ITEA 3 Call 4: Smart Engineering

D3.7 Recommendations for homologation and certification of multi-variant and configurable safety-critical systems

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, A MEMBER OF THE ALSTOM GROUP, SWEDEN		
WEBSITE	HTTPS://WWW.XIVT.ORG/		

Document References

WORK PACKAGE	WP 3: TESTING OF CONFIGURABLE PRODUCTS	
TASK	T3.2	
VERSION	V 1.1	JAN. 31ST, 2022
DELIVERABLE TYPE	R (REPORT)	
DISSEMINATION LEVEL	PUBLIC	

Summary

This deliverable reports on the XIVT achievements w.r.t. homologation and certification of multi-variant and configurable safety-critical systems. It contains a survey of the representation of software product line testing in various standards, most notably, the new ISO/IEC 26554 on tools and methods for product line testing. Furthermore, it gives an overview of the state of practice as defined in the XIVT project via the use cases. Finally, the deliverable contains a set of recommendations for homologation and certification which derived from the experiences within the XIVT project.

Table of Contents

1. Introduction

Most industrial products are offered in different variants, with a varying set of features, and similar but not identical functionalities. This holds also for safety-critical systems, which can cause severe damage when malfunctioning. Safety-critical systems usually are subject to extensive homologation and certification procedures, to assure their overall quality and safety. However, present-day techniques usually focus on complete-system certification, where a system is seen as a monolithic block, with no relation to other, similar products. Thus, there is a significant overlap in activities when assessing several members of a product line. This document reports on experiences within the XIVT project with quality assurance of safety-critical variant and configurable systems, their representation in various safety-standards, and gives some recommendations for assessors and notified bodies on how to deal with this situation.

2. Representation of SPL-testing in standards

In this section we review the requirements for the assessment of multi-variant and configurable systems, as can be found in different safety-related standards and norms. We start with the ISO/IEC 61508 family of standards including Cenelec 50128 and ISO 26262, look at domain-specific standards like DO-178C in Aerospace and ISO 17025 in calibration laboratories, and then discuss specialized standards for testing (ISO/IEC/IEEE 29119) and cybersecurity (ISO/SAE FDIS 21434 and ISO 27000). Finally, we review the new standard 26554 for software product line testing.

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 50129 – System safety EN 50159 – Railway applications	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)

2.1 IEC 61508

IEC 61508 was first conceived in 1998 to replace several previous, more specialized standards on functional safety of electric / electronic/ programmable electronic systems. It quickly became generally accepted and spawned other, domain specific standards (e.g., IEC 61511 for the process industry, IEC 61513 for nuclear power plants, IEC 62061 for general machinery, etc; see subsequent chapters). Within Europe, the standard is known as EN 61508, the latest release being IEC/DIN EN 61508:2010. The aim of this standard is to define procedures that allow products to be manufactured such that they do not pose unacceptable hazards to users and the environment. The central notion in the standard is the *safety integrity level* (SIL) of a system, which ranges from SIL1 to SIL4. Important factors to determine the reliability of the safety function are the *probability of dangerous failure per hour* for systems operating continuously (in high-demand mode), and the *probability of dangerous failure on demand* (for other systems).

Part 3 of IEC 61508 is concerned with requirements on software. With respect to testing, it describes requirements for software module testing and integration testing. Module testing "*shall show whether or not each software module performs its intended function and does not perform unintended functions*". The standard explicitly confirms that this does not imply testing of all input combinations, nor of all output combinations. It may be sufficient to reduce the number of test cases by some appropriate technique such as boundary value analysis or control flow analysis. This is relevant for the XIVT results, since the test prioritization algorithms considered in XIVT can be regarded to be such techniques. Moreover, the standard notes that "*quantified statistical evidence ... is acceptable if all the relevant conditions for statistically valid evidence are satisfied.*" This implies for variant testing, that it is permissible to use statistical arguments to reason about the safety of configurable systems.

With respect to software integration testing, IEC 61508 requires that the software system integration test specification shall state the constituents and test cases for a specific software product. For software product lines, this means that an automatic assembly of test cases from the test cases for component software modules is admissible, as long as test criteria for judging the outcome of a test can also be automatically composed.

Module and integration testing can be used "*to ensure, in so far as it is appropriate, that configuration of programmable electronic systems by data fulfils the specified requirements for the software systematic capability.*" In other words, the standard allows to be applied to variant and configurable systems, as long as the requirements are adequately specified, in particular, as long as the software developer adequately considers the configuration or architecture of the system. Whenever safety requirements are expressed or implemented by configuration data, the data must shown to be consistent with the safety requirements, the ranges and authorized combinations of operational parameters must be specified and validated, and they must be defined in a manner such that they are compatible with the underlying software. For software product lines, this means that for each module in the baseline it must be specified and tested in which combinations it can be used.

2.2 Cenelec EN 50128 and DIN EN 50657

The Cenelec European Norm 50128:2011 is a specialization of EN 61508 to railway industry. It concerns safety-critical software in railway systems, originally both for track-side and in-vehicle equipment. Since 2017, the successor standard 50657 is concerned with on-board railway systems, whereas most roadside signalling solutions are still certified according to EN 50128.

The standard prescribes a safety-oriented design process, where each development phase must be validated with respect to its requirements. It inherits the notion of safety integrity level from the EN 61508, and all measures are mandatory/highly recommended/recommended/not recommended with

respect to a specific SIL. Concerning software product lines, the standard concedes that "*modern application design often makes use of generic software that is suitable as a basis for various applications. Such generic software is then configured by data, algorithms, or both, for producing the executable software for the application*". Here, the term "generic software" is defined to mean "*software which can be used for a variety of installations purely by the provision of application-specific data and/or algorithms*". This includes all baseline modules of a software product line. An example mentioned in the standard is a software module realizing the signalling principles of an interlocking system (e.g. signal management, point management), which is implemented by a set of application algorithms and parametrized by the specific layout of a certain railway track or station. For such systems, which are configured by application data or algorithms, the standard makes some special requirements:

Application Preparation Plan: This is an additional document which has to "*include verification and validation activities to ensure that the application data/algorithms are complete, correct and compatible with each other and with the generic application, and to provide evidence that the application conditions of the generic application are met*". It describes the production and compilation of the source code of the generic and specific data/algorithms, as well as verification and testing activities related to this production. There are several prescriptions on application preparation procedures, tools and testing. In particular, the Application Test Specification shall specify tests to be carried out to ensure correct integration of data/algorithms on generic hardware and software, if needed, and the correct integration of data/algorithms with a complete installation. In the context of XIVT, this requirement strongly implies that only those variants are tested which are actually to be deployed.

Tool validation: Tools relevant to the phases of the software development life cycle include ... *"f) application data tools that produce or maintain data which are required to define parameters and to instantiate system functions e.g. function parameters, instrument ranges, alarm and trip levels, output states to be adopted at failure, geographical layout*." This includes product line tools for the materialization of products from a feature model and baseline components. To our understanding, however, it does not include XIVT tools for testing of variant and configurable systems, since these do not directly influence the executable code of the application. However,

Documentation of generic software: In the Software Requirements Specification, all relevant modes of operation must be detailed and all relevant modes of behaviour, in particular failure behaviour, must be documented or referenced. The selected combinations for testing must be justified and documented. That is, for XIVT variant testing, not only the prioritization of test cases, but also the origin and coverage of each test suite must me documented. Moreover, the following important requirement is made in section 8:

Development of application data or algorithms: "*Care shall be taken in the verification process and validation test phase of the generic software in order to assure that* ***all relevant combinations of data and algorithms are considered****. If all relevant combinations of data and algorithms have not been considered in the verification, testing and validation process of the generic software, it shall be clearly identified as a limit of use of the generic software. A complement to verification, testing and validation process of the generic software shall be performed when some data or algorithms are defined beyond this limit.*" For the testing of multi-variant systems, this means that the components by themselves cannot be certified, and that an exhaustive test is necessary for all "relevant" combinations.

2.3 ISO 26262-1:2018 Road vehicles — Functional safety

Safety certification requires compliance with standards. For development of the variant-intensive safety-critical systems in the context of software product line (SPL), we still need to comply with safety standards and regulations. The assignment of system safety requirements to the components of the system under design is done using the Safety Integrity Levels (SILs) concept defined in the standards.

However, these safety requirements may vary in different products. Variation in the design can change the potential hazards in each product which can affect the safety requirements of the individual components in different products.

The standard ISO 26262 (Road vehicles — Functional safety) is an adaptation of the Functional Safety standard IEC 61508 for Automotive Electric/Electronic that uses the concept of Automotive Safety Integrity Level (ASIL) for allocation of safety requirements to the components of the system under design.

Product Certification: Compliance with the standard is accomplished by obtaining those ASIL levels. When the safety requirements are high the compliance will be very expensive. In variant-intensive automotive systems, the design and context variations affect not only hazard analysis, but it also impacts assignment of top-level safety goals and ASIL decomposition. The ASIL decomposition enables allocation of the targeted ASIL to a hazard while avoiding the exact contribution of all components to meet the targeted ASIL. In other words, the responsibility of reaching the ASIL allocated to a particular hazard will be shared among the components. Thus, assigning safety requirements to the components of the SPL demands defining the ASILs for the components to guarantee their safe use within the SPL. It is more beneficial for the economy to obtain safety certification of variant-intensive components while avoiding the unnecessarily stringent or expensive. However, this introduces challenges for safety engineering.

Safety analysis can be performed by utilizing a Model-Based Safety Assessment (MSBA) technique. It automatically identifies the combinations of component failures that result in hazards that can enhance ASIL decomposition. One of the commonly used techniques for MSBA is Hierarchically Performed Hazard Origin & Propagation Studies (HiP-HOPS). Recently, an approach is proposed by [*1] as an extension to HiP-HOPS which is called DEPendable-SPLE. It supports MBSA and SIL Allocation for product lines. In another study [Oli 20], a new approach is proposed based on integration of MBSA and ASIL decomposition techniques, Eclipse Process Framework (EPF) Composer and Base Variability Resolution (BVR) tools within the Advanced Material Analysis and Simulation Software (AMASS) Platform. This approach allows low-cost certification of the components in compliance with the assigned ASILs. However, it is limited to manual generation of the 150% EPF process model for the whole standard and a BVR model.

2.4 ISO/PAS 21448:2019 Road vehicles — Safety of the intended functionality

The ISO/PAS21448, namely SOTIF (Safety of the Intended Functionality), was introduced in 2019 to address safety challenges that autonomous vehicles software developers. Its scope is complementary to ISO 26262 and addresses unintended functionality and focuses on non-deterministic parts and AI algorithms (e.g. neural networks).

Product Certification: The safety goal of an autonomous vehicles is usually obtained by applying hazard analysis techniques and following the standard processes defined in ISO 26262 and ISO/PAS21448. There are various hazard analysis techniques such as Event Tree Analysis (ETA), Fault Tree Analysis (FTA), and Failure Mode and Effect Analysis (FMEA). However, they cannot properly analyze the hazards related to the variabilities in the current form. Thus, it is required to identify associated risks with different system variabilities and try to mitigate the associated risks by implementing the necessary safety guards. To overcome this issue, a study has been conducted through extending hazard analysis techniques to account for system with variabilities in [3#].

2.5 DO-178C and supplements

The DO-178C norm “Software Considerations in Airborne Systems and Equipment Certification” is the primary standard in the certification of aeronautical and airborne software. It was developed by RTCA and in Europe has been accepted by EUROCAE under the name ED-12C. DO-178C succeeded its predecessor DO-178B in 2012. It is supported by three supplements:

- DO-331: Model-based Development and Verification
- DO-332: Object-Oriented Technology and Related Techniques
- DO-333: Formal Methods

With respect to multi-variant and configurable software, a problem with DO-178C is that certification is only possible for a complete aircraft, engine, or propeller (10.0); the certification authorities consider the software as part of the airborne system or equipment installed on the certified product. Every software re-use from one model to another is problematic and in principle requires re-assessment. If software has been approved for one aircraft, however, approval within another model may be simplified.

DO-178C has five *Item Development Assurance Levels* (A to E); in the highest level, MC/DC code coverage is mandatory, and in level B decision coverage and statement coverage is required.

Specifically, in section 2.5.1 the distinction between software parametrization and configuration is made: a *Parameter Data Item* is “a data set that influences the behavior of the software without modifying the Executable Object Code and is managed as a separate configuration item”. Parameter Data Items (e.g., configuration tables and databases) are not part of the *Executable Object Code* and, hence, do not fall under the requirements for code verification and validation. Instead, the use of such data must be analysed, whether it is by the user, whether it selects/deselects some options in the code, or whether it is field-loadable data which may be corrupt or incompatible to the Executable Object Code. Parameter Data Items have the same criticality as the software which uses them. Usually, Executable Object Code and Parameter Data Item Files should be verified together, which forbids any of the variant testing methods developed in the XIVT project. However, verification of a Parameter Data Item can be conducted separately from the verification of the Executable Object Code if all four following conditions apply:

1. The Executable Object Code has been developed and verified by normal range testing to correctly handle all Parameter Data Item Files that comply with their defined structure and attributes.
2. The Executable Object Code is robust with respect to Parameter Data Item Files structures and attributes.
3. All behaviour of the Executable Object Code resulting from the contents of the Parameter Data Item File can be verified.
4. The structure of the life cycle data allows the parameter data item to be managed separately.

Verification of a Parameter Data Item File consists in checking that the structure of the file conforms to its description, and each data element in the file has been covered (i.e., is shown to have an admissible value which is also consistent with other elements).

With respect to the testing methods investigated in XIVT, the requirement that a Parameter Data Item may not modify the Executable Object Code excludes all variant generation methods (e.g., via feature mapping and instantiation) at the software developer’s side, and all code-modifying configuration methods (e.g., via conditional compilation or dynamic link libraries) at the user’s side. To include the latter possibility, DO-178C includes the notion of *User-Modifiable Software*. In 5.2.3 it requires that the non-modifiable components must be protected from modifiable components to prevent interference in their safe operation. Similar requirements are given for deactivated code (5.2.4). For software functions

which are selectable by software switches, a means must be provided to ensure that inadvertent selections involving non-approved configurations cannot be made (2.5.4). For software which can be loaded “in the field” (2.5.5), i.e., without removing the system or equipment from its installation, special measures must be taken (e.g., for detecting corrupted or partial files).

2.6 ISO 17025: General requirements for the competence of testing and calibration laboratories

ISO/IEC 17025, **General requirements for the competence of testing and calibration laboratories**, is the international reference for testing and calibration laboratories wanting to demonstrate their capacity to deliver reliable results.

ISO / IEC 17025 was developed by laboratory experts from all over the world, along with 18 liaison organizations, such as the International Laboratory Accreditation Cooperation (ILAC), and many associations representing laboratories.

ISO / IEC 17025 enables laboratories to demonstrate that they operate competently and generate valid results, thereby promoting confidence in their work both nationally and around the world, and as consequence promoting confidence in certified products.

It also helps facilitate cooperation between laboratories and other bodies by generating wider acceptance of results between countries. Test reports and certificates can be accepted from one country to another without the need for further testing, which, in turn, improves international trade.

Generic Product Certification: ISO / IEC 17025 was designed for generic products. In other words, associated to each product it is mandatory to have a test method. The test method defines what are the quality requirements that are mandatory to validate so that the product can receive the “Certified Product” logo.

The test method also defines the validation associated to each quality requirement. ISO / IEC 17025 defines validation as “the **confirmation by examination** and the provision of objective evidence that the particular requirements for specific intended use are fulfilled”.

However, when industry tried to use the ISO / IEC 17025 to software products, there was a problem. Each version of the software is seen as a product (different from the previous version) and this could create the nightmare of having a new test method for each software version.

And the test method must be accredited by the accreditation body, so that the Testing Lab can be empowered to certify the associated Product.

Software Product Certification: As stated above, it would not be feasible to submit for accreditation a new test method for each new software version. Instead, the accreditation body validates the capability of the Testing Lab personnel to understand software requirements and design test cases.

In Portugal there was this experience for software used by doctors to prescribe medicines, as an example. The software must follow specific rules like “if doctor prescribe medicine X, he cannot prescribe, at the same time, medicines of type B”. However, each software vendor has its own user interface and one test method for one product could not be used (as is) to validate another product from other vendor.

Therefore, the test method needs to be generic and it is support by ISO / IEC 25051 standard. This standard defines “Requirements for quality of Ready to Use Software Product (RUSP) and instructions

for testing". The term RUSP means a Software-as-a-Service and it also includes a product bought and installed on client premises.

ISO/IEC 25051 establishes:

- Quality Requirements for Ready to Use Software Product (RUSP);
- Requirements for test documentation for the testing of Ready to Use Software Product (RUSP), including test plan, test description, and test results;
- Instructions for conformity evaluation of Ready to Use Software Product (RUSP)

ISO/IEC 25051 divides the testing process into two different paths: one way is to install / use one sample unit of the software and evaluate its accuracy, or as alternative, use vendor test documentation as evidence of software compliance.

Testing Methods: Therefore, as stated above, the Testing Lab submit for accreditation its personnel skills (in software testing) followed by the process of received software products, defined the details of the test method (test cases design) using ISO / IEC 25051 as reference for quality requirements.

The test cases will be summarized in a document called "RTI" - Registration of Test Items. Each test item is a test step in the test case procedure and for each item there will be a "Passed"/"Failed" indication.

If there is one single test item failed, the product must be fixed and submitted again for certification.

The end result is the "Test Report" where it is described all test items, and the associated "Passed"/"Failed" result for each item. This report is the core of the product certification because it supports the evidence of "Certified Product" logo.

2.7 ISO/IEC/IEEE 29119 1-5: Software and systems engineering — Software testing

ISO/IEC/IEEE 29119 is a standardization series which covers systems and software testing. It consists of five parts - Concepts and definitions, Test processes, Test documentation, Test techniques and Keyword-driven testing.

According to the norm, the term "test item" is defined as "A test item is the result of an activity, such as management, development, maintenance, test itself, or other supporting processes." There are examples given, such as a software component, a subsystem, or the complete system. Nonetheless, the specification of the testing vocabulary does not exclude software product lines and can be applied to such an entity as a test item without adaption.

Testing techniques are classified as specification-based, structure-based or experience based. The first two subsets include Equivalence Partitioning, Combinatorial testing and the Classification tree method, State Transition testing, Statement, Branch, Decision and Branch Condition testing. The derivation of test coverage items is emphasized for all the techniques and elaborated in a separate chapter.

It is noteable that the above recommended techniques and coverage criteria are - with minor adaptations - to Software Product Lines on the level of variability models resp. domain testing in the notion of ISO/IEC 26554.

2.8 IEC 62443 Industrial communication networks - IT security for networks and systems

IEC 62443 is a set of security standards for the secure development of Industrial Automation and Control Systems (IACS) based on already established standards (ISO 27001). It provides a thorough and systematic set of cybersecurity recommendations. It's used to defend industrial networks against cybersecurity threats. The IEC 62443 structure is shown in the following figure.

Figure : Overview of IEC 62243 by TÜV Süd AG

A key part is security levels (SL). SL is used to assess the cybersecurity risks to each system. And it helps you understand how to best address those risks. There are five Security Level values, 0–4. SL 0 is the minimum level of risk and SL 4 is the maximum. That means that SL 4 has stricter compliance requirements than SL 0.

Security level	**Protection against**
0	No specific requirements or security protection are necessary.
1	Protection against casual or coincidental violation.
2	Protection against intentional violation using simple means with: • Low resources. • Generic skills. • Low motivation.
3	Protection against intentional violation using sophisticated means with: • Moderate resources. • IACS specific skills. • Moderate motivation.
4	Protection against intentional attacks with sophisticated means with: • Extended resources. • IACS specific skills. • High motivation.

Table : Security level according to IEC 62443

Another important point of IEC 62443 is the classification into different maturity levels, according to the following table

Maturity level	Category	Organization is capable of performing a service...
1	Initial	without a documented process that is poorly controlled
2	Managed	with a formal documented process with evidence of expertise and trained personnel
3	Defined	at maturity level 2 and demonstrates the of defined, established and documented processes as well as defined training schemas for personnel
4&5	Improving	at maturity level 3 as well as demonstration of continuous improvement (e.g., internal audit report)

Table : Maturity level according to IEC 62443

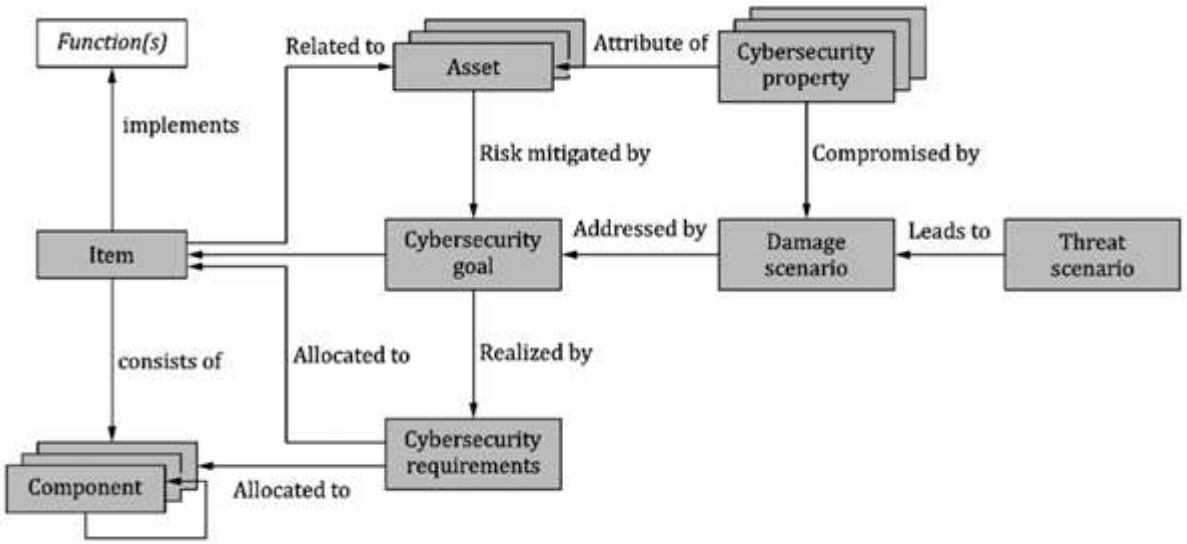
The protection level is a derived combination of the maturity level and the security level. The higher the maturity and safety level, the higher the protection level. The IEC 62443 series of standards does not contain any precise specification on the subject of testing variant-rich systems. Only one configuration is considered at a time, the standardization effort is quite high and must be approved by an external testing institute such as TÜV Süd.

2.9 ISO/SAE FDIS 21434

ISO/SAE FDIS 21434 is the international standard that addresses the cybersecurity perspective in engineering of electrical and electronic (E/E) systems within road vehicles.

By ensuring appropriate consideration of cybersecurity, this standard aims to enable the engineering of E/E systems to keep up with changing technology and attack methods. To do so, it specifies requirements for cybersecurity risk management regarding engineering for concept, development, production, operation, maintenance, and decommissioning for road vehicle electrical and E/E systems, including their components and interfaces.

It is important to notice that the application of this standard is limited to cybersecurity relevant items and components inside or on the vehicle perimeter including aftermarket and service parts. Systems outside the vehicle perimeter can be considered for cybersecurity purposes but are not in the scope of this standard. The following are examples of what can be considered for the vehicle level as a whole: the vehicle E/E architecture; and the cybersecurity cases of the cybersecurity relevant items and components. The relationship between the item and component and its applicable cybersecurity requirements is shown in the next figure.

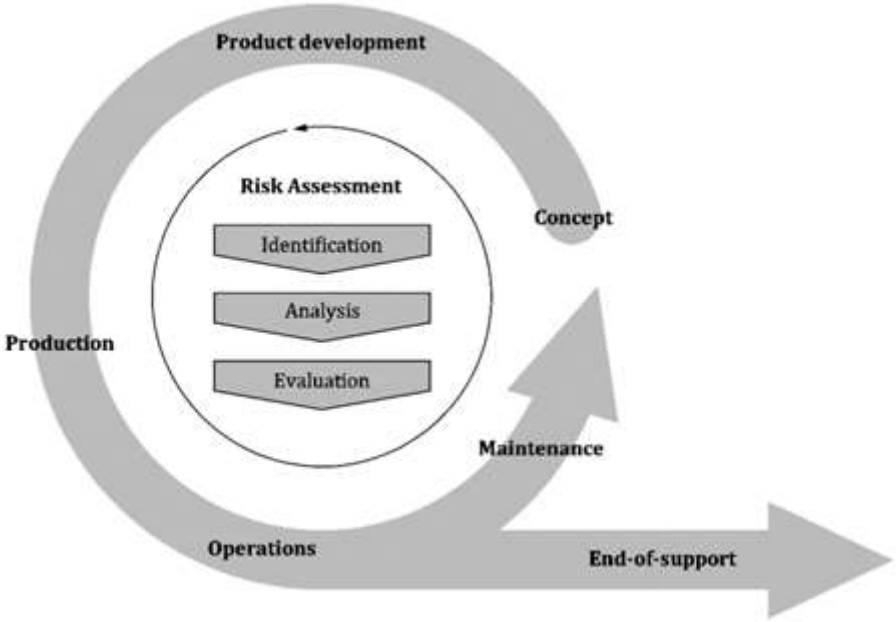


Requirement generation for cybersecurity relevant items or components

A defence-in-depth approach can be used to mitigate threat scenarios and risk. The defence-in-depth approach utilizes layers of cybersecurity measures to improve the cybersecurity of the item or vehicle. If an attack is able to penetrate or bypass one layer, another cybersecurity layer can help contain the attack and continue to maintain a sufficient degree of cybersecurity

The implementation of a cybersecurity management system is based on the definition of a framework that includes requirements for cybersecurity processes and a common language for communicating and managing cybersecurity risk. ISO/SAE FDIS 21434 can be used to implement a cybersecurity management system including cybersecurity risk management in accordance with ISO 31000.

To enable cybersecurity engineering involves specifying organization-specific rules and processes that are independently audited. These rules and processes cover concept, development, production, operation, maintenance, and decommissioning, including cybersecurity risk management, information sharing, vulnerability disclosure, cybersecurity monitoring, and incident response. On the other hand, the overall cybersecurity risk management of an organization is implemented in accordance with both cybersecurity engineering and ISO 31000 and applies throughout all phases. The following figure illustrates the mapping of the general risk assessment tasks of ISO 31000 and its adaptation for the purpose of ISO/SAE FDIS 21434.



Mapping ISO31000 tasks and its adaptations for the purpose of ISO/SAE FDIS 21434

2.10 ISO/IEC 27000:2018

This section gives an overview of the ISO/IEC 27000:2018 standard, in terms of what is an Information Security Management system (ISMS), its purpose in term of security within an organization and which aspects organizations must take into consideration to turn them protected against threats. Right way, we notice that this standard is not specific for software testing. However, as software is the pillar of any process and task provided and performed by organizations, and considered an asset of an ISMS, it is essential testing it adequately and intensively.

ISO/IEC 27000:2018 is one of the international standards for management systems and provide a model to follow in setting up and operating a management system. Specifically this standard is for management systems based on information security, and hence known as the Information Security Management System (ISMS). However, before setting up such system, it is need organizations be aware of what is information security and how can they use and process information from their organization infrastructure's assets to reach the former.

Organizations collect, process, store, and transmit information. They recognize that information, and related processes, systems, networks and people as being important assets for achieving organization objectives. However, organizations, since some years ago, have faced a range of risks that can affect the functioning of assets, which result in asset misbehaviours and consequently serious damages to organizations. To tackle such risks, they address their perceived risk exposure by implementing information security controls (e.g., firewalls, IDSs), which are monitored by security operation centre's analysts (SOC) for finding attempts of assets' disruptions and taking actions against such threats.

Information security (IS) is associated with information belonging to an asset which has a value within organization and, hence , requires appropriate protection. More specifically, IS ensures the confidentiality, availability and integrity of information. For example, a database (i.e., an asset) containing critical and sensitive data that must be protected against the loss of confidentiality, integrity and availability. Therefore, protecting information assets is mandatory and essential to enable an

organization to achieve its objectives and keep it secure against threats and cyberattacks. Such protection is made through defining, achieving, maintaining, and improving IS effectively. This means that IS involves the application and management of appropriate controls that involves consideration of a wide range of threats, with the aim of ensuring sustained business success and continuity, and minimizing consequences of information security incidents. Furthermore, enabling accurate and complete IS timely will allow business efficiency and organizations' reliability.

Risks associated with an organization's information assets need to be addressed. Achieving information security requires the management of risk, and encompasses risks from physical, human and technology related threats associated with all forms of information within or used by the organization.
As IS risks and the effectiveness of IS controls can change depending on shifting circumstances, organizations need to monitor and evaluate the effectiveness of implemented controls and procedures. For that, SOC's analysts inspect the information collected from the monitored assets with the goal to identify emerging risks to be treated, and then to select, implement and improve appropriate controls as needed. To interrelate and coordinate such information security activities, each organization needs to establish its policy and objectives for IS and achieve those objectives effectively by using a management system, i.e., an ISMS.

An ISMS consists of the policies, procedures, guidelines, and associated resources and activities, collectively managed by an organization, in the pursuit of protecting its information assets. An ISMS is a systematic approach for establishing, implementing, operating, monitoring, reviewing, maintaining and improving an organization's IS to achieve business objectives. It is based on a risk assessment and the organization's risk acceptance levels designed to effectively treat and manage risks. Analysing requirements for the protection of information assets and applying appropriate controls to ensure the protection of these information assets, as required, contributes to the successful implementation of an ISMS. Through ISMS family of standards, organizations can develop and implement a framework for managing the security of their information assets.

As we stated initially, software is in base of every system and process and task performed by any organizations. The process of testing in the software life cycle development plays an important role because through it is the only way to test software's functionalities and security properly. However, testing intensively whole a software system takes a long time, which is not feasible for organizations that want to place their software in the market. However, this does not mean that software is not tested before making available to consumers, but can be available with some vulnerabilities that were not detected during the testing process. For example, there was not generated a test case that was capable of exploiting them.

The risk assessment of software is measured by the implementation of security aspects during its development, meaning that as much vulnerabilities it contains as greater is the risk. Within an ISMS, software is considered as assets, mainly those that are considered as being valuable to the organization. They are monitoring in order to check whether any threat arise and compromise its functioning, as for instance by the exploitation of a vulnerability that have been undisclosed. In that time, the risk of such software increases in the ISMS and remains high until the vulnerable software is updated with a patch that removes the vulnerability. To obtain the patch, the software provider must develop it and test it along with the target software.

In sum, despite the ISO/IEC 27000:2018 standard is not specific to software testing, it deals indirectly with it, by measuring the risk of software, since software is part integral of an ISMS.

2.11 ISO/IEC 26554: Information technology — Software and systems engineering — Tools and methods for product line testing

ISO/IEC 26554 is a standard covering testing methods and tools for software product lines. It gives a reference model for product line testing and derives a process for product line test management. The focus lies on the differentiation between domain and application testing, where domain testing takes place before variability is resolved, and application testing, where variability is included in domain artefacts.

Domain testing considers tests for non-executable domain artifacts, as variability is not yet resolved. Domain test assets should be correctly reused in application testing. In particular, it is recommended that domain test assets include variability in a way that "testing for a member product of a product line should devise a method to reuse domain test cases and minimize regression testing for the parts that are already tested in domain testing or tested by another member product". Variability models are an important tool to reach this goal. The need for quality for domain testing artefacts is emphasized, in particular the challenging task to develop strategies for including non-executable domain artifacts, due to absent variants.
The application testing validates the final member product against the application requirements by re.testing or regression testing already tested domain parts and by performing additional application-specific tests.
The two-fold process shall guide to minimize test efforts by avoiding redundant test execution and reusing domain assets. The crucial challenge here are the heterogeneous binding times for variability. This makes the use of stubs and drivers for system and integration testing necessary depending on the specific time in the development process when binding occurs, leading to the need to balance between defect correction costs and test development effort.
Variability management therefore plays an important role in enabling the smooth transition and efficient and effective use of artefacts between domain and application testing. Requirements for methods and tools for variability management are in particular named as:

- *Variability mechanism category in testing* maintains a set of variability implementation mechanisms that can be used for expressing or realizing variability in domain test artefacts.
- *Variability in test artefacts* serves to define variability types and ways to express variability included in test artefacts such as test plans, test cases and test scenarios.
- *Traceability of variability in test* establishes and maintains trace links between variability in test artefacts and variability models in test defined separately. The trace links are required to support the reuse of test artefacts in application testing.

The product line test management strategy is derived from the test management strategy described in ISO/IEC/IEEE 29119, but has to take both the domain and application testing level into account. Test strategies alternatives should be provided and test case selection criteria for both levels should be in place depending on the product line evolution, and additional metrics should be defined that evaluate the strategies for a product line engineering viewpoint.

Some high level examples for SPL test strategies are given at the end, ranging from domain-testing-only over testing of common assets of the SPL via sampling and deriving reusable domain test assets to an application-testing only approach

The need for a potent variability management and a level-overarching asset and artefact management are strongly emphasized in the content of the norm. However, the norm leaves the description and

requirements for these on a high level, and opposed to ISO/IEC 29119, no specific methods are recommended for the testing of SPLs.

2.12 Reflection

As we have seen, the standards mentioned in this section treat testing of configurable and highly-variant systems not explicitly or, as in the case of ISO/IEC 26554, at an abstract level, without specific recommendations for assessors. A question is how the certification rules have been fulfilled when the amount of testing is reduced and how the certification rules have been fulfilled in different domains.

The traditional approach to certification is to validate each product individually. This validation is mostly done by systematic testing. The test method defines which quality requirements must be satisfied so that the product can receive the “Certified Product” logo. When we have a highly variant and configurable product, with such an approach the amount of testing would be infeasible. A huge number of test cases would be required to have evidence of compliance of each possible combination under all configuration options. Certification would be impossible both from a financial and practical point of view. Therefore, in the assessment of such product lines, the focus must shift from a product centric certification to a process centric one. As will be elaborated in Section 4.2, there are several items which can be checked for a quality-oriented software product line development. Of course, there is no guarantee that a good process leads to high-quality products; all experiences, however, show that a bad process results in low-quality products. Thus, an assessor must pay increased attention to process-related topics and artefacts.

3. State of practice in XIVT

In this section, we report on the tools and experiences within the XIVT project with quality assurance of safety-critical multi-variant and configurable systems. The methods investigated include quality modelling and measurement, test case generation for software product lines, simulation-based testing, variant selection for testing, risk-based test scoring, and cybersecurity analysis.

3.1 Measuring the quality of the requirements using ISO 25000:2014

Within XIVT, a toolchain for test case generation bases on requirements provided by the XIVT partner ifak was employed. Requirements define what the system should do. The IEEE Standard Glossary of Software Engineering Terminology [IEEE 90] defines a requirement as: *A condition or capability needed by a user to solve a problem or achieve an objective*. In order to gain high quality test cases covering as much as possible features, the requirements preferably should be completely represent the functionality of the system. Furthermore, the requirements should be free of contradictions and complementary. Contradictory test cases as well as test cases addressing the same functionality result in failed test executions and unnecessary long test execution times. So, the *quality of the requirements* is crucial for the quality of the test cases and finally for the quality of the product.

The ISO/IEC 25000:2014-03 Systems and software engineering – Systems and software Quality Requirements and Evaluation – Guideline for SQuaRE introduces and provides guidance for the use of the series of International Standards named Systems and software Quality Requirements and Evaluation (SQuaRE). The purpose of ISO/IEC 25000:2014 is to provide a general overview of SQuaRE contents, common reference models and definitions, as well as the relationship among the documents, allowing users of the Guide a good understanding of those series of standards, according to their

purpose of use. It also contains an explanation of the transition process between the old ISO/IEC 9126 and the ISO/IEC 14598 series and SQuaRE.

As part of ISO/IEC 25000, the ISO/IEC 25010:2011-03 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models defines

a. A quality in use model composed of five characteristics (effectiveness, efficiency, satisfaction, freedom from risk, context coverage) that relate to the outcome of interaction when a product is used in a particular context of use and – more important for us –
b. A product quality model composed of eight characteristic (functional suitability, performance, efficiency, usability, reliability, security, maintainability, portability) that relate to *static properties of software* and dynamic properties of the computer system.

The characteristics defined by both models are relevant to all software products and computer systems. The characteristics and subcharacteristics provide consistent terminology for specifying, measuring and evaluating system and software product quality.

The quality models together serve as a framework to ensure that all characteristics of quality are considered. These models provide a set of quality characteristics relevant to a wide range of stakeholders, such as: software developers, system integrators, acquirers, owners, contractors, quality assurance and control professionals, and users. Activities during product development that can benefit from the use of the quality models include:

- Identifying software and systems requirements;
- Validating the comprehensiveness of a requirements definition;

The measurable quality-related properties of a system are called quality properties, with associated quality measures. Internal measures based on inspecting static properties can be used to measure inherent properties of a software work product. Static analysis methods include inspection and automated analysis tools. Work products include requirements and design documents, source code, and test procedures.

According to the models described in the ISO/IEC 25010, using the quality models and particularly the product quality model helps measuring the *quality of the requirements* indicating the quality of the test cases.

3.2 Test generation for variant-intensive industrial control software

Tools and methods: Simulink Design Verifier (**SLDV)** is the de-facto standard tool for validating and verifying Simulink models. Using automatic theorem-proving and model checking technologies, it provides automated test-case generation and property proving. SLDV can operate in three different modes: Design Error Detection, Test Generation, and Property Proving. For design error detection, it can detect several different errors, e.g., dead logic, overflows, division by zero, and specified minimum and maximum value violations. In property proving mode, the objectives are manually specified using Simulink as a specification language. Assertion blocks will be used by the model-checker to prove that the assertions can never be raised. The assertion blocks also enable online test execution, stopping simulation if an assertion fails. For design error detection and property proving, SLDV provides counterexamples. SLDV supports a subset of the modeling language, the same subset as supported by the code generation with a few exceptions. For unsupported blocks, SLDV can perform subbing, and the user can manually specify block replacement rules. SLDV approximates floating-point arithmetic with rational number arithmetic. Test generation is based on a user-specified coverage objective. SLDV relies on Simulink Coverage for measuring the coverage during test generation and of the generated

test cases. The test generation can use both the model and the generated production code as a target for the coverage objective. SLDV accepts a coverage filter of objectives to exclude during test generation. It is also possible to provide a coverage report that SLDV will try to extend as to achieve any missing coverage. **SEAFOX** and **COMPLETETEST** are test generation tools developed by MDH. These tools can 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 and CompleteTest tools create test cases by generating the inputs using one of the available algorithms implemented in the tool: Random, Base choice, Pairwise algorithm or code coverage. 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.

Influence of Standards: The International Electrotechnical Commission (IEC) **61131 standard** was created as a basis for uniform PLC programming by a working group of the international standardisation organisation IEC that represented different PLC manufacturers, software developers and users of both kinds of PLCs. This standard has influenced the development of SEAFOX and COMPLETETEST. Using the concepts of the modern software technologies the IEC established this standard to ensure that the common guidelines were followed amongst the PLC hardware manufacturers and software developers for PLCs. The first publications of the assembled documentation of the standard appeared in 1993. Since then, it has been updated continuously and now consists of a total of ten parts, where part three (i.e., IEC 61131-3) is dedicated to PLC programming languages. There are five standard defined PLC programming languages. All the programs written in standard defined languages can be exchanged amongst the software tools that follows the IEC 61131-3 standard. In that way, they can be reused or adapted for further development. The file format of the exported programs can differ from tool to tool but the most used file format in the field today is PLCopen XML exchangeable format, that recently became known also as a part ten (i.e., IEC 61131-10) of the standard. The solution that we use in our SEAFOX tool increases the generalizability of the SEAFOX tool to be used with IEC 61131-3 PLC software.

When we evaluated these tools on industrial FBD programs and to acquire experience regarding its efficiency and usability, we conducted a set of analyses using programs developed by industry. The system has been in development for more than two years and uses processes influenced by safety-critical requirements and regulations including the **EN 50128 standard** which requires different logic coverage levels (e.g., DC and MC/DC). This influenced the set of coverage criteria we adapted for our case.

Related to SLDV, the studies we performed relate to the development of safety-critical systems for train and rail products. The software deployed on a train has to be developed in accordance with relevant safety-regulation and standards. For the rail industry, the standard already mentioned **EN 50128:2011** - Railway applications - Communication, signaling and processing systems describes the requirements and procedures for software testing of railway systems. The standard **EN 50657:2017** Railways Applications - Rolling stock applications -Software on Board Rolling Stock adapts EN 50128:2011 for software in use on rolling stock. These standards require rigorous validation and verification practices. Our results suggest that the generated test cases achieve similar code coverage (i.e., decision and condition coverage) as the test cases manually engineered by expert testers. However, for Modified Condition/Decision Coverage, the automatically generated test cases achieve higher coverage, and the difference is significant. Our studies suggests that while much effort is required for custom tooling and initial setup, test generation can prove useful for covering Simulink models as well as manually created test cases and can be used for early fault detection during component-level testing in an industrial setting. These results are influenced by the code coverage specified by these standards.

3.3 Simulation-based testing

For complex cyber-physical systems, extensive testing of variants in a physical environment faces some barriers which can have the effect that the approach may not be feasible. Firstly, as explained elsewhere, the number of possible variants in a feature-reach design can be very large. Secondly, it may not be practical to actually build all variants in the physical reality, since the production and set-up of each variant may require a lot of resources. Thirdly, executing a single test may take a lot of time, since physical constraints such as acceleration and mass inertia are relevant. To overcome these barriers, in the XIVT project a simulation-based testing approach for variant-rich cyber-physical systems has been conceived. The testing environment has been developed and demonstrated with the example of a flexible production cell, where several robotic arms cooperate to handle a common workpiece. Variability arises from the different types of robotic arms, the different grippers and tools these robots can use, and the positioning of the robots in the production cell.

The simulation environment is based on the Gazebo simulator, which is part of the ROS (robotic operating system) ecosystem. Test cases are specified and executed with the help of the Robot Framework and the Robot ROS Gazebo Library, a custom keyword library developed within XIVT.

As part of the project, we created a robotics demonstrator for evaluating simulation-based testing in comparison to physical testing. The main focus was on reducing the efforts for test setup, changeover and test execution for a configurable robotics application. The setup consists of two desktop-sized robotic arms which can each be positioned at 12 different spots on the work area. Given additional physical constraints, this creates a total of 80 positional variants. In addition, each robot can be equipped with one of five different tools, leading to a total of 1680 variants. Obviously, even with such a “toy example”, not all possible variants can actually be tested in the physical setup. Instead, we measured individual tasks for setup and changeover, i.e., tool change and re-positioning. They were each performed multiple times and the average duration is reported below. The total duration of the test setup was then estimated as the sum of two re-positionings and two tool changes. We estimate a test execution strategy optimized towards minimal changeover time to reduce this by approx. 50% for consecutive test executions. Consequently, we report the full duration for test suite setup and the reduced duration for test case setup.

In contrast to the physical setup, the simulation requires only minimal changeover time since new configurations can be launched instantly. The reported times for test setup arise from the startup phase of the simulator. We expect this could be further reduced by a more optimized use of resources. Further, simulation allows taking advantage of parallelization, since virtual resources are easier to replicate than physical ones. Hence, we created a Docker based setup running multiple simulations for testing in parallel. The reported test execution duration is the average duration for executing a single test case. The reported test suite execution time is the average duration for executing 80 test cases with eight concurrent simulation workers. As we are running the same test cases in both, the physical and the simulated setup, the test execution time is the same for the physical setup. The test suite execution time, however, corresponds to a sequential execution of 80 test cases including test setup for each test case.

	Virtual	Physical	Factor
tool change	N/A (instant)	**49.61 s**	N/A
re-position	N/A (instant)	**1:35.53 m**	N/A
test case setup		*2:25.14 m*	
test case execution	**1:10 m**	*1:10 m*	1.0
test suite setup	**9:41 m**	*4:50.28 m*	0.5
test suite execution	**12:45 m**	*4:46:48 h*	22.5

Note: Bold font indicates **actual measurements**, values set in italics were *calculated* as described above.

From this comparison it becomes apparent, that testing in simulation bears great potential for reducing the manual as well as the total effort for testing of variant-rich cyber-physical systems while maintaining full test coverage. It must be noted, however, that not all properties of a cyber-physical system can sufficiently be replicated in a simulation and thus testing in simulation should be accompanied by physical testing for such properties.

3.5 Variant selection for testing

The XIVT variant selection tool **VarSel** takes a given set of variants that have been derived from a variability-including feature model (such as generates in **BeVAR** or **TESTONA**) and gives out a subset of these variants that fulfills a desired combinatorial coverage criterion (feature coverage or pairwise coverage). **VarSel** therefore enables combinatorial coverage criteria, recommended for testing in ISO 29119, to be applied to the level of variant-defining features. Coverage levels are optimized and can be traced for subsubsets of the generated variant subsets, enabling a trade-off between coverage and time/effort constraints.

TESTONA is a tool for modelling and test case generation based on the classification tree method. Its modelling capabilities were enhanced to variability modelling by enabling the generation of a "150%" classification tree, from which product specific classification trees can be derived and then undergo the test case generation process on the product level (or application testing level as in ISO 26554).

As there exists a mapping between classification trees and feature models, the interface between TESTONA and VarSel now enables generation of test cases overarching the levels of domain and product testing and the tracing and ensuring of combinatorial coverage in testing.

The requirement management tool **MERAN** has been enriched by attributes to map requirements and requirement-internal parameters to specific variants. The capabilities of VarSel were adapted and incorporated into MERAN in a way that from the allowed set of variants, a subset of variants can be chosen such that all requirements are covered. For requirement driven testing, this gives a way to trace and ensure requirement coverage while optimizing the testing effort.

3.6 Risk-based test scoring

Within XIVT, we have developed a solution for scoring the test cases called **RiSco**. The inputs are the “abstract test suite” and a domain “specific knowledge base” to a knowledge-based system and the output is a set of scored test suites based on a risk which could be a safety and security related risk.

The selected Abstract Test Case has three main parts, a pre-execution condition, input value, and impacted sets. The Impacted set is scored based on the number of components relative to pre-condition set (In fact, number of components in the impacted set indicates the number of possible faults that can be identified by a particular test case and the priority level of the components in this set (high-risk components should have a higher priority which can affect overall score of the test cases).

From the domain specific requirements, the operational situation will be identified which consist of two parts, “System states” and “Environment condition”. Each part has some attributes, and each attribute may include a number of states. For example, for 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. 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, controllability factors provided in ISO 26262. 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.

3.7 Natural language requirements analysis

Natural Language (NL) requirements are at the centre of many software development processes. This holds in particular for software products of the XIVT partner Bombardier Transportation, a member of the Alstom Group. In a normal project delivery scenario, NL requirements are received that are to be developed and tested. In this regard, the team might get a common requirement that is already implemented in a variant or the standard product before. Reusing code and other artifacts related to those common requirements might help in reducing development time, cost, the efforts for safety re-certification and testing [AJL+20]. RISE, in collaboration with Bombardier Transportation, a member of the Alstom Group and MDH, developed a tooled approach called Variability Aware requirements Reuse Analysis (**VARA**) to help engineers in identifying reuse opportunities based on NL requirements [ASE+20]. VARA uses Natural Language Processing (NLP) together with Machine Learning (ML) to index existing requirements and their links to existing software so that they can be later recommended for reuse. Given new NL requirements for a new project, VARA recommends the reuse of existing software components for each requirement using ML. VARA was applied on two projects at Bombardier Transportation, a member of the Alstom Group, and results show that VARA is able to recommend reuse with around 82% of accuracy. Results further indicate that the recommendations from VARA are useful and can save development, testing, and re-certification efforts at Bombardier Transportation, a member of the Alstom Group.

FCUL, in collaboration with Bombardier Transportation (BT), a member of the Alstom Group, developed **ARRINA** (Association and Recommendation for Requirements in Natural Language), a recommendation system to help engineers in linking requirements of new projects to design specifications of existing Bombardier's projects and in prioritizing requirements for testing software, both in an automated manner. Also, ARRINA provides the knowledge contained in a Propulsion Control Subsystem (PCS) based on knowledge extraction with chunking and rule mining [LM21]. The ARRINA model establishes a pipeline of NLP and Information Retrieval methods and a weight similarity metric to process design specifications and customer requirements, both written in unstructured natural language and belonging to two different, but related, domains.

ARRINA processed four PCSs of BT, and results showed that it is capable to recommend design specifications with an accuracy of 90%, representing a reduction of engineer time effort that can reach 85%.

3.9 Cybersecurity

Cybersecurity has become a top priority for most organizations. To more aptly protect themselves, organizations are moving from reactive to proactive defensive measures. They are investing in cyber threat intelligence (CTI) combined with information from their network infrastructure and assets to provide them forewarning about the risks they face, as well as to accelerate their response times in the detection of attacks. For that, ISO/IEC 27000:2018 purposes an Information Security Management system (ISMS), which aims the management of security within an organization and which aspects organizations must take into consideration to turn them protected against threats. Also, in a more specific domain, ISO/SAE FDIS 21434 addresses the cybersecurity perspective in engineering of electrical and electronic (E/E) systems within road vehicles. Despite none of these standards being specifically related to software security testing, actually organizations' assets run software which are

potential targets for cyber-attacks. Therefore, security testing is an utmost important task before deploying these assets. Moreover, it is critical to maintain the software updated during its lifetime and free of vulnerabilities, patching them with new releases and, hence, protecting organizations from cyber-threats.
Within XIVT, security testing solutions for software have been developed, which are capable of detecting vulnerabilities and removing them automatically by correction of the source code. The solutions are domain agnostic and look for vulnerabilities in C/C++ programs, like software of embedded systems and operating systems.

DeltaFuzzer is a grey box fuzzer based on the AFL fuzzer to detect several classes of vulnerabilities presented in software constructed for C/C++. It is the first fuzzer that implements a *Targeted Fuzzer Approach*, making the fuzzer focus on the (novel) parts that needed to be tested and reusing knowledge acquired in previous testing campaigns. DeltaFuzzer generates a test case (randomly or through a mutation strategy of existing test cases) for running it in the software under test (SUT) and collects various metrics. Next, it determines if the program suffered a failure, saving thus the test case, and determining if the test case is “interesting”, i.e., if it is capable of uncovering new execution paths and causing a SUT failure. In this case, it is saved it and reused to generate other test cases. DeltaFuzzer evolved from **PandoraFuzzer** which reuses the results of testing session between variants that share same software functionalities [AMN20]. PandoraFuzzer first retrieves information about the SUT structure, identifying basic blocks. Next, it generates test cases, deciding which blocks are target and determining which tests are able to cause failures. Afterwards, the tool determines which of these test cases are interesting, i.e., capable of uncovering new paths and causing a SUT failure.

CorCA (CORrection of C Automatically) is a tool that applies code correction to fix the vulnerable C/C++ code of an application, confirming the existence of the flaws and testing the new (fixed) code. The tool employs static analysis to find various types of buffer overflow (BO) vulnerabilities, attack injection techniques to confirm the vulnerability and validate the fixed code, and fix the code automatically with dynamically generated fixes. CorCA was tested with C/C++ code of a software product line from Bombardier Transportation (BT), a member of the Alstom Group, and the results showed that the code does not contain BOs, i.e., is safe with respect to this class of vulnerabilities.

Deep Learning (DL) model capable of classifying code excerpts of web application variants as vulnerable (or not) to SQL Injection. The model uses an intermediate language to represent the excerpts and interpret them as text, resorting to well-studied Natural Language Processing (NLP) techniques. This work can help back-end programmers discover SQL Injection in an early stage of the project, avoiding attacks that would eventually cost a lot to repair their damage [FMAN+20]. The DL network is composed of a minimum of five layers that work sequentially. It produces a final output, between 0 and 1, indicating the probability of the excerpt being vulnerable. It receives as input a numeric vector that goes sequentially through the Embedding, LSTM, Dropout, and two Dense layers, suffering successive transformations and producing the final output. We conducted experiments on four datasets of intermediate language with different excerpt representations. All datasets led to model with good performance, in which accuracy scored, on average, more than 60%.

4. Recommendation for homologation and certification

This chapter contains recommendations for the homologation and certification process of safety-critical multi-variant systems. It describes experiences gathered by members of the XIVT consortium with this topic, and gives a list of recommendations for assessors and engineers who are involved in the homologation process.

4.1 Experiences from the XIVT project

Within XIVT, we have applied our tools to a number of use cases. Since the goal of a research project is not to develop concrete marketable products, no assessment of products has been performed within the project. We did, however, advance several product line developments and prepared concrete products for homologation. In this section, we report on experiences we gathered within these activities, and their implications for the assessment process.

- The XIVT project was mainly about testing methods and tools for configurable and multi-variant systems. There are, of course, other quality assurance methods which need to be considered. In particular, we did not advance process assurance methods, as suggested, e.g., in CMMI or Automotive SPICE. When confronted with a (software) product line, an assessor must look also at the quality of the development processes (see next subsection).
- For product lines, it is often not feasible to manually create test suites. Therefore, some automated test generation framework is necessary. The approach of generating the test stimulus (test sequences) as well as the test evaluation scripts (assessments) saves a large amount of development time. Moreover, automatic generation of test projects based on requirements specification document makes it suitable for agile software development. However, while it is easy to create large test suites with automated test generation, it is important to check the test coverage (requirements and/or code coverage) which is achieved by the tool. Current test coverage goals from the single-product level (e.g., MC/DC code coverage) may not be sufficient for thoroughly testing variant-intensive software.
- Automatically generated test suites are significantly less costly than manually created test suites. However, automatic test generation may not be quite as effective as manual testing, in particular, if the test suite is well-constructed and comprehensive for one specific application. For a comparison of manually and automatically created test suites and testing efforts, confer the results of [WS 14]. An assessor should be cautious not to adopt the simplicity of use of automated test cases versus the potentially more effective coverage of manually designed test suites. Test generation complementing manual testing can provide early detection of faults and discrepancies in integration-level conformance for variant-intensive systems. Test generation would best complement manual testing in a continuous integration work flow. This would allow the early detection of some faults for new and changed models before functional tests have been created.
- When developing software or test cases from requirements, traceability links between the requirements and the software are necessary. We found that for reuse of product line artefacts in different product developments it is essential to have traceability at the right level of granularity. This way, the reuse recommendations are even more useful and might require fewer adaptations in the software under reuse. As the tools are only providing recommendations to the expert for their consideration without affecting the links or requirement text, the tool itself is not considered safety relevant and does not need any formal verification. However, an assessor must be aware that the recommendations rely on the granularity of the requirements

definition and do not constitute an “objective truth”. It does require a knowledgeable reviewer to look at the traceability and the recommendations derived from it.

- With the help of formalized requirements and feature models, different variants of a model (e.g., a Simulink model) for testing can be handled. While it is useful to have automated test generation from requirements, according to our experience it is still not possible to generate test stimuli for all sorts of formalized requirements. Therefore, an assessor should be cautious about using this technique in all situations. For example, if a trigger condition of a requirement contains the reaction of an output signal, then the current algorithm will not be able to calculate a solution. In such cases, it would be necessary to analyse the whole set of requirements to identify the connection between certain requirements.
- Model variants can be handled by using test case variation. With the help of parametrized variation lists, it is possible to generate a large number of test stimuli for such variants. These lists can be specified to test different values at the desired positions. The current algorithm supports the complete permutation of all variation lists as well as the synchronization of specific variation lists. However, this algorithm is for a specific (Simulink) model. An assessor should be cognoscente that a different model will need different permutation algorithms for handling its variants.
- One possible way to deal with the problem of testing a product line is to create separate test sets for each component and exercise a set of interactions between components just to test the interfaces. Similar to the procedure in module and integration testing, there is a test set for each component and each data interface. This can eliminate the need to test all combinations and reduce the risk for each individual product to an acceptable level. However, the assessment must check whether precautions for feature interactions and infeasible combinations have been made, and as well, check that the most frequent used combinations are tested in an end-to-end testing approach (if there is meaningful information to select the most frequent used ones).
- For data-intensive applications which are configured by large datasets from external providers (e.g., a routing app relying on external map data), the data might have gaps and other defects. The most common methodology in test case generation for the validation of features based on these data is based on performing a brute-force (exhaustive) search to select the test cases. These approaches are not efficient as they end up having a significant number of repeated test cases for validation and execution. We recommend the assessor looks at approaches that minimize the redundancy in the process of test case generation through utilizing intelligent route planners and focusing on testing variability that exists in the data.
- The FFT use case involves a test software for joining elements in a robotic production line. The aim is to follow up the joining elements defined virtually in the engineering phase and to determine deviations on the real production line. Carrying out the tests completely manually is extremely time and cost intensive; therefore, each test case is usually performed only once. Absent an automated virtual testing process, the assessor needs to be aware of the many complexities of testing the production line manually. However, the accuracy and capabilities of the simulation platform to faithfully represent the physical reality must be checked. This concerns not only the simulator as such, but also the specific setup used for the campaign. An important evaluation aspect in the assessment is the conformance of simulation results with actual physical facts.
- Simulation-based testing should be mandatory for safety-critical cyber-physical systems, as it can achieve a much higher test coverage than physical system-level testing, as well as a much higher precision than pure software-based testing. However, simulation-based testing does not replace physical testing, as not all properties of a cyber-physical system can sufficiently be replicated in a simulation with present-day technology. The assessment must confirm that the testing objectives are met by adequate testing means. Physical aspects must be tested on selected variants, and a large set of variants and test runs must be tested in the simulation.

4.2 A list of recommendations for assessors and safety engineers

With the shift from single products to variant and configurable systems, the focus of the assessment shifts **from the product view to the process view**. It is impossible to individually assess all members of a product family with, say, 10^{100} elements. However, it is very well possible to assess the engineering and development processes which lead to the existence of each individual product. Moreover, quality assurance for the artefacts on the product family level can be assessed.

The following checklist can help an assessor to evaluate multi-variant product lines.

- Is there a separate product line engineering process established, with separate budget, personnel, and management? Is there a separate product line management process, controlling the product line engineering and product derivation processes? Is the product derivation process inheriting methods and tools from the product line engineering process?
- Do there exist the following artefacts at the product line level:
    - description of base functionality
    - description of additional features and variability
    - reference architecture
    - documentation of the product line engineering process
    - documentation of the product derivation process
- Is there a separate quality assurance of generic artefacts? In particular, for re-usable modules, do there exists stubs and drivers to test them separately?
- Have the conditions for using a module in a product been formally defined? Are they automatically checked before a module is used?
- Is there an analysis of potential feature interactions, excluding certain products? Has there been an analysis of possible interferences of different modules (e.g., via shared memory)?
- Has it been made sure that unused features, dead code and safety-elements out of context do not interfere with other safety elements?
- Has the product line as such been tested? If so, what is the feature and module coverage? What is the strategy for selection of products to be tested?
    - Have the borderline products (with minimal/maximal number of features) been tested?
    - Is there a risk score, depending on the SIL-level, which determines which combinations are “relevant” and must be tested?
- How is the quality of the product line measured, how is the quality of the testing process measured? Are the quality goals given in advance, or is QA stopped when resources are exceeded?
    - Is there a numerical balance for test effort distribution, a test prioritization scheme?
- Are the products generated from product line artefacts in a well-defined, reproducible way? Is this product derivation process automated?
- Are there generic test cases which can be instantiated to different products? Is the execution of product tests automated, with test cases being automatically adapted to the particular variant?
    - Is there a simulation environment, in which products can be tested even if they are never built?
    - If so, has the simulation environment been shown to faithfully reflect the reality with respect to the safety features under consideration? Are simulation results reproducible?
    - Has the simulation environment abstraction capabilities, e.g., a simulation time which can be faster or slower than real-time?

- If certain artefacts of the product line are advanced, is there a change impact analysis ranging over all possible products? What are the prescriptions for impacted products? Which re-assessments are done for product line deltas?
- For systems configurable at the customer's site:
    - Are the product-line artefacts protected against side-effects from user-provided configurations and files?
    - Has cybersecurity according to IEC 62443 been taken into account? Are stakeholder roles well-defined? What are the maturity and security levels of the products? Has there been a vulnerability analysis, an assessment of risks and a consideration of damages which an intruder could do?
- For systems configured via AI (artificial intelligence) and ML (machine learning): Are data protection topics been cared for, is the GDPR (article 22, explainable AI) respected? Is the use of training data well-documented? Is there a quality assurance for training data?

A further research topic is how to deal with the situation of already certified components in a new product. To which extent can existing certificates be re-used? This depends on the extent of independence between the components, and the inclusion of new features in the components. As feature models of a software product line show the dependencies this may ease the task at hand.
In variant-intensive safety-critical systems, the safety requirements of the individual components in each product variant are not unique. Each individual component failure may cause different hazards event which results in having different SIL levels and consequently different safety requirement. Therefore, it is suggested to identify a cost-effective approach to certify variant-intensive components such each Individual component has a unique safety requirement.

5. Conclusion

In this report, we have summarized requirements of various standards for the homologation and certification of safety-critical software in different domains with respect to configurable and multi-variant systems. We have described methods and tools developed within the XIVT project to deal with these requirements. Finally, we have reported on our experiences, and from that, extracted recommendations for assessor and notified bodies how to deal with this subject.

What remains to be done is to define a "standard process" on how actual certification can be done. Since, again, current certification processes vary significantly according to region, domain, and safety integrity level, such a standard process itself should be defined in a configurable way. It should be possible to adapt it to different application scenarios and concrete actualities. For this, feature analysis and modelling techniques used within XIVT can be used. To contribute to the success of such a configurable certification process, some extensive case study should be conducted. Such a case study should follow the defined process in some selected variants. However, this is beyond the scope of the present report.

References

[AHH 20] Ali, Nazakat, Manzoor Hussain, and Jang-Eui Hong. "Analyzing Safety of Collaborative Cyber-Physical Systems Considering Variability." IEEE Access 8 (2020): 162701-162713.

[AJL+ 20] Abbas, M., Jongeling, R., Lindskog, C., Enoiu, E. P., Saadatmand, M., & Daniel, S. (2020). Product Line Adoption in Industry: An Experience Report from the Railway Domain. In 24th ACM International Systems and Software Product Line Conference SPLC 2020, 19 Oct 2020, Montreal, Canada (Vol. 164267, pp. 130-141).

[AMN 20] Francisco Araújo, Ibéria Medeiros, Nuno Neves. Generating Tests for the Discovery of Security Flaws in Product Variants. In Proceedings of the International Workshop on Testing Extra-Functional Properties and Quality Characteristics of Software Systems (ITEQS), Porto, Portugal, October 2020

[ASE+ 20] 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.

[Bre 20], Bressan, Lucas, et al. "An Integrated Approach to Support the Process-Based Certification of Variant-Intensive Systems." International Symposium on Model-Based Safety and Assessment. Springer, Cham, 2020.

[FMAN+20] Ana Fidalgo, Ibéria Medeiros, Paulo Antunes, Nuno Neves. Towards a Deep Learning Model for Vulnerability Detection on Web Application Variants. In Proceedings of the Workshop on Testing of Configurable and Multi-variant Systems (ToCaMS), Porto, Portugal, October 2020.

[IEEE 90] IEEE Computer Society (1990). "IEEE Standard Glossary of Software Engineering Terminology". IEEE Standard.

[ISO 11] ISO/IEC 25010:2011-03 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models

[ISO 14] ISO/IEC 25000, 2014-03, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guideline for SQuaRE, https://www.iso.org/standard/64764.html

[LM 21] 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.

[Oli 20] de Oliveira, André L. "Safety Analysis and Requirements Allocation for Software Product Lines.".http://easyconferences.eu/imbsa2020/wp-content/uploads/2020/09/Safety-Analysis-and-Requirements-Allocation-for-Software-Product-Lines.pdf

[WS 14] Weißleder, Stephan, and Holger Schlingloff. "An evaluation of model-based testing in embedded applications." In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*, pp. 223-232. IEEE, 2014.