A user-centric computer-aided verification process in a virtuality-reality continuum

Victor Romero, Romain Pinquié, Frédéric Noël

Version of Record: https://www.sciencedirect.com/science/article/pii/S0166361522000756
Manuscript_186a772bead21b156ef4f2199cf163a0

A User-Centric Computer-Aided Verification Process in a Virtuality-Reality Continuum

Victor Romero[1], Romain Pinquié[1], Frédéric Noel[1]

[1]Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, Grenoble, France

Abstract. Although companies systematically strive for a full digitalisation of their products and their processes, the design phase shows that the quality of models is very unequal. Indeed, detailed design benefits from much more sophisticated methods and tools than the specification and architecture activities. Although, we should note the recent paradigm shift from document-based to model-based systems engineering, these models, which are mainly static 2D diagrams, remain poor to facilitate design verification early on. Thus, to detect most errors during the design phase, companies have no other alternative than to wait up to the testing phase which occurs after several years of development for complex systems. Thus, we propose a user-centric computer-aided verification process to ensure that the design meets the requirements under realistic operational conditions. The verification process provides a progressive immersion into the virtual system before seamlessly transitioning to the real system. Our work is built upon state-of-the-art MBSE methods such as the Property Model Methodology, which enables systems engineers to co-simulate specification models and design models. We improve such MBSE methods by increasing the level of realism that experiences the end-user during the verification of a design by the original combination of Model-In-the-Loop, Immersive Model-In-the-Loop, Human-In-the-Loop, and Hardware-In-the-Loop simulation strategies. A robot arm is used as a use case to illustrate the verification process.

Keywords: Model-Based Systems Engineering; Validation; Verification; User-Centered Design; Virtual Reality; Human-in-the-Loop Simulation; Immersion; Realism.

1. Introduction

1.1. Context

In the past decade, Model-Based Systems Engineering (MBSE) replaced the traditional document-centric systems engineering approach for the design of large engineered systems. In its systems engineering vision 2020 [1], the International Council On Systems Engineering (INCOSE) defined Model-Based Systems Engineering (MBSE) as *"the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases*". To date, we perceive the motivation for engineering systems using models through two different lenses.

On the one hand, some practitioners argue that the use of models to support an engineering activity is an age-old solution since models have been used for decades. Indeed, the roots of MBSE can be traced back to the pioneering work achieved by Wayne Wymore. In his book entitled "*Model-Based Systems Engineering*" [2], Wymore has shown the crucial role of mathematics in the specification and design of systems. By theorising systems engineering concepts, he paved the way for MBSE, but his visionary mathematical theory of systems engineering was so far ahead of systems engineers that it took several decades before his contribution was recognized [3].

On the other hand, the model-based approach is in the limelight today because there is another community of engineers who see MBSE as the salvation of systems engineering. The intellectual Renaissance of MBSE results from academic researchers and industrialists who have worked out a set

This paper is an extended version of the paper entitled "A Computer-Aided Verification Process for Engineered Systems" presented by Romero, V., Pinquié, R., and Noel, F. in May 2021 at the conference CIGI Qualita21, Grenoble, France.

of theoretical and methodological concepts related to systems engineering, a branch that originally was more an irrational practice rather than a rigorous method [4].
MBSE rests on a four-activity process: specification, validation, design, and verification [5, Sec. 9.2.1]. Our study focuses on verification. It is important not to confuse verification that intends to ensure that the "product is built right" and validation that intends to ensure that the "right product is built" [5].

According to the INCOSE, there exists four verification techniques: inspection, demonstration, testing, and analysis. Here we concentrate on the analysis.

> *"Analysis – This technique is based on analytical evidence obtained without any intervention on the submitted element, using mathematical or probabilistic calculation, logical reasoning (including the theory of predicates), modelling, and/or simulation under defined conditions to show theoretical compliance. Mainly used where testing to realistic conditions cannot be achieved or is not cost-effective."*

The recent development of cutting-edge computational capabilities has opened new vistas for the analytical verification of design solutions [6]. Carried out early in the design process, analytical verification using executable models enables companies to drastically reduce costs and time-to-market while improving quality [7], [8].
Nevertheless, when we observe the verification activity in the industry, we notice that there are several – more or less advanced – practices including, but not limited to, the sharing of 2D captures of models, dynamic co-simulation of specification and design models, immersive 3D design reviews, hardware-in-the-loop simulation, physical testing, etc., but there is no computer-aided verification process that logically articulates them.

1.2. Problem(s)

The Document-Based System Engineering (DBSE) approach is reaching its limit when we deal with large engineered systems. Indeed, the content of documents is difficult to maintain, synchronise, and access [5]. When looking for design errors during the Validation & Verification (V&V) activity, engineers do not have other alternatives than to get together for a pragmatic design review where they share screenshots of their models (SysML, CAD, physics-based simulation, testing reports, etc.) supporting a relatively subjective argumentation process that aims at justifying the satisfaction of requirements stored in large and ambiguous document-based specifications.

To make the verification activity more objective, recent MBSE methods such as the Property Model Methodology [7], [8] and FORM-L [9] propose to verify whether the design meets the requirements by the co-simulation of design models and specification models. In these approaches, requirements are not natural language statements but logical assertions that monitor the state of design variables in near real-time. Although these new MBSE methods and tools are essentials for the early detection of design errors, the lack of visualisation and interactions of MBSE virtual environments do not enable stakeholders to experience the virtual system in realistic conditions. Figure 1, which is borrowed from state-of-the-art research results in MBSE where simulation capabilities support the V&V activities, is a virtual cockpit used at Airbus Helicopters for the early validation of model-based specifications and verification of model-based designs. During an R&D review, a test pilot is invited to sit in front of the virtual cockpit of a helicopter to perform the operational scenarios. While performing the operational scenarios in the front end, specification models are factually validated by a simulation engine that simulates the set of semi-formal requirements in the back end [7], [8], [10]. However, Figure 1 illustrates that current MBSE tools offer a poor level of realism with a two-dimensional operational view that enables the end-user – i.e., a pilot – to operate the system-of-interest – i.e., a helicopter. To sum up, such simulation-based V&V increases the behavioural realism, that is, the fidelity of the virtual world with the laws of physics, but the 2D panel control shows a poor level of appearance realism or immersion [11], that is, the

degree of resemblance of the virtual world and the real world. Furthermore, the lack of appearance realism may decrease the psychological realism or presence [12], that is, the pilot who experiences the system knows that he is experiencing a virtual world without behaving as he was in the real world.

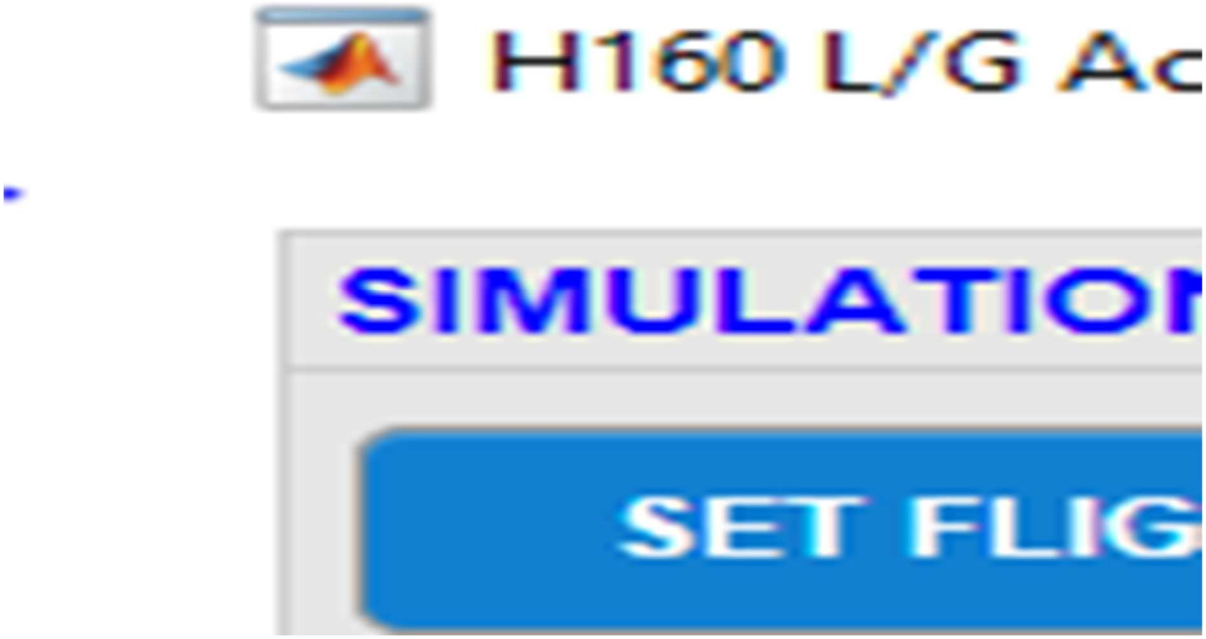

Figure 1: Virtual cockpit used for simulation-based validation and verification with the MathWorks suite

There are numerous ways to experience a virtual system in realistic conditions, but there is no integrated and continuous verification strategy. For instance, to increase the appearance realism or immersion, we can immerse stakeholders in an immersive environment using Virtual Reality (VR) [13]. However, the isolation of the specification and the design also leads to a subjective design review since no formal specification models are monitoring the design models. Indeed, even in an immersive virtual environment, the designers would look for potential design errors without a reliable strategy since requirements are stored in external ambiguous documents. This scenario would only lead to classic design reviews with structural properties represented with a better level of appearance realism. Moreover, in practice, the integration of the multi-engineering physics-based simulations used to faithfully mimic the laws of physics in systems engineering with virtual reality remains exceptional, which lower the level of behavioural realism. Regarding the behavioural properties of a system, we can replace models of sub-systems with real components and carry out Hardware-In-the-Loop simulations to get rid of some modelling assumptions and therefore be more realistic [14], but the specifications remain isolated from the design too.

Research Question:

How to integrate and articulate state-of-the-art modelling and simulation capabilities (Model-, Immersive Model-, Human-, and Hardware-In-the-Loop) in a computer-aided process to support the verification of engineered systems?

1.3. Proposal

The proposal aims to support the systems engineering community, especially the architects who practice the model-based systems engineering approach for specifying requirements, decomposing the system into subsystems, managing interfaces, and conducting a systematic top-down integration and verification. To connect the dots, we propose to logically integrate modelling and simulation capabilities in a computer-aided verification process. The model-centric integration strategy relies upon a virtuality-reality continuum for continuously verifying that the design satisfies the requirements with objective evidence in realistic conditions from the system architecture to the first operational prototype.

Note that the integration of advanced modelling and simulation capabilities is not forced but motivated by sound arguments:

- Modelling of semi-formal requirements leads to complete and correct specifications thanks to simulation capabilities.
- Integration of specification and design models in a unique environment leads to a tightly integrated simulation-based validation and verification.
- Virtual reality increases the level of realism of structural properties thanks to immersion and a higher level of presence since the human-in-the-loop simulation of the system in near-real-time enables the end-users to inject operational randomness.
- The virtuality-reality continuum supported by hardware-in-the-loop simulation progressively replace modelling assumptions with real structural and behavioural system properties.

Research Hypothesis

We assume that state-of-the-art modelling and simulation capabilities can be continuously integrated and articulated in a user-centric computer-aided process that supports the verification under realistic operational conditions in a virtuality-reality continuum.

This proposal is limited to dynamic engineered systems and especially for the functional properties although non-functional ones could be considered without guaranteeing an added value. Finally, it is not assured that the process and the technological implementation choices, especially monitoring techniques, would work for system-of-systems.

2. Literature review

Before detailing our contribution, we will review the recent advances in model-based systems engineering in general and model-based verification in particular.

2.1. Model-Based Systems Engineering

The Renaissance of MBSE can be attributed to the adoption of the Systems Modeling Language (SysML) [15], [16] by the Object Management Group (OMG) in July 2006. SysML results from the creation of a Unified Modeling Language (UML) profile tailored to systems engineering. The creation of this new graphical modelling language led to the need for an MBSE method, that is, a systematic logically ordered sequence of steps to be followed by systems engineers to specify, validate, design, and verify a system. Numerous SysML-based MBSE methods have been proposed. IBM developed the Harmony for Systems Engineering (Harmony-SE) method [17]. Though Harmony-SE was created as tool-neutral, today it is supported by the IBM Rhapsody software. In collaboration with Lockheed Martin, the INCOSE worked out the Object-Oriented Systems Engineering Method (OOSEM) [18]. OOSEM is consistent with the “V” process. It also relies on SysML and can be implemented with any SysML environment. The SysML-based method developed by Mhenni et *al.* [19] harnesses the fundamental engineering concepts of “black box” and “white box” analyses that enable engineers to

distinguish the “what” from the “how”, that is, the specification domain from the design domain. This method provides MBSE practitioners with a recipe recommending which SysML diagrams use, in which order, and for which intent. The Systems Modeling (SYSMOD) method proposed by Weilkiens [20] also relies on SysML and can be implemented with several SysML editors including the MagicDraw SysML plugin or the Eclipse Process Framework. Many companies that were not satisfied with the existing SysML-based methods have defined their own SysML profiles that meet their domain-specific needs. For instance, Alstom and Valeo created the Advance System Architect Program (ASAP) [21] and SysCARS [22] methods, respectively.

The Rational Unified Process (RUP) for Systems Engineering (RUP SE) [23], which is an MBSE method derived from the well-known software development RUP method [17], is an alternative to the SysML-based methods previously introduced. As reported by Estefan [24], “*RUP SE brings the RUP style of concurrent design and iterative development to systems engineering*”. There are no major improvements in RUP SE compared to RUP. Instead, RUP has been tailored to systems engineering concerns, and a RUP SE plugin for the Method Composer (RMC) product is distributed by IBM Rational.

Experiments conducted inside Thales have shown that systems engineers who do not have a computer science background do not get along with the object-oriented concepts of UML and hence SysML [25]. Therefore, Thales worked out Arcadia [26], an MBSE method that relies on its own Domain Specific Language supported by the open-source visual modelling software Capella.

Based on the previous review, systems engineers, point out two major disadvantages for software-native MBSE methods:

[Software genesis] Above all, one crucial observation is that authors cobbled these MBSE methods together from the fundamental concepts of the software development domain, especially the object-oriented paradigm, to cope with systems engineering issues. SysML-based methods were for instance developed based on the assumption that a tailored UML profile would suit the needs of systems engineers, whereas several studies [25], [27], [28] have pointed out the limits of SysML. The idea of formatting SysML according to the “black box” and “white box” viewpoints is relevant but it is a palliative solution to inject some fundamental systems engineering principles into a language that originates from software engineering. The influence of software engineering on SysML does not only adversely affect MBSE methods but also the tools which inherit numerous features from the UML world that are irrelevant, not to say confusing for systems engineering [27].

[Standardised notations as a means of communication without computational capabilities] Most systems modelling language – e.g. SysML, UML, or Object-Process Language (OPL) [29] – do not have a grammar. Therefore, they are not languages but standardised notations. The graphical models or diagrams created with such standardised notations are foremost a pragmatic means of communication among stakeholders [27]. We use the adjective pragmatic because the interpretation of such models requires implicit knowledge from an external interpreter. In other words, not all information required to interpret such a model is explicitly defined in the model. Indeed, the interpretation of a diagram relies on domain-specific knowledge shared by the stakeholders. Compared to a document-centric approach, when practitioners are trained to use a modelling method and a modelling tool, it is no surprise that SysML improves the consistency and readability of a system as a whole [28]. However, SysML diagrams do not serve to predict the evolution of behavioural or structural properties of an engineered system. To do so, a few researchers took the initiative to couple SysML with a simulation language. For instance, Paredis *et al.* [30] worked out SysML4Modelica to perform bi-directional transformations between SysML and Modelica. However, such attempts remain very limited to some SysML diagrams, especially the parametric diagram, and their implementation are not integrated with a sound systems engineering method yet. Thus, to date, MBSE artefacts is a set of static descriptive semiotic systems often called *graphical models* or *diagrams* created with standardised notations so as to ease the communication among stakeholders

without providing any simulation capabilities. For instance, SysML requirements are textual statements linked between each other via pre-defined relationships (satisfy, derive, trace, refine, etc.) that neither provide computational capabilities to validate a specification model or verify a design model against a valid specification model.

So far, we have reviewed the MBSE methods built on top of fundamental software engineering principles. Various alternatives relying upon fundamental systems engineering principles have been proposed. Vitech Corporation defined its own MBSE method that is mapped onto an iterative and recursive top-down 4-activity systems engineering process: requirements analysis, functional analysis, synthesis, design validation and verification [31]. Though it is not explicitly stated, this MBSE method is strongly dependent on the language and CORE tool provided by Vitech Corporation. Moreover, the models developed are inert diagrams like the SysML and UML-based ones. The NASA Jet Propulsion Laboratory (JPL) also came up with its own MBSE method: State Analysis [32]. State Analysis combines states – i.e., representations of the momentary condition of an evolving system – and models – i.e., representations of how states evolve – in a common framework. The concept of state is deeply rooted in the control theory. Piaszscyk [33] proposed an MBSE method relying on the Department of Defense Architectural Framework (DoDAF). In the method, information is sporadically redundant, especially between the architecture (OV-2) and activity sequence (OV-6c) diagrams. Moreover, the author proposes to automate several operations such as the derivation of functional requirements from the functional architecture, but there is no empirical validation of these propositions. We strongly believe that the specification of the performance domain and the activation conditions of a requirement can hardly be automated because they require domain-specific knowledge that only resides in the head of sufficiently educated individuals. Finally, the author states that the proposed MBSE method can be implemented with various modelling languages such as SysML, IDEF0, or BPMN, but there is no guarantee that a single language can support all the required concepts. Dori [29] proposed the Object-Process Methodology (OPM) that involves three language constructs: object, process, and state. OPM models are bi-represented as diagrams and text that complies with the Object-Process Language (OPL) which is actually not a language but a standardised notation. OPM, which is supported by the OPCAT Software Solutions, was recently standardised by the International Standards Organisation (ISO) as ISO 19450 [34]. OPCAT also integrates a software solution to generate SysML diagrams from OPM. To date, OPM is tightly integrated with SysML [35]. The *a-posteriori* need to tightly integrate OPM with SysML is a proof showing that both standardised notations were initially incomplete to cover the modelling of the core systems engineering activities, that is, specification, design, and V&V. The Whole-of System Analytical Framework (WSAF) MBSE method introduced by Robinson et al. [36] which is based on the DAF guidance, is dedicated to the defence area. The WSAF implemented with the CORE tool has very limited simulation capabilities that do not enable engineers to validate specifications and verify designs early on. The software editor Dassault Systèmes offers the RFLP (Requirements, Functional, Logical, Physical) framework that is supported by the Catia Systems software solution [37] without any explicit modelling method.

Although readers may notice that there exist various MBSE methods, none of them covers the four core activities (specification, validation, design, and verification) of the systems engineering development process with the capability to validate specifications and verify designs early in the process thanks to simulation capabilities. Furthermore, each method is closely linked to a modelling language and a modelling tool. This feature is not in harmony with the heterogeneous environment within which each discipline uses the language and tool that best suit their domain-specific needs.

2.2. Model-Based Verification

According to the main systems engineering standards: "*The purpose of the verification process is to provide objective evidence that a system or a system element fulfils its specified requirements and*

characteristics" (ISO/IEC/IEEE 15288, [6.4.9.1]) [5]. Except for analysis, all the verification types require a physical prototype that is excessively expensive for engineered systems. The starting point of the verification activity is the formalisation of the requirements to objectively compare the actual system properties with what is expected. For this purpose, standardised graphical annotations such as SysML [16], [38] and Capella [25] help to develop and manage the requirements and designs without natively providing any formal verification capabilities. However, this section will discuss recent studies that focus on the use of modelling and simulation software to formally model computable requirements that fulfil a monitoring function. We will see that a tight integration of specification models and design models in a co-simulation environment helps to identify design errors early on.

As Suh [4] reports "*Often designers find that the precise description of "what we want to achieve" is a difficult task. Many designers deliberately leave them implicit rather than explicit and then, start working on design solutions even before they have clearly defined their design goals. They measure their success by comparing their design with the implicit design goals they had in mind. They spend a great deal of time to improve and iterate the design until the design solution and "what they had in mind" converge, which is a time-consuming process at best.*" [4]. Today, the most common verification approach consists of modelling and simulating the properties of the system and organising design reviews to approve or reject a design candidate based on the results of the simulations without a rigorous definition of the requirements [39] and their integration with design solutions.

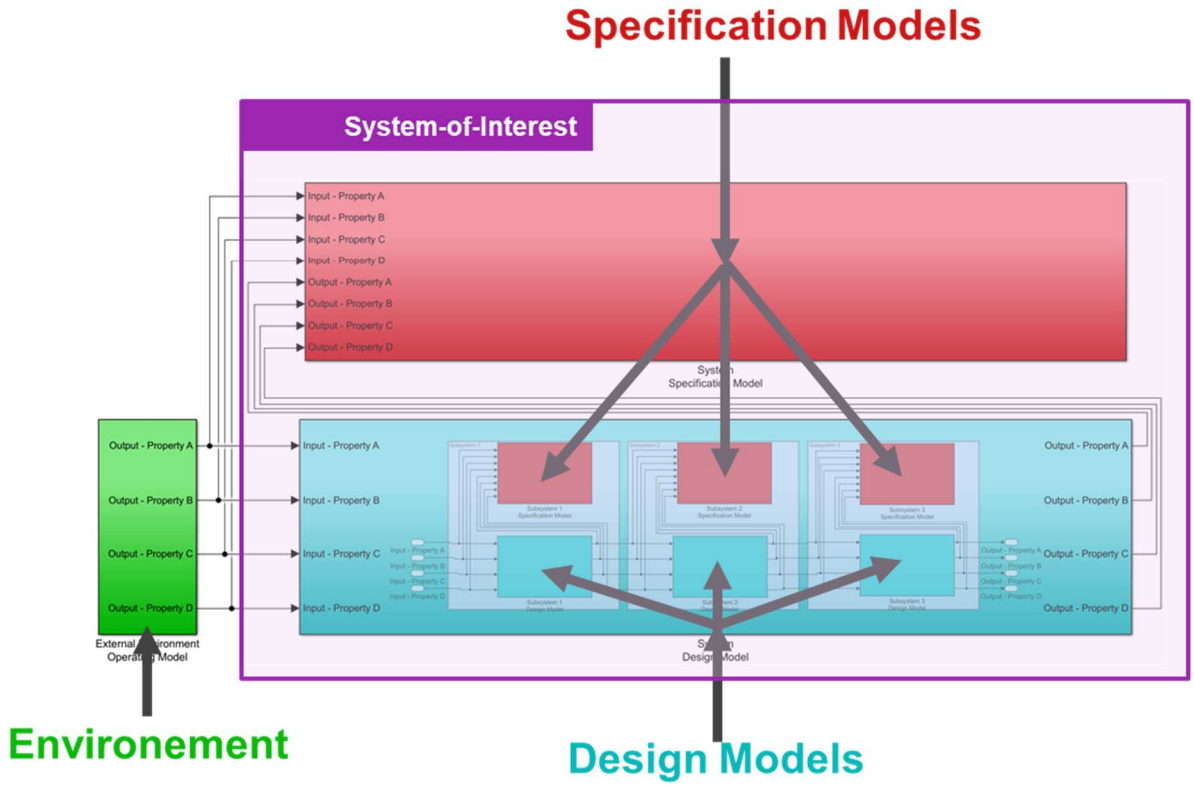

Figure 2: With the Property Model Methodology, the model- and simulation-based verification activity requires: the top-down modelling of (1) the system specification model (block in red), (2) the system design model (block in blue), (3) the sub-systems specification models (blocks in red within the system design model), (4) the design of the sub-systems models (block in blue within the system design model), (5) the operating environment and scenarios that stimulates the specification and design models (block in green).

To satisfy Suh's recommendation "*To be efficient and to generate the design that meets the perceived needs, we must specifically state the design goals in terms of 'what we want to achieve' and begin the design process."* [4]*,* recent MBSE methods and tools propose formal specification and verification activities [39], [40]. For instance, Micouin proposed the Property Model Methodology (PMM) [41], which arises from fundamental systems engineering concepts such as a top-down recursive engineering process, formally defined requirements using the concept of Property-Based Requirements (PBR) [42], zig-zagging from the specification domain and the design domain [4], and industrial standards compliance [43]. PMM is by definition language- and tool-agnostic as academic and industrial case studies have been implemented with various modelling and simulation languages and tools including VHDL-AMS [41], Modelica [44] with the support of packages developed to create requirements models [9], [45], [46], or Matlab-Simulink-Stateflow [40]. Nevertheless, one should be aware that the quality of the models and simulations depends on the capabilities and maturity of the language and the tool preferred for the implementation. PMM was recently successfully applied to an operational Airbus Helicopter program [7], [8]. Figure 2 shows that in the PMM method, the verification relies upon a top-down recursive co-simulation of specification models and design models stimulated by a model of the operational environment and scenarios.

The limits of the model- and simulation-based verification approaches broached out during the introduction are:

- **[Limit 1]** the weak **visualisation** capabilities of the physics-based simulation tools that support the method, and which lead to a poor appreciation of the structural properties, especially for the end-user operational perspective.
- **[Limit 2]** the weak **interaction** capabilities of the physics-based simulation tools that support the method, and which prevent the end-user to experience the random evolution of structural and behavioural system properties resulting from human-in-the-loop unintended operational scenarios that are never captured in traditional test cases.

Virtual Reality (VR) technologies, which support the 3D representation of a system, gives the end-user a realistic experience [47]. This encourages companies to adopt VR for various applications including the verification activity [48], [49] that is limited to the inspection-like verification. Indeed, immersive environments increase the end-user experience when analysing structural properties, but they hardly integrate a physics-based simulation for emulating the behavioural properties and do not integrate formal specification models. Therefore, the end-user's experience remains unrealistic regarding the dynamic of the system and the design remains formally separated from the specification. Finally, all these analytical approaches that support the early verification of systems include modelling and simulation assumptions that could be progressively replaced by hardware-in-the-loop simulation.

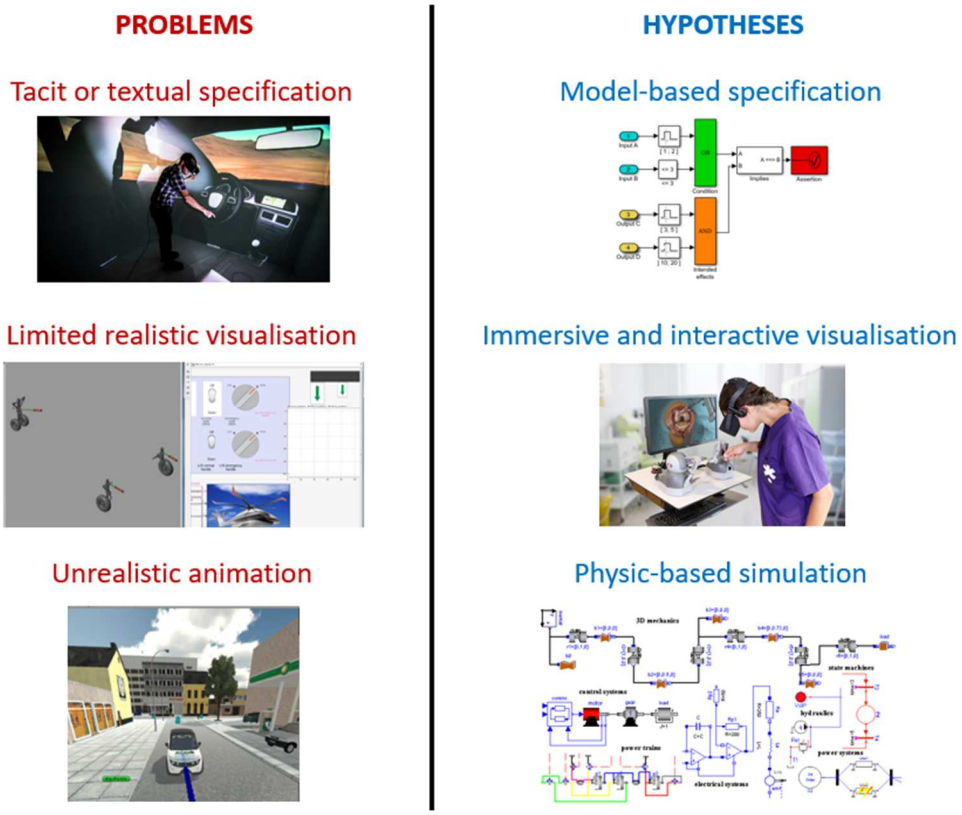

Figure 3: The empty space between visualisation realism and simulation realism

The literature review shows that several recent researches proposed various advanced methods and tools for verifying designs early on, but that there is a lack of continuous integration (Figure 3). Therefore, in the next section, we propose a user-centric computer-aided verification process in a virtuality-reality continuum that supports the verification of engineered systems under realistic operational conditions.

3. A Virtuality-Reality Continuum for the V&V of Engineered Systems

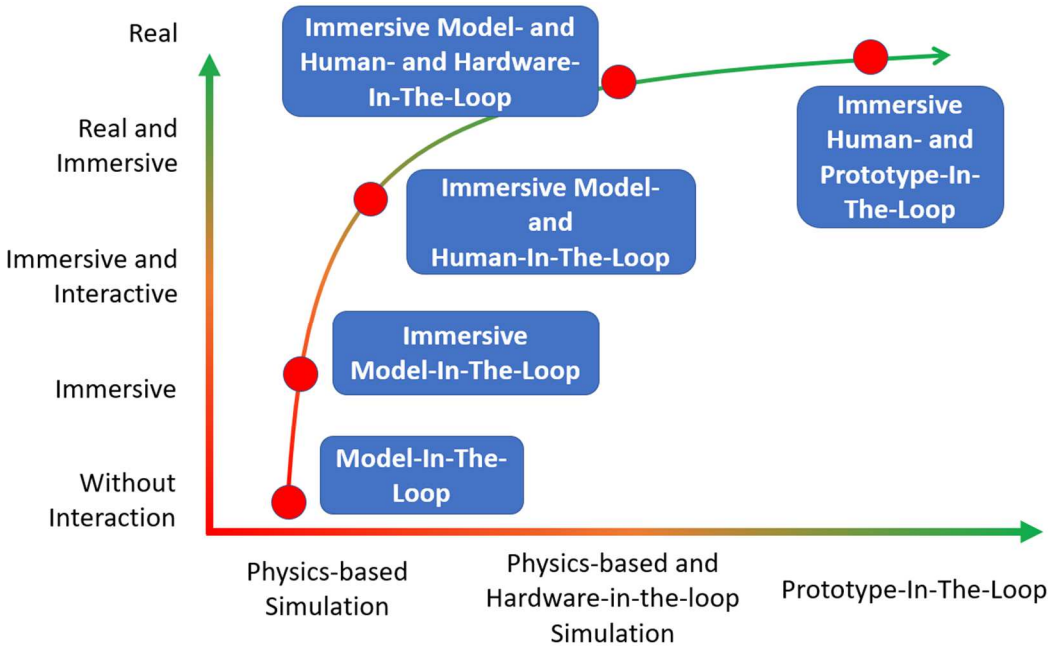

Figure 4: User-centric computer-aided verification process in a virtuality-reality continuum

We assume that an efficient computer-aided verification process (Figure 4), which provides increasing analytical evidence in realistic conditions, starts with a state-of-the-art analytical verification activity, that is, the co-simulation of design models with specification models – **Model-In-the-Loop Verification**. The second activity of the computer-aided verification process consists of immersing the end-user in a virtual world to improve his experience in terms of visualisation – **Immersive Model-In-the-Loop Verification**. Always with the desire to improve the degree of realism, especially the interactions with the virtual world, the third activity integrates the human to realistically stimulate the models in near real-time with some randomness that may occur in operational scenarios – **Immersive Model- and Human-In-the-Loop Verification**. Finally, the fourth activity removes modelling and simulation assumptions by a continuous substitution of virtual building blocks from the design model by real ones up to the first real prototype of the system - **Immersive Model-, Human- and Hardware-In-the-Loop Verification**.

Figure 5 shows the model-based design workflow that temporally sequences the simulation capabilities required to implement the user-centric computer-aided verification process in a virtuality-reality continuum. As illustrated, the implementation requires one or several machines to run the physics-based simulation and the virtual reality scene with data exchange interfaces that will be specified later on. At the beginning of the process, the system-of-interest is fully virtual and its visuals and interactions are desktop-based before being replaced by immersive metaphors for improving the level of realism. Later in the process, virtual components are progressively replaced by real ones without code generation at first and finally with the code deployed on hardware prototypes. Each verification activity will be further detailed in the following sections and illustrated with a robot arm case study in section 4.

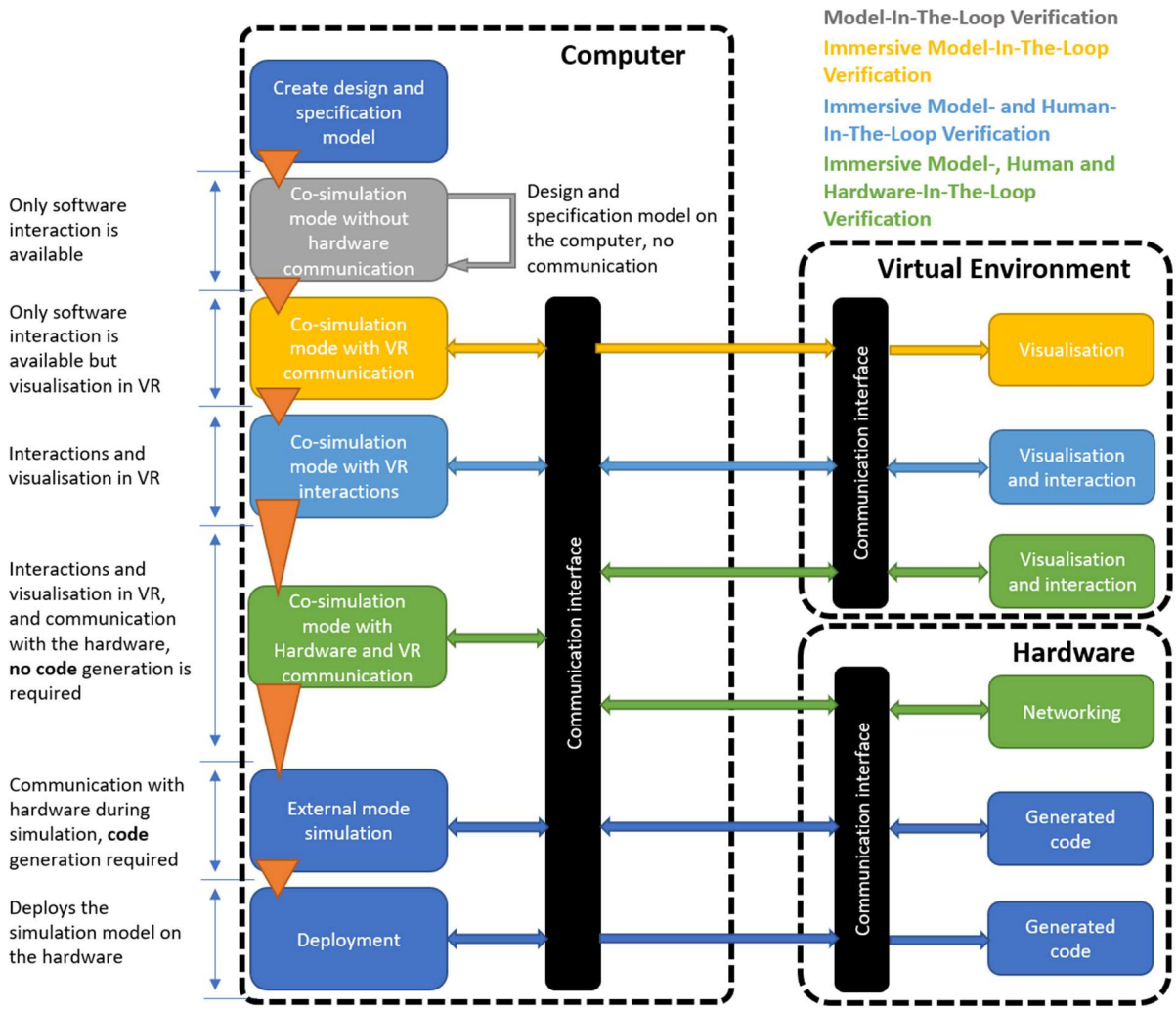

Figure 5: The computer-aided verification process included in a model-based design workflow. This is not a detailed software architecture, but a logical architecture where activities of the verification process are allocated to simulation capabilities requiring hardware and software materials. The horizontal arrows between the simulation capabilities and the communication interfaces stand for conceptual data interfaces which will be detailed for each verification activity.

3.1. Model-In-The-Loop Verification

The process starts with the state-of-the-art MBSE method Property Model Methodology [41] for its **Model-In-the-Loop verification** strategy implemented with the MathWorks suite [7], [8] (Figure 6). Model-In-the-Loop verification consists of the co-simulation of formal specification models with discrete and/or continuous design models. Thus, as conceptually illustrated in Figure 2 and on the case study in Figure 12, the specification models monitor the design models. If the design models do not satisfy one or several requirements of the specification models for a given operational scenario, then the simulation environment either logs the event or stops the execution before sending an alert (see example in section 4.1 or [7], [8], [10]). Specification models are engineered and validated for correctness and completeness by systems engineers before starting the design activity at the system, sub-system, or component level. The recursive top-down decomposition of the design and system specification models at the system level into design and specification models at the sub-systems level is driven by various influential factors including, but not limited to:

1. The enterprise architecture often mirrors the hierarchical decomposition of the system-of-interest. For instance, an aircraft is decomposed into subsystems (avionic subsystem, landing gear subsystem, hydraulic subsystem, electrical generation and distribution subsystem, etc.) and each of them corresponds to an engineering department.
2. Standards often provide systems developers with decomposition guidelines (not to say constraints). For instance, the Air Transport Association (ATA) provides a specification tree (since a specification should be written for each subsystem) and the ATA 100 is a common referring standard for commercial aircraft decomposition.
3. In various industrial processes, the decomposition of systems into subsystems is driven by a functional analysis that is part of a top-down zigzagging process between the problem/specification domain and solution/design domain.
4. Finally, the design of large systems is often a derivative design, that is, a design that utilises major components of existing systems as the basis for the new system. Therefore, the decomposition is relatively unchanged.

We assume that these models exist and results from design activities carried out by engineers. If not, the company shall consider updating its design process, methods, and tools.

[Limits] First of all, Model-In-the-Loop verification and, ipso facto, the entire process introduced in this paper require systems engineers to be able to specify unambiguous requirements in a semi-formal graphical modelling language leading to complete and correct specifications. Therefore, at first glance, the approach seems to be more suitable for the derivative design of large engineered systems (e.g. aircraft, automotive, rail industry) that utilises major components – including requirements – of existing systems as the basis for the development of the new system which meets some new requirements. Although it is not a method that boosts creativity, it supports new product development as the simulation of requirements does not allow the engineer to escape the translation of ambiguous needs into SMART requirements; a common practice that leads to the failure of numerous projects. In addition, Model-in-the-Loop verification is limited to predefined or recorded operational scenarios, whereas sound verification requires to extensively stimulate the system. Therefore, there is a need to run numerous and heterogeneous operational scenarios to make sure the design meets the requirements. The definition of operational scenarios is limited to the nominal and well-known critical scenarios while missing potential unintended situations. Furthermore, the visuals are graphs, tables, and 2D or, at best, desktop-based 3D animations, whereas simulation outputs require an intuitive visualisation thanks to immersive visualisation techniques [13], [50]. By integrating virtual reality into PMM, the level of immersion and, consequently, the level of realism would be higher because of the improvement of many visualisation performance criteria, including but not limited to stereoscopy, the field of view, display size, resolution, frames per second [51].

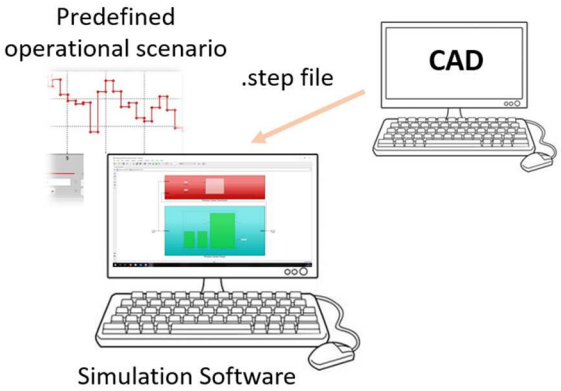

Figure 6: Model-In-The-Loop verification

3.2. Immersive Model-In-The-Loop Verification

Reminding that our goal is to continuously verify the systems by providing stakeholders with increasing realistic conditions, we expect to improve the realism of the Model-In-the-Loop verification with visual immersion. Thus, to perform a much more intuitive visualisation of the system properties, we conserve the co-simulation of specification models and design models, but we replace the native visualisation capabilities of the physics-based simulation software with an immersive environment giving birth to the **Immersive Model-in-the-Loop** verification strategy. Figure 7 shows the framework combining the physics-based simulation software and a rendering engine that displays an immersive virtual scene to the user wearing a virtual reality Head-Mounted Display (HMD). The virtual scene contains a physics-based simulation that emulates the system's behavioural properties, a data visualisation dashboard that contains the requirements and a 3D metaphor that represents their status (untested, tested and satisfied, tested and unsatisfied) in real-time.

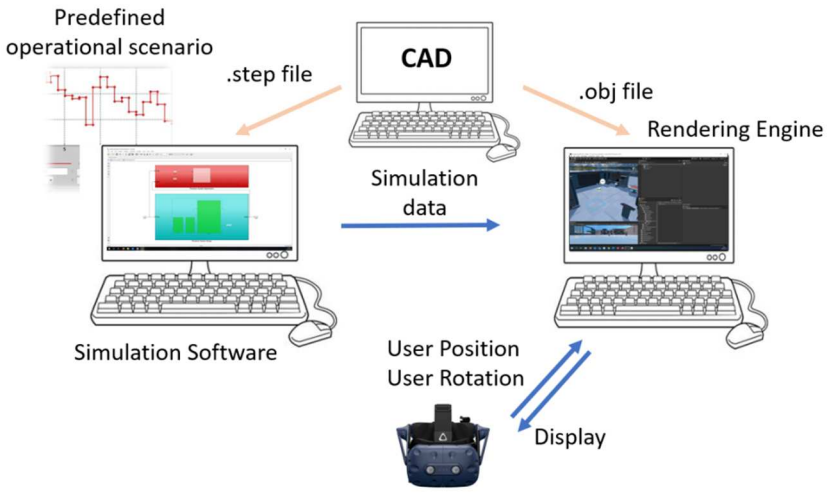

Figure 7: Immersive Model-In-The-Loop verification

[Limits] Immersive Model-In-the-Loop drastically improves the visual appreciation of the structural properties and the coupling with behavioural properties (e.g. kinematics and dynamics of multi-body systems). For instance, the end-user could visually appreciate the overshooting dynamic phenomenon of a robot arm. However, we still have a lack of realism in the interactions provided by this simulation. Indeed, the user cannot directly interact with the system limiting the virtual experience to planned (models or recordings) operational scenarios without unintended events.

3.3. Immersive Model- and Human-In-The-Loop Verification

In the last section, we improved the realism during the verification activity, but VR must not be restricted to 3D visual perception as it offers advanced interactive capabilities. The integration of Human-In-the-Loop simulation into the Immersive Model-In-the-Loop verification strategy aims at providing the end-user with the opportunity to naturally operate the system with some operational randomness. We name this new verification strategy Immersive Model- and Human-In-the-Loop verification (Figure 8). For this step, VR captures intentional and unintentional hand gestures of the user to replace the predefined operational scenarios. The choice of the device to capture hand gestures in virtual reality – remote controller, joystick, haptic arm, gloves, etc. – depends on the use case. For instance, Figure 9 shows a user interacting in real-time with the simulated system and visualising the results of the stimulus he gives to the system. He also can appreciate the state of the requirements defined for this system. Stimuli in Human-In-the-Loop simulation are not limited to hand gestures.

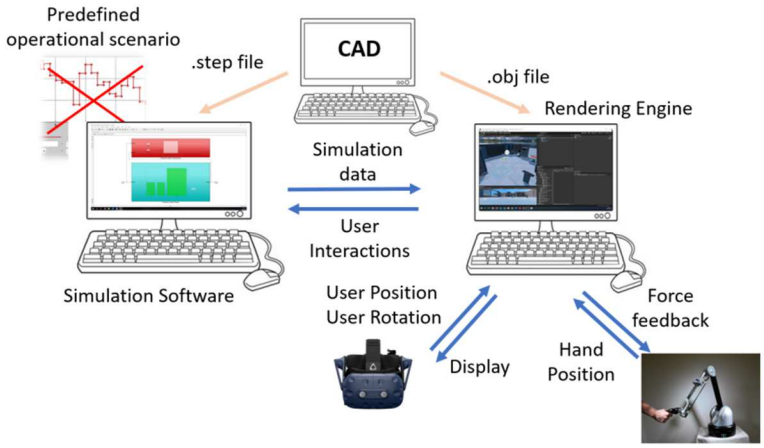

Figure 8: Immersive Model- and Human-In-The-Loop verification

[Limits] Immersive Model- and Human-In the Loop verification provides end-users with a virtual experience that integrates the co-simulation of specification models and design models with a combination of realistic behavioural properties, structural properties, and operational scenarios. However, as the system is fully virtual, modelling and simulation assumptions may still lead to some gaps between the verified virtual system and the first real prototype.

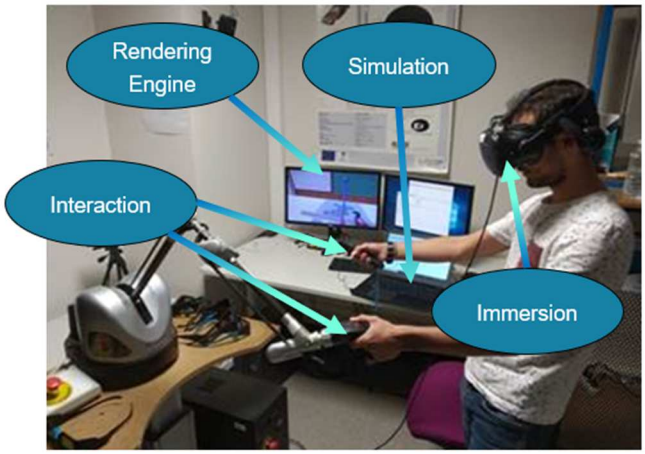

Figure 9: Example of an Immersive Model- and Human-In-the-Loop verification

3.4. Immersive Model-, Human-, and Hardware In-The-Loop Verification

Immersive Model-, Human- and Hardware-In-the-Loop verification consists in substituting virtual building blocks from the design models with real ones up to the first real prototype instrumented with sensors that feed the system specification model. The simulation of the virtual system embedding real components enables designers to get rid of modelling and simulation assumptions. Immersive Model-, Human- and Hardware-In-the-Loop verification (Figure 10) requires connecting a control board to the simulation software to pilot real components or subsystems and collect measured data from sensors. Today, the verification with advanced Hardware-In-the-Loop (HIL) [14] integrated dynamic test-bed like the Dynamic Helicopter Zero[1] in Airbus Helicopters remains

[1] https://www.airbushelicopters.com/website/en/press/DHC0:-New-dynamic-testing-method-brings-mature-helicopters-to-market-quickly_1808.html (accessed on 11/12/2021)

expensive and such realistic hybrid (virtual-real) verification strategy speeds up the maturity process by detecting design and specification errors before moving to fully real testings.

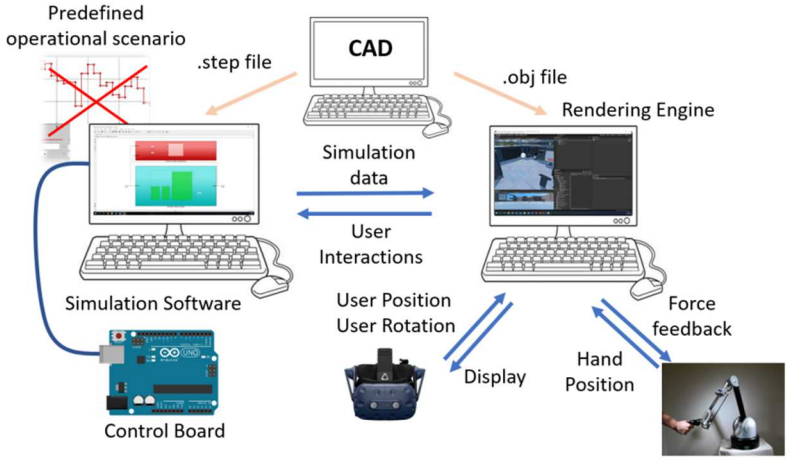

Figure 10: Immersive Model-, Human-, and Hardware-In-The-Loop Verification

4. Use Case: Computer-Aided Validation & Verification of a Robot Arm

To illustrate the user-centric computer-aided verification process, we implement the workflow on the design of a robot arm.

4.1. Model-In-The-Loop Verification

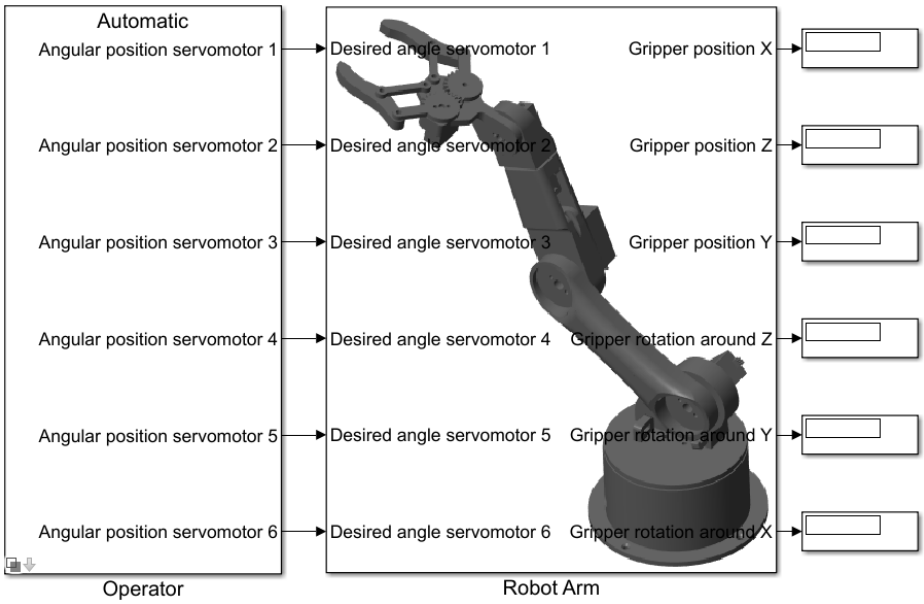

Figure 11: The system-of-interest robot arm as a black box piloted by an operator. The operator model is a variant bloc enabling the selection of automatic recorded operating scenarios or, later on in the paper, manual human-in-the-loop operating scenarios.

The process starts with the development of the physics-based simulation model that includes the definition of the operation that executes the operational scenarios (block on the left in Figure 11) and the system of interest (block on the right in Figure 11) with outputs corresponding to the position of the gripper in space. The operational scenarios are predefined time series that stimulate the servomotors of the robot arm. By opening the “Robot Arm” block in Figure 11, we find the co-

simulation of the system specification model with the system design model where the former monitors the inputs and outputs of the latter (Figure 12).

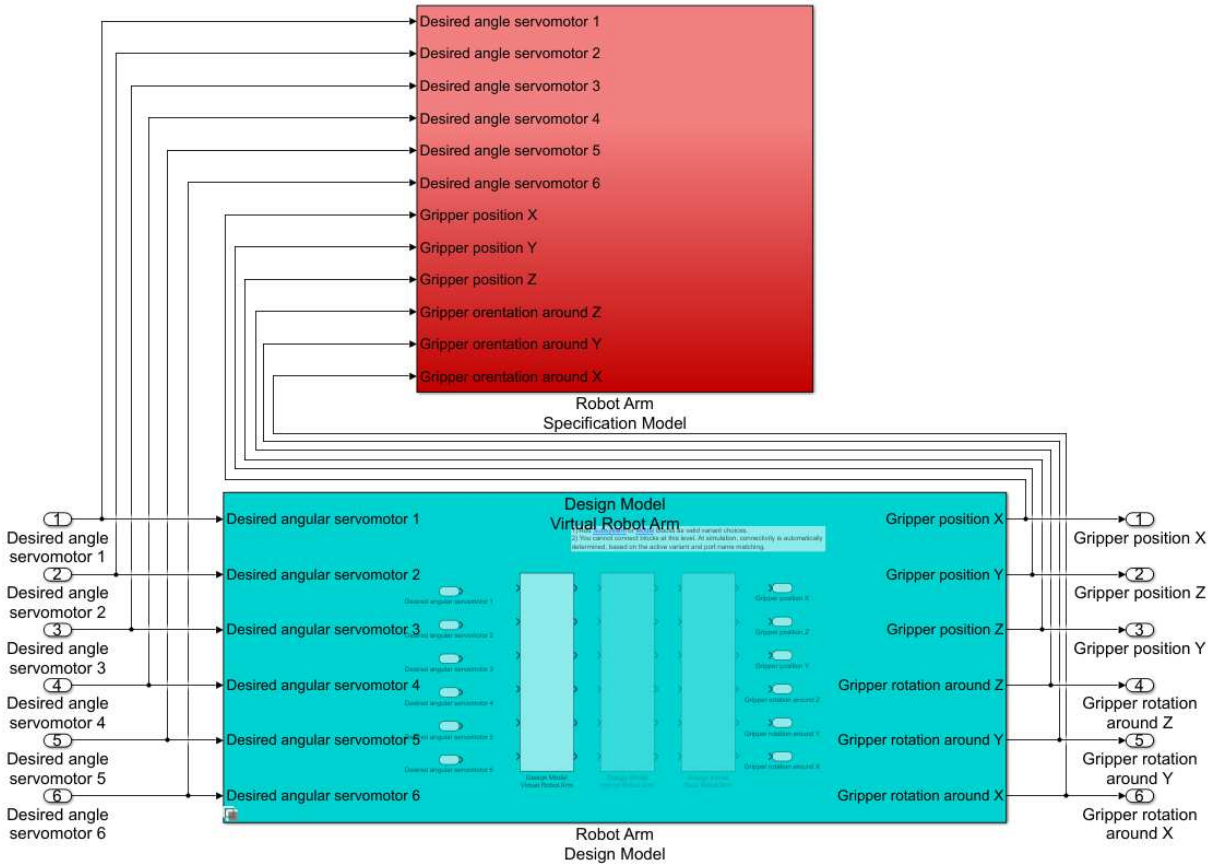

Figure 12: Co-simulation of a design model (in blue) and a specification model (in red)

The specification model is a set of Property-Based Requirements (PBRs) [42] that, at every simulation step, evaluate whether the intended effects are satisfied when the condition is true. Each PBR is modelled in a block (Figure 13 and Figure 14) within the specification model.

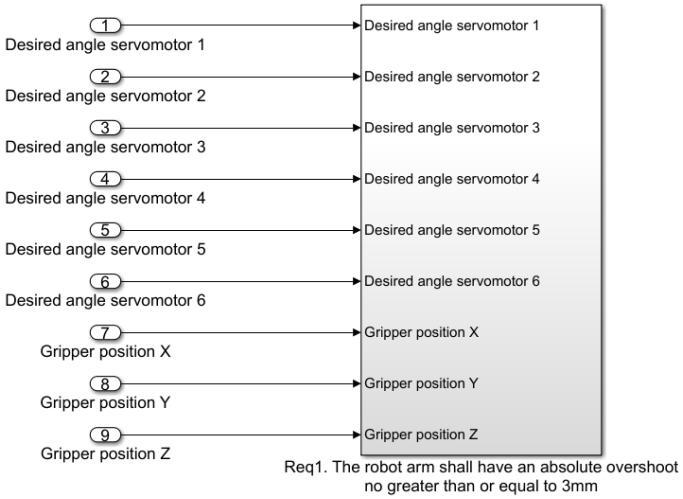

Figure 13: A Property-Based Requirement within the specification model

As illustrated in many previous studies [7], [8], [10], [41], [44], [52], a PBR model (Figure 16) is a logical predicate modelled with a block *Implies* that lets us specify that the requirement is satisfied when the condition is *True* (active) and the intended effect is also *True*. However, if the condition is

True but that the intended effect is *False*, then the assertion is *False* and the simulation stops or the signal corresponding to an unsatisfied requirement is logged in a reporting file. If the condition is *False*, then the requirement is not evaluated. Modifying the definition of a PBR without adding or removing interfaces does not impact any other models. If a new interface is required (e.g., desired angle servomotor 7), then it is necessary to add the interface in the requirement model (Figure 13 and Figure 14), in the specification model (Figure 12), in the design model (Figure 12) and to add communication blocs (e.g. send/receiveUDP) for communicating signals to remote applications (e.g. Unity models).

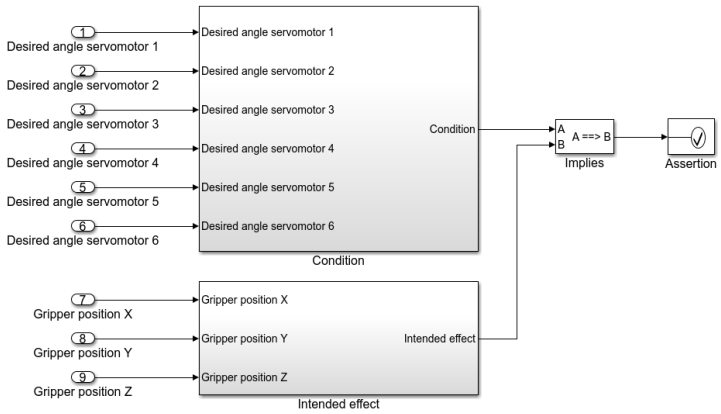

Figure 14: The definition of the Property-Based Requirement of Figure 13

The system design model (Figure 12) is a variant block that enables us to select a fully virtual design of the robot arm for Model-In-the-Loop verification, or a hybrid virtual-real design model for Hardware- and Model-In-the-loop verification, or a fully real design model corresponding to the last Prototype-In-the-Loop verification strategy. For Model-In-the-Loop verification, the design model of the robot arm is an architecture that is recursively decomposed into multi-physical components (Figure 15) where the co-simulation of specification and design models is achieved at all systemic levels.

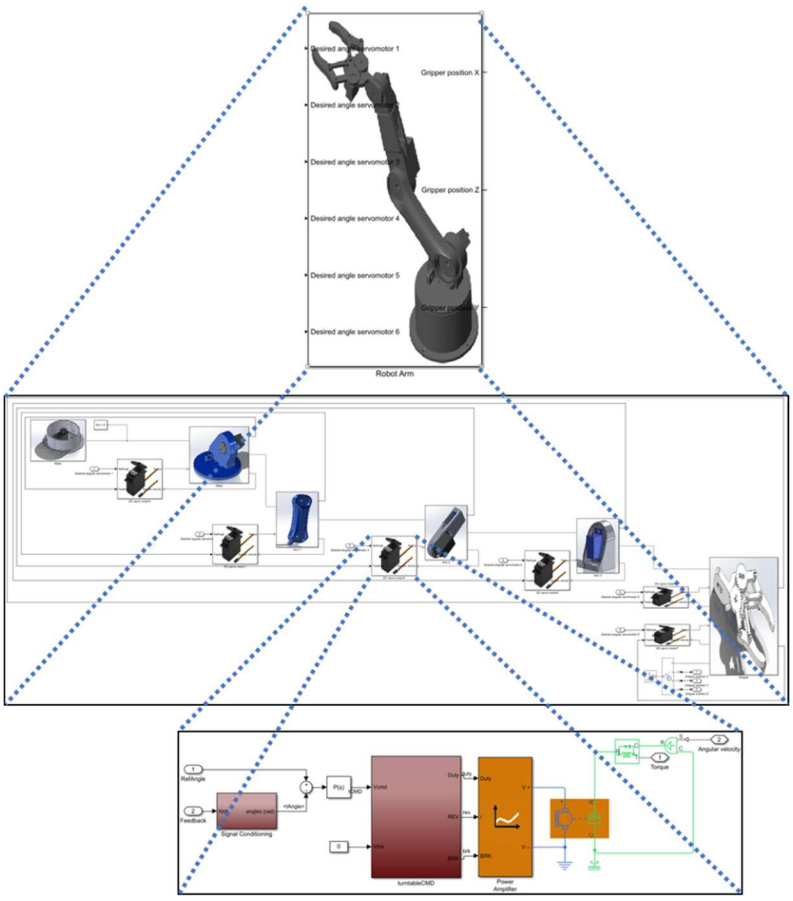

Figure 15: Decomposition in sub-systems of the robot arm

Figure 16 shows the robot arm 3D animation resulting from the simulation of the design models stimulated by predefined operational scenarios.

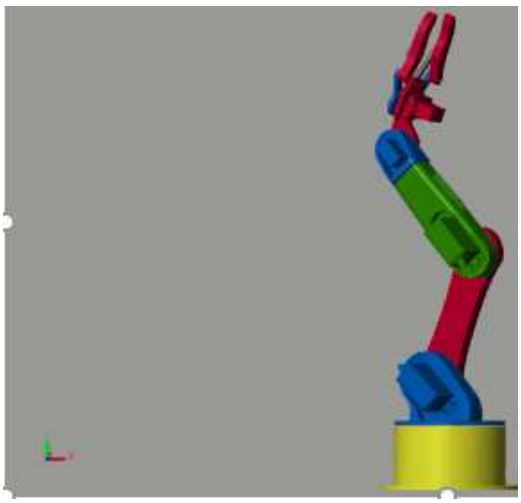

Figure 16: Multibody animation of the robot in Simscape

4.2. Immersive Model-In-The-Loop Verification

Immersive Model-in-the-Loop verification requires a VR engine to communicate with the Model-In-the-Loop physics-based simulation. This interconnection uses UDP network communication because

there is no verification from the server of the data received by the client leading to faster data exchange. Data exchange alternatives can be the Open Platform Communications Unified Architecture (OPC UA), Open Services for Lifecycle Collaboration (OSLC), Functional Mock-up Interface (FMI), or Model-Driven Engineering, but this paper does not intend to compare candidates communication protocols to find the best one although we consider it as a potential future work. The VR visualisation module was developed within Unity3D [53]. The Simulink blocks SendUDP send the positions of all the elements composing the robot to another computer running Unity3D (Figure 7).

Figure 17: Immersive Model-In-The-Loop Verification of the robot arm

In Unity3D, to animate the system in the virtual environment, we open the communication ports and read the received data with C# scripts. Figure 17 shows that the user can visualise the robot at scale 1 animated by a realistic physics-based behavioural simulation that is still stimulated by predefined operational scenarios. Regarding the visual metaphors for the specification model, we use a 2D screen that is always present in the scene, but the user can freely grab and move it. With this screen, the user can select a function (e.g. To control overshooting) by navigating through a tree of functions. Once a function is selected, the screen displays the navigable list of requirements that specify the function (e.g. For a step command of 90°, the robot arm shall have an absolute overshoot no greater than or equal to 3mm). These requirements serve to verify the design. A virtual lamp, on the top right of a requirement name (grey in Figure 17), indicates the state of the requirement: grey if it is not tested, green if it is verified and red if it is falsified.

All Unity models can be easily reused for another project including a robotic arm. Indeed, the definition of the kinematic joints for assembling parts is automated by a script based on the CAD model created during the design activities. The allocation of UDP communication ports associated with the controlled angular positions is also easy to automate with a pre-defined list of reserved ports on the enterprise network. Of course, if one wants to define an assembly line with multiple robot arms, he will have to instantiate the robot arm several times and set up the positions within the assembly line.

4.3. Immersive Model- and Human-In-The-Loop Verification

For Immersive Model- and Human-In-the-Loop verification, the predefined (recorded or manually defined) operational scenarios, which were preferred in the previous verification activities, are replaced by Human-In-the loop simulation. Practically, this simply requires switch in from the bloc

"Automatic" to the bloc "Manual" within the block "Operator" in Figure 11. The human operator is now able to perform on-the-fly operating scenarios in real-time from the immersive environment. Figure 18 illustrates the human-machine interaction through a tablet that enables the user to operate the robot arm by setting the angular position of each virtual servomotor from Unity. The desired angular position is sent to the physics-based design model in Simscape to update the simulation before sending back the new positions and orientations of the robot arm to the virtual reality scene. With such a phygital environment, we can imagine various interactions, such as haptic control and feedback, or to use the virtual avatar of the end-user hand in Unity to indicate the desired gripper position in the scene. Interactions would still generate events for updating the physics-based simulation that calculates the angular positions of the motor to reach the desired position.

Figure 18: Immersive Model- and Human-In-The-Loop Verification of the robot arm

We can use the same communication protocol in the other direction: the VR software sends the interaction data to the physics-based simulation through the communication network using a UDP protocol. It is the HMD HTC Vive remote controllers that capture the human-machine interactions. We also use a haptic arm, the Virtuose 6D 35-45, which captures the position of the hand and gives force feedback when the user touches the virtual robot arm. When providing significant force feedback, the user develops a certain apprehensiveness, not to say fear, to touch the robot again leading to an increase of the level of presence, that is, the sense of "being in" the scene and the user starts behaving as he was experiencing the system in the real world. This scenario is of major interest to evaluate the collaboration between a cobot and an operator, especially the safety properties.

4.4. Immersive Model-, Human-, and Hardware-In-The-Loop Verification

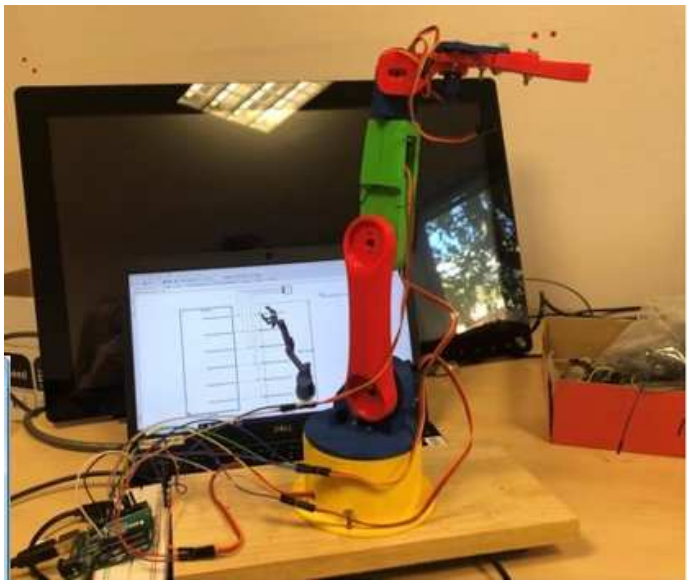

Figure 19: Robot arm prototype

In our case study, we start replacing the design model of the "Controller" subsystem with the Arduino Uno board (Figure 10). Then, we can replace the design model of the "Servomotor" subsystems with MG996R servos and SG90 micro servos. Note that just the sound of real servos resulting from the changes of angular positions ordered by the human-in-the-loop simulation in virtual reality significantly increases the realism of the virtual verification experience. We can proceed to a step-by-step verification of the real components before integrating them up to the verification of the full prototype instrumented with sensors (Figure 19). In our case, we use a potentiometer (COM-KY040 from Joy-it) to measure the angle of the servomotors in each joint. The robot arm prototype can finally be verified by the system specification model (Figure 20).

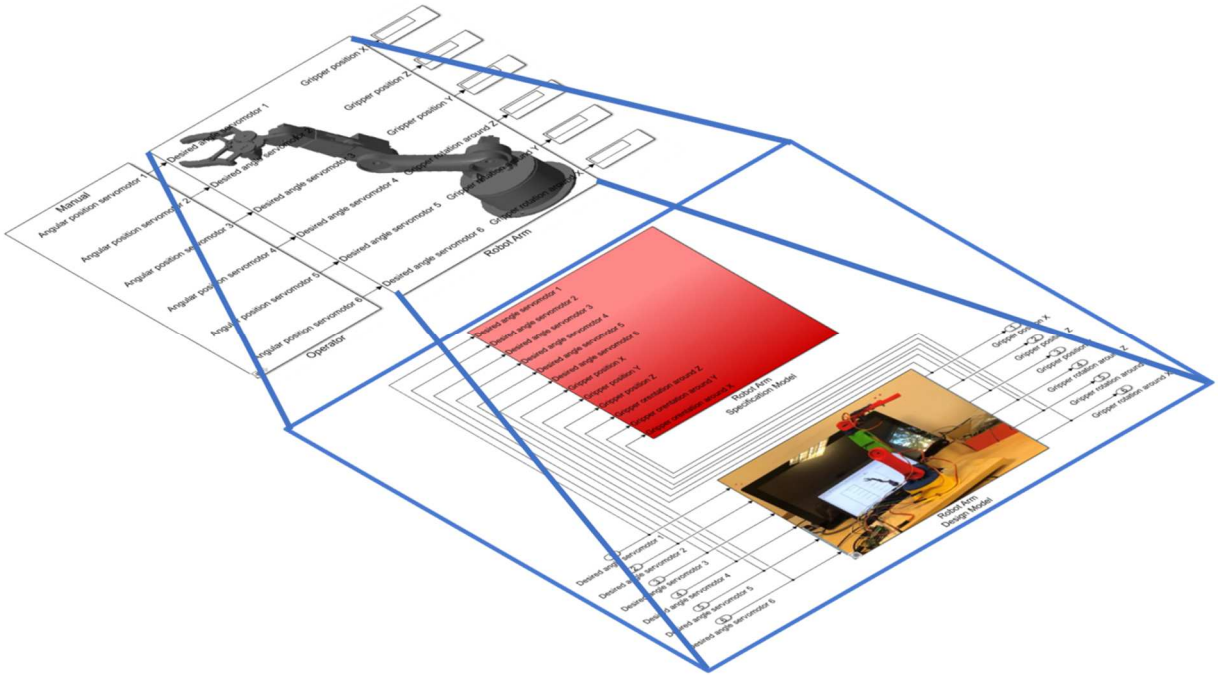

Figure 20: Prototype-In-the-Loop verification (represented at the bottom but actually within the bloc "Robot Arm" at the top) manually stimulated by a human operator

5. Discussion

Analytical verifications strategies strive to integrate numerous technologies and develop immersive virtual environments to experience more and more realistic conditions. Nevertheless, we hardly know: how much realism is enough to verify a design? In this section, we discuss the development efforts (Table 1) to implement our process. The implementation time required to apply the method to the robot arm is also detailed.

5.1. Development efforts

We may derive parsimonious recommendations that would help companies to reach the right level of realism according to the verification goals and the available resources.

- **Model-In-The-Loop Verification**

To perform a Model-In-the-Loop verification, we need a CAD model in a STEP format to import it into the physics-based simulation software. A system simulation model including three types of models – operational scenarios, specifications, and designs – at all systemic levels must be complete. Reaching such a level of maturity in terms of methods and tools for engineering technical systems is already requiring a big change in most companies.

- **Immersive Model-In-The-Loop Verification**

To add the immersive visualisation capabilities to the Model-In-the-Loop Verification, we need to send data from the simulation software through the network to the rendering engine. Interoperability alternatives including shared memory and data exchange standards such as Functional Mock-Up Interface (FMI) [54] could also be considered. The simulation model should be updated to add “send data” functions to the outputs of any moving element. To visualise the state of the requirements, we also must modify the specification models by adding the same “send data” functions. Figure 21 illustrates the exchanges between the computer running the co-simulation and the virtual environment. These two operations are manual but they could be automated with Matlab scripts. In the rendering engine, we must import the 3D model. The conversion of a B-REP CAD model in STEP format into OBJ format is automated thanks to our CAD converter. It also gets the relative positions of parts within an assembly. An ad-hoc macro in Solidworks extracts the definition of kinematic joints from the native kinematic CAD model and a C# script automatically rebuilds the kinematic joints in Unity. The definition of text-based requirements in the VR panel is manual but the task could be automated with a macro parsing the requirements in the Simulink specification models.

- **Immersive Model- and Human-In-The-Loop Verification**

When adding end-user interactions with the system-of-interest, to perform an Immersive Model- and Human-In-the-Loop verification, human-machine interactions must be implemented in virtual reality. Some interactions can be reused from models in previous projects, but every model can be different leading to the need for new VR developments, making this step very time expensive, mostly for systems with complex interactions or with the use of other devices such as haptic arms. To communicate the data collected by the user interface to the simulation software, the functions to send and receive data from the network can be reused. For example, in the robot arm illustration, the interaction provided to the user with the robot is a tablet to control the angular position of all piloted joints. This requires 4 steps:

1. Create a 3D model of a tablet, which is an effortless step since this 3D model remains simple.

2. Create interactions with the tablet. That's the most time-consuming task. For each object with which we want to interact, a new C# script has to be created to describe the behaviour of the interactive objects and to compute the new values. In our example, we get the position of the sliders to compute the desired position of joints.
3. Send data to the simulation. Thanks to a reusable script, sending data means calling a function in the C# script which describes the behaviour of the interactive objects.
4. In the simulation software: collect the data received from the virtual environment and link it with the "command" block. Again, blocks already exist in the simulation software reducing the development effort.

- **Immersive Model-, Human- and Hardware-In-the-Loop Verification**

Immersive Model-, Human- and Hardware-In-the-Loop verification requires only modifications on the simulation model. Once the prototype or a sub-system prototype is manufactured, we need to connect it to the simulation software by instrumentalising it with sensors. The outputs of the simulation pilot the hardware in the loop, whereas the outputs of the hardware feedback the physics-based simulation. All these tasks may be time expensive, mostly if the simulation software and the hardware are not easily compatible for Hardware-In-the-Loop simulation.

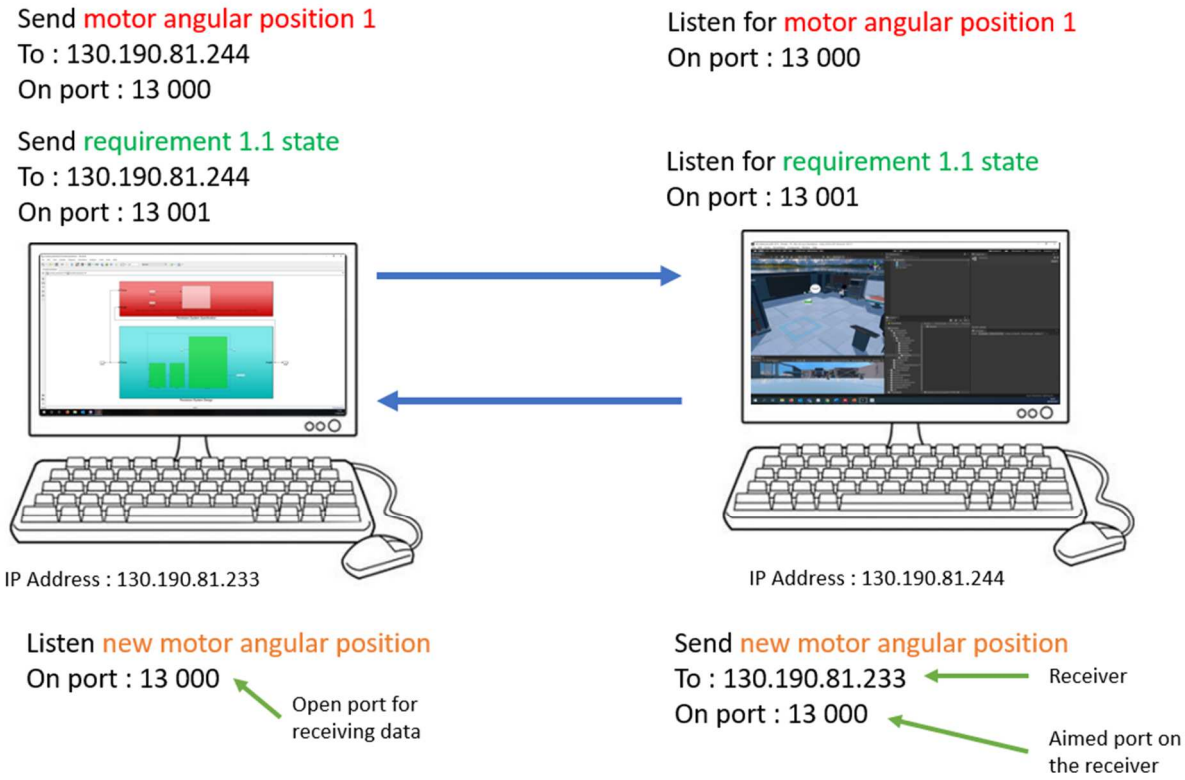

Figure 21: Data exchange between the co-simulation

Table 1: Summary of the development efforts

Phase	Task	Resources			Implementation Effort
		Hardware	Software	Skills	
Model-In-the-Loop	Create the CAD Model	• PC	CAD	CAD	Nominal (3h)
	Export CAD Model to STEP format		CAD	CAD	Very low (0.25h)
	Import CAD Model to Behaviour Model		Simulation	Simulation	Very low (0.25h)
	Create the Behaviour Model		Simulation	Simulation	Nominal (4h)
	Create the Specification Model		Simulation	Simulation	High (5h)
Immersive Model-In-the-Loop	Send positions from Simulation	• PC • Network	Simulation	Simulation	Nominal (1h)
	Send information from Specification		Simulation	Simulation	Low (0.5h)
	Import CAD Model (OBJ format)		VR	VR	Low (0.25h)
	Setup communication port for CAD Model		VR	VR	Low (0.25h)
	Setup communication port for Specification		VR	VR	Low (0.25h)
Immersive Model- and Human-In-the-Loop	Implement interactions with 3D Model	• PC • Network • VR HMD • Human-Machine interaction device	VR	Scripting (C#)	Very high (12h)
	Send data from VR		VR	Scripting (C#)	Very Low (0.25h)
	Use received data as input (Simulation)		Simulation	Simulation	Low (0.5h)
Immersive Model-, Human-, and Hardware-In-the-Loop	Manufacture subsystem / prototype	• PC • Network • VR HMD • Human-Machine interaction device • Subsystems prototype	Rapid prototyping machines	Manufacturing	Nominal (20h)
	Instrumentation of the physical system			Mechatronics	Nominal (3h)
	Connect subsystems to simulation		Simulation	Hardware-In-the-Loop simulation	High (7h)

		• Sensors			

5.2. How much realism is enough?

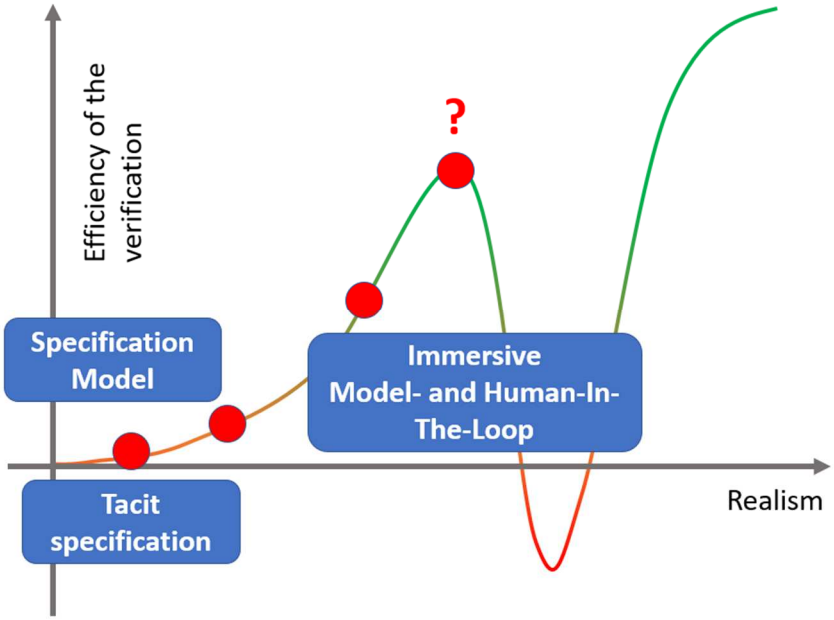

Figure 22: An attempt to conceptually define the uncanny valley of immersive verification

To provide more objective criteria, we may borrow the phenomenon of the uncanny valley for humanoids [55]. The uncanny valley translates the unexpectedly negative dislike reactions provoked by imperfect human-likeness in human robots [56]. By analogy, we may assume that more sensorial feedback does not systematically increase the level of realism. For example, Berger *et al.* has shown that is the case for haptics feedback [57]. We may therefore assume that, at a certain point, more realism does not always increase the efficiency of the verification process. Figure 22 illustrates a new research hypothesis stating that this phenomenon may exist in our virtual environment for systems verification. Therefore, as future work, we will attempt to experimentally demonstrate the existence or not of the uncanny valley phenomenon in our user-centric computer-aided verification process to finally be able to characterise the "right need" of the user, that is, the perfect balance between the time and cost spent for developing a realistic virtual environment and its efficiency in terms of verification (e.g. the number of design errors identified early on).

6. Conclusion

Model-based requirements and systems engineering, virtual reality, physics-based simulation, and hardware-in-the-loop simulations are key technologies for the early verification of engineered systems. However, there was a need for a continuous integration of these technologies into a user-centric computer-aided verification process for verifying engineered systems in realistic conditions. We also argued that it is possible and relevant to integrate VR technologies with an MBSE method to recreate realistic operational conditions while keeping the added value of the co-simulation of specification models and design models. In addition, by progressively substituting the virtual building blocks with real ones, the designer can be confident that the structural and behavioural properties of the prototype will satisfy the requirements, avoiding then extra costs due to design errors.

As future works, we may work on specific tasks such as the evaluation of candidate data exchange solutions (e.g. Open Service for Lifecycle Collaboration[2], Functional Mock-Up Interface[3], Open Platform Communications Unified Architecture[4], etc.) to find the most suitable one. However, before anything else, we will look for an industrial use case to evaluate our proposal on a system that is more complex than the robot arm. In parallel, we plan to characterise the uncanny valley to derive recommendations for companies desiring to reach a parsimonious balance between the time and cost spent for developing a realistic virtual environment and its efficiency in terms of verification.

Acknowledgement

This work was supported by the LabEx Persyval Mine'VR Project.

References

[1] H. Crisp, "INCOSE systems engineering vision 2020," 2007.
[2] A. W. Wymore, *Model-Based Systems Engineering*. CRC Press, 2018.
[3] T. Bahill, "In Memoriam A. Wayne Wymore," *INSIGHT*, vol. 14, no. 2, pp. 57–61, Jul. 2011.
[4] N. P. Suh, *Axiomatic design : advances and applications*. Oxford University Press, 2001.
[5] INCOSE, *Systems engineering handbook A guide for system life cycle processes and activities*. 2015.
[6] M. Hoppe, A. Engel, and S. Shachar, "SysTest: Improving the verification, validation, and testing process— Assessing six industrial pilot projects," *Syst. Eng.*, vol. 10, no. 4, pp. 323–347, 2007.
[7] R. Becquet, P. Micouin, L. Fabre, F. Guérin, P. Paper, and T. Razafimahefa, "Property Model Methodology: A Landing Gear Operational Use Case," *INCOSE Int. Symp.*, vol. 28, no. 1, pp. 321–336, 2018.
[8] L. Fabre, P. Micouin, C. Gaurel, and P. Pandolfi, "Advances in Property Model Methodology (PMM ®)," in *Vertical Flight Society Forum*, 2020.
[9] M. Otter *et al.*, "Formal Requirements Modeling for Simulation-Based Verification," *Proc. 11th Int. Model. Conf. Versailles, Fr. Sept. 21-23, 2015*, vol. 118, pp. 625–635, 2015.
[10] P. Micouin, "PMM in the light of FAA Requirement Engineering Management Handbook Part 1 - From A Conops to the Validated Top Level Specification," *Manag. Handb. Part 1 - From A Conops to Validated Top Lev. Specif.*, January, 2022.
[11] D. Mestre and J.-L. Vercher, "Immersion and presence," *Virtual Real.*, 2012.
[12] T. Stoffregen, R. Pagulayan, L. Smart, and B. Bardy, "On the Nature and Evaluation of Fidelity in Virtual Environments," *Virtual Adapt. Environ.*, August 2014, pp. 111–128, 2003.
[13] I. Han and J. B. Black, "Incorporating haptic feedback in simulation for learning physics," *Comput. Educ.*, vol. 57, no. 4, pp. 2281–2290, 2011.
[14] P. Sarhadi and S. Yousefpour, "State of the art: hardware in the loop modeling and simulation with its applications in design, development and implementation of system and control software," *Int. J. Dyn. Control*, vol. 3, no. 4, pp. 470–479, Dec. 2015.
[15] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*. The MK/OMG PRESS, 2014.
[16] OMG, "OMG SysML Home | OMG Systems Modeling Language," 2017. [Online]. Available: http://www.omgsysml.org/.
[17] H. P. Hoffmann, "Model-based systems engineering with rational rhapsody and rational

[2] https://open-services.net/ (accessed on 01/03/2022)
[3] https://fmi-standard.org/ (accessed on 01/03/2022)
[4] https://opcfoundation.org/about/opc-technologies/opc-ua/ (accessed on 01/03/2022)

harmony for systems engineering," *Deskb. Release*, vol. 3, no. 2, 2011.
[18] S. Friedenthal, R. Griego, and M. Sampson, "INCOSE model based systems engineering (MBSE) initiative," *INCOSE 2007 Symp.*, no. August, 2007.
[19] F. Mhenni, J.-Y. Choley, O. Penas, R. Plateaux, and M. Hammadi, "A SysML-based methodology for mechatronic systems architectural design," *Adv. Eng. Informatics*, vol. 28, no. 3, pp. 218–231, Aug. 2014.
[20] T. Weilkiens, *SYSMOD-The systems modeling toolbox-pragmatic MBSE with SysML*. Lulu.com, 2016.
[21] H. Chalé Gongora, M. Ferrogalini, and C. Moreau, "How to boost PLE by using MBSE-A case study of a Rolling Stock product line," *Proceedings*, vol. 5, 2014.
[22] J.-D. Piques, "SysML for embedded automotive systems: SysCARS methodology," *Embed. Real Time Softw. Syst.*, 2014.
[23] RATIONAL THE SOFTWARE DEVELOPMENT COMPANY, "The Rational Unified Process for Systems Engineering 1.1. A rational software White Paper TP 165A, 5/02," 2002.
[24] J. A. Estefan, "Survey of model-based systems engineering (MBSE) methodologies," in *Incose MBSE Focus Group*, 2007, vol. 25, no. 8, pp. 1–12.
[25] P. Roques, "MBSE with the ARCADIA Method and the Capella Tool," *8th Eur. Congr. Embed. Real Time Softw. Syst. (ERTS 2016)*, 2016.
[26] J.-L. Voirin, *Model-based System and Architecture Engineering with the Arcadia Method*. Elsevier, 2017.
[27] E. Herzog and A. Pandikow, "2.3.1 SysML - an Assessment," *INCOSE Int. Symp.*, vol. 15, no. 1, pp. 293–305, Jul. 2005.
[28] J. Maurandy, E. Gill, A. Helm, and R. Stalford, "Cost-benefit analysis of sysml modelling for the atomic clock ensemble in space (ACES) simulator," *22nd Annu. Int. Symp. Int. Counc. Syst. Eng. INCOSE 2012 8th Bienn. Eur. Syst. Eng. Conf. 2012, EuSEC 2012*, vol. 3, pp. 1842–1861, 2012.
[29] D. Dori, "Object-Process Methodology for Structure-Behavior Co-Design," in *Handbook of Conceptual Modeling*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 209–258.
[30] C. J. J. Paredis *et al.*, "5.5.1 An Overview of the SysML-Modelica Transformation Specification," *INCOSE Int. Symp.*, vol. 20, no. 1, pp. 709–722, Jul. 2010.
[31] J. Maley and J. Long, "5.7.2 Systems Engineering: a Natural Approach to C4ISR," *INCOSE Int. Symp.*, vol. 13, no. 1, pp. 1185–1194, Jul. 2003.
[32] J. Morris, M. Ingham, R. Rasmussen, T. Starbird, and A. Mishkin, "Application of State Analysis and Goal-Based Operations to a MER Mission Scenario," in *SpaceOps 2006 Conference*, 2006.
[33] C. Piaszczyk, "Model Based Systems Engineering with Department of Defense Architectural Framework," *Syst. Eng.*, vol. 14, no. 3, pp. 305–326, Sep. 2011.
[34] Y. Grobshtein and D. Dori, "Generating SysML views from an OPM model: Design and evaluation," *Syst. Eng.*, vol. 14, no. 3, pp. 327–340, Sep. 2011.
[35] D. Dori, *Model-Based Systems Engineering with OPM and SysML*. New York, NY: Springer New York, 2016.
[36] K. Robinson, D. Tramoundanis, D. Harvey, M. Jones, and S. Wilson, "Demonstrating model-based systems engineering for specifying complex capability," *Syst. Eng. Eval. Conf.*, 2010.
[37] S. Kleiner and C. Kramer, "Model Based Design with Systems Engineering Based on RFLP Using V6," in *Smart Product Engineering. Lecture Notes in Production Engineering*, A. M. and S. R., Eds. Springer, Berlin, Heidelberg, 2013, pp. 93–102.
[38] R. Karban, R. Hauber, and T. Weilkiens, "MBSE in Telescope Modeling," *INSIGHT*, vol. 12, no. 4, pp. 24–31, Dec. 2015.
[39] D. Bouskela, T. Nguyen, and A. Jardin, "Towards a rigorous approach for verifying cyber-physical systems against requirements," in *2015 IEEE Electrical Power and Energy Conference (EPEC)*, 2015, pp. 250–255.
[40] V. Nilsson, "Model-Based Testing with Simulink Design Verifier," Chalmers University of Technology, 2014.

[41] P. Micouin, “Property-Model Methodology: A Model-Based Systems Engineering Approach Using VHDL-AMS,” *Wiley Online Libr.*, vol. 14, no. 3, pp. 305–326, 2013.
[42] P. Micouin, “Toward a Property BasedRequirements Theory:System RequirementsStructured as a Semilattice,” *Wiley Intersci.*, vol. 14, no. 3, pp. 305–326, 2010.
[43] S. ARPA, “Guidelines for development of civil aircraft and systems,” *SAE Int.*, 2010.
[44] R. Pinquie *et al.*, “Property Model Methodology : A Case Study with Modelica,” *Proc. Tools Methods Compet. Eng.*, April, pp. 1–12, 2016.
[45] T. Nguyen, “FORM-L: A MODELICA Extension for Properties Modelling Illustrated on a Practical Example,” *Proc. 10th Int. Model. Conf. March 10-12, 2014, Lund, Sweden*, vol. 96, pp. 1227–1236, 2014.
[46] L. Buffoni and P. Fritzson, “Expressing Requirements in Modelica,” *SNE Simul. Notes Eur.*, vol. 25, no. 3–4, pp. 21–22, 2016.
[47] L. de Casenave and J. E. Lugo, “Effects of Immersion on Virtual Reality Prototype Design Reviews of Mechanical Assemblies,” in *Volume 7: 30th International Conference on Design Theory and Methodology*, 2018, vol. 7, pp. 1–11.
[48] L. P. Berg and J. M. Vance, “Industry use of virtual reality in product design and manufacturing: a survey,” *Virtual Real.*, vol. 21, no. 1, 2017.
[49] PWC, “For US manufacturing, virtual reality is for real,” *2015 Disruptive Manufacturing Innovations Survey*, 2015. [Online]. Available: https://www.pwc.com/us/en/industrial-products/next-manufacturing/augmented-virtual-reality-manufacturing.html. [Accessed: 23-May-2019].
[50] R. Pausch, D. Proffitt, and G. Williams, “Quantifying immersion in virtual reality,” *Proc. 24th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH ’97*, August 1997, pp. 13–18, 1997.
[51] D. A. Bowman and R. P. McMahan, “Virtual reality: How much immersion is enough?,” *Computer (Long. Beach. Calif).*, vol. 40, no. 7, pp. 36–43, 2007.
[52] P. Micouin, “Model Based Systems Engineering using VHDL-AMS,” *Procedia Comput. Sci.*, vol. 16, pp. 128–137, 2013.
[53] Unity Technologies, “Multiplatform - VR-AR,” 2020. [Online]. Available: https://unity3d.com/fr/unity/features/multiplatform/vr-ar. [Accessed: 10-Mar-2020].
[54] V. Waurich and J. Weber, “Interactive FMU-Based Visualization for an Early Design Experience,” *Proc. 12th Int. Model. Conf. Prague, Czech Republic, May 15-17, 2017*, vol. 132, pp. 879–885, 2017.
[55] M. B. Mathur and D. B. Reichling, “Navigating a social world with robot partners: A quantitative cartography of the Uncanny Valley,” *Cognition*, vol. 146, pp. 22–32, 2016.
[56] M. Mori, “The uncanny valley,” *Energy*, vol. 7, pp. 33–35, 1970.
[57] C. C. Berger, M. Gonzalez-Franco, E. Ofek, and K. Hinckley, “The uncanny valley of haptics,” *Sci. Robot.*, vol. 3, no. 17, p. eaar7010, 2018.