Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments

Project Acronym:

EMC²

Grant agreement no: 621429

Deliverable no. and title	**D6.1 State of the Art for System Qualification and Certification, V&V Survey**	
Work package	WP6	System qualification and certification
Task / Use Case	T6.1	Certification and Qualification challenges of EMC²-Systems
Lead contractor	Infineon Technologies AG Dr. Werner Weber, mailto:werner.weber@infineon.com	
Deliverable responsible	AIT Austrian Institute of Technology GmbH Erwin Schoitsch, erwin.schoitsch@ait.ac.at	
Version number	v1.2	
Date	24/11/2014, extended V1.1 20/02/2015	
Status	Final	
Dissemination level	Public (PU)	

Authors

Partici-pant no.	Part. short name	Author name	Chapter(s)
02A	AVL	Eric Armengaud	Initial document structure
02G	AIT	Thomas Gruber	4; Publishable executive summary.
18B	IFXUK	Fabio Bruno	2.1
01O	FhG	Daniel Schneider	4.7, document review
01ZI	TUKL	Max Steiner	4.8
02G	AIT	Christoph Schmittner	2.1.2; 3.5 safety-related input to 3.; 6.1, document review
15F	TASE	Manuel Sanchez	2.2
01O	FhG	Tiago Amorim	4.7
04B	Freescale	David Baca	3.1
11A	Alenia	Renato Campaniello	2.4
02F	ViF	Helmut Martin	Document review
02G	AIT	Erwin Schoitsch	3.7, first ToC, document review and extension

Document history

Version	Date	Author name	Reason
0.1	16/06/2014	Eric Armengaud	Set-up document
0.2	02.07.2014	Eric Armengaud / Thomas Gruber	Input from AIT regarding V&V survey integrated
0.3	07/08/2014	Fabio Bruno	Input from Infineon on State-of-art
0.4	25.08.2014	Max Steiner	Input from TUKL and IESE
0.5	22.09.2014	Manuel Sanchez	Input from TASE on State-of-art
0.5	29.08.2014	Christoph Schmittner	Input from AIT added
0.6	24.11.2014	UTRC, AIT	Input from UTRC and more input from AIT; chapter structure changed; References changed to Word citations; appendix "Abbreviations" and chapters "Publishable executive summary" as well as "Conclusions" added.
0.65	2.12.2014	Helmut Martin	Review ViF, added
0.7	4.12.2014	Renato Campaniello	Input from Alenia Aermacchi added
0.7	4.12.2014	Daniel Schneider	Review IESE, added
0.8	20.12.2014	Erwin Schoitsch	General Review finished, 2.5 and 3.7 added
1.0	23.12.2014	Thomas Gruber	Final version
1.1	20.02.2015	Erwin Schoitsch	Extended final version after thorough ViF review
1.2	24.02.2015	Christoph Schmittner	Sorted references
	24.02.2015	Alfred Hoess	Final formatting and editing, deliverable submission

Executive summary

Multi- and many-core architectures pose specific challenges, in particular when they are used in safety-critical systems. Currently functional safety standards consider their use in safety-critical applications as problematic due to the difficulties regarding predictability. Therefore they provided no specific guidance how to use them and which precautions to take in order to minimize the specific risks. This document analyses first which specific risks may be encountered in general when using many- or multi-core architectures, and it describes then the areas which should be covered by such guidance, partly illustrated by examples:

- Processor configurations settings
- Interference channels of multi-core processors
- Shared cache/memory
- Dynamic allocation of cores or memory
- Error detection and handling
- Processor errata survey

Special emphasis is laid on challenges for safety and security in multi- and many-core architectures, which are systems of systems and - when interconnected - can be seen as Cyber-Physical-Systems (CPS).

Then the document gives an overview of current domain-specific practices for system qualification and certification for the automotive, avionics, space and smart grid domains, where processes are guided by different sets of well-introduced standards. A big potential is identified in the expected information on multi-/many-core issues in future standards extensions. As an example, IEC 61508, particularly IEC 61508-3 (software requirements), is preparing for Edition 3.0 because the extended stability phase ends 2016 but there are many new issues concerning methods, techniques and architectures arising, one of them is software architectures for multi-core/many-core systems. The combined safety & security issues addressed before are now discussed in a recently founded IEC TC65 Ad-hoc Group 1 on “Framework for coordination of safety & security” (both activities started October/November 2014).

The following V&V (verification and validation) survey provides a classification of verification and validation methods, and emphasis is laid on safety- and security-related methods, among them also risk analysis methods. IEC 31010:2009 provides descriptions of 31 risk assessment techniques, some are mentioned explicitly in this document, in particular FMEA, HAZOP and FTA, which are described in detail in IEC 60812, IEC 61882 and 61025, respectively.

Functional safety standards, for instance the generic IEC 61508, but also derived standards from IEC, ISO or CENELEC provide a lot of guidance on V&V methods. More than 20 testing techniques are recommended depending on safety criticality in these standards. For the particular case of multi- or many-core systems, many of the (conventional) test methods can be applied, although mostly at higher complexity level than with single-core architectures. For safety qualification/certification, extensive use or multi-core/many-core potential is rather limited particularly because of the predictability requirements.

For security validation, several standards are considered: IEEE 1012 Standard for Software Verification and Validation, Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408), FDA General Principles of Software Validation, FIPS 140-2, and the Microsoft Security Development Lifecycle (MSDL). The deliverable describes shortly numerous methods, frameworks and standards for security risk modelling and analysis. Considering the high complexity as well as the additional potential problems imposed by multi- and many-core architectures, many of the methods need to be adapted before they can be applied accordingly.

Nevertheless, it has to be emphasized that all security-related assurance techniques need to consider the inter-relation between safety- and security-oriented aspects and architectural elements of a system. Safety and security goals need to be aligned and weighted. This combined view on safety and security is today mainly still a research topic, but it is finding its way into standards groups and into industry. A few already published approaches are shortly described in the document.

Further, attention is dedicated to robustness testing and to runtime assurance. These techniques play an important role in particular for multi- and many-core systems, and mobile autonomous systems. Finally, virtual evaluation and testing is considered, which is developed in several research projects. Due to the complexity of multi-core systems, considerable benefits can be expected from using these techniques,

Table of contents

List of Figures

List of Tables

1. Introduction

1.1 Objective and scope of the document

Scope of this document is to summarize the state of the art with respect to system qualification and certification.

Regarding the V&V survey, the description comprises all methods adequate for verifying requirements and for validation and respective tools; this includes (without claiming to be exhaustive) inspection of artefacts, static analysis, dynamic testing, model-checking, and semi-formal as well as formal methods. A special focus is laid on performance and robustness testing, and the challenges in testing and verifying multi- and many-core systems are explained.

Note that the following EMC² deliverables are related to this report:

- *D6.2 Requirements elicitation*: This deliverable shall provide an overview of the requirements coming from the living labs and therefore consolidate this deliverable from the user needs point of view

- *D6.3 Preliminary definition of the runtime certificate approach*: This deliverable shall provide first direction for runtime certification, based on this report and on the deliverable D6.2

2. Safety and Security Challenges in Qualification and Certification

Using multi-core processors in safety-critical systems can be regarded as a challenge as well as an opportunity. Considering multi-core as an opportunity stems from the additional level of redundancy and independence that is added by multiple cores. On the other hand, multi-core technology provides a safety-challenge since it allows for “real parallelism”, which hampers guaranteeing synchronism and determinism.

In the past, multi- and many-core architectures were considered inadequate for safety-critical applications as predictability cannot be guaranteed. Industry has actually little experience with certification of multi-core based real time software. Then, the concern is that using the existing guidance as defined for instance in DO-254, DO-178C and ARP4754A for the avionics domain may not provide enough assurance that final product would comply with its requirements when executing in parallel with the mechanisms of the multi-core processor. Therefore it is considered that some additional guidance resulting in additional requirements is necessary regarding the multi-core based products certification in avionics and other domains. The main topics to be addressed by this guidance are presented hereafter:

Processor configurations settings

Multi-core processors have many features and options that are configurable through registers or pins. Inappropriate settings could significantly change the way that the processor behaves so as to alter its outputs in an undesirable manner.

Configuration settings may also be used to deactivate some of the features of multi-core processors that could produce non-deterministic behavior, or to activate features that should not be active during the normal operation of the multi-core processor, such as debug features or uncontrolled dynamic features.

Multi-core processors contain more features than single core processors, some of the features of multi-core processors can cause unwanted interference between the software hosted on the cores, and some of the dynamic functionality such as the ability to switch off a core while leaving one operating was not available until the introduction of multi-core processors.

These features have generally been incorporated for use in installations that are not safety critical.

Another point is that configuration settings could be inadvertently altered by single event upsets or other problems.

Interference channels of multi-core processors

Applications running on different cores of a multi-core processor do not typically execute independently from each other because the cores are sharing resources. Even if there is no explicit data or control flow between these applications, coupling exists on the platform level since they are implicitly sharing platform resources (e.g. multi-core shared cache/memory, peripherals, interconnect mechanism) and this could result in interference between these applications.

Interconnect features may result in incorrect transactions or in jitter in data arrival times.

If the overall available resources of the multi-core processor are exceeded by the combined resource demands of the separate software programs, this could result in a non-predicted behavior of the applications hosted on the different cores.

Shared cache/memory

Some multi-core processors have areas of memory or cache memory that are shared between the processing cores or that may be allocated so that they are shared.

The use of shared cache greatly influences the worst-case execution time (WCET) of the software applications hosted on one core of a multi-core processor. This WCET could significantly increase in case of repeated cache accesses by the processes hosted on the other core, leading to repeated cache misses.

The use of shared memory can lead to situations in which the software hosted on one of the cores may be locked out from accessing the shared memory locations. Such problems can cause applications to be unable to obtain the data they need at the time when they need it and could even result in an application halting due to memory access problems.

Dynamic allocation of cores or memory

In existing certified systems, each software application is statically allocated to execute on a designated core within designated memory boundaries for their code and variables as a result of software integration. The use of dynamic software features has been deliberately restricted so as to provide a predictable and deterministic environment for the execution of safety-critical software.

With Symmetric Multi-Processing (SMP), processing threads can be dynamically reallocated by the processor or its single operating system during operation. Neither the existing guidance nor the existing industry practices for deterministic, safety-critical software cover the development, integration or verification of threads that are controlled in this dynamic manner.

Error detection and handling

Features of a multi-core processor may contain unintended functionality that may cause errors and produce behavior that is not normally seen with single core processors. Therefore additional more complex mechanisms (Safety Nets, providing protective measures against such failures) have to be developed and verified to detect and handle these specific errors associated with these features.

Processor errata survey

Most commercial multi-core processors were not designed for safety applications.

If the user does not have access to the complete list of errata for its selected multi-core processor then he might not be aware of known faults and would not, therefore, employ the necessary fault mitigation in his installation.

2.1 Challenges for multi-/many-core systems

In the course of the ARAMiS project, a classification of the various safety-related challenge caused by the use of multi-core processors was made [1] [2]. There are two main challenge classes:

Segregation challenges due to unintentional interactions between software components which complicate temporal predictability and the segregation of mixed-critical systems and synchronization challenges emerging from the “real” parallelism introduced by multi-core processors.

2.1.1 Segregation challenges

The goal of segregation or partitioning mechanisms is to allow the integration of mixed criticality software components on the same hardware platform. To this end, interferences between different software components via shared resources must be avoided, especially in case a resource user fails and over-utilizes or misuses the resource. Corresponding requirements are found both in aviation and automotive-related standards [ISO 01, RTCA 01].

The interference scenarios classified in [2] are structured according to the shared resource that is affected by the interference scenario:

- Cache
- CPU
- Memory Interconnect
- Shared Memory
- Peripheral
- Interrupts
- Scheduling

A special focus of the analysis lies on interferences that influence the timing behavior of the affected software component. Additional material and a good overview on temporal interference can be found in [3].

Example: Shared cache-related challenges

Cache hierarchies with more than one level create several challenges that have to be dealt with independent of the number of cores. But multi-core systems create new challenges when caches are shared among different cores (see Figure 1).

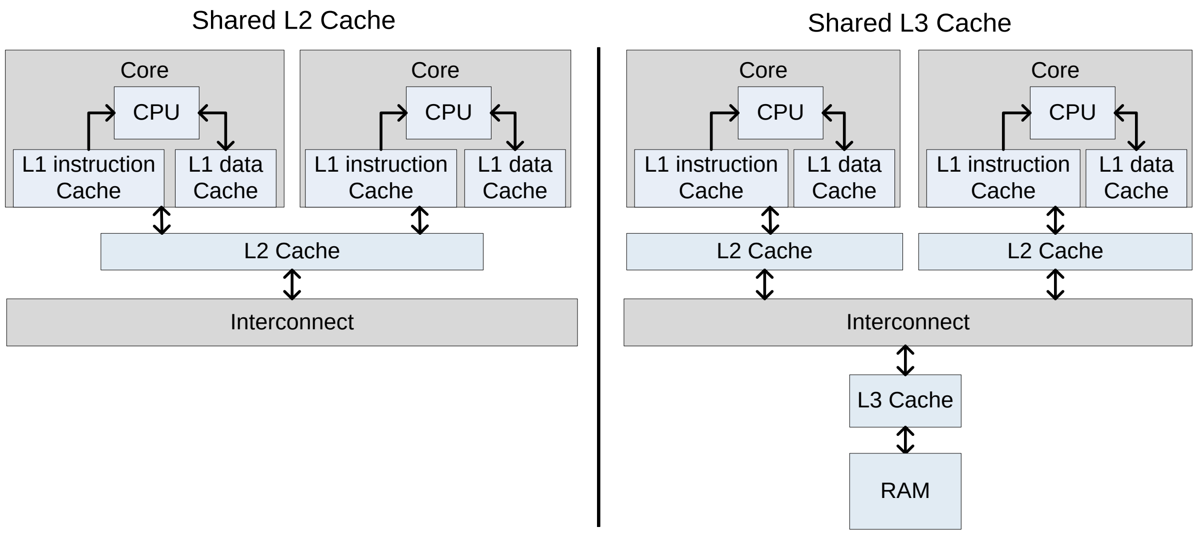

Figure 1: Multi-core system with shared level 2 cache (left) and with separate level 2 caches (right)

Assume two tasks running on two different cores which access the same memory region. Then the first task might evict a line from a shared cache when accessing the memory. As a consequence, there is the chance that the second task running on another core experiences a cache miss. As a worst case, the two tasks might repeatedly evict the shared cache line, leading to alternating cache misses.

Whether an eviction of a lower level cache line directly affects the local L1 cache of another core depends on whether the caches are implemented inclusive or exclusive. In an inclusive cache, all data stored in the L1 cache must also be stored somewhere in the L2 cache at any time. However, if the caches are exclusive, an eviction of a L2 cache line does not cause an eviction of a L1 cache line.

If the different cores use private caches there are other challenges, because in this case the different caches have to be coherent. This leads to additional overhead and, consequently, potential timing problems.

2.1.2 Synchronization challenges

Some single-core synchronization techniques do not work anymore in multi-core architectures. Others like semaphores would still work, but produce too much overhead. An alternative for multi-core systems are *spinlocks*. A spinlock is based on an atomic test-and-set instruction that must be provided by the microarchitecture of the processor.

If the lock is currently taken, accessing a spinlock causes a process to busy-wait for the other process to release the lock. A switch to a wait-state and a context switch to the other process are not performed. If the processes that want to access the same resource run on different cores, it is indeed faster to let one core busy-wait a view cycles than to perform a complete context switch.

The necessity of additional synchronization mechanisms leads to problems, especially when legacy code is ported to multi-core platforms. Furthermore, mechanisms that prevent deadlock or priority inversion need to be capable of handling the mixed usage of traditional and multi-core specific synchronization mechanisms.

When different processes access a shared resource and they are executed concurrently on different cores, a multi-core safe implementation for critical sections has to be used. This is especially important to consider when porting existing legacy code from a single to multi-core systems.

2.1.3 Multi-/Many-core Challenges for Safety and Security

Connected system of systems

In the past critical systems were stand-alone units without outside connectivity. With multi-/many-cores connectivity transforms embedded systems into system of systems, in which they are connected with other systems and infrastructure. They become Cyber-Physical-Systems (CPS). CPS rely on seamless data exchange and information flow. The implication of connectivity is beyond the system. The increased connectivity and interaction gives rise to new hazards. Hazards and their causes, faults and vulnerabilities are no longer restricted within a single system. Due to connectivity, vulnerabilities and faults from a single system can propagate further, leading to hazards affecting multiple levels at once. An attacker might exploit vulnerabilities or tamper data, causing hazards on multi-system level. The change of the infrastructure from a passive system to an active system makes it susceptible to security threats which lead to safety hazards. For example, an attack on a control system might cause hazards in multiple connected systems. Since CPS form a connected system of systems, safety and security must be ensured at the sub-system level as well as the systems combined [4].

Openness and increased attack surface

Security in an open system is a new factor to be considered in system engineering and safety analysis. Open systems have increased attack surface, allowing misuse and manipulation of systems. From a safety engineering point of view, security breaches are new causes for existing hazards. With an open system we cannot regard a system to be safe unless the security of the system is assessed and assured.

Privacy and its conflict to safety and security

Connected and cooperative systems generate and exchange a large amount of data. While tampering with these data could cause hazards, eavesdropping on the communication or unauthorized access to stored data could breach a user's privacy. An arising challenge is the conflict between safety and security [5]. To ensure safety, systems need to make the information available. On the one hand, for security reasons, certain identity information needs to be included in the exchanged data, enabling verification and ensuring data integrity. Connected and cooperative systems will have more conflicts between safety and security requirements, which demand an integrated approach for solving these conflicts.

2.2 System Qualification and Certification - State of the Art in the Automotive Domain

Qualification of safety system is dictated by ISO 26262 [ISO 01]. The safety concept is the document that specifies the safety features to be implemented in order to achieve a given ASIL (Automotive Safety Integrity Level). Failure modes of a device are associated to either hardware or software safety features in the FMEDA (Failure Mode Effects and Diagnostic Analysis), directly derived from the safety concept. The fault injection campaign is the main way to prove the FMEDA claims and the target diagnostic coverage (DC). Functional verification tests plus some more ad hock tests are used as stimuli in the fault insertion campaign, where one fault at the time is inserted and a simulation is performed to make sure the relevant safety feature detects the inserted fault. Results from the safety qualification campaign are finally back-annotated into the FMEDA and deviation, if any, are justified in the safety assessment report.

Table D.1 of ISO 2626 chapter 5 specifies the type of faults that need to be inserted, in relation to the type of logic and the targeted ASIL; for high level of DC (e.g. 99% for ASIL D) the direct current fault model is mandated by the standard. Direct current fault model consists of **static faults** (faults that would always be present in silicon), such as stuck-at, stuck-open, open and short and **soft errors** (for sequential logic) resulting from transient faults (these faults will only be visible for a limited amount of time).

<u>The main challenges of today's safety qualification are:</u>

The late start of the Safety qualification activity

The ISO standard dictates that the fault qualification campaign needs to be performed in the final, post layout, net-list. This forces the qualification stage to be the last stage of the project development, with a direct impact to the time-to-market and especially too late to be able to implement any hardware adjustment that may be needed. Despite precise methods and flows (such as ASIL decomposition) can be used to reduce the risk of late deficiencies discovery, the risk of having to introduce additional software safety mechanisms and application assumptions of use, in order to achieve the target diagnostic coverage (DC), is still very high.

The identification of the safety relevant logic (or safety subsystem):

Not all logic is safety relevant and the non-safety-relevant portion can be excluded from safety qualification, but this is not straightforward, because a mapping of logical functions into physical nets needs to be performed and the complexity of this task depends on the abstraction level selected for the safety qualification (post layout, gate level or net list as currently mandated by the standard).

The modelling of the Direct Current faults:

Only stuck-at faults can be considered mainstream and easily implemented, by forcing the selected net to 0 and 1, for all other types of faults, although a smaller sample can be selected, the distribution and the model is non-trivial. For instance for shorts (also known as bridging faults) the subset of nets can be derived by selecting pairs of nets using capacitance and proximity information from layout tools, then the faults need to be modelled (and there are multiple possibilities) and the simulations performed to prove that the faults would be detected by the safety mechanisms. Alternatively a more empiric method, called N-Detect [6] [7], can be used to derive the final DC for the direct current fault model, based on the stuck-at DC. In fact, according to the N-Detect method, the probability to detect direct current faults is directly correlated to the number of times (N) each stuck-at fault can be detected by running a different simulation. If the average N for all stuck-at faults is greater than six, the probability to detect bridging, stuck-open or open faults is 99% of the measured stuck-at DC. Challenge of using this method is in selecting multiple simulations that can activate the stuck-at fault in different ways and also the explosion of simulation time.

2.3 System qualification and certification state of art for space activities

The European Cooperation for Space Standardization (ECSS) is a cooperative effort of the European Space Agency, national space agencies and European industry associations for the development of a coherent, single set of user-friendly standards for use in all European space activities. The result of this effort is the ECSS series of standards (ST), handbooks (HB) and Technical Memoranda (TM) which are organized in four branches, which is high-lightened in Figure 2:

- M: Space project management;
- Q: Space product assurance;
- E: Space engineering;
- U: Space sustainability

The main software standard is ECSS-E-ST-40 [ECSS 01], part of the ECSS engineering branch (E). It covers all aspects of space software engineering, from requirements definition to retirement. It defines the scope of the space software engineering processes, including details of the verification and validation processes, and their interfaces with management and product assurance, which are addressed in the management (M) and product assurance (Q) branches of the ECSS system.

ECSS-E-ST-40 refers the ECSS-Q-ST-80 [ECSS 02] for the Software Product Assurance requirements related to the development and maintenance of software for Space Systems. Both apply to any software project. The ECSS-E-ST-40 provides a process model for the SW development activities, without prescribing a particular software life cycle.

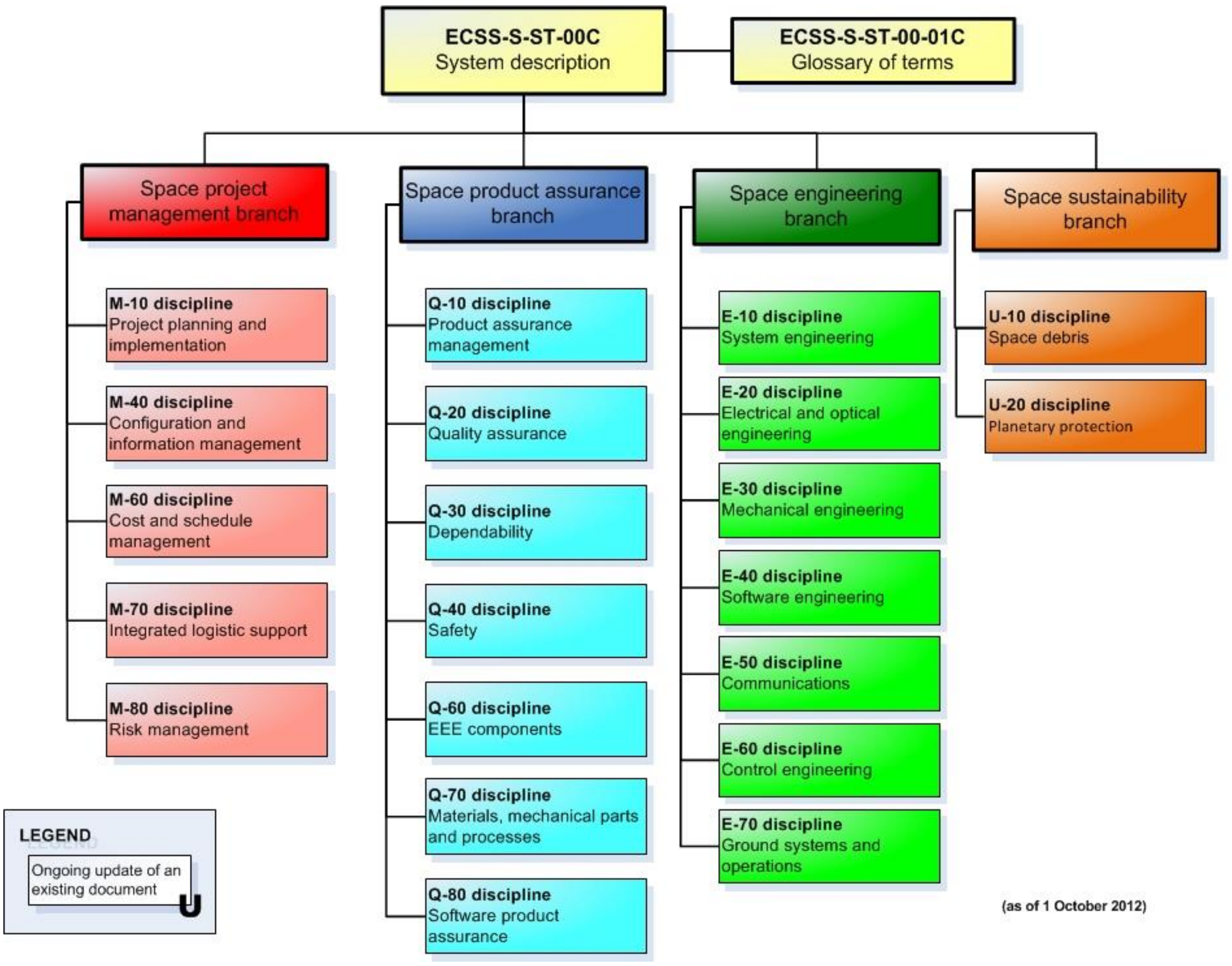

Figure 2: ECSS Architecture (from ECSS website)

2.3.1 High level process

One of the fundamental principles of the ECSS standard series, and distinctive difference compared to the standards from the other domains, is the explicit customer supplier relationship, assumed for all system and software developments, where the supplier demonstrates compliance with the customer requirements and provides the specified evidence of compliance. How and which parts of the ECSS must be applied is specified through contract, in a way that depends on the given mission.

In the ECSS the development of the software is always related to a complete space project and its different phases, with a strong focus on the integration with system level activities.

ECSS is both a process-based and product- based framework, in fact, ECSS is based on “processes”, and lets at user’s choice an own life-cycle and approach, with appropriate methods and tools.

ECSS has the peculiarity of allowing the tailoring, on project-basis. Tailoring means that on a project-basis, and hierarchically in the chain, each customer may determine the applicable ECSS standards and requirements therein, for his own suppliers. This tailoring must be justified, coherent, and consistent throughout the customer-supplier chain, and always be visible to the higher level customer.

2.4 System Qualification and Certification - State of the Art for Avionics

2.4.1 Introduction

In the sensitive domain of safety critical system certification, systems designed for air and rail transport (rail see chapter 2.5) present major technical challenges. These challenges arise from the extensive regulation of these domains due to the potential for catastrophic loss, both in terms of physical damage and human life, from any fault.

The “classical” certification basis JAR/FAR-25/EASACS-25 and the well-known certification guidance and development standard documents ED-14D/DO-160D, ED-12B/DO-178B, ED-80/DO-254, can be used as reference in the frame of the EMC2 effort on multi-core technology (see Table 1). The reason why DO-297 is mentioned herein is the involvement of the multi-core technology in the scope of IMA system architectures. However, the position with respect to ED-124/DO-297 needs to be clarified (e.g. the usage of ED-124/DO-297 as an applicable document may increase the workload due to overlay with other applicable documents).

Certification basis	**RTCA/EuroCAE Documents**	**Additional Material**
JAR-25	DO-160D/ED-14D	ARP-4754
FAR-25	DO-178C/ED-12C	ARP-4761
EASA CS-25	DO-254/ED-80	
	DO-297/ED-124	

Table 1: Documents in RTCA/EuroCAE context of airworthiness certification

The promise of model based certification and formal verification to simultaneously improve test coverage while reducing the cost of design is driving aerospace companies to investigate such technologies through avionics standards such as DO-333 [RTCA 02], a supplement to DO-178C and DO-278A. The growing complexity of modern electronic systems on aircraft increasingly conflicts with the requirements of certification – avionic engineers can spend upwards of 10 times more effort in verifying their products than designing them. To this end, with respect to safety assurance of a multi-core motor actuator, research activities will have to focus on the certification of safe and efficient motor management strategy and its low level control devices.

2.4.2 Multi-core challenges for avionics systems

Existing and future avionics research and development efforts face great challenges in delivering to market, as qualification and certification requires increasingly onerous effort and cost [8]. System verification and certification are the main problems to be solved [9] with respect to compliance with a variety of EASA, FAA, and other avionics regulations [10], [11], [RTCA 01].

EASA recently published the report from MULCORS project [12]. This study has been carried out for EASA by an external organization and expresses the opinion of the organization undertaking the study. It would be of interest to analyze this report and, possibly, create a broader industrial consensus about the contents based on opinions of EMC2 members.

Certification authorities are not interested in performance but their judgment is mainly oriented to the reliability of the HW. Multi-core technology in avionics context and the associated processors have been considered in the environment of RTCA DO 254 “Highly Complex Electronics”. This definition, based on EASA papers, declares the difficulties to certify this typology of items [13].

Even if multi-core processor technical aspects have been covered from different researches and working teams [14] [15] [16] [17], the certification aspects for the introduction of this technology would require investigation.

2.4.3 The multi-core motor actuator example

In the current motor engineer life cycle, embedded systems engineers, architects and developers need to begin the testing, verification and validation of their prototypes early in the product life cycle design. Late discovery of errors, e.g. in the implementation or the testing phases, often causes serious delays in the certification process. Further to this, the management of the growing complexity of modern avionics systems is dependent not only on their degree of criticality but also in terms of their expected functional performance. This highlights the need for rigorous and robust verification techniques that will successfully manage the complexity of system, especially related to multi-core architectures. In particular in the motor actuation case, hardware control devices that perform the vital power control algorithms for the motor, have to be compliant with specific avionics regulations that refer to low voltage switchgear. Capturing current experiences in the motor actuator protection and final certification in the aerospace domain is summarized as follows:

- Aerospace systems require software certification following the guidelines of DO-178C [RTCA 01] (software) and DO-254 [RTCA 03]) (hardware) in order to manage Safety Integrity Levels (SILs) activities and objectives during the implementation. In the multi-core motor actuator case additional assurance evidence should be provided for the flawless operation of the motor, e.g. when executing actions concurrently for low level motor control and high level prognostics.
- Current verification processes focus on test coverage of functional requirements, through verification of traceability of each high-level requirement to it(s) low level implementation.
- On-going research in many academic and government institutions has been addressing many parts of the desired future state, but these have seen limited use in the commercial sector.

2.5 System Qualification and Certification - State of Art for Railways

In the railway domain, system qualification and certification by legal authorities and/or notified bodies (e.g. arsenal RACE in Austria) is mandatory, comparable to the certification approach in avionics. Certification is primarily guided by four standards, including RAMS, Software, Safety-Case and safety-related communication systems, until now only considering safety (which has to be argued for the certification authorities). Security issues to be included in future editions of the standards are under development, particularly driven by VDE/DKE in Germany.

- EN 50126 Railway Applications – the specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS):
  - Part 1 – Basic Requirements and generic process (CENELEC 1999, Corr. 2010)
  - Part 2: - Guide to the application of EN 50126-1 for safety (informative) (2007)
- EN 50128 Railway Application – Communication, signalling and processing systems – Software for railway control and protection systems (CENELEC June 2011)
- EN 50129 Railway applications - Communication, signalling and processing systems – Safety related electronic systems for signaling (CENELEC 2003, corrigenda 2010)
- EN 50159 Railway applications - Communication, signalling and processing systems. Safety-related communication in transmission systems (CENELEC 2010).

The standard EN 50126 covers the railway system in total and addresses risk analysis, reliability and safety issues. The standards EN 50128 and EN 50129 are more specific. EN 50129 describes in detail the contents of a safety case. EN 50128 covers all critical software-related issues in a satisfactory and comprehensive manner, providing very detailed guidance what to do in each of the Safety Life Cycle phases, listing requirements, documents etc. It follows the SIL safety integrity level concept of IEC

61508, but starting with SIL 0, which is rather software quality management according to EN ISO 9001. EN 50159 focusses on safety-related communication and transmission systems

EN 50129 defines the requirements for the Safety Case. It provides details for safety related HW and SW for railway control and protection systems and for the overall system. The structure of such a Safety Case is defined in detail. This detailed description of how to achieve safety approval with the Safety Case makes this standard different from IEC 61508 which does not require a safety case. This approach supports very effective the developer as well as the assessor/licensing authority. Additionally, EN 50129 gives a clear description of the different levels of safety cases that could be developed:

- Generic product safety case (independent of application): A generic product can be re-used for different independent applications. It is one of the building blocks for the applications.
- Generic application safety case: A generic application can be re-used for a class/type of application with common functions. It is configurable, but is not particularly configured.
- Specific application safety case: is used for only one particular installation.

Some more details, particularly covering the interdependences between safety and security issues, are discussed in EMC² D6.2, Requirements Elicitation.

2.6 Multi-core Challenges for Smart Grid Systems

A smart grid puts information and communication technology into electricity generation, distribution and consumption, making systems safer, more reliable and efficient. These systems include smart networks that integrate remote terminal units (RTU) for data acquisition and substations automation. A RTU is a modular and networked embedded system that usually includes a CPU. They are safety critical electronic systems that should comply with safety smart-grid standards [IEC 06], such as IEC 61508 [IEC 04] and others.

System certification is normally supported by modular and component based safety analysis techniques [18]. During the last years, the RTU hardware is evolving to multiple core architectures and the challenge is to extent current safety analysis techniques to many-/multi-core platforms without safety-oriented features (such as the Freescale MPC564xL Family[1]). Additionally, several frameworks have been developed to support embedded-systems design tool chains [19]. They are mainly oriented to model-based software development and the challenge is to extent these tools to HW/SW safety analysis and validation.

[1] http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MCP564xL

3. Verification & Validation Method Survey

3.1 General Verification Methods

3.1.1 Static Verification

Static Methods do not require an execution of the system or model. They are based on the structure of the system or on the data and control flow. The structure of the system, data and control or energy flow is analyzed.

3.1.1.1 Formal Verification

A formal verification is being conducted using formal methods of mathematics in order to verify the correctness of a system. Formal verification can be conducted for both hardware and software [20]. One general approach for formal verification is model checking. Here, a mathematical model is used to verify a system in all its possible states. Notable implementation techniques are abstract interpretation, symbolic simulation or state space enumeration. Model checking is often fully automatic, but generally it does not scale well to large systems. Deductive verification is another approach to formal verification. Here, a collection of mathematical proof obligations is generated from the specifications, which are then being tested by a (interactive or automatic) verification system. This approach requires the user to understand in detail why the system works correctly.

3.1.1.2 Measurement techniques

Measurement techniques cover all techniques that measure properties of a system such as error proneness, understandability, and well-structured-ness [20]. Measurements can help to gain a better understanding if a component or system is fit for use. Furthermore, a high error proneness coupled with a low well-structured-ness can indicate future security problems.

3.1.2 Dynamic Verification

Dynamic Verification is also known as test or experimentation. This type of testing involves the execution of a system or its components [20]. There are numerous methods available; as an example for embedded software test methods, the book Testing Embedded Software [21] gives a good overview.

3.1.2.1 Software Testing

Dynamic testing methods cover all tests and experiments that can be performed to test of a system conform to the specification. Software testing can be categorized into three different categories, depending on their scale and extend. While software testing is not per se a security verification method, it can help to find bugs and software errors that could be exploited in a deployed system.

The most basic category, unit testing, is a method for testing software by which the individual parts, or units, of software are tested in short, independent and self-contained test cases in order to ensure that code meets its design and works as intended. The units are tested together with their associated control data, usage procedures, and operating procedures. Units can be a whole module (in procedural programming), an entire interface (in object-oriented programming) or an individual function. To ensure the independent testing of single modules, techniques like mock objects, method stubs, test harnesses of fakes are often used. Unit tests are usually written and run directly by software developers and are considered best-practice for software projects (test-driven development). The next step after unit testing is integration testing, where individual software modules are combined into aggregates and tested as a group. An integration test plan can help to keep track of different test cases. Purpose of integration tests is the verification of performance, reliability, and functional requirements. The last step in this testing process is system testing, where tests are being conducted on a complete and integrated system.

3.1.2.2 Acceptance Testing

Acceptance testing aims to determine if the requirements of a specification are met. Functional tests verify all functions of a project against design documents or specifications. Non-functional tests are being conducted if a product follows its non-functional requirements. Examples for non-functional testing are stress tests, load tests or performance tests.

3.2 General Security Validation Methods

Important validation methods, apart from those described in Section 3.4, are security specific validation tests, like penetration testing or fuzz testing. Security testing aims to find flaws in software systems that have an impact on confidentiality, integrity, authentication, availability, authorization or non-repudiation.

To assure that a system follows the relevant security standards, a security review can be conducted. It usually consists of a gap analysis. During a security review, code audits or reviews of design and architectural documents are being conducted.

A different approach for security testing is performing a security assessment on a given system, which aims to give a broad overview of the system. A security assessment usually consists of a discovery phase, where all relevant systems and services (and if possible their versions) are being identified. The next step is a vulnerability scan, which looks for known security vulnerabilities by using automated tools. Within a vulnerability assessment, these results are then analyzed in the context of the tested environment. In the last step, a manual verification of the previously found vulnerabilities is then conducted. The vulnerabilities are not being exploited in this step, but verified by checking system settings or examining log data, service responses, etc. Penetration testing is building on the information generated through a security assessment. Here, an attack to a system is being simulated by exploiting the found vulnerabilities in order to gain access to the system.

Fuzz testing, or fuzzing, is another approach to testing a system. It can be used to find flaws in an implementation, by conveying invalid, unexpected, or random data to a software system. Fuzzing can help to detect security relevant bugs, such as assertion failures or memory leaks, thus helping to improve both safety and security of a software system. Fuzzing is usually done in an automatic fashion. Input data can either be completely random, based on modified existing input data, or based on a model of the specifications of the input data.

3.3 Safety-related V&V methods

Risk-oriented analysis methods as a subset of V&V methods for hardware and software include a great variety of methods of different granularity for the assessment of mainly safety-critical systems. These methods, which are usually related to system safety management, need to be clearly distinguished from project risk analysis methods, which are common in project risk management, and which deal with commercial aspects like project delays, increased cost or customer dissatisfaction, which shall be detected early and mitigated by planning appropriate counter measures.

Risk-related quality attributes can be subsumed under the term "dependability" [22]. Although Dependability includes Reliability, Availability, Maintainability, Safety and Security (RAMSS) there is a distinction between analysis methods for RAMS (Reliability, Availability, Maintainability and Safety) and methods for Security.

Basically, technical risk analysis methods have the goal

- to assess one or several of the mentioned risk related quality attributes of systems, subsystems, components or even processes with respect to causes and consequences
- to identify existing risk reduction measures and to propose additional ones if necessary

They can be performed as qualitative or as quantitative methods. Quantitative analyses use empirical data about reliability and repair time, a qualitative approach is based on risk estimations of the experts in the analysis team.

The technical risk analysis is performed by a team of experts, which shall comprise members of the various stakeholder groups: Project manager, architect, designer, developer (hardware and software), test engineer, reliability expert, quality manager, safety manager (for safety-critical projects), risk manager for commercial risks, maintenance manager, production/process engineer, external expert and moderator. It is recommended on the one hand to have a sufficient selection of representatives of the mentioned groups, on the other not to overdo it and thereby cause time-consuming discussions with accordingly high cost.

Risk analysis can be performed based on tables like for instance HAZOP sheets or FMECA sheets; in former times (and partly even today), these were physical paper sheets with empty tables on them, and nowadays mostly spreadsheet tools like MS-Excel or other commercial tools are in use. RAMS analysis is also supported by commercial products of different vendors and even some free-ware tools.

Using a tool for analysis is not in all cases helpful. Analysis is often easier and more efficient with a spreadsheet because the input interface based on traditional dialogue boxes is rather clumsy to handle, and copying analysis parts of components already treated to the new ones to be analyzed is hardly possible between dialogue boxes but very easy in a spreadsheet program or even in a MS-Word table.

Nevertheless, tools offer certain advantages, e.g. by directly importing the FMECA results into the FTA (Fault Tree Analysis) data, by directly linking from a failure mode of a component to the associated failure model based on component reliability data catalogues, or by offering facilities for calculating the minimal cut sets in a FTA.

In any case, the analysis itself is conducted in the form of (usually several) sessions of the analysis team, where each item of the (sub) system is analyzed with respect to possible risks or failures. For each failure or malfunction, the analysis systematically covers causes, local and global consequences, probability of occurrence, degree of hazard, possibilities for detection, and existing measures for risk reduction. From all that follows the resulting risk and which additional risk reduction is necessary.

3.3.1 Standards related to Functional Safety V&V

IEC 31010:2009: Risk management — Risk assessment techniques

IEC 31010 [IEC 01] includes an in-depth presentation of 31 techniques for risk assessments and explains the process, benefits and disadvantages for each method.

Some of the most important methods for RAMS analyses which are presented are the following:

- PHA or PrHA - Preliminary Hazard Analysis
- PHA - Process Hazard Analysis (unfortunately the same acronym)
- HAZOP - HAZard and Operability study

- FMEA - Failure Modes and Effects Analysis
- FMECA - Failure Modes, Effects and Criticality Analysis
- FMEDA - Failure Modes, Effects and Diagnostic Analysis
- FTA - Fault Tree Analysis
- ETA - Event Tree Analysis
- LOPA - Layer Of Protection Analysis
- DFM - Double Failure Matrix

IEC 60812: Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA)

IEC 60812 [IEC 02] is a profound guideline towards the application of Failure Modes and Effects analysis. FMEA is used to identify potential failure modes, determine their effect on the operations of the product, and identify actions to mitigate the failures. FMEA is a bottom-up, inductive analytical method which may be performed at either the functional or piece-part level.

In addition the standard includes FMECA and FMEDA.

FMECA extends FMEA by including a criticality analysis, which is used to chart the probability of failure modes against the severity of their consequences. The result highlights failure modes with relatively high probability and severity of consequences, allowing remedial effort to be directed where it will produce the greatest value.

FMEDA is an FMEA extension, which combines standard FMEA techniques with extensions to identify online diagnostics techniques and the failure modes relevant to safety instrumented system design. It is a technique recommended to distinguish failure rates of different categories (safe detected, safe undetected, dangerous detected, dangerous undetected, fail high, fail low) in the safety models.

IEC 61882: Hazard and operability studies (HAZOP studies)

The Hazard and Operability Analysis (HAZOP) [IEC 05] is a structured and systematic examination of a planned or existing process or operation in order to identify and evaluate problems that may represent risks to personnel or equipment, or prevent efficient operation. The HAZOP technique was initially developed to analyse chemical process systems, but has later been extended to other types of systems and also to complex operations and to software systems. A HAZOP is a qualitative technique based on guidewords; and it is carried out by a multi-disciplinary team (HAZOP team) during a set of meetings.

IEC 61025: Fault Tree Analysis

FTA [IEC 03] is a deductive technique that focuses on one particular accident or main system failure, and provides a method for determining causes of that so-called "top event". It uses Boolean logic to combine a series of lower-level events which, in combination, cause the top event.

IEC 61508:2010 Functional safety of electrical/electronic/ programmable electronic safety-related systems

IEC 61508 [IEC 04] is the generic safety standard on which most domain specific safety standards are based on. The derived standards address the same V&V methods in principle, but in most cases only a subset or a selection is explicitly addressed in standards. An overview over the IEC 61508 group of functional safety standards is provided in Figure 3.

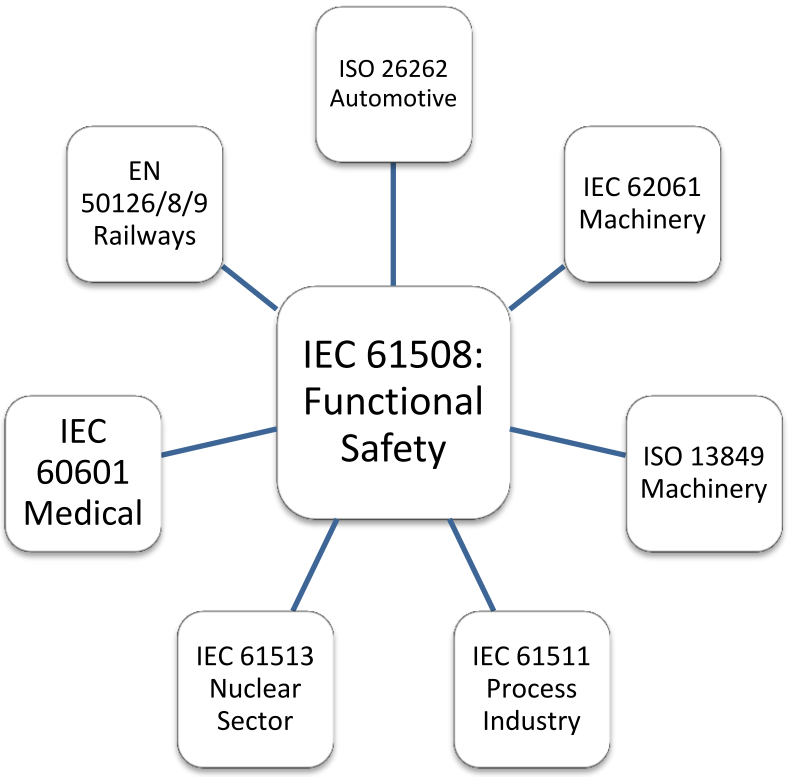

Figure 3: Overview of Functional Safety Standards

The V&V-Part of the overall lifecycle for safety relevant systems is described in Part 1 of the IEC 61508. Part 7 of the IEC 61508 contains an overview of techniques and measures and includes advice regarding Verification and Validation. Presented testing techniques include:

- Probabilistic testing
- Interface testing
- Boundary value analysis
- Error guessing
- Error seeding / fault injection
- Equivalence classes and input partition testing
- Structure-based testing
- Control flow analysis
- Data flow analysis
- Symbolic execution
- Formal proof (verification)
- Model checking
- Complexity metrics
- Formal inspections
- Walk-through (software)
- Design review
- Process simulation
- Performance requirements / modelling
- Avalanche/stress testing
- Response timing and memory constraints
- Model based testing (test case generation)
- Common cause failure analysis

Each method is described and references for the method are given.

3.3.2 Safety V&V methods with respect to multi-core architecture

Reliability and safety analyses have a remarkably long history; different domains developed safety-awareness at different times. Already in the 1930s statistical methods were used for industrial product quality control and for assessing the probability of airplane collisions, later for analyzing the German V1 rocket. In the 1950s FMEA was introduced by the US DOD for reliability analyses of military equipment, FTA was developed later in the 1960s for reliability assessment of ballistic missiles (Minuteman) and space rockets. In the 1970s risk analyses were intensively applied to nuclear reactors. The use of risk analyses in the chemical process industry was strongly encouraged by the Bhopal disaster in 1984. Since the 1990s several industry domains have developed their specific safety standards; first chemical process, space and avionics industries, in the mid-1990s the machinery industry, later the rail domain, in recent years the automotive industry. Today, risk analysis methods are also used for process improvement (process FMEA) in order to gain efficiency and to control product quality.

Caused by this long history most of the RAMS Methods are difficult to apply for the analysis of multi-core systems. While random faults of hardware components can usually be treated well with statistical failure models, due to the complexity of the parallel processing, potential side effects between multiple cores and the increased number of system states the effects of failures can be hard to determine.

Therefore only in rare cases, correctness can be proven formally; therefore fault mitigation measures resulting from risk analysis must usually be provided in the system concept. Most of the described analysis methods can be used for both hardware and software, i.e. to identify effects and costs of their failing.

Testing methods like fault injection or model/software/processor/system in the loop can also be applied to multi-core systems although the increased complexity also increases the effort needed.

3.4 Security-related V&V methods

This section reviews the existing verification and validation methods that focus on software security. The methods are group into (1) verification methods, (2) validation methods, (3) standards and best practices, and (4) risk analysis. Our review on verification and validation focuses on the general methods, i.e. although we focus on security-related verification and validation, some of the methods are also generic to be applied to safety. Since risk analysis methods are used transversally for verification and validation, they are organized into one subsection.

3.4.1 Standards and Best Practice

IEEE 1012 – 2004: Standard for Software Verification and Validation

This IEEE standard was released in 2004 and applies to systems, software, and hardware being developed, maintained, or reused. The standard covers all life cycle processes for systems and software and aims to be compatible with all life cycle models. It includes Agreement, Organizational Project-Enabling, Project, Technical, Software Implementation, Software Support, and Software Reuse process groups. The standard defines four integrity levels, describing the importance of a component to the user, and describes the corresponding minimum V&V tasks, which need to be performed in order to reach a certain integrity level. Furthermore, the standard specifies the intensity and rigor applied to the V&V tasks, which can vary according to the integrity level, and lists detailed criteria, such as minimum criteria for correctness, consistency, completeness, testability or accuracy, for the tasks. To address system issues at a system viewpoints level, the standard lists tasks for hazard analysis, security analysis, risk analysis, migration assessment, and retirement assessment, for both hardware and software.

ISO/IEC 15408 – YYYY: Common Criteria for Information Technology Security Evaluation

ISO/IEC 15408 [ISO 02] specifies the process of specification, implementation, and evaluation of security-critical, high-assurance systems in a rigorous and standardized manner. The effort of the evaluation process is ranked numerically from one to seven in Evaluation Assurance Levels (EAL). The seven assurance levels of EAL are defined in seven levels, from EALs Functionally tested to EAL7 formally verified design and tested.

FDA General Principles of Software Validation

FDA General Principles of Software Validation is a guideline that outlines general validation principles that the Food and Drug Administration (FDA) recommends for the validation of medical device software or the validation of software used to design, develop, or manufacture medical devices [23]. While the focus of these guidelines lies in the field of medical software, many aspects held also true for other fields of software engineering. The document covers among other things principles and context of software validation, as well as the typical tasks supporting validation.

FIPS 140-2

The Federal Information Processing Standard (FIPS) Publication 140-2, (FIPS PUB 140-2 is a standard published by the U.S. government and used for the accreditation of cryptographic modules [24]. The requirements for cryptographic modules formulated in FIPS 140-2 are the foundation of the Canadian / U.S. the Cryptographic Module Validation Program (CMVP).

Microsoft Security Development Lifecycle (MSDL)

[MSDL 01] is a software development security assurance process to integrate security and privacy practices into all phases of software development lifecycles. MSDL is divided into five phases: requirements, design, implementation, verification, and release.

3.4.2 Security Risk Analysis Methods

Risk analysis is an important component and step in any verification and validation process. The main objective of a risk analysis is to identify the risks to system security and determining the probability of occurrence, the resulting impact, and additional safeguards that would mitigate this impact. The results of risk analysis are the input for security requirements in the early phase in the lifecycle as well as for defining and plan particular tests.

3.4.2.1 Relevant standards and frameworks

AS/NZS 4360:2004 Risk Management [AS-NZS] is an Australian/New Zealand standard for documenting and managing risk. The standard defines five steps, which include establish context, identify the risks, analyze the risks, evaluate the risks, and treat the risks. It mostly addresses business and system risks. Technical risks are not the focus, but the general approach to system risk is notable.

IT Baseline Protection (IT-Grundschutz) [IT-BLP] is a standard from the German Federal Office for Security in Information Technology that provides information for identifying security risks and implementing protections in IT systems. It focuses on identifying overall hazards to an IT system and how to implement safeguards to mitigate them. Since 2005, it was renamed to “Baseline Protection Catalogues”, since it contains many different manuals and books on many security issues.

OCTAVE [25] (Operationally Critical Threat, Asset, and Vulnerability Evaluation) is a framework for identifying and managing information security risks. It contains methods, techniques and tools for asset-driven evaluation approach, focusing on security practices and strategic issues, and self-direction.

Cybersecurity and Software Assurance Measurement and Analysis [26] is an approach developed by the Software Engineering Institute at Carnegie Mellon University for measuring and monitoring the security characteristic of interactivity complex, software-reliant systems across the lifecycle and supply chain. It includes an Integrated Measurement and Analysis Framework (IMAF) and Mission Risk Diagnostic (MRD).

Common Vulnerability Scoring System (CVSS) [27] is a vulnerability scoring system designed to provide an open and standardized method for rating information security vulnerabilities. CVSS can be used together with Common Vulnerability Enumeration (CVE) to determine and rank risks.

3.4.2.2 Securitiy Risk Modelling and Analysis

HMG Infosec Standard No. 1 is a UK security standard for the assessment of technical risk to information security of government computers. It uses Domain Based Security for modeling and analyzing information security risks and security controls in a business context. Domain Based Security is a set of notations and techniques for describing and assessing business driven information security requirements for network architectures [28].

UML-based approach [IEC 04, Part 7] defines a UML meta model in form of UML profiles for model-based risk assessment. It integrates risk analysis methods like HAZOP, FTA, and FMEA with UML modeling in a framework for risk assessment of IT systems.

Misuse and abuse cases [29] are a requirement engineering method to identify security risks and communicate them to stakeholders. UML use case diagrams or text can be used in the process.

Threat modeling [30], is a general concept referring to the description of threats to a system. Probably the most well-know is the Microsoft threat modelling procedure, designed for Microsoft Secure Development Lifecycle to identify threats and their impact on a system design. Threat modeling is divided into six steps: (1) identify assets, (2) create an architecture overview, (3) decompose the application, (4) identify the threats, (5) document the threats, and (6) rate the threats.

STRIDE is a method developed in Microsoft to help software developer in the process. STRIDE stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. The acronym is used as a mnemonic for identify network, host-based, and application threats in a Web application or a software system design.

TRIKE is an open source threat modeling methodology (and tool] similar to STRIDE.

Attack graph [31] is an approach to enumerate all possible network-based attacks to an IT system. It is based on model checking. The network is modelled as a finite state machine. Nodes represent network states with respect to security properties and edges represent attack actions that cause the transition of states. An attack graph can be generated by using vulnerability information and network connectivity. Each path in the graph indicates a series of exploits which starts from an initial secure state to an unsecured (attacker's) goal state. A limitation of attack graph is that it expands exponentially with the size of the network. Therefore, it is still impractical for analyzing large networks.

Attack tree is a method to identify all potential attacks an attacker will perform in order to comprise a system. Attack trees consist of nodes and edges, whereby nodes in an attack tree represent an attacker's actions. Attack trees use a root node to specify an attacker's goal and systematically expand the tree with leaf nodes to enumerate possible attacks that contribute to reach the goal. The leaf nodes are grouped by logical AND and OR relations. Attack trees provide a structured way of security analysis and have the potential to identify hidden risks.

Petri net models of cyber physical systems [32] are an academic approach to assess vulnerability of SCADA systems. Petri net is often used to model concurrent systems. In this approach, a SCADA system is modeled in Petri net. Attacks and impacts are determined by simulations.

Further, Behnia et al. [33] point out the range of differences between security analysis methodologies, including OCTAVE, ISRAM [34], CORAS [35] and several others. Furthermore, the European Network and Information Security Agency (ENISA) maintain a repository of risk assessment standards, methods and tools from a European perspective [36]. The Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE) method can be used for identifying and managing information security risks. It contains methods, techniques and tools for an asset-driven evaluation approach, focusing on security practices and strategic issues, and self-direction. Similarly, MAGERIT is a risk analysis and management methodology that has been developed in Spain [37]. In a similar manner to OCTAVE, it is driven by an analysis of the assets that are associated with an organization.

Although there are many more security risk analysis approaches, they are mostly related to similar to the methods presented above.

3.4.3 Security V&V methods with respect to multi-core architecture

Multi-core architecture poses specific security problems. Previous research shows that the main security risks include parallelization, shared resources, and open interface [38]. Parallelization occurs when multiple tasks are executed on multi-core CPUs. This might lead to new vulnerabilities to attacks such as denial of service or starvation. In multi-core system it is hard to pre-compute the worst case execution time. Multi-core systems rely on shared resources such as caches, memory, and communication infrastructures. The challenge is to properly separate multiple tasks and codes with different criticality and security levels on different cores. The non-proprietary architecture improves interoperability and encourages the building of an ecosystem, in which all stakeholders can benefit. However, it also creates security problems, in which the PC-like approach will also lead to PC-like problems like easy malware infection and enlarged attack surface.

To this date, we think some of the security V&V methods can be directly applied to multi-core systems if they can be adapted to take specific multi-core properties into account. However, what is more important is that the V&V methods need to integrate safety aspect. For example, the impact of a denial of service attack itself might not be security-critical, but is fatal from a safety-perspective.

3.5 Combined safety and security related V&V methods

Combined safety and security related V&V methods are still mostly a research topic. Existing approaches lack in quantitative results and rigorousness. None of them was explicitly developed with multi-core systems in mind.

Combined Harm Assessment of Safety and Security for Information Systems (CHASSIS)

CHASSIS [39] uses Misuse Case for the modeling and analysis of potential effect for failures and attacks on a system. A misuse is an event, triggered by the action of an attacker or a failure. Misuse Cases are further refined with misuse and failure sequence diagrams. The new “mitigation” relationship defines elements which are useful for mitigating failures and attacks. Based on the diagrams and the “mitigation” relationship Safety & Security Requirements are defined.

Boolean logic Driven Markov Processes (BDMP)

BDMP [40, 41] is based on combined Fault an Attack Trees. Both are extended with additional nodes and temporal dependencies. The resulting Tree depicts a model where logical connections (A and B causes C) and temporal connections (A can only happen after B happened) are included. This allows the modeling of attack paths.

Failure Modes, Vulnerability and Effects Analysis FMVEA

FMVEA [41] is a combined analysis method for safety and security. It is based on the Failure Mode and Effects Analysis, as described in IEC 60812.

FMVEA extends the approach with a security analysis, based on threat modeling [30]. Components are not only examined for failure modes, but also for threat modes. While a failure mode describes how a system quality attribute fails, a threat mode describes how a security attribute of the components fails. It is the manner by which an occurred threat is observed. Causes for failing security attributes are vulnerabilities. In order to estimate the frequency of threat modes, potential attackers (threat agents) are identified. The probability of a threat mode is determined based on the threat agent and the vulnerability. The results of a FMVEA are the failure and threat modes of a system, and their causes and consequences. In addition, failure and threat modes are evaluated in terms of probability and severity.

3.6 Robustness and performance testing methods

In this context, it has to be examined how a target system (SUT) behaves under unusual conditions. For that purpose, it is confronted with critical or extreme situations. This means that it is provided with corresponding input or stimuli. It should be noted that robustness against internal faults is regarded as fault tolerance and not treated here; instead, methods such as fault injection as addressed in 2.2 are used for that purpose.

In principle, robustness and performance are considered as two different dimensions of non-functional system properties: robustness is concerned with to what a degree the SUT operates correctly on invalid input or stressful environmental conditions, while performance describes how the SUT performs in terms of responsiveness and stability under a particular workload. However, high workloads cause stress; hence, there is an overlap between robustness and performance testing.

3.6.1 Robustness and performance testing of SUTs with low-dimensional input

“Low-dimensional” denotes stimuli that can be presented with a small number of values – e.g. temperature, position, or documents that are treated as compact units by the SUT). Robustness testing then should provide inputs for at least following situations:

Invalid value(s) (value(s) out of range): A certain input parameter can have a data type that includes values which are invalid for a given use (for instance, room temperatures outside range [0, 40]°C).

Value(s) at range borders feeds the SUT with value sequences around borders of valid ranges.

Invalid sequences: Sensors typically provide measured values in repeated manner (e.g. periodically) to the system. Often, changes between consecutive values should not exceed certain ranges in normal operation (e.g. acceleration). Hence, the SUT should be tested with sensor value sequences that violate this rule.

Invalid input combinations: Combinations of values for different sensors input parameter may be invalid, even if each single value stays in its valid range. For instance, if a train must have doors closed while moving, a test input vector [”door open”, “speed > 0”] would be such an invalid input combination.

Unusual workload: If varying density of input provision is possible, for instance in case of event-based input, then feeding SUTs with input densities outside of the usual range is also part of robustness testing. Of course, this comprises overload, but for certain applications underloading may also be an issue. In case of input of varying size and complexity, e.g. in document processing systems, providing input out of usual size and range is another mode of unusual workload.

Environmental conditions: Another aspect of robustness is vulnerability against conditions not controlled via input, e.g. available power or memory. Varying such parameters should also be art of robustness testing.

In principle, these situations should be tested for all input parameters of a SUT. However, an exhaustive coverage of *invalid input combinations* could easily require unacceptable efforts. An alternative is to restrict to pairwise selection of inputs, as experiences indicate that faults caused by combination of more than two input parameters can mostly be found already by combination of two inputs.

Performance testing usually varies individual parameters, both input and environmental, in the usual range to measure corresponding quality characteristics (e.g. throughput or precision). Often, results are presented as curves or diagrams, illustrating dependencies between workload and performance. An alternative is the use of benchmarks, predefined test settings, which allow comparing different SUTs. Of course (and as already mentioned), for measuring performance under extreme situations, similar methods as outlined for robustness testing will be applied.

3.6.2 Robustness testing of SUTs with high-dimensional Input

Examples for such SUTs are systems using digital cameras as sensor devices, because each pixel has to be considered as individual sensor. Here, the test situations listed above are of little use, because each input value (pixel) contributes only a tiny fragment to the overall input information (the captured image).

Such systems are part of the use cases 10.4 (Manufacturing quality control by 3D inspection), 12.2 (Video surveillance for critical infrastructure) and autonomous vehicles/systems in general.

Instead, it has to be tested how vulnerable a camera-based SUT is with respect to difficulties in observed scenes. Examples for such difficulties (“visual criticalities”) are shadows, occlusion among objects, reflections, unusual object orientations, or complex object shapes and textures.

The predominant method to test such systems consists in manually recording test images or videos in a real/physical environment. Recorded test images, however, lack coverage information of included visual criticalities. Further, it may be too dangerous or expensive to arrange certain situations in reality. Another problem is the lack of expected results, denoted as ground truth (GT), in order to assess the quality of the SUTs output. Such output ranges from *low-level* – e.g. depth map (distance per pixel) – via *mid-level* – e.g. image segmentation – to *high-level* – e.g. identified persons. Manual generation of such data not only is extremely tedious and expensive, but also error prone.

An alternative is synthetic generation of test data (input images/sequences and GT) from geometric models, exploiting techniques that are widely used in computer graphics and virtual reality. This not only allows for including relevant 3D-scenes as well as visual difficulties, it also enables provision of precise GT almost for free (once the corresponding GT-generator has been implemented). A solution that takes this approach is VITRO[2], a coverage-based methodology and tool set for automatic generation of test

[2] This work was partially funded by the ARTEMIS-project R3-COP, grant agreement 100233, by the ARTEMIS JU and the partners national funding authorities

images for computer vision algorithms. It relies on an extensible catalogue of computer vision criticalities, which consist of artefacts coming from the 3D scenarios (shadows etc. as already mentioned) and artefacts caused by the sensor itself (aberration or thermal noise, over-discretization of input data). This extensive list of criticalities was assembled in a systematic fashion by applying the approved hazard analysis method HAZOP [IEC 05] to the generic computer vision case. The resulting catalogue[3] with about 1000 entries forms a strong basis for identifying criticalities.

3.7 Test methods for other dependability and quality attributes

General Analysis and V&V methods with respect to safety and security (and combined approaches) are discussed in section 3.1, 3.2, 3.3, 3.4 and 3.5., robustness and performance testing methods in 3.6. Dependability according to Laprie, Avizienis, Kopetz, Voges, Anderson et. al. in "Dependability – Basic Concepts and Terminology" [22], covers additional attributes critical for sustainable safety-related systems in the long term (see Figure 4):

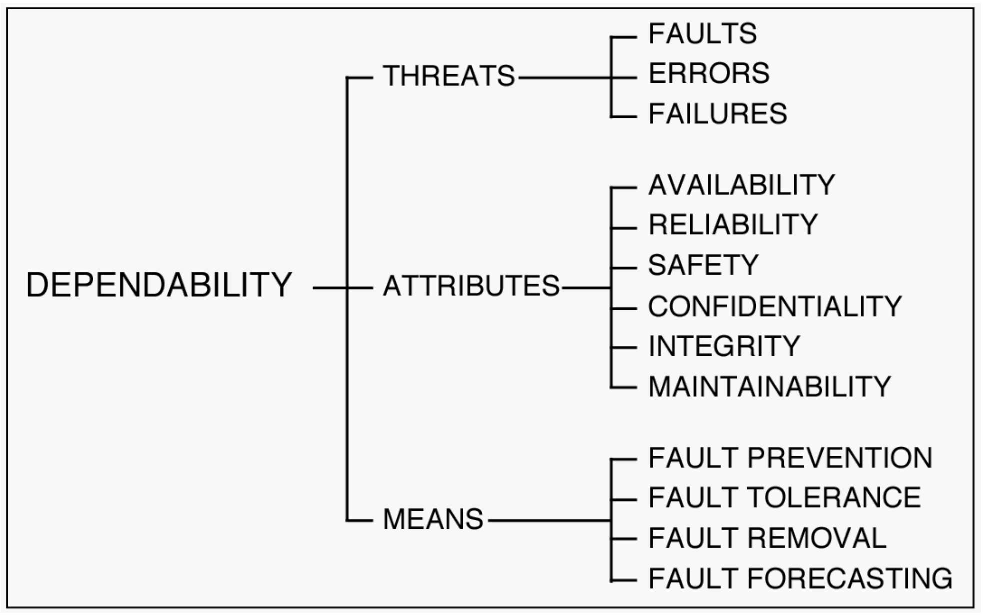

Figure 4: Dependability – basic concepts and terminology

The attributes cover safety, security (confidentiality, partially integrity and availability), reliability, maintainability (enhancement and modifications), and in some recent version, additionally attributes such as sustainability, resilience, robustness. Part 7 of IEC 61508, Ed. 2.0: 2010 - Overview of techniques and measures, contains a comprehensive list and short descriptions of hardware, software and system level methods and techniques (informative).

Reliability – overview of techniques and measures for control of random hardware failures (Annex A):

The annex A describes a set of techniques for hardware monitoring (components such as relays, processors, memory, signals, sensors, actuators, power supply, communication on HW level, etc.) by different methods (tests by redundant hardware, self-test by software, comparison, checksums, signatures, RAM tests, test patterns for I/O units and interfaces, watchdogs and the like. Many of these techniques can be applied against security breaches as well to detect anomalies in the system before they become effective.

[3] http://www.r3-cop.eu/wp-content/uploads/2013/03/R3-COP-D4.2.3_CV-HAZOP_r1.1.pdf

Operation and Maintenance are covered in annex B, avoidance of systematic failures (besides specification and design methods and tools, like formal methods, inspection, walk-through etc.). B.4 addresses issues like operation and maintenance instructions, user-friendliness, maintenance friendliness, limited operation possibilities, protection against operator mistake and against unqualified modification. Testing methods during system integration as already discussed in chapters before are part of B.5., validation methods for system validation are in B.6. this includes FMEA, Cause-consequence diagrams, ETA (event tree analysis), FMECA, FTA, Markov models, Reliability block diagrams, Monte Carlo simulation and fault tree models, stochastic Petri Nets, fault insertion testing etc.

Software safety integrity is covered in Annex C, the relevant methods and techniques for V&V under the sub-header C.4, "Architecture design". It includes static (UML modelling, use case and activity diagrams) as well as dynamic (run-time) techniques (fault detection and diagnosis, error detecting codes, failure assertion programming, diverse monitoring and execution, recovery- and AI-based fault correction, dynamic reconfiguration).

Safety and Performance in Real-Time: Under this header is Time-Triggered Architecture referenced, particularly because it allows to guarantee safety and real-time properties by static configuration of the system.

Performance, Verification and Modification is handled under C.5, e.g. Probabilistic testing, data recording and analysis, structure based testing and control flow analysis, symbolic execution, formal verification (proof), model checking, process simulation and performance modelling, stress testing, impact analysis, regression validation and last but not least model-based testing [43] and test case generation, just to mention those relevant in our case, besides rather simple and well-known quality assurance techniques.

3.8 Run-time assurance, verification and certification methods

Runtime assurance and certification, as it is our goal in EMC², has rarely been addressed in research up to now. Ideas for the certification at runtime were first introduced by Rushby [42]. He describes ideas that it should be feasible to execute formal analyses at runtime, making it possible to formally verify that a component behaves as specified during run-time. However, he does not provide concrete solutions. In a second publication [43] Rushby outlines a framework in which the basis for certification is changed from compliance with standards to the construction of explicit goals, evidence, and arguments (generally called an "assurance case"). He then describes how runtime verification can be used within this framework, thereby allowing certification to be partly performed at runtime. The core of this approach is the usage of runtime monitors, which were defined outside the context of an assurance case in order to dynamically monitor assumptions, anomalies, and safety, respectively.

In the automotive domain, Östberg and Bengtsson [44] propose an extension to AUTomotive Open System Architecture (AUTOSAR) which consists of a safety manager that actively enforces the safety rules described in dynamic safety contracts. Also in the automotive domain, Frtunikj et. Al. [45] presents a runtime qualitative safety-assessment that considers Automotive Safety Integrity Level (ASIL) and its decompositions in open automotive systems. In their solution, the authors consider the modularization of safety-assessment using Safety Elements out of Context (SEooC) from ISO 26262 [ISO 01].

As one possible approach, Schneider and Trapp introduced the concept of Conditional Safety Certificates (ConSerts) [46] [47], which facilitate the modular definition of conditional safety certificates for single components and systems using a contract-like approach. These are post-certification artefacts, whose actual certification has been conducted in the traditional way with safety engineers analyzing the system and developing an adequate safety concept. The conditional certificates are to be evaluated automatically and autonomously by the system at the moment of integration at runtime, based on runtime representations of the certificates of the involved units of composition.

For more extensive State of the art in this matter refer to the document *D6.3 Preliminary definition of the runtime certificate approach*.

3.9 Virtual Evaluation and Testing

Virtual evaluation and testing is considered to be a crucial part of the product finalization is the overall certification process of a prototype, being present in all the product's lifecycle stages. Model-based design has the potential to enhance the development of such systems offering virtual evaluation of alternative designs before full-scale physical prototypes are built. State of the art tools incorporating advanced testing techniques in the complete product life cycle are though limited, often involving highly competent people as end-users, and specialized virtual testing environments. There are efforts in the research bibliography that tackle virtual testing and evaluation problems by addressing the tools ability to combine discrete and continuous models successfully into collaborative model or co-models.

This has been achieved both at the level of single embedded systems, using a common underlying formalism and through semantic unification of separate formalisms permitting design space exploration and holistic verification through joint simulation (co-simulation). Current EU projects such as FP7 IMVITER[4], FP7 VERITAS[5], describe state of the art tools for enabling virtual testing capabilities for component level designs, especially for the aerospace domain. Following recent activities for example in the aerospace domain, projects like VIVACE also provide fruitful guidelines in order to manage engineering data in a broader perspective. The challenges here for EMC^2 will be the management of information interchange and semantic integration between those tools, while adapting existing tools that offer mature verification services, in order for the necessary evidence to be produced and presented adequately back to the designer.

[4] http://www.imviter.com/
[5] http://veritas-project.eu/

4. Conclusions

In this document, the state of the art in the area of system qualification and certification has been presented with focus on applicability and adequacy for multi- and many-core systems. The specific challenges in multi- and many-core architectures have been identified, and the resulting verification and validation needs in order to minimize the associated risks have been derived.

It will be necessary to include verification and validation methods not only for safety but also for security because interconnected multi- and many-core architectures represent Cyber-Physical-Systems (CPS), which are strongly exposed to security threats. As a consequence and because of the interrelation of countermeasures against weaknesses in respect to safety and security, the document describes shortly existing combined safety- and security-engineering approaches, but here a strong need for improvement is identified.

An overview of current domain-specific practices for system qualification and certification for the automotive, avionics, space and smart grid domains makes clear that - for the time being - multi- and many-core-specific issues are not yet sufficiently treated in existing functional safety standards. Here an improvement is highly recommendable and the EMC2 consortium should endeavor to start respective discussion processes in the standards working groups.

The survey of system qualification and certification comprises safety and security risk analysis methods as well as static and dynamic verification techniques. Many verification and validations methods can be derived from applicable security and functional safety standards. There is, however, no specific guidance for multi- and many-core architectures. Available risk analysis standards give on the one hand overview information, and describe on the other hand individual methods. As the standards describing the individual methods are generic in respect to the analyzed systems, guidance for their application in the multi- and many-core area should come from the safety and security standards.

To summarize, the application of existing qualification and certification methods will need to be adapted for the use in the multi- and many-core domain. Safety and security should be considered equally, and the mutual influence of mitigation measures should be considered with view to the overall safety and security goals. Guidance should come from the EMC2 consortium, and the project results should also impact standardization, for instance future maintenance of functional safety standards.

5. References

5.1 Standards

[AS-NZS] AS/NZS (Australian/New Zealand Standard) 4360:2004 Risk Management

[ECSS 01] ECSS-E-ST-40 ECSS Space Engineering Standard, Software Engineering

[ECSS 02] ECSS-Q-ST-80 ECSS Product Assurance Standard, Software Product Assurance

[IEC 01] IEC 31010: Risk management — Risk assessment techniques, 2009.

[IEC 02] IEC 60812 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA), IEC, 2006.

[IEC 03] IEC 61025: Fault Tree Analysis, 2007.

[IEC 04] IEC 61508: Functional Safety of Electrical / Electronic / Programmable Electronic Safety-Related Systems, Ed. 2.0, 2010

[IEC 05] IEC 61882: Hazard and operability studies (HAZOP studies), IEC, 2001.

[IEC 06] IEC Smart Grid Standardization Roadmap, 2010.

[IEEE 01] IEEE Std 1012 - Standard for System and Software Verification and Validation, 2004

[IEEE 02] IEEE Standard Glossary of Software Engineering Terminology, 1990.

[ISO 01] ISO, 26262 Road Vehicles - Functional Safety, 2011.

[ISO 02] ISO/IEC 15408: Common Criteria for Information Technology Security Evaluation

[IT-BLP] IT Baseline Protection Manual (German: IT Grundschutz-Handbuch), German Federal Office for Security in Information Technology (BSI), since 2005 “Baseline Protection Catalogues”

[MSDL] Microsoft Security Development Lifecycle http://www.microsoft.com/security/

[RTCA 01] DO-178C/EUROCAE ED-12C Software Consideration in Airborne Systems and Equipment Certification, 2012.

[RTCA 02] DO 178C Technology Supplements:

- DO-330 "Software Tool Qualification Considerations" - clarifying software tools and avionics tool qualification
- DO-331 "Model-Based Development and Verification Supplement to DO-178C and DO-278" - addressing Model-Based Development (MBD) and verification and the ability to use modeling techniques to improve development and verification while avoiding pitfalls inherent in some modeling methods
- DO-332 "Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A" - addressing object-oriented software and the conditions under which it can be used
- DO-333 "Formal Methods Supplement to DO-178C and DO-278A" - addressing formal methods to complement (but not replace) testing

[RTCA 03] DO-254/ EUROCAE ED-80 “Design Assurance Guidance for Airborne Electronic Hardware”.

[RTCA 04] DO-278 / EUROCAE ED-109 “Guidelines for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance”, is the ground based complement to the DO-178B airborne standard.

[RTCA 05] DO-178B / EUROCAE ED-12B “Software Considerations in Airborne Systems and Equipment Certification. Requirements and Technical Concepts for Aviation”

[SAE 01] ARP-4761 System Safety Assessment, 1996.

5.2 Publications

[1] B. Zimmer (ed.), "D2.4 System Architecture, Version 2.0: Safety, E2.3.1.1 Specification of Certification-Related Elements of the Network-, Logical-, and Technical Computer Architecture," ARAMiS Deliverable, 2013.

[2] B. Zimmer, M. Damm (eds.), "D4.4 Specifications to Ensure Safety/Certifiability on Software-Level, E4.3.1.1 Requirements for the Certification of Multicore Architectures," ARAMiS Deliverable, 2014.

[3] Ondrej Kotaba, Jan Nowotsch, Michael Paulitsch, Stefan M. Petters, Henrik Theiling, "Multicore in Real-Time Systems – Temporal Isolation Challenges due to Shared Resources," in *Workshop on Industry-Driven Approaches for Cost-effective Certification of Safety-Critical, Mixed-Criticality Systems (WICERT 2013), co-located with DATE 2013*, Grenoble, France, 2014.

[4] CPSoS Towards a European Roadmap on Research and Innovation in Engineering and Management of Cyber-Physical Systems of Systems, "D2.4 - Analysis of the State-of-the-Art and Future Challenges in Cyber-Physical Systems of Systems," 2015. [Online]. Available: http://www.cpsos.eu/wp-content/uploads/2015/02/D2-4-State-of-the-art-and-future-challenges-in-cyber-physical-systems-of-.pdf.

[5] David Peter Eames, Jonathan Moffett, "The Integration of Safety and Security Requirements," in *Lecture Notes in Computer Science, Computer Safety, Reliability and Security*, Toulouse, 1999.

[6] Goel Sandeep Kumar, Narendra Devta-Prasanna, Mark Ward, "Comparing the effectiveness of deterministic bridge fault and multiple-detect stuck fault patterns for physical bridge defects: A simulation and silicon study," in *International Test Conference*, 2009.

[7] R. Guo, S. Mitra, E. Amyeen, J. Lee, S. Sivaraj, S. Venkataraman, "Evaluation of test metrics: Stuck-at, bridge coverage estimate and gate exhaustive," in *Proccedings of VLSI Test Symposium*, 2006.

[8] Bruno Dutertre, Victoria Stavridou, "A model of non-interference for integrating mixed-criticality software component," in *In Weinstock and Ruishby, DCCA-7, Proceedings*, San Jose, CA, 1999, pp. 301-316.

[9] John Rushby, "Formal methods and the certification of critical systems," SRI International, Menlo Park, CA, 4551, 1993.

[10] RTCA Inc., *Advisory Circular 20 115B., Document RTCA/DO-178B*, Federal Aviation Administration, 1993.

[11] SAE ARP-4761, ARP-4761 System Safety Assessment, 1996.

[12] European Aviation Safety Agency, "The Use of Multicore Processors in Airborne Systems," 2012.

[13] European Aviation Safety Agency., "Development Assurance of Airborne Electronic Hardware," *Certification Memorandum,* no. 01, 2011.

[14] DOT/FAA/AR-11/5, "Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure," 2011.

[15] European Aviation Safety Agency, "Software Aspects of Certification," *Certification Memorandum,* no. 01, 2011.

[16] Certification Authorities Software Team, "Multi-core Processors," Position Paper CAST-32, 2014.

[17] AFE75 COTS AEH, "Assurance Issues, Problems, and Emerging Solutions, Final Report," To be published.

[18] A. Soderberg, B. Vedder, "Composable Safety-Critical Systems Based on Pre-Certified Software Components," in *ISSREW*, 2013.

[19] F. Vernadat, C. Percebois, P. Farail, R. Vingerhoeds, A. Rossignol, P. Talpin, D. Chemouil, "The TOPCASED project-a toolkit in open-source for critical applications and system development," in *Data Systems In Aerospace (DASIA)*, Berlin, 2006.

[20] Tran Eushiuan, "Verification/Validation/Certification," Spring, Carnegie Mellon University, 1999.

[21] Bart Broekman, Edwin Notenboom, Testing Embedded Software, London: Addison Wesley, 2003.

[22] Jean-Claude Laprie, "Dependable computing: Concepts, limits, challenges," *FTCS-25, the 25th IEEE International Symposium on Fault-Tolerant Computing-Special Issue*, p. 42–54, 1995.

[23] U.S. Department Of Health and Human Services, "General Principles of Software Validation; Final Guidance for Industry and FDA Staff," 2002.

[24] National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules,* Federal Information Processing Standards Publications, 1994.

[25] Christopher J. Alberts, Audrey Drofee, Managing information security risks: the OCTAVE approach, Addison-Wesley Longman Publishing Co., Inc., 2002.

[26] Carnegie Mellon University, "Cybersecurity and Software Assurance Measurement and Analysis," [Online]. Available: http://www.cert.org/cybersecurity-engineering/research/cybersecurity-and-sw-assurance-measurement-and-analysis.cfm?.

[27] Peter Mell, Karen Scarfone, Sasha Romanosky, "Common Vulnerability Scoring System," *IEEE, Security & Privacy,* vol. 4, no. 6, pp. 85 - 89, 2006.

[28] Brian Monahan, Frederic Gittler, William Horne, Simon Shiu, Adrian Baldwin, Chris Dalton, "DBSy in a Commercial Services Context," in *1st DBSy User Conference*, Great Malvern, UK, 2005.

[29] Chun Wei (Johnny), Sia, "Misuse Cases and Abuse Cases in Eliciting Security Requirements," 25 October 2005. [Online]. Available: https://www.cs.auckland.ac.nz/courses/compsci725s2c/archive/termpapers/csia.pdf.

[30] Frank Swiderski, Window Snyder, Threat Modeling, Redmond: Microsoft Press, 2004.

[31] A. Singhal, X. Ou, *Security risk analysis analysis of enterprise networks using probabilistic attack graphs.,* NIST Interagency Report 7788, 2011.

[32] Ten Chee-Wooi, Chen-Ching Liu, Govindarasu Manimaran, "Vulnerability assessment of cybersecurity for SCADA systems," *Power Systems, IEEE Transactions on,* vol. 23, no. 4, pp. 1836-1846, 2008.

[33] A. Behnia, R. Rashid, J. Chaudhry, "A Survey of Informatoin Security Risk Analysis Methods," *Smart Computing Review,* vol. 2, no. 1, 2012.

[34] I. Sogukpinar, B. Karabacak, "ISRAM: information security risk analysis method," *Computers & Security,* vol. 24, no. 2, pp. 147--159, 2005.

[35] Folker den Braber, Theo Dimitrakos, Bjørn Axel Gran, Ketil Stølen, Jan Øyvind Aagedal, "The CORAS methodology: model-based risk assessment using UML and UP," 2003.

[36] European Union Agency for Network and Information Security, "Inventory of risk management/risk assessment methods and tools," 2013. [Online]. Available: https://www.enisa.europa.eu/activities/risk-management/current-risk/risk-management-inventory.

[37] Francisco López Crespo, Miguel Angel Amutio Gómez, Javier Candau, José Antonio Mañas, "MAGERIT - version 2 Methodology for Information Systems Risk Analysis and Management Book I - The Method," Ministerio de administraciones públicas, 2006.

[38] Claudia Eckert, Thomas Kittel, "Security Issues of Multi-Core Architectures—The Automotive Case," *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik,* vol. 55, no. 1, pp. 5--9, 2013.

[39] Christian Raspotnig, Peter Karpati, Vikash Katta, "A Combined Process for Elicitation and Analysis of Safety and Security Requirements," in *Lecture Notes in Business Information Processing, Enterprise, Business-Process and Information Systems Modeling*, Gdańsk, Poland, 2012.

[40] Ludovic Piétre-Cambacédès, Marc Bouissou, "Attack and defense dynamic modeling with BDMP," *Lecture Notes in Computer Science, Computer Network Security,* 2010.

[41] Ludovic Piètre-Cambacédès, Marc Bouissou, "Modeling safety and security interdependencies with BDMP (Boolean logic Driven Markov Processes)," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, 2010.

[42] Christoph Schmittner, Thomas Gruber, Peter Puschner, Erwin Schoitsch, "Security application of failure mode and effect analysis (FMEA)," in *Computer Safety, Reliability, and Security*, Florenz, 2014.

[43] Hesham Shokry, Mike Hinchey, "Model-based verification of embedded software," *Computer,* pp. 53 - 59, 2009.

[44] John Rushby, "Just-in-time certification," in *12th IEEE International Conference on Engineering Complex Computer Systems*, 2007.

[45] John Rushby, "Runtime certification," in *Runtime Verification*, Springer Berlin Heidelberg, 2008, pp. 21 - 35.

[46] Kenneth Östberg, Magnus Bengtsson, "Run time safety analysis for automotive systems in an open and adaptive environment," in *Proceedings of Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems) of the 32nd International Conference on Computer Safety, Reliability and Security. 2013*, 2013.

[47] Jelena Frtunikj, Michael Armbruster, Alois Knoll, "Data-Centric Middleware support for ASIL assessment and decomposition in open automotive systems," *GMM-Fachbericht-AmE 2014–Automotive meets Electronics,* 2014.

[48] Daniel Schneider, Mario Trapp, "A safety engineering framework for open adaptive systems," in *Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO),* 2011.

[49] Daniel Schneider, Mario Trapp, "Conditional safety certification of open adaptive systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS),* p. 8., 2013.

6. Appendix: Abbreviations

Abbreviation	Meaning
EMC^2	Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments
µC	Micro-Controller
ASIL	Automotive SIL (Safety Integrity Level)
CPS	Cyber-Physical-Systems
CPU	Central Processing Unit
DC	Diagnostic Coverage; Direct Current (depending on context)
EASA	European Aviation Safety Agency
ECSS	European Cooperation for Space Standardization
FAA	Federal Aviation Administration (US Authority)
FDA	Food and Drug Administration (US Authority)
FMEA	Failure Mode and Effects Analysis
FMECA	Failure mode, effects and criticality analysis
FMEDA	Failure Modes Effects and Diagnostic Analysis
FTA	Fault Tree Analysis
HAZOP	Hazard and Operability Analysis (Study)
HW	Hardware
MSDL	Microsoft Security Development Lifecycle
RAMS	Reliability, Availability, Maintainability *and* Safety
RAMSS	Reliability, Availability, Maintainability, Safety and Security
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition (in Industrial Automation and Control)
SIL	Safety Integrity Level
SMP	Symmetric Multi-Processing
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Escalation of Privilege (MicroSoft Threat Model)
SUT	System under Test
SW	Software
TRIKE	A Threat Modeling Methodology and Tool
V&V	Verification & Validation
WCET	Worst Case Execution Time