of the literature," *Information and Software Technology*, vol. 126, 2020.
10. Gröpler, R., Sudhi, V., Calleja García, E.J., Bergmann, A. (2021). NLP-Based Requirements Formalization for Automatic Test Case Generation. In: *29th International Workshop on Concurrency, Specification and Programming (CS&P'21)*, Vol. 2951, pp. 18-30.
11. Gröpler, R., Kutty, L., Sudhi, V., & Smalley, D. (2022). Automated Requirement Formalization using Product Design Specifications. In: *5th Workshop on Natural Language Processing for Requirements Engineering (NLP4RE'22)*.
12. Parkkila, Christoffer. "Clone detection in model-based design: An evaluation in the safety-critical railway domain." Master Thesis Report. MDH (2021).
13. Karp Richard M.: *Reducibility Among Combinatorial Problems*; University of California at Berkeley 1972; <https://people.eecs.berkeley.edu/~luca/cs172/karp.pdf>
14. Young N.: *Greedy Set-Cover Algorithms*. In: Kao MY. (eds) *Encyclopedia of Algorithms*. Springer, Boston, MA. 2008; https://doi.org/10.1007/978-0-387-30162-4_175
15. Takeda, T., Takahashi, M., Yumoto, T., Masuda, S., Matsuodani, T., & Tsuda, K. (2019). Applying change impact analysis test to migration test case extraction based on IDAU and graph analysis techniques. In: *ICSTW 2019*, pp. 131-139.
16. Benlarabi, A. (2014). Towards a co-evolution model for software product lines based on cladistics. In: *RCIS 2014*, pp. 1-6.
17. Brink, C., Heisig, P., and Wackermann, F. (2016). Change impact in product lines: a systematic mapping study. In: *ICIST 2016*, pp. 677-694.
18. Käßmeyer, M., Schulze, M., & Schurius, M. (2015). A process to support a systematic change impact analysis of variability and safety in automotive functions. In: *SPLC '15*, pp. 235-244.
19. Goknil, A., Kurtev, I., van den Berg, K., & Spijkerman, W. (2014). Change impact analysis for requirements: A metamodeling approach. *Information and Software Technology*, 56(8), 950-972.
20. Arora, C., Sabetzadeh, M., Goknil, A., Briand, L. C., & Zimmer, F. (2015). Change impact analysis for natural language requirements: An NLP approach. In: *RE'15*, 6-15.
21. Alkaf, H., Hassine, J., Binalialhag, T., & Amyot, D. (2019). An automated change impact analysis approach for user requirements notation models. *Journal of Systems and Software*, 157, 110397.
22. Díaz, J., Pérez, J., Garbajosa, J., & Wolf, A. L. (2011). Change impact analysis in product-line architectures. In: *ECSA 2011*, 114-129.
23. Gethers, M., Dit, B., Kagdi, H., & Poshyvanyk, D. (2012). Integrated impact analysis for managing software changes. In: *ICSE 2012*, 430-440.
24. Hajri, I., Goknil, A., Briand, L. C., and Stephany, T. (2018). Change impact analysis for evolving configuration decisions in product line use case models. *Journal of Systems and Software*, 139, 211-237.
25. Angerer, F., Grimmer, A., Prähofer, H., & Grünbacher, P. (2015). Configuration-aware change impact analysis (t). In: *ASE 2015*, 385-395.

26. Lima, C., do Carmo Machado, I., de Almeida, E. S., & von Flach G. Chavez, C. (2018). Recovering the product line architecture of the Apo-Games. In: SPLC '18, 289-293.
27. Lima, C., Assunção, W. K., Martinez, J., Mendonça, W., Machado, I. C., & Chavez, C. F. (2019). Product line architecture recovery with outlier filtering in software families: the Apo-Games case study. *Journal of the Brazilian Computer Society*, 25(1), 1-17.
28. Eyal Salman, H., Seriai, A. D., & Dony, C. (2015). Feature-level change impact analysis using formal concept analysis. *International Journal of Software Engineering and Knowledge Engineering* 25, 69-92.
29. Angerer, F., Prähofer, H., & Grünbacher, P. (2016). Modular change impact analysis for configurable software. In: *ICSME 2016*, 468-472.
30. Lity, S., Kowal, M., & Schaefer, I. (2016). Higher-order delta modeling for software product line evolution. In: *FOSD 2016*, 39-48.
31. Lity, S., Morbach, T., Thüm, T., & Schaefer, I. (2016). Applying incremental model slicing to product-line regression testing. In: *ICSR 2016*, 3-19.
32. Maâzoun, J., Bouassida, N., and Ben-Abdallah, H. (2016). Change impact analysis for software product lines. *Journal of King Saud University - Computer and Information Sciences*, 28(4), 364-380.
33. Dintzner, N., Kulesza, U., van Deursen, A., & Pinzger, M. (2015). Evaluating feature change impact on multi-product line configurations using partial information. In: *ICSR 2015*, pp. 1-16.
34. Dintzner, N., van Deursen, A., & Pinzger, M. (2017). Analysing the Linux kernel feature model changes using FMDiff. *Software & Systems Modeling*, 16(1), 55-76.
35. Kahraman, G., Cleophas, L., & Schiffelers, R. (2021). Modeling Relationships Between Feature Model Views. In: *MODELS-C 2021*, 437-446.
36. O'Brien L., Stoermer C., & Verhoef C. (2002). *Software Architecture Reconstruction: Practice Needs and Current Approaches*. Technical Report CMU/SEI-2002-TR-024, Software Engineering Institute.
37. van Deursen A., Hofmeister C., Koschke R., Moonen L., & Riva C. (2004). Symphony: view-driven software architecture reconstruction. In: *WICSA '04*, 122-132.
38. Medvidovic, N., Egyed, A., & Gruenbacher, P. (2003). Stemming architectural erosion by coupling architectural discovery and recovery. In: *STRAW'03*, 61-68.
39. Tran J.B., & Holt R.C. (1999). Forward and reverse repair of software architecture. In: *CASCON'99*, 12-20.
40. Salas Cordero, S., Vingerhoeds, R., Zolghadri, M., & Baron, C. (2020). Addressing Obsolescence from day one in the conceptual phase of complex systems as a design constraint. In: *IFIP*, 369-383.
41. Souza, S.M.O.D.A. (2020). *Processo de gestão de riscos integrado a qualificação tecnológica em projetos de inovação*. Dissertation, Centro Universitário SENAI CIMATEC.
42. Walker, R. J., Holmes, R., Hedgeland, I., Kapur, P., & Smith, A. (2006). A lightweight approach to technical risk estimation via probabilistic impact analysis. In: *MSR '06*, 98-104.
43. Garg, T., Eppinger, S., Joglekar, N., & Olechowski, A. (2017). Using TRLs and system architecture to estimate technology integration risk. In: *ICED 17*, 301-310.
44. Pett, T., Eichhorn, D., & Schaefer, I. (2020). Risk-based compatibility analysis in automotive systems engineering. In: *MODELS '20*, pp. 1-10.
45. J. Harrold, R. Gupta, M.L. Soffa. "A methodology for controlling the size of a test suite." in: *Proceedings. Conference on Software Maintenance*, Nov 1990, pp. 302-310.
46. Matinnejad, Reza, Shiva Nejati, Lionel Briand, and Thomas Bruckmann. "Test generation and test prioritization for simulink models with dynamic behavior." in *IEEE transactions on software engineering*, vol. 45, no. 9, pp. 919-944, 1 sept. 2019.
47. Annibale, Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. "Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets." (2018): 122-158.
48. Galeotti, Juan Pablo, Gordon Fraser, and Andrea Arcuri. "Improving search-based test suite generation with dynamic symbolic execution." In *2013 IEEE 24th international symposium on software reliability engineering (issre)*, pp. 360-369. IEEE, 2013.
49. Manikumar, T., A. John Sanjeev Kumar, and R. Maruthamuthu. "Automated test data generation for branch testing using incremental genetic algorithm." *Sādhanā* 41, no. 9 (2016): 959-976.
50. Zhang, Yan, and Dunwei Gong. "Generating test data for both paths coverage and faults detection using genetic algorithms: multi-path case." *Frontiers of Computer Science* 8, no. 5 (2014): 726-740.
51. Amalfitano, Domenico, Nicola Amatucci, Anna Rita Fasolino, and Porfirio Tramontana. "AGRippin: a novel search based testing technique for Android applications." In *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, pp. 5-12. ACM, 2015.
52. Delgado-Pérez, Pedro, and Inmaculada Medina-Bulo. "Search-based mutant selection for efficient test suite improvement: Evaluation and results." *Information and Software Technology* 104 (2018): 130-143.

53. Esfandyari, Sajad, and Vahid Rafe. "A tuned version of genetic algorithm for efficient test suite generation in interactive t-way testing strategy." *Information and Software Technology* 94 (2018): 165-185.
54. Jiang, Bo, Wing Kwong Chan, and T. H. Tse. "PORA: proportion-oriented randomized algorithm for test case prioritization." In *2015 IEEE International Conference on Software Quality, Reliability and Security*, pp. 131-140. IEEE, 2015.
55. Pradhan, Dipesh, Shuai Wang, Shaikat Ali, Tao Yue, and Marius Liaaen. "Stipi: Using search to prioritize test cases based on multi-objectives derived from industrial practice." In *IFIP International Conference on Testing Software and Systems*, pp. 172-190. Springer, Cham, 2016.
56. Lachmann, Remo, Sascha Lity, Sabrina Lischke, Simon Beddig, Sandro Schulze, and Ina Schaefer. "Delta-oriented test case prioritization for integration testing of software product lines." In *Proceedings of the 19th International Conference on Software Product Line*, pp. 81-90. ACM, 2015.
57. Lopez-Herrejon, Roberto Erick, Javier Javier Ferrer, Francisco Chicano, Evelyn Nicole Haslinger, Alexander Egyed, and Enrique Alba. "A parallel evolutionary algorithm for prioritized pairwise testing of software product lines." In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 1255-1262. ACM, 2014.
58. Arrieta, Aitor, Shuai Wang, Goiuria Sagardui, and Leire Etxeberria. "Search-Based test case prioritization for simulation-Based testing of cyber-Physical system product lines." *Journal of Systems and Software* 149 (2019): 1-34.
59. Zamli, Kamal Z., Norasyikin Safieny, and Fakhruddin. "Hybrid Test Redundancy Reduction Strategy Based on Global Neighborhood Algorithm and Simulated Annealing." In *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, pp. 87-91. ACM, 2018.
60. Shi, August, Alex Gyori, Milos Gligoric, Andrey Zaytsev, and Darko Marinov. "Balancing trade-offs in test-suite reduction." In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 246-256. ACM, 2014.
61. Saberi, A. Khabbaz, F. Benders, R. Koch, J. J. Lukkien, and M. G. J. van den Brand. "A method for quantitative measurement of safety culture based on ISO 26262." In *26th Safety-Critical Systems Symposium (SSS'18)*, pp. 203-218. CreateSpace Independent Publishing Platform, 2018.
62. de Souza, L. S., Prudêncio, R. B., & Barros, F. D. A. (2014, October). A hybrid binary multi-objective particle swarm optimization with local search for test case selection. In *2014 Brazilian Conference on Intelligent Systems* (pp. 414-419). IEEE.
63. Choudhary, Ankur, Arun Prakash Agrawal, and Arvinder Kaur. "An effective approach for regression test case selection using pareto based multi-objective harmony search." In *Proceedings of the 11th International Workshop on Search-Based Software Testing*, pp. 13-20. ACM, 2018.
64. de Souza, Luciano Soares, Ricardo Bastos Cavalcante Prudêncio, and Flávia A. de Barros. "A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection." *Journal of the Brazilian Computer Society* 21, no. 1 (2015): 19.
65. Beatriz Pérez Lamancha, Macario Polo, and Mario Piattini: *Systematic Review on Software Product Line Testing*, ICSoft 2010: Software and Data Technologies pp 58-71, Springer
66. Jiyeon Lee, Sungwon Kang, and Danhyung Lee: A survey on software product line testing. SPLC 2012, ACM Digital Library.
67. Thüm, T., Apel, S., Kästner, C., Schaefer, I., & Saake, G.: A classification and survey of analysis strategies for software product lines. *ACM Computing Surveys (CSUR)*, 47(1), 6. (2014).
68. Utting, M., and B. Legeard, "Practical Model-Based Testing: A Tool Approach", Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2006.
69. Utting, M., Pretschner, A., Legeard, B. A Taxonomy of Model-Based Testing Approaches. *Software Testing, Verification and Reliability*, 2012.
70. A. Pretschner et al., One Evaluation of Model-Based Testing and its Automation. In *Proceedings of the 27th International Conference on Software Engineering*, St. Louis, Missouri, USA, 2005, pp. 392-401.
71. Briand, L., Nejati, S., Sabetzadeh, M. and Bianculli, D., 2016, May. Testing the untestable: model testing of complex software-intensive systems. In *Proceedings of the 38th international conference on software engineering companion* (pp. 789-792). ACM.
72. Arrieta, A., Wang, S., Markiegi, U., Sagardui, G. and Etxeberria, L., 2017. Search-based test case generation for cyber-physical systems. In *IEEE Congress on Evolutionary Computation (CEC)*, IEEE.
73. Orso, A. and Rothermel, G., 2014, May. Software testing: a research travelogue (2000–2014). In *Proceedings of the on Future of Software Engineering* (pp. 117-132). ACM.
74. QATest, QATestingTools.com: Model Based Testing, <http://www.qatestingtools.com/testing-tool-article/model-based-testing>
75. M. Veanes, C. Campbell, W. Schulte and N. Tillmann, "Online Testing with Model Programs," in *ESEC/FSE-13*, 2005.
76. Grindal Mats, Lindstrom Birgitta, Offutt Jeff, and Sten F Andler. An evaluation of combination strategies for test case selection. *Empirical Software Engineering*, 11(4):583–611, 2006.

77. Ammann, Paul and Offutt, Jeff. Using formal methods to derive test frames in category-partition testing. In Computer Assurance, 1994. COMPASS'94 Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security. Proceedings of the Ninth Annual Conference on, pages 69–79. IEEE, 1994.
78. Bergström, Henning, and Eduard Paul Enoiu. "Using timed base-choice coverage criterion for testing industrial control software." In 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 216-219. IEEE, 2017.
79. Weilkens, Tim. Variant modeling with SysML. Lulu. com, 2012.
80. Pohl, Klaus, et al. "Advanced model-based engineering of embedded systems." Advanced Model-Based Engineering of Embedded Systems. Springer, Cham, 2016.
81. Felderer, M., Zech, P., Breu, R., Büchler, M., and Pretschner, A. (2016) Model-based security testing: a taxonomy and systematic classification. Software: Testing, Verification and Reliability, vol26: 119– 148.
82. Ahmed M A, Hermadi I. GA-based multiple paths test data generator. Computer & Operations Research, 2008, 35(10): 3107–3124
83. Zhou Z Q, Huang D H, Tse T H, Yang Z Y, Huang H, Chen T Y. Metamorphic testing and its applications. In: Proceedings of the 8th International Symposium on Future Software Technology. 2004.
84. A. Rebert, S. K. Cha, T. Avgerinos, J. Foote, D. Warren, G. Grieco, and D. Brumley. Optimizing Seed Selection for Fuzzing. In In Proceedings of the USENIX Security Symposium, pages 861–875, August 2014.
85. Luo, Yaping, Arash Khabbaz Saberi, and Mark van den Brand. "Safety-Driven Development and ISO 26262." In Automotive Systems and Software Engineering, pp. 225-254. Springer, Cham, 2019.
86. Xie, Guoqi, Gang Zeng, Yan Liu, Jia Zhou, Renfa Li, and Keqin Li. "Fast functional safety verification for distributed automotive applications during early design phase." IEEE Transactions on Industrial Electronics 65, no. 5 (2017): 4378-4391
87. EN50126. "Railway Applications–The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)." The European Standard (1999).
88. EN50128. "Railway Applications-Communication, signalling and processing systems-Software for railway control and protection systems." Comité Européen de Normalisation Electrotechnique (2001).
89. EN50129, "Railway application–Safety related electronic systems for signaling." (2000).
90. ISO26262. "Road vehicles – Functional safety. ", 2011.
91. Dajsuren, Yanja, and Mark van den Brand. "Automotive Software Engineering: Past, Present, and Future." In Automotive Systems and Software Engineering, pp. 3-8. Springer, Cham, 2019.
92. Dajsuren, Yanja, and Mark van den Brand, eds. Automotive Systems and Software Engineering: State of the Art and Future Trends. Springer, 2019.
93. Gosavi, Mukul Anil, Benjamin B. Rhoades, and James M. Conrad. "Application of Functional Safety in Autonomous Vehicles Using ISO 26262 Standard: A Survey." In SoutheastCon 2018, pp. 1-6. IEEE, 2018.
94. Li, Wenjun, and Hongkun Zhang. "A software hazard analysis method for automotive control system." In 2011 IEEE International Conference on Computer Science and Automation Engineering, vol. 3, pp. 744-748. IEEE, 2011.
95. Bello, Lucia Lo, Riccardo Mariani, Saad Mubeen, and Sergio Saponara. "Recent advances and trends in on-board embedded and networked automotive systems." IEEE Transactions on Industrial Informatics 15, no. 2 (2018): 1038-1051.
96. Gharib, Mohamad, Paolo Lollini, Andrea Ceccarelli, and Andrea Bondavalli. "Dealing with Functional Safety Requirements for Automotive Systems: A Cyber-Physical-Social Approach." In International Conference on Critical Information Infrastructures Security, pp. 194-206. Springer, Cham, 2017.
97. Xie , Guoqi, et al. "Recent Advances and Future Trends for Automotive Functional Safety Design Methodologies." IEEE Transactions on Industrial Informatics 16.9 (2020): 5629-5642.
98. Henriksson, Jens, Markus Borg, and Cristofer Englund. "Automotive safety and machine learning: Initial results from a study on how to adapt the ISO 26262 safety standard." In 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS), pp. 47-49. IEEE, 2018.
99. Schwalbe, Gesina, and Martin Schels. "A Survey on Methods for the Safety Assurance of Machine Learning Based Systems." 10th European Congress on Embedded Real Time Software and Systems (ERTS 2020). 2020.
100. Kaindl, Hermann, and Stefan Kramer. "Towards Probability-based Safety Verification of Systems with Components from Machine Learning." arXiv preprint arXiv:2003.01155 (2020).
101. Amberkar, Sanket, Barbara J. Czerny, Joseph G. D'Ambrosio, Jon D. Demerly, and Brian T. Murray. "A comprehensive hazard analysis technique for safety-critical automotive systems." SAE transactions (2001): 282-292.
102. Sulaman, Sardar Muhammad, Armin Beer, Michael Felderer, and Martin Höst. "Comparison of the FMEA and STPA safety analysis methods—a case study." Software Quality Journal 27, no. 1 (2019): 349-387.
103. Soltanali, Hamzeh, et al. "Development of a risk-based maintenance decision making approach for automotive production line." International Journal of System Assurance Engineering and Management 11.1 (2020): 236-251.
104. Qin, Jindong, Yan Xi, and Witold Pedrycz. "Failure mode and effects analysis (FMEA) for risk assessment based on interval type-2 fuzzy evidential reasoning method." Applied Soft Computing 89 (2020): 106134.

105. Adedjouma, Morayo, Gabriel Pedroza, and Boutheina Bannour. "Representative Safety Assessment of Autonomous Vehicle for Public Transportation." In 2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC), pp. 124-129. IEEE, 2018.
106. Stolte, Torben, Gerrit Bagschik, Andreas Reschka, and Markus Maurer. "Hazard analysis and risk assessment for an automated unmanned protective vehicle." In 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 1848-1855. IEEE, 2017.
107. Khastgir, Siddhartha, et al. "Towards increased reliability by objectification of Hazard Analysis and Risk Assessment (HARA) of automated automotive systems." *Safety Science* 99 (2017): 166-177.
108. Sini, Jacopo, and Massimo Violante. "A simulation-based methodology for aiding advanced driver assistance systems hazard analysis and risk assessment." *Microelectronics Reliability* 109 (2020): 113661.
109. Abdulkhaleq, Asim, Daniel Lammering, Stefan Wagner, Jürgen Röder, Norbert Balbierer, Ludwig Ramsauer, Thomas Raste, and Hagen Boehmert. "A systematic approach based on STPA for developing a dependable architecture for fully automated driving vehicles." *Procedia Engineering* 179 (2017): 41-51.
110. Yoshida Junko. "AV Safety Ventures Beyond ISO 26262", Retrieved from https://www.eetimes.com/document.asp?doc_id=1334397, 2019.
111. ISO21448. "Road vehicles – Safety of the intended functionality.", 2019.
112. Neurohr, Christian, et al. "Identification & Quantification of Hazardous Scenarios for Highly Automated Driving." (2020).
113. Sorensen, Matthew B., Cherie M. Holland, Chris Van Berendonck, and Daniel S. Hourigan. "Challenges to achieving functional safety compliance of railway tunnel automated systems." In CORE 2018: Conference on Railway Excellence, p. 599. Railway Technical Society of Australasia (RTSA); Technical Society of Engineers Australia, 2018.
114. Jafarian, E., and M. A. Rezvani. "Application of fuzzy fault tree analysis for evaluation of railway safety risks: an evaluation of root causes for passenger train derailment." *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit* 226, no. 1 (2012): 14-25.
115. Pievatolo, industio, Fabrizio Ruggeri, and Raffaele Argiento. "Bayesian analysis and prediction of failures in underground trains." *Quality and Reliability Engineering International* 19, no. 4 (2003): 327-336.
116. Zhang, Wen-Jin, and Nan Lan. "Research on the reliability growth management techniques of high-speed train for whole life cycle." In *The 19th International Conference on Industrial Engineering and Engineering Management*, pp. 529-539. Springer, Berlin, Heidelberg, 2013.
117. Bearfield, George, and William Marsh. "Generalising event trees using Bayesian networks with a case study of train derailment." In *International Conference on Computer Safety, Reliability, and Security*, pp. 52-66. Springer, Berlin, Heidelberg, 2005.
118. Peng, Zhaoguang, Yu Lu, Alice Miller, Chris Johnson, and Tingdi Zhao. "Risk assessment of railway transportation systems using timed fault trees." *Quality and Reliability Engineering International* 32, no. 1 (2016): 181-194.
119. Wang, Yifei, Wenzheng Liu, Zhongping Yang, Xuan Ge, and Xiankai Liu. "Research on design evaluation of high-speed train auxiliary power supply system based on the AHP." In *2014 IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*, pp. 1-4. IEEE, 2014.
120. Martani, Claudio, Natalia Papathanasiou, and Bryan T. Adey. "A review of the state-of-the-art in railway risk management." In *Proc. ART2016 Jeju*. 2016.
121. Smoczyński, Piotr, and Adam Kadziński. "Estimation and evaluation of risk in the railway infrastructure." In *International Conference on Reliability and Statistics in Transportation and Communication*, pp. 182-191. Springer, Cham, 2017.
122. Alawad, Hamad, Sakdirat Kaewunruen, and An Min. "Risk management prediction for overcrowding in railway stations utilising Adaptive Neuro Fuzzy Inference System (ANFIS)." In *IOP Conference Series: Materials Science and Engineering*, vol. 603, no. 5, p. 052030. IOP Publishing, 2019.
123. Luo, Zhenhua, Li Zeng, Haize Pan, Qijun Hu, Bo Liang, and Jianqiang Han. "Research on Construction Safety Risk Assessment of New Subway Station Close-Attached Undercrossing the Existing Operating Station." *Mathematical Problems in Engineering* 2019 (2019).
124. Zhu, Xiangdong, Xiang Xiao, and Chaoran Wu. "The Study on Risk Assess Model of Rail Transit Projects." In *LTGLB 2012*, pp. 539-546. Springer, Berlin, Heidelberg, 2013.
125. Leitner, Bohus. "A general model for railway systems risk assessment with the use of railway accident scenarios analysis." *Procedia engineering* 187 (2017): 150-159.
126. Syeda, Kanza Noor, Syed Noorulhassan Shirazi, Syed Asad Ali Naqvi, Howard J. Parkinson, and Gary Bamford. "Big Data and Natural Language Processing for Analysing Railway Safety: Analysis of Railway Incident Reports." In *Human Performance Technology: Concepts, Methodologies, Tools, and Applications*, pp. 781-809. IGI Global, 2019.
127. Hadj-Mabrouk, Habib. "Contribution of artificial intelligence and machine learning to the assessment of the safety of critical software used in railway transport." *AIMS Electronics and Electrical Engineering* 3, no. 1 (2019): 33-70.
128. Alawad, Hamad, Sakdirat Kaewunruen, and An Min. "Utilizing big data for enhancing passenger safety in railway stations." In *IOP Conference Series: Materials Science and Engineering*, vol. 603, no. 5, p. 052031. IOP Publishing, 2019.

129. Hadj-Mabrouk, Habib. "Contribution of Artificial Intelligence to Risk Assessment of Railway Accidents." *Urban Rail Transit* 5, no. 2 (2019): 104-122.
130. Qin, Yong, and Limin Jia. *Active Safety Methodologies of Rail Transportation*. Springer, 2019.
131. Opricovic, Serafim, and Gwo-Hshiung Tzeng. "Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS." *European journal of operational research* 156, no. 2 (2004): 445-455.
132. IEC61508. "Functional safety of electrical / electronic / programmable electronic safety-related systems.", Geneva, 2010.
133. IEC61511. "Functional Safety - Safety Instrumented Systems for the Process Industry Sector, Part 1.", Geneva, 2016.
134. Bhatti, Zeeshan E., Partha S. Roop, and Roopak Sinha. "Unified functional safety assessment of industrial automation systems." *IEEE Transactions on Industrial Informatics* 13, no. 1 (2016): 17-26.
135. Bhatti, Zeeshan. "Model-Based Safety Assessment of Industrial Automation Systems using IEC 61499." PhD diss., ResearchSpace@ Auckland, 2017.
136. Baier, Christel. "Katoen, Joost-Pieter. Principles of Model Checking." (2008).
137. Suyama, Koichi, and Noboru Sebe. "A safety assessment framework of control systems according to international standards." *Asian Journal of Control* (2019).
138. Suyama, Koichi, and Nobuko Kosugi. "Safety assessment of control laws using Markov analysis." *Applied Mathematical Sciences* 6, no. 95 (2012): 4737-4762.
139. IEC62061. "Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems.", Geneva, 2005.
140. Ciucias, Michal, Waldemar Nowakowski, and Daniel Pietruszczak. "Safety of industrial automation systems." *AUTOBUSY–Technika, Eksploatacja, Systemy Transportowe* 24, no. 6 (2019): 50-55.
141. Lin, Kuo-Sui. "A Pattern Recognition Based FMEA for Safety-Critical SCADA Systems." In *Asian Conference on Intelligent Information and Database Systems*, pp. 26-39. Springer, Cham, 2019.
142. OCE. "Cybersecurity for Connected and Autonomous Vehicles: Considerations and opportunities for growth", Retrieved from <https://www2.deloitte.com/ca/en/pages/risk/articles/cyber-connected-autonomous.html>, 2019
143. Morris, David, Garikayi Madzudzo, and Alexeis Garcia-Perez. "Cybersecurity threats in the auto industry: Tensions in the knowledge environment." *Technological Forecasting and Social Change* 157 (2020): 120102.
144. El-Rewini, Zeinab, et al. "Cybersecurity challenges in vehicular communications." *Vehicular Communications* 23 (2020): 100214.
145. Hamid, Brahim, Barbara Gallina, Asaf Shabtai, Yuval Elovici, and Joaquin Garcia-Alfaro. *Security and Safety Interplay of Intelligent Software Systems*. Springer International Publishing, 2019.
146. Islam, Mafijul Md, Aljoscha Lautenbach, Christian Sandberg, and Tomas Olovsson. "A risk assessment framework for automotive embedded systems." In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pp. 3-14. ACM, 2016.
147. Kukkala, Vipin Kumar, Sudeep Pasricha, and Thomas Bradley. "SEDAN: Security-Aware Design of Time-Critical Automotive Networks." *IEEE Transactions on Vehicular Technology* (2020).
148. Török, Árpád, Zsolt Szalay, and Balázs Sághi. "Development of a Novel Automotive Cybersecurity, Integrity Level, Framework." *Acta Polytechnica Hungarica* 17.1 (2020).
149. Schoitsch, Erwin, and Christoph Schmittner. "Ongoing Cybersecurity and Safety Standardization Activities related to Highly Auto-mated/Autonomous Vehicles." 2020
150. Monteuiis, Jean-Philippe, Aymen Boudguiga, Jun Zhang, Houda Labiod, Alain Servel, and Pascal Urien. "Sara: Security automotive risk analysis method." In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, pp. 3-14. ACM, 2018.
151. Kaja, Nevruz. "Artificial Intelligence and Cybersecurity: Building an Automotive Cybersecurity Framework Using Machine Learning Algorithms." (2019).
152. Ma, Zhendong, and Christoph Schmittner. "Threat modeling for automotive security analysis." *Advanced Science and Technology Letters* 139 (2016): 333-339.
153. Xiong, Wenjun, and Robert Lagerström. "Threat modeling—A systematic literature review." *Computers & Security* (2019).
154. Schmittner, Christoph, et al. "ThreatGet: Threat modeling based approach for automated and connected vehicle systems." *AmE 2020-Automotive meets Electronics; 11th GMM-Symposium*. VDE, 2020.
155. Hamad, Mohammad, and Vassilis Prevelakis. "SAVTA: A Hybrid Vehicular Threat Model: Overview and Case Study." *Information* 11.5 (2020): 273.
156. Lee, Yousik, et al. "Practical Vulnerability-Information-Sharing Architecture for Automotive Security-Risk Analysis." *IEEE Access* 8 (2020): 120009-120018.
157. SAEJ3061, "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. ", 2016.
158. Arai, Kohei, Supriya Kapoor, and Rahul Bhatia, eds. *Advances in Information and Communication Networks: Proceedings of the 2018 Future of Information and Communication Conference (FICC)*. Vol. 2. Springer, 2019.

159. Bolovinou, Anastasia, Ugur-Ilker Atmaca, Obaid Ur-Rehman, Gerhard Wallraf, and Angelos Amditis. "TARA+: Controllability-aware Threat Analysis and Risk Assessment for L3 Automated Driving Systems." In 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 8-13. IEEE, 2019.
160. Gleirscher, Mario. "Risk Structures: Towards Engineering Risk-aware Autonomous Systems." arXiv preprint arXiv:1904.10386 (2019).
161. ISO26262. "Road vehicles – Functional safety Parts 1-12", 2018.
162. Sabaliauskaite, Giedre, and Sridhar Adepu. "Integrating six-step model with information flow diagrams for comprehensive analysis of cyber-physical system safety and security." In 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), pp. 41-48. IEEE, 2017.
163. Sabaliauskaite, Giedre, and Jin Cui. "Integrating autonomous vehicle safety and security." In Proceedings of the 2nd International Conference on Cyber-Technologies and Cyber-Systems (CYBER 2017), Barcelona, Spain, pp. 12-16. 2017.
164. Martin, Helmut, et al. "Combined automotive safety and security pattern engineering approach." Reliability Engineering & System Safety 198 (2020): 106773.
165. Bramberger, Robert, et al. "Co-engineering of Safety and Security Life Cycles for Engineering of Automotive Systems." ACM SIGAda Ada Letters 39.2 (2020): 41-48.
166. Kavallieratos, Georgios, Sokratis Katsikas, and Vasileios Gkioulos. "Cybersecurity and Safety Co-Engineering of Cyberphysical Systems—A Comprehensive Survey." Future Internet 12.4 (2020): 65.
167. Kawanishi, Yasuyuki, Hideaki Nishihara, Daisuke Souma, Hirotaka Yoshida, and Yoichi Hata. "A Comparative Study of JASO TP15002-Based Security Risk Assessment Methods for Connected Vehicle System Design." Security and Communication Networks 2019 (2019).
168. Promsri, Chaiyaset. "Passengers' Perception Towards Physical Security Measures of Suvarnabhumi Airport Rail Link Service." Mediterranean Journal of Social Sciences 6, no. 1 (2015): 309.
169. Marrone, Stefano, Ricardo J. Rodríguez, Roberto Nardone, Francesco Flammini, and Valeria Vittorini. "On synergies of cyber and physical security modelling in vulnerability assessment of railway systems." Computers & Electrical Engineering 47 (2015): 275-285.
170. Bloomfield, Robin, Marcus Bendele, Peter Bishop, Robert Stroud, and Simon Tonks. "The risk assessment of ERTMS-based railway systems from a cyber security perspective: Methodology and lessons learned." In International Conference on Reliability, Safety, and Security of Railway Systems, pp. 3-19. Springer, Cham, 2016.
171. Winther, Rune, Ole-Arnt Johnsen, and Bjørn Axel Gran. "Security assessments of safety critical systems using HAZOPs." In International Conference on Computer Safety, Reliability, and Security, pp. 14-24. Springer, Berlin, Heidelberg, 2001.
172. IEC62443. "Electric signaling systems for railways – Part 104: IT Security Guideline. ", Geneva, 2015.
173. Braband, Jens. "Towards an IT Security Risk Assessment Framework for Railway Automation." arXiv preprint arXiv:1704.01175 (2017).
174. Rekik, Mouna, Christophe Gransart, and Marion Berbineau. "Cyber-Physical Security Risk Assessment for Train Control and Monitoring Systems." In 2018 IEEE Conference on Communications and Network Security (CNS), pp. 1-9. IEEE, 2018.
175. Kaewunruen, Sakdirat, Hamad Alawad, and Silviu Cotruta. "A decision framework for managing the risk of terrorist threats at rail stations interconnected with airports." Safety 4, no. 3 (2018): 36.
176. Yi, Shengwei, Hongwei Wang, Yangyang Ma, Feng Xie, Puhua Zhang, and Liqing Di. "A safety-security assessment approach for communication-based train control (CBTC) systems based on the extended fault tree." In 2018 27th International Conference on Computer Communication and Networks (ICCCN), pp. 1-5. IEEE, 2018.
177. Schmittner, Christoph, Peter Tummeltshammer, David Hofbauer, Abdelkader Magdy Shaaban, Michael Meidlinger, Markus Tauber, Arndt Bonitz, Reinhard Hametner, and Manuela Brandstetter. "Threat Modeling in the Railway Domain." In International Conference on Reliability, Safety, and Security of Railway Systems, pp. 261-271. Springer, Cham, 2019.
178. Gao, Jingcheng, Jing Liu, Bharat Rajan, Rahul Nori, Bo Fu, Yang Xiao, Wei Liang, and C. L. Philip Chen. "SCADA communication and security issues." Security and Communication Networks 7, no. 1 (2014): 175-194.
179. Holmberg, David G. "BACnet wide area network security threat assessment." (2011).
180. Sadeghi, Ahmad-Reza, Christian Wachsmann, and Michael Waidner. "Security and privacy challenges in industrial internet of things." In 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-6. IEEE, 2015.
181. Sajid, Anam, Haider Abbas, and Kashif Saleem. "Cloud-assisted IoT-based SCADA systems security: A review of the state of the art and future challenges." IEEE Access 4 (2016): 1375-1384.
182. Mukama, Joseph. "Risk Analysis as a Security Metric for Industrial Control Systems." 2016.
183. Peng, Yuan, Kaixing Huang, Weixun Tu, and Chunjie Zhou. "A Model-Data Integrated Cyber Security Risk Assessment Method for Industrial Control Systems." In 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS), pp. 344-349. IEEE, 2018.
184. Zhu, Haoxiang, Jingqi Fu, Weihua Bao, and Zhengming Gao. "Quantitative Safety Assessment Method of Industrial Control System Based on Reduction Factor." In Intelligent Computing and Internet of Things, pp. 191-201. Springer, Singapore, 2018.

185. Abdo, H., M. Kaouk, J-M. Flaus, and F. Masse. "A safety/security risk analysis approach of Industrial Control Systems: A cyber bowtie—combining new version of attack tree with bowtie analysis." *Computers & Security* 72 (2018): 175-195.
186. Abdo, H., J. M. Flaus, and François Masse. "A fuzzy safety/cybersecurity risk analysis approach for more safe/secure industrial systems." 2018.
187. Poletykin, Alexey. "Cyber security risk assessment method for SCADA of industrial control systems." In 2018 International Russian Automation Conference (RusAutoCon), pp. 1-5. IEEE, 2018.
188. Lin, Kuo-Sui. "A New Evaluation Model for Information Security Risk Management of SCADA Systems." In 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS), pp. 757-762. IEEE, 2019.
189. Qassim, Qais Saif, Norziana Jamil, Maslina Daud, Ahmed Patel, and Norhamadi Ja'afar. "A review of security assessment methodologies in industrial control systems." *Information & Computer Security* 27, no. 1 (2019): 47-61.
190. Rashidi, Bahman, and Carol J. Fung. "A Survey of Android Security Threats and Defenses." *JoWUA* 6, no. 3 (2015): 3-35.
191. Theoharidou, Marianthi, Alexios Mylonas, and Dimitris Gritzalis. "A risk assessment method for smartphones." In IFIP International Information Security Conference, pp. 443-456. Springer, Berlin, Heidelberg, 2012.
192. ISO/IEC27005, "ISO/IEC: Information technology – Security techniques - Information security risk management. ", 2008.
193. Wang, Yang, Jun Zheng, Chen Sun, and Srinivas Mukkamala. "Quantitative security risk assessment of android permissions and applications." In IFIP Annual Conference on Data and Applications Security and Privacy, pp. 226-241. Springer, Berlin, Heidelberg, 2013.
194. Rashidi, Bahman, Carol Fung, and Elisa Bertino. "Android resource usage risk assessment using hidden Markov model and online learning." *Computers & Security* 65 (2017): 90-107.
195. Lederm, Thomas, and Nathan L. Clarke. "Risk assessment for mobile devices." In International Conference on Trust, Privacy and Security in Digital Business, pp. 210-221. Springer, Berlin, Heidelberg, 2011.
196. Zheng, Tian, Tian Jianwei, Qiao Hong, Li Xi, Zhu Hongyu, and Qi Wenhui. "Design of automated security assessment framework for mobile applications." In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 778-781. IEEE, 2017.
197. Sahay, Sanjay K., and Ashu Sharma. "A survey on the detection of android malicious apps." *Advances in Computer Communication and Computational Sciences*. Springer, Singapore, 2019. 437-446.
198. Abdullah, Zubaile, and Madihah Mohd Saudi. "RAPID-risk assessment of android permission and application programming interface (API) call for android botnet." *Int J Eng Technol* 7 (2018): 49-54.
199. Jing, Yiming, Gail-Joon Ahn, Ziming Zhao, and Hongxin Hu. "Riskmon: Continuous and automated risk assessment of mobile applications." In Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, ACM, 2014.
200. Padrihi, Deepthi Naidu. "Risk Assessment of Android Malwares Using Machine Learning Techniques." (2017).
201. Park, Mookyu, Jaehyeok Han, Haengrok Oh, and Kyungho Lee. "Threat Assessment for Android Environment with Connectivity to IoT Devices from the Perspective of Situational Awareness." *Wireless Communications and Mobile Computing* 2019 (2019).
202. Vecchiato, Daniel, Marco Vieira, and Eliane Martins. "Risk assessment of user-defined security configurations for android devices." In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 467-477. IEEE, 2016.
203. Asnar, Yudistira, and Bayu Hendradjaya. "Confidentiality and privacy information security risk assessment for Android-based mobile devices." In 2015 International Conference on Data and Software Engineering (ICoDSE), pp. 1-6. IEEE, 2015.
204. Mylonas, Alexios, Marianthi Theoharidou, and Dimitris Gritzalis. "Assessing privacy risks in android: A user-centric approach." In International Workshop on Risk Assessment and Risk-driven Testing, pp. 21-37. Springer, Cham, 2013.
205. Hamed, Asma, and Hella Kaffel Ben Ayed. "Privacy risk assessment and users' awareness for mobile apps permissions." In 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pp. 1-8. IEEE, 2016.
206. Agarwal, Yuvraj, and Malcolm Hall. "ProtectMyPrivacy: detecting and mitigating privacy leaks on iOS devices using crowdsourcing." In Proceeding of the 11th annual international conference on Mobile systems, applications, and services, pp. 97-110. ACM, 2013.
207. Beresford, Alastair R., Andrew Rice, Nicholas Skehin, and Ripduman Sohan. "Mockdroid: trading privacy for application functionality on smartphones." In Proceedings of the 12th workshop on mobile computing systems and applications, pp. 49-54. ACM, 2011.
208. Zhou, Yajin, Xinwen Zhang, Xuxian Jiang, and Vincent W. Freeh. "Taming information-stealing smartphone applications (on android)." In International conference on Trust and trustworthy computing, pp. 93-107. Springer, Berlin, Heidelberg, 2011.
209. Hatamian, Majid, Sebastian Pape, and Kai Rannenber. "ESARA: A Framework for Enterprise Smartphone Apps Risk Assessment." In IFIP International Conference on ICT Systems Security and Privacy Protection, pp. 165-179. Springer, Cham, 2019.
210. Schlingloff, Holger, Kruse, Peter M., Saadatmand, Mehrdad. "Excellence in variant testing" In Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VAMOS 2020). Association for Computing Machinery, New York, NY, USA, Article 12, 1–2. DOI:<https://doi.org/10.1145/3377024.3377028>