Continuous Engineering for Trustworthy Learning-enabled Autonomous Systems.[⋆]

Saddek Bensalem[1], Panagiotis Katsaros[2], Dejan Ničković[3], Brian Hsuan-Cheng Liao[4], Ricardo Ruiz Nolasco[5], Mohamed Abd El Salam Ahmed[6], Tewodros A. Beyene[7], Filip Cano[8], Antoine Delacourt[9], Hasan Esen[4], Alexandru Forrai[10], Weicheng He[1], Xiaowei Huang[11], Nikolaos Kekatos[2], Bettina Könighofer[8], Michael Paulitsch[12], Doron Peled[13], Matthieu Ponchant[9], Lev Sorokin[7], Son Tong[14], and Changshun Wu[1]

[1] University Grenoble Alpes, VERIMAG, Grenoble, France
[2] School of Informatics, Aristotle University of Thessaloniki, Greece
[3] AIT Austrian Institute of Technology, Vienna, Austria
[4] DENSO AUTOMOTIVE Deutschland GmbH, Eching, Germany
[5] RGB Medical Devices, Spain
[6] Siemens EDA, Cairo, Egypt
[7] fortiss GmbH, Munich, Germany
[8] Graz University of Technology, Graz, Austria
[9] Siemens Industry Software SAS, France
[10] Siemens Digital Industries Software, The Netherlands
[11] University of Liverpool, Liverpool, U.K.
[12] Intel Labs, Germany
[13] Bar Ilan University, Israel
[14] Siemens Industry Software NV, Belgium

Abstract. Learning-enabled autonomous systems (LEAS) use machine learning (ML) components for essential functions of autonomous operation, such as perception and control. LEAS are often safety-critical. The development and integration of trustworthy ML components present new challenges that extend beyond the boundaries of system's design to the system's operation in its real environment. This paper introduces the methodology and tools developed within the frame of the FOCETA European project towards the continuous engineering of trustworthy LEAS. Continuous engineering includes iterations between two alternating phases, namely: (i) design and virtual testing, and (ii) deployment and operation. Phase (i) encompasses the design of trustworthy ML components and the system's validation with respect to formal specifications of its requirements via modeling and simulation. An integral part of both the simulation-based testing and the operation of LEAS is the monitoring and enforcement of safety, security and performance properties and the acquisition of information for the system's operation in its environment. Finally, we show how the FOCETA approach has been applied to realistic continuous engineering workflowsfor three different LEAS from automotive and medical application domains.

Keywords: Learning-enabled Autonomous Systems · machine learning · safety · security · continuous engineering · formal analysis.

⋆ **Acknowledgment**: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 956123.

1 Introduction

The crucial criteria for the design of *learning-enabled autonomous systems* (LEAS) are correctness and safety, especially for real-world operability. The complexity of LEAS stems, to a large extent, from the interplay between classically engineered and learning-enabled components (LECs). In current design practice, correctness and safety are formulated by requirements that must be satisfied by the LEAS. However, the presence of LECs is not addressed in an adequate manner in this context; there is need for new design, verification and validation (V&V) approaches that take into account the presence of LECs in LEAS, address their qualitative and quantitative capabilities throughout their entire life cycle and consider the real-world complexity and uncertainty.

A holistic engineering approach will adequately assess the correctness and safety of LEAS operating in the real world. Such an analysis requires that the engineering method employed will determine the (quantitative) contexts in which the correctness and safety of systems will be (qualitatively) evaluated. Integrating system design and operation sets more specific considerations for LEAS and the engineering approach: their safety will have to be assessed with respect to an ever evolving set of (critical) scenarios, instantiating difficult conditions for their sensors and underlying algorithms.

In the FOCETA project, we provide a holistic methodology for designing and addressing the correctness and safety of LEAS, bridging the gap between the currently applied development, verification, and validation techniques for LE systems and their operation in the real world. We allow updates to the LECs, in response to emerging requirements from new scenarios, imperfect knowledge of the machine learning models (noise in data observations) or contextual misbehavior of them and possible security threats. This need is addressed within a continuous development and testing process mixing software development and system operations in iterative cycles.

In this process, formal specifications allow designers to formulate requirements in a rigorous manner. They are used throughout the whole system life-cycle, both during development and operations. During the concept design phase, formal specifications can improve the process of engineering requirements by making them precise. This facilitates communication between design teams, removes ambiguities in the exchange of knowledge and renders the requirements verifiable. During the system implementation, specifications can be used for formal verification of critical components, but also as oracles in the testing activities. Finally, formal specifications can be monitored during system operation to detect violations of requirements (runtime verification) and take corrective actions (runtime enforcement). The role of specification languages, fairly well understood in the context of classical system design, is more ambiguous when reasoning about LEAS. The addition of LECs considerably adds to the system's complexity and requires rethinking the entire development and operation process, including the role of formal specifications, which we explain in this document.

The FOCETA methodology (Figure 1) includes two design flows. The first for designing trustworthy LECs, and the second for integrating them with classically engineered components, resulting in correct and safe LEAS, through iterative cycles of development and system operation. Our methodology relies on verifying simulation results and collecting additional data for system improvement. Runtime monitoring is used to supervise LEAS decision-making and record environmental situations.

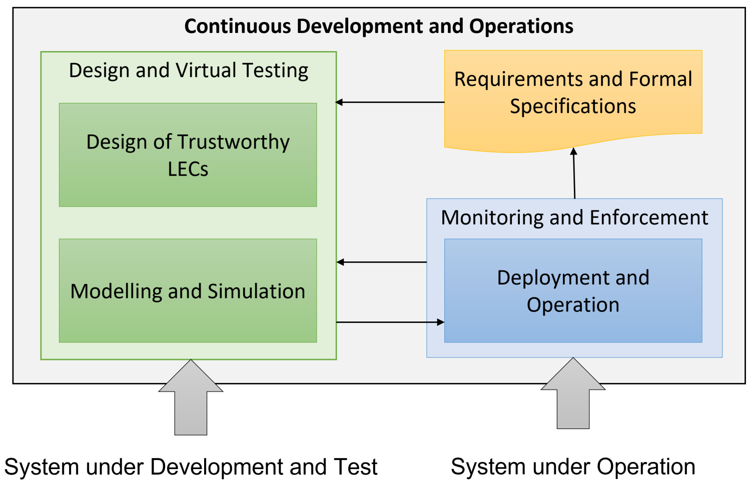

Continuous Development and Operations
Design and Virtual Testing
Design of Trustworthy LECs
Modelling and Simulation
Requirements and Formal Specifications
Monitoring and Enforcement
Deployment and Operation
System under Development and Test
System under Operation

Fig. 1: Continuous development and operations of LEAS

The contributions of this paper are the following. First, we introduce the main goals of the two design flows. Then, we present the key technological developments, i.e. the methods and tools, that are the most representative for realizing our methodology:

- techniques for trustworthy perception, sensing and safe data-driven control
- a technique for trustworthy updates of LECs
- a methodology and tool for continuous specification, validation and update of LEAS requirements
- a compositional digital twin architecture and tools for the formal modeling, simulation-based analysis and runtime verification of LEAS
- techniques for efficient virtual testing
- specification mining techniques for learning system properties while the LEAS operates in its environment

At the end, we show how these techniques are combined in LEAS case studies.

2 Design Flow for Trustable Learning Components

Deep learning is a mathematical framework for inferring (predictive) models from data. Its main feature is to use the prediction score as a feedback signal to adjust the weights' value slightly in a direction that will lower the loss score. This tuning is an optimization process which implements the leading deep learning algorithm to design the neural network model. Machine learning can be considered as input-target matching, done by observing different examples of inputs and targets. The learning process consists of finding a set of values for the weights for all the network layers that correctly match the inputs to their associated targets. Finding the correct value for each of them can seem like a daunting task, since changing the value of one parameter will affect the behavior of all the others; a deep neural network can contain more than a trillion of parameters [33].

A network with a minimal loss is one for which the outputs are as close as possible to the targets, i.e. a trained network. Initially, the network weights are assigned random values, so the network implements a series of random transformations; its output is far from what it should ideally be and the loss score is very high. However, the weights are adjusted in the correct direction and the loss score is decreased. This training loop, repeated several times, yields weight values that minimize the loss function. This iterative process only considers the optimization of the resulting neural network model. We propose, in a complementary way, rigorous methods and tools that guarantee the trustworthiness of the neural network model obtained at the end of this iterative process.

The rigorous design approach for developing a safe and trustworthy LEC heavily depends on the LEC's functionality. We consider mainly two classes of LECs - perception modules and data-driven controllers. Designing a trustworthy perception module is very different from designing a safe data-driven controller. Also, within the context of continuous engineering, we address the problem of replacing LECs with others.

2.1 Trustworthy Perception and Sensing

An object detection component based on a neural network (NN) does not admit a natural formal specification of its expected behavior and characterizing a correct perception module's behavior using mathematics and logic is notoriously hard. It is even hard to formulate an appropriate verification and validation question for such LECs. There are, however, a number of more indirect but still effective approaches to reason about correctness of such components that we explore in FOCETA. Figure 2 provides a general framework illustrating how a machine learning based perception component might be developed and certified. It refines the usual development cycle, which forms three stages - training, offline V&V (or testing), and deployment - by including a collection of activities that support the certification of the component.

Falsification and Explanation The first approach consists of evaluating low-level properties of the LEC, such as its *robustness* to input perturbations, adversarial attacks and the *explanation* to the perception task. This belongs to the family of design-time testing approaches. Robustness defines how sensitive the NN is to small perturbations in the input. There are situations where a change of a single pixel in a picture can result in a mis-classification or mis-localization, even for state-of-the-art NNs [37,42]. As the first step towards a complete evaluation, the general robustness of a NN can be assessed with testing strategies, resulting in research on defining meaningful notions of coverage for NNs, fault-injection mechanisms for LECs and adversarial testing methods. For example, in FOCETA, coverage-guided testing [19] and distribution-aware testing [20] have been developed. In addition, simulated hardware faults have been found effective in testing NN performance [34]. Within the context of continuous engineering, it is also possible to apply refinement techniques on top of the commonly known combinatorial testing scheme for LECs [8], in order to avoid the need of complete re-verification. For explanation, saliency maps are usually generated to highlight the important features

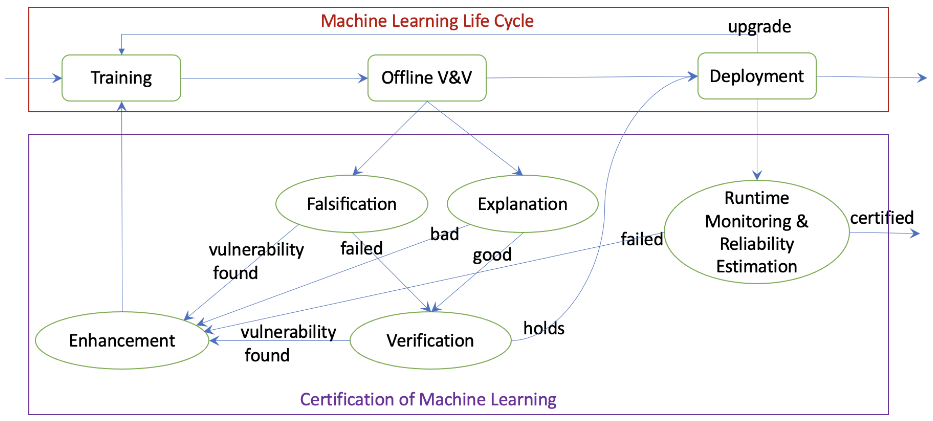


Fig. 2: Continuous development of machine learning based perception component

of an input instance; we consider further the robustness of the generated explanations [46,21].

Verification While the above methods may be effective in finding bugs in NNs, they cannot be used to prove their correctness. This problem is typically tackled using formal verification. For example, formal verification and optimization can be applied to find the maximum resilience of the NN to local perturbations [26]. The challenge for verification is its scaling to LECs of realistic size; to deal with this, research has been extended to the verification over geometric transformations [43] rather than pixel-level changes.

Enhancement The counterexample found through either falsification, explanation, or verification, can be utilised for enhancing the machine learning model. This can be done through fine tuning, which slightly adjusts the weights according to the counterexamples. The other mainstream, and arguably more effective, approach is through adversarial training, which adapts the training algorithm to consider the properties. Towards this, we have considered the enhancement to generalisation and robustness [23,22].

Runtime Monitoring and Reliability Assessment Most of the falsification, explanation, and verification approaches are limited to point-wise analysis and are typically subject to the selection of testing data points. To support the deployment, it is necessary to be able to work with any data that may appear during operation. Two methods are considered. First, we developed runtime monitoring techniques for LECs. In general, these techniques can be divided into white-box and black-box approaches. The former class requires access to NN parameters. For example, neuron activation patterns or interval values can be recorded over the training dataset and applied as an abstraction for runtime monitoringe [9,18,41]. Another approach of online monitoring developed in FOCETA uses small efficient neural networks to detect either hardware faults or input abnormalities efficiently as presented and shown in [14]. The black-box approaches build verdicts based on the NN inputs and outputs. For example, temporal consistency of the detected objects can be modelled as a post-algorithmic abstraction and can be applied at runtime [7]. A variant of temporal logic is used to define intended relations in time between detected objects, e.g. forbidding the situation where a detected object of type A suddenly becomes classified as object of type B in the next frame. Detecting such temporal inconsistencies in object detection algorithms could help to prevent accidents such as the Arizona accident with the Uber test vehicle in 2018, in which the vehicle could not correctly and persistently identify a bicycle before an imminent collision. Second, we employ an operational profile, which has been evolved from the field of programming and testing, to model operational data, and then estimate the operational time reliability of the component by considering both the operational model and the verification method [44,45].

2.2 Safe Data-Driven Control

The design of data-driven controllers can benefit from several aspects used for the design of classical control applications. In particular, data-driven control naturally admits formal specifications for expressing its intended behavior. In FOCETA, we identify and investigate two complementary approaches for developing safe control applications that use the formal specification to formulate their functional requirements. The first approach uses formal specifications to guide the training of the controller towards exhibiting safe behavior [6]. The second approach builds an external mechanism, often called a shield, to prevent an untrusted data-driven controller from doing unsafe actions. A *safety shield* is an enforcer that overwrites commands from the controller whenever executing the command could result in a safety violation. It extends the known safety supervision approaches of Responsibility Sensitive Safety[15] and Safety Force Shield[16] of major automated driving provider. Safety is defined via a temporal safety specification φ in linear temporal logic and a threshold $\delta \in [0, 1]$. During runtime, the shield overwrites any command from the controller for which the probability of violating φ is larger than δ. A shield is automatically computed from the safety specification and a model of the environment using value iteration. Thus, a shield guarantees that safety is satisfied under the assumption that the underlying environmental model accurately captures the safety relevant dynamics of the environment.

2.3 Trustworthy Updates of LECs

For a number of reasons, LECs usually have to be updated within the context of continuous engineering. For example, security concerns such as the need to withstand data perturbations (e.g. adversarial examples) or out-of-distribution data cases, trigger the need to replace LECs with improved ones that exhibit functionally comparable behavior in certain important aspects with the ones to be replaced.

In FOCETA, in addition to the enhancement approaches discussed in Section 2.1, this problem is addressed [12] by *formally verifying the equivalence* of two LECs as follows: for two pretrained NNs of different architectures, or of the same architecture

[15] https://www.mobileye.com/technology/responsibility-sensitive-safety/
[16] https://www.nvidia.com/en-us/self-driving-cars/safety-force-field/

with different parameters, we check whether they yield similar outputs for the same inputs. Contrary to strict functional equivalence that is desired for software component design, similar outputs do not necessarily mean identical outputs in the case of NNs. For example, two NNs used for classification may be considered equivalent if they always select the same top output class, even though the remaining output classes are not ordered in the same way. The definition of equivalence depends on the application and NNs at hand (e.g. classifier, regression, etc.). Therefore, we considered strict, as well as approximate equivalences and formalized them as relations characterizing the similarity of NN outputs, for identical inputs. Equivalence is essential for replacing one LEC by another, but it does not ensure by itself the transfer of any other guarantees to the updated LEC. When an improved LEC has been developed that fulfills some additional requirements or that withstands evident security threats, not previously identified, equivalence checking is applied to the outdated and the new LEC, out of the LEAS's overall context.

3 Design Flow for LEAS

The design flow for LEAS consists of two parts: (1) the design-time part including the design, implementation, and verification of the system, and (2) the run-time part, focusing on the deployment of the LEAS and its operation in the real world.

The current state of practice is extended towards the transfer of knowledge about systems and their contexts (e.g. traffic situations) from the development to operations and from the operations back to development, in iterative steps of continuous improvements. Over the complete life cycle of LEAS - from specification, design, implementation, and verification to the operation in the real world – the methodology enables their continuous engineering with particular focus on the correctness with respect to an evolving set of requirements and the systems' safety. Moreover, the whole design flow enables traceability between the requirements and the system/component design.

A key feature is the use of runtime monitoring for the seamless integration of development and operations. Monitors observe a system (part) via appropriate interfaces and evaluate predefined conditions and invariants about the LEAS behavior based on data from these interfaces. This allows the monitor to identify needs for LEC/other component updates during continuous engineering, if, for example, some data in a test scenario result in a safety violation or if a new requirement will emerge during system's testing/operation.

3.1 Requirements and Formal Specifications

Requirements specification and semantic validation. In continuous engineering, it should be possible to take into account additional requirements, to address needs which come up from scenarios that have not been taken previously into account. The specification of these additional requirements should not pose issues of consistency, ambiguity and completeness with respect to the existing requirements. Moreover, any additional requirements will have to be appropriate such that together with the existing ones will make it possible to devise an acceptable (i.e. feasible and cost-efficient) LEAS design.

We have introduced an ontology-based approach/tool for specifying, validating and formalizing LEAS requirements during the design phase of the continuous engineering iteration cycles [31]. Requirements are expressed in controlled natural language restricted to terms from an ontology with precisely defined concepts and semantic relationships in the system's domain (domain specific ontology - DSO). To tackle the lack of a unique interpretation for the natural language syntax, we employ boilerplates, i.e. textual templates with placeholders to be filled with ontology elements. These elements are semantically interrelated and are part of a well-defined ontology architecture.

The whole semantic framework, together with the ontology terms mentioned in requirements, enable automated semantic analyses that guide the engineer towards improving the requirements specification. These analyses detect specification flaws.

However, even if the requirement specifications are semantically validated, this does not guarantee that they are satisfied by the LEAS under design. The requirements must be transformed into formal specifications of monitorable properties (formalisation) and then mapped to a component-based simulation model of the LEAS. This transformation is based on predefined mappings of boilerplates to logic-based property specifications and it is supported by our requirements specification and formalisation tool.

Formal specification of requirements. Adapting an existing specification formalism to LEAS is a challenging task because of the presence of LECs, such as CNN-based object detection modules, which do not admit natural logic-based formulation of their safety requirements (see Section 2). In FOCETA, we suggested past first-order linear temporal logic (P-FO-LTL) [5] as the main specification formalism to reason about LEAS. It abstracts away the internal structure of the NN-based LEC, including the internal values of the different neurons. This simple, yet expressive formalism allows the user to (1) quantify over (possibly unbounded number of) agents, (2) define timing constraints between events and (3) naturally synthesize online monitors. The runtime verification of P-FO-LTL is implemented in the DejaVu tool [17] used in the project.

We also use Signal Temporal Logic (STL) [30] as a popular specification language among the cyber-physical systems community. The main advantage of STL is that it naturally admits *quantitative semantics* and hence allows one to measure *how far* is the behavior from satisfying or violating a specification. A disadvantage of STL compared to P-FO-LTL is that it does not allow quantification over agents. The quantitative runtime STL monitors are implemented in the RTAMT library [32]. Still it is evident that sequential specifications based on temporal logic cannot capture every aspect during design of LEAS. For example, many core properties of LECs, such as robustness of the NN discussed in Section 2, do not admit natural formal specification using logic.

3.2 Simulation-based Modeling, Testing and Monitoring at Design Time

For design-time V&V of LEAS, FOCETA invests on formal modeling and simulation-based analysis. Formal modeling is a prerequisite for specifying correctness and safety properties and eventually verifying them. Simulation-based testing provides a cost-effective means to verify the LEAS performance over diverse parameter ranges and to generate massive sets of scenarios. Critical scenarios can be identified in the virtual environment, thus limiting those needed to be replayed in the much more expensive

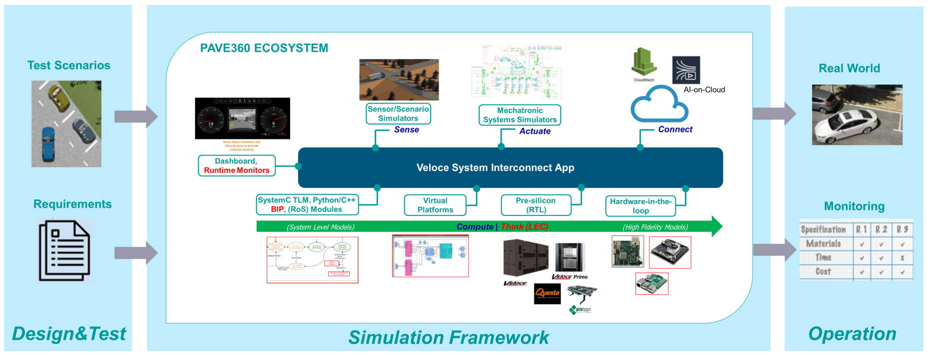


Fig. 3: Compositional Simulation Architecture for Digital Twins of LEAS

physical setting. Simulation-based testing also allows generating scenarios (e.g. very rare events) that may be impossible to realize when the LEAS operates in its environment (e.g. with specific weather conditions, such as snow). Finally, it also enables creating safety-critical situations (e.g. crashes) without compromising the safety of the real-life actors.

Formal modelling. For the formal modeling of LEAS, a component-based modeling approach is introduced that allows mixing model-based and LE components with digital twin simulation [39]. This is achieved by having extended the BIP component framework [4], in order to enable the design of executable models with formal semantics for the LECs. In this extended BIP framework, LECs are represented by atomic BIP components which can make machine learning inference.

Extended BIP models have been integrated into the FOCETA Compositional Simulation Architecture for virtual testing (Figure 3) using Veloce System Interconnect (VSI) [4]. VSI is used as a middleware to connect different tools and models. It enables integration of heterogeneous components and models that are designed with different tools and allows mixing discrete modeling for the computing elements with continuous modeling for the physical components. In essence, the engineer can use diverse tools/models at different abstraction levels for modeling the sensing, control - actuation functions, the system's physical dynamics, and all interactions with its environment, as long as these tools/models can be integrated via standards-based interfaces (FMI [40] or TLM).

V&V at design time is based on virtual testing. This encompasses all cases in which one or more physical elements (software, hardware) of the LEAS are replaced by their simulation model(s). The FOCETA Compositional Simulation Architecture supports model-in-the-loop (MiL), software-in-the-loop (SiL) and hardware-in-the-loop (HiL) testing of the system's functions, at various abstraction levels. MiL testing is focused on design correctness, the performance of LECs, the control strategies and the associated trade-offs. Having done MiL testing, fine-tuning processes using different mechanisms can be applied, such as requirement-encoded and safety-oriented loss function [27] for object detection or uncertainty-aware loss function [24] for trajectory prediction tasks.

SiL testing is used to check the correctness of code in closed loop with a model of the physical system. Finally, HiL testing consists of real-time simulations that include

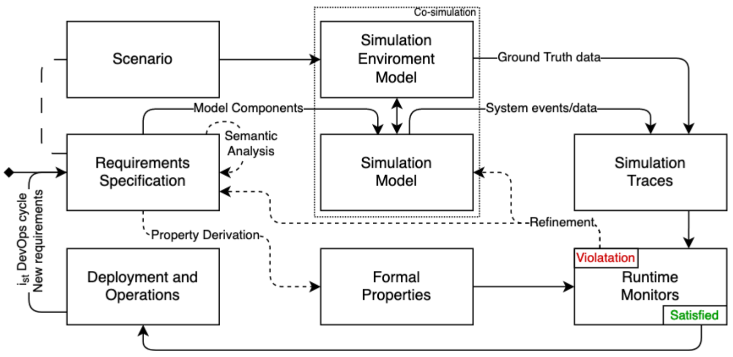


Fig. 4: FOCETA requirements specification/formalization approach

the target hardware; the hardware-specific impact can be also simulated. A production-friendly approach has been developed with the PyTorchALFI tool chain [29], which allows to efficiently perform fault injection [16].

For verifying safety properties for the overall system model, the online formal analysis of simulation traces is supported. Within the context of the FOCETA Compositional Simulation Architecture, this takes place through the integration of runtime monitors, which are generated by the DejaVU and RTAMT monitor synthesis tools based on formal specifications in P-FO-LTL [38] and Signal Temporal Logic (STL), respectively. P-FO-LTL is the target language of the FOCETA requirements specification/formalization approach that is summarized in Section 3.1 (Figure 4).

Simulation allows the designer to control the execution of scenarios and efficiently explore the system's operational design domain. Scenarios define the temporal evolution between several snapshots of the environment's state in a sequence; they are specified as a set of actions and events, goals (e.g. staying between the lane markings) and values (e.g. prioritize safety of pedestrians).

To identify safety-critical scenarios, we adopt search-based testing (SBT), in which the testing is framed as an optimization problem that guides the search toward finding interesting/failure-revealing test cases. Such testing can be performed by applying the modular and extendable SBT framework OpenSBT [36] or a toolchain consisting of the Simcenter Prescan [17] and HEEDS [18] tools. In particular, Simcenter HEEDS provides solutions for optimization, test orchestration, and visualization and analysis, whereas Simcenter Prescan is a physics-based simulation platform for advanced driver assistance systems that has been integrated into the FOCETA Compositional Simulation Architecture. In addition to classical SBT, we also explored black-box testing combined with light-weight learning [13] to accelerate generation of relevant tests.

[17] https://www.plm.automation.siemens.com/global/en/products/simcenter/prescan.html
[18] https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-heeds.html

3.3 Deployment, Operation and Analysis of LEAS at Runtime

The analysis of LECs and their integration into LEAS during design time, together with the protective mechanisms synthesized around LECs support the safety assurance of the overall system during real-time operation. These measures are complemented by runtime verification that plays a central role during the LEAS operation. Runtime monitors allow users to observe the system and its interaction with the environment and gather useful information regarding (1) violations of safety or other requirements, (2) new operational scenarios that were not captured by the training data and/or models, and (3) other unexpected situations and anomalies that are not characterized by the existing set of requirements. In order to be effective, monitors must be present both at the component level (LE and classical) and at the system level; the information gathered by different monitors must be fused to derive useful information that can be used to (1) ignore the situation (e.g. detected object misclassification) that does not impact the system-level control decision, (2) take a protective measure (e.g. switch from the advanced to a base controller) or (3) improve the design (e.g. provide a new scenario for the training data).

The last point refers to the concept of *evolvable* LEAS, in which information from the system operation is collected and used to go back to the design and enhance its functionality based on new insights, thus effectively closing a loop between the design and the operation phase. To enable evolvable LEAS, two questions need to be answered: (1) how to extract and summarize information from raw observations, and (2) how to use such information to increase the quality of the design.

In FOCETA, we advocate specification mining [2] as one approach to answer the first question. It is the process of learning system properties from observing its execution and the behavior of its environment. We use inferred specifications to understand the specificities of the system behavior, characterize its operational design domain (ODD) and identify system aspects that can be improved. However, specification mining has broader application potentials. Specifications mined from a system can be also used to complete the existing incomplete or outdated specifications, confirm expected behaviors, and generate new tests. Answering the second question is highly dependent on the specific component or sub-system that needs to be updated and improved over time. For example, we developed a proof-on-demand mechanism [28] to expand the region in which a data-driven controller can safely function without the need to activate the shield while the system is operating

3.4 Assurance Cases for LEAS

The continuous engineering of safety-critical LEAS requires constructing and maintaining an assurance case. In FOCETA, we adopted the off-the-shelf continuous integration framework Evidential Tool Bus (ETB) [10]. ETB allows: 1) the automated and decentralized execution of V&V tools to provide assurance evidence for an assurance case construction, and 2) incrementally maintain an assurance case. An incremental update is required when, for instance, the system under assurance changes, e.g., when the requirements of the system (e.g., its ODD) must be updated either after inference of corner case properties at runtime with specification mining or when the system must be deployed in another country with different regulations.

4 Case Studies Demonstrating the FOCETA Methodology

4.1 Traffic speed detection and Path Lane Following

Figure 5 depicts a SiL setup that shows how digital twin simulation is extended with formal analysis and runtime monitoring to verify the functionality of a traffic speed detection and path lane following system. In this case study, a formal model was developed for the perception and throttle/brake control modules of the LEAS using the BIP component framework. This executable model was then integrated with a Simcenter Prescan model for the environment, a Simcenter Amesim model for the EGO vehicle dynamics and a ROS-based implementation of the steering control for path lane following and twist control. The vehicle has an RGB camera sensor. The camera feed is sent to the BIP model for speed sign detection and classification, and the controller component takes the action of acceleration/deceleration based on the speed sign detected. Formal specifications derived from the system requirements using the FOCETA requirements formalisation tool (cf. Figure 4) are then used for synthesizing DeJaVu runtime monitors for safety property checking. DeJaVu monitors are then seamlessly integrated,

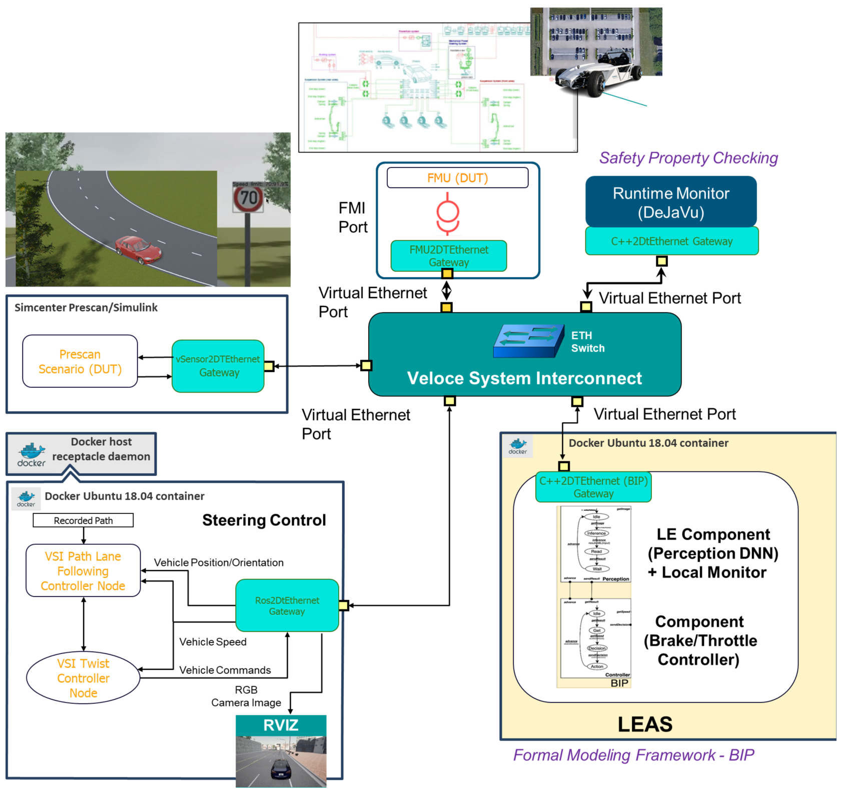
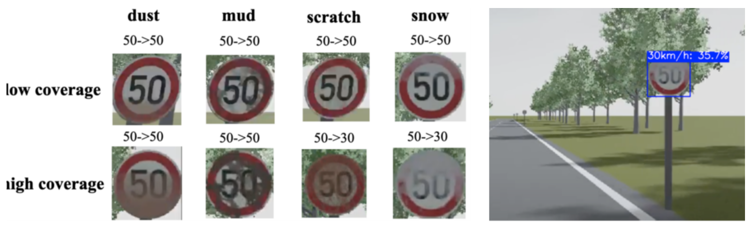

Safety Property Checking
FMU (DUT)
FMI Port
FMU2DTEthernet Gateway
Runtime Monitor (DeJaVu)
C++2DtEthernet Gateway
Virtual Ethernet Port
Virtual Ethernet Port
Simcenter Prescan/Simulink
Prescan Scenario (DUT)
vSensor2DTEthernet Gateway
ETH Switch
Veloce System Interconnect
Virtual Ethernet Port
Virtual Ethernet Port
Docker host receptacle daemon
Docker Ubuntu 18.04 container
Recorded Path
Steering Control
VSI Path Lane Following Controller Node
Vehicle Position/Orientation
Ros2DtEthernet Gateway
Vehicle Speed
VSI Twist Controller Node
Vehicle Commands
RGB Camera Image
RVIZ
Docker Ubuntu 18.04 container
C++2DtEthernet (BIP) Gateway
LE Component (Perception DNN) + Local Monitor
Component (Brake/Throttle Controller)
BIP
LEAS
Formal Modeling Framework - BIP

Fig. 5: ML-based traffic speed detection and Path Lane Following



Property	Param.	Param.	Synth. [s]	#Events	#Violat.	Anal. [s]
Rule 1	-	-	0.401s	4001	**1063**	0.415s
Rule 2	$\epsilon = 15\%$	T=20	1.104s	4001	**42**	0.317s
Rule 2	$\epsilon = 30\%$	T=20	1.149s	4001	**21**	0.301s

Monitoring Results with `DejaVU`

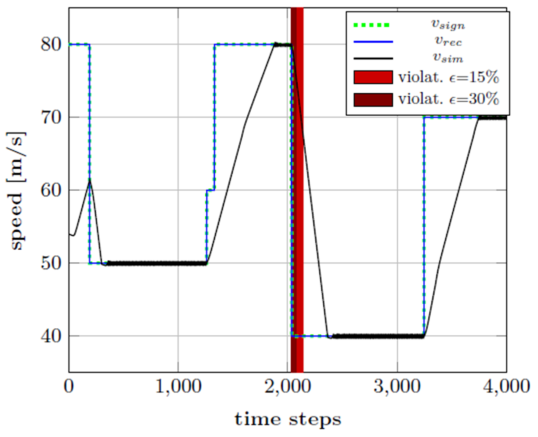


Fig. 6: A trace of the SiL example co-simulated for a horizon of 4001 time units, output: speed over time. The violating events of property/rule 2 are shown in red.

together with the aforementioned components/models, into the compositional digital twin shown in Figure 5 based on the VSI.

The upper part of Figure 6 shows how the classification of the YOLO model used for traffic sign detection is affected via imperfections of varying coverage due to various weather conditions. The tests took place by implementing different scenarios through changing the types of imperfection in Prescan. Our YOLO model appears to be i) robust against low-coverage imperfections, and ii) sensitive against high-coverage scratch and snow. On the right, an incorrect prediction found is displayed, via a bounding box. In the table below, we provide the verification results over the simulation trace of a sample scenario with respect to the following properties:

P_1: "EGO should always identify the traffic sign value correctly and the vehicle's speed should be always smaller or equal to the current speed limit."

P_2:"If EGO detects a new speed limit, the speed should not exceed this speed limit by more than ϵ for T time units after detection, where ϵ is a given percentage.".

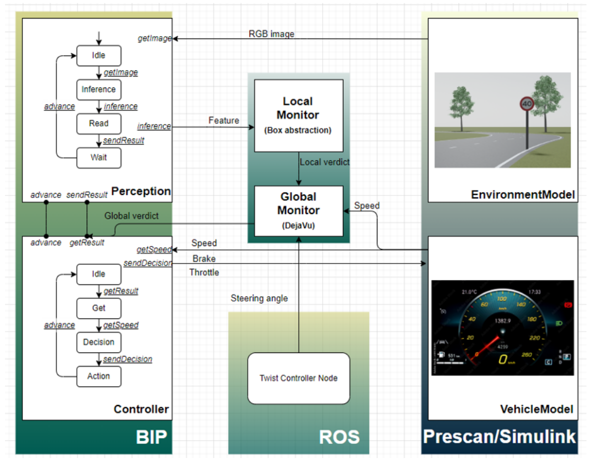


Fig. 7: Digital twin simulation with runtime monitoring the behaviour of a LEC

For simulation traces extended up to 4001 time units we found that property violations occur: (i) when a new traffic sign appears that changes the speed limit from a higher to a lower value, and (ii) due to oscillations of the vehicle speed, when reaching a new speed limit.

The results in Figure 6 are for LEAS that do not account for potential failures of the YOLO model to correctly identify the speed limit. In this case, the combined interaction of the throttle/brake control and steering control modules will have to ensure that when an abrupt steering angle change is commanded, the vehicle's speed is less than a safe margin (constant ζ) relative to the speed limit. Figure 7 showcases the interactions between runtime monitors at two different levels. At a local level for monitoring the performance of the YOLO model, we employ a runtime monitoring technique from those discussed in Section 2.1. At the system level, we use a DeJaVu global monitor for verifying the following property:
P_3:"The traffic sign should be correctly detected (local monitor) and EGO's speed should be smaller than v_{det} (speed limit) if the difference δ_{st} between current and previous steering angle is less than ε_{deg_1} degrees, and v_{sim} should be smaller than v_{det} by at least ζ if δ_{st} is greater than ε_{deg_2}".

4.2 Safe and Secure Intelligent Automated Valet Parking

This case study addresses an automated valet parking (AVP) system, a highly automated driving system in a relatively controlled environment, or technically termed operational design domain (ODD) [35]. Still, in this ODD, there may be mixed traffic that involves pedestrians as well as parked and moving vehicles. In addition, environmental attributes such as illumination, precipitation, and fog conditions may change. For example, Figure 8 shows a functional scenario of pedestrian avoidance in a parking lot and the diversity of the ODD considered in this case study. As seen, even within a relatively controlled parking space, there can be many challenging factors, among them (i) the

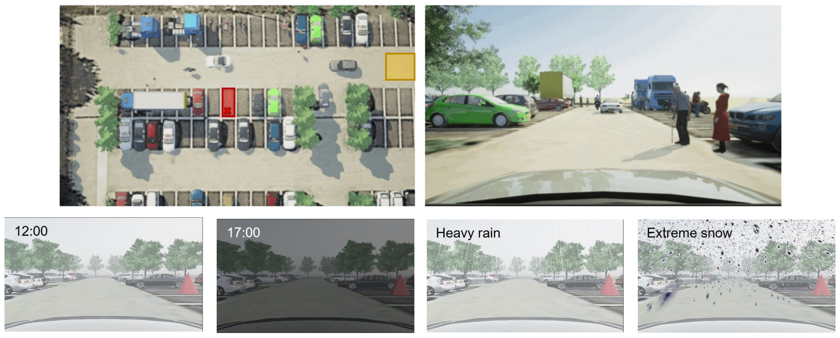


Fig. 8: *(Top)* An example pedestrian-avoiding scenario in the AVP use case. The left image shows a bird's eye view with the drop-off zone in yellow and the designated parking slot in red. The right image shows a driving view, highlighting a pair of pedestrians at risk. *(Bottom)* The effect of various environmental attributes including illumination and precipitation conditions.

high variability of scenarios, (ii) the intricate actor dynamics, and (iii) the constantly changing world (e.g., unknown objects) are the most pressing.

To tackle such challenges and ultimately deliver a trustworthy AVP system, we follow the FOCETA holistic approach and apply multiple techniques presented in the previous sections. Figure 9 gives the overall use case architecture encompassing the baseline system and the applied techniques, which can be grouped into four phases in a continuous engineering process, namely baseline construction with requirement validation, design-time testing and fine-tuning, run-time monitoring and enforcement, and lastly incremental assurance and argumentation. In the following, we instantiate every phase of this workflow with concrete methods and tools and illustrate their application.

(A) Baseline construction with requirement validation. For constructing an AVP baseline system, we consider a component-based design with sensing, planning, and acting components, as shown by the green area in Figure 9. In particular, we implement two LECs, including an NN-based 3D object detector and an RL-based controller. We also focus on virtual testing in the Simcenter Prescan simulator[19]. Overall, this design offers two benefits: (i) The assurance of the overall AVP system can now be attributed to more tractable and efficient verification efforts at the component level; (ii) Adopting the continuous engineering paradigm, we only need to refine and reverify components, typically the learning-enabled ones, that call for an incremental update at some point during testing or run-time.

To illustrate the benefits, we specify a high-level safety goal: The automated vehicle (AV) does not cause a collision unless it is hit by other actors in a static state. With the component-based design, the safety goal can be decomposed into low-level requirements. For instance, the learning-enabled controller is required to always follow a given path within a maximum deviation. Similarly, the (rule-based) emergency brake has to

[19] The virtual simulation platform can be seamlessly changed for HiL or ViL testing.

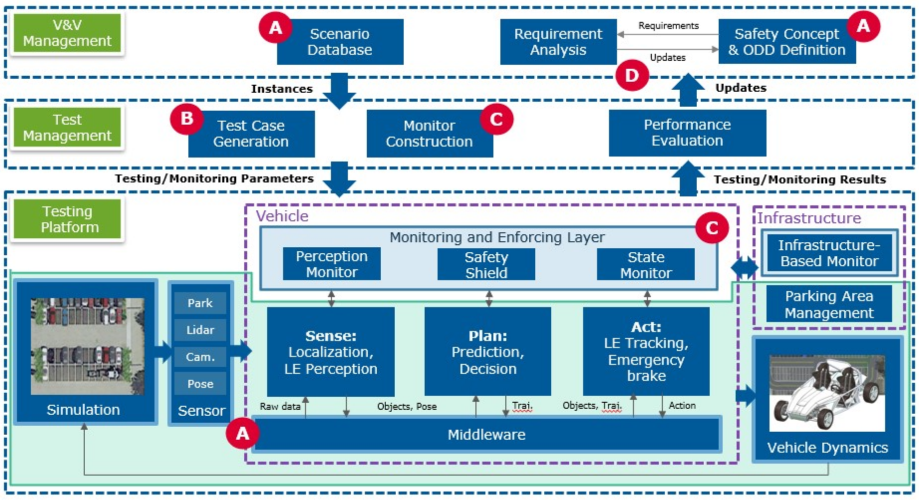


Fig. 9: The overall AVP use case architecture encompassing the testing platform and the V&V and testing techniques discussed in previous sections. The red tags mark the four phases in our continuous engineering process, including (A) baseline construction with requirement validation, (B) design-time testing and fine-tuning, (C) run-time monitoring and enforcement, and (D) incremental assurance and argumentation.

stop the AV whenever there is an obstacle situated within a safety distance (e.g., the pedestrians in Figure 8). Intuitively, this safety distance can be derived from the maximum controller deviation and the maximum braking distance. Finally, the learning-enabled object detector must then always recognize an obstacle (e.g., the pedestrians) within the derived safety distance. We formally specify these requirements to facilitate rigorous V&V, testing, and monitoring of the AVP components/system. These requirements are managed and validated using semantic analysis based on the DSO modeled from the AVP system (see Section 3.1). More specifically, the analysis helps to identify potential weaknesses (e.g., incompleteness, inconsistency, and redundancy) in the requirements.

(B) Design-time testing and fine-tuning. During design time, testing is done at two levels. At the lower level, we focus on testing the two LECs. Due to the space limit, we provide here brief explanations and links to their results. As mentioned in Section 2, we propose three complementary assessment approaches for the 3D object detector: (1) reliability assessment, (2) safety verification, and (3) fault-based testing. The statistical reliability assessment study takes into account the operational profile (i.e., the described ODD) and local robustness of the object detector [44]. The safety verification for the object detector follows an intuitive requirement demanding the predictions properly enclose the ground truths, such that there is an explicit metric and mitigating mechanism for a lower risk of collision. The requirement is formalized using predicate logic with spatial operators on bounding boxes and applied to evaluate the object detector's safety performance. Additionally, the safety performance can be fine-tuned via a safety-oriented loss function [27]. Lastly, the fault injector simulates hardware faults (e.g. bit

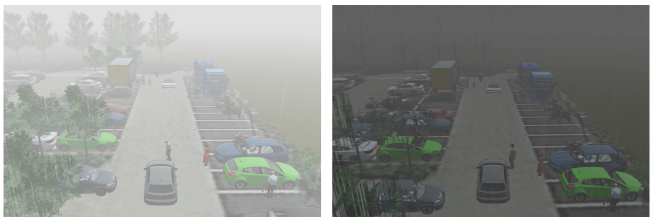


Fig. 10: Critical test cases generated for the AVP system in the pedestrian-avoiding scenario, including a rainy and foggy case *(left)* and a rainy and dark case *(right)*.

flips in NN weights) that allow analyzing their impact on the correctness of the object detector. Likewise, an additional fault mitigating mechanism based on neuron activation interval analysis is developed to safeguard the working of the object detector [15]. As for the learning-enabled controller, we apply RL guiding techniques as described in [1] to ensure its performance and safety.

With the components tested and integrated, system-level testing is then conducted (see Section 3.2). In particular, given the ODD, we employ search-based testing (SBT) with evolutionary algorithms for our AVP system [36]. Specifically, the following steps are taken: (1) Select a *scenario* within the ODD, e.g., the pedestrian-avoiding scenario in Figure 8; (2) Define the variables and specify their ranges in which they will be varied to generate different instances of the scenario such as time of day (e.g., 9:00-17:00), precipitation (e.g., none, medium, or heavy rain), and different pedestrian types (e.g., child, adult, and elder); (3) Define fitness functions based on the introduced safety goal, e.g., the distance between the AV and the pedestrian and the velocity of the AV at its closest point to the pedestrian; (4) Apply a search algorithm and optimize the fitness functions to find failure-revealing test cases. Figure 10 shows two such critical cases qualitatively, and quantitatively we observe about 30% of generated test cases are critical to the AVP system using NSGA-II [11]. By doing such optimization-driven search-based testing, we are able to extensively test the AVP system and mitigate the challenge of high scenario variability.

(C) Run-time monitoring and enforcement. Considering the challenges of intricate actor dynamics and the ever-changing world, an AVP system can hardly be fully assured by design-time testing only. Therefore, monitors and enforcers are usually added to the system to safeguard it during run-time. We highlight two lines of work in this regard. Firstly, we create an out-of-distribution perception monitor (see Section 2.1) and link it to a system safety enforcer. Essentially, the perception monitor provides the confidence level of the object detector's prediction and triggers the system safety shield (see Section 2.2) to correct the underlying controller's action whenever needed [25]. Figure 11 demonstrates an interim result of applying the safety shield (with ground-truth perception). Secondly, a global state monitor modeled (see Section 3.2) with Signal Temporal Logic (STL) is implemented via RTAMT [32] to ensure the overall safety of the AVP system. For example, when the perception monitor sends a warning, the state monitor can check if the shield is actually activated. Additionally, it can also directly monitor different vehicle states such as tracking deviation, as decomposed from the high-level

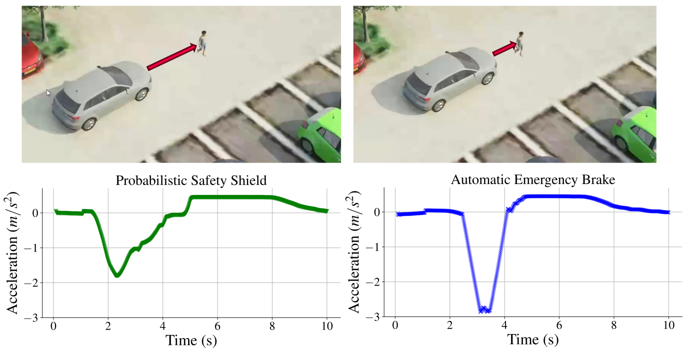


Fig. 11: In the pedestrian-avoiding scenario, the shield *(left)* mitigates the risk of colliding more smoothly and earlier than the baseline emergency brake *(right)*.

safety goal. Altogether, as depicted by the light blue area in Figure 9, these techniques form another layer of safety measures at run-time for the AVP system.

(D) Incremental assurance and argumentation. Finally, to close the loop of the continuous engineering process, we underline the utilization of the specification mining technique (Section 3.3) [3] which helps characterize system behaviors or identify missing requirements with system operations. For instance, by repeating the simulation of the pedestrian-avoiding scenario in Figure 8, we find that the AV (1) will never go within 0.5 meters to the first pedestrian and (2) will only make a collision if the second pedestrian is a child (but not an adult or an elder). With these results, we can then refine or relax the original set of requirements (from phase A) and thereby repair or fine-tune our AVP system for incremental assurance (Section 3.4). Another iteration cycle of requirements validation will then have to take place (cf. Figure 4).

4.3 Anaesthetic Drug Target Control Infusion

This case study develops a concept design and a test platform, depicted in Figure 12, for an intelligent infusion pump controller for Depth of Anaesthesia (DoA). The main aim for a smart infusion controller is to provide support to the anesthesiologist in monitoring the DoA of a patient undergoing surgical intervention in operating room and calculating the next drug infusion dose. The anesthesiologist must take right decisions in possibly stressful environments, which might even include working on more than one patient at the same time. This type of assistance is crucial to ensure that the anesthesiologist can always take the optimal decision and timely react to unforeseen situations.

The main intelligence is embedded in the controller. We developed a data-driven controller based on a recurrent NN for the pump infusion from the collected data used for training. While the data-driven controller may provide near to optimal drug infusion doses, this process is not expected to be fully automated, both from the perspective of regulatory bodies and the user acceptance. Hence, we take the approach of having the *human shield* (anesthesiologist) that ensures the safety of the patient by taking the final decision regarding the next dose. Another LEC in design is the monitor that observes the vital signs of the patient and predicts her current DoA state.

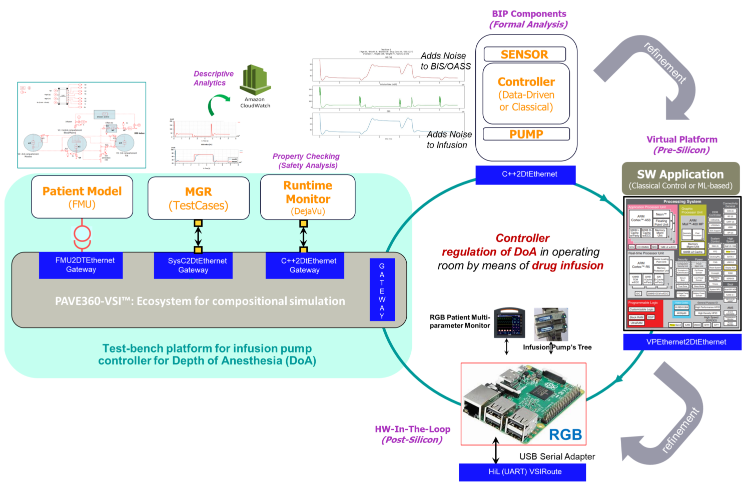


Fig. 12: HiL/SiL platform for the medical case study.

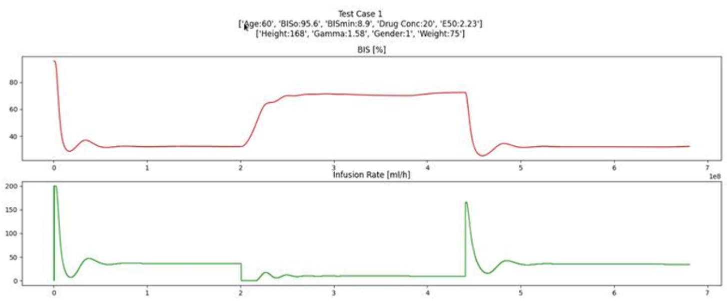


Fig. 13: Simulation results: BIS vs. infusion rate for a given patient Profile TestCase

To support the design of this intelligent drug pump controller, we need several other components and a virtual integration and testing platform. Given that no clinical studies can be done in FOCETA, there is a need for replacing the real patient in the platform. Hence, we developed a patient model in Simcenter Amesim. The model, based on the relevant research in the medical literature, is parameterizable in function of the patient's characteristics, such as the age, weight and gender. The patient is modelled using traditional compartmental analysis in which the interpersonal variability is considered. The patient´s model determines the DoA level achieved with a certain drug dose in bispectral index (BIS) values.

Another key component in this platform is the test case manager. It is used for the initial configuration of the other components according to the selected test case. The main initial settings are the initial drug infusion dose and the target DoA level. The abstract test scenarios are (partially) constructed from interviewing doctors and recording their usual sequences of steps during operations.

We used the FOCETA Compositional Simulation Architecture (cf. Figure 3) to intergrate Simcenter Amesim, DejaVU (Runtime Monitoring), BIP (Component based Modeling), Virtual Platform Modeling, and HiL (external Hardware board), as shown in Figures 12 and 13. Furthermore, we demonstrate the refinement of the DoA controller under test while keeping all other digital twin components intact within the loop (i.e. Testbench platform). The success criterion in this case was to obtain the same results, which affirms the effectiveness of the controller design refinement.

5 Conclusion

In this paper, we presented the FOCETA methodology that consists of two workflows: one for designing trustworthy LECs and one for building safe LEAS by integrating both classical and LE components. We summarized the main steps of the methodology and associated them to the most appropriate methods and tools developed in FOCETA. We finally showed how the case studies in FOCETA demonstrate the various facets of the methodology.

References

1. Edgar A. Aguilar, Luigi Berducci, Axel Brunnbauer, Radu Grosu, and Dejan Nickovic. From STL rulebooks to rewards. *CoRR*, abs/2110.02792, 2021.
2. Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Nickovic. Survey on mining signal temporal logic specifications. *Information and Computation*, page 104957, 2022.
3. Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Ničković. Mining hyperproperties using temporal logics. *ACM Trans. Embed. Comput. Syst.*, 2023.
4. Ananda Basu, Saddek Bensalem, Marius Bozga, Paraskevas Bourgos, and Joseph Sifakis. Rigorous system design: the BIP approach. In *International doctoral workshop on mathematical and engineering methods in computer science*, pages 1–19. Springer, 2011.
5. Saddek Bensalem, Chih-Hong Cheng, Xiaowei Huang, Panagiotis Katsaros, Adam Molin, Dejan Nickovic, and Doron Peled. Formal specification for learning-enabled autonomous systems. In *Software Verification and Formal Methods for ML-Enabled Autonomous Systems*, pages 131–143, Cham, 2022. Springer International Publishing.
6. Luigi Berducci, Edgar A. Aguilar, Dejan Ničković, and Radu Grosu. Hierarchical potential-based reward shaping from task specifications. arXiv, 2021.
7. Yuhang Chen, Chih-Hong Cheng, Jun Yan, and Rongjie Yan. Monitoring object detection abnormalities via data-label and post-algorithm abstractions. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6688–6693, 2021.
8. C. Cheng and R. Yan. Testing autonomous systems with believed equivalence refinement. In *2021 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 49–56. IEEE Computer Society, 2021.
9. Chih-Hong Cheng. Provably-robust runtime monitoring of neuron activation patterns. In *IEEE DATE*, 2021.
10. Simon Cruanes, Grégoire Hamon, Sam Owre, and Natarajan Shankar. Tool integration with the evidential tool bus. In *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, pages 275–294, 2013.

11. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
12. Charis Eleftheriadis, Nikolaos Kekatos, Panagiotis Katsaros, and Stavros Tripakis. On neural network equivalence checking using smt solvers. In Sergiy Bogomolov and David Parker, editors, *Formal Modeling and Analysis of Timed Systems*, pages 237–257, Cham, 2022. Springer International Publishing.
13. Roi Fogler, Itay Cohen, and Doron Peled. Accelerating black box testing with light-weight learning. In *Model Checking Software - 29th International Symposium, SPIN 2023, Paris, France, April 26-27, 2023, Proceedings*, pages 103–120, 2023.
14. Florian Geissler, Syed Qutub, Michael Paulitsch, and Karthik Pattabiraman. A low-cost strategic monitoring approach for scalable and interpretable error detection in deep neural networks. In *Computer Safety, Reliability, and Security - 42nd International Conference, SAFECOMP 2023, Toulouse, France, September 19-22, 2023, Proceedings*, 2023.
15. Florian Geissler, Syed Qutub, Sayanta Roychowdhury, Ali Asgari, Yang Peng, Akash Dhamasia, Ralf Graefe, Karthik Pattabiraman, and Michael Paulitsch. Towards a safety case for hardware fault tolerance in convolutional neural networks using activation range supervision. *CoRR*, abs/2108.07019, 2021.
16. Ralf Gräfe, Qutub Syed Sha, Florian Geissler, and Michael Paulitsch. Large-scale application of fault injection into pytorch models -an extension to pytorchfi for validation efficiency. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, pages 56–62, 2023.
17. Klaus Havelund, Doron Peled, and Dogan Ulus. Dejavu: A monitoring tool for first-order temporal logic. In *3rd Workshop on Monitoring and Testing of Cyber-Physical Systems, MT@CPSWeek 2018, Porto, Portugal, April 10, 2018*, pages 12–13, 2018.
18. Thomas A Henzinger, Anna Lukina, and Christian Schilling. Outside the box: Abstraction-based monitoring of neural networks. In *ECAI 2020*, pages 2433–2440. IOS Press, 2020.
19. Wei Huang, Youcheng Sun, Xingyu Zhao, James Sharp, Wenjie Ruan, Jie Meng, and Xiaowei Huang. Coverage-guided testing for recurrent neural networks. *IEEE Transactions on Reliability*, 71(3):1191–1206, 2022.
20. Wei Huang, Xingyu Zhao, Alec Banks, Victoria Cox, and Xiaowei Huang. Hierarchical distribution-aware testing of deep learning, 2022.
21. Wei Huang, Xingyu Zhao, Gaojie Jin, and Xiaowei Huang. Safari: Versatile and efficient evaluations for robustness of interpretability. *arXiv preprint arXiv:2208.09418*, 2022.
22. Gaojie Jin, Xinping Yi, Wei Huang, Sven Schewe, and Xiaowei Huang. Enhancing adversarial training with second-order statistics of weights. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15273–15283, June 2022.
23. Gaojie Jin, Xinping Yi, Liang Zhang, Lijun Zhang, Sven Schewe, and Xiaowei Huang. How does weight correlation affect generalisation ability of deep neural networks? In *Advances in Neural Information Processing Systems*, volume 33, pages 21346–21356, 2020.
24. Neslihan Kose, Ranganath Krishnan, Akash Dhamasia, Omesh Tickoo, and Michael Paulitsch. Reliable multimodal trajectory prediction via error aligned uncertainty optimization. In *Computer Vision - ECCV 2022 Workshops - Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part V*, pages 443–458, 2022.
25. Bettina Könighofer, Julian Rudolf, Alexander Palmisano, Martin Tappler, and Roderick Bloem. Online shielding for reinforcement learning. *CoRR*, abs/2212.01861, 2022.
26. Brian Hsuan-Cheng Liao, Chih-Hong Cheng, Hasan Esen, and Alois Knoll. Are transformers more robust? Towards exact robustness verification for transformers. In *SafeComp*, 2023.
27. Brian Hsuan-Cheng Liao, Chih-Hong Cheng, Hasan Esen, and Alois Knoll. Improving the safety of 3D object detectors in autonomous driving using IoGT and distance measures. abs/2209.10368, 2023.

28. Benedikt Maderbacher, Stefan Schupp, Ezio Bartocci, Roderick Bloem, Dejan Nickovic, and Bettina Könighofer. Provable correct and adaptive simplex architecture for bounded-liveness properties. In *Model Checking Software - 29th International Symposium, SPIN 2023, Paris, France, April 26-27, 2023, Proceedings*, pages 141–160, 2023.
29. Abdulrahman Mahmoud, Neeraj Aggarwal, Alex Nobbe, Jose Rodrigo Sanchez Vicarte, Sarita V. Adve, Christopher W. Fletcher, Iuri Frosio, and Siva Kumar Sastry Hari. Pytorchfi: A runtime perturbation tool for dnns. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 25–31, 2020.
30. Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
31. Konstantinos Mokos, Theodoros Nestoridis, Panagiotis Katsaros, and Nick Bassiliades. Semantic modeling and analysis of natural language system requirements. *IEEE Access*, 10:84094–84119, 2022.
32. Dejan Nickovic and Tomoya Yamaguchi. RTAMT: online robustness monitors from STL. In *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings*, pages 564–571, 2020.
33. OpenAI. Gpt-4 technical report, 2023.
34. Syed Qutub, Florian Geissler, Yang Peng, Ralf Gräfe, Michael Paulitsch, Gereon Hinz, and Alois Knoll. Hardware faults that matter: Understanding and estimating the safety impact of hardware faults on object detection DNNs. In *Lecture Notes in Computer Science*, pages 298–318. Springer International Publishing, 2022.
35. SAE. J3016: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.
36. Lev Sorokin, Tiziano Munaro, Damir Safin, Brian Hsuan-Cheng Liao, and Adam Molin. Opensbt: A modular framework for search-based testing of automated driving systems, 2023.
37. Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, oct 2019.
38. Anastasios Temperekidis, Nikolaos Kekatos, and Panagiotis Katsaros. Runtime verification for fmi-based co-simulation. In Thao Dang and Volker Stolz, editors, *Runtime Verification*, pages 304–313, Cham, 2022. Springer International Publishing.
39. Anastasios Temperekidis, Nikolaos Kekatos, Panagiotis Katsaros, Weicheng He, Saddek Bensalem, Hisham AbdElSabour, Mohamed AbdElSalam, and Ashraf Salem. Towards a digital twin architecture with formal analysis capabilities for learning-enabled autonomous systems. In *Modelling and Simulation for Autonomous Systems*, pages 163–181, Cham, 2023. Springer International Publishing.
40. Stavros Tripakis. Bridging the semantic gap between heterogeneous modeling formalisms and FMI. In *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 60–69. IEEE, 2015.
41. Changshun Wu, Yliès Falcone, and Saddek Bensalem. Customizable reference runtime monitoring of neural networks using resolution boxes, 2021.
42. Zuxuan Wu, Ser-Nam Lim, Larry Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors, 2019.
43. Peipei Xu, Fu Wang, Wenjie Ruan, Chi Zhang, and Xiaowei Huang. Sora: Scalable black-box reachability analyser on neural networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
44. Xingyu Zhao, Wei Huang, Alec Banks, Victoria Cox, David Flynn, Sven Schewe, and Xiaowei Huang. Assessing the reliability of deep learning classifiers through robustness evaluation and operational profiles. In *Proc. of the Workshop on Artificial Intelligence Safety 2021 (co-located with IJCAI 2021)*, volume 2916 of *CEUR Workshop Proceedings*, 2021.

45. Xingyu Zhao, Wei Huang, Vibhav Bharti, Yi Dong, Victoria Cox, Alec Banks, Sen Wang, Sven Schewe, and Xiaowei Huang. Reliability assessment and safety arguments for machine learning components in assuring learning-enabled autonomous systems. *ACM Transactions on Embedded Computing Systems*, 2022.
46. Xingyu Zhao, Wei Huang, Xiaowei Huang, Valentin Robu, and David Flynn. Baylime: Bayesian local interpretable model-agnostic explanations. In *Proc. of 37th Conference on Uncertainty in Artificial Intelligence*, volume 161, pages 887–896. PMLR, 27–30 Jul 2021.