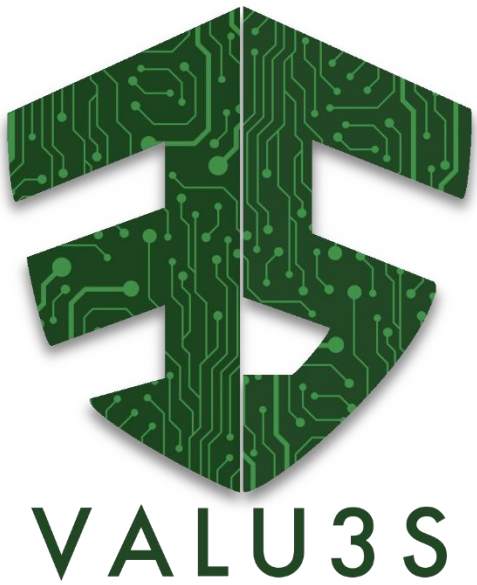


Verification and Validation of Automated Systems' Safety and Security

Concept for Using Virtual Prototypes in the Context of the Simulated Verification

Document Type	Report
Document Number	D4.3
Primary Author(s)	Thomas Bauer (FRAUNHOFER IESE)
Document Date	2021-10-29
Document Version	1.1 Final
Dissemination Level	Public (PU)
Reference DoA	2021-02-26
Project Coordinator	Behrooz Sangchoolie, behrooz.sangchoolie@ri.se, RISE Research Institutes of Sweden
Project Homepage	www.valu3s.eu
JU Grant Agreement	876852

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.

Disclaimer

Project Overview

Manufacturers of automated systems and the manufacturers of the components used in these systems have been allocating an enormous amount of time and effort in the past years developing and conducting research on automated systems. The effort spent has resulted in the availability of prototypes demonstrating new capabilities as well as the introduction of such systems to the market within different domains. Manufacturers of these systems need to make sure that the systems function in the intended way and according to specifications which is not a trivial task as system complexity rises dramatically the more integrated and interconnected these systems become with the addition of automated functionality and features to them.

With rising complexity, unknown emerging properties of the system may come to the surface making, it necessary to conduct thorough verification and validation (V&V) of these systems. Through the V&V of automated systems, the manufacturers of these systems are able to ensure safe, secure and reliable systems for society to use since failures in highly automated systems can be catastrophic.

The high complexity of automated systems incurs an overhead on the V&V process making it time-consuming and costly. VALU3S aims to design, implement and evaluate state-of-the-art V&V methods and tools in order to reduce the time and cost needed to verify and validate automated systems with respect to safety, cybersecurity and privacy (SCP) requirements. This will ensure that European manufacturers of automated systems remain competitive and that they remain world leaders. To this end, a multi-domain framework is designed and evaluated with the aim to create a clear structure around the components and elements needed to conduct the V&V process through identification and classification of evaluation methods, tools, environments and concepts that are needed to verify and validate automated systems with respect to SCP requirements.

In VALU3S, 12 use cases with specific safety, security and privacy requirements will be studied in detail. Several state-of-the-art V&V methods will be investigated and further enhanced in addition to implementing new methods aiming to reduce the time and cost needed to conduct V&V of automated systems. The V&V methods investigated are then used to design improved process workflows for V&V of automated systems. Several tools will be implemented supporting the improved processes, which are evaluated by qualification and quantification of safety, security and privacy as well as other evaluation criteria using demonstrators. VALU3S will also influence the development of safety, security and privacy standards through active participation in related standardisation groups. VALU3S will provide guidelines to the testing community, including engineers and researchers, on how the V&V of automated systems could be improved considering the cost, time and effort of conducting the tests.

VALU3S brings together a consortium with partners from 10 different countries, with a mix of *industrial partners* (24 partners) from automotive, agriculture, railway, healthcare, aerospace and industrial automation and robotics domains, as well as leading *research institutes* (6 partners) and *universities* (10 partners) to reach the project goal.

Consortium

RISE RESEARCH INSTITUTES OF SWEDEN AB	RISE	Sweden
STAM SRL	STAM	Italy
FONDAZIONE BRUNO KESSLER	FBK	Italy
KNOWLEDGE CENTRIC SOLUTIONS SL - THE REUSE COMPANY	TRC	Spain
UNIVERSITA DEGLI STUDI DELL'AQUILA	UNIVAQ	Italy
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO	ISEP	Portugal
UNIVERSITA DEGLI STUDI DI GENOVA	UNIGE	Italy
CAMEA, spol. s r.o.	CAMEA	Czech
IKERLAN S. COOP	IKER	Spain
R G B MEDICAL DEVICES SA	RGB	Spain
UNIVERSIDADE DE COIMBRA	COIMBRA	Portugal
VYSOKE UCENI TECHNICKE V BRNE - BRNO UNIVERSITY OF TECHNOLOGY	BUT	Czech
ROBOAUTO S.R.O.	ROBO	Czech
ESKISEHIR OSMANGAZI UNIVERSITESI	ESOGU	Turkey
KUNGLIGA TEKNISKA HOEGSKOLAN	KTH	Sweden
STATENS VAG- OCH TRANSPORTFORSKNINGSINSTITUT	VTI	Sweden
UNIVERSIDAD DE CASTILLA - LA MANCHA	UCLM	Spain
FRAUNHOFER GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V.	FRAUNHOFER	Germany
SIEMENS AKTIENGESELLSCHAFT OESTERREICH	SIEMENS	Austria
RULEX INNOVATION LABS SRL	RULEX	Italy
NXP SEMICONDUCTORS GERMANY GMBH	NXP-DE	Germany
PUMACY TECHNOLOGIES AG	PUMACY	Germany
UNITED TECHNOLOGIES RESEARCH CENTRE IRELAND, LIMITED	UTRCI	Ireland
NATIONAL UNIVERSITY OF IRELAND MAYNOOTH	NUIM	Ireland
INOVASYON MUHENDISLIK TEKNOLOJI GELISTIRME DANISMANLIK SANAYI VE TICARET LIMITED SIRKETI	IMTGD	Turkey
ERGUNLER INSAAT PETROL URUNLERI OTOMOTIV TEKSTIL MADENCILIK SU URUNLER SANAYI VE TICARET LIMITED STI.	ERARGE	Turkey
OTOKAR OTOMOTIV VE SAVUNMA SANAYI AS - OTOKAR AS	OTOKAR	Turkey
TECHY BILISIM TEKNOLOJILERI DANISMANLIK SANAYI VE TICARET LIMITED SIRKETI - TECHY INFORMATION TECHNOLOGIESAND CONSULTANCY LIMITED COMPANY	TECHY	Turkey
ELECTROTECNICA ALAVESA SL	ALDAKIN	Spain
INTECS SOLUTIONS SPA	INTECS	Italy
LIEBERLIEBER SOFTWARE GMBH	LLSG	Austria
AIT AUSTRIAN INSTITUTE OF TECHNOLOGY GMBH	AIT	Austria
E.S.T.E. SRL	ESTE	Italy
NXP SEMICONDUCTORS FRANCE SAS	NXP-FR	France
BOMBARDIER TRANSPORTATION SWEDEN AB	BT	Sweden
QRTECH AKTIEBOLAG	QRTECH	Sweden
CAF SIGNALLING S.L	CAF	Spain
MONDRAGON GOI ESKOLA POLITEKNIKOA JOSE MARIA ARIZMENDIARRIETA S COOP	MGEP	Spain
INFOTIV AB	INFOTIV	Sweden
BERGE CONSULTING AB	BERGE	Sweden

ECSEL JU, grant agreement No 876852.

Executive Summary

This deliverable is part of WP4, which focuses on designing and implementing tailored V&V process workflows and documents the initial results from Task 4.2 “Initial detailed description of improved process workflows”. Task 4.2 develops specific workflows and solution patterns for verification and validation that address the challenges and goals stated by the industrial use cases. The goal is to bring partner-specific and tool-specific workflows and contributions into a holistic and integrated verification and validation process. One solution strategy is the virtualization of the V&V process by exploitation models, prototypes, and digital twins for dedicated product and process aspects to improve and accelerate quality assurance processes.

As a result, the deliverable D4.3 describes the concepts of partners for virtual validation and virtual prototyping in the VALU3S project. These concepts and solutions are part of the V&V workflows and V&V toolchains. Additionally, basic terms are defined, and challenges from different domains when introducing virtual validation and virtual prototyping are described.

Contributors

Thomas Bauer	FRAUNHOFER IESE
Martin Hrubý	BUT
Maytheewat Aramrattana	VTI
Ricardo Ruiz	RGB
Hamid Ebadi	INFOTIV
Jack Jensen	BERGE
Deborah Hugon	STAM
Davide Ottonello	STAM
Aitor Agirre	MGEP
Stefano Tonetta	FBK
Srajan Goyal	FBK
Thorsten Tarrach	AIT
Arturo S. García	UCLM
José P. Molina	UCLM
Mustafa Karaca	IMTGD
Cem Baglum	IMTGD
Manuel Schmidt	NXP
Martin Karsberg	INFOTIV
Joakim Rosell	RISE
Thanh Bui	RISE
A. Taha Arslan	TECHY
Georgios Giantamidis	UTRCI
Bernhard Fischer	SIEMENS
Martin Matschnig	SIEMENS
Rupert Schlick	AIT
Stylianos Basagiannis	UTRCI
David Pereira	ISEP
José Proença	ISEP
Johnny Öberg	KTH
José Luis de la Vara	UCLM
Pascual González	UCLM
Alim Kerem Erdogmus	IMTGD
Metin Ozkan	ESOGU
Bernhard Fischer	SIEMENS
Martin Matschnig	SIEMENS

Reviewers

Mustafa Karaca	IMTGD	2021-10-04, 2021-10-18
Alim Kerem Erdogmus	IMTGD	2021-10-05, 2021-10-15
Cem Baglum	IMTGD	2021-10-19
Fredrik Warg	RISE	2021-10-07, 2021-10-19
Felix Schulte-Langforth	FRAUNHOFER IESE	2021-10-07
Markus Damm	FRAUNHOFER IESE	2021-10-07, 2021-10-19
Katia Di Blasio	INTECS	2021-10-19
Manuel Schmidt	NXP	2021-10-19
Behrooz Sangchoolie	RISE	2021-10-27, 2021-10-29

Revision History

Version	Date	Author (Affiliation)	Comment
0.1	2020-07-23	Thomas Bauer (FRAUNHOFER IESE)	Initial structure
0.2	2021-09-17	Thomas Bauer (FRAUNHOFER IESE)	Update structure and initial content for chapter 2
0.3	2021-10-04	Thomas Bauer (FRAUNHOFER IESE)	Integration of partner content; Release of document version for 1st review
0.4	2021-10-08	Thomas Bauer (FRAUNHOFER IESE)	Version after first review
0.5	2021-10-11	Thomas Bauer (FRAUNHOFER IESE)	Version for corrections after first review
0.6	2021-10-15	Thomas Bauer (FRAUNHOFER IESE)	Version after second review
0.7	2021-10-24	Thomas Bauer (FRAUNHOFER IESE)	Version for submission to project management
0.8	2021-10-27	Behrooz Sangchoolie (RISE)	Review of the final draft while making minor formatting changes and adding additional comments to be addressed
0.9	2021-10-28	Thomas Bauer (FRAUNHOFER IESE)	Version that addressed the final review comments
1.0	2021-10-29	Behrooz Sangchoolie (RISE)	Review of the final draft while making minor formatting changes.
1.1	2021-10-29	Behrooz Sangchoolie (RISE)	Final version of the report to be submitted.

ECSEL JU, grant agreement No 876852.

Table of Contents

List of Figures

List of Tables

Acronyms

ACC	Adaptive Cruise Control
AD	Autonomous Driving
ADAS	Advanced Driver-Assisted System
ADS	Automated Driving System
AI	Artificial Intelligence
AR	Augmented Reality
CV	Computer Vision
DT	Digital Twin
ECU	Electronic Control Unit
FDD	Failure Detection Diagnosis
HAD	Highly Autonomous Driving
HIL	Hardware-In-the-Loop
ISO	International Organization for Standardization
LIDAR	Light Detection And Ranging
LTL	Linear-time Temporal Logic
MIL	Model-In-the-Loop
ML	Machine Learning
NMT	NeuroMuscular Transmission
RADAR	Radio Detection And Ranging
ROI	Return on Investment
SCP	Safety, Cybersecurity, and Privacy
SDC	Silent Data Corruption
SIL	Software-In-the-Loop
SONAR	Sound Navigation And Ranging
TRL	Technology Readiness Level
VR	Virtual Reality
V&V	Verification and Validation
WP	Work Package

Chapter 1 Introduction

The complexity of systems continues to increase rapidly, especially due to the multi-level integration of subsystems from different domains into cyber-physical systems. This results in particular challenges for the efficient verification and validation (V&V) of these systems with regard to their requirements and properties. In order to tackle the new challenges and improve the quality assurance processes, the V&V workflows have to be documented and analysed. The large-scale European research project VALU3S deals with the improvement and evaluation of V&V processes in different technical domains, focusing on safety, cybersecurity, and privacy properties.

1.1 Scope

In Task 4.2, workflows and solution patterns for verification and validation are developed, which address the challenges and goals stated by the use cases. The goal is to bring partner-specific and tool-specific workflows and contributions into a holistic and integrated verification and validation process. Virtual validation and virtual prototyping are promising concepts for enabling early and efficient quality assurance in complex product development processes.

Within the scope of Task 4.2 and deliverable D4.4, KPI-4 of the project proposal is being addressed, which deals with the development of at least 13 novel tailored V&V workflows that will improve the time and cost of V&V processes. D4.3 supports this KPI by defining concepts for specific validation solutions, which also contain workflows for conducting V&V activities.

The concepts of virtual validation and virtual prototyping solutions will be further extended and implemented by the partners. The results will be presented in D4.11 - *Virtual engineering simulation-based verification, for coupling tools, and enabling missing drivers and environment* data in M30.

1.2 Document Structure

This document is structured as follows: Chapter 2 defines terms and describes challenges for virtual validation and virtual prototyping. Concrete validation approaches are described in Chapter 3. Chapter 4 provides a summary of the deliverable.

Chapter 2 Definitions and Challenges

This chapter provides definitions for the relevant terms, the outcomes, and challenges for virtual validation and virtual prototyping, and the integration of virtual validation and virtual prototyping in the development and test processes.

2.1 Definitions of Terms

This section contains definitions for the relevant terms: Virtual Validation, Virtual Prototyping, and Digital Twins and relates them to each other. The validation of technical software-intensive systems requires dedicated tool chains and execution platforms. By exploiting virtualization and specific models for describing properties of the system under test and its environment, validation can be advanced in time.

2.1.1 Virtual Validation

Virtual validation uses a fully virtual set-up for the validation, i.e., all components of the validation scenarios, the test object and its environment are realized by software or executable models [1]. For complex technical products, the validation of hardware and mechanical parts may incorporate the 3D version of a product created with the help of CAD and CAE to inspect validity, which helps companies to "plan, design and implement their systems before investigating significant capital" [2]. It causes an improvement in costs, time-to-market and the product or machine performance [2]. Although Virtual Validation is also today indispensable, it is expected to become even more crucial for the competitiveness and ROI of companies in future due to the increasing complexity of regulatory compliance and required proofs for certification [2]. Due to its virtual nature, varied combinations of different environments and conditions can be tested with Virtual Validation. Furthermore, modifications and updates require repeated testing, which would be unmanageable in physical validation. The validation in a fully virtual set-up is essential for testing novel technical features like autonomous driving or update-over-the-air of vehicle software components.

2.1.2 Virtual Prototypes

For the development of technical systems prototype, a product, e.g., early samples and product releases, must be built to validate new concepts, features, and processes. A Virtual Prototype is defined as "an executable software model of a hardware system that runs on a host computer" [3]. It simulates the hardware properties at the required level for the software development and abstract hardware details not relevant for the software development [3]. Thus, virtual prototypes provide functional representations of the target systems on which the software parts can be deployed, executed, and validated [3]. The construction and validation of a virtual prototypes is called Virtual Prototyping [4]. Virtual Prototyping is defined as "a computer simulation of a physical product that can be presented,

analysed, and tested from concerned product life-cycle aspects such as design/engineering, manufacturing, service, and recycling as if on a real physical model" [4].

Virtual prototypes can be shared between software and hardware teams to accelerate and parallelize the work in different teams [5]. By substituting the use of a physical prototype using a virtual prototype, costs and time are reduced, while the degree of flexibility increases due to the ease of modification. Re-using earlier developed software can reduce the time for development even more [5]. For this reason, virtual prototypes are often used for complex designs, which would cause the equivalent physical prototype to be very costly and time-consuming [5].

In addition to Virtual Prototyping also Interactive Virtual Prototyping and Mixed Prototyping exist, especially in User Testing [6]. While Virtual Prototypes enable the user to evaluate the design in a visual and auditive way, an Interactive Virtual Prototype includes more senses, e.g., the sense of touch. Mixed Prototyping combines the use of virtual prototypes with real, physical prototypes. Whereas the physical one is less modifiable, it serves to get closer to a realistic insight.

2.1.3 Digital Twins

The concept of Digital Twins goes beyond the usage of specific simulation models and tool chains for the early virtual validation. Digital Twins (DT) are virtual executable models that are fully consistent with selected properties of physical entities or processes in the real world and used for representing, monitoring, and controlling these real-world entities or/and processes [7].

The difference between Digital Twins and simulation models is the following: the digital twin will change in real-time according to the change of state of the physical entity, which is an inversion and mapping of the current state of the physical entity. Simulation is based on the existing entity to build a model and to simulate the possible state of the physical entity in advance, and analyse the possible state change of the physical entity based on the simulation result. Therefore, the Digital Twin is a model, a description of things, while the simulation is based on the modelling of various activities. There is a big difference between the simulation and Digital Twins, and should not be confused with each other [8].

By using the virtual version for design, test, manufacture, and use, the Digital Twin leads to reduced costs, risks, and time. Moreover, it can prevent humans to get injured and avoids the waste of resources. The inherent value of a Digital Twin resides in the capability, for human managers, to exploit such a digital representation to perform what-if-analysis and to perform reliable simulations that help in making sense of the growing complexity and serve as an invaluable decision-making support [7].

The concept of Digital Twins has been applied and adopted to various domains discussed in the following.

Analysed at its roots, NASA pioneered the concept of twins in the 1960s with the Apollo project where two identical spacecrafts were built, with the one on Earth called twin which reflected (or mirrored) the status of spacecraft on mission. By the early 2000s, the approach had taken hold in the manufacturing sector. The abundance of data and monitoring is of little help for decision-making because human controllers are easily overwhelmed by the data and need help to make sense of the raw data to take

informed decisions. For this reason, its concept came into being in relation to Product Lifecycle Management (PLM) in 2002 by Michael Grieves [7], whereupon it was referred to as Mirrored Spaces Model' and 'Information Mirroring Model. The term "digital twin" was introduced only about a decade ago in NASA's draft version of the technological roadmap in 2010 [9]. There, DT was referred to as 'Virtual Digital Fleet Leader' and described as "an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin".

In general, Digital Twins possess three main characteristics [10]:

- "The **linked collection** of the relevant digital artifact […]",
- "**Evolves along with the real system** along the whole lifecycle […]" and
- "Is not only used to describe the behaviour but also to **derive solutions relevant for the real system**."

According to [11] the term Digital Twin in the industrial automation domain is defined as: "…a virtual double of a product, a machine, a process or of a complete production facility. It contains all the data and simulation models relevant to its original. Digital twins not only enable products to be conceived, simulated, and manufactured faster than in the past, but also to be designed with a view to improved economy, performance, robustness, or environmental compatibility. The virtual twin of a product can also accompany it like a digital shadow through all the stages of the value chain – from design through production to operation to servicing and even recycling. It seamlessly and ideally links together the three Ps: product, production, and performance."

Considering different intentions, there are three digital twin types in industrial automation [10]:

- **Digital product twin**, which includes all design artifacts of a product
- **Digital production twin**, which covers models for the manufacturing process and production system itself
- **Digital performance twin**, which derives insight from utilization data and analyses actual performance

Overall, a DT must be:

- A model of the system and its physical environment, which can be simulated and analysed to reason about the current and potential future scenarios of the actual system.
- Connected to the actual system for continuous feedback of data about the system and the physical environment.
- Adaptable to react to changes in the physical asset and modify model parameters accordingly.

Many research works discussed the connotation, definition of the digital twin concept, independent of industry field. One common aspect of most definitions of DT other than being a virtual representation of the physical world is the bidirectional transfer or sharing of data between the physical counterpart and the digital one, including qualitative and quantitative data, historic data, environmental data, and most importantly, real-time data [12]. As such, the Digital Twin is both a collection of algorithms and a

data structure of high fidelity. It integrates multiple physical fields, including the real working conditions of physical entities, materials, and data obtained through real-time sensors [13].

2.2 Outcomes and Challenges Addressed by Virtual Validation and Virtual Prototyping

Virtual validation and virtual prototyping have become part of novel quality assurance processes and applied to several technical domains. By using virtual validation and virtual prototyping solutions, specific challenges in different domains are addressed, which are described in the following subsections.

2.2.1 Automotive-Specific Challenges and Use Case

In the automotive domain, virtual prototypes have been widely used to accelerate design and development of systems in a vehicle, and the vehicle itself. Especially nowadays, when original equipment manufacturers (OEMs) are on a race towards rapid development of new vehicles and their functionalities. Furthermore, in the context of intelligent transportation systems (ITS), virtual prototyping is no longer limited to vehicles, but all the ecosystems around it (e.g., surrounding traffic, road network, road infrastructure, etc.), due to needs to verify and validate new advanced services from the perspectives of the transportation systems. This is sometimes referred to as a "digital twin" in smart cities context.

Virtual prototyping allows system developer to save time and cost in their development processes. Having virtual prototypes of all components enables V&V processes to start in early development phases, which is more efficient in terms of cost and time, compared to testing them after they have been produced. A system can be tested to initially verify its feasibility and functionality even before it is built. Moreover, some systems in automotive domain involve several vehicles, and thus it is very costly to have access to several vehicles for verification and validation of the systems. Lastly, verification and validation of safety-critical automotive systems are often too dangerous or costly to be repeated several times in real-world. Virtual prototyping and validation allow repeatable tests of these systems in a safe environment, which is the virtual environment.

One of the most important challenges related to virtual prototyping in automotive domain is to address increasing complexity of future transportation systems and vehicles. Given complexity and interactions between systems within a vehicle, it is already challenging to create a virtual prototype for a complete vehicle. It is then extremely difficult to create a single prototype that has enough fidelity and contains all aspects of the transportation system. Although virtual prototypes of most components are thoroughly developed already during their individual development processes, each virtual prototype is often created using specialized tools, and they are verified and validated using specialized simulation environment.

To properly conduct virtual validation, two main challenges in the automotive domain are: 1) lack of common approaches to integrate existing virtual prototypes; and 2) lack of methodologies to utilize existing virtual prototypes together in a complete system verification and validation.

2.2.2 Industrial Automation-Specific Challenges and Use Case

The Industrial Automation and Robotics domain benefits of and drives technological acceleration. Tight market competition scenarios and short innovation cycles constantly force industry to rapidly adapt to new methods, tools, and processes. The concept of a smart digital factory based on modern automation approaches such as decentralized decision-making, autonomous machinery, machine-to-machine communication, and artificial intelligence leads to highly interconnected systems, from enterprise resource planning systems (ERP) down to the industrial internet of things (IIoT) and edge devices. Next to purely functional design challenges, many other attributes must be considered, for example availability, reliability, maintainability, or quality attributes such as safety, cybersecurity, privacy, and performance to reduce risks, optimize designs, and support safety and security standards compliance early in the product life cycle. Overall, smart factory design is a highly demanding activity, often involving large investments and the expectation of high returns. Virtual prototypes and digital twins address many of these challenges by the digitalization of real-world objects supporting verification and validation.

The major challenge in digitalization lies in reaching satisfactory simulation performance for complex smart factory designs, and their verification and validation. The large variety of different models (e.g., discrete/continuous) and their abstraction levels (system-level to integrated circuit level) requires the concurrent execution of heterogeneous simulations. Solutions for coupling and synchronizing simulations are already available, however, real-time simulation is barely possible.

Another important objective for virtual prototypes regards model and simulation accuracy. The latter should be sufficiently high to reach design goals without causing significant drawbacks on simulation performance. Matching accuracy is especially critical for the verification and validation of safety scenarios.

Debugging various models executed by heterogeneous simulations on distributed host machines requires a lot of effort and expert knowledge – Currently, this is an inconvenient situation for developing systems.

Designing new products or enhancing (upgrading) existing ones usually leads to the exchange of components with their newer counterpart. Due to the fast-paced innovation cycles, there is a large probability that part replacements happen on a regular basis. Exchanging in-place components (e.g., an IC circuit component) causes additional effort for redesigning verification environments and updating interfaces of virtual prototypes (digital twins) for seamless concurrent simulation.

Even small methodological and technological progress for the above-mentioned challenges could ease the design, verification, and validation of cyber-physical systems in the industrial domain.

2.2.3 Medical and Health-Specific Challenges and Use Case

Virtual prototypes are very popular in many domains of industry and engineering, as they significantly accelerate the development and reduce all costs. In the case of medical devices, purpose of virtual prototypes goes even further and becomes practically the only suitable approach for design and

development of new medical devices. For many ethical and medical reasons, development of medical devices using real respondents is impossible and even prohibited. Specifically, in cases of devices that actively affect physiological processes in human body (e.g., by intravenous administration of drugs), or in cases of monitoring and diagnostics that are potentially painful or generally dangerous.

Practical deployment of medical devices require their prior accreditation by national health agencies such as U. S. Food and Drug Administration (FDA) or European Medicines Agency (EMA). An applicant needs to prove the medical effect and safety of the device, which is typically based on numerous studies and experiments on a corresponding virtual prototype of the device.

Lack of clinical data is the main difficulty during development of a medical device. We need practical measurements of pharmacological effects of drugs if we develop a medical device intended to automatize their administration. For instance, in clinical anaesthesiology, these drugs control patient's blood pressure, neurotransmission, pain reduction etc. Practical experiments published in the medical literature (certainly under very strict ethical and medical circumstances) are usually very limited in the number of respondents, thus making it difficult to extrapolate the findings into a general mathematical pharmacokinetic/pharmacodynamic model.

For this reason, the main challenge in this domain is to substitute missing clinical data with massive experimenting using the virtual prototype of the device and appropriate mathematical model of the patient. Large experimenting with a device's virtual prototype gives us necessary validation of the device before any clinical experiments become thinkable.

2.2.4 Aerospace-Specific Challenges and Use Case

Digital twins are well known in the space domain. As described above, NASA pioneered the concept of twins in the 1960s with Apollo project where two identical spacecrafts were built, with the one on Earth called twin which reflected (or mirrored) the status of spacecraft on mission.

Soon after the publication of technological roadmap by NASA on digital twin in 2010 [9], the US Air Force used DT technology for the design, maintenance, and prediction of their aircraft [14], [15]. The idea was to use DT to simulate physical and mechanical properties of the aircraft to forecast any fatigue or cracks in the structure, thus prolonging the remaining useful life of the aircraft. According to the digital twin paradigm jointly proposed by NASA and US Airforce in 2012 [16], “Digital twin is a life management and certification paradigm whereby models and simulations consist of as-built vehicle state, as-experienced loads and environments, and other vehicle-specific history to enable high-fidelity modelling of individual aerospace vehicles throughout their service lives.” Essentially, digital twin would reduce uncertainty by incorporating into available models as much initial and in-service information as possible.

Many future vehicles will have little direct precedent to follow, and in most space missions, the inspection of vehicle components after going into service is extremely difficult and impossible in some long-duration flight cases. Thus, the ability to fully understand degradation and anomalous events and foresee previously unknown unknowns may represent the difference between mission success and

mission failure. The digital twin method will provide a capability to predict and prevent safety issues by constantly monitoring usage and simulating possible future damage states [17].

In the domain of structural health management, future generations of aerospace vehicles will be lighter in mass while being subjected to higher loads and more extreme service conditions over longer time periods than the present generation of vehicles. As a result, demands on structural materials will be greatly increased while the structural margins will be decreased. Additionally, demands for long-term reliability will increase. The vehicles are likely to encounter conditions that cannot be foreseen. Thus, the ability to modify and evaluate the modification of mission parameters in near-real time will be required [16].

Digital twin will provide an individualized approach to inspection, repair, and replacement. Automated monitoring will enable detection of unforeseen damage initiation in real-time and provide the digital twin with updates of actual usage and damage states. Such preventative and individualized condition-based inspection intervals will significantly reduce the costs by minimizing unnecessary inspections and early retirement of parts [17].

A major challenge is to apply the digital twin concept to robotic rover systems. The communication architectures of extra-terrestrial exploratory rovers and spacecrafts are similar, whereby they receive telecommands and send telemetry data. However, rovers are commanded by an operator, who needs contextual awareness about the evolving environmental conditions of the rover. Future rovers are expected to have the capability of autonomous decision-making and control based on local information and less dependence on frequent human intervention even in unexpected situations that are likely to occur [9]. The adaptation of models for simulation, planning, diagnosis, and mission management is the target of the ROBDT project [18].

2.2.5 Use Cases in VALU3S

In the project, 12 use cases are given and addressed. Table 2.1 shows domain and providers of all VALU3S use cases. Use case #12 have been removed from the table since the corresponding use case provider has terminated its participation to project.

Table 2.1 Use Cases in the VALU3S Project

Use Case ID	Use Case	Domain	Use Case Provider
1	Intelligent Traffic Surveillance	Automotive	CAMEA
2	Car Teleoperation	Automotive	ROBO
3	Radar system for ADAS	Automotive	NXP
4	Human-Robot-Interaction in Semi-Automatic Assembly Processes	Industrial robotics/ automation	PUMACY
5	Aircraft engine controller	Aerospace	UTRCI
6	Agriculture robot	Agriculture	ESTE

Use Case ID	Use Case	Domain	Use Case Provider
7	Human-Robot Collaboration in a disassembly process with workers with disabilities	Industrial robotics/ automation	ALDAKIN
8	Neuromuscular Transmission for muscle relaxation measurements	Healthcare	RGB
9	Autonomous train operations	Railway	CAF
10	Safe function out-of-context	Railway/ Multi-domain	BT
11	Automated robot inspection cell for quality control of automotive body- in-white	Industrial robotics/ automation	OTOKAR
13	Industrial Drives for Motion Control	Industrial robotics/ automation	SIEMENS

2.3 Integration of Virtual Validation and Virtual Prototyping in the Development and Test Processes

The use of virtualization and platforms for early validation with dedicated models and tools leads to changes in the development and test processes. Development processes distinguish construction and quality assurance activities and represent them in separate paths. The outputs of every construction activity, such as requirements, architecture design models, or source code, must be verified by appropriate quality assurance activities. In the early process stages, analysis activities, such as requirements inspections or static interface analysis, are conducted. Testing is performed when executable artifacts, such as executable software units or subsystems, become available. Test activities are usually split into several phases, each tackling a specific abstraction level with dedicated test objects such as software, control units, or fully integrated systems. Typical test phases comprise the testing of single components, integrated subsystems, and complete systems. In model-driven development processes, the architectural and design stages may additionally deliver executable models, which can be exploited for testing.

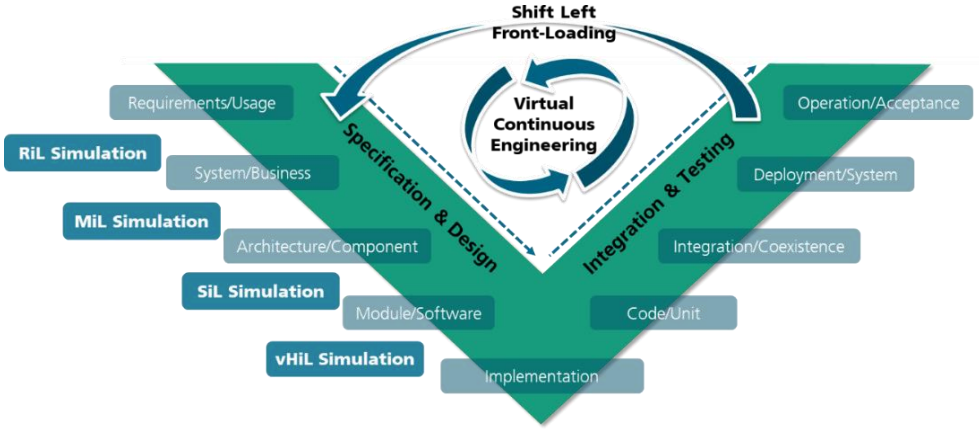


Figure 2.1 Development process with virtual validation stages

By introducing virtual prototypes along the development process, simulation at different levels of abstraction of the architecture can be used to enable precise and fast feedbacks and validation results.

An updated developed process with dedicated virtual validation stages on different levels is shown in Figure 2.1, which has been derived from [3]. Concrete validation stages with virtual prototypes comprise [19]:

Requirements in the Loop Simulation (RiL). This phase aims at the verification of system requirements with respect to functional adequacy, completeness, consistency, and robustness by means of larger simulation scenarios. RiL requires a rather complete formal specification of the relevant system properties as executable models at an appropriate level of level of abstraction, such as the description of the input-output behaviour of new system functions. It represents the earliest possible detection of specification flaws through intensive validation [19].

Model in the Loop Simulation (MiL). The stage enables the verification of the quality of architecture design decisions on different levels in order to check whether the architecture design is suitable for fulfilling the system requirements. At this stage, no hardware or software elements are considered, but only the interfaces and interactions between the functional and logical architecture elements [19].

Software in the Loop Simulation (SiL). The phase aims at verifying to which extent the implemented software realizes the functional and logical architecture specifications. Examples of these are architecture specifications of software components, packages, classes, and interfaces [19].

Virtual Hardware in the Loop Simulation (vHiL). The stage considers dedicated models for virtual hardware in order to verify the adequacy of software deployment on control units and the functional realization of dedicated hardware functions. Virtual prototypes on the vHiL stage use dedicated simulation modelling notations such as SystemC/TLM, which are very close to the real hardware and allow the validation of hardware-specific properties by a purely virtual set-up [19].

Hardware in the Loop Simulation (HiL). This step covers the execution of validation scenarios on real hardware. In combination with virtual prototypes and the validation results from the vHiL stage, HiL tests can be improved to consume less resources and time, because critical hardware issues have already been identified in the previous vHiL phase [19].

Chapter 3 Virtual Validation and Virtual Prototyping Approaches in VALU3S

This chapter describes the approaches for virtual prototyping and virtual validation of the VALU3S partners.

3.1 Contract-Based Virtual Integration [FBK]

Name of the virtual validation approach: Contract-Based Virtual Integration
Introduction Contract-based specification of components enable the analysis of the virtual integration of components by reasoning about the relationship between their assumption and guarantees. In the design of embedded systems, contracts are typically specified in temporal logic to formalize functional, real-time, safety, and dependability aspects of system and components.
Outcomes and features
Current features The main advantages of the contract specification are the followings: • The architectural structure of components can be validated and verified in the early design phases, when components are not yet implemented or modelled in detail. • Contracts enable a compositional reasoning where leaf component implementations are verified in isolation while their composition is virtually verified by meta reasoning. This results in a higher scalability of the verification. • Libraries of components can be verified upfront so that reuse of components is achieved and assured by comparing formally the contracts in the architecture with those verified on the components.
Application domains and use cases
Domains In the last decade, contracts specification for embedded systems have been applied in various domains including automotive, aerospace, railways, and robotics.
Tool Chain
List of tools • One of the tools that supports the formal analysis of contracts specified in temporal logic is OCRA [CB1]. OCRA provides functionalities for checking contract refinement, contracts validation, model checking the behavioural implementation of contracts, and contract-based fault-tree generation. OCRA's input is a textual specification of components, contracts, and their refinement. • CHESS [CB2] is an Eclipse-based tool supporting the specification of contracts on SysML diagrams. It integrates OCRA as backed to provide the above-mentioned functionalities ad the level of SysML.
Tool interfaces FBK developed the Eclipse plugin SDE [CB3] for wrapping OCRA functionalities. SDE provides interfaces for an Eclipse-based integration, as well as OSLC-based interfaces for a web-based interaction with OCRA, as shown in Figure 3.1.

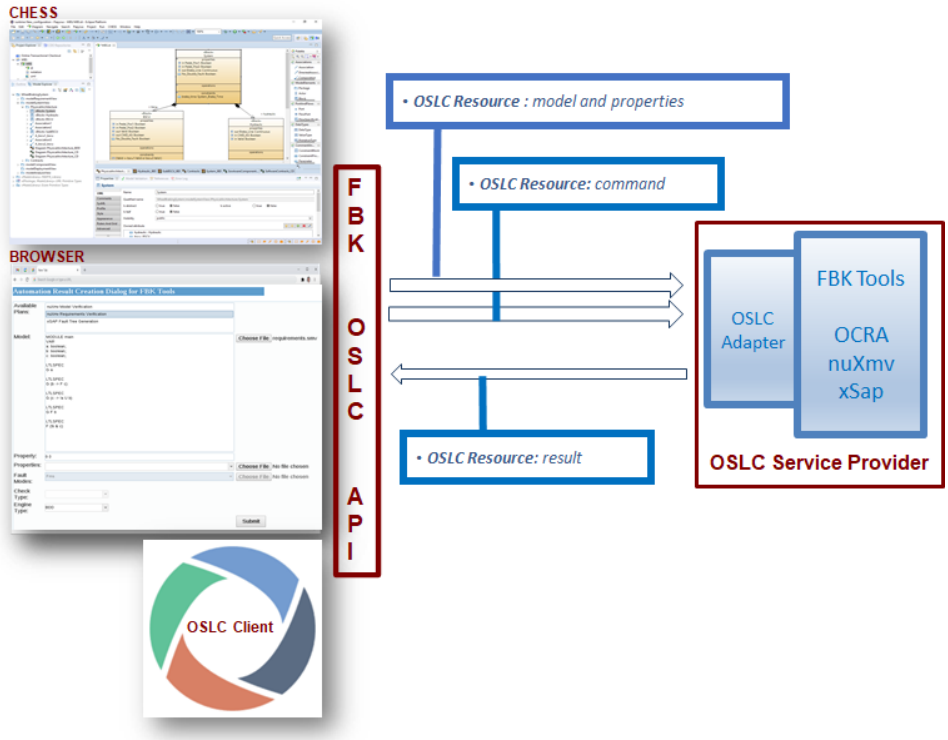


Figure 3.1 Tool interfaces of contract-based virtual integration

Models, notations, and formats
List of model types and modelling notations The OCRA input language is textual specification of components and contracts. Contract assumptions and guarantees are specified in an extension of Linear-time Temporal Logic (LTL). The extension comprises SMT predicates (e.g., expressions over integers and reals) and timing/hybrid aspects. In fact, the logic is parametrized by the model of time and can include clocks and continuous-time variables. OCRA contracts specification has been integrated in higher-level languages such as SysML. The behavioural specification of components can be specified in SMV (the input language of model checkers NuSMV and nuXmv), in the HyDI (an extension of SMV for hybrid systems).
Support for heterogeneous models and prototypes In general, the specification of components can be heterogeneous as far as the implementation is syntactically compatible with the component interface specified in the system architecture.
V&V Process
Description of the V&V process Once the contracts refinement is specified in the input language, the verification process is automated: internally, the tool generates a set of proof obligations, i.e., temporal formulas that are valid if and only if the refinement is correct. The proof obligations are discharged by a model checking algorithm.
Input information and pre-conditions The contracts refinement must be specified in the input language of OCRA.
Construction approach for models and prototypes Firstly, the component interface is specified in terms of input/output ports. Secondly, the component interface is enriched with the contract specification. Thirdly, the component is decomposed into subcomponents, the ports are connected, and the contract is linked to the contracts of the subcomponents.
Output information

The result shows an OK/NOT OK answer. If a proof obligation is not valid (NOT OK result), also a counterexample is generated, i.e., an execution of the component and its subcomponents violating the proof obligation (e.g., where all contracts of the subcomponents are satisfied but the contract of the parent component is violated).
Requirements' traceability Traceability is maintained by the CHESS tool. Requirements are linked to the relevant component, and to formal properties, which formalize the requirements in terms of the ports/variables of the component. Formal properties are then combined to form the component contracts.
Quality properties
Safety-related properties OCRA supports safety-related properties that can be expressed into the extended LTL. Furthermore, contract-based fault-tree generation can be used for safety assurance.
Cybersecurity-related properties OCRA supports security-related properties that can be expressed into the extended LTL.
Privacy-related properties OCRA supports privacy-related properties that can be expressed into the extended LTL.
Further quality properties Other properties such timing properties can be expressed in the timed extension of LTL supported by OCRA.
References
Links to VALU3S deliverables N/A
Links to V&V workflows N/A
Links to resources N/A
Links to tools • [CB1] OCRA https://ocra.fbk.eu/ • [CB2] CHESS https://www.eclipse.org/chess/ • [CB3] Eclipse plugin SDE https://gitlab.fbk.eu/ESProjects/SDE
References Contract theories for embedded systems • Albert Benveniste, Benoît Caillaud, Roberto Passerone: A Generic Model of Contracts for Embedded Systems. CoRR abs/0706.1456 (2007) • Albert Benveniste, Benoît Caillaud, Alberto Ferrari, Leonardo Mangeruca, Roberto Passerone, and Christos Sofronis. Multiple Viewpoint Contract-Based Specification and Design. FMCO 2007. • Werner Damm, Hardi Hungar, Bernhard Josko, Thomas Peikenkamp, Ingo Stierand: Using contract-based component specifications for virtual integration testing and architecture design. DATE 2011: 1023-1028 • Manfred Broy: Towards a Theory of Architectural Contracts: - Schemes and Patterns of Assumption/Promise Based System Specification. Software and Systems Safety - Specification and Verification 2011: 33-87 • Sebastian S. Bauer, Alexandre David, Rolf Hennicker, Kim Guldstrand Larsen, Axel Legay, Ulrik Nyman, Andrzej Wasowski: Moving from Specifications to Contracts in Component-Based Design. FASE 2012: 43-58 • Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto L. Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger, and Kim G. Larsen. Contracts for Systems Design. Rapport de recherche RR-8147, INRIA, 2012

- Alessandro Cimatti, Stefano Tonetta: Contracts-refinement proof system for component-based embedded systems. Sci. Comput. Program. 97: 333-348 (2015)

Tool support

- Darren D. Cofer, Andrew Gacek, Steven P. Miller, Michael W. Whalen, Brian LaValley, Lui Sha: Compositional Verification of Architectural Models. NASA Formal Methods 2012: 126-140
- Alessandro Cimatti, Michele Dorigatti, Stefano Tonetta: OCRA: A tool for checking the refinement of temporal contracts. ASE 2013: 702-705
- Pierluigi Nuzzo, Michele Lora, Yishai A. Feldman, Alberto L. Sangiovanni-Vincentelli: CHASE: Contract-based requirement engineering for cyber-physical system design. DATE 2018: 839-844

3.2 HiL Testbench Platform with NMT Controller/Simulator/TCM [RGB]

Name of the virtual validation approach: HiL Testbench platform with NMT Controller/Simulator/TCS

Introduction

The purpose of this tool is to substitute efficiently experimental/research/development platform needed for development of an NMT (NeuroMuscularTransmission) controller that regulates the relaxant drug delivery in order to keep the patient in the pre-established target. This way, presence of a real (human) patient is not required and a large number of automated experiments could be conducted. The main challenge in virtual validation and prototyping is managing simulation of a complex heterogeneous system consisting of hardware and software components reacting to the behaviour of a human body.

Goals

VALU3S will use the testbench platform as a NMT controller simulation tool with the purpose to define the best possible control strategy at laboratory level, so that once the control algorithm is fine-tuned, it can be integrated as a new functionality to the RGB NMT monitor for the automatic closed-loop control of the NMT monitored vital sign.

General set-up

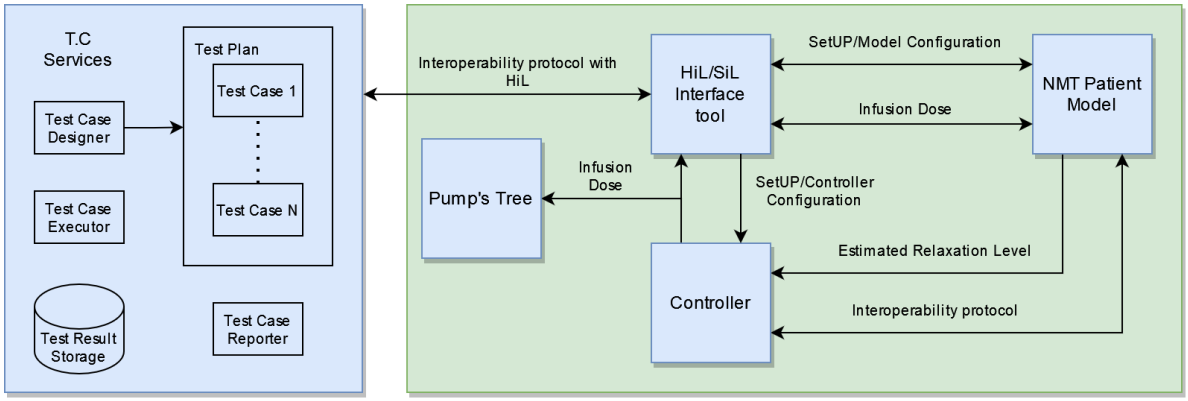


Figure 3.2 Tool interfaces of HiL Testbench platform with NMT Controller/Simulator/TCM

Figure 3.2 illustrates the general set-up i.e., tool interfaces of HiL Testbench platform. Test Case System (TCS) is a tool for managing automated tests above our distributed simulation platform that combines hardware and software components (HiL – Hardware in the loop). We plan to verify various features of the anaesthesiologic medical device, such as:

- Safety – over-dosing the patient must be impossible,
- Efficiency – infusion process causes the shortest total recovery time of the patient and minimalistic dosage.
- Stability – neuromuscular blockade remains within the predefined range during the surgery.

In the general setup, TCS generates simulation experiments based on previously defined test case scenarios. Recorded simulation data are statistically processed.

The Verification Plan shall be executed as a test plan execution flow, that consist of the following steps:

1. Test Plan, Test Case and Test Case parameters, which will be created in the Test Case Services Provider TCSP.
2. HiL/SiL gets a Test Plan from the Requirements tool. Every Test Plan consist of several Test Cases, every Test Case contains a set of different parameters.
3. HiL executes all the Test Cases sequentially.
4. HiL initiates the Neuromuscular Transmission –NMT- model developed, with a configuration file that contains the patient´s model set of parameters (e.g., change of patient´s sensitivity to drug, or noise injection).
5. The NMT patient´s model sends to the controller the value of the simulated NMT value.
6. The Controller calculates the new relaxant dose value to inject and sends it to the Infusion pump and to the HiL Manager Tool.
7. HiL Manager tool sends the dose to the NMT model that simulates the NMT reaction to the dose.
8. This closed loop is one in sequence, for a predefined amount of time, e.g., 2 hours. When the test case finishes the NMT model writes the result of the execution in a file that is read by the HiL software.

When all the Test Cases finish their execution, HiL/SiL interface tool can upload the Test Results to the Test Case Service Tool (kind of Quality Manager).

Outcomes and features

Current features

The main outcome in the validation process is in the developed infusion algorithm that satisfy all three features defined above. NMT-Simulator and TCM tools are currently working in prototype version. This method provides a general platform for experimenting (injecting failures like errors in NMT measurements, infusion pumps etc).

There will be Key performance parameters in order to make comparisons around different control strategies.

Planned features for the future

The tools that are currently being developed are the NMT controller, the Infusion Pump driver, the NMT-Simulator as the main simulation system and the TCS operating with NMT-Simulator.

Application domains and use cases

Domains

Medical Domain. The works will be applied in anaesthesiology, medical devices

VALU3S Use cases

Use CASE 8 (NMT controller).

Tool Chain

List of tools

The list of tools that are part of the tool chain are all around the REDIS Database system.

Tool interfaces

NMT-Simulator is basically a distributed simulation system. Communication via message sending

and data storage capability needs are both provided by REDIS, which is the main technical interface. Controller, TCS and NMT-Simulator communicates via REDIS too.
REDIS stores data of the currently executed experiment (experiment inputs, infusion setting, physiological quantities of the patient, measure/simulated values of neuromuscular blockade in TOF/PTC and others). PTC is supposed to collect ongoing data, process them statistically and evaluates the experiments.

Simulator coupling, and co-simulation capabilities

TCS defines input conditions for experiments. For each experiment, TCS initiates a new simulation in NMT-Simulator.
REDIS DB is the technical platform for all kinds of interconnection, in the control meaning and data storage meaning.
Technically, the system is deployed in Docker containers, where:

1. REDIS is dockerized
2. NMT-Simulator is dockerized and connected to REDIS
3. TCS is dockerized and connected to REDIS
4. TCS provides a web application for management and visualisation of experiments

Models, notations, and formats

List of model types and modelling notations
The objective will be to provide an environment to be used for the functional verification of the system covering the following points:

- Automatic execution of the test plans.
- Increase requirement traceability incorporating tools into the development process.
- Accelerate design or optimize design cost by using tools to analyse the performance of different potential strategies.
- Increase the safety of the patient by using Hardware in the Loop System to verify the functionality of the controller before clinical validation.

Support for heterogeneous models and prototypes
NMT-Simulator is a heterogeneous system with independent components (patient model is a simulation model, controller is either hardware or simulated component, etc).

V&V Process

Description of the V&V process
The following parameters as well as statistical data should be the result of the performance evaluation of the NMT controller:

- **Control Time**: Indicates the duration of the session, which is graphically presented.
- **Total dose**: Indicates the total amount of drug infused in ml.
- **Average Dose**: Indicates the relation between the total infused dose and the control time in hours.
- **Target NMT level**: Shows the last target NMT level sent by the TOFCuff NMT monitor.
- **NMT in Target**: Indicates the percentage of time that the NMT is in the "target range". It is computed only when the target NMT is available.
- **Mean Error**: Indicates the mean of errors during the control time. The error is defined as the difference between the NMT value and the upper bound of the target range, when this value is positive. This value is computed only if the NMT value is available.
- **Time without infusing**: Indicates the percentage of time that the infusion rate is 0, related to the total control time.

Input information and pre-conditions
For every Test Case, we define:

1. The patient´s data e.g., identification, male/female, weight, height, pathologies
2. The prescribed target NMT value and threshold values for the particular operation
3. Information about the infusion pump

4. The type of control, either manual /automatic, semiautomatic, continuous/Bolus/combined Dose based
Construction approach for models and prototypes There is no specific approach for constructing the model. NMT-Simulator is a heterogeneous system with independent components (patient model is a simulation model, controller is either hardware or simulated component, etc).
Output information The Type of output information, data, results that are produced is: • Tests versus different patient´s sensitivity. • Tests versus Artefacts (Noise injection in NMT) For each Test: • Mean NMT Error /versus type of patient • Percentage of Time without infusing (Dose Equal Zero) • Percentage of Time in target gap • Percentage of time at different levels: Very high, high (above target); Low, Very Low (below target) • Cost function (Mean Drug Dose Value Infused) The TCS will be in charge of compiling, processing and recording results
Requirement's traceability The Testcase System module will keep track of requirements fulfilment.
Quality properties
Safety-related properties The safety-related properties that the tool address is safe infusion of anaesthesiologic drugs during surgery, i.e., assuring infusion without overdosing the patient.
Cybersecurity-related properties N/A
Privacy-related properties N/A
Further quality properties N/A
References
Links to VALU3S deliverables • D3.5 [20] (Sections 2.10, 2.11, 2.14, 2.16, 2.28, and 2.30)
Links to V&V workflows Part of the workflows of UC8.
Links to resources N/A
Links to tools N/A
References N/A

3.3 V2X Network Simulation Framework Based on Veins_INET [VTI]

Name of the virtual validation approach: V2X network simulation framework based on Veins_INET
Introduction

Connectivity between vehicles and their surrounding actors by means of vehicle-to-everything (V2X) communication has enabled several functionalities in the cooperative intelligent transportation systems (C-ITS) context. Such connectivity relies on wireless communication between the actors, which is difficult to manipulate in reality while still maintaining a full control.

Goals

The goal for this approach is to create a model of communication network and related systems in simulation. The models are then intended to be used together with simulation-based fault and attack injection methods in the scope of UC2 in the VALU3S project.

General set-up

This approach includes three main open-source simulation frameworks and tools: Veins, INET, and SUMO. Veins stands for *Vehicle in Network Simulation*; it consists of V2X communication modules according to the IEEE 802.11p standard. INET provides models for internet networks and protocols such as IPv4, IPv6, TCP, UDP, etc., with possibilities to include models for mobile networks such as LTE and 5G. Lastly, SUMO is a road traffic simulator, which is included in the framework to provide a model of how a communication node (vehicle) should move in the simulation. Veins and INET are already compatible, as they are both extensions of the same underlying tool, namely OMNeT++. Furthermore, SUMO and Veins are already integrated via a Traffic Control Interface (TraCI).

Outcomes and features

Current features

This tool intends to integrate fault and attack injection methods into a widely used simulation framework to evaluate effects of faults in wireless communication networks. This allows system developers to evaluate their systems under different wireless communication conditions, which contains faults and attacks. Robustness of the wireless communication network is especially important for the car teleoperation use case within the VALU3S project (UC2). Results from this tool is expected to assist V&V activities of the use case.

Planned outcomes and features for the future

Tools are being extended to integrate and support simulation-based fault and attacks injection (being developed by RISE). Simulation models of the teleoperation system in UC2 is being developed in parallel by VTI.

Application domains and use cases

Domains

Automotive domain.

VALU3S Use cases

Use case 2 (Car teleoperation)

Tool Chain

List of tools

- SUMO: Simulation of Urban Mobility [VT1]
- Veins: Vehicles in Network Simulation [VT2]
- INET framework
- OMNeT++

Tool interfaces

- TraCI: TCP/IP interface between Veins and SUMO
- Veins and INET are both based on OMNeT++, hence the underlying structure in OMNeT++ is used for exchange of data between models in Veins and INET
- All tools are implemented in C++, and thus any other that can be implemented in C++ is potentially available

Simulator coupling, and co-simulation capabilities

The interlink between simulation framework use typical internet protocol (i.e., TCP protocol). Thus, as long as the data format are according to the TraCI definition, any other simulator can also interact

with SUMO. This could be done via Veins, as done in [VT3]. Therefore, this framework can be easily coupled with other simulation frameworks (e.g., see [VT3]).
Each simulator is a standalone module by itself, hence a network connection between each simulator is required to interlink between them.

Models, notations, and formats

List of model types and modelling notations

Models in SUMO:

- *Car-following models*: models describing how a vehicle(s) in a simulation regulate its speed with respect to its preceding vehicle. A specific car-following model for each vehicle can be pre-defined or changed during the simulation. However, only one model can be active at a time for each vehicle. These models are implemented as a class in C++.
- *Lane-changing model*: models describing how a vehicle(s) make decision to change lane taking into account its surroundings and desired speed. A specific lane-changing model for each vehicle can be pre-defined or changed during the simulation. However, only one model can be active at a time for each vehicle. These models are implemented as a class in C++.
- *Road network:* An XML-based description of a road network. The road network can be manually generated or automatically generated from other road network formats such as OpenSteetMap or OpenDRIVE.

Models in Veins:

- *Application layer:* This defines application layer of a communication node. A functionality of a communication node is typically implemented here. One node may have several applications activated. These models are implemented as a class in C++, according to the format defined by OMNeT++.
- *Mobility:* A module related to mobility of a communication node. This corresponding node needs to be created in SUMO and its movement is regulated by the models in SUMO. Through this model, the "application" layer in Veins can manipulate state of the vehicle in different ways, as defined in SUMO. These models are implemented as a class in C++, according to the format defined by OMNeT++.

Models in INET:

- *Transport layer, Network layer, and Link layer:* Different layers of a communication node. Each layer implements different components of a communication node according to the selected protocols and communication medium/standards. These models are implemented as a class in C++, according to the format defined by OMNeT++.

An example of a communication node in Veins_INET is depicted in Figure 3.3. The figure shows an example of model structures inside a communication node, which is separated in to: application layer, transport layer, network layer, and link layer. Each layer could contain one or several models with interfaces between layers. A *Network Description File (NED)* is used to describe structure of the node including modules and their interfaces.

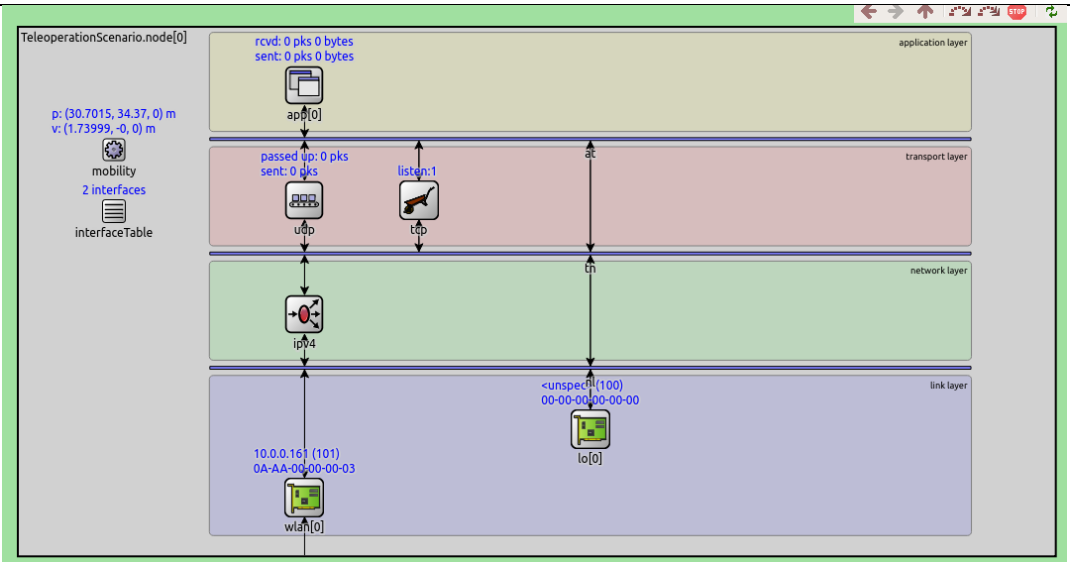


Figure 3.3 Example structure of a communication node in Veins_INET simulation framework

Support for heterogeneous models and prototypes

The tool currently contains a combination of models from communication network domain with automotive (traffic simulation) domain. Creating possibilities to simulate and explore scenarios in the context of C-ITS, connected and automated vehicles (CAVs), etc.

V&V Process

Description of the V&V process

Integration with the "simulation-based fault and attack injection" is planned for this tool chain.

Input information and pre-conditions

Main components that need to be set up in the simulation are:

- Road network (SUMO)
- Specification of vehicle attributes (SUMO)
- Test scenario (Veins_INET)
- Communication modules (Veins_INET)
- Test configuration to run (Veins_INET)

Construction approach for models and prototypes

There is no specific approach for constructing the model. Several standard models that are typically used are already available. Constructing new models and prototypes require understandings of the tools and their formats, which are described in the documentation of the tools.

Interfaces to fault and attack injection methods is not commonly available in such tools, and therefore this project will come up with an approach for the integration.

Output information

Outputs from SUMO are available in XML format, which can easily be further converted to other formats such as CSV files. This output will be used to evaluate the behaviour of vehicles in the simulation and verify if they match the expectations/requirements.

Veins_INET output the data in its own formats (.vec, .sca, and .vci). This is somewhat similar to CSV approach, and can be converted to other formats using standard tools like Python or R. This output is mostly communication network attributes such as delay, number of received packets, number of sent packets, etc. These are mostly used to validate that the injected faults and attacks are implemented correctly and that they occur at a correct time in simulation.

Requirements traceability

This is not available in the current tool chain. This has to be done manually at the moment.

Quality properties

Safety-related properties

The approach can be used to evaluate safety of the system-under-test when there are faults or attacks in wireless communication network, by evaluating impacts on vehicle behaviour.
Cybersecurity-related properties The approach can be used to evaluate cybersecurity of the system-under-test when attacks occur via wireless communication network, by evaluating whether the system is secured enough to detect and/or prevent the attacks.
Privacy-related properties N/A
Further quality properties Quality of service in wireless communication network is potentially possible to derive using this tool.
References
Links to VALU3S deliverables Plan to implement simulation-based fault and attack injection methods developed by RISE (this is described in D3.4 [21]and D3.5 [20])
Links to V&V workflows Part of the workflows of UC2, namely Simulation-based fault and attack injection workflow
Links to resources • https://veins.car2x.org/documentation/modules/#veins_inet
Links to tools Documentation: • Documentation - SUMO Documentation (dlr.de) • Documentation - Veins (car2x.org) • OMNeT++ Documentation (omnetpp.org) • INET Framework Documentation — INET 4.3.0 documentation (omnetpp.org) Source code: • https://github.com/eclipse/sumo • https://github.com/sommer/veins • https://github.com/omnetpp/omnetpp • https://github.com/inet-framework/inet
References • [VT1] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2575-2582, doi: 10.1109/ITSC.2018.8569938. • [VT2] C. Sommer, R. German and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," in IEEE Transactions on Mobile Computing, vol. 10, no. 1, pp. 3-15, Jan. 2011, doi: 10.1109/TMC.2010.133. • [VT3] Maytheewat Aramrattana, Tony Larsson, Jonas Jansson, Arne Nåbo, "A simulation framework for cooperative intelligent transport systems testing and evaluation," Transportation Research Part F: Traffic Psychology and Behaviour, Volume 61, 2019, Pages 268-280, ISSN 1369-8478, https://doi.org/10.1016/j.trf.2017.08.004.

3.4 Simulation-Based Robot Verification [IMTGD]

Name of the virtual validation approach: Simulation-based Robot Verification
Introduction Simulation-based robot verification provides simulations of autonomous system operation in a virtual environment, which resembles the real environment where the system will operate. The created test suite provides a test scenario where different tests can be created in a virtual environment to monitor an autonomous system. In a physical simulation environment, behaviours of an

autonomous system can be monitored, and safety of robots can be verified without causing any environmental hazard. In this method, it is aimed to observe robot behaviour in a simulation environment with focus on enabling safety in autonomous operations and detecting anomalies.
Outcomes and features
Current features The purpose of simulation-based robot verification is assuring robot's trajectory safety for body in-white inspection systems. With simulation-based fault injection, system performance and system behaviour will be observed. Through observations, the safety of the robot system is verified.
Planned outcomes and features for the future It is aimed to integrate to simulation-based testing system with source code mutation tool IM-FIT in order to verify robotic systems.
Application domains and use cases
Domains Industrial Robotics and Automation Domain
VALU3S Use cases Use Case 11 - Automated robot inspection cell for quality control of automotive body-in-white
Tool Chain
List of tools • IM-FIT (Software-based Fault Injection Tool) • SRVT (Simulation-based Robot Verification Tool) • ROS (Robot Operating System)
Tool interfaces • Fault Library - For IM-FIT fault injection (the structure of the file is .json to read and write the data for workloads, faults can apply ROS Publisher/Subscriber/Services, can use YAML file type to read the task data, PyQt for UI) • Fault injection Plan - Plan is sent to SRVT for simulation execution • Data / Log Files – After execution, simulation data is sent back
Simulator coupling, and co-simulation capabilities
Gazebo and MoveIt planner can be integrated for route planning and simulation. To enable this integration, on the physical side; robot models, physical properties and inertia forces should be matched. On the Software side interoperability of both systems should be enabled.
Models, notations, and formats
List of model types and modelling notations Physical Models in STL format.
Support for heterogeneous models and prototypes SRVT can be integrated with ROS compatible modelling and route planning software.
V&V Process
Description of the V&V process The simulation-based testing approach has a first step of source code and workload selection. In selected source code and workload, mutations will be made, and fault scenarios will be run in the simulation environment to assess safety of the system in erroneous situations.
Input information and pre-conditions For IM-FIT: Python Source Code, Workload file (.json), User Requirements, User specified code snippets (.json) For SRVT: Robot Models, Scenario Specific World Models, MoveIt parameters
Construction approach for models and prototypes Our approach is based on importing real world and robot models into a realistic simulation environment. It is based on the operation of these systems as in real environments.
Output information System Verification Report

Requirements traceability First, the identified requirements will be listed, and then updates on GitHub will show level of success to provide these requirements.
Quality properties
Safety-related properties IMTGD will analyse the durability of the software developed for robotic systems with the IM-FIT software tool it has developed and estimate how safe and robust it is in real environments.
Cybersecurity-related properties N/A
Privacy-related properties N/A
Further quality properties N/A
References
Links to VALU3S deliverables • D3.1 [22] (Section 3.2.1) • D3.3 [23] (Section 3.2.1) • D3.4 [21] (Section 2.26)
Links to V&V workflows Part of the workflows of UC11.
Links to resources • https://github.com/inomuh/imfit.git
Links to tools • Source code repository: https://github.com/inomuh/imfit.git
References • Erdogmus, A. K., Karaca, M., Yayan, U., & Cokunlu G. (TOK 2021). Development of Simulation based testing for automated robot cell for quality inspection of automotive body-in-white system • Erdogmus A. K., & Yayan, U. (JSTER 2021). Industrial Robot Motion Planning Algorithms Performance Benchmarking • Erdogmus, A. K., Karaca, M., & Yayan, U. (2021). Manipulation of Camera Sensor Data via Fault Injection for Anomaly Detection Studies in Verification and Validation Activities For AI. arXiv e-prints, arXiv-2108.

3.5 Multiverse Simulation in Robotics [TECHY]

Name of the virtual validation approach: Multiverse Simulation in Robotics
Introduction This approach is an extension to the method described in Section 3.4. The goals are running multiple simulated environments at once in which different faults may be injected separately while several parameters are observed, data are collected, and analysis of each case is conducted and reported. In order to achieve these tasks, a container management system is being introduced and ML algorithms are utilized.
Outcomes and features
Current features Main outcome is to identify points of failure within the overall work process. Automation is the number one priority of the V&V method. Significant reduction in time is also aimed.
Planned features for the future This method is currently under development with other partners who take part in the implementation of the use case scenarios. At a later stage, a graphical user interface which will make

it easier to define desired inputs and at the end offer rich visualizations for the results is being planned.
Application domains and use cases
Domains Industrial robotics domain within VALU3S project.
VALU3S Use cases This approach being under development at the writing of this deliverable, is tested for the virtual environment of the use case #11 named "Automated robot inspection cell for quality control of automotive body-in-white", provided by OTOKAR.
Tool Chain
List of tools Robot Operating System (ROS) • Gazebo • MoveIt Motion Planning Framework • Docker • Scikit-Learn
Tool interfaces • Python API scripts • YAML files • Log files (CSV, TXT) • Dockerfiles
Simulator coupling, and co-simulation capabilities
Simulator coupling and co-simulation capabilities are under investigation at this stage
Models, notations, and formats
List of model types and modelling notations Gazebo world model files, STL files, Config files, Image files (BMP, PNG)
Support for heterogeneous models and prototypes N/A
V&V Process
Description of the V&V process In the first step, Dockerfiles are created automatically based upon the input parameters. Multiple worlds are executed within separate containers. Observation outputs are logged continuously. The results are analysed at the end of the process and a final report is created.
Input information and pre-conditions Gazebo models, ROS launch files and parameters to be observed are inputs to the toolchain. Parameters are located in config files (YAML).
Construction approach for models and prototypes Simulated worlds must be at hand before execution.
Output information Fault type, timestamp, localization data, parameter outputs are logged.
Requirements traceability Requirement descriptions are formalized by the corresponding module. The observations are tracked by the logging module and investigated by the analysis engine.
Quality properties
Safety-related properties The approach provides a simulated check on fault tolerance and robustness of an industrial robot in varied environments by incorporating different fault injection strategies.
Cybersecurity-related properties N/A
Privacy-related properties N/A

Further quality properties N/A
References
Links to VALU3S deliverables • D3.1 [22] (Section 3.2.1) • D3.3 [23] (Section 3.2.1) • D3.4 [21] (Section 2.26)
Links to V&V workflows Part of the workflows of UC11.
Links to resources • http://gazebosim.org/
Links to tools • RoboSimIT: https://github.com/techy-digital/robosimit • Docker: https://www.docker.com/ • MoveIt Motion Planning Framework: https://moveit.ros.org/ • Scikit-Learn: https://scikit-learn.org/
References N/A

3.6 Simulation-Based Robot Verification [OTOKAR]

Name of the virtual validation approach: Simulation-Based Robot Verification
Introduction Creating a safety trajectory with Otokar's C# based simulation tool so that the robots do not experience any collisions in the system, and a 2D-2D comparison task is carried out in order to monitor the missing or incorrectly welded part of any part. In this way, the error and time factors are reduced.
Outcomes and features
Current features Tool receives the CAD data as an input and creates and verifies the safe robot trajectory as an output.
Planned features for the future To show which part is missing or incorrectly welded in the form of a report.
Application domains and use cases
Domains Industrial Robotics and Automation Domain
VALU3S Use cases Use Case 11 – Automated Robot Inspection Cell for Quality Control of Automotive Body-in White
Tool Chain
List of tools GAZEBO, ROS, Otokar Simulation Tool
Tool interfaces C#, Python, FreeCAD
Simulator coupling, and co-simulation capabilities
As the tool is developed by OTOKAR it is open to interlinking with other simulations. It can be integrated into other simulations with the code to be written.
Models, notations, and formats
List of model types and modelling notations Step format of the vehicle's CAD data for Otokar Simulation Tool.
Support for heterogeneous models and prototypes

No.
V&V Process
Description of the V&V process First of all, CAD data is getting to the system which vehicle is tested. Cartesian robots in the system take pictures and then the pictures are transformed from 3D to 2D. Finally, the system does a comparison 2D-2D.
Input information and pre-conditions There should be nothing in the working area of the robots to encounter any collision with the robot, human or anything else. Otherwise, the robots will not work for system security. Some CAD files of the vehicles can be updated to be developed by the relevant team members of the company, if a new part is added to the vehicle and we do not know this information, we may encounter a collision with the cameras against the vehicle body. Thus, the latest updated version of the CAD file should be integrated into the system PC.
Construction approach for models and prototypes It relies on transferring real-world and vehicle's CAD data which is in virtual-world models in a close to one-to-one manner and reducing time and error.
Output information After simulation's report we will see which part is missing or incorrect welded.
Requirements traceability After the test, we will evaluate which requirements have been done with the simulation report.
Quality properties
Safety-related properties Robotic System Fault Tolerance. The robustness of the system will be tested with different faults to be injected into the system.
Cybersecurity-related properties N/A
Privacy-related properties N/A
Further quality properties N/A
References
Links to VALU3S deliverables • D3.1 [22] (Section 3.2.1)
Links to V&V workflows Part of the workflows of UC11.
Links to resources There are no publicly available resources connected to this tool.
Links to tools There are no publicly available resources connected to this tool.
References N/A

3.7 Combined Heterogeneous Validation for Discrete Event and Multi-physics Simulations [SIEMENS]

Name of the virtual validation approach: Combined Heterogeneous Validation for Discrete Event and Multi-physics Simulations
Introduction

Nowadays, cyber-physical systems are built by the combination of models and simulation engines of all different kinds coming in a lot of different flavours. Some parts of a modelled system may require a very small abstraction level for electrical circuits together with fixed-step ODE (Ordinary Differential Equation) solvers for model equations, while others need discrete-event simulators for simulating hardware on the register-transfer level (RTL). Some modelling environments are specialized in physical world modelling – they often intend to be close to continuous simulation (e.g., AMESim). On the other side of the simulation and modelling tool domain, in the digital domain, there is also a vast number of different tools and methods which have their focus on modelling discrete systems (e.g., SystemC). For combined simulation, simulation engines must be interfaced and synchronized to generate meaningful results that fulfil the required level of simulation accuracy. By combining simulation and modelling tools, all specialized in their domain, a holistic view of the CPS can be achieved.
This virtual validation approach has its focus on combining and concurrently running the simulators SystemC, QEMU (Quick EMUlator) and AMESim (Advanced Modelling Environment Simulator). Additionally, MoMuT is utilized to model the system and create test cases for the digital twin in advance. RTAMT, a tool for signal monitoring, evaluates simulation signals that must stay within the specification.

Goals
The main goals of Combined Simulation are to have the ability to:

- reduce the risk of errors and enable optimization by full-system simulation (modelling sub-systems on their own doesn't reveal the interplay with other sub-systems)
    - Model and simulate (sub-)systems with an arbitrary level of detail
    - Interface (sub-)models and simulators to simulate the whole system at once
    - Co-simulation of hardware and software
- validate the system in the context of a full virtual system setup during runtime

General set-up
Specialized simulation engines run concurrently and execute models in their simulation domain. Simulations with data dependencies are interfaced for exchanging data, simulation time information and time synchronization. Due to the different abstraction levels, time resolutions of simulations and performance of simulations, synchronization time intervals must be chosen carefully – intervals too large will cause a loss of important simulation data.
The general setup of this approach has the following components, which are interfaced to each other by various bridges:

- SystemC (discrete event simulation) – Models are hardware peripherals
- QEMU (Quick EMUlator, a virtual machine monitor) – Models are hardware peripherals for running software
- AMESim (multi-physics simulator) – Models are based on the multi-physics simulator AMESim.
- MoMuT(::UML) (Family of model-based test case generation tools) – System Models are used to generate test cases
- RTAMT (signal monitoring tool) – Checks if simulated signals conform with their specification

Outcomes and features

Current features
The orchestration of various tools and combined simulation enables runtime verification and validation of the overall system (hardware and software). Functional verification for basic system functionalities as well as the verification and validation of non-functional properties such as safety, security, privacy, and performance can be performed. For example, the basic operation of a system can be validated at runtime, or fault-injection on critical system components tests the behaviour of

safety functions, or manipulation of an encryption key enables tests of security features. There are many thinkable scenarios for verification and validation that can be carried out easier on a virtual prototype (digital twin) instead of a real-world system.
The strength of this approach lies in the ability to simulate software (including operating systems) utilizing hardware machine emulation including processors (QEMU) in combination with SystemC for individual hardware modelling (easy re-use of (legacy) IP) in the discrete domain. Furthermore, a multi-physics simulation tool such as AMESim for modelling and simulating physical objects (e.g., an electrical motor) is interfaced to the simulation cluster. This setup enables (runtime) validation of a full system by simulation and emulation: the effects between simulated physical objects, system hardware and software can be observed, and appropriate measures (debugging, optimization) taken. A further benefit of this approach is the ability to refine the digital twin by timing back annotation taken from RTL simulations.
Heterogeneous simulation and interfacing simulators are the most important features of this approach, which are addressed by AMESim (multi-physics simulation) by providing standard interfaces for connecting simulators (shared memory, TCP, FMU, etc.).
Improvements for validation results with this approach depend on the advancement of:

- Evaluating (and validating) the delta between real-world and digital twin
- Simulation accuracy versus simulation time
- Efficient heterogeneous simulation (multi-domain simulation) with synchronization/interfacing
- Efficient combined simulation of simulation engines with different abstraction levels
- Dependability analysis for safety/security/performance through the whole PLC
- Runtime-configurable dynamic fault-injection for functional and safety validation

Planned features for the future

Plans for the enhancement of the current setup include the insertion of fault-injection hooks that enable dynamic manipulation of data inside the virtual prototype. This enhancement shall enable the integration of Test Oracle Observation at Runtime, Model-Based Mutation Testing, Test-Driven Model Review and Model-Based Robustness Testing in the verification flow. The expected benefit is that the digital twin will be verified and validated with additional test cases.
Following tool/method extensions are planned:

- **MoMuT::UML extensions**
  - Behaviour-Driven Model Development and Test-Driven Model Review
    Adding support for activity diagrams, integration with Enterprise Architect and model de-factoring
  - Model-Based Testing
    Improved handling of time triggers
- **MoMuT**
  - Model-Base Mutation Testing
    Additional mutation operators
- **RTAMT extensions**
  - Test Oracle Observation at Runtime
    Adding fault explanation if the property is violated. The fault explanation is a small segment of the trace that is the core explanation of the violation.
- **Heterogeneous modelling tool for SystemC/AMS development**
  - System-level design and modelling of embedded Analogue/Mixed-Signal (AMS) systems.
- **C++ framework for the definition of fault-injection hooks interfacing fault-injection directives**
  - Digital Twin (Virtual Prototype) enhancement for interfacing Model-Based Testing and fault-injection techniques

Application domains and use cases

Domains

This approach is of generic nature and its application would fit to many domains where heterogeneous simulation is required. The methodology is planned to be applied for the Industrial Robotics/Automation domain.

VALU3S Use cases

UC13 – Motion Control for Industrial Drives

Tool Chain

List of tools

- SystemC [CHV-1]
  - Simulation kernel
  - Implementation of hardware peripherals
  - Implementation of test cases (such as fault-injection)
- QEMU [CHV-2]
  - Emulation machine
  - Configuration/Implementation of system (machine) peripherals
  - Processor emulation
  - Host for real-time operating system (FreeRTOS)
- AMESim [CHV-3]
  - Data recording for validation and verification
  - Display of data relevant for verification and validation
- FreeRTOS [CHV-4]
  - Runs software (e.g., motor control software)
- QuestaSim [CHV-5]
  - Timing measurements and back annotation for digital twin modelling
- MoMuT::UML / MoMuT [CHV-6]
  - Behaviour-Driven Model Development and Test-Driven Model Review
  - Model-Based Mutation Testing
  - Model-Based Robustness Testing
  - Model-Based Testing
- Enterprise Architect
  - Behaviour-Driven Model Development and Test-Driven Model Review
- RTAMT [CHV-7]
  - Test Oracle Observation at Runtime (signal monitoring for physics models)
- Heterogeneous modelling tool SystemC/AMS[CHV-11][CHV-12]
  - Design and implementation of Analog/Mixed-Signal components

Tool interfaces

- SystemC Models
  - Interface to AMESim with proprietary SystemC-AMESim bridge
  - Interface to QEMU with libsystemctlm-soc [CHV-8]
  - Interface between SystemC components with TLM interfaces
  - Interface to MoMuT with structured text configuration files
- Models in QEMU
  - Interface to SystemC with libsystemctlm-soc
  - Interface to FreeRTOS with low-level drivers/libraries/HW-abstraction layers
- AMESim
  - Interface to SystemC with proprietary SystemC-AMESim bridge
  - Interface to RTAMT with structured text files containing signal traces
- FreeRTOS
  - Interface to QEMU with low-level drives/libraries/HW-abstraction layers
  - Interface to MoMuT with structured text configuration files
- QuestaSim

- Timing annotations from measurements in structured text files
- MoMuT
  - Interface to FreeRTOS with structured text configuration files
  - Interface to SystemC models with structured text files
- RTAMT
  - Interface to AMESim with structured text files

Simulator coupling, and co-simulation capabilities

In a full system simulation three simulation instances (SystemC (including multi-physics simulation and hardware peripherals), QEMU-1 (hardware peripherals and software with RTOS), QEMU-2 (hardware peripherals and server software based on Linux)) are linked together on the host operating system level (e.g., Ubuntu Linux). Synchronization is realized utilizing the SystemC (Transaction-Level Modelling "TLM-2.0") simulation kernel [CHV-1]. The simulation time is synchronized for all simulation instances. Synchronization points define a simulation time boundary ("Quantum") - the amount of simulation time that instances must run freely (de-coupled) before they are synchronized. Increasing the time interval of synchronization increases overall simulation speed (less context switches). However, synchronization intervals must be chosen carefully, because with each increase a potential loss in accuracy occurs (more events are missed, loss of chronological data order within the interval). The QEMU simulation instances utilize libsystemctlm [CHV-8] as an interface to the SystemC (TLM) models.

Models, notations, and formats

List of model types and modelling notations

- Multi-physics models (AMESim)
  On the front-end, multi-physics models (AMESim) are created using graphical notations, which are represented in programming languages (such as C/C++) on the backend.
- Signal Monitoring
  Formal notation of physical-object signals using Signal Temporal Logic [CHV-10]
- Control software running on RTOS
  These are modelled with C++/C.
- Hardware peripherals
  System-Level models are written in SystemC/C++/C

Support for heterogeneous models and prototypes

Physical object models are designed in a multi-physics simulation tool that approaches continuous behaviour of models. Hardware and software models in the discrete domain are created with SystemC and QEMU.

V&V Process

Description of the V&V process

In this V&V flow, the digital twin serves mainly as a vehicle for runtime tests with several V&V methods and tools applied.

One group of verification methods is dedicated to digital twin component development (e.g., functional tests in SystemC), while another group of verification and validation methods is applied for runtime verification of the overall system (SystemC functional tests, runtime tests implemented in the digital twin's software and hardware for safety features). Additionally, the simulation of components is used to retrieve timing information for the implementation and refinement of the digital twin (e.g., RTL module verification for timing measurement and timing back annotation in the digital twin components implemented with SystemC).

Further methods shall enhance the capability of the digital twin for generating test cases, fault-injection, and signal monitoring.

Model-Based testing techniques are planned for generating new system test cases, a more detailed flow is provided in *D4.4 - Initial detailed description of improved process workflows*.

Input information and pre-conditions

As a pre-condition, requirements (textual) and test plans (textual) for model-based testing (fault-injection) and signal monitoring (signal temporal language (STL)) must be available.
Additional input needed for running evaluation scenarios is limited, since the digital twin models are all in place and just need structured text-based configuration files according to the evaluation scenario to be run.
The evaluation scenarios tests executed by the digital twin require adaption of structured text configuration files for running the digital twin and for fault-injection activities. Fault-injection hooks in the digital twin code will be dynamically activated by structured text files coming from Test-Based V&V methods.

Construction approach for models and prototypes

Designed models and their related simulation engines must be transformed either into binaries that are executable by a common underlying operating system (e.g., a Linux distribution), or libraries that can be included in other executables (e.g., shared object or dynamically linked libraries). The concurrent execution of binaries is synchronized internally (e.g., SystemC/TLM quantum keeper).
A brief overview of the construction steps for generating the models for prototyping and simulation:

- Multi-physics object models are authored in AMESim (graphically and textually). After compilation, a shared library (.so file) containing the model behaviour, model interfaces and the simulation engine is generated.
- Simulated hardware peripherals are coded with SystemC (TLM), C and C++. SystemC is the core simulator interfacing other QEMU instances and AMESim.
- Software running in FreeRTOS is coded in C/C++ and configured for QEMU and eventually compiled into an executable binary (containing software, real-time operating system, and QEMU).
- Emulated hardware machine models are configured with QEMU (e.g., processor emulation). A binary is created for all necessary machines and on-top running software (e.g., one executable binary for QEMU, FreeRTOS and software, and another for running QEMU, a Linux kernel and software).
- Testing techniques (Behaviour-Driven Model Development and Test-Driven Model Review, Model-Based Mutation Testing, Model-Based Robustness Testing, and Model-Based Testing) are supported by embedding fault-injection hooks at relevant code areas (e.g., software running in FreeRTOS, SystemC hardware peripheral models, etc.).
- The system is modelled based on UML with MoMuT. As a result of system analysis with MoMuT, test cases (fault-injection) are generated with above mentioned Model-Based approaches and stored in a structured text document. Affected models can access, read and interpret such fault-injection configurations and change the modelled behaviour accordingly.
- A formal specification (Signal Temporal Logic (STL)) is the language in which signals of the multi-physics models are specified. Based on the specification, checkers (signal monitors) are implemented (e.g., in Python) that evaluate signals (simulation outputs in structured text format) for their conformity to the specification.

Output information

Outputs produced by this whole setup are mostly structured text files that indicate test results. Ad-hoc graphical representation is produced by the multi-physics tool for visual inspection. For example, the multi-physics simulator produces outputs of physical object properties that are saved textually. These outputs are imported and analyse by the signal monitor, which in turn generates text files containing V&V results.

Requirements traceability

An industry-grade solution for requirements traceability is the tool Polarion [CHV-9].

Quality properties

Safety-related properties

The prototype addresses following safety-related properties:

- Electrical motor speed

- Safe-Stop behaviour
- Heart-Beat for control and monitor signals

Cybersecurity-related properties

Following cybersecurity-related properties are addressed:

- Authentication for motor sensor values and motor set values
- Encryption for motor sensor values and motor set values

Privacy-related properties

N/A

Further quality properties

An important addressed additional quality attribute is electrical motor control accuracy.

References

Links to VALU3S deliverables

- D3.1 [22] (Sections 3.3.1, 3.3.4, 3.3.5, 3.3.6)
- D3.4 [21] (Section 2.17)
- D3.5 [20] (Section 2.20)
- D4.1 [24] (Section 4.3)

Links to V&V workflows

- **WF6.** Process workflow starts with a requirements specification in Signal Temporal Logic (STL) and a model construction of the system. From the STL specification a runtime monitor in C++ is generated. The model is used to generate test cases that test the input/output behaviour of the system. Finally, the monitor is attached to the system to perform runtime verification. The tests are used to drive the system and verify its outputs are correct while the runtime monitor is also able to monitor the internal state of the system.
- **WF23.** The process workflow starts with construction of two models of the system: A functional model and an architecture/data-flow model. The functional model is used to generate test cases that test the input/output behaviour of the system. The architecture/data-flow model is used to identify weaknesses in the architecture that allow an attacker to penetrate the system. The weaknesses found in the architecture guide the fault injection and the tests are used to run the system and verify the injected tests are triggered.

Links to resources

N/A

Links to tools

- SystemC, https://systemc.org/
- QEMU, https://www.qemu.org/
- FreeRTOS, https://www.freertos.org/
- AMESim, https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-amesim.html
- QuestaSim, https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator/
- MoMuT, www.momut.org
- RTAMT, https://github.com/Digital-Safety-and-Security/rtamt-cpp
- libsystemctlm-soc, https://github.com/Xilinx/libsystemctlm-soc
- SystemC AMS, https://ieeexplore.ieee.org/document/7448795
- COSIDE, https://www.coseda-tech.com/coside-overview

References

- [CHV-1] SystemC, https://systemc.org/
- [CHV-2] QEMU, https://www.qemu.org/
- [CHV-3] AMESim, https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-amesim.html
- [CHV-4] FreeRTOS, https://www.freertos.org/

- [CHV-5] QuestaSim, https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator/
- [CHV-6] MoMuT, www.momut.org
- [CHV-7] RTAMT, https://github.com/Digital-Safety-and-Security/rtamt-cpp
- [CHV-8] libsystemctlm-soc, https://github.com/Xilinx/libsystemctlm-soc
- [CHV-9] Polarion, https://www.plm.automation.siemens.com/global/en/products/polarion/requirements.html
- [CHV-10] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FORMATS/FTRTFT), 2004, pp. 152–166.
- [CHV-11] SystemC Analog/Mixed-Signal, https://ieeexplore.ieee.org/document/7448795
- [CHV-12] COSIDE, https://www.coseda-tech.com/coside-overview

3.8 Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance [UCLM]

Name of the virtual validation approach: Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance

Introduction

This approach aims to involve users at an early stage of the development process of an industrial robot and measure its acceptance. Robot operators will be able to interact with the robot before it is even built, either in a full Virtual Reality (VR) simulation or using Augmented Reality (AR) in the real-world facilities in which the robot will be deployed. This could be used to assess the acceptation of the robot and to test different approaches to the behaviour of the robot, its appearance, the reaction of the robot operators under certain situations and failures, etc. In this context, technology acceptance refers to how users accept and use a technology that, normally, is new to them. It is typically a subjective view, which is perceived by the user and gathered using questionnaires.

Goals

The main goal is to obtain information about the experience of the end-user while using the simulator. Another important outcome of this approach is to observe the reaction of the user when faults are injected. This is important to know how they could react under specific and controlled situations.

General set-up

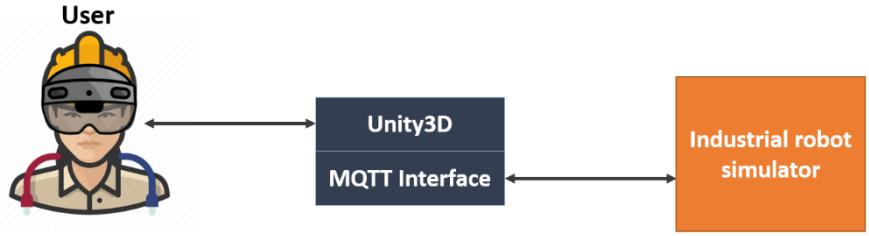


Figure 3.4 Tool interfaces of Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance

In the general set-up (see Figure 3.4), an end-user (operator) uses VR/AR devices to connect to virtual/augmented representation of the workplace. In this simulation, they would be able to interact in a similar way as they would do in the real world, and this may vary depending on the particular needs of the application environment (picking and dropping objects, assembling/disassembling, etc.). Then, the simulation would be connected to an industrial robot simulator controlling the task to be performed by the operator, including fault injection if needed. Information about the user interaction

and reactions can be stored for further analysis, as well as a customised acceptance questionnaire could be used when the task is finished.

Outcomes and features

Current features

- This approach can evaluate the response of the human operator under different and controlled situations.
- The aim is also to evaluate the acceptance of the industrial robot before it is built, allowing the evaluation of different approaches for behaviour, appearance, etc.
- Making the scene as realistic as possible is a challenge as well as the user interaction for assembly/disassembly tasks.

Planned features for the future

- We aim to connect to an industrial robot simulator in order to be able to validate it.

Application domains and use cases

Domains

This approach will be used in the industrial domain.

VALU3S Use cases

The Use cases in which this virtual validation approach is planned to be applied are UC4 and UC7.

Tool Chain

List of tools

For this approach, an inhouse development based on Unity3D is foreseen. This will make use of the MQTT library to connect to an external robot simulator.

Tool interfaces

MQTT is planned to be used to provide interconnection with other tools. The protocol/messages to be exchanged have not been defined yet. However, it is planned that user position updates together with user interactions will be sent to the robot simulator and the robot and other objects position will be received from the robot simulator.

Simulator coupling, and co-simulation capabilities

This approach deals with the user interaction and acceptance evaluation, and therefore requires the connection to an external robot simulator. As mentioned in the previous points, this connection will be implemented using MQTT and a set of messages/protocol defined beforehand. This connection has not been defined yet, but will include the exchange of positional data of the user and the robot as well as any other interactable object (e.g., the objects to assemble/disassemble), and the interactions performed by the user.

Models, notations, and formats

List of model types and modelling notations

3D models and information about the assets (position, orientation and movement).

Support for heterogeneous models and prototypes

It is planned to be combined to other simulators (robot simulator).

V&V Process

Description of the V&V process

The process involves:

- Adding the 3D model of the industrial robot and the objects to be manipulated to the VR/AR simulator.
- Configuring the scene. For VR, include the 3D model of the facilities. For AR, attach the virtual objects to the corresponding place inside the real environment.
- Connecting to the robot simulator and send the user positional information and interactions, and update the 3D representation of the robot with the information received from the simulator.
- Allowing the user to interact with the robot. At the end, obtain the result of an acceptance questionnaire as an output of the test.

Input information and pre-conditions

Configuration of the scene (VR or AR) and description of the test. 3D models of the robot and environment, streamed data about robot pose. Similar for other objects that can be manipulated by the robot and the user.
Construction approach for models and prototypes The virtual 3D environment in which the user is immersed is created beforehand and set up. If VR is to be used, then the industrial facilities in which the robot is located should be modelled (or at least approximated) in order to increase realism. If AR is to be used, the position of the virtual robot needs to be mapped into the real-world working environment. 3D models of the robot and the objects to be manipulated are needed in both cases (AR and VR). User interaction with the environment needs to be modelled beforehand as well. In assembly/disassembly tasks, different levels of realism can be achieved and needs to be stablished beforehand. Moreover, communication with the robot simulator needs definition as well. It comprises the definition of an interface and the messages that can be exchanged (protocol).
Output information Streamed data about human and their interactions with objects (which can also be stored for further analysis). The final output would be a questionnaire of technology acceptance filled in by the user.
Requirements traceability Not addressed.
Quality properties
Safety-related properties Not addressed directly by this approach but can be addressed in combination with other methods. For example, functional safety would require the combination with a simulator that can reproduce scenarios in which the user has to interact with the robot. Moreover, the robustness of the human-robot interaction could also be evaluated when combined with a method that include fault injections.
Cybersecurity-related properties None.
Privacy-related properties None.
Further quality properties User acceptance of the technology.
References
Links to VALU3S deliverables • D3.1 [22] (Section 3.2.5). • D3.3 [23] (Section 3.2.5). • D3.5 [20] (Section 2.37).
Links to V&V workflows Not included in a workflow yet.
Links to resources N/A.
Links to tools • https://unity.com/ • https://mqtt.org/
References • Belmonte, L.; Garcia, A.S.; Segura, E.; Novais, P.J.; Morales, R.; Fernandez-Caballero, A. Virtual Reality Simulation of a Quadrotor to Monitor Dependent People at Home. IEEE Transactions on Emerging Topics in Computing, 2020. doi:10.1109/TETC.2020.30003. • Belmonte, L.M.; García, A.S.; Morales, R.; de la Vara, J.L.; López de la Rosa, F.; Fernández-Caballero, A. Feeling of Safety and Comfort towards a Socially Assistive Unmanned Aerial Vehicle That Monitors People in a Virtual Home. Sensors 2021, 21, 908. doi: 10.3390/s21030908.

3.9 Static and Runtime Verification of Real-Time Systems [ISEP]

Name of the virtual validation approach: Static and Runtime Verification of Real-Time Systems
Introduction This approach uses a specification of a concurrent system with temporal requirements to (1) facilitate the verification of functional and temporal properties of systems under development, and (2) facilitate the testing of the coverage of generated monitors. The system and requirements are described in a dedicated DSL (Domain Specific Language), in Uppaal, and in a set of specially formatted spreadsheets. These are used to generate timed traces of the system, which in turn are used to validate the monitors after injecting faults.
Outcomes and features
Current features This approach can verify if, for a given scenario, a set of desirable states can be reached, and a set of undesirable states is never reached. It further allows **developers not familiarised with formal methods** to customise and **experiment with different configurations** of the formal models, providing quicker insights over the tools being developed. The core addressed challenges for virtual prototyping are (1) how to guarantee that the system specifications and the requirements are correct-by-constructions, and (2) how to facilitate the usage of the formal machinery by system developers.
Planned features for the future The tool is expected to support variability, i.e., the possibility of generating variations of the real-time specifications. This will allow to statically verify stronger requirements over simplified versions of the specifications, and to generate traces of faulty variants of the system, to be used to validate the generated monitors. After the project we plan to further tighten the gap between system developers and the formal specification process by introducing new dedicated domain specific languages, to be used in the generation of models for model checkers or artifacts used in the system implementation.
Application domains and use cases
Domains This approach is being applied to the Railway domain and is planned to be applied to the Healthcare domain in the context of the upcoming use-case of a new VALU3S' partner. Many other domains could be applicable as well, involving the development of real-time critical systems.
VALU3S Use cases This approach is being applied to the UC10 and planned to be applied to the new UC of the upcoming VALU3S partner.
Tool Chain
List of tools Uppaal will be used to describe the real-time behaviour of the system under evaluation. Uppx will facilitate the parameterisation and creation of multiple instances of Uppaal specifications, including instances describing faulty scenarios or implementations. MARS will support a dedicated DSL and will be used to generate monitors. Safety requirements will be specified both in Uppaal and in MARS' DSL and will be analysed both statically by Uppaal and dynamically by providing traces produced by Uppaal to the generated monitors. Traces with faults could be also generated by tools developed by other partners involved in UC10, which is still under investigation.
Tool interfaces The behavioural model provided to Uppaal includes special annotations and uses the following 3 artifacts: (1) the behaviour with real-time aspects, (2) the system architecture, and (3) a set of functional and temporal requirements. The MARS tool will use the same system architecture, as well as source code from the system and extra requirements to be verified at runtime, described in an internal DSL. MARS will use these artifacts to produce runtime monitors.

Uppaal will then generate traces of expected and faulty behaviour, which will be used to simulate the environment of the runtime monitors in a co-simulation environment, to assess the quality of the monitors. The specific platform where the monitors will run is still not yet fixed.
Simulator coupling, and co-simulation capabilities
The runtime monitors of MARS will be able to execute on a co-simulated scenario, using a provided set of traces instead of using the system under test. The monitors will interact with a message broker that can simulate an environment (as described in the combined method RVF-FIN). The tool RMTLD3Synth, also used by method RVF, will be used, and further improved, to deploy and run the generated monitors.
Models, notations, and formats
List of model types and modelling notations Real-time specifications will be modelled in UPPAAL together with a configuration table in Excel to guide the generation of system variations. The requirements will be specified both in UPPAAL (within the Excel tables or in a dedicated DSL), and in MARS' DSL, depending on whether they target requirements to be verified statically or at runtime. The traces will be provided in a structed textual format such as XML or JSON.
Support for heterogeneous models and prototypes No support.
V&V Process
Description of the V&V process The V&V process starts by compiling the target safety requirements, formalising these in temporal logics for a model checker such as UPPAAL, and as a logic for runtime verification used by the RMTLD3Synth tool. These are included in the configuration table (method MCF) and in the MARS specification file (method RVF), respectively, used during model checking and monitor generation. The UPPAAL model is further used to produce timed traces that are passed to the generated monitors for testing purposes.
Input information and pre-conditions MARS specification file (still under development), Excel table (to configure real-time specifications), and annotated Timed Automata, e.g., for UPPAAL model checker.
Construction approach for models and prototypes Two approaches will be taken. On one hand we will parameterise the formal specifications and describe the configurations of these parameters in dedicated configuration tables in Microsoft Excel. On the other hand, we will generate traces from manually defined formal specification, mutate these, and use mutations of these traces with faults to test the coverage of the runtime monitors.
Output information Reports of which properties hold, and which are violated, and if applicable, the context of when they are violated. The specific format is not yet decided.
Requirements' traceability When Uppaal concludes that a safety property does not hold, it sometimes gives evidence of why that is the case, such as a trace to an infinite loop, whenever possible. Safety properties that fail by the runtime monitors have an associated trace of actions and a state that can be used to detect the source of the problem.
Quality properties
Safety-related properties This approach analyses reachability properties for static verification (i.e., that some combinations of internal states can be reached). When performing static verification, it uses simplifications of the real-time specification to be able to verify safety properties such as deadlock freedom, that error states cannot be reached in good environments, and that certain events occur within a predefined time window. When performing runtime verification, accumulative values can be used to check if certain bad states do not occur for certain time windows.

Cybersecurity-related properties N/A
Privacy-related properties N/A
Further quality properties N/A
References
Links to VALU3S deliverables • D3.1 [22] (Section 3.4.2) • D3.4 [21] (Section 2.23) • D3.5 [20] (Sections 2.16, 2.27, and 3.3)
Links to V&V workflows Each of the three methods in D3.5 contain a VVML diagram describing the workflows of our methods: • Runtime Verification based on Formal Specifications, • Model Checking Families of Real-Time Specifications, and • Combination of Runtime Verification and Fault Injection.
Links to resources N/A
Links to tools • MARS: not yet available. • RMTLD3Synth: Source code and documentation – https://github.com/anmaped/rmtld3synth • Uppx: Source code and documentation – https://github.com/cister-labs/uppx • Uppaal: Download and documentation – https://uppaal.org
References N/A

3.10 Virtual Validation for ML-based systems [INFOTIV+RISE+BERGE]

Name of the virtual validation approach: Virtual validation for ML-based systems
Introduction Developing a safe and functional behaviour model, that is based on simulator generated/synthesized data founded on real-world data, in a V&V pipeline of ML systems. The reality-level of real traffic and sensor responses simulated as well as the coverage of the traffic scenarios simulated will affect the uncertainty of ML-based systems regarding e.g., the correctness and coverage of the validation processes. **Goals** • Numerous different traffic scenarios can be generated within a shorter period of time to fulfil the ML requirements, compared to real-world traffic scenarios acquired. Further, numerous simulated "unusual" traffic scenarios and environmental conditions can be generated with the simulator, which will enhance the validation of the adopted ML-based systems while not causing danger in real traffic. • In order to omit the privacy issue of sensitive personal data acquired from the real-world camera sensor used within the use-case (UC1), a so-called "digital-twin" will be generated, where no personal data will be generated and acquired. • The generated simulated data will be used as verification data for the virtual validation of the ML-based system adopted. • Data arguments as well as ML verification arguments will also contribute to the virtual validation of the ML-based system adopted.

- Comprehension of how to use the simulator generated/synthesized data and real-world data in a V&V pipeline for ML systems.

General set-up

A simulated three-dimensional (3D) rendering environment with realistic, time resolved, traffic scenarios simulating the real-world perception sensor responses as inputs for ML-based object detection/tracking used within UC1, including position, orientation, field-of-view and acquiring rate of the real-world sensor, in order to achieve an as good as possible digital-twin of the generated simulated scenarios. Further, different environmental effects will also be simulated, such as e.g., wet road surface due to rain, sensor contamination due to precipitation, glare due to sunlight, etc.

Outcomes and features

Outcomes and features

- Main outcome regarding validation scenarios and validation results: Verification data, data arguments, ML verification arguments and metrics for evaluating the ML-models
- Unique selling point: How much better/worse are the adoption of the simulated scenarios and environments for the ML systems adopted compared to only using real world data. How much more (or less) cost, and time, effective is the adoption of the simulated scenarios for the ML systems adopted.
- Supporting safety assurance for ML based systems is a challenge that will be studied during the project.

Planned outcomes and features for the future

- Automated scenario generation
- GAN-based object rendering within the simulator

Application domains and use cases

Domains

Automotive domain

VALU3S Use cases

UC1

Tool Chain

List of tools

- Esmini and OpenEd are used to create and evaluate scene and scenarios.
- Unreal Engine & Nvidia Omniverse are used for 3D modelling.
- CARLA, SVL and Berge (based on unreal engine) simulators will be used for prototyping the 3D environment for validating the ML component.
- Enterprise Architect

Tool interfaces

Python API to communicate with test and scenario script
Configuration files

Simulator coupling, and co-simulation capabilities

The co-simulation using Simulink and SUMO is possible, but there is no plan for it within the scope of this project.

Models, notations, and formats

List of model types and modelling notations

- OSM (Open Street Map), OpenDrive and OpenScenario formats
- Sensor response output
- Camera outputs (image format)
- 3D models (.fbx and USD format)
- JSON meta data files

Support for heterogeneous models and prototypes

N/A

V&V Process

Description of the V&V process

In the first step, map data will be used to design a realistic model of the environment in the simulator. This is done by adding routing and scenario information using OpenDrive and OpenScenario formats and 3D geometry data and animation data, including textures and colours using FBX and/or USD format. The ML component will finally be integrated within the simulator and its performance will be evaluated using python test scripts.

Input information and pre-conditions

- Map of the area in the OSM (Open Street format)
- 3D model of the area
- Road information in OpenDrive format
- Traffic Scenario in OpenScenario format
- Textual requirements for machine learning system under test
- Test script

Construction approach for models and prototypes

1. Calibration of sensor with digital twin
2. Integration of ML model into the simulation pipeline

Output information

- Synthetic data
- Sensor outputs
- ML system output
- KPI analysis

Requirements traceability

Since the requirements will not change during the project, they can be traced manually.

Quality properties

Safety-related properties

- It reduces the number of real-word test drives for verification and validation of the system.
- It supports the safety assurance process.

Cybersecurity-related properties

N/A

Privacy-related properties

Simulator does not require personal sensitive data (no real human faces in the real location, no real license plates, etc).

Further quality properties

- Faster verification and validation process
- Flexibility in including new models and sensors and varying the environmental conditions

Flexibility in executing

- Allows multiple and periodic execution of verification and validation process

References

Links to VALU3S deliverables

- D3.1 [22] (Section 3.2.6)
- D3.4 [21] (Section 2.36)
- D3.5 [20] (Section 2.40)
- D4.1 [24] (Section 4.4)

Links to V&V workflows

Part of the workflows of UC1.

Links to resources

N/A

Links to tools

- CARLA (https://github.com/carla-simulator/carla),
- SVL simulator (https://www.svlsimulator.com/)

• Unreal Engine (https://www.unrealengine.com/) • Environment Simulator Minimalistic (esmini) (https://github.com/esmini/esmini) • OpenRoadEd (https://github.com/fhwedel-hoe/OpenRoadEd)
References N/A

3.11 Behaviour Driven Model Development and Review-based Acceptance Testing [AIT]

Name of the virtual validation approach: Behaviour Driven Model Development and Review-based Acceptance Testing
Introduction The method aims at developing UML models expressing the expected behaviour. The model is essentially a virtual prototype and can be used to simulate the behaviour of a component or sub-system if there is no implementation available yet. The approach works by taking a set of requirements in plain text. The requirements are translated into a UML model by the modeler, with tool support to ensure that given scenarios are actually working on the model. The UML model serves as an input to the tool MoMuT, which subsequently generates test cases. Those tests can be used to review the model, ensuring that the achieved behaviour is in line with the expectations of a business expert. Those and additional (e.g., robustness, performance) tests are later applied to the implementation. Further it can judge the quality of existing, manually derived test-cases.
Outcomes and features
Outcomes and features • The method is able to automatically generate test cases from a model. Models are easily produced from a set of requirements. • The test cases can be used by engineers to test the real system
Planned outcomes and features for the future • We are currently developing a de-factoring for models. De-factoring means that some actions in the model will be duplicated to mimic what a naïve programmer may do (e.g., re-use by copy & paste). That will result in more test scenarios where the implementation uses non-minimal code. The reason is that separate tests will be generated for these duplications. • We aim to marry fault injection and model-based testing. That will allow us to find optimal places for faut injection using our mutation engine for models.
Application domains and use cases
Domains The approach is applicable to any domain where a reactive system is developed that interacts with its environment.
VALU3S Use cases UC13, UC10
Tool Chain
List of tools In the use cases, we use the tool MoMuT – its integration into Enterprise Architect is currently adapted to better support the behaviour driven modelling approach.
Tool interfaces MoMuT and Enterprise Architect interact using an add-in to Enterprise Architect based on Enterprise Architect's API. The add-in uses a REST-based web-API to use the services provided by MoMuT. Access to the UML model is provided by the model access layer UniqueMint from LieberLieber.
Simulator coupling, and co-simulation capabilities

The solution does not yet use simulator coupling or co-simulation. The tool provides an event-based animation, that currently can only be accessed on the command-line.
Models, notations, and formats
List of model types and modelling notations Our input uses UML models with state machines, OCL for guards and activity diagrams to model actions. Only a subset of UML is supported. The output are tests which are expressed as simple text files containing instructions of signals sent to the model and signal received from the model or as UML sequence diagrams in Enterprise Architect.
Support for heterogeneous models and prototypes N/A
V&V Process
Description of the V&V process The process involves actually multiple V&V activities: • Running given scenarios against the newly built behaviour model verifies that the expected behaviour is in the model • During scenario generation, safety properties can be checked (partial model checking) • Analysing scenarios generated from the model (=implicit acceptance testing) validates the model against the knowledge of the domain expert. • Using the model as a stand-in for a future implementations allows various V&V steps in a virtual simulation environment • Running the given and generated scenarios against the implementation increases trust that properties of the model also hold in the implementation. • Further tests (robustness, performance) can be generated from the model and run against the implementation.
Input information and pre-conditions A set of requirements in text form.
Construction approach for models and prototypes A human produces a model from the requirements. The tool produces test-cases in the form of sequence diagrams. The test-cases represent the input/output behaviour of the system.
Output information We produce a safe and functional model, and a "complete" set of tests/scenarios (UML class diagrams, state machines, activity diagrams and sequence diagrams. Only a subset of UML is supported).
Requirements traceability Enterprise Architect provides features to relate model elements to requirements. The test case generation approach in MoMuT ensures a traceability chain from model elements via model element mutations to test cases/scenarios exposing those mutations. Thereby, traceability from requirements to test cases can be guaranteed.
Quality properties
Safety-related properties Safety properties that can be expressed with OCL in the UML model.
Cybersecurity-related properties Robustness testing out of the complete model.
Privacy-related properties N/A
Further quality properties There are options to use the model and operations data of the system to generate performance/stress tests.
References
Links to VALU3S deliverables • D3.1 [22] • D3.4 [21]

Links to V&V workflows • **WF6.** Co-usage of model-based testing and run-time verification, to provide test inputs for run-time verification experiments during development. • **WF23.** Co-usage of model-supported safety & security analysis. Model-based testing and fault injection to provide targeted, efficient test inputs for fault injection campaigns that verify fault tolerance features introduced as a result of the combined safety & security analysis.
Links to resources https://momut.org/?page_id=59
Links to tools • Documentation: (included in Download below) • Download: https://momut.org/?page_id=80 • Source code repository: closed-source
References • B. Aichernig, H. Brandl, E. Jöbstl, W. Krenn, R. Schlick, and S. Tiran, "MoMuT::UML model-based mutation testing for UML," in *Software testing, verification and validation (icst), 2015 ieee 8th international conference on*, 2015, pp. 1-8.

3.12 Correlation Between Reality and Virtual Simulation / UNREAL [BERGE]

Name of the virtual validation approach: Correlation between reality and virtual simulation / UNREAL
Introduction The approach is to take 3D data and requirements for camera (and other sensors) into a simulator and generate output for software that is specialized for traffic surveillance.
Outcomes and features
Outcomes and features • The simulator is built in 3D, both the geometry and assets generating sensor data. • The simulator is able to automatically output generated simulated sensor data based on a set of requirements. • Our simulator will save both time and money because you can test your software virtually collecting hundreds of hours' worth of data in a fraction of the time. • Making the scene as realistic as possible is a challenge and also making the output from the simulator the same format as software can read. Simulating collaboration between camera and radar.
Planned outcomes and features for the future • We are currently working on setting up the scene, generating assets and basic logic for traffic. • The next phase is to model the whole scene complete and populating it with assets such as cars, pedestrians, bicycles, birds and then adding weather, fog, UX interface for instance.
Application domains and use cases
Domains Mainly domains working with sensors: radar, lidar and cameras. In the future automotive domain could also be added since autonomous cars have a lot of sensors that need simulation capability.
VALU3S Use cases UC1
Tool Chain
List of tools • **Modelling:** Autodesk Maya, 3ds Max and Blender.

• **Texturing:** Autodesk Maya, 3ds Max and Blender, Substance painter, Quixel mixer. • **Simulation Executing:** Unreal Engine.
Tool interfaces When models are done, they are textured to get a material and a realistic look. After that they are placed in Unreal Engine scene. If needed, additional modifications are implemented on the model, for instance animation, intelligence and/or logic. The scene is loaded and the user place and angle the camera and/or radar as wanted and then start the simulation. Output is generated and saved on disk or streamed to another device.
Simulator coupling, and co-simulation capabilities
No coupling to other simulations. Interface will be visual, on the screen.
Models, notations, and formats
List of model types and modelling notations 3D models, information about the assets (position, orientation and movement) and information about the sensors (frequency, field of view, resolution and more).
Support for heterogeneous models and prototypes N/A
V&V Process
Description of the V&V process V&V of Machine Learning-Based Systems Using Simulators The process involves setting up the scene in the simulator from defined requirements, run the simulations and an output is generated in the format that was defined when the scene was set up.
Input information and pre-conditions Camera and radar spec (frequency, field of view, resolution and more), FBX- or uproject-file of Unreal Scene surroundings and interactions. USD-file if scene is made in Omniverse.
Construction approach for models and prototypes The scene is created beforehand and set up. Camera and radar, with specifications, are adjusted to wanted levels. The simulation simulates and output a result that are then input into partners software.
Output information Output will be pictures (file format not defined), offline video (file format not defined), streaming video (not defined if necessary) and radar point cloud (file format not defined).
Requirement's traceability We have full control of the scene so we know how assets move and what should happen.
Quality properties
Safety-related properties A system that has the ability to detect vehicles going in the wrong direction in traffic needs to be able to simulate and verify its functionality in a safe virtual environment.
Cybersecurity-related properties Not applicable.
Privacy-related properties Not applicable.
Further quality properties Because of the ability of simulating sensors virtually, the accuracy and quality can be tested and verified early in the development process. This will lead to a better product because of the extended testing that can be done.
References
Links to VALU3S deliverables • D1.1 [25] (Chapter 1) • D1.2 [26] (Chapter 2) • D1.3 [27] (Chapter 2) • D2.1 [28]

• D2.2 [29]
Links to V&V workflows • **WF3.** Improving the V&V process by generating realistic data for training ML-based systems within perception and cybersecurity. • **WF24.** Generating inputs for simulators based on defined potentially problematic situations that will be used for at least partly automated testing of accuracy and reliability of detectors. Testing and verifying communication between sensors - triggering camera based on the radar detection input (time criticality + detection reliability and accuracy) via e.g., static analysis + run-time monitoring or alternatively data-driven testing (aiming at potential hazardous time-critical situations).
Links to resources N/A
Links to tools • Unreal Engine, https://www.unrealengine.com/en-US/ • Omniverse, https://www.nvidia.com/en-us/omniverse/ • Autodesk Maya, https://www.autodesk.com/products/maya/overview • Blender, https://www.blender.org/ • Autodesk 3ds Max, https://www.autodesk.com/products/3ds-max/overview • Adobe Substance 3D, https://www.substance3d.com • Quixel,https://quixel.com/mixer?utm_source=google&utm_medium=cpc&utm_content=mixer%20exact%20responsive%20search&utm_campaign=search%20brand
References N/A

3.13 Simulator Coupling and Network Simulation with FERAL [FRAUNHOFER]

Name of the virtual validation approach: Simulator coupling and network simulation with FERAL
Introduction The FERAL (Focused Evaluation on Requirements and Architecture Level) simulation framework creates virtual prototypes by coupling simulation models and simulators, existing code, and virtual hardware platforms. It allows the evaluation across different abstraction levels and in an early stage of the development process. One of the core abilities of FERAL is to hierarchically couple simulators and simulation models and bridging of the different models of computation and communication (MOCC) across the different coupled simulators and simulation models. FERAL is realized as a JAVA library with components that are instantiated and connected through a simulation script, which describes the composition of the system parts and the configuration of the simulation components and connectors. This approach is comparable to SystemC, which is also realized as a library that is instantiated through a C compiler. By using a high-level programming language such as JAVA simulation scenarios can be easily set-up and conducted.
Outcomes and features
Current features Simulator coupling enables the integration of simulators, simulation models, and other implemented behaviour into one semantically integrated simulation. The joint execution of the various distributed existing simulators which are specialized on their own specific demands allows simulating complex heterogeneous ecosystems. FERAL supports due to its flexibility a variety of connection abilities like the coupling of various models of computation and communication and the coupling to well-known simulators models such like: SystemC, CANoe, Simulink, and Functional Mock-up Interface (FMI).

This flexibility results in the ability of the (re-)use of different simulators and simulation models allowing for quick virtual prototyping which saves time and costs in the product development.
Specifically, FERAL supports:

- Focused validation of architecture concepts down to actual software and virtual hardware component implementations
- Prototyping of executable models and solution artefacts
- X-in-the-Loop (XiL) testing with different levels of abstraction (high-level vs. high-fidelity)
- Co-simulation/simulator coupling of complementary tools
- Design space exploration through systematic integration of software/virtual hardware system artefact variants
- Holistic evaluation of interleaved operation of heterogeneous systems beyond pure functionality

Planned features for the future

Within the VALU3s project one of our planned outcomes is the extension of our simulator coupling library: Coupling with the CIROS Studio simulation models. This coupling is heavily under development and is based on an OPCUA connector which enables the communication between FERAL and CIROS Studio. In addition, the MQTT connector was developed, which is designed as a data flow bridge between a simulation model of a POZYX indoor real-time localization system and the CIROS studio. As a feature FERAL provides fault injection for these MQTT network messages for the MQTT quality service levels zero and one. It is planned to further extend the fault injection possibilities of FERAL in respect of the simulation model which is developed in UC4. FERAL will be able to inject failure in different levels of abstraction and shall evaluate the system behaviour of the simulation model in CIROS.

Application domains and use cases

Domains

The virtual validation approach with FERAL has been applied to systems of the automotive and industrial automation domains.
Depending on the domain, specific communication protocols and interfaces for connecting system parts are applied. Furthermore, validation projects in a concrete domain have specific modelling, simulation, and validation tools, which have to be connected with dedicated adapters.
The FERAL framework and the underlying validation approach support the extension with new communication protocols and tool adapters, and thus the applicability for new types of systems and further domains.

VALU3S Use cases

The ability to couple with various simulation tools was applied in UC4 which is about the Human-Robot-Interaction in Semi-Automatic Assembly Processes. The target system is simulated by the simulation tool CIROS Studio.

Tool Chain

List of tools

- FERAL
- SystemC
- CANoe
- Simulink

Tool interfaces

The following interfaces are available:

- Integration of complementary (co-)simulation platforms with approved models via unified API
- Support for vECUs, Simulink, SystemC, Amalthea, and FMU models via dedicated communication bridges (FCAPI)
- Rest-bus simulation of, e.g., CAN/-FD, FlexRay-, Ethernet-, and LIN-based networks
- Connectivity of distributed virtual components via UDP, TCP, or shared memory

A sample set-up of FERAL with tool coupling using different connector types is shown in Figure 3.5.

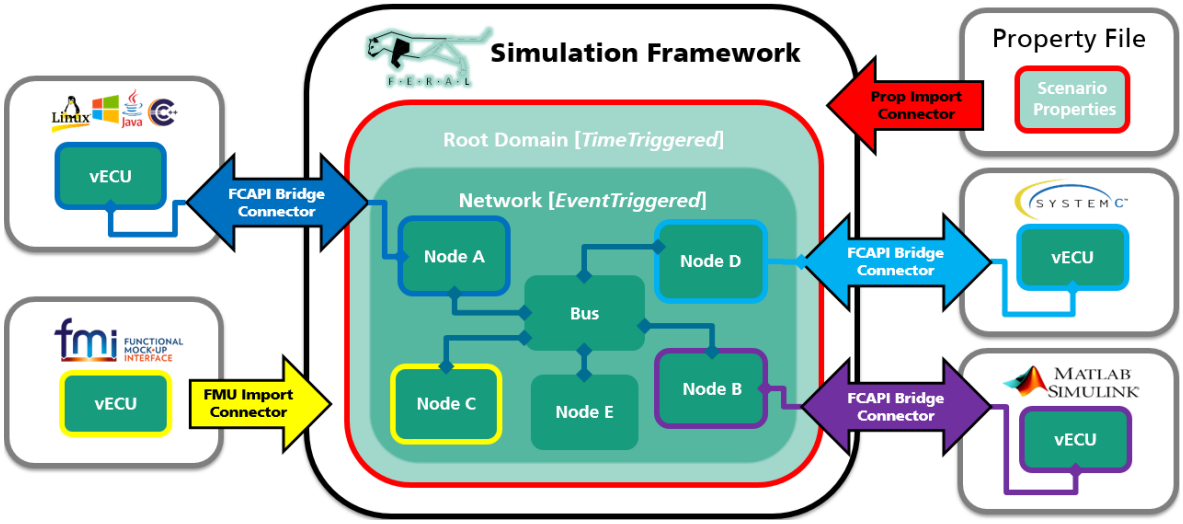


Figure 3.5 Tool interfaces of FERAL for simulator coupling and network simulation

Simulator coupling, and co-simulation capabilities

FERAL supports import/export/interchange connectors to upstream/downstream (co-simulation) tools available for some tools otherwise creatable. Additionally, co-simulation with other simulator(s) as well as pure interconnection of external tooling with FERAL as master algorithm is supported. Couplable tools must provide integration interface(s), either native API (Java/C++), via standard format (FMU), or network-based data exchange (REST API).

The following tools and simulators can already be interlinked by FERAL

- Matlab / Simulink
- SystemC
- CANoe
- gem5
- CARLA
- SCADA
- FMI (v2)
- Enterprise Architect
- R
- C
- Python

Furthermore, different models of communication and computation supported (e.g., time-triggered, real time-triggered, event-triggered). Combination of heterogenous models adhering to same or different structuring/execution semantics by means of nestable domains. Consecutive and iterative activation of domains with sequential triggering of contained models in FERAL or adjacent (co-simulation) tool during simulation execution.

Models, notations, and formats

List of model types and modelling notations

Models are similar to component diagrams in UML and Simulink diagrams. There is no native graphical front-end yet, but two options for creating models in FERAL are supported:

- behaviour models as executable Java code (either native or with support of builder-pattern)
- specification as tool-specific property files.

All modules in FERAL are implemented as JAVA projects. Each of the modules implements a certain behaviour/ logic functionality. The basic module is the FERAL core which defines the important infrastructure and API classes. One of the main simulation models are the network simulation

models. A variety of bus protocols are supported such as CAN(FD), FlexRay, Ethernet and LIN. Each network model requires its own network coder and message notation.

Support for heterogeneous models and prototypes

FERAL allows the hierarchical coupling of Models of Computation and Communication (MOCC) to enable heterogeneous execution models.

V&V Process

Description of the V&V process

Virtual validation is part of the early validation activities in system design stages to assess design decision of system and software architectures before detailed design and component implementations.

Input information and pre-conditions

The following input information is required:

Model file(s) of relevant system component(s) to be evaluated in programming code format (e.g., Java, C/C++, Python) and/or as FMU files and/or co-simulation tool-specific formats (e.g., Simulink, gem5, SystemC). Configuration settings for execution of simulation run(s) in textual key-value pair format.

Construction approach for models and prototypes

For constructing a virtual prototype, the following steps are conducted:

- Create visual models of evaluation target objects (ETOs), i.e., system component(s) to be evaluated based on available elicited requirements, in modelling tool (e.g., EA)
- Translate created visual models and/or transform available tool-specific models (the format of which is yet unsupported or if the tools are not yet interfaceable) into supported format (e.g., Java programming code). This process is partially automated for a restricted set of UML diagrams.
- Combine translated/transformed models with complementary standard FERAL components and define properties relevant for evaluation goals in terms of measurable metrics within key-value pair-based configuration files

Output information

The following output information is provided:

- General simulation run log and evaluation goal-specific result output in textual format (console output, text file)
- Co-simulation or downstream analysis tool output in correspondingly available format (e.g., R diagram)
- CSV files created by logger components in the workflow that listen on specific connections

Requirements traceability

Elicited textual/graphical requirement descriptions for ETO(s) stored in requirements engineering tool traceable via ID and hyperlink to model instantiation(s) in complementary modelling tool (e.g., EA) and/or to simulation test case(s) in code repository (e.g., GitLab)

Quality properties

Safety-related properties

The virtual validation approach with FERAL supports the validation of functional and reliability properties such as functional suitability, robustness, and fault tolerance. For checking the functional suitability of safety-critical features, an implementation of the required behaviour must be developed and deployed as simulation scenarios with defined sets of components. For the validation of robustness, specific environmental components must be created to represent different usage scenarios and profiles. The validation of fault tolerance requires the development of specific fault injection components that represent typical system-internal faults and their characteristics such as probability and frequency of occurrence and duration.

Cybersecurity-related properties

For the validation of cybersecurity-related properties dedicated environmental and attack components must be created from the use cases and system architecture.

Privacy-related properties N/A
Further quality properties The virtual validation approach with FERAL can be used for any quality property that requires the execution of validation scenarios. For example, the validation of performance efficiency needs adequate simulation models and components that consider timing aspects and resource utilization.
References
Links to VALU3S deliverables • D3.1 [22] • D3.4 [21] • D3.5 [20]
Links to V&V workflows Part of the workflows of UC4.
Links to resources https://www.iese.fraunhofer.de/en/services/virtual-engineering.html
Links to tools Documentation: https://feral.iese.de/doc/index.php/General:_About_FERAL Download: N/A Source code repository: N/A
References • T. Kuhn, T. Forster, T. Braun, R. Gotzhein: Feral - framework for simulator coupling on requirements and architecture level. ACM/IEEE MEMOCODE, pp. 11–22 (2013) • P. O. Antonino, J. Jahic, B. Kallweit, A. Morgenstern, and T. Kuhn: Bridging the Gap between Architecture Specifications and Simulation Models. IEEE International Conference on Software Architecture Companion, Seattle, WA, USA, pp. 77-80 (2018), DOI: 10.1109/ICSA-C.2018.00029. • A. Bachorek, F. Schulte-Langforth, A. Witton, T. Kuhn, P. Oliveira Antonino: Towards a Virtual Continuous Integration Platform for Advanced Driving Assistance Systems. IEEE International Conference on Software Architecture Companion, pp. 61-64 (2019), DOI: 10.1109/ICSA-C.2019.00018 • T. Kuhn, P. O. Antonino, A. Bachorek: A Simulator Coupling Architecture for the Creation of Digital Twins. IEEE International Conference on Software Architecture Companion, pp. 326-339 (2020), DOI: 10.1007/978-3-030-59155-7_25

3.14 Simulation-Based Robot Validation for Assuring Robustness and Fault Tolerance [PUMACY]

Name of the virtual validation approach: Simulation-based robot validation for assuring robustness and fault tolerance
Introduction The development of complex manufacturing plants requires systematic planning and its validation by appropriate measures and tools. CIROS Studio is a tool for 3D factory simulation from FESTO didactic. CIROS Studio enable to model layouts and processes, simulate robotic work cells and automated manufacturing plants, and visualize complex sequences. For validating the robustness and fault tolerance of architecture design of the manufacturing plant in the CIROS framework additional tools and components have been integrated. This approach extends the approach for simulator coupling and network simulation with the tool FERAL. It implements a fault injection strategy based on a fault model that have been derived from the product risk analysis.

General set-up The approach comprises a main component: the robot model which is modelled and running in CIROS, the sensor model for the position data of robots and humans, and the simulator coupling and network simulation framework FERAL, which has been extended with a fault injection component for the communication protocol.
Outcomes and features
Current features Fault injection is currently realized on the level of communication middleware to check the robustness and fault tolerance of the robot model against network and transmission faults of position data. In the current configuration, the communication protocols MQTT and OPC UA are supported.
Planned features for the future The approach is being extended to support the injection of implementation and configuration faults of system parts and to enable the impact analysis of these faults on the overall system quality.
Application domains and use cases
Domains The virtual validation approach is applied within the industrial automation domain and more precise for the assembly of transformers. Depending on the application domain, specific communication protocols and interfaces for connecting system parts must be applied. Furthermore, validation projects in a concrete domain have specific modelling, simulation, and validation tools, which must be connected with potentially customized adapters.
VALU3S Use cases The approach of coupling different simulation tools is applied in UC4 which is about a Human-Robot-Interaction (HRC) in semi-automatic assembly processes.
Tool Chain
List of tools • CIROS Studios (including RAPID and IRL programming scripts) • Codesys (Virtual PLC) • Sensor Model (POZYX indoor real-time localization system) • FERAL
Tool interfaces Connection between FERAL, sensor model, and robot model via TCP ports Python script (Creating external connection for data sending and fault injection)
Simulator coupling, and co-simulation capabilities
FERAL provides adapters to couple specific tools and simulate the behaviour of communication protocols. FERAL controls the simulation scenario by transmitting and manipulating the position and status data between the sensor model and the robot model. CIROS provides its own simulation environment of the manufacturing plant model and interfaces to the exchange of information with the other components of the simulation scenarios.
Models, notations, and formats
List of model types and modelling notations CIROS Studio has its own simulation environment with specific file format. Virtual 3D models are created inside the software. FERAL uses a specific fault injection model to enable the activation and deactivation of selected faults during the execution of the simulation scenarios.
Support for heterogeneous models and prototypes CIROS Studios is directly communicating with a development system software by Codesys which is a simulation SW-tool for creating virtual PLC controllers. The Codesys software defines virtual input and output variables to store and send/receive simulation data through OPC UA server.
V&V Process
Description of the V&V process

CIROS studio is communicating in parallel with Codesys, Python and FERAL to send/receive virtual simulation data to each tool. When the simulation is running, FERAL injects faults based on pre-defined fault injection use-cases (UC4) which is received by CIROS simulation. These faults produce an unexpected behaviour in the simulation, which is then fed back to FERAL for validation and verification.

Input information and pre-conditions

The following input information is required:

- User input to start/restart the simulation.
- Model file(s) of relevant system component(s) to be evaluated in programming code format (e.g., Java, C/C++, Python).

Construction approach for models and prototypes

Currently, the robots, sensor, and fault models are manually created.

Output information

The following output information is provided:

- General simulation run log and evaluation goal-specific result output in textual and numerical format (console output, text file)
- CSV files created by logger components in the workflow that listen on specific connections

Requirements traceability

Currently not supported, but in progress

Quality properties

Safety-related properties

The approach supports the validation of fault tolerance, robustness, and reliability of architecture design of complex technical systems.

Cybersecurity-related properties

For the validation of cybersecurity-related properties dedicated environmental and attack components must be created from the use cases and system architecture.

Privacy-related properties

N/A

Further quality properties

N/A

References

Links to VALU3S deliverables

D3.1 [22]
D3.5 [20]

Links to V&V workflows

CIROS Studio is part of the workflows of UC4.

Links to resources

- https://www.verosim-solutions.com/
- https://www.iese.fraunhofer.de/en/services/virtual-engineering.html

Links to tools

Documentation:

- https://www.festo-didactic.com/int-en/services/printed-media/manuals/ciros.htm?fbid=aW50LmVuLjU1Ny4xNy4zMi44MjguNjkwMg
- https://feral.iese.de/doc/index.php/General:_About_FERAL

Download:

- https://ip.festo-didactic.com/InfoPortal/CIROS/DE/Download.php

Source code repository:

N/A

References N/A

3.15 Simulation-Based Testing for Human-Robot Collaboration [MGEP]

Name of the virtual validation approach: Simulation-Based Testing for Human-Robot collaboration
Introduction MGEP's method's main objective is creating an automatic system to verify and validate what is happening in a simulation environment without human supervision. This system is composed of two main subsystems: the test case generation subsystem and the subsystems that compose the ULISES framework. The general system is composed of two main modules. The first module will be responsible for creating tests that are going to be simulated and selecting the most critical tests based on different objectives (e.g., time, computational cost, consumed energy) and restrictions. Once test cases have been selected, each of the tests will be executed and go through the system implemented with the ULISES framework, which will observe, interpret, and diagnose humans' and robots' activity. The result of the diagnosis will be the test validation result.
Outcomes and features
Current features • The main outcome of ULISES is the automatic test validation in both simulation and real environments. ULISES' diagnosis module generates sufficient information to validate humans' and robots' actions and to explain the root causes why a test has failed. • This framework is able to validate robots' behaviour automatically in an ill-defined domain and without human supervision. • ULISES framework has shown to be effective in giving educational feedback in VR domains such as truck simulators or when diagnosing human skills. • New diagnosing module for V&V purposes and automated test selection in simulation environments • Multi-objective optimization to select best test case scenarios
Planned features for the future • Automatic test generation
Application domains and use cases
Test case generation and optimization methods have been previously applied in drone, maritime and elevator software domains. Regarding ULISES framework, it has been applied in driving (truck and bus) simulators with the aim to assist students in their learning process. Additionally, it was also used in a system to diagnose physical skills.
VALU3S Use cases Use case 7
Tool Chain
List of tools • ROS • ROS MQTT bridge • Isaac SIM • Gazebo • ULISES framework) [SBT1]
Tool interfaces

MQTT Ros bridge (open-source software) is used to communicate Ros with the rest of the components. All the activity that is being carried out in the simulation environments is published to a MQTT queue via the MQTT Ros bridge. On the other hand, we have developed a MQTT client in the ULISES observation subsystem. This subsystem translates the streams or the information provided in the MQTT topics and translate it to observations: the base elements that the upper subsystems need to generate interpretation and diagnosis results.
Regarding the observation level, all the observations and their properties need to be defined in a XML file. The upper interpretation and diagnosis levels need their respective interpretation and task models to be defined. These two models are stored in a database.
Regarding diagnosis results, which include validation results, they are stored in a database, and it is planned to publish this information via MQTT, so it can be used by other components too.

Simulator coupling, and co-simulation capabilities

The ULISES framework is designed to be integrated with any VR based interactive system. In order to carry out the integration with a simulation, it requires to adapt the observation subsystem, so it transforms data streams to suitable information for the higher interpretation and diagnosis subsystems. Therefore, it can be integrated with several simulators at the same time.
In the current project, the communication between the simulator and the ULISES observations agents will be established via the MQTT protocol. The information that is going to be exchanged via the topics is yet to be defined. It will depend on the specific diagnosis results that will need to be generated, which depend on the specific simulation scenarios that will be tested.

Models, notations, and formats

List of model types and modelling notations

Observation model: observations and their properties are defined in an XML file

Interpretation model: steps and situations are defined via a constraint-based modelling method. This model is stored in a SQL database.

Task model: step and situations' validation are defined by a constraint-based modelling method. This model specifies how steps and situations are going to be diagnosed and generate suitable validation results. Both this model and the results generated by the diagnosis subsystem are stored in a SQL database.

Support for heterogeneous models and prototypes

As mentioned above, the ULISES framework can be integrated with any VR based interactive systems. Each integration requires a programmatic effort to modify the observation subsystem. Nevertheless, the interpretation and diagnosis subsystems are generic and don't need to be modified.

V&V Process

Description of the V&V process

The first phase of the proposed method will provide a mechanism to create random simulation scenarios based on a specification that needs to be validated. In the VALU3S project, some test cases will be provided manually, and others will be created automatically.
Once test case related simulations are defined and created, the test case selection method will prioritize previously created scenarios based on a single-objective or multi-objective-based optimization algorithm. Based on constraints imposed by the physical system and objectives determined by end users (e.g., time, CPU consumption of the simulation), the best test cases will be selected.
The ULISES framework transforms data streams generated by a Virtual Reality Interactive System into data suitable to diagnose a test case in real time. This diagnosis will generate sufficient information to validate a test case specification.

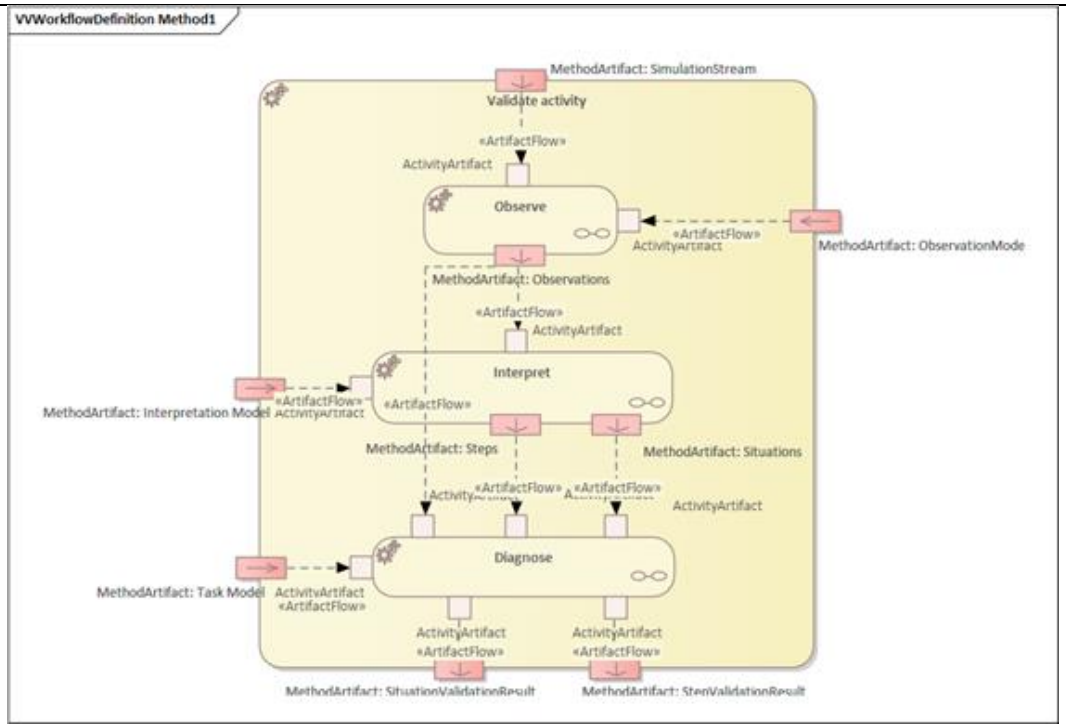


Figure 3.6 Workflow of Simulation-Based Testing for Human-Robot Collaboration

The workflow of the approach is shown in Figure 3.6. ULISES is a three-layered framework that explicitly models the unconscious process that a real human use when they supervise real activities: they first perceive the environment through their senses, then they interpret what is happening and lastly, they make a diagnosis about what happened. In order to ensure that the runtime kernel is able to observe, interpret and diagnose operators' and robots' activity, the ULISES framework defines the ULISES metamodel, which is divided into three abstraction levels and each of them generically describes a set of elements that have to be particularized into the Task, Interpretation and Observation Models. In other words, the metamodel defines elements to specify how to observe the actions that are being carried out in the Interactive System, how to interpret the steps taken by user or robots and the context in which they are taken, and how to diagnose them. Hence, each level of the metamodel represents a different phase in the task creation process.

Input information and pre-conditions

- Test case definition: textual description in txt files.
- Test execution: carried out with gtest (at this moment).
- Observation model: XML file. It defines observations and its properties. That is to say, those elements that need to be observed from the simulation environment.
- Interpretation model: stored in a SQL database. It specifies the steps and situations (context) that need to be validated.
- Task model: stored in a SQL database. It specifies the steps and situations that need to be diagnosed.

Construction approach for models and prototypes

- The VR environment(s) are created beforehand.
- Observation, interpretation and task models need to be defined beforehand too. This is the correct approach that an instructional designer (IS) (role that specifies these three models) would need to follow: the IS needs to know about the tests that are going to be validated, so he/she can define steps and situations that are going to be diagnosed. This first step will be specifying all the signals (observations) that need to be captured from the virtual environment.

- As it has been mentioned before, data streams that are coming from the simulator (in this case via MQTT) need to be transformed into observations. This requires an implementation in the observation level, which must be made just once (until the simulator and its signals are fully integrated).

Output information

Output information is generated by the diagnosis subsystem. This subsystem will generate validation results in real time, which include:

- Each step and situations' correctness: validation passed or not
- Reasons why the validation was passed / not passed: constraint violations
- Other information needed to create the final report

Output information is stored in a SQL database and has been used in other domains to create reports with the diagnosis results.

Requirements traceability

Not addressed.

Quality properties

Safety-related properties

The approach supports the validation of functional and safety properties. In particular, it will be used in the validation of the collaborative human-robot process for the disassembly of fridges. A safety requirement to be validated is that in the disassembly process if the separation distance between a hazardous part of the robot system and any operator falls below the protective separation distance, the robot system makes a speed reduction, followed by a transition to safety-rated monitored stop if the distance is below a certain threshold.

Cybersecurity-related properties

N/A

Privacy-related properties

N/A

Further quality properties

User acceptance of the technology.

References

Links to VALU3S deliverables

N/A

Links to V&V workflows

Not included in a workflow yet.

Links to resources

N/A

Links to tools

- ROS bridge: http://wiki.ros.org/mqtt_bridge
- MQTT: https://mqtt.org/
- ISAAC SIM: https://developer.nvidia.com/isaac-sim
- GAZEBO: http://gazebosim.org
- ULISES: No public access

References

- [SBT1] Aguirre, A., Lozano-Rodero, A., Matey, L. M., Villamañe, M., & Ferrero, B. (2014). A novel approach to diagnosing motor skills. IEEE Transactions on Learning Technologies, 7(4), 304-318.
- [SBT2] Takaya, K., Asai, T., Kroumov, V., & Smarandache, F. (2016, October). Simulation environment for mobile robots testing using ROS and Gazebo. In 2016 20th International Conference on System Theory, Control and Computing (ICSTCC) (pp. 96-101). IEEE.

3.16 Virtual Validation of Multi-Core FPGA Prototypes [KTH]

Name of the virtual validation approach: Virtual validation of Multi-core FPGA prototypes
Introduction When one updates an FPGA Design of reasonable size, it takes roughly between 45 minutes-60 minutes to resynthesize the HW and update the configuration file. Although programming FPGAs is considerably faster than the turnaround in ASIC-designs (hours instead of months), it still takes enough time to get a designer restless. Instead of emulating the design on the FPGA, one could build a virtual prototype, i.e., a digital twin of the system, which emulates the system in Software. This allows for software-speed compilation and quick turnaround times (seconds instead of hours), so SW designers can get quick feedback if the SW is correct or not. The downside of Virtual Prototypes is that the implementation sometimes is not a 100% match of the hardware behaviour, and that the simulation speed is much slower than if running it. **Goals** By building a Virtual prototype of the intended system, we can get early feedback by validating that the system SW is going to work with the HW on the intended FPGA platform. In addition, we also gain access to the fault-injection process of the simulation environment that enables us to inject faults in the SW part of the System. **General Setup** The setup is that the NSG system builder generates a digital twin of the System in the OVP simulator, to get early validation that the System SW running on the target system is correct.
Outcomes and features
Outcomes and features With the virtual prototype of the system, it is possible to simulate the SW binaries running on the ARM-cores and uBlazes of the FPGA system, perform fault-injection in the SW binaries and evaluate how the System reacts to these faults.
Planned outcomes and features for the future Add OVP as target technology for the NoC System Genreator (NSG) tool Including the virtual prototype feature in the next planned release of the NSG tool
Application domains and use cases
Domains Transport domain
VALU3S Use cases UC10
Tool Chain
List of tools • NSG for generating the Virtual Protype of the System • OVP for simulating the prototype
Tool interfaces The NSG generates scripts that are used to import and setup an image of the system in the target tool. It produces scripts that are native for the target technology. For the Xilinx backend, it creates. tcl-scripts, for the Intel/Altera backend it creates QSYS-scripts, and for the OVP simulator it will generate either OVP iGen commands or OVP OP API functional calls.
Simulator coupling, and co-simulation capabilities
The OVP simulator can be used for co-simulation. The research literature describes several approaches for using the OVP simulator in conjunction with other simulators, for instance QEMU, System-C, Matlab-Simulink, and Modelsim (VHDL/Verilog).
Models, notations, and formats

List of model types and modelling notations

- The system description used by the NSG tool is an XML-based that describes the HW and the Model-of-Computation used by the SW on the target processing cores of the FPGA System. The SW itself is described using C.
- The NSG has a (not yet stable) ForSyDe-SystemC import function.
- It can also import some Simulink c-models that comply to the NSG notion of Synchronous MoCs.

Support for heterogeneous models and prototypes

The NSG tool can be used to generate FPGA prototypes that contain ARM cores, uBlaze/NIOS cores (depending on target technology), and VHDL-cores. With the addition of OVPSim, it will be possible to simulate the system before implementing it on the FPGA.

V&V Process

Description of the V&V process

In the SafePower project, Imperas (the OVPSim provider) developed a library that used the OVPSim intercept technology to access memory, instructions etc and create bit and other faults in the architecture. We will use this technology to simulate bit faults in the SW binaries.
Exactly where in the V&V process this will fit in will be decided during the project.

Input information and pre-conditions

Any model that is NSG compliant in terms of Model-of-Computation and can be described in or translated to the NSG format (XML+C).

Construction approach for models and prototypes

Transformation of the NSG models into a representation that is native to the OVP tool.

Output information

Simulation results of system behaviour when faults are inserted into the SW binaries.

Requirements traceability

Injected faults in SW functions should result in deterministic behaviour of the system.

Quality properties

Safety-related properties

Bit-flips in memories and the architecture. Also, it provides redundancy by duplicating SW and run it on different cores.

Cybersecurity-related properties

Cybersecurity-related properties will not be investigated within the project.

Privacy-related properties

Cybersecurity-related properties will not be investigated within the project.

Further quality properties

User acceptance of the technology.

References

Links to VALU3S deliverables

N/A

Links to V&V workflows

We plan to include it as part of the workflows of UC10.

Links to resources

N/A

Links to tools

- OVP - https://www.ovpworld.org/
- NSG - https://noctegra.github.io/

References

A number of research articles using the OVPSim has been produced by the research community over the years. The ones co-authored by Imperas (the OVP provider) during the SafePower project are:

- Maher Fakih, et. al, "Experimental Evaluation of SAFEPOWER Architecture for Safe and Power-Efficient Mixed-Criticality Systems", Journal of Low Power Electronics and Applications, 2019.
- Sören Schreiner, et. al, "A Functional Test Framework to Observe MPSoC Power Management Techniques in Virtual Platforms", In Proc. of the Euromicro conference, 2017.
- Razi Seyyedi, et. al, "Towards virtual prototyping of synchronous real-time systems on noc-based MPSoCs", In Proc. of the 12th International Symposium on Industrial Embedded Systems

3.17 Model-Based Formal Specification and Verification for Robotic Systems [ESOGU]

Name of the virtual validation approach: Model-Based Formal Specification and Verification for Robotic Systems

Introduction

Simulators are widely used in the development and testing of robotic systems. In order to operate the test scenarios made during the development process, it is challenging to arrange the robot system in accordance with the test conditions. In addition, system errors that occur during system verification can cause serious safety problems. Therefore, simulators are the most effective tools for robotic system development and validation.

Goals

In VALU3S, a method named Model-Based Formal Specification and Verification for robotic systems (MBF) is being developed. In the method, verification is performed in two stages: model checking and runtime verification. During the runtime verification phase, system software needs to run and send its states. Furthermore, in order not to encounter safety problems, the system software will run in a simulated environment.

Also, the runtime verification stage needs additional information, such as the robot's closest distance to obstacles in the environment, in addition to the states of system software. This information will be provided from the simulated environment of the system. The virtual validation scheme is shown in Figure 3.7.

General set-up

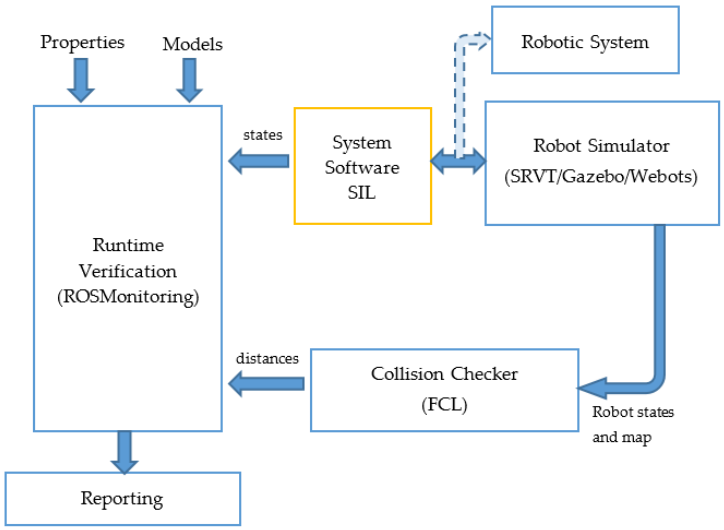


Figure 3.7 Tool interfaces of Virtual Validation of Robotic Systems with Model-Based Formal Verification

Outcomes and features
Current features The main outcome is the automation of virtual validation of the software developed for robotic systems. In this approach, the robot software runs in a simulation environment. The states of the software during execution are monitored online, and the fulfilment of the given requirements is checked. It may be possible to reduce the human workload. It is safer than using the real system. Moreover, in case of unexpected errors, the robot does not harm itself, its environment, or humans while applying the tests for validation. The main challenge is to be able to represent a real system in a simulation environment.
Planned features for the future The runtime verification (ROSMonitoring) will be integrated with the robot simulator (SRVT/Gazebo) and collision checker. The instrumentation of the software to be verified and the formal specification of properties will be implemented automatically by using the verified system model and the properties queried during model-checking.
Application domains and use cases
Domains This approach is intended to be used in the industrial domain, especially for robotic systems.
VALU3S Use cases This approach is planned to be applied primarily to UC11.
Tool Chain
List of tools The models of the system are verified by the model-checking tool, which is UPPAAL. After the queries of given properties verify the model, the model and queries are fed to a runtime verification tool which is ROSMonitoring. The simulated environment is constructed in simulators (SRVT, Gazebo, Webots). The collision checker (FCL) provides the minimum distances between the robot and its environment. The minimum distances with the states of the system software are monitored and decide whether the safety requirements of the system software are confirmed.
Tool interfaces All software tools run on ROS (Robot Operating System) framework. It provides a facto-standard for the exchange of data/information between processes. System models and properties are created in a formal structure suitable for the tools used for the model checking technique. Adapters will be developed to convert these formalisms into the form used by ROSMonitoring. In addition, adapters for exchange data contents between simulator, collision checkers, and ROSMonitoring will be developed.
Simulator coupling, and co-simulation capabilities
Simulator, system software to be verified, and ROS monitoring run on ROS framework. ROS provides time functionality. All exchange data include a timestamp, which provides synchronization between processes.
Models, notations, and formats
List of model types and modelling notations • Timed Automata models. It serves to describe system behaviour as networks of automata extended with clock and data variables. • CAD models in STL format. Solid entities for the robotic system (the robot and its environment) are represented by CAD models. The CAD models (here, in STL format) are used for the construction of the simulation environment. • URDF (The Unified Robotic Description Format) described using XML. It is an XML file format used in ROS to describe all elements of a robot. • SDF (the Simulation Description Format) described using XML. It is a complete description of everything from the world level down to the robot level.

Support for heterogeneous models and prototypes N/A
V&V Process
Description of the V&V process The workflow starts with user requirement specifications and functional descriptions. Then, the state-based models are created as well as defining formal specifications of requirements. Next, the state-based models are verified based on the formal specifications by utilizing model checking. If the models are verified based on the specifications, the next verification stage is runtime verification. Otherwise, the models are revised by the user. At the next verification stage, specifications, state-based models, and codes are used to generate a monitor needed for runtime verification. Next, runtime verification is implemented by running the monitor and the programs of the system concurrently. Collision/kinematics checkers supports the safety verification of the running programs of robotic systems. If the programs during execution are verified based on the specifications, the workflow ends. Otherwise, the user needs to revise the codes. The virtual validation approach is implemented at the second stage of the workflow, which is runtime verification.
Input information and pre-conditions As a precondition, the requirements, system functional descriptions, and behaviour models are needed. In addition, to construct a simulation environment, CAD models of the solid entities existing around the robot, the technical specifications of robots, sensors, and apparatus.
Construction approach for models and prototypes The robot, sensors, apparatuses, and its environment must be specified in SDF/URDF as similar to the real system as possible.
Output information The output is a report. It contains the violated properties at the time of system execution and the system status when the violation occurred.
Requirements traceability The model-checking tool is used to ensure that the model obeys the requirements.
Quality properties
Safety-related properties • It verifies safety properties in real-time with a minimum human intervention. • In the verification process, human-induced uncertainties are reduced. • Faults that may arise during verification are prevented from damaging the physical system and humans.
Cybersecurity-related properties N/A
Privacy-related properties N/A
Further quality properties N/A
References
Links to VALU3S deliverables • D3.1 [22] (Section 3.7.4) • D3.4 [21] (Section 2.15) • D3.5 [20] (Sections 2.18)
Links to V&V workflows Part of the workflows of UC11.
Links to resources N/A
Links to tools

- UPPAAL, https://uppaal.org/
- ROSMonitoring, https://github.com/autonomy-and-verification-uol/ROSMonitoring
- GAZEBO, http://gazebosim.org/
- WEBOTS, https://cyberbotics.com/
- FCL, https://github.com/flexible-collision-library/fcl
- ROS, https://www.ros.org

References
N/A

3.18 Simulation-Based Verification [UTRCI]

Name of the virtual validation approach: SiLVer (SimuLation-based Verification)

Introduction
Model-based simulation is a widely used methodology for early prototype validation in a variety of domains, including aerospace. In UTRCI we have developed a tool for this kind of analysis that allows us to probe a system model for requirement satisfaction / violation. Two kinds of analyses are supported currently: (i) verification, which essentially performs symbolic simulation / execution of the system given a range of input values up to a bounded time horizon, aiming to prove requirement satisfaction for the provided input range, and (ii) falsification, which performs a set of standard simulation runs attempting to reveal requirement violations.

Goals
The approach is intended to be used for analysis of the aerospace use case (UC5) in the context of the VALU3S project. At the same time, we intend to improve the tool in various aspects, throughout execution of the project, e.g. (a) partially automate translation of models and requirements necessary to generate input to be fed to the tool, (b) increase type of models for which we can perform verification (instead of just falsification), and (c) improve performance by e.g., parallelization of the underlying algorithms.

General set-up
Currently the input to the tool is manually prepared models and requirements, expressed in C++ code, and is fed to the tool along with a configuration file including input parameters as well as method to be applied (verification or falsification). The tool then analyses the given system with respect to the requirements and generates a textual report with results.

Outcomes and features

Outcomes and features

- The main outcome of the developed approach is early validation of system models with respect to requirements.
- The selling point of the developed approach is that it performs verification (falling back to falsification if necessary) of C++ code, in contrast to most verification approaches that operate on abstract models, enabling the use of the method not only on early design phases, but also on the implementation stage.
- Biggest challenges for virtual validation / prototyping that we face are (1) model fidelity with respect to real world implementation (currently addressed by being able to use C++ code as input to the tool, but also to be improved by increasing the supported subset of C++), (2) types of conducted analysis (currently we support verification for dynamical systems, but the aim is to extend this to systems with switching dynamics)

Planned outcomes and features for the future

- During execution of the VALU3S project, the plan is to (a) partially automate translation of models and requirements necessary to generate input to be fed to the tool, (b) increase type

of models for which we can perform verification (instead of just falsification), and (c) improve performance / scalability by e.g., parallelization of the underlying algorithms.
- After execution of the VALU3S project, the plan is to increase support for the subset of C++ code handled by the tool, in order to eventually be able to handle code autogenerated by Simulink.

Application domains and use cases

Domains
This approach is intended to be applied to the aerospace domain.

VALU3S Use cases
This approach is intended to be applied to UC5

Tool Chain

List of tools
- MATLAB / Simulink used for system modelling
- FRET used for requirements specification
- SiLVer (in-house developed tool) used for analysis (verification or falsification)

Tool interfaces
Models designed in Simulink and requirements specified using FRET are used to generate input for SiLVer, the tool that conducts the analysis. Presently, the translation of both model and requirements to C++ code (representation used by SiLVer) is manual, however the aim is to (at least partially) automate the process.

Simulator coupling, and co-simulation capabilities

Everything must be included in the provided model (C++ model) in order for the analysis to be carried out.

Models, notations, and formats

List of model types and modelling notations
Dynamical / Hybrid system along with solver (e.g., Runge-Kutta) provided as C++ code (using a subset of the complete syntax of C++).

Support for heterogeneous models and prototypes
No. Everything must be included in the provided model (C++ model) in order for the analysis to be carried out.

V&V Process

Description of the V&V process
The inputs to the V&V process are a model, the properties to be checked and a configuration file providing additional parameters for the V&V method to be used. The first step of the V&V process is transforming the model and properties / requirements into C++ code able to be consumed by the tool as input. This is currently done manually, however the aim is to eventually, at least partially, automate this. Once model and requirements are brought to the appropriate representation, depending on the user choice (given in the configuration file) a V&V method is performed; this can be either reachability analysis (if verification is chosen) or Monte Carlo simulation (if falsification is chosen). Finally, a textual report is generated containing the results of the analysis.

Input information and pre-conditions
Apart from the model itself, the tool also requires simulation parameters (e.g., timestep, duration) and initial conditions (e.g. input range) given in a textual configuration file, as well as the properties to be checked. These are currently provided as ad-hoc checks embedded in the model; however, the goal is to eventually provide them in the form of signal temporal logic (STL) properties (which would be automatically translated to C++ code, expected by the tool as input).

Construction approach for models and prototypes
The model given as input to the tool is C++ code. This choice was made in order to reduce the gap between model and real-world implementation. The end goal here is to eventually be able to model in Simulink and then feed the autogenerated C++ code as input to the verification tool, however at

the moment simplified handcrafted versions are used instead, as the subset of C++ supported by the tool is not big enough.
Output information Textual report with analysis (falsification / verification) results.
Requirements traceability This is handled using MATLAB/Simulink and FRET.
Quality properties
Safety-related properties Any safety-related properties that can be expressed in the supported subset of C++ code and embedded in the system model. Eventually, we aim to increase the expressiveness of property specification by also allowing signal temporal logic (STL) specifications.
Cybersecurity-related properties Any cybersecurity-related properties that can be expressed in the supported subset of C++ code and embedded in the system model. Eventually, we aim to increase the expressiveness of property specification by also allowing signal temporal logic (STL) specifications.
Privacy-related properties Any privacy-related properties that can be expressed in the supported subset of C++ code and embedded in the system model. Eventually, we aim to increase the expressiveness of property specification by also allowing signal temporal logic (STL) specifications.
Further quality properties N/A
References
Links to VALU3S deliverables The reachability analysis method implemented in the tool is the one described in D3.1 [22].
Links to V&V workflows The workflow implemented in the tool is the one described in D4.4
Links to resources N/A
Links to tools N/A
References N/A

3.19 Virtual Prototyping for Cyber-Physical Crypto Analysis [ERARGE]

Name of the virtual validation approach: Virtual prototyping for cyber-physical crypto analysis
Introduction The most important component of a cyber-physical system, especially when performing crypto-analysis of the cryptographic schemes, is the hardware-based vulnerability analysis of the random number generators (RNG). For this purpose, first, a real data output of the candidate RNG system to be evaluated is needed to be captured. The proposed virtual prototyping presents a 3-stage crypto-analysis application within the workflow of CPS. These stages are: 1. Randomness Evaluation, 2. Unpredictability Analysis, and 3. Irreproducibility and Robustness Analysis. **Goals** Evaluating the vulnerabilities of RNGs in the core of a target cyber-physical cryptosystem and

reporting the measures to be taken. Thus, the verification and validation approach is implemented on the RNGs of cryptosystems that are still in the design phase. By doing so, the secret generation step is assured to be secure which can be used either for more trusted node authentication (e.g., IoT systems) or person authentication. The RNG design is critical here at forms the basis of cryptographic key generation which is the fundamental layer of security for key distribution, key management, and end-to-end security. Thus, vulnerability analysis of such key generation mechanism, which should be based on true random number generation (TRNG, which can only be realised with hardware-based techniques to meet NIST-800-22 randomness criteria), is indispensable in verification and validation of security solutions in typical CPSs.

General set-up

The TRNG design to be evaluated and its output, i.e., generated random, should be published or shared with the employed authority (e.g., ERARGE as the experienced partner just to prove the concept). The authority then analyses the bit string with the help of a set of tools listed below. This is a semi-automated procedure as the data sharing and subsequent reporting can be implemented through a part of a V&V workflow.

The prototype is a JAVA-based online application, namely Rastgele (means "random" in Turkish) which is being developed and improved within VALU3S. It takes a bitstream, an output of an RNG, and applies the statistical randomness test, evaluate, and present the randomness test results as the reported output. On the other hand, an expert takes the RNG design document, analyses the architecture and gives feedback to the program about the main types and parameters of the RNG (e.g., case-1: chaos-based, case-2: ring-based RNG). Depending on the architecture, either scalar time series data (case-1) or bitstreams (case-2) are given as inputs, then the application applies the conditional Lyapunov exponents checking, or correlation-based analysis required for the verification and validation of the targeted cryptosystem in terms of reliability and robustness.

Outcomes and features

Outcomes and features

The main output of vulnerability analysis is the detailed report on the security assessment and crypto analysis of the cyber-physical system. This report will clearly reveal the vulnerabilities of the cryptosystem. Measures to be taken against identified vulnerabilities beyond this will be determined and shared with the system designer. Suggestions for improvement or modification of the design will be submitted to the system designer, especially the crypto-experts and particularly TRNG designers. The unique selling point of the proposed technique is that such a comprehensive crypto analysis method is already unique, that is, it is a business/method that has not been done before in the way offered in VALU3S. Previous methods usually focus on the randomness analysis itself but the proposed technique also encounters unpredictability, irreproducibility and robustness analyses of TRNGs. Since this method analyses TRNGs, which are the basis of highly critical and confidential projects where security is a must, it will be of interest to many institutions and organizations.

With this approach, we have addressed just for cyber-physical crypto analysis of chaotic oscillator based and ring oscillator based systems. We have solved the issue of detailed analysis of RNGs addressed. Other types of RNGs that are not based on FPGA or FPAA are excluded from this approach. The proposed method is not fully automated, thus it requires expert work. The analysis results should also be interpreted with high-level security assessment (e.g., network intrusion detection, cloud security) within the context of the targeted CPS.

Planned outcomes and features for the future

Vulnerability analysis of cryptographic modules against hardware-based attacks has been designed and developed within the scope of the VALU3S project. This method has been introduced in WP3. Consequently, our primary goal is to test our method in UC11 and then generalize the method for other use cases of VALU3S. We plan to consider as many different TRNG structures as possible (at least 2 types planned). Automating this whole method and presenting it as an external service in the long term after the project will also be among the long-term outputs.

Application domains and use cases
Domains This approach can be applied in all domains that need cyber-physical security. We plan to implement this approach within UC11 under VALU3S, that is, in the Industrial Domain. Then, the strategies and recommendations for generalizing the proposed scheme for other use cases are planned to be reported e.g., in scientific papers.
VALU3S Use cases The "Virtual prototyping for cyber-physical crypto analysis" approach will be applied for ERARGE HSM and Secure Gateway which will utilize the TRNG efficiently. By doing so, the proposed vulnerability analysis method will be demonstrated (mainly in UC11) and the countermeasure which applies the effective use of TRNGs will be presented.
Tool Chain
List of tools The underlying tools that are used for cryptanalysis are as follow: • *iPerf*: iPerf3 is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test, it reports the bandwidth, loss, and other parameters. • *Anadigm Designer2*: Second-generation AnadigmDesigner®2 EDA software lets you design and implements dynamically reconfigurable analogue circuits within a matter of minutes. Build your circuit by dragging and dropping Configurable Analog Modules (CAMs), each of which can be used to implement a range of analogue functions for which you set the parameters. • *Xilinx Vivado Design Suite*: Vivado Design Suite delivers an SoC-strength, IP-centric and system-centric, next-generation development environment that has been built from the ground up to address the productivity bottlenecks in system-level integration and implementation. • *BigCrush*: Empirical statistical tests perform checks to determine whether a random number generator produces random numbers which behave the way we expect random numbers to behave. One of the most well-respected statistical test suites is TestU01 • *DieHard*: The diehard tests are a battery of statistical tests for measuring the quality of a random number generator. They were developed by George Marsaglia over several years and first published in 1995 on a CD-ROM of random numbers • *NIST 800-22*: The NIST Test Suite is a statistical package consisting of 16 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software-based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests.
Tool interfaces The tools to be used are not hierarchical. They can be applied to TRNG output data to be evaluated in series or parallel. Anadigm Designer and Vivado Designing Suite tools will be used to review and edit RNG designs. One can easily work with documents in a format suitable for these design programs. TRNG output data must be at least 1 million bits long that can be in formats such as .bit or .hex. These file types can be used in randomness tests. The randomness test outputs are logic outputs. They produce a positive or negative binary result presenting whether the system has passed the test or not.
Simulator coupling, and co-simulation capabilities
We do not yet use simulator coupling or co-simulation.
Models, notations, and formats
List of model types and modelling notations

For cryptanalysis, we only need raw output data known to be originated from the evaluated RNG. There is no model in which this can be tested. The file format can be in .bit or .hex format.
Support for heterogeneous models and prototypes N/A
V&V Process
Description of the V&V process Before cryptanalysis of the cyber-physical system, output data must be collected from the analysed system (at least 1 million bits). The collected data should first be tested with the NIST 800-22 randomness test suite, and then they should be subjected to the more stringent big crush and DieHard tests, the order of which is not very important. After these tests, the design of the RNG can be analysed. For this purpose, Vivado Design Suite and/or Anadigm Designer tools are used. The Iperf tool UC11 will also be used by ERARGE to measure the data communication speed of the HSM and Secure gateway hardware that will be integrated into the OTOKAR system (UC11). Likewise, RNG analyses will be performed on Secure Gateway and HSM which are specifically designed cyber-physical security components produced by ERARGE.
Input information and pre-conditions For cyber-physical system/design crypto analysis, design files must be executable and editable with Anadigm Designer or Vivado Design Suite. If there is a circuit other than this, the PCB layout and the schematic can also be taken as input. For RNG outputs, there should be at least 1 million bits of data, for a high-quality result, at least 1GB of data should be collected for more accurate vulnerability analyses.
Construction approach for models and prototypes Manual preparation of data outputs and design documents of the system to be cryptanalyzed should be prepared.
Output information The output of the V&V method is a feedback report. It can be in .doc or .pdf format. This report document offers suggestions to the designer of the cryptanalyzed cyber-physical system and provides feedback on the robustness and compliance of the designed system with the standards. Thanks to this report, the designer corrects or changes his cyber-physical system.
Requirements traceability Since the general operation of the method is semi-automatic, the person who operates the method should also follow the requirements. This will be handled by expert's scrutinization and elaboration that relies on the quantitative analysis of randomness, robustness and reliability checking.
Quality properties
Safety-related properties N/A
Cybersecurity-related properties The cryptanalysis of a cyber-physical system can be approached in many ways, but the most important components of such systems are key generation and thus the underlying RNG. The open crypto algorithms (AES, RSA etc.) on which the cyber-physical system is based are implemented by the outputs of RNGs (e.g., crypto keys), which are secret and cannot be inferred about how their random numbers are generated. We also analyse the RNG of the system, which is handled with the tools we use, and we produce a report on its immunity to external attacks.
Privacy-related properties N/A
Further quality properties N/A
References

Links to VALU3S deliverables

- D3.1 [22] (Section 3.6.1.7)
- D3.4 [21] (Section 2.34)
- D3.5 [20] (Section 2.4)

Links to V&V workflows

- WF18: Implementation of vulnerability analysis methods proposed by ERARGE for analytical, simulation and experimental based verification of robotics systems' secure communication.

Links to resources

- NIST 800-22: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf

Links to tools

- iPerf: https://iperf.fr/
- Anadigm Designer2: https://www.anadigm.com/anadigmdesigner2.asp
- Xilinx Vivado Design Suite: https://www.xilinx.com/products/design-tools/vivado.html
- BigCrush: http://simul.iro.umontreal.ca/testu01/tu01.html
- DieHard: https://webhome.phy.duke.edu/~rgb/General/dieharder.php
- NIST 800-22: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf
- NIST Statistical Test Suite https://github.com/stevenang/randomness_testsuite

References

N/A

Chapter 4 Conclusion

This document describes concepts and solutions for virtual validation and virtual prototyping, which enable early and efficient quality assurance in complex product development processes. Major challenges for applying and exploiting the novel validation concepts are described as well as concrete approaches for virtual validation and virtual prototyping are presented.

The validation concepts will be further extended and implemented in the project. The results will be presented in D4.11 - *Virtual engineering simulation-based verification, for coupling tools, and enabling missing drivers and environment* data in M30.

References

[1] L. Mikelsons and R. Samlaus, "Towards Virtual Validation of ECU Software using FMI," *12th International Modelica Conference,* pp. 307-311, 2017.

[2] "What Is Virtual Validation?," SIEMENS, [Online]. Available: https://www.plm.automation.siemens.com/global/de/our-story/glossary/virtual-validation/57459. [Accessed 21 9 2021].

[3] T. de Schutter, Better Software. Faster! Best Practices in Virtual Prototyping, Synopsys, 2014.

[4] G. Wang, "Definition and Review of Virtual Prototyping," *J. Comput. Inf. Sci. Eng.*, vol. 2, no. 3, pp. 232-236, 2002.

[5] C. Allmacher, M. Dudczig, P. Klimant and M. Putz, "Virtual Prototyping Technologies Enabling Resource-Efficient and Human-Centered Product Development".

[6] L. E. Bjoerkl, "MixedUX: A Mixed Prototyping Framework for Usability Testing in Augmented Reality," Norwegian University of Science and Technology Trondheim (NTNU), 2015.

[7] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems*, Springer, 2016, pp. 85-113.

[8] H. Yin and L. Wang, "Application and Development Prospect of Digital Twin Technology in Aerospace," *IFAC-PapersOnLine,* vol. 53, no. 5, pp. 732-737, 2020.

[9] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne and L. Wang, "Modeling, simulation, information technology & processing roadmap.," National Aeronautics and Space Administration, 2012.

[10] S. Boschert, C. Heinrich and R. Rosen, "Next Generation Digital Twin".*Proceedings of the TMCE 2018.*

[11] "Digital Twins - The Birth of Constant Optimization," SIEMENS, [Online]. Available: https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/digital-twin.html. [Accessed 28 9 2021].

[12] A. Madni, C. Madni and S. Lucero, "Leveraging Digital Twin Technology in Model-Based Systems Engineering," *Systems,* vol. 7, no. 1, p. 7, 2019.

[13] A. Rasheed, O. San and T. Kvamsdal, "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective," *IEEE Access,* vol. 8, pp. 21980-22012, 2020.

[14] E. Tuegel, "The airframe digital twin: some challenges to realization," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Honolulu, Hawaii, 2012.

[15] B. Gockel, A. Tudor, M. Brandyberry, R. Penmetsa and E. Tuegel, "Challenges with Structural Life Forecasting Using Realistic Mission Profiles," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Honolulu, Hawaii, 2012.

[16] E. H. Glaessgen and D. S. Stargel, "The digital twin paradigm for future NASA and U.S. Air force vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Honolulu, 2012.

[17] J. D. Hochhalter, W. P. Leser, J. A. Newman, E. H. Glaessgen, V. K. Gupta, V. Yamakov, S. R. Cornell , S. A. Willard and G. Heber, Coupling Damage-Sensing Particles to the Digitial Twin Concept, Langley Research Center: National Aeronautics and Space Administration, 2014.

[18] S. Tonetta, "ROBDT," FBK, 16 September 2021. [Online]. Available: https://es.fbk.eu/index.php/projects/robdt/.

[19] P. Oliveira Antonino, M. Jung, A. Morgenstern, F. Faßnacht, T. Bauer, A. Bachorek, T. Kuhn and E. Y. Nakagawa, "Enabling Continuous Software Engineering for Embedded Systems Architectures with Virtual Prototypes," *ECSA,* pp. 115-130, 2018.

[20] AIT et al., "Deliverable D3.5 - Interim description of methods designed to improve the V&V process," VALU3S Consortium, October 31, 2021.

[21] AIT et al., "Deliverable D3.4 - Initial description of methods designed to improve the V&V process," VALU3S Consortium,, April 30, 2021.

[22] UCLM et al., "Deliverable D3.1 - V&V methods for SCP evaluation of automated systems," VALU3S Consortium, December 31, 2020.

[23] RULEX et al., "Deliverable D3.3 - Identified gaps and limitations of the V&V methods listed in D3.1," VALU3S Consortium, April 30, 2021.

[24] IMTGD et al., "Deliverable D4.1 - Initial plan on designing the improved process workflows," VALU3S Consortium, April 30, 2021.

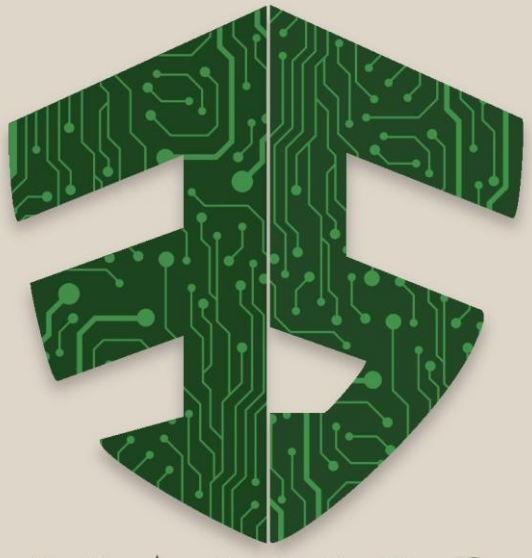
[25] NXP et al., "Deliverable D1.1 - Description of use cases as well as scenarios," VALU3S Consortium, August 31, 2021.

[26] UTRCI et al., "Deliverable D1.2 - SCP requirements as well as identified test cases," VALU3S Consortium, December 31, 2021.

[27] UTRCI et al., "Deliverable D1.3 - Detailed description of the test cases with respect to different dimensions/layers of the framework," VALU3S Consortium, April 30, 2021.

[28] MGEP et al., "Deliverable D2.1 - Initial multi-dimensional layered framework," VALU3S Consortium, August 31, 2021.

[29] MGEP et al., "Deliverable D2.2 - Final multi-dimensional layered framework," VALU3S Consortium, December 31, 2021.



VALU3S

www.valu3s.eu

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.