Early Validation and Verification of System Behaviour in Model-Based Systems Engineering: A Systematic Literature Review

JOHAN CEDERBLADH, Mälardalen University, Sweden
ANTONIO CICCHETTI, Mälardalen University, Sweden
JAGADISH SURYADEVARA, Volvo Construction Equipment, Sweden

In the Systems Engineering (SE) domain there has been a paradigm shift from document-based to model-based system development artefacts; in fact, new methodologies are emerging to meet the increasing complexity of current systems and the corresponding growing need of digital workflows. In this regard, Model-Based Systems Engineering (MBSE) is considered as a key enabler by many central players of the SE community. MBSE has reached an adequate level of maturity and there exist documented success stories in its adoption in industry. In particular, one significant benefit of utilising MBSE when compared to the traditional manual and document-centric workflows is that models are available from early phases of systems development; these enable a multitude of analyses prior any implementation effort together with other relevant capabilities, like the automation of development tasks. Nonetheless, it is noticeable there is a lack of a common understanding for how formal analyses for the verification and validation (V&V) of systems behaviour, specifically in the early phases of development, could be placed in an MBSE setting.

In this article, we report on the planning, execution, and results of a systematic literature review regarding the early V&V of systems behaviour in the context of model-based systems engineering. The review aims to provide a structured representation of the state-of-the-art with respect to motivations, proposed solutions, and limitations. From an initial set of potentially relevant 701 peer-reviewed publications we selected 149 primary studies, which we analysed according to a rigorous data extraction, analysis, and synthesis process. Based on our results, early V&V has usually the goal of checking the quality of a system design to avoid discovering flaws when parts are being concretely realised; SysML is a *de facto* standard for describing the system under study, while the solutions for the analyses tend to be varied; also V&V analyses tend to target varied properties with a slight predominance of functional concerns, and following the variation mentioned so far the proposed solutions are largely context specific; the proposed approaches are usually presented without explicit limitations, while when limitations are discussed, readiness of the solutions, handling of analyses simplifications/assumptions, and languages/tools integration are among the most frequently mentioned issues.

Based on the survey results and the standard SE practices, we discuss how the current state-of-the-art MBSE supports early V&V of systems behaviour with a special focus on industrial adoption, and identify relevant challenges to be researched further.

CCS Concepts: • **Computing methodologies** → **Model verification and validation**; **Modeling methodologies**.

Additional Key Words and Phrases: MBSE, Validation, Verification, System behaviour, Systematic literature review

1 INTRODUCTION

Systems engineering (SE) is a paradigm that involves various processes and methodologies for life-cycle management of systems [9]. In its basic form a system is defined as *"... an integrated set of elements, subsystems, or assemblies that accomplish a defined objective."* in the SE handbook by the International Council on Systems

Authors' addresses: Johan Cederbladh, Mälardalen University, Universitetsplan 1, Västerås, Sweden, johan.cederbladh@mdu.se; Antonio Cicchetti, Mälardalen University, Universitetsplan 1, Västerås, Sweden, antonio.cicchetti@mdu.se; Jagadish Suryadevara, Volvo Construction Equipment, Bolindervägen 5, Eskilstuna, Sweden, jagadish.suryadevara@volvo.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2023 Copyright held by the owner/author(s).
1049-331X/2023/11-ART
https://doi.org/10.1145/3631976

Just Accepted

Engineering (INCOSE) [88]. The practices of SE are meant to aid the engineers and practitioners of various disciplines to communicate and rigorously perform activities to ensure that the system is delivered correctly. Indeed, SE prescriptions and guidelines are followed for larger projects and activities typically encompassing a multitude of disciplines across various fields (e.g., satellites, aeroplanes, nuclear plants, etc.); in these development endeavours it is vital that the participants can communicate efficiently and align efforts via standard practices and methods. Viewing SE from the INCOSE SE handbook, it is clear that there is a multitude of processes and methodologies that could be utilised when performing activities related to a system's life-cycle. The life-cycle of a system is usually divided into stages with clear separations. Notably, the ISO generic life-cycle standard (ISO/IEC/IEEE 15288:2015)[1] formulates the following stages of a system life-cycle: Concept, Development, Production, Utilisation, Support, and Retirement. Similar definitions are found in various SE methodologies across different domains and although stages might differ in scope depending on any specific standard or methodology, a clear distinction of activities *before* implementation begins and *after* implementation has started is found in most standards (e.g., the V-model is a very common example [35, 88]). Indeed, standard SE practices put a lot of emphasis on the precondition that starting to build/implement a System of Interest (SoI) should be only done once there exists enough confidence that it will meet stakeholder expectations and needs [88]. Therefore, the initial phases of the system life-cycle, related to stakeholders' requirements and system design, are key for the rest of the development. In fact, these activities are performed before the SoI enters the development or production stages and need to convey strong arguments that the design meets all the considered requirements.

Providing guarantees that a system will meet the stakeholders' expectations requires firstly that these expectations are correctly interpreted in the system requirements, and poor communication with a system stakeholder has direct impact on poor development results [46]. Once the requirements have been analysed and deemed correct, it is necessary for a rigorous process to design a system accordingly. It is important to note that often there exist many potential designs to satisfy a particular set of requirements, and in this case there is a need to additionally decide which design is the most suitable for a particular SoI [69]. The INCOSE handbook recommends, among other things, that during the concept stage early validation should be performed to align requirements with stakeholder expectations and for identification of problems of the concepts used in the system design [88]. As a matter of fact, discovering errors or faulty design issues during the implementation of the SoI is costly, and improper estimations of system properties at design time can lead to dramatic effort increases and even termination of the system implementation [81].

While the arguments for assuring that a design will meet the requirements are widely accepted, there is less agreement about the involved assurance processes and approaches. In fact, providing evidence that a system not yet developed will be produced and will perform correctly with some level of confidence is not simple [88]. A common strategy adopted at this stage is the re-use of previously successful processes and solutions; this strategy however comes with the risk to overlook certain viable and perhaps more attractive alternatives [1, 22]. Moreover, always re-using the same solutions will eventually lead to missing out on potential new advancements or improvements, regardless of how important and necessary re-use is from an industrial perspective.

Traditionally SE has been document-centric in its activities, however with the rise in complexity and growing needs of industry, more methods and practices are becoming model-centric, i.e., utilising models as the main artefacts during the SE phases, and in particular the stages before implementation [45, 61]. To confirm this, the INCOSE 2035 vision states: **"The future of Systems Engineering is predominantly Model-Based"**[2]. A model is any description of a system that is not the thing-in-itself [56]. The quote: "*all models are wrong, but some are useful*", from Box and Draper [10], captures a vital essence of modelling. Modelling is often not for the sake of describing something in great detail, in fact it is often the opposite, to describe a subject "good enough" with as a

[1]https://www.iso.org/standard/63711.html
[2]The INCOSE 2035 vision can be found at: https://www.incose.org/about-systems-engineering/se-vision-2035

Just Accepted

high degree of abstraction as possible in a particular context. A natural trade-off comes between abstraction and detail, and defining a model with a high level of abstraction that still contains the necessary details is a difficult task, but is often a precondition for a "useful" model [30, 52].

INCOSE defines Model-Based Systems Engineering (MBSE) as "*[...] the formalised application of modelling 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.*" [88]. Therefore, it is expected that modelling will assist the user from the first phases of the system life-cycle until the last ones, with models as the primary artefacts to support the SE activities. However, modelling at early phases of a system life-cycle comes with important challenges: by definition, capturing properties and making design decisions at an early phase is performed with a limited understanding of the system, and parts of the system might even not be understood or remain to be determined.

The behaviour of a system is one aspect that is paramount to validate and verify, but when dealing with early phases of development the lack of detail especially limits the understanding and the analytical power of modelled behaviour. Besides, the competition between abstraction and detail mentioned before becomes critical, since hidden details might convey implicit hypotheses and/or design decisions that could be contradicted in later development phases. With this in mind, it is of interest to understand how behavioural models are created, both in industry and academia, and for what validation or verification purpose the models are useful for, together with corresponding identified limitations. Therefore, in this article we contribute a systematic literature review that aims to answer various questions regarding validation and verification of system behaviour in early MBSE phases. Through the review process we identified 149 papers, which have been processed systematically and key data have been extracted to be presented in this work. We illustrate our findings from the review and discuss their implications with a specific emphasis on the industrial perspective in contrast with the academic one. Notably, from an industrial perspective we aim to elicit the kind of V&V analyses available in the state-of-the-art together with the preconditions/efforts demanded, from a modelling perspective, to adopt such analyses. Instead, from an academic point of view, we aim to identify open challenges that could be worth to investigate as future research directions.

The rest of the paper is structured as follows: Section 2 describes the background and motivation for the review. In Section 3 related work is discussed in regard to MBSE and V&V. Our research methodology is described in Section 4 in addition to our research questions. Section 5 discusses the threats to validity. In Section 6 we present our results, and in Section 7 we perform a horizontal mapping of results. We discuss our findings in Section 8 and present challenges for industrial adoption of early behaviour validation in MBSE. Finally, Section 9 summarises the paper with our conclusions, key findings, and future work.

2 BACKGROUND AND MOTIVATION

Model-based practices are often considered as an enabler for performing early V&V as part of the life-cycle of SE processes [27, 73]. In particular, the improvement of early analysis is expected from the usage of models, which enable more robust reasoning and evaluation compared to traditional document-centric development [36]. In the early phases of SE processes much emphasis is put on requirements management and conceptual design, often related to some system architecture. In this respect, MBSE is often deemed to have several benefits when compared to the traditional means of SE; some of these benefits are more intuitive and easy to identify, like added capabilities of traceability and understandability/communication from diagrams [31, 55]. Nonetheless, as the main artefacts of MBSE at all stages of development are models, it becomes interesting to understand and demonstrate the benefits and advantages of modelling activities. Notably, a key motivation often referred to as a main benefit of MBSE is the increased opportunity for V&V [17, 36]. On the one hand, when dealing with models the balance between fidelity and purpose of modelling has large implications leading to modelling trade-offs with respect to

Just Accepted

partially or fully developed models: too little details and the model cannot provide much in terms of analysis; too many details and the benefit of modelling early is reduced, as developing models often involves a significant effort [77]. On the other hand, SE practices and experience demonstrate that the cost of addressing faults and errors increases rapidly as development progresses [88], emphasising a clear incentive towards moving V&V activities earlier rather than later. When dealing with early behaviour analysis, it is often required to understand the dynamics of the system at a finer level of granularity than what is captured by standard languages like SysML[3]; these additional details can either be embedded/hard-coded in the analysis tools, or need to be explicitly provided by introducing more details in the models. In addition, most tools do not support execution of many standard MBSE languages, therefore transformations to other languages and platforms are typically required to perform behavioural analyses.

Representing and analysing system behaviour at an early stage falls into the SE modelling activities mentioned so far, both in terms of potential benefits and challenges [64]. In particular, a benefit of utilising models as the primary artefacts is the improved semantic integration of digital assets, enabling more robust early analysis towards an integrated system behaviour. Notably, the simulation of systems is seen as an essential capability of MBSE [25]. However, the current landscape lacks system simulation maturity regarding commonly used languages such as SysML [67, 94]. The lack of analytical capabilities, further noted from an industrial perspective [25, 83], hampers potential adoption. Additionally, tooling is often seen as a limiting factor regarding model-based practices, especially in industry [20, 53]. A remarkable gap in this regard is the interoperability between tools and languages: since the analysis of complex system behaviour often entails the incorporation of several domains, the lack of interoperability represents a prominent limiting factor (particularly for industrial SE processes) [66]. In this regard, there is a need to further investigate the common view of what early V&V details for system behaviour, considering the benefits and limitations of current technologies and techniques.

By summarising what we discussed so far, it is widely accepted that a move towards early V&V is an attractive endeavour and MBSE is considered as a possible way forward; however, the adoption of MBSE is not trivial, and research literature reports on several practical and fundamental challenges in the adoption [61, 83, 87]. Therefore, in this article we survey the state-of-the-art related to early V&V of systems' behaviour to elicit what are the properties of interest for the existing analyses and the corresponding proposed approaches.

2.1 Motivating example

To situate our paper more clearly, we extract a typical industrial scenario from our MBSE experience in Construction Equipment (CE) [18, 83]. We refer to the SE discipline and relate to the notion of different stages of development as defined by standards such as ISO 15268 and common processes such as the V-model. In particular, we focus on the early phases of system development, where the system under study still contains much uncertainty towards the eventual design and implementation. At this stage the system often is made-up of system *views* at high levels of abstraction. This is typically guided by standards such as ISO 42010[4] and has clear industrial definitions regarding view definitions. Here, it is worth noting that early stage is a relative concept: it could refer to a completely new system design, starting from "scratch" with customer's requirements and feasibility analysis; it could also refer to re-designing an already in-use system, perhaps a new variant of a product family (typical for domains as CE). What associates these cases is that a new idea or concept is to be evaluated and the available artefacts and information contain limited details. Nonetheless, these evaluations are important since they are used as a base to reduce the solution space, which otherwise can be considered practically endless, and hence support the engineers progress in the development. With the advent of MBSE, companies are incentivised to change their processes to incorporate models for increasing design effectiveness and improve decision-making.

[3]The SysML specification is found at: https://www.omg.org/spec/SysML
[4]The ISO 42010 standard can be found at: https://www.iso.org/standard/74393.html

Just Accepted

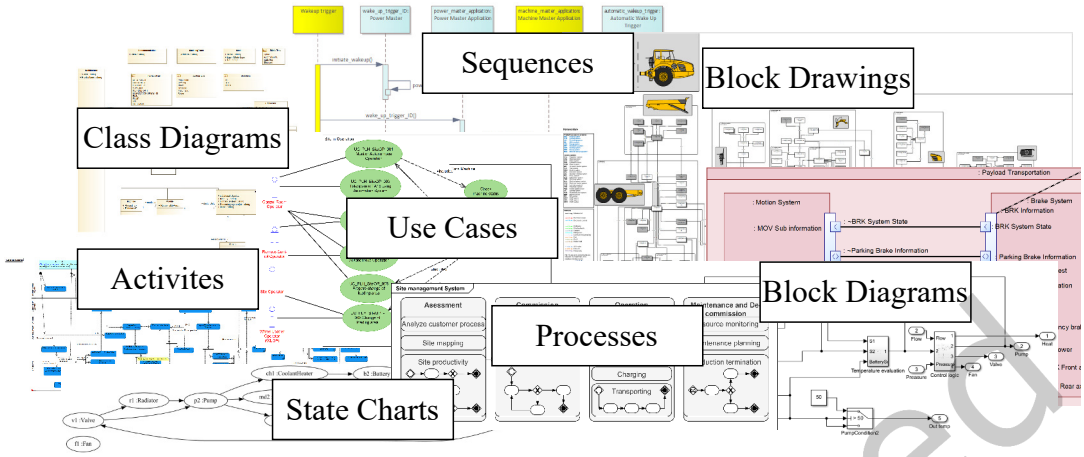


Fig. 1. Typical MBSE views early in system development.

However, changing the way of working in industrial processes is a costly endeavour, thus requiring convincing proofs that the Return of Investment (RoI) motivates the move to change.

The CE domain has a strong legacy in hardware-intensive SE, which relies heavily on re-use and product families with high variability [8]. The development processes are mature and rooted in well established standards. With the digitalisation paradigm shift, there has been a growing interest in MBSE as a critical enabling technology to manage the complexity increase. Due to the heavily integrated variability, change and configuration management techniques already rely on modular model approaches, particularly for system architectures. A product can be customised depending on stakeholder needs in the form of customer or regulatory concerns. Various drawings/diagrams are typically used in conjunction with tables to present variable options in design, and the customer is free to pick options suitable for the context. Similarly, engineers utilise various system definitions and drawings to define valid system compositions and perform feasibility/trade-off analysis toward customer demands. In this respect, the traditional SE development already includes system model views of various types. Some examples are provided in Figure 1 from the CE domain at its early stages (architecture and high-level design). Although we do not discuss any of the views in detail, we emphasise the wide range of artefacts available as a result of several parallel/joint engineering activities. Despite many of these system views being models, it is worth noting that often they are not linked meaningfully and/or they rely on informal semantics (e.g., Visio drawings), which limits model-based analysis "as is".

In these early stages it is often the case that *what if* analysis or high-level trade-offs are required to make high-impact early design decisions. A common example to be considered even at the very start of a design process is a machine's brake functionality, as design guarantees are required as per industrial standards for several aspects (e.g., maximum brake distance, fault tolerance, environmental robustness, etc.). In this context, early V&V of both functional and non-functional requirements could front-load activities to speed-up the decisions related to high-level design and reduce risk of extensive iterations on design. However, providing guarantees that a system will meet strict requirements demands strong confidence, and in early phases the high-degree of uncertainty makes that challenging, and often pessimistic assumptions are used to make "safe" estimations for the design. With the considered legacy as a starting point, MBSE is being gradually adopted by the CE domain. In particular, taking the already-in-place model views and further leveraging them for model-based analytical capabilities currently not in place could increase competitiveness by improving the ability of early *valid* analysis. Notably, the

Just Accepted

behavioural aspects of systems become easier to analyse via rigorous methods compared to legacy SE methods that rely on semi - or non-formal models and often implicit expert knowledge. By leveraging the traditional SE views with MBSE technologies, design decisions could be made earlier than traditional workflows. The value proposition from early V&V relates to the difference from traditional decision-making, for example, via the time required to reach decision maturity or analysis coverage with system descriptions. Practically, information to make informed decisions about system viability can become available earlier by leveraging model-based methods for re-use, analysis, simulation, etc. However, as anticipated earlier in this article, MBSE adoption is an industrial challenge. In our experience within CE, model-based methods generally map poorly to the overall SE context and are overly complex and/or abstract. The value proposition is also challenging to be demonstrated in practice, leading to "convincing" stakeholders of the potential value. Additionally, MBSE is associated with tooling and extensive training to change the way of working.

By considering industrial contexts like the CE domain, this article and subsequent research questions aim to provide a better understanding of the literature landscape for early V&V of system behaviour. We aim to elicit and disseminate the current research results and industrial readiness for early behaviour validation utilising MBSE. Notably, we want to understand how it is defined and motivated, how it is implemented, what tools, languages, and methods are used, what aim the authors have in mind with early V&V, and finally, observed limitations. In this way, the review can serve the necessary information due to a step forward for MBSE adoption of SE disciplines already considering models as a part of development. Moreover, it can highlight potential challenges to be addressed in the wider community. In particular, the review results would clarify what types of early V&V behaviour analysis can be expected in MBSE, what is the required effort in terms of modelling activities, what are the methods, languages, and tools involved. In other words, this review aims to elicit the distance between currently used early stage artefacts as the ones described in Figure 1, and the models necessary to enable MBSE early analysis techniques. Furthermore, we aim to shed the light on open research challenges and possible future investigation directions, especially to close the gap between research and practice.

3 RELATED WORK

Although we have found no other survey or literature review regarding the subject described and reported in this article, several other works address similar or related issues. Therefore, in this Section we highlight other reviews discussing related aspects of MBSE.

Ma *et al.* [60] aimed to understand the state-of-the-art and state of practice for the tool chains used for MBSE. They define a tool chain as two or more modelling, simulation, and design tools which when combined support/construct SE workflows with advanced features. The review identifies that SysML is the most adopted modelling language for MBSE tool chains. The authors note that although tool chains based on SysML are the most mature, there are still major challenges for robust industrial adoption. Furthermore, the authors highlight some primary indicators of tool readiness, namely integration capabilities, interoperability, and traceability. While Ma *et al.* discuss some aspects of MBSE that are related to this review, there are significant differences in scope, depth, and context. Notably, we emphasise the notion of early phases in MBSE, and focus our analysis around system behaviour analysis while their work does not necessarily consider system behaviour analysis. In addition, we emphasise an industrial perspective through a motivational example and present a deeper analysis from that context in addition to adoption barriers for early V&V.

Another study from Rashid *et al.* [73] investigated the tools used for MBSE activities within the embedded systems domain. Similar to other reviews, they identify that UML and the profiles SysML and MARTE are the most commonly utilised means of modelling. They also note that UML profiles or UML alone do not meet the existing modelling challenges, and some combination of languages is often employed. On the contrary, the authors note that SysML provides a sufficient foundation to model structure and behaviour. The work by Rashid *et al.* focuses

Just Accepted

on embedded systems while the work in this paper is not tailored for a particular domain. Additionally their paper is situated more towards code implementation as opposed to early V&V, so to this regard the observations in the reviews capture different aspects of MBSE.

De Saqui-Sannes *et al.* [25] provide a taxonomy of MBSE approaches in the scope of languages, tools, and methods. In the review, they note the prominence of SysML as a MBSE language, with an assortment of tools to support SysML. However, the authors note that robust methods of MBSE are still lacking. In addition, they argue that many challenges still exist for the industrial adoption of MBSE and that SE education needs to capture the current reality of MBSE better. The work by De Saqui-Sannes *et al.* has a different focus compared to this work as it aims to give a brief overview of the field. Further, the work discusses experiences from the authors' experience with a drone project. Comparatively, our work has a narrower view on the behaviour of systems but provides a more complete review in terms of literature coverage and corresponding results.

Nigischer *et al.* [66] provide a systematic review on multi-domain simulation utilising SysML. In their review, they argue that MBSE provides support for analysis at early stages, utilising SysML to capture information that can be exchanged with suitable simulation tools. The review discusses various means of managing simulation via SysML and notes that the Functional Mock-up Interface (FMI)[5] is a promising standard. However, the authors conclude that there are still many issues with SysML-based simulation, and a particular challenge is interoperability between tools. While the review by Nigischer *et al.* covers aspects present in this review, it focuses on SysML and simulation, which can be considered as a subset of early V&V of system behaviour.

Zeigler *et al.* [94] argue that simulation is an essential capability for MBSE toolsets, which they identify as lacking in the current landscape. Their paper states that MBSE practices need to be expanded to manage more complex systems engineering practices, especially when dealing with System of Systems (SoS). The authors identify that most of the efforts in MBSE regard implementations in notations such as UML and SysML. However, such notations are limited regarding simulation capabilities, which the authors argue could raise questions regarding the adequacy of those notations for developing complex systems. The review by Zeigler *et al.* provides arguments and insight into simulation, instead the work presented in this paper aims to discuss and review concepts related to early V&V. Although simulation is a commonly used technique in early phases of systems engineering, we are not focusing on these specific techniques and their applicability. Further, the distinct focus on SoS is not present in this review.

Henderson and Salado [41] review the reported benefits and value of MBSE practices from the existing literature. The primary finding of the authors is that most of the argued benefits in literature are expected and not measured, leading to their conclusion that MBSE benefits remain inconclusive. Li *et al.* [58] also identify that although MBSE-affiliated research is growing, several independent research clusters exist with little interaction. These works discuss topics also reflected in our review, however our focus is different. Firstly our work has a strict focus on early V&V in the context of MBSE, and while the other reviews also regard MBSE the focus is broader. Secondly, our work discusses similar topics, but it is part of the work and not the main focus of the work, creating a more holistic view for the aforementioned area of early V&V. And lastly, the aim of the reviews are different, we catalogue and review aspects of interest for early V&V while Henderson and Salado call for more empirical studies in MBSE and Li *et al.* aims to promote collaboration across existing domains to address future concerns.

Laing *et al.* perform a survey of industrial MBSE practitioners in France regarding model-based verification [53]. They identify several success criteria that if met are believed to lead to positive effects in the adoption of MBSE. They also note two major weaknesses from an industrial perspective for model-based verification, namely for verifying the system architectures and how multi-physics or multi-disciplinary designs can be integrated in MBSE frameworks. While the review by Laing *et al.* focus on verification in MBSE, they catalogue industrial

[5]https://fmi-standard.org/

Just Accepted

views on the subject and present success criteria, while the review presented in this article instead focuses on the V&V in itself via a systematic study.

Arauju *et al.* [5] perform a systematic literature review on testing, verification, and validation of robotic and autonomous systems (RAS). Their review indicates a growing need for extending traditional means of testing and V&V for complex RAS systems. Similar to our work their review also focused on the industrial perspective, targeting both industry and academia and finding a lack of industrial applications of methods and tools. While the review discusses similar concepts to this work, its focus is narrower and systems descriptions are not expected to be given by means of low-fidelity models (as investigated in this review). In addition the focus of their review is closely related to the technical aspects, while our work also discusses MBSE from a more holistic view.

Ahmad *et al.* perform a survey on model-based testing utilising UML activity diagrams [2]. A few results of the review are highlighted, namely the lack of non-functional testing, lack of industrial or elaborate evaluation, high representation of domain specific solutions with tight restrictions, and the lack of holistic approaches. While the review shares some common aspects with the work presented in this article, it focuses on UML and model-based testing specifically as opposed to the broader and differently positioned review presented here. Further, the work by Ahmad *et al.* does not consider the context of MBSE.

Chaudemar and De Saqui-Sannes [21] investigated the combination of Multidisciplinary Design Analysis and Optimisation (MDAO) and MBSE for early validation of design. The authors argue that MDAO could be a good fit with MBSE as it, among other things, can be integrated with low-fidelity models and take model uncertainty into account. However, the authors note that these methodologies remain mostly separated in the literature, and some challenges must be addressed to join them. While our work discusses many concepts related to design, we do not have the same strict focus as Chaudemar and De Saqui-Sannes about early validation. Further, this review considers early behaviour validation independently of its coupling with MDAO.

Tsioptsias *et al.* [86] investigated the simulation model validation and testing via a literature review. Three distinct fields of research are observed: Operational research, Modelling & Simulation, and Computer Science. Some of their main findings include the distinct lack of common terminology for the reviewed concepts, a lack of linkage between theory and practice, and insufficient empirical studies to support the claims in papers. The authors also argue that validation should be performed continuously and that modellers and users should work closely during simulation model validation. Although discussing models, the review by Tsioptsias *et al.* focuses on a different scope compared to the review presented in this article. Their focus is on simulation model validation, while this review is centred around V&V in the context of MBSE. While simulation model validity is central to V&V, it can be considered a sub-problem of (early) V&V as a whole and our review presents a more broad discussion.

4 RESEARCH METHOD

This section presents the research methodology used to conduct the survey. We followed the steps described by Kitchenham for a systematic literature review [51]. In particular, the research method included three distinct phases, *planning*, *performing*, and *reporting*.

The purpose of the planning phase included: i) the identification of gaps in the literature and needs for the review, discussed in Section 2 and 3; ii) the definition of the research questions to drive the work, presented in Section 4.1; iii) the definition of the review process and guidelines for the involved authors, illustrated in the remaining of this section.

During the performing phase, we executed the review in several concrete steps, namely *Search*, *Selection*, *Snowball*, *Definition of data collection table*, *Data extraction*, and *Data analysis*. The search step consisted of defining a search string and a consequent automated search for relevant papers through several scientific databases. The selection consisted of a rigorous process for identifying primary studies for the review. We complemented

Just Accepted

the identified papers via an exhaustive snowballing process [91] to identify potentially missing papers. At this point, a data collection table was constructed and validated on a few pilot papers. Finally, we performed the data extraction on the included papers and coded data for easier interpretation. We analysed the extracted data vertically and horizontally, resulting in the findings of this review.

In the reporting phase, we documented the findings resulting from the review (see Section 6). Further, we analysed potential threats to validity and corresponding mitigation strategies to be employed (see Section 5).

4.1 Research questions

This study aims to investigate the current methods and practices for describing and analysing behaviour by means of system models in early stages of development when adopting MBSE. We are also interested in inspecting model-based techniques from an academic and industrial perspective, highlighting the current similarities and differences with respect to adopted methods and tools. With this and the motivating scenario illustrated in Section 2.1 in mind, we formulated the following research questions (RQs) to drive the work:

RQ1: *How is early V&V defined and motivated in the MBSE literature?* This question investigates the definition of early V&V activities in the literature together with the main motivations reported for performing those activities.

RQ2: *What are the means for describing system behaviour at an early stage of development?* By considering the trade-off between purpose of modelling and fidelity of the models, it is of interest to understand how a system behaviour is initially described. Moreover, it is relevant to capture the languages and formalisms utilised for analytical purposes, since they might differ from the initial behavioural descriptions. Eventually, in the cases where different models are used due to behaviour description and analysis, it is of interest to elicit the types of approaches adopted to map the different representations.

RQ3: *What are the results of interest for the early V&V, and what techniques are employed for performing the analysis?* Given the early stage of the development, it is relevant to understand what type of analysis results are reported in the literature. Similarly, it is important to elicit what methods or techniques are suitable for computing the analysis results, and to understand how these results are presented to the user.

RQ4: *Which are the application domains employing early V&V?* By considering the trade-off between modelling efforts and reliability of the analysis results, it is interesting to know which application domains adopt the proposed solutions and whether these solutions are domain-specific or not. Additionally, it is important to report whether the solutions have been validated in an industrial setting or not, aiming to elicit potential gaps between academia and industry.

RQ5: *What are the limitations of the existing approaches for early V&V?* By considering the growth in complexity of the developed systems and the impacts of problems discovered late in the development process, it is critical to understand what limitations are reported when performing early V&V, both specific for the proposed solution and more broadly for early V&V in general.

4.2 Search-process

To find the papers included in the study we opted for an automatic search across several scientific databases. By operating some preliminary exploration of the databases of interest we noticed a limited number of hits on relevant publications; moreover, based on our own experience we expected a relevant spread of keywords and definitions due to MBSE being broad in nature. For example, the notion of validation and verification will differ between several domains, and the "MBSE" keyword is used in several orthogonal disciplines. Therefore, we kept the search strings relatively tight and decided to perform an exhaustive snowballing to ensure the search process

Just Accepted

would capture as many relevant papers as possible for the study. The following databases were searched for information: ACM[6], IEEE[7], ScienceDirect[8], and Scopus[9]. Moreover, we used the following search string:

("MBSE" OR "Model-based systems engineering" OR "Model based systems engineering") AND ("Validation" OR "Verification" OR "V&V" OR "Evaluation") AND ("Behavior" OR "Behaviour").

4.3 Inclusion process with inclusion and exclusion criteria

Starting from a set of initial papers collected via the search strings across the chosen databases, we selected additional papers through an iterative process until a final set of papers was identified. The process was guided by well-defined inclusion criteria (IC) and exclusion criteria (EC), summarised as:

IC1: The paper regards the problem of model-based early V&V of system behaviour
IC2: The paper presents one or more concrete solutions for early V&V
EC1: Not in English
EC2: Not Peer-reviewed
EC3: Scope outside of model-based systems engineering
EC4: Not available in full text
EC5: Short papers, tutorials, WiPs, research agendas, papers shorter than 5 pages
EC6: Paper overlaps with a more complete paper (e.g., a conference publication extended by a journal article).

IC1 and IC2 are the main criteria we consider for a paper to be included in the review, and each included paper meets both criteria. If any of the EC are met, regardless of the IC, a publication is automatically not qualified for the review process. EC1, EC2, and EC4 remove papers not meeting the basic criteria. EC3 removes papers that do not discuss the topic of early validation in the MBSE context, such as papers discussing non-model-based approaches or parallel domains such as software engineering (also referred to as MBSE). With EC5, we aim to remove any work that does not present a complete research or application paper. Furthermore, with EC6 we aim to avoid bias by including the same general source with minor editions, and in the case of overlap, the more mature paper is included.

Figure 2 provides an overview of how the number of papers changed over the process of applying the IC/EC to arrive at the final set of 149 papers. The original search was conducted the 15th of March 2022, and the additional search was conducted the 31st of March 2023.

The steps taken in the search process and shown in Figure 2 were performed in a review management tool called Covidence[10]. First, the database search was performed, resulting in 495 papers, which became 431 after duplicate removal. Then, the IC and EC were applied to the title and abstract of each paper, resulting in 179 papers to be included for the full text review. We note a drastic decrease of the papers at this stage and ascribe much of that to catching papers not focused on system behaviour; in fact, the "MBSE" search-term caught many unrelated papers. In the full text review each paper was read again with the IC and EC in mind, and finally 69 papers were chosen as the set of primary papers. Subsequently, as planned at the search string definition time, an exhaustive snowballing process took place following the guidelines in [91]. The snowballing process took 8 rounds until no new papers were found, resulting in 206 newly identified papers for the review process and eventually 152 included papers in total. Again, we note a large increase in the papers after the snowballing process, in part due

[6]The ACM databse can be found at: https://www.acm.org/
[7]The IEEE database can be found at: https://www.ieee.org/
[8]The ScienceDirect database can be found at: https://www.sciencedirect.com/
[9]The Scopus database can be found at: https://www.scopus.com
[10]The Covidence tool is available at: https://www.covidence.org/

Just Accepted

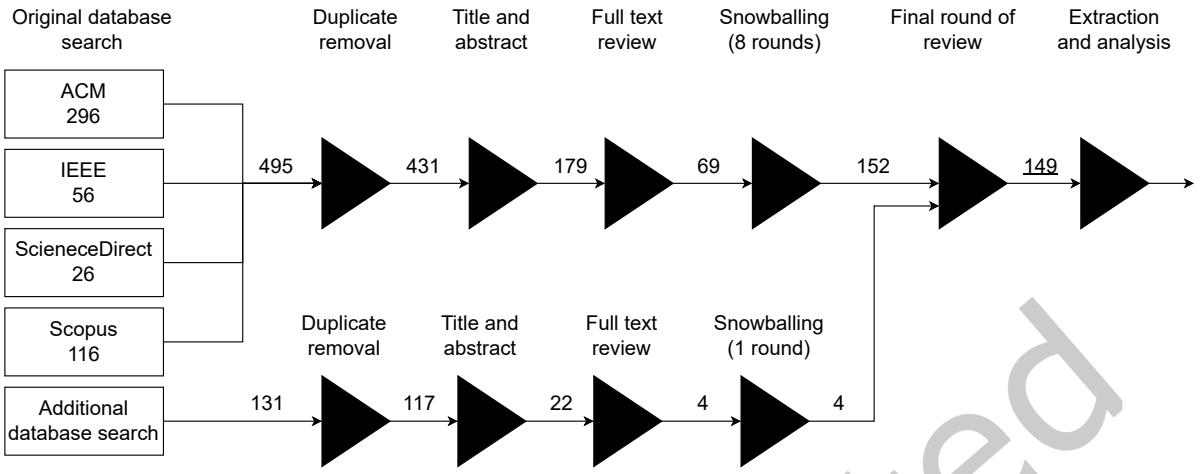


Fig. 2. The process for paper identification and corresponding number of papers at each step

to its exhaustive nature, but partly due the initial search string missing some key papers because of the adoption of slightly different terminology. Authors would refer to the notion of early in a process differently, for example, "early stage", "early phase", simply "early", and more commonly for our snowballing, not explicitly in the title itself. It is also worth noting that the snowballing phase followed the same screening procedure adopted for the initial set of papers (for the sake of readability we omit the cycles due to the snowballing in the picture).

The final round of review concerned removal of overlapping papers (e.g, a conference paper and a journal paper with same introduction and background sections) and a final check for the IC and EC, resulting in papers that were finally considered for data extraction. As visible in the figure, given the extent of this work we decided to perform a search update at this stage to catch any new papers published during the review and writing process. Such an update resulted in new entries to be considered, the numbers of which are detailed correspondingly in the figure (starting from "Additional database search"). Due to the aforementioned removal of overlapping papers the total number of publications was reduced from (152+4) to 149.

When applying the IC and EC, two reviewers were assigned to each paper, and in case of disagreement a third reviewer would make a final decision. In the snowballing phase, a single reviewer identified the potential papers, while the selection process followed the procedure mentioned before. With a final set of 149 papers, the extracted data was analysed vertically and horizontally for presentation in this article.

4.4 Data collection and analysis

Once a set of studies has been identified, the relevant data has been extracted as shown in Table 1. By going into more details, Question 1 targets general publication details. Questions 2 and 3 aim to answer RQ1 by extracting the authors' definition of early V&V and specified motivations, respectively. Questions 4, 5, and 6 target RQ2 by extracting the languages and formalisms used for description and analysis of behaviour. It is worth to notice that we differentiate language and formalism based on an existing classification[11]. RQ3 is answered by Questions 7, 8, and 9 which target the techniques, results of interest, and tools utilised in the process of analysis. To answer RQ4, Questions 10 and 11 extract more information about the domain of application considered in the paper; moreover, Question 12 targets how the solution was validated based on the classification by Shaw [79]. As part

[11]Multi paradigm modelling classification of languages and formalisms: https://zenodo.org/record/2538711#.Y0etVExBxaR

Just Accepted

Table 1. Data collection table

No	Description	Expected input	Focus
1	Publication details	Title, Venue, authors, year	-
2	How do the authors define early V&V	Phase and definition	RQ1
3	Motivation for performing early V&V	Background and goal for solution	RQ1
4	Behaviour description	Languages and formalisms	RQ2
5	Behaviour analysis	Languages and formalisms	RQ2
6	If 4 and 5 differ, how is that managed	Manually, semi-automatic or automatic	RQ2
7	How is V&V performed	Technique or approach for early V&V	RQ3
8	What result of V&V is of interest	Results from analysis	RQ3
9	What tool(s) are used for V&V	List of tools for analysis	RQ3
10	What is the modelled domain	From examples in paper	RQ4
11	Is the V&V solution domain specific	Yes, no, partly	RQ4
12	How is the method validated	Examples, empirical measurements	RQ4
13	Limitations observed for early V&V	Authors' reported limitations	RQ5

of RQ4 we also extract whether the study evaluates the solution in an industrial context or not based on the case study description (or lack thereof) in the paper. Finally RQ5 is answered by Question 13, which extracts the limitations of the presented solutions as explicitly discussed in each paper.

Similarly to the previous steps, the Covidence tool was used to perform the data extraction. In particular, Covidence allows for automatic identification of conflicts in data extraction. This feature was utilised on a set of pilot studies to verify whether two researchers would extract the same data and hence elicit possible issues with the extraction form. After performing the data extraction on 10 (random) pilot papers and comparing the results for inconsistencies, only minor issues were identified, which led to a minor refinement of the guidelines for the data extraction to better match the expected outcome. Subsequently, one researcher took charge of the data extraction for all the papers. Nonetheless, after the completion of the extraction task another researcher performed a round of random checks on the extracted data. This round included half of the extracted papers and no major inconsistencies were discovered.

Based on the data collected according to Table 1 we performed vertical and horizontal analyses, the results of which are reported in Section 6. Vertical analysis refers to the deeper discussion of each particular RQ and corresponding data extraction. Horizontal analysis instead focuses on cross-data patterns and correlations. In this respect, it is important to notice that additional coding has been adopted on certain categories to perform the horizontal analysis (see Section 7). Examples of such coding could be whether papers used SysML or not, the type of licensing for tools, and categories for limitations. The coding has been especially helpful for the categories where input varied greatly. In those cases, almost all entries would appear once or twice at most, thus making the extracted information spread across a large set of data. For the interested reader, we refer to the publicly available review replication package, which includes the review protocol, the set of collected papers, the ones selected for the extraction, the complete set of extracted data, and the adopted coding for specific subsets[12].

5 THREATS TO VALIDITY

The review presented in this article has been performed according to well-established research guidelines. Moreover, a research protocol with correlated data is located in a publicly available replication package. Still,

[12]The replication package is openly available at: https://github.com/BeeCub3/Replicate-package

Just Accepted

we acknowledge that a study of this magnitude and scope might contain some threats to the overall validity. Therefore, in the following we discuss the potential threats by adopting the terms by Wohlin *et al.* [92] together with the corresponding countermeasures we have considered.

5.1 Data reproduceability

For the sake of readability and conciseness, we summarise the results and highlight selected peculiar outcomes. A complete list of the included papers and the corresponding data extraction, along with other technical details related to the systematic review process, can be found in a publicly available repository both for replication and to use this data source for other kinds of analyses and investigations. The replication package consists of the search strings used, in addition to a table that places unique publications on the rows and the extracted data in the columns. Some columns have been created explicitly for horizontal analysis, presented later in the paper, detailing for example whether a publication uses SysML or what type of tool licenses exist for the tools used in the publications. Furthermore, we have shared the tables and graphs used for the various analysis.

5.2 External validity

The threats of external validity primarily relate to the retrieval of the papers to be analysed. In fact, the selected papers are the source of all the analyses and significantly impact the extracted results and their quality. We utilised the search string in several databases to identify potential studies for our analysis. We performed the search with a relatively refined search string to reduce the initial set of papers. In order to mitigate the risks of missing relevant papers, e.g. due to a missing explicit reference to MBSE or V&V, an exhaustive snowballing (forward and backward) procedure was performed until no new papers were identified. The snowballing was performed in eight rounds, which allowed us to capture the papers missed by the initial search string and to accurately retrieve the relevant papers for analysis. Additionally, we re-iterated the search process at a later stage of this research work to catch any papers released during the time of data analysis and of writing.

5.3 Internal validity

Internal validity refers to any threats primarily arising from the bias of the reviewers involved in the study. To mitigate the bias of individual researchers, we required a majority consensus of the reviewers for all the selection steps until the data extraction. In other words, two researchers performed the selection in parallel and in case of decision conflicts a third researcher took the final decision. Moreover, before performing the data extraction, several pilot extractions were performed to evaluate and harmonise how different reviewers would interpret the extraction forms. The pilot extractions did not reveal any significant conflicts, which gave us confidence that an individual reviewer could perform the extraction. Nonetheless, at the end of the data extraction a sanity check was performed on a significant subset of random papers to verify the similarity of the results.

To reduce the bias even further, the researchers were not tasked with evaluating any of the papers' claims. Instead, the extraction consisted of reporting the claims by the authors of the papers. As such, any interpretation of the data was left for after the extraction. For the entire process we utilised the Covidence tool for maintaining consistency among reviews as the tool highlights potential miss-matches of extraction. After extraction was complete we exported all data from the tool and performed the horizontal analysis and vertical analysis directly on the data.

5.4 Construct validity

Construct validity mainly relates to the risk of deriving an incorrect conclusion from the relations between treatment and outcome. For this article, this would mean that the way we searched and selected the papers and the approach adopted for the extraction could have affected the results we obtained. We used several literature

Just Accepted

sources for the search string to mitigate this risk, 4 to be exact. Moreover, we performed exhaustive snowballing to mitigate the threats of poorly formulated search strings or missing papers due to exclusion from databases. The snowballing was done according to the best practices recommended in literature and was performed exhaustively, i.e. until no new papers were found.

5.5 Conclusion validity

Threats to conclusion validity refer to any risk of misinterpreting the results of the findings. To mitigate these risks, we have followed well-established systematic literature review approaches [51, 92]. Moreover, we did not adopt any preliminary interpretation of papers' contents, and we applied automated analysis tools to collect data and elicit relevant cross-relations.

Admittedly, there could be still some risk for bias due to our close experience with the CE domain, and hence a potentially limited/incomplete interpretation of the extracted results. Nonetheless, our experiences in other industrial domains point out very similar MBSE adoptions scenarios and issues, making us confident about the broader validity of our reasoning. Additionally, we provide a publicly accessible replication package with all the details about how the study was performed and the corresponding extracted data.

6 FINDINGS

In this section, we present the findings of the data extraction while targeting the research questions formulated in Section 4. We refer to the online replication package to see the precise extraction of data from publications. An appendix is attached in Appendix A for more explicitly mapping the papers to the RQ categories, which are presented and summarised in the following section.

6.1 Publication details

In this section we present the overarching publication details. First, we map the trend of publications over the years, as illustrated in Figure 3.

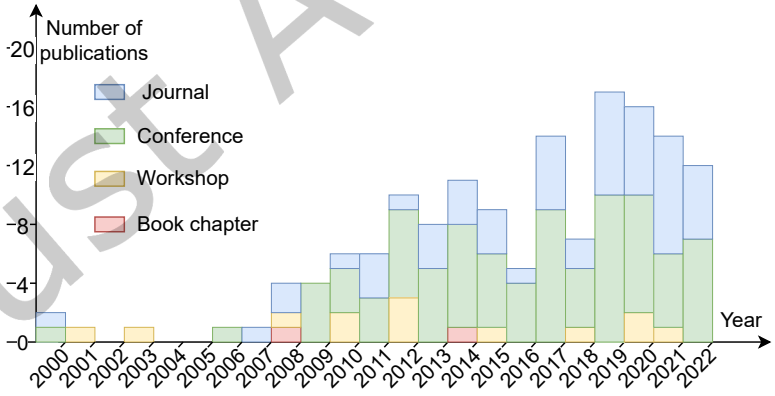


Fig. 3. The distribution of the analysed papers as clustered by year and type of publication.

Figure 3 indicates a rise in the interest for the topics discussed by this review over the years. Moreover, even if no condition for inclusion was set on the earliest year, the earliest publications can be found in 2000 and few publications are found before 2008. Since then, the general trend shows an increase in the number of publications, hinting a growing interest in the topic. Such a trend also matches quite well the birth and maturation of MDE

Just Accepted

techniques, which are a key enabler for early V&V [78]. In the figure we also distinguish the selected papers by the publication types and notice a large majority of either conference or journal publications. Such prevalence might indicate a certain degree of complexity and/or maturity for the proposed solutions that are difficult to enclose in workshop publications.

Figure 4 presents a word cloud of the publication keywords, where each keyword is included regardless of the number of occurrences, and no keyword clustering has been performed. Ignoring the references to MBSE or modelling in general in Figure 4, the most common keywords in the publications are SysML (n=30), Simulation (n=20), Verification (n=16), UML (n=10), Model checking (n=10), Requirements (n=9), and Model transformations (n=9). We note a clear bias for SysML and related concepts, as well as for simulation and related tools, languages, and techniques. We also highlight a lack of mentioning for the word "design" or similar concepts. Apart from the more represented topics in the keywords, many are represented only once or twice, often related to specific techniques or domain-specific concepts.

6.2 RQ1 - How is early validation and verification motivated and defined in the literature for MBSE?

This section summarises the data extracted to answer RQ1, specifically questions 2 and 3 in Table 1.

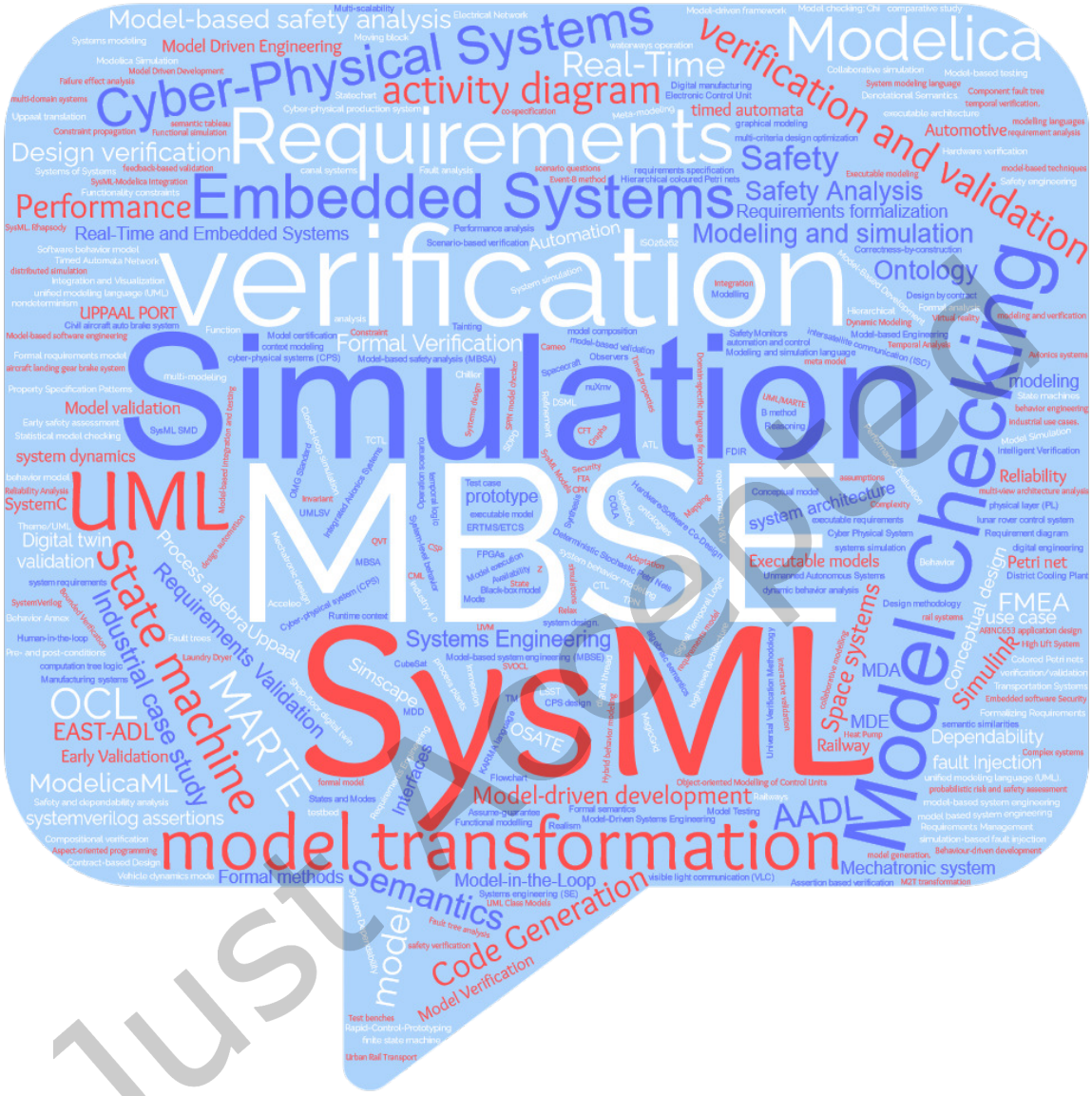
6.2.1 How does the community define early V&V?. Although all the analysed papers regard the concept of V&V at an early stage of development, few papers explicitly state what that means in their context. Instead, most of the papers implicitly infer that the targeted V&V takes place at some point of system design or requirements elicitation, often referring to the INCOSE definition that states that verification and validation begin in the conceptual design phase [88]. Out of the papers explicitly describing the phase or context of early V&V, a majority refers to the design phase and, as previously stated, often refers to the INCOSE definition of the "Conceptual design phase". Apart from the design phase, some papers argue that early V&V targets the "requirements phase", and some authors report that their solution targets both the requirements and design phases. Figure 5 visualises the target phase of the solutions among the papers.

In more detail, 107 of the 149 publications (71.8%) report that the proposed solution applies for the design phase, while 29 (19.5%) report that the solution exists in the requirements phase. 13 (8.7%) of the papers report that their solution covers both requirements and design and tend to be large in scope. As an example, Lemazurier *et al.* [P28] use requirement boiler-plates as a starting point to leverage five unique domain-specific languages and several views to generate a functional architecture. Bouffaron *et al.* [P44] instead define an iterative process of system refinement that spans several stages of development. In both cases there is a large emphasis on the process, and the solutions play a supporting technical role.

From the extracted data we can see that authors consider system requirements and system design as both targets for early system behaviour V&V. While few authors explicitly define early V&V, it is clear that the majority of them considers the topic of early validation to regard system design rather than system requirements. Moreover, the problem of "will a particular design meet the requirements" prevails over "will a particular set of requirements capture the system of interest adequately". Eventually, when papers deal with both requirements and design phases they use the available information to perform cross-checks, notably for better understanding the system and reason about constraints [P4, P48, P149]; to raise the quality of models and reduce "bad smells" [P15, P28, P53, P95]; to improve traceability and understand artefact dependencies [P28]; to validate non-functional requirements [P44, P60, P96]; to reason about trade-offs or to reduce the design space [P51, P63].

6.2.2 What are the main motivating reasons for doing early V&V?. To understand the motivations for doing early V&V, we extracted from each paper the reason(s) the authors use to motivate their activities. The exact extractions are found in the replication package, while Figure 6 summarises the findings. In particular, it displays all the

Just Accepted



Just Accepted

Fig. 4. Word cloud of the keywords in the included papers of the review.

motivations mentioned at least twice in the papers, while those mentioned only once are valued as “Other” in the figure.

The reasons listed in Figure 6 can partially overlap and many papers report multiple reasons for performing V&V activities. The most common motivation for performing early V&V is to ensure a desired level of quality for the design before proceeding with the implementation. This is followed by reducing risks with late flaw

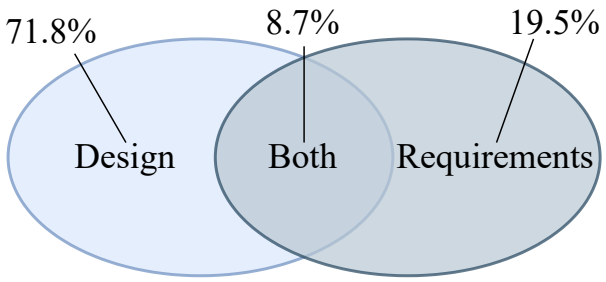
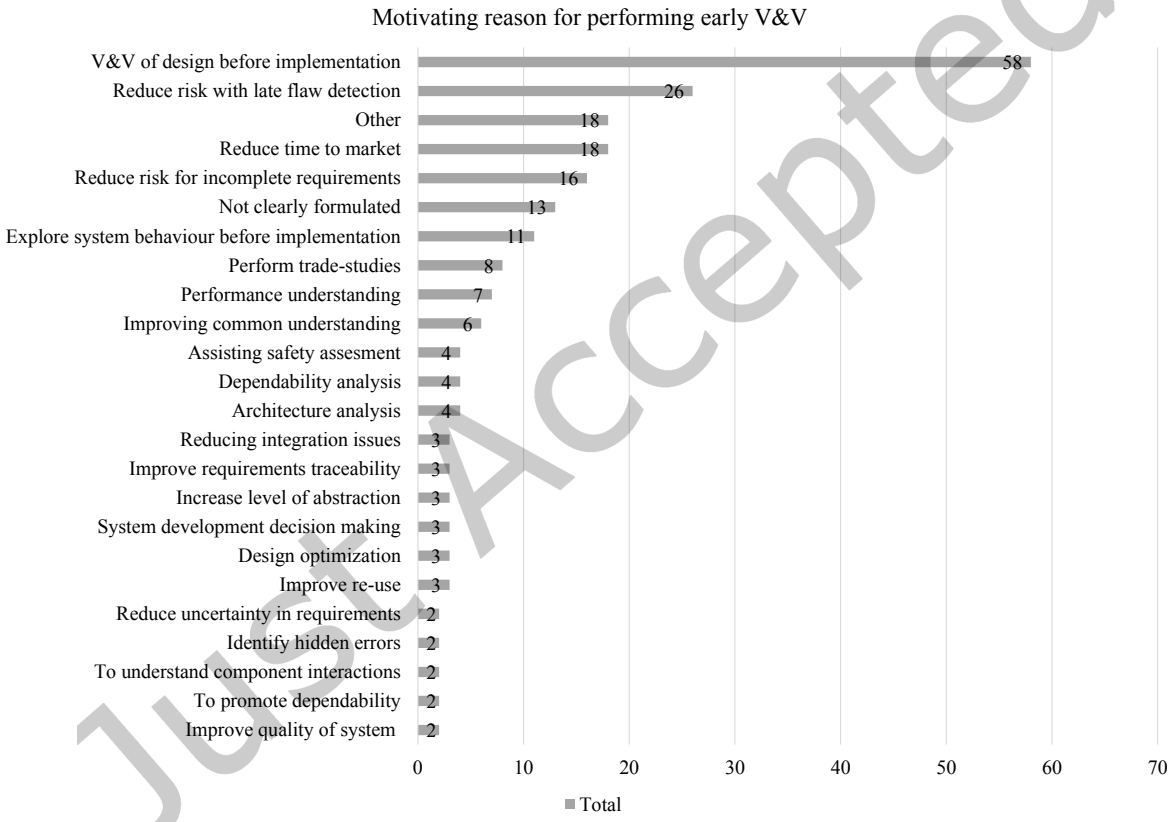


Fig. 5. The phase targeted by the solutions as reported in the included papers.



Just Accepted

Fig. 6. The reported motivating reasons for authors to perform early V&V of system behaviour using model-based practices.

detection, reducing time to market, reducing risk for incomplete requirements, and exploring system behaviour before implementation. In this respect, the predominant motivations seem to directly target the reduction of risks associated with introducing errors or creating incomplete specifications in the early phases, which should lead to a more streamlined process. This is also confirmed by the fact that reducing time to market is an often

quoted motivation for the activities performed by authors. Other potential risks to be prevented are those caused by sub-optimal (or even wrong) design decisions that could impact critical quality attributes of the system. In this respect, several of the more frequently mentioned reasons are related to performances and dependability (safety in particular).

A set of papers does not clearly motivate the reasons for employing their solutions targeting early V&V; we note that many of these papers describe more theoretical works without any real target case study, which is perhaps why there is a lack of motivation for the activities. For example, Kahani and Cordy discuss bounded verification for state machines in a train system controller [P40]. However, they do not discuss their solution motivation at length as they deem the concepts already well established. Similarly, Liu *et al.* [P138] discuss Assume-Guarantee Reasoning in the context of scheduled components and present a general theory instead without a concrete motivation. In other words, these research works deal with techniques that might support early V&V solutions but do not explicitly mention concrete usage scenarios.

RQ1 discussion: To summarise, while there is some spread in the extracted motivating reasons, there seems to be a consensus about V&V in the earlier phases of development as being key to minimising development costs and reducing risks with flaws later in the development life-cycle. Although the main category of early V&V motivation can be formulated as "anticipating V&V before implementation" and is perhaps what is to be expected, it does not even map to half the papers[13]. Apart from the more expected results, commonly reported motivations include "reducing risks associated with faulty design or requirements", "performing trade-off studies", and "improving communication and integration between system aspects". Most of the reported motivations for V&V are well in line with SE state of the art and practices [88], in which MBSE is seen as means of going earlier with the involved activities while maintaining the necessary rigour. Additionally, the authors mostly regard their solutions as situated in the design phase, considering the requirements phase to a smaller extent. Moreover, phases past design are not represented to any significant extent in the paper solutions. The extracted data indicate that MBSE is reaching maturity for using methods of analysis in system design with significant benefits and it may also signify that the requirements phase has a lower need for added capabilities to current methods. On the other hand, the lower amount of papers targeting the requirements phase could also highlight that solutions need to be more mature, something reflected in the lower representation of industrial cases in the selected papers targeting the requirements phase. We also observe that few papers seem to target both the requirements and design phases, probably due to the issues related to connecting different artefacts (and corresponding tools), often reported as a weak point for MBSE [41].

Perhaps, more interesting is what is not reported by the authors to any significant extent, notably aspects such as re-use (of the V&V artefacts for other phases of the development process as well as for other projects) and improved traceability between development stages. This is surprising, since the mentioned aspects are often argued to be strong points for MBSE adoption [41]. In the same fashion, there is little discussion on the broader integration of the V&V activities in the SE landscape, which is argued as a benefit of MBSE. Notably, there are few mentions of the digital thread [80], which can be enabled with model-based methods. In conjunction with this, few papers discuss cross-domain integration, which also is something to be expected when leveraging abstraction. In essence the solutions seem to overall lack a holistic motivation, that is how early V&V supports parallel and down-stream activities.

Actionable Insights RQ1

[13]Here we remark that to the reduce the risk of introducing bias by means of subjective interpretations we only extracted the explicit information as written by the authors.

Just Accepted

From the analysis performed we believe the following action point to be valuable for the community in the context of RQ1:

- Motivations for V&V stakeholders seem to be somewhat unclear or implicit. Seemingly, there is no clear view of the expected value in performing early V&V.
- Processes typically span several development stages, seen through the divide between requirements and design; yet, there is little discussion on holistic approaches or possible benefits expected by models to bridge stages.
- Artefacts should be used by several domains and users during development, yet there is little discussion on the interoperability or re-use of V&V artefacts.
- There is a broad range of definitions (and many papers make no attempt) for early V&V. Subsequently the expected stakeholders tend to diverge in the papers, and there is a vague understanding about how early V&V slots into the overall SE processes and how it can guide development effectively.

6.3 RQ2 - What are the means of describing system behaviour at an early stage of development?

This section summarises the data extracted to answer RQ2, namely questions 4, 5, and 6 in Table 1.

6.3.1 How is system behaviour represented in early V&V?. Figure 7 shows the languages utilised in the solutions to describe system behaviour. The category "Other" refers to entries only represented once in the extraction which are not created specifically for the solution, which instead is represented by "Custom language". When we consider the implementation of the solutions, we differentiate between tools, languages, and formalisms (where possible) based on existing classifications as presented in Section 4.

The data in Figure 7 is based on the reported languages/formalisms from the analysed papers. Here, it is worth noting that the input varied a lot in detail. Notably, some papers state "SysML" without specifying or showing what sub-set is utilised. Similarly, different types of diagrams or means of representing behaviour are shared among the various languages, such as state-based formalisms. Moreover, the selected papers reported languages or formalisms in an inconsistent way. These issues do not allow us to perform meaningful clustering of the results and therefore we limit the reporting of the extraction results to the languages, which all analysed papers mention.

While there is a large spread of languages to represent system behaviour in the early phases of development, it is clear that SysML is by far the most common language, and much of the language is utilised, especially the behavioural diagrams and block diagrams. Figure 8 illustrates the sub-sets of SysML utilised in the papers adopting SysML as part of the proposed solution. We note that the distribution of diagram types is found in a similar way for UML and other UML profiles.

When considering the sub-parts of SysML utilised in the selected papers, activity diagrams and state machine diagrams are the most popular means of using SysML for describing system behaviour, particularly it is utilised in many papers aiming for automated translations. For example, Staskal *et al.* [P140] map SysML activity diagrams to the symbolic model checker nuXmv, and Mahani *et al.* [P36] similarly map SysML state machine diagrams to the NuSMV model checker. Many solutions use more than one diagram of the language to describe the behaviour and often it is not entirely clear what is used and what is not by simply reading the paper. Nonetheless, of the four diagrams classified as behavioural diagrams in the SysML standard, activity and state machine diagrams are preferred over use case diagrams and sequence diagrams. Besides, one paper argues that the solution utilises the entire SysML specification [P114], and four others that all of the behavioural diagrams are utilised [P12, P46, P91, P122].

SysML, as a general-purpose language, aims to be a solution for all types of systems. Further, it is evident that UML and languages related to UML, such as MARTE, are among the most commonly used languages and formalisms apart from SysML (which indeed has been typically implemented as a UML profile until the recent

Just Accepted

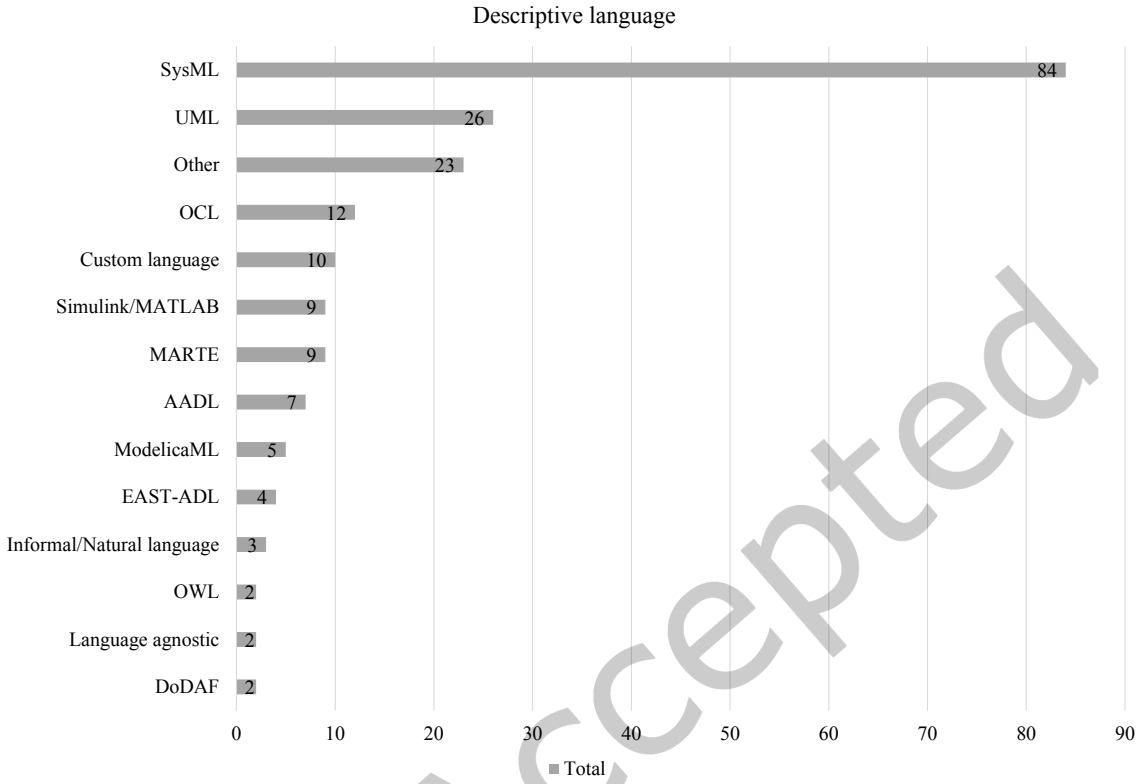


Fig. 7. Reported language used for describing the system behaviour.

Just Accepted

version SysML v2). This might indicate that general-purpose languages such as SysML and UML are suitable for describing early systems specifications in general and behaviour in particular. Nonetheless, a large number of papers utilise a means of description that is only observed once in the selected papers, indicating a large variety of domains and possibly the need for domain-specific support. For example, Supremica is used by Markovski [P26], which extends finite automata to handle large complex industrial systems, KARMA used by Ding *et al.* [P78] to unify formalisms across several MBSE models and simulations, or SWRL used by Chen *et al.* [P95] in combination by OWL to leverage ontology reasoning for verification.

A few papers propose custom languages or formalisms in their solutions. Miao *et al.* [P1] present a Python-like custom informal language for their system descriptions. Lemazurier *et al.* [P28] utilise a domain-specific language for nuclear power plants, along with several other domain-specific languages, to manage their system descriptions. Similarly, Deng *et al.* [P48] utilise a custom language and environment for mechanical product design, and Stachtiari *et al.* [P60] use their custom language for describing satellites. Singh and Muller [P80] use a tool-based approach to describe their system of interest based on needs observed in their industrial context of manufacturing systems. Bernaerts *et al.* [P126] use a specific type of model for describing safety-related aspects in the automotive domain. Miyazawa *et al.* [P99] use their custom state machine formalism for describing robotic

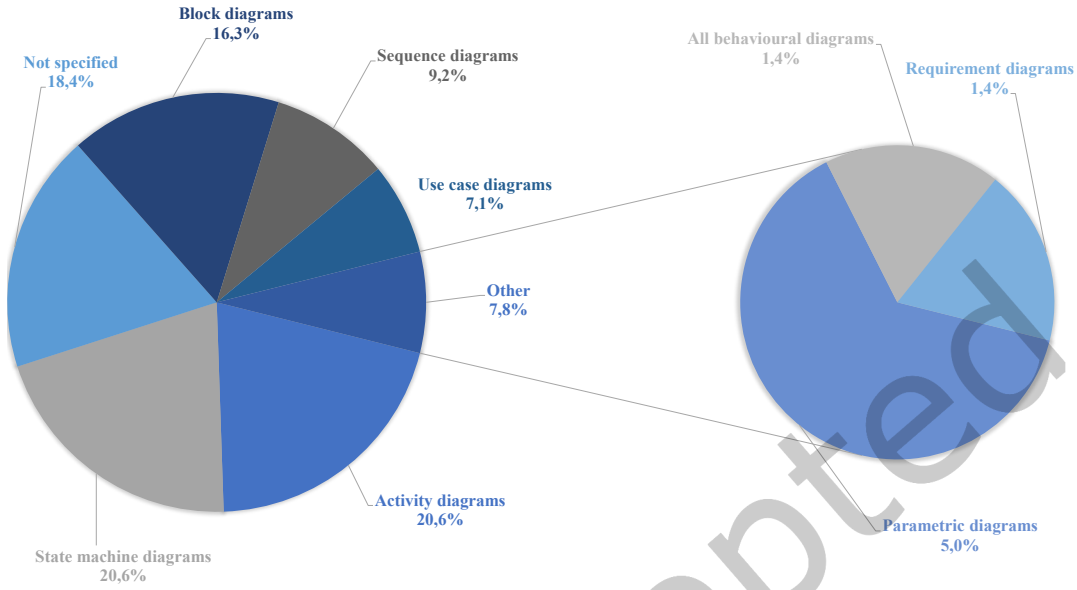


Fig. 8. The sub-sets of SysML used for solutions when describing system behaviour.

Just Accepted

systems. Zhang *et al.* [P141] propose an integrated intelligent modelling and simulation language to cope with the combination of domain specific models in System of Systems.

6.3.2 What language or formalism is utilised for behaviour analysis? To present the results from this data extraction we differentiate between the reported languages and the reported formalisms. Analogously to the behaviour description, authors rarely report in a precise way how a language is implemented or what parts are employed for what purpose. Further, some papers report on the particular formalisms employed while others do not. In particular, Figure 9 shows the reported languages and their frequency for analysing behaviour, where "Other" groups the entries reported only once that are not custom made for a particular solution; Figure 10 depicts the reported formalisms for analysing the behaviour.

Figure 9 and 10 clearly show that languages and formalisms used for analysis are typically case-dependent, since a large majority of the entries is represented only once or twice in the papers. In this respect, many reported languages and formalisms serve narrow/domain-specific purposes that are not easily portable to a more general case. Examples include languages such as Event-B [P24], OWL [P104], LabVIEW [P97], or Sabotage [P137]. In addition, despite the widespread use of SysML and UML-based languages for the system description, there is no evidence about potential "standard" analysis approaches for early V&V of specific properties.

Apart from the case dependent solutions, some general-purpose languages and formalisms are frequently used, namely SysML, MATLAB/Simulink, Modelica, Petri nets, and various state-based formalisms. Moreover, it is interesting to notice that there exists a group of solutions presented as implementation-agnostic with respect to the particular type of formalism and language used for the analysis [P77, P94, P95, P125]. Friedl *et al.* [P77] argue that their solution can be adapted to cover various case-dependent languages. Similarly, the solution from Castet *et al.* [P94] involve ontologies and is presented as being adaptable in terms of languages, where an implementation

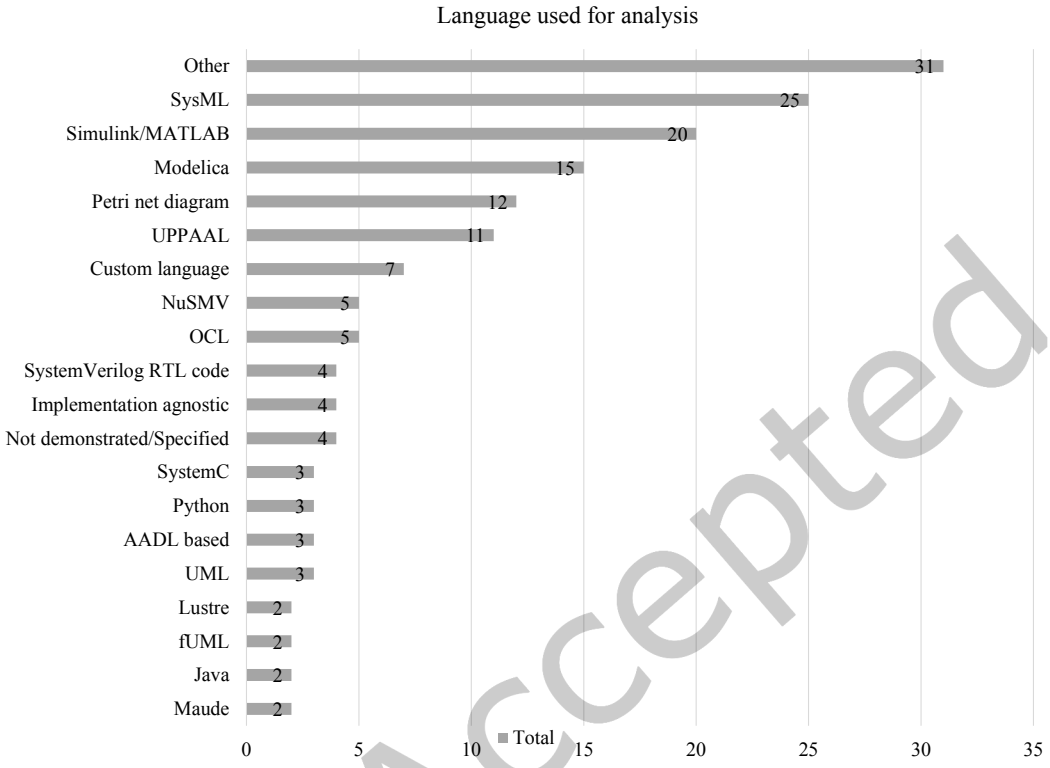


Fig. 9. The language used for performing analysis in the solutions

example is given in the Modelica language. Chen *et al.* [P95] also utilise a method of automatically creating specific design ontologies and rules from requirements, and argue that the target implementation depends on the context considered for the system under study. Damm *et al.* [P125] showcase a method for contract-based virtual integration testing and consider it to be language and tool agnostic. Similarly, as for languages used for behaviour description, there are a few custom languages and formalisms, of which two overlapping with custom languages for describing behaviour [P1, P28]. Moreover, Kang *et al.* [P4] extend previous work to analyse EAST-ADL models, and Brandstetter *et al.* [P59] use validation rules for automation process software requirements validation that is not tooling dependent.

6.3.3 If the description and analysis language differ, how is the transformation performed? Although SysML is the primary language used to represent system behaviour, it is seldom used for analysis. More in general, only in 24 cases out of all the papers is the language or formalism for representation and analysis are the same, where SysML is used 13 times [P22, P31, P50, P51, P55, P58, P63, P70, P101, P104, P107, P114, P130, P147]. As a consequence, in all the other cases a transformation (which can be composed of several sub-transformations) is required between the different languages and formalisms, at least to translate concepts from the representation language towards

Just Accepted

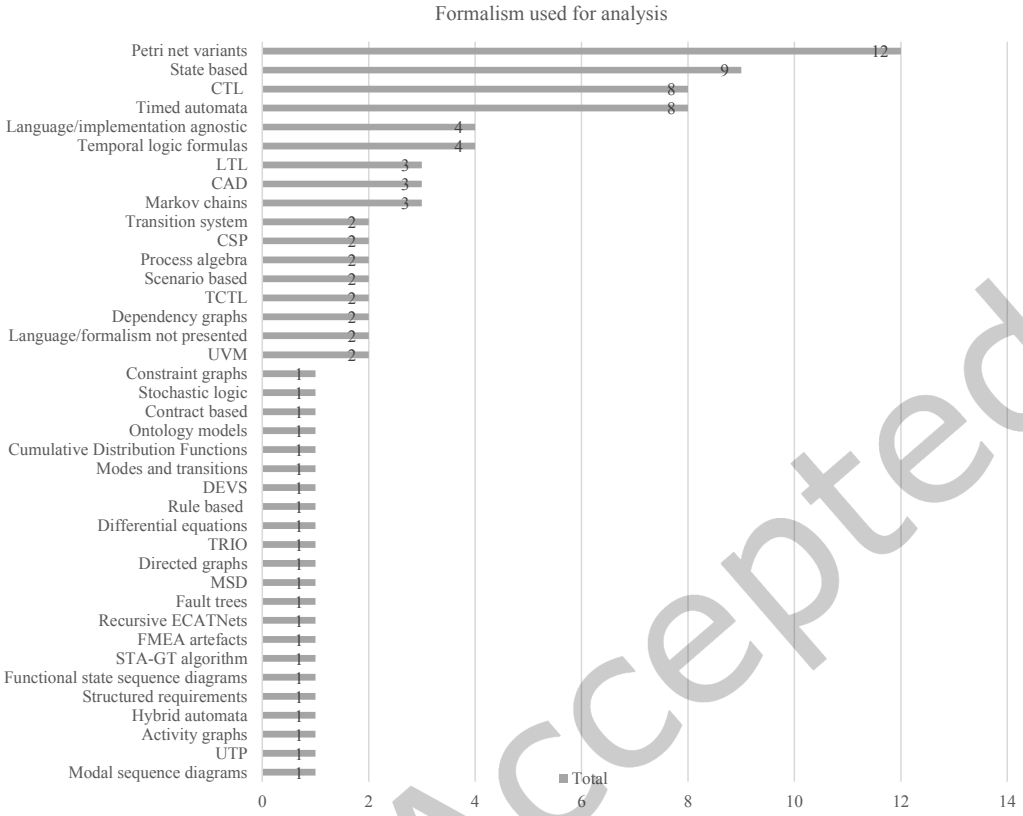


Fig. 10. The reported formalisms used for analysis in the papers.

Just Accepted

the analysis one. In more complex scenarios, such transformation would be used to extract and synthesise the necessary information to perform the analysis due to early V&V.

We note that 94 of the papers (63.0%) utilise fully automated transformations, while 19 (12.7%) solutions utilise semi-automated transformations, and 12 papers (8%) use some form of manual transformation. We note that rarely do authors provide proof of transformation correctness in case of automated solutions, possibly highlighting a gap in the works. Moreover, as previously mentioned, 24 (16.1%) use no transformation. The results from the extraction indicate that if a transformation is performed, it is mostly done via automatic means; otherwise, either no transformation or semi-automatic means are employed. Singh and Muller use Dynamic A3 Architectures as an approach for validation leveraging tool support without the need for transformations [P80], particularly by promoting view management and cross-communication between teams via integrated tool capabilities, early validation is achieved by increased communication, collaboration, and integration between engineering teams. Farooq models directly in Simulink and leverages the tool capabilities for simulation and verification [P84]. Since some of the proposed solutions utilise several languages and formalisms in both the description and analysis, semi-automated transformations also include cases where some of the multiple translations are automatic while some are not. For example, González *et al.* [P62] leverage co-simulation mechanisms in conjunction with SysML models and only transforms a sub-set of the model information to MATLAB equations. Friedl *et al.* [P77] discuss the use

of SysML architecture models in a process for model integration between system architects and model experts and highlights a semi-automated guided approach for integration. Only few papers rely on a completely manual approach for mapping different languages and formalisms [P1, P30, P44, P43, P48, P59, P65, P85, P120, P123, P127]. It is interesting to note that these solutions tend to be reported as not validated in an industrial context by the publication authors [P30, P44, P43, P48, P65, P120, P123, P127] and targeting the requirements phase; *vice versa*, the solutions deploying automated transformations are often validated in industry and do not target requirements.

RQ2 discussion: When describing a system's behaviour, there is an apparent representation of SysML and UML or other types of UML profiles, such as MARTE. The prominence of these languages is not surprising and has been reported previously in other types of reviews [21, 60, 66]. Of the different types of behavioural diagrams in SysML, activity diagrams and state machine diagrams are the most commonly used. On the other hand both use case diagrams and sequence diagrams are used significantly less, seemingly not good candidates for describing the systems at this stage to enable V&V. However, as opposed to the description being uniform in the language, analysis mechanisms vary a lot. Indeed most papers utilise languages or formalisms only found once or twice in the set of selected papers. The most often reported languages are SysML, Simulink, and Modelica. Petri net diagrams (with various formalisms), UPPAAL, and other similar languages are also used often. The mappings between the languages in the cases where the behaviour representation and analysis use different means (which is mostly the case), show a clear tendency towards automated or semi-automated solutions. This is consistent with the pragmatics of model-based development (and consequently of MBSE) [78], since handling such *discontinuities* by hand would introduce accidental complexity to the solution, making its adoption difficult and even not possible at all.

In a broader perspective, the wide adoption of SysML and UML-based languages for behaviour description might confirm their status of de-facto standards for early systems' modelling. In fact, keeping those languages for early behaviour description eases the technology transfer for the developed analysis mechanisms by avoiding learning new languages (and also adopting new tools). However, being SysML and UML-based languages general-purpose, they often do not convey enough support for domain-specific analysis, thus requiring information extraction/translation towards semantic domains in which the analysis can be performed. Relying heavily on such technologies introduces further complexity into the process, which might make industrial adoption a larger challenge.

While we see some expected results from the analysis there is at the same time a lack of many important topics. Relying on model-transformations to create analytical models creates a relevant dependency on these transformations, but there is little discussion on the viability of transformations. Similarly, the notion of consistency management across different languages in addition to interoperability and scalability is mostly missing. Implementing MBSE in industrial contexts will often be reliant on large tool-chains, and relying on a set of model transformations in a landscape of changing tools, standards, and users is a considerable risk. For languages not using transformations many are implemented in advanced tooling like Simulink or integrated MBSE tool-kits like Cameo Systems Modeller or MagicDraw, hinting at these solutions being more closely tied with industry needs. Overall, the large range of analytical languages and notations proves powerful flexibility, but on the other hand, introduces complexity in the process which is a considerable trade-off.

Actionable Insights for RQ2

From the analysis performed we believe the following action points to be valuable for the community in the context of RQ2:

- There is little discussion on the transformation details, are they two-way, how often should they be employed, consistency management, coupling, maintainability, etc.

Just Accepted

- Scalability of languages for analysis could be a weak point, top choices are implemented in industrial tools while many academic choices are based on small scale examples.
- Few solutions present general approaches that can be implemented in more than one language and/or formalism for description or analysis. Considering many commonly used languages like SysML are semi-formal the reliance on specific notations and languages (and per extension tools) reduces generalisability.

6.4 RQ3 - What are the results of interest for authors performing early V&V, and what techniques are employed for the required analysis?

This section summarises the data extracted to answer RQ3, specifically answering questions 7, 8, and 9 from Table 1.

6.4.1 What methods and techniques are used for analysis? Figure 11 reports the V&V analysis methods and their frequency as extracted from the papers. Similarly to what done previously, we do not display entries only represented once in the papers and group them as "Other" in the figure.

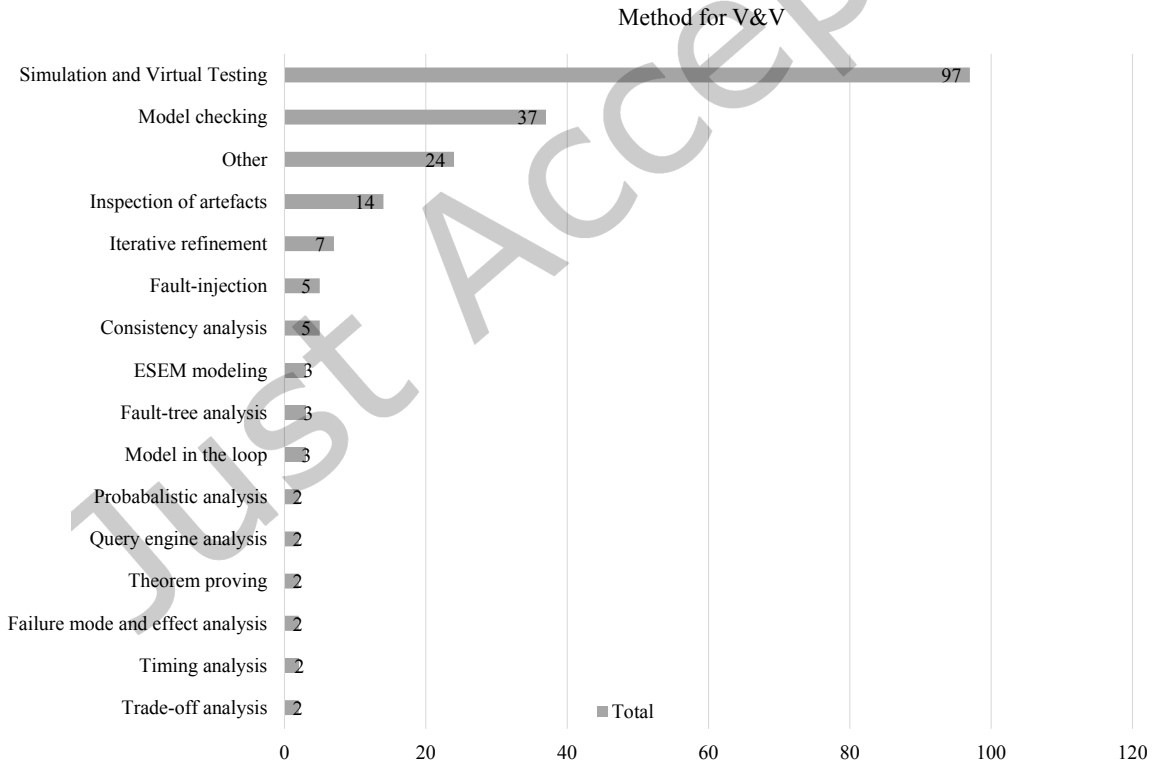


Fig. 11. The reported method for performing V&V in the papers.

Just Accepted

The most frequently reported method is simulation, which for some papers is also referred to as model execution, virtual testing, and co-simulation. The papers not utilising simulation employ many different means of analysing models, often through static analysis methods. More common types of analysis are model checking, and manual inspection/review. There exists some overlap between the reported techniques, partly due to some solutions employing several means of analysing system behaviour or because of solutions being broad in the sense that several aspects are analysed either in parallel or through different steps. As an example, Quadri *et al.* [P81] discuss a framework with several interconnected components for V&V on SysML/MARTE models, including simulation, compile time visualisation, and temporal property checks. Similarly, Liu *et al.* [P16] introduce a methodology to leverage a sub-language of AADL for simulation, schedulability analysis, syntax analysis, lexical analysis, and more.

Methods with less entries are usually devoted to domain-specific analysis, and many of the reported entries relate to techniques for reasoning about or analysing the models directly without any need for execution. Examples include: program analysis [P13], compositional verification [P138], model reasoning [P116], and correctness by construction [P60].

6.4.2 What results are of interest for the authors? This section illustrates the results collected due to the data extraction related to question 8 in Table 1. To further understand early V&V from the authors' perspective, we map the results of interest in Figure 12. Again, we do not include entries only represented once, which are instead classified as "Other".

Similarly to previous results, there is a large spread of interest in the analysis feedback and outcomes. Even more, the results for this question are influenced by the lack of harmony and consistency in the terms used and the presentation of data from the authors, which is likely due to the broad range of domains involved in the selected papers. Execution traces refer to works using the traces of execution themselves for analysis, for example, Delvi *et al.* [P74] plot execution traces to compare and evaluate energy consumption, while Kotronis *et al.* [P100] visualise and compare traces for railway system configuration performance. Safety features can include, for example, guarantees of execution as in the case of Dragomir *et al.* [P148], or safety assessments from Fault Trees or Failure Mode and Effect Analysis in the case of Krishnan *et al.* [P139], or more comprehensive V&V safety analysis through a set of techniques as in the case of Bozzano *et al.* [P67]. Inconsistency can refer to inconsistency between different system behavioural views, as in the case of Duhil *et al.* [P58], or between different tools and notations belonging to traditionally separated teams as in the case of Gregory *et al.* [P117]. Intended functional behaviour can refer to simulation to validate the execution of generated code like for Bocciarelli *et al.* [P10], or to detect design and integration errors with model-checking as in the case of Braspenning *et al.* [P88]. Functional requirements verification, on the other hand, can refer to the explicit requirement satisfaction checks as with Kang *et al.* [P4], or as in the case of Anwar *et al.* [P86] where SystemVerilog assertion code is generated for a particular SoI based on the design requirements defined using OCL, which can be leveraged during the eventual detailed system design for requirements verification.

Domains such as power plants [P44], canal/waterway systems [P85], web applications [P132], and medical systems [P140] have varied vocabularies compared to the more represented domains such as aerospace or automotive.Nevertheless, it is possible to identify some clear trends, such as execution traces and intended functional behaviour being of high interest. Similarly, results commonly obtained through static analysis, such as liveness, inconsistency, deadlock-freeness, reachability, etc., are also reported often.

A few papers argue that their analysis approaches are adaptable to the application of interest for a specific context, and as such the results of interest might vary depending on the utilisation scenario. Notably, Anwar *et al.* [P41] present a meta-model for modelling FPGA-related concepts and argue that their solution can be used to generate test benches for various purposes. Votintseva *et al.* [P47] reason about multi-domain systems and propose solutions for managing cross-domain analysis, suggesting that their approach depends on the chosen

Just Accepted

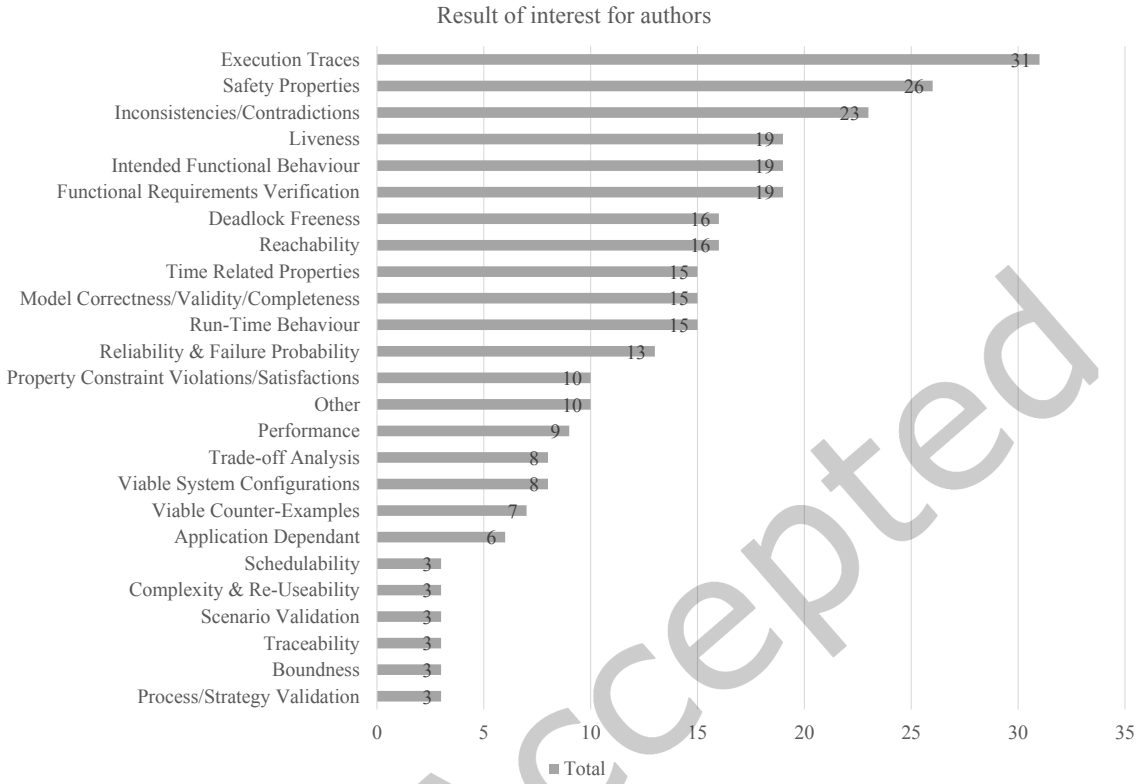


Fig. 12. The results of interest for the authors.

Just Accepted

domains and applications. Kaslow *et al.* [P55] use various inspection forms to perform their analysis, which is argued to be adapted for the stakeholder needs. González *et al.* [P62] propose a means of using co-simulation for early V&V and argue that the results of interest depend on the evaluation context. Hecht and Chen [P130] utilise various query-based operations on SysML models and argue that this analysis should be adapted for the particular system of interest and corresponding models. Zhang *et al.* [P141] use their custom language X with a prototype tooling that covers an extensive array of potential analysis, and as such, the results vary accordingly.

6.4.3 What are the tools used for analysis? This section discusses the outcomes of the data extraction related to question 9 in Table 1. We present a categorisation of tools used in Figure 13 and refer to Appendix A for an explicit mapping of tools to solutions.

Consistently with previous results on languages for system description and V&V analysis, the number of tools used is extensive, and most of the tools are found only once or twice among the analysed papers. The more common categories are integrated MBSE toolkits, graphical programming/simulation environments, model-checkers, modelling frameworks, and simulation toolkits. The most commonly used individual tools are MATLAB/Simulink with corresponding libraries and EMF based solutions. EMF is one of the leading free/open-source platforms for modelling, which makes it an attractive tool/platform for academic investigations [82]. MATLAB/Simulink, in a

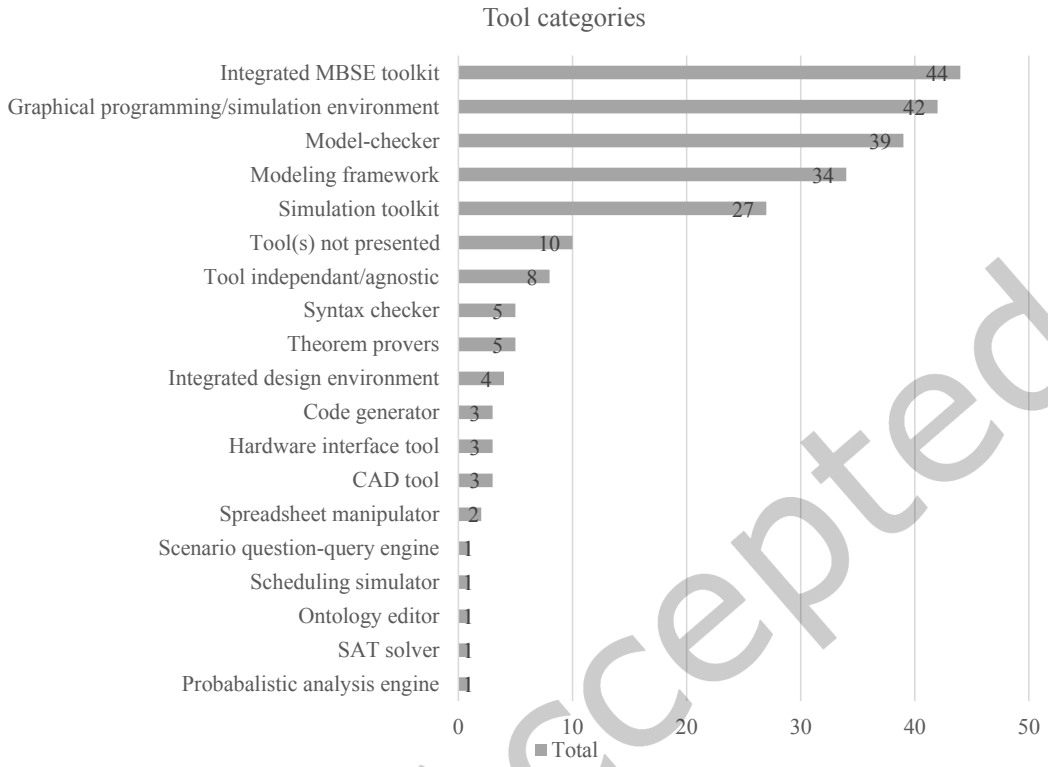


Fig. 13. The reported tools used for analysis in the papers

Just Accepted

similar fashion, is suitable for both academics and industry and has a rich history of applications in the model-based community [4]. Apart from these tools, many proprietary SysML editors are commonly used for their integrated analysis capabilities, eliminating, at least partially, the need for different tools for describing and analysing behaviour. In some cases, the tools are not presented [P12, P31, P37, P38, P43, P54, P66, P92, P98, P104], and other times it is argued that the solution is not tool dependent [P47, P64, P76, P95, P101, P109, P125, P134]. Moreover, some authors have proposed custom tools or environments for their analysis [P27, P48, P68, P82, P87, P141, P144].

RQ3 discussion: Simulation is by far the most commonly used means of analysis. This perhaps indicates that for meaningful analysis at early stages, it is required to have advanced means of analysis, particularly for dynamic behaviour. Apart from simulation, there is a broader spread of different methods that can be reduced to model checking or automated reviews/inspections.

The results of interest are typically tightly coupled with the domain, the adopted languages, the tools, and so forth, hence showing a large variety of target properties. More explicit and shared set of results of interest can be observed when model checking is adopted, namely freedom of deadlock, liveness, safety, and reachability. In general, the embedded systems domain seems to have a clearer view of the process and scope of early V&V, which is reflected in the more compact set of tools, languages, and results compared to the other types of domains.

Moreover, the embedded systems domain shows some maturity regarding early validation practices, with earlier publications in the observed timeline for early V&V.

Another interesting aspect is that models describing the behaviour in a somewhat uniform way, often in SysML, can lead to a wide array of analyses; this shows the powerful flexibility of the semi-formal nature of general-purpose languages like SysML and UML profiles. Of course, such semi-formal descriptions often entail the need of transformations to more structured representations for analysis, which increases the complexity and reduces the freedom of modelling, as automatic transformations require structured formatting.

The tooling reported for analysis in the papers is the category with the most significant spread observed. Very few tools are seen as good enough for the general audience of MBSE. Of the more frequent entries, Eclipse/EMF, MATLAB/Simulink, Papyrus, OpenModelica, and UPPAAL are the more commonly reported tools. Some tools that do not perform transformations between languages are also represented in the papers, like MagicDraw and Cameo Systems Modeler. These latter two are tools used in solutions where early analysis is performed directly on SysML diagrams thanks to functionalities embedded in the tooling. Moreover, as expected, these solutions are often observed in industrial applications, where it can also happen that the tooling is customised on-demand to perform specific tasks.

While tooling is tightly connected with MBSE, it is surprising that so few solutions claim to be tool agnostic. In fact, if tooling is central, MBSE likewise considers methods, methodology and languages. However many of the solutions rely on tool-specific analysis, regardless if the solution is extensive or not. Academic tools tend to be more compact and openly accessible for the best impact, while industrial tooling should often integrate into larger processes and there is a reluctance on openly available tooling as it rarely is scalable and maintainable for large enterprises in addition to intellectual property (IP) protection concerns. Therefore, the fact that much of the tooling is very case-specific feeds into the known problems of interoperability and maintainability.

Actionable Insights RQ3

From the analysis performed we believe the following action points to be valuable for the community in the context of RQ3:

- Tooling should carry much of the weight in terms of MBSE processes, many of the observed tools are disjoint from the overall processes and need to be integrated efficiently while few solutions emphasise interoperability.
- Few solutions discuss approaches that can be adapted or applied in more than one notation/tool, further reinforcing the vendor lock-in.
- While simulation is the most common analysis method it is rarely discussed for what conditions and boundaries its results can be considered valid or not. Moreover, how the analysis should be integrated into decision-making concretely, apart from high-level observations, is left unspecified.
- The target properties for the proposed early analysis are many with no evident pattern of classification or catalogue referenced in the literature.

6.5 RQ4 - Who are the users of early V&V?

This section summarises the data extracted to answer RQ4, more specifically questions 10, 11, and 12 from Table 1.

6.5.1 What is the reported domain? Figure 14 displays the domains identified in the selected papers: there is a multitude of domains observed in the papers, of which the most prominent are Aerospace and Avionics[14]. Other domains with a significant presence are embedded systems, cyber-physical systems, safety critical systems,

[14]We distinguish these domains as they are apparently differentiated in the observed papers.

Just Accepted

automotive, and railway. Finally, there is also a large category of domains only reported once in the papers, such as nuclear power plants [P28], canal systems [P85], web applications [P132], and cloud computing [P134].

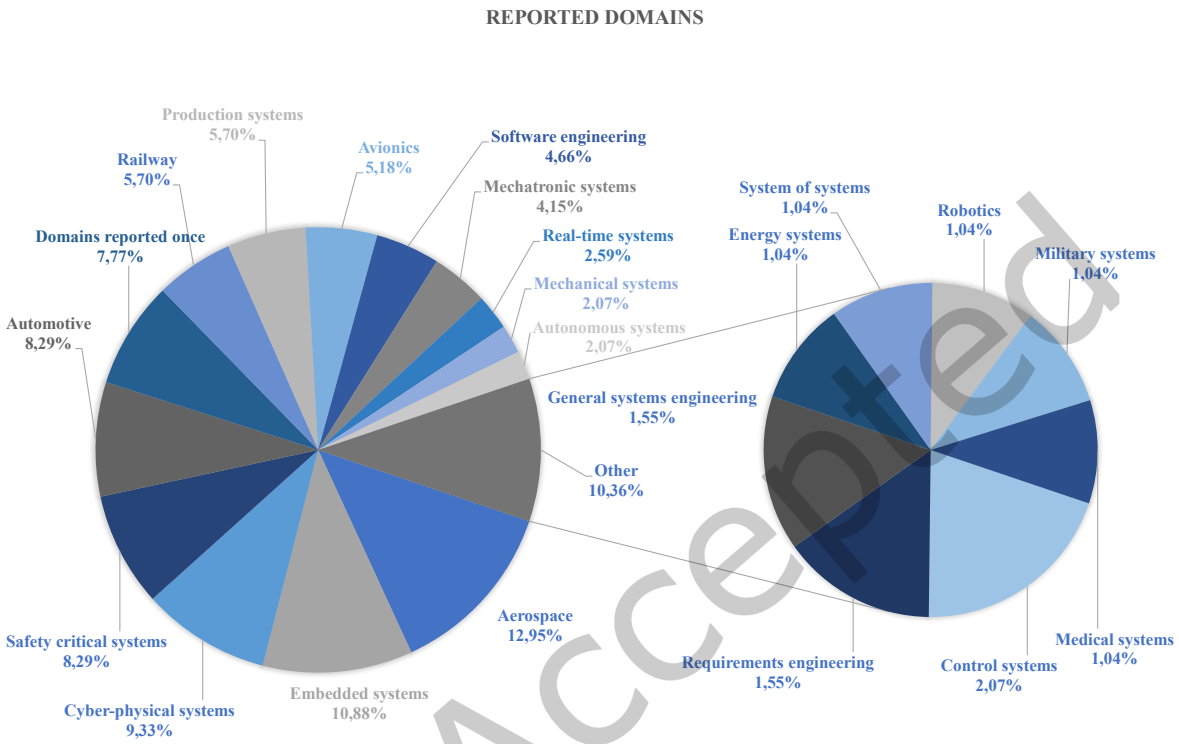


Fig. 14. The reported domain of solutions

Aerospace is somewhat expected to be one of the most commonly reported and investigated domain given the complexity of the developed systems and the historical relevance of this field. Indeed, there are prominent promoters for MBSE, such as NASA, which produce quality research on the topic [42, 70]. Similarly, embedded systems are quite mature regarding formal methods and model checking [54, 85], which are often reported in the papers. Moreover, the automotive and railway industry have strong foundations on model-based practices [7, 24], and standards such as AUTOSAR exist for automotive [32]. Cyber-physical and safety critical systems are broader categories than previously mentioned ones, but we observe a strong presence regardless. In this case, CPSs typically benefit from the unified view models provide (e.g. for integration analysis purposes) while safety critical systems require early analysis for providing the necessary evidence that systems will meet the imposed requirements.

6.5.2 Is the solution domain specific? Of the analysed papers, 117 contain domain-specific solutions (78.5%), 16 (10.7%) papers argue to be domain-independent, and 16 (10.5%) illustrate a domain-specific solution but make a case for it being extensible for more domains. Examples include Bocciarelli *et al.* [P10], which describe an

Just Accepted

approach based on the IEEE 1516-2010 - HLA standard and apply it concretely in the SoS domain, but argue for wide applicability. Similarly, Stewart *et al.* [P17] describe a process for safety analysis in the aerospace domain but argue that it can be extended for any safety critical domain such as automotive or nuclear power plants. In the same way, Herzig *et al.* [P30] apply a SysML-based simulation analysis method on a telescope use case but argue that the approach can be extended for other domains. The fact that such a large majority of papers are domain-specific and many different domains are reported is consistent with the broad range of solutions proposed in the analysed papers.

Of the non-domain-specific papers, only 3 are validated in an industrial setting [P1, P34, P105], indicating that such solutions might not have the necessary maturity required for industry. The three papers discuss their applications in at least two domains. For example, Cheng *et al.* [P34] discuss an approach for automatic analysis of multi-view architectures and present both an automotive and micro-service cloud application.

6.5.3 How was the solution validated? This section answers question 13 from Table 1 by extracting whether the solution was validated in an industrial setting or not, and what kind of validation is discussed in the selected papers.

51 papers (34.2%) evaluate their results in an industrial setting, which showcases a strong industrial presence. We re-emphasise here that the validation of the approach is extracted from the publication itself, hence we rely on the definition adopted by the authors.

114 (76.5%) of the analysed papers use some form of running example as defined by Shaw [79] for the validation of the solution; in particular, here we group toy examples with slice of life examples. A sub-set of contributions, 18 (12%), present empirical observations of the solutions [P3, P4, P5, P11, P14, P40, P49, P60, P62, P66, P67, P80, P87, P103, P110, P134, P149]. For example, Mens *et al.* [P3] introduce a new method for testing and validating state-charts, and part of their evaluation is a controlled user study to measure the tools' effectiveness and usability. Andrade *et al.* [P5] introduce a method to map SysML activity diagrams to Petri Nets for the evaluation of real-time systems with energy constraints and measure the obtained analysis results with hardware measurements, which confirm the adequacy of analysis results. Anwar *et al.* [P11] introduce a model-based method for design of FPGAs and measure the time taken for baseline methods compared to their approach in man-hours. Eventually, 17 (11.4%) papers have no example or empirical observation for their solution [P15, P25, P31, P32, P42, P53, P58, P59, P70, P98, P100, P104, P119, P124, P128]. For example, Lima *et al.* [P25] describe a semantic for reasoning about SysML diagrams via refinement and support their approach through abstract high-level disjoint diagrammatic examples; instead, Gauthier *et al.* [P119] explain a process to transform SysML models to Modelica code without any example in the publication apart from the transformation rules.

The outcome about users of early validation is consistent with the other categories in the sense that there is a large and significant spread. The outliers in this regard are aerospace (with a related domain referring to avionics), embedded systems, CPS, safety-critical systems, railway, and automotive. Interestingly, solutions in the embedded systems domain tend to be consistent in terms of solutions proposed by the analysed papers, while the same cannot be said for the other domains. In fact, in most of the larger reported domains there is no common/shared view of how early V&V is expected to be performed. For example, in aerospace there are examples of requirement analysis [P7], simulation [P10], schedulability analysis [P16], model checking [P17], inspection of diagrams [P55], and so on.

RQ4 discussion: As it might be expected, most of the solutions are domain-specific, and only around 11% claim to be applicable to any domain. As a matter of fact, since performing analysis early in the SE process requires assumptions to be made or uncertainties to be managed, the solutions tend to be domain-specific due to more precise and reliable information to build-up the analysis on. Further, as many target languages or tools are often coupled strongly to a domain, profiles and constraints need to be in place to enable SysML and similar semi-formal (general-purpose) languages to be applied with enough rigour.

Just Accepted

Around 34% of the analysed papers describe research performed in an industrial setting, seemingly indicating the industrial applicability of many solutions. Indeed MBSE is a paradigm with a solid industrial perspective, which puts many requirements on the types of solutions. Related to this substantial industrial scope we note also that most of the solutions are based on examples to various degrees, with less focus on empirical measurements. Indeed, only few papers discuss empirical evidence to support the solutions that instead largely rely on arguing for the perceived benefits. These observations are consistent with the previously reported weakness of MBSE [41], that is the lack of empirical evidence. In this respect, we argue that empirical evidence could be hard to produce for solutions of this nature, as the measurements are intricate to define and might be challenging to perform in industrial settings without a high risk of introducing bias and other validity threats. Nevertheless, this is a reoccurring problem regarding model-based practices that hinders evidence-based discussions on a broader scale [16].

Actionable Insights RQ4

From the analysis performed we believe the following action points to be valuable for the community in the context of RQ4:

- SE at its core pursues customer satisfaction and correct delivery of systems, yet there are few attempts to measure and compare solutions, and there is a general lack of baselines for V&V approaches.
- MBSE processes are large and map well to complex domains. However many of the examples included in academic works seem to be too simplistic to be convincingly adopted in industrial contexts and make no attempt to discuss these concerns.
- MBSE artefacts are often complex and domain-specific. More unified/standard approaches could be very valuable, and a common-ground for different domains could improve training and enable easier comparison of methods and solutions.
- There is seemingly no clear indication of whom the target user(s) is in the broader context of early V&V, further indicating a lack of clear definition and placement in MBSE processes.

6.6 RQ5 - What limitations do authors see, if any, with their implemented solutions?

This section summarises the data extracted to answer RQ5, namely question 13 in Table 1.

Not all authors identify limitations for their proposed approaches for early V&V. The results of those that do identify limitations are presented in Figure 15. The largest identified limitation is that the proposed solution is not fully developed, often partially covering what is of interest for the authors. Concretely, the solution by Zhu *et al.* [P2] does not consider the validity of SysML models that are used as input for their automated approach, and Ghitri *et al.* [P118] cannot perform automation for all parts of their transformation from SysML to UPPAAL. The second biggest limitation is related to the proposed analysis due to the simplifications introduced by the adopted level of abstraction in the modelling activities. For example, Zhang *et al.* [P45] note that their approach does not adequately cover more complex systems compared to their case study due to the simplicity of the models. Brandstetter *et al.* [P59] conclude that their approach can only confirm whether the model of requirements is valid and not if the actual system is. Another significant challenge is due to integrating languages when there is a difference from description and analysis language or formalism [P26, P36, P47, P57, P62, P64, P86, P109, P123, P130, P134]. For example, in the work by Mahani *et al.* [P36], automation is tricky between SysML and NuSMV due to the differences in representations. Other issues reported are: the lack of automation [P15, P19, P22, P47, P65, P106, P116, P140]. For instace Staskal *et al.* [P140] argue that due to the semi-formal method of modelling for SysML, it is difficult to achieve consistent automation. Limited expressiveness of the behaviour description language is another common limitation [P30, P34, P70, P122, P128, P134, P135, P138, P148]. For example, Liu *et al.* [P138] discuss how AADL semantics makes it difficult to express timing and execution

Just Accepted

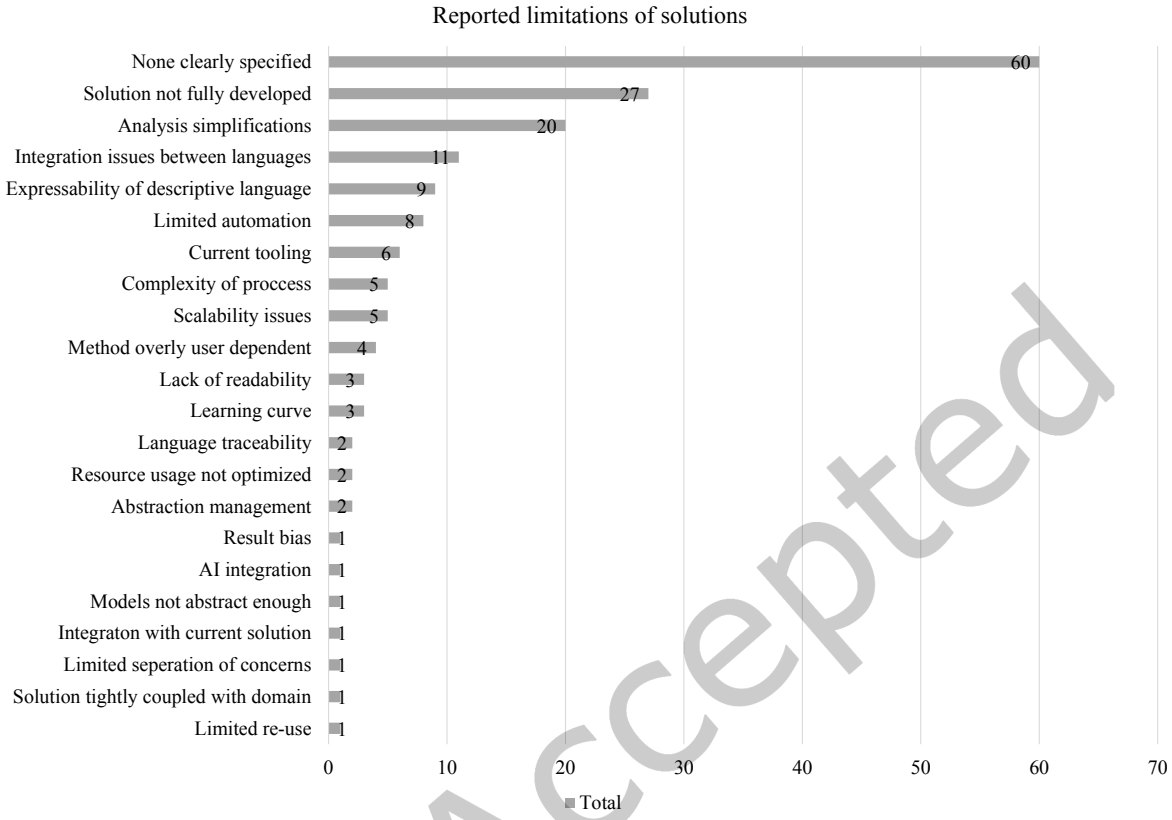


Fig. 15. The reported limitations of implemented solutions

behaviour for analytical purposes, Schamai *et al.* [P90] discuss how ModelicaML cannot describe all kinds of requirements used in their context for verification.

RQ5 discussion: We note that many of the stated limitations are related directly or indirectly to the fact of being early in the development process. Indeed, if the level of abstraction is high and/or the known system's details are little, the analysis should be expected to be limited or superficial. Still, there is a clear statement from the authors that this is an observed liability of the solution, highlighting the difficulty of balancing fidelity and abstraction of models in the early stages. Similarly, we notice many issues relating to the tooling and learning curve, which are commonly reported problems in software and systems modelling.

The extracted input for RQ5 is the lowest in terms of quantity. Indeed many papers did not discuss to any considerable extent the limitations with the developed solutions and/or for early V&V in a broader perspective. However, we highlight some patterns that emerge from those works that performed a specific discussion related to limitations. The most commonly reported limitation is that the solutions presented in the paper are not fully developed. An incomplete solution is a limitation most likely common regardless of the review subject, as papers often discuss partial solutions or work in progress towards specific goals, so it is hard to say how relevant is the connection between this pattern and the subject of this review. Nevertheless, this could be seen as a weakness for MBSE, as there is a need for mature solutions to be adopted substantially from an industrial perspective.

Just Accepted

Eventually, more typical limitations related to model-based practices are also frequently discussed, namely issues due to tooling, interoperability, learning curves, and scalability. Indeed, these limitations are not surprising and testify the need for more maturity in the field towards industrial applicability, as reported in the research literature and mentioned earlier in this article.

In a broader perspective, MBSE and SE generally aim at the entirety of a system life-cycle. Many MBSE methodologies, such as MagicGrid [65], provide a framework for activities and methods across the various life-cycle stages. Interestingly, only few solutions discuss the potential limitations of interconnecting early V&V activities and results with later stages. Notably, while some papers make a connection between phases, the discussion and application of traceability between artefacts in each phase are limited. In the visions presented by organisations such as INCOSE, it is often reiterated that future modelling should encompass all parts of a system life-cycle. However, without more sophisticated traceability means between the models produced at various stages, there is an inherent risk of introducing inconsistencies or additional efforts related to managing information across artefacts. As a matter of fact, this is one of the identified issues with document-centric development that has often been used as an argument to move towards model-based practices [38].

Actionable Insights RQ5

From the analysis performed we believe the following action points to be valuable for the community in the context of RQ5:

- MBSE processes are continuous (and often iterative) methodologies for system refinement. The limitations hint at the difficulty in managing model fidelity and abstraction to correctly leverage analysis results from considered artefacts.
- MBSE artefacts are expected to evolve, therefore the limitations related to integration, abstraction management, and scalability have an important impact.
- Often the main stakeholders in MBSE are the engineers; since many of the limitations are due to the difficulty to apply the methods for the corresponding scenario, it can be derived a general lack of prioritisation for usability concerns.
- Frequently recurring limitations are due to analytical simplifications, which seems to contradict the inherent essence of *early* V&V. This hints at a missing clear definition of what early is, and what can be expected from analysis at such a stage.

7 HORIZONTAL ANALYSIS

This section presents significant results coming from the horizontal analysis performed on the extracted papers. All the data originates from the intersection of the extracted data based on the questions presented in Table 1 and discussed so far. Again here we underline that a portion of the collected data has been coded into different categories so that more mappings across the papers could be elicited via automated means. In this respect, the coding is available in the publicly available replication package, together with the original extraction results.

In the reminder of this section we present the more interesting outcomes of the two-way mappings across the coded data. Moreover, we also present selected three-way mappings we have extracted and deemed interesting.

7.1 SysML and simulation

In this mapping, we have extracted the papers reported using SysML, not including UML, MARTE, or other related languages. Then, we intersected this extraction with whether a solution employs simulation. The mapping demonstrates that solutions that employ simulation tend to utilise SysML, and *vice versa* solutions not using simulation do also tend to use SysML less frequently. In particular, in the case simulation is used, ca 60% of solutions also utilise SysML, while if the solution does not, only ca 42% of these solutions use SysML.

Just Accepted

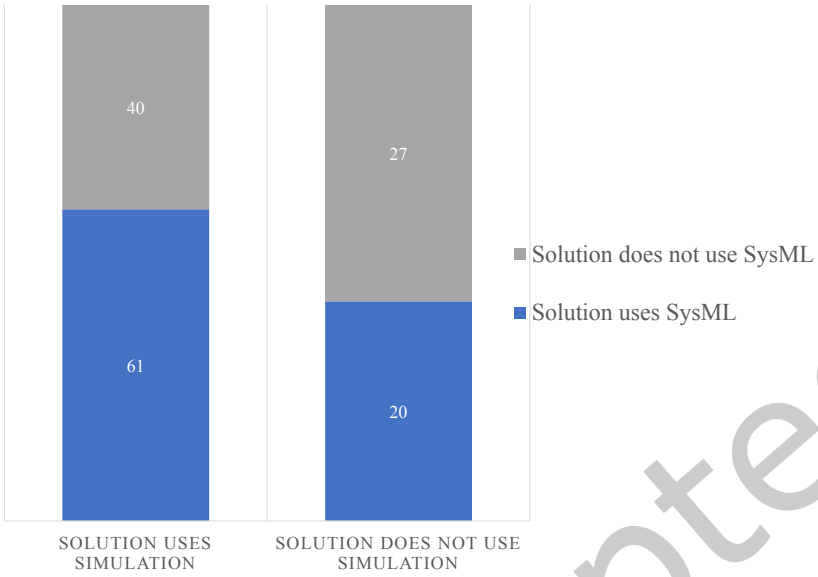


Fig. 16. Horizontal mapping between whether a solution uses simulation and employs SysML

Just Accepted

7.2 Tools and extra results

This section regards the mapping of the data presented in Figure 12 and Figure 13. In this case, we only report on whether a solution uses more than one tool for analysis, and if the solution argues for results apart from V&V (see this horizontal mapping in Figure 17). The outcomes show that when the authors are interested in a broader feedback than just V&V results they are less likely to use more than one tool for analysis. On the contrary, when authors are not interested in results apart from V&V, they are likely to use more tools for analysis. Perhaps this could be attributed to the maturity of the proposed solutions: less mature approaches might include more soft benefits for persuasion purposes while mature solutions would focus more on the performances measured in the application scenarios.

7.3 License and development phase

This section discusses the intersection of the data related to the tool licensing and the development phase involved in the solution, the results of which are shown in Figure 18. The tool licensing tends to be proprietary for the design phase, while it is more likely to be free in the requirements phase. Moreover, solutions involving both phases show an (approximately) even mix of open-source and free tools compared to proprietary licenses. The reported mapping could be partly explained by the fact that the affected artefacts for requirements contra design will vary much in complexity. Notably, requirements in their most basic form are well suited to tables and other structured text forms. Therefore, it makes sense that the corresponding tooling is less extensive and hence licenses might be an observable side effect.

7.4 Industry, license and SysML

In this section, we perform a horizontal mapping of the licensing category of a solution, whether it is investigated in industry or not, and whether the solution uses SysML; the results of this mapping are presented in Figure

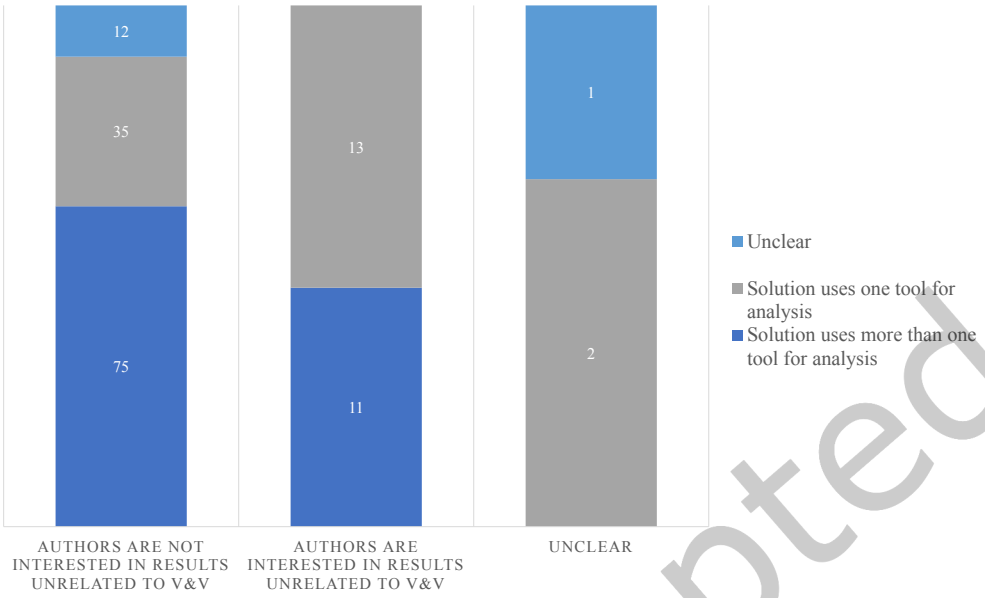


Fig. 17. Horizontal mapping between whether authors are interested in results in addition to V&V aspects and whether one or more tools are used.

19. While (as expected) academic solutions tend to favour free or open source tools and the industry favours proprietary tools, we see two divergent categories in the three-way mapping. First, we note that if SysML is used in academia, the tooling is more likely to be proprietary instead of free, while non-SysML solutions are most likely free or open-source. Moreover, for industry, the non-SysML solutions are more likely to be free as opposed to proprietary. In this case, we observe that regardless of where SysML is employed, the tools utilised are more likely to be proprietary, while the license is more likely to be free or open-source if the solution does not use SysML.

7.5 Other observations from horizontal analysis

In analysing the extracted data, we performed a horizontal mapping of most parameters that could be quantified, leaving out the more qualitative reporting. In this regard, we found some interesting patterns apart from the more significant ones reported so far, and we briefly mention these here:

- SysML is used more in design than for requirements. 13 of the 29 (44.8%) investigations performed in the requirements phase use SysML, while 61 of the 106 (57.5%) solutions in the design phase use SysML. This may be partly because solutions in the requirements phase can be done formally instead of relying on an informal language like SysML, and consequently avoid the need of moving towards a more formal language for the analysis;
- Simulation is more common in domain-specific solutions, as 83 of the 117 (70.9%) domain-specific solutions utilise simulation, while only 7 of 16 (43.7%) solutions utilise simulation in non-domain-specific solutions. This can be partly attributed to the fact that domain knowledge can be implicitly (re-)used to enable simulation in early stages, while the same details need to be provided for domain-independent solutions;

Just Accepted

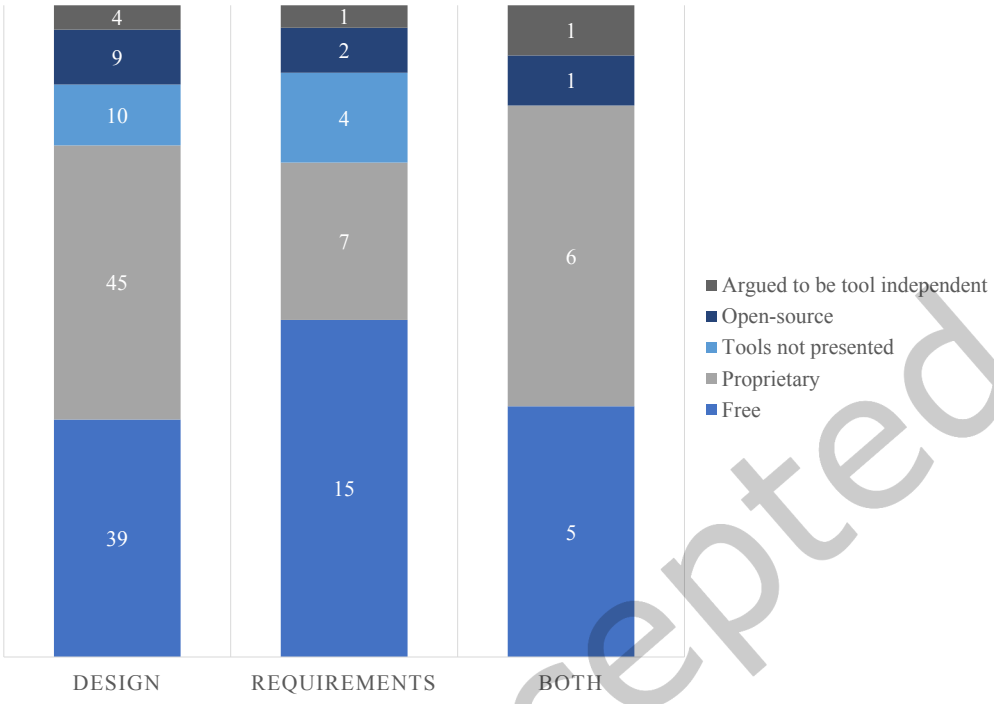


Fig. 18. Horizontal mapping between developmental phase and tool licensing.

Just Accepted

- Domain-specific solutions have more proprietary licenses, and 43 out of 117 (36.7%) domain-specific solutions use proprietary licenses, while 17 of the 32 (53.1%) partly or non-domain-specific tools are free. In this respect, domain-specific solutions may need more advanced or specialised software, leading to fewer available open-source or free solutions;
- 32 of the 51 (62.7%) solutions in industrial settings use SysML, and 49 of the 98 (50%) academic solutions utilise SysML. Here we expected a more significant separation between industry and academia, especially considering that most SysML tools are not free or open-source. However, the extracted data is relatively similar for both industry and academia. This further strengthens the notion that SysML is a de facto standard language for MBSE;
- Industrial settings tend to use more than one tool, as 34 of the 51 (66.6%) solutions use more than one tool in industrial settings. Whereas, 52 of the 98 (53%) solutions in academic settings utilise more than one tool. This tendency can be explained by the fact that industrial settings are more likely to need advanced or specialised tools for their purposes, and seldom one tool can be used for all analysis purposes;
- Authors are more likely to find limitations in academic settings, as 64 of 98 (65.3%) contributions in academic settings discuss limitations in their solution. In comparison, 28 of 51 (54.9%) papers in industrial settings describe limitations in their settings. This can be linked to academy and industry having different interests in disseminating research results (e.g. discussing open challenges vs highlighting successful practices);
- Out of the 117 domain-specific solutions, 71 (60.6%) use SysML. In contrast, of the 16 solutions that are partly-domain specific, 7 (43.7%) use SysML, and 3 of the 16 (18.7%) non-domain specific solutions use SysML. This is interesting because despite SysML being a general-purpose language its usage appears to

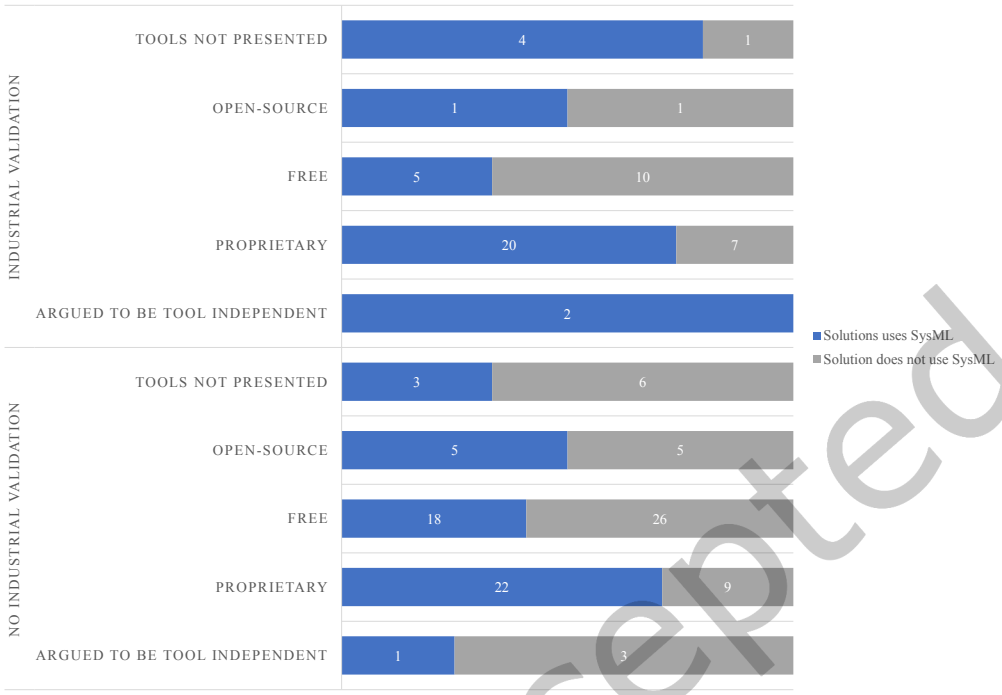


Fig. 19. Horizontal mapping between License type, industrial validation, and if SysML is utilized

decrease when the solutions tend to be more general-purpose. This contradiction could be explained by the fact that most implementations utilising SysML require anyway some extension of the language or transformation to another language or formalism for analytical purposes. Therefore, in domain-independent cases it might be preferred to adopt a behaviour description language requiring less efforts to perform V&V analysis.

Similarly to the previous Section we summarise this analysis with a set of actionable insights.

Actionable Insights Horizontal analysis

From the analysis performed we believe the following action points to be valuable for the community in the context of the horizontal analysis:

- SysML as a language is more dependent on proprietary tooling and extensions with other mechanisms for analysis compared to other languages, reducing the option for analysis without heavy investment in surrounding technologies.
- The gap between academia and industry is noticeable in many categories, hinting at a greater misalignment. Particularly, the tools, languages, and methods differ in terms of preferences.
- Few general purpose solutions utilise simulation, hinting at the need for domain-specific information for valid simulations. This, in conjunction with the overwhelming use of simulation as a V&V method, could explain the lack of general approaches.

Just Accepted

8 DISCUSSION AND OUTLOOK

In this Section we discuss an outlook about future research directions related to early V&V in the context of MBSE. The list of topics is not intended to be exhaustive, however we consider all the covered ones as critical for the INCOSE 2035 vision to become reality.

8.1 Limitations of early V&V

This literature review shows a clear wish and need for moving activities relating to V&V earlier in development. We observe a significant industrial presence in the papers with a nuanced and broad range of underlying motivations and scope of the presented activities and solutions. Indeed the problem formulations for industry are based on the precise foundation of traditional SE knowledge, that is, detecting problems early reduces costs that increase exponentially as the product traverses the development steps [88]. Therefore the need and motivation for moving activities earlier in the chain of development is a means of making the process more effective and increasing the competitiveness of the respective company. A critical factor for industries adopting MBSE in a more effective way is related with tooling. In fact, thanks to good tools it would become easier to integrate more advanced workflows with powerful capabilities, as largely discussed in other reviews [60, 73, 74, 94]. Despite tooling being central to the taxonomy of MBSE, to this day it remains a crucial inhibitor for a broader adoption of model-based practices [11, 25]. Issues often associated with tools, such as interoperability and scalability [83], seem to limit the application of academic solutions in the industrial landscape. These issues are reflected also in the results obtained in our review: analysis approaches tend to be domain-specific and ad-hoc, which makes those difficult to transfer to other application scenarios. Moreover, the limitations often refer to tool interoperability and usability issues. In this respect, it would be interesting to pursue standard interchange mechanisms, as successfully done for FMI [37, 39, 47], which supports interoperability via co-simulation. In the case of early V&V of system behaviour, the interchange mechanisms could include e.g. the representation of requirements and corresponding analysis to check for their consistency/completeness, or the description of use cases and related validation procedures.

Another relevant characteristic of early stages of development is a strong presence of *uncertainty* [57], and the systems under analysis need to consider the uncertainty for possible results. In particular, there is an increase in uncertainty due to the use of models at high levels of abstraction. Furthermore, complexity is added due to the various types of uncertainty and interactions between different models with uncertainty when dealing with heterogeneous models and systems [15]. Indeed, of the more commonly reported limitations of solutions, there is a clear relation to high abstraction, resulting in simplified analysis or limited expressiveness. These groups of limitations could be related to the very notion of *early* V&V, which is also the primary motivation for most solutions: simplifications and limited expressiveness are inherently there because of the development stage. In this respect, it would be important to have a clearer understanding of what it means to perform early V&V and what the results of these analytical methods can truly *validate* or *verify* in a system described in low fidelity models. The presence and management of uncertainty in MBSE are discussed in the literature for several contexts [7, 63, 90]. As far as these authors know, no concrete metrics or approaches exist for the efficient management of uncertainty in MBSE. Instead, other domains have had more progress in the classification and definition of metrics for uncertainty. For example, Asmat *et al.* [6] review uncertainty in the context of CPS and classify several metrics, Yan *et al.* [93] discuss uncertainty in the context of failure mode and effect analysis, and Hu *et al.* [43] discuss error metrics for model uncertainty in models with variable fidelity.

Apart from the technical limitations of solutions it emerges a remarkable emphasis on the complexity of modelling processes, learning curves, and user unfriendliness at large. There is seemingly a lack of easily accessible and fruitful means of learning the model-based approaches discussed in the reviewed literature. A study by Kahani *et al.* analysed the most commonly reported problems with EMF and found a large and growing portion of issues due to the lack of documentation and tutorials for the usage of EMF-based solutions [50]. Moreover, a

Just Accepted

state-of-the-art review of MBSE notes that one of the main weaknesses of adopting MBSE is the lack of knowledge of what MBSE is and readily available training resources [45]. As a matter of fact, many industry related solutions rely on proprietary tools that are customised for the company taken into account. Simulation is, for example, seen as a significant benefit of utilising model-based methods, and the most commonly used method for early V&V in the review. Nevertheless, it is seldom directly applicable for freely available tools, such as EMF [82] or Papyrus [34] and there are apparent issues with tooling interoperability hampering the use of languages such as SysML for simulation [66]. In this regard, the tooling-related aspects of model-based methods are a vital challenge for promoting and potentially adopting MBSE. To this end, there is a clear need to reshape the means and environments for beginners to learn and adopt the principles related to model-based practices. A potential path forward to address this challenge could be the alignment of education to match better industry needs [11], and education is currently considered poor when mapped to industrial MBSE needs [25].

8.2 Early V&V in the broader context of MBSE

The analysed papers demonstrate a wide array of solutions and means of performing the desired V&V. The observed variety is expected as a heterogeneous landscape exists among the papers regarding domain, application, and analysis' results of interest. However, considering the heterogeneous problem and solution space, the authors note a distinct lack of connection and discussion with related paradigms. For example, the usage of AI is presented in position and vision papers as part of the future in model-based practices and is expected to tackle many challenges [12, 14, 64]. However, there is an apparent lack of representation in the reviewed papers regarding such paradigms, which is unexpected considering how MBSE is positioned to move forward. More practically, AI, for example, could tackle the challenges related to uncertainty in early development [71], and applications like recommender systems that are seeing success in model-driven engineering, could assist novices with adoption [3]. There are plenty of examples of applications of AI in software-centric domains [13]. AI is foreseen to augment the already in-place technologies related to modelling, and the two paradigms are expected to have beneficial interplay. More evaluations of AI as part of the modelling and analysis activities would be an interesting research avenue for the MBSE community.

MBSE, or SE in general, detail methods and processes for managing systems throughout their whole life-cycle. When reviewing the papers, there is a missing discussion on propagating the artefacts developed at various life-cycle stages and how the SE life-cycle relates to the early V&V activities. MBSE is aimed for the entire SE life-cycle [88], so the apparent lack of focus on these aspects in the reviewed papers is surprising. There are examples in the literature of models aimed for all parts of the SE life-cycle [28, 59], and discussions on how to manage models across the MBSE life-cycle [29]. However, managing these models is not trivial, and consistency management is challenging to manage for industrial systems [49]. Creating models for analysis at early stages incorporates uncertainty and often high levels of abstraction. Without propagation of the uncertainty or evolution/refinement of early models to the later stages of development, there is a significant chance that such models are made and discarded, which is typically referred to as a weakness of document-based development and might work against re-use [36]. Therefore, if the models created for early V&V are not used in a larger context, the potential benefits of model-centric development are partially missed as the workflow would resemble the traditional document-centric development. Notably, it would be difficult to manage the uncertainties introduced in the early phases and how those would propagate to the rest of the process.

Possible solutions for more robust propagation of artefacts in the SE life-cycle can also be related to agile MBSE [76] and DevOps [62]. These paradigms are seen as enablers for the digital thread [80] and are seeing growing interests from industry. Incorporating and improving agility for the development process is well in line with the INCOSE vision [64], and initiatives to merge the paradigms of model-based engineering and DevOps are seeing some initial success [23, 44]. DevOps might be an enabler for connecting models at different stages of

Just Accepted

development and introducing increased automation in the workflows. The notion of DevOps is also heavily tied to the way of working and company culture, ideally breaking down potential silos found during development [26]. Another related aspect is using Digital Twins (DTs) [48]: utilising a DT is seen as a powerful enabler during design and as a means of integrating analytics past the system implementation. Similar to the use of DevOps, DT could bridge the gap between silos in development.

8.3 Empirical measurements and benchmarks in MBSE

In the data extraction for the selected papers we distinguish between empirical measurements and running examples. In this respect we observe a lack of proper benchmarks and empirical evidence or evaluation of proposed workflows, methods, and frameworks, even in the case of empirical measurements. In particular, there is no benchmark or similar reference measure for the type of solutions under study, and typically the authors employ ad-hoc or case-specific metrics. Moreover, the measurements are usually limited to the quality attributes of the proposed V&V solution, while attempts to measure the effectiveness of the solutions in the MBSE context are missing. Critical reviews of the perceived value of MBSE find that only around 10% of the benefits originating from the adoption of MBSE is observed against any metric [16, 41], and the low portion of measured evaluations is similarly reflected in this review at around 11%.

The expected benefits of employing MBSE reported by seminal papers are minimising errors due to manual steps [42], management of complexity [87], analysis and testing [24], and improvement of communication, traceability, and consistency [89]. However, as previously mentioned, these benefits are primarily argued instead of demonstrated via measurements or observations. As Henderson and Salado note [41], the benefits could very well be valid but lack in rigorous proof. A risk noted by Campo *et al.* [16] with this lack of proof is that the positive aspects might be overemphasised, especially since MBSE is positioned as *the future of systems engineering*. In this regard, more solid empirical observations of MBSE adoption would greatly assist in breaking down and quantifying the current challenges and benefits. However, measuring the benefits and challenges can be difficult, particularly in industrial settings, since empirical measurements might not make it to potential dissemination due to confidentiality concerns from the company. Moreover, it could be very difficult to limit bias issues, since MBSE adoption would impact the development process as a whole. Nevertheless, the lack of proper metrics for reporting on the use of MBSE limits the impact of the research as it is often limited to anecdotal examples or observations, which are difficult to put in a broader context.

Eventually, Garousi *et al.* [33] performed a literature review on the challenges of industry and academia collaboration and found that common challenges relate to a lack of applicability of academic solutions in industrial contexts and different perceptions of valuable solutions. Additionally, Garousi *et al.* highlight that the best practice for collaboration is to base research on real-world problems [33]. However, although many challenges are identified in industry [19, 24, 75], there is still a considerable gap in the research and practice. In this regard, a possible solution might be to concretise industrial needs related to MBSE through more explicit requirements on solutions developed by academics. Such type of concrete requirements can be found for other domains such as Wireless Sensor Networks [68], Augmented Reality [72], and Internet of Things security [84]. Defining more rigorous requirements on solutions for MBSE from an industrial perspective could be a means of closing some of the gaps currently observed and would also provide means to measure the developed solutions. More recently, there has been attempts to work towards metrics for the wider area of Digital Engineering, although it requires further investigation and eventual translation to MBSE [40].

8.4 Barriers for industrial adoption of Early V&V in MBSE

Leveraging the results from Section 6, 7, and the discussion proposed so far, we present a summary of challenges in Figure 20. We have grouped the challenges into the areas of *Model-Based*, *Systems Engineering*, and *Validation*

Just Accepted

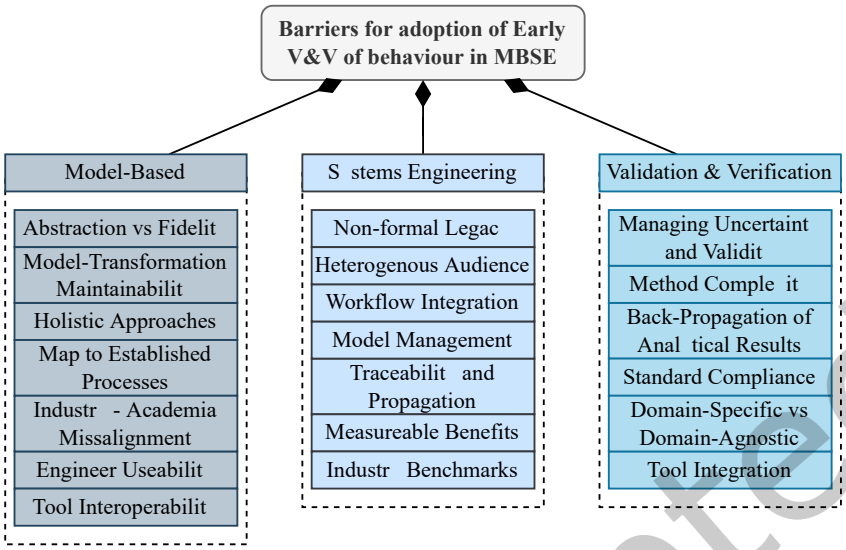


Fig. 20. Challenges for industry adoption of Early V&V for system behaviour in MBSE.

& Verification. Particularly, these challenges relate to the industrial adoption of MBSE, and the current inhibitors we see from the analysis in the observed papers. We have chosen the presented categories based on the data extraction in conjunction with the motivation example presented in Section 2.1.

8.4.1 Model-Based. The main artefacts of MBSE, the models themselves, need to fulfil certain criteria to be *useful*. For example, the models must have adequate abstraction and address different stakeholder concerns depending on the context while keeping enough rigour to facilitate analysis. Finding the correct balance between abstraction and fidelity is a re-occurring challenge in the papers and is often a pre-requisite for enabling mature early V&V in the context of MBSE. Overcoming the gap of abstraction and fidelity often results in model transformations from one notation for description to one or more other notations for analysis, seen in the large heterogeneity in RQ2 and RQ3. While these are valid techniques, the introduced complexity makes the MBSE adoption more difficult for new practitioners. Particularly, the need to maintain and support model transformations in changing environments regarding tools, languages, users, and the systems themselves introduces technical complexity and usability concerns, seen in literature [60, 83] and reconfirmed by RQ5. Little attention is as well paid to the holistic nature of MBSE. Most solutions are situated in very narrow applications that do not consider the surrounding development or overall system development, noticeable in the extraction from RQ1. In the same fashion, solutions often follow a particular working process. Many solutions enforce a particular process on the users, which can often clash with engineers and workflows already in place, again seen in RQ3 in the techniques and tools. Indeed, the practitioners themselves are mostly ignored from the papers, and the overall usability of solutions is not a priority. There is seemingly no clear view of how this can be addressed, perhaps due to the wide application in area in many domains seen in RQ4. Additionally, model-based practices are largely heavily tool-dependent, and many traditional issues such as interoperability, automation, and scalability are encountered in the extraction which reconfirms previous work in the field [11, 25]. In addition to the explicit challenges (still) existing, we also notice a distinct lack of discussions on academic tool integration with more significant tooling landscapes, observable in RQ3. As expected the challenges overlap, but many of the academic solutions observed suffer from a very narrow target with unique tools and languages when the industry is increasingly moving

Just Accepted

towards standards such as OSLC[15] to promote more standardised data exchanges. Additionally, there seems to be a gap between the industry practitioners and the academic researchers, and there is a risk that the tools developed for academia are aiming to solve other problems compared to industry needs.

8.4.2 Systems Engineering. As discussed in Section 2.1, a large part of the industry utilising SE is undergoing a transformation towards at least partial MBSE adoption. However, the legacy is often substantial and cannot simply be discarded in favour of new methods or working methods, yet there is little discussion on how to facilitate such integration in the RQ3 data extraction. Meaningful integration of legacy formats, such as office tools, drawings, and text, must be incorporated in solutions to be viable on a larger scale. Similarly, the audience of SE is heterogeneous, and the discussed context of *early* is not strictly defined. A wide audience is expected to leverage early V&V. For example, a common solution pattern in the RQ2 and RQ3 analysis is engineers leveraging SysML to orchestrate simulations without the need for simulation knowledge. However, the solutions are often designed with an expert audience in mind, which limits the wider applicability of solutions (particularly since many engineers are complete novices regarding Model-Based methods). Furthermore, the extracted data hints at a limited integration in workflows to promote the integration of models across development, seen through the authors properties of interest and their usage in RQ3 and limitations in RQ5. Many solutions are limited in extension to surrounding activities, e.g., a solution lacks up and downstream linkage. Similarly, there is little discussion on how analysis models should be managed to facilitate re-use and efficient future decision-making, noticeable in the intersection of RQ1 and RQ3. Without a means of classifying models or results for future use, the value of approaches becomes heavily reduced. Similarly, traceability, and more recently, the digital thread, are largely omitted from any discussions thus reducing their applicability. Indeed, there needs to be propagation of results and their impact across the development to maximise the benefits of the analysis. As mentioned previously in the section, a significant barrier to adoption is the difficulty of providing measurable benefits of MBSE approaches. Often, the solutions' value is argued based on anecdotal examples or rough estimates, which hinders the rigorous evaluation of solutions, visible in the RQ4 extraction. In the same fashion, when measurements are performed in some way, no proper benchmarks can be used to position papers in the wider context. A more common view of benchmarks and metrics for SE at large would help reduce some issues related to the measurement and evaluation of solutions.

8.4.3 Validation & Verification. A large gap in the solutions is the discussion of uncertainty and analysis validity, also recognised among limitations in RQ5. In the early stages of development, uncertainty is present in all models and requires considerable efforts to mitigate and reason about so that results can be correctly positioned and used for decision-making. However, there is rarely any discussion on how uncertainty can be managed in a solution and what implications the analysis abstraction introduces, which is one of the more prominent limitations reported by authors in RQ5. The same is true for analysis validity. If re-use is of interest, there is a need to catalogue when a method or model can be used for correct analysis, and how a user might be able to determine on a case basis when the considered models are rich *enough* to yield valid analysis results. In addition, the methods themselves are often complex and can introduce a burden on the user to incorporate effectively the methods "as is". Furthermore, the back-propagation of analytical results to the engineers, or descriptive models, is generally missing from the solutions, and results are often simply presented in the analytical tools presented in RQ3. In particular, how analytical results should affect decision-making, how the analytical results should be translated back to the system design directly, and how these results can be leveraged in industrial processes is rarely part of the methods of authors. For example, the notion of standards should be included in articles which guide practitioners during development, and how the proposed solutions can support standard processes or certification activities, and is inline with the major extraction for RQ1. While research might not be

[15]The OSLC specification is available at: https://open-services.net/

Just Accepted

expected to produce technology with industrial readiness, it still needs to consider the context, which is mostly standard-based and standard-driven. The analysis methods are generally domain-specific and cater to specific needs as per RQ4 extraction, and often implicit domain knowledge is a prerequisite for useful analytical results. Nonetheless, the general lack of domain-agnostic and more general solutions inhibits the adoption of existing solutions due to the difficulty of transferring those solutions to different domains as the implicit knowledge is hidden or not valid across different domains. In the same fashion, many solutions utilise custom tools for the implementation, limiting integration in SE processes. Particularly, relying on several specific tools increases the process complexity and can lead to increased vendor lock-in.

9 CONCLUSIONS

This article reports the results of a systematic literature review on early behaviour validation and verification in model-based systems engineering. From a set of 701 papers retrieved through searches and snowballing activities we selected 149 relevant contributions, we extracted and coded the obtained data, and we performed analyses whose results and findings are presented in the work. In this respect, we notice an increased interest in performing early V&V and observe a broad range of domains in the analysed papers, with a corresponding variety of methods, tools, and languages. Further, we note a strong industrial presence in the literature and several industrial perspective trends that differ from the academic ones. To name a few of our findings, we note that SysML is the most represented language in industry and academia for describing system behaviour. In contrast, the language or formalism for analysis varies between most solutions. Additionally, several limitations are identified, indicating a lack of readiness for the solutions together with the concerns about managing analysis with low-fidelity models. Finally, a significant divide emerges between the academic and industrial implementation of solutions; such a divide is especially observable for SysML, utilised across all contexts, but relying on different tooling for the contexts.

We contextualise the review findings and discuss the current status of early validation of system behaviour in the context of industrial MBSE adoption. The review is structured according to the needs of the industry to promote the eventual adoption of early V&V and MBSE processes at large. The review provides actionable insights for the five presented research questions to promote further investigation into this area. Furthermore, we distinguish three areas, *Model-Based*, *Systems Engineering*, and *Validation & Verification*, and highlight a set of corresponding barriers for each area, which we feel need to be addressed in order to promote and support industrial adoption of early V&V techniques. As such, we hope the findings of this review can provide an adequate state-of-the-art view and pave the way for future investigations for researchers and practitioners.

Just Accepted

PRIMARY STUDIES

[P1] Weikai Miao, Qianqian Yan, Yihao Huang, Jincao Feng, and Hanyue Zheng. 2019. A Domain Experts Centric Approach to Formal Requirements Modeling and V&V of Embedded Control Software. *IEEE*, 15–22. DOI:https://doi.org/10.1109/APSEC48747.2019.00012

[P2] Shaofan Zhu, Jian Tang, Jean-Marie Gauthier, and Raphaël Faudou. 2019. A formal approach using SysML for capturing functional requirements in avionics domain. *Chinese Journal of Aeronautics* 32, 12 (2019), 2717–2726. DOI:https://doi.org/10.1016/j.cja.2019.03.037

[P3] Tom Mens, Alexandre Decan, and Nikolaos I. Spanoudakis. 2019. A method for testing and validating executable statechart models. *Softw Syst Model* 18, 2 (2019), 837–863. DOI:https://doi.org/10.1007/s10270-018-0676-3

[P4] Eun-Young Kang, Eduard Paul Enoiu, Raluca Marinescu, Cristina Seceleanu, Pierre-Yves Schobbens, and Paul Pettersson. 2013. A methodology for formal analysis and verification of EAST-ADL models. *Reliability Engineering & System Safety* 120, (2013), 127–138. DOI:https://doi.org/10.1016/j.ress.2013.06.007

[P5] Ermeson Andrade, Paulo Maciel, Gustavo Callou, and Bruno Nogueira. 2009. A Methodology for Mapping SysML Activity Diagram to Time Petri Net for Requirement Validation of Embedded Real-Time Systems with Energy Constraints. *IEEE*, 266–271. DOI:https://doi.org/10.1109/ICDS.2009.19

[P6] Messaoud Rahim, Ahmed Hammad, and Malika Ioualalen. 2017. A methodology for verifying SysML requirements using activity diagrams. *Innovations Syst Softw Eng* 13, 1 (2017), 19–33. DOI:https://doi.org/10.1007/s11334-016-0281-y

[P7] Alfredo Garro and Andrea Tundis. 2012. A Model-Based Method for System Reliability Analysis. *Society for Computer Simulation International.*

[P8] Gan Wang and Saulius Pavalkis. 2019. A Model-Based V&V Test Strategy Based on Emerging System Modeling Techniques. *INCOSE International Symposium* 29, 1 (2019), 771–787. DOI:https://doi.org/10.1002/j.2334-5837.2019.00634.x

[P9] Faleeha Moin, Farooque Azam, and Muhammad Waseem Anwar. 2018. A Model-driven Approach for Formal Verification of Embedded Systems Using Timed Colored Petri Nets. *IEEE*, 2580–2584. DOI:https://doi.org/10.1109/CompComm.2018.8780731

[P10] Paolo Bocciarelli, Andrea D'Ambrogio, Alberto Falcone, Alfredo Garro, and Andrea Giglio. 2019. A Model-Driven Approach to Enable the Simulation of Complex Systems on Distributed Architectures. *Simulation* 95, 12 (2019), 1185–1211. DOI:https://doi.org/10.1177/0037549719829828

[P11] M.W. Anwar, M. Rashid, F. Azam, M. Kashif, and W.H. Butt. 2019. A model-driven framework for design and verification of embedded systems through SystemVerilog. *Design Automation for Embedded Systems* 23, 3–4 (2019), 179–223. DOI:https://doi.org/10.1007/s10617-019-09229-y

[P12] Faïda Mhenni, Jean-Yves Choley, Olivia Penas, Régis Plateaux, and Moncef Hammadi. 2014. A SysML-Based Methodology for Mechatronic Systems Architectural Design. *Adv. Eng. Inform.* 28, 3 (2014), 218–231. DOI:https://doi.org/10.1016/j.aei.2014.03.006

[P13] L. Alawneh, M. Debbabi, F. Hassaine, Y. Jarraya, and A. Soeanu. 2006. A unified approach for verification and validation of systems and software engineering models. *IEEE*, 10-p. DOI:https://doi.org/10.1109/ECBS.2006.17

[P14] Muhammad Waseem Anwar, Muhammad Rashid, Farooque Azam, Aamir Naeem, Muhammad Kashif, and Wasi Haider Butt. 2020. A Unified Model-Based Framework for the Simplified Execution of Static and Dynamic Assertion-Based Verification. *IEEE* Access 8, (2020), 104407–104431. DOI:https://doi.org/10.1109/ACCESS.2020.2999544

[P15] R. Delmas, A. F. Pires, and T. Polacsek. 2013. A verification and validation process for model-driven engineering. *EDP Sciences*, 455–468. DOI:https://doi.org/10.1051/eucass/201306455

[P16] Jing Liu, Tengfei Li, Zuohua Ding, Yuqing Qian, Haiying Sun, and Jifeng He. 2019. AADL+: A Simulation-Based Methodology for Cyber-Physical Systems. *Front. Comput. Sci.* 13, 3 (2019), 516–538. DOI:https://doi.org/10.1007/s11704-018-7039-7

[P17] Danielle Stewart, Jing (Janet) Liu, Darren Cofer, Mats Heimdahl, Michael W. Whalen, and Michael Peterson. 2021. AADL-Based safety analysis using formal methods applied to aircraft digital systems. *Reliability Engineering & System Safety* 213, (2021), 107649. DOI:https://doi.org/10.1016/j.ress.2021.107649

[P18] Stefan Björnander, Cristina Seceleanu, Kristina Lundqvist, and Paul Pettersson. 2011. ABV - A Verifier for the Architecture Analysis and Design Language (AADL). *IEEE*, 355–360. DOI:https://doi.org/10.1109/ICECCS.2011.43

[P19] Ana Rugina, Cristiano Leorato, and Elena Tremolizzo. 2012. Advanced Validation of Overall Spacecraft Behaviour Concept Using a Collaborative Modelling and Simulation Approach. *IEEE*, 262–267. DOI:https://doi.org/10.1109/WETICE.2012.37

[P20] Erwan Bousse, David Mentré, Benoît Combemale, Benoît Baudry, and Takaya Katsuragi. 2012. Aligning SysML with the B method to provide V&V for systems engineering. *ACM Press*, 11–16. DOI:https://doi.org/10.1145/2427376.2427379

[P21] Michel Batteux, Tatiana Prosvirnova, and Antoine B Rauzy. 2019. AltaRica 3.0 in ten modelling patterns. *International Journal of Critical Computer-Based Systems* 9, 1–2 (2019), 133–165.

[P22] Aurelijus Morkevicius and Nerijus Jankevicius. 2015. An approach: SysML-based automated requirements verification. *IEEE*, 92–97. DOI:https://doi.org/10.1109/SysEng.2015.7302739

[P23] Eamonn Linehan and Siobhán Clarke. 2012. An Aspect-Oriented, Model-Driven Approach to Functional Hardware Verification. *J. Syst. Archit.* 58, 5 (2012), 195–208. DOI:https://doi.org/10.1016/j.sysarc.2011.02.001

[P24] Nissaf Fredj, Yessine Hadj Kacem, and Mohamed Abid. 2021. An event-based approach for formally verifying runtime adaptive real-time systems. *J Supercomput* 77, 3 (2021), 3110–3143. DOI:https://doi.org/10.1007/s11227-020-03386-9

[P25] Lucas Lima, Alvaro Miyazawa, Ana Cavalcanti, Márcio Cornélio, Juliano Iyoda, Augusto Sampaio, Ralph Hains, Adrian Larkham, and Vaughan Lewis. 2017. An Integrated Semantics for Reasoning about SysML Design Models Using Refinement. *Softw. Syst. Model.* 16, 3 (2017), 875–902. DOI:https://doi.org/10.1007/s10270-015-0492-y

[P26] J. Markovski. 2013. An integrated systems engineering framework for supervisor synthesis, verification, and performance evaluation. 650–657. DOI:https://doi.org/10.23919/ecc.2013.6669190

[P27] M. Li, J. Xu, K. Yang, and B. Ge. 2018. An interactive model-driven simulation approach for dynamic behavior analysis in armed conflicts. *IEEE Access* 6, (2018), 36744–36756. DOI:https://doi.org/10.1109/ACCESS.2018.2852803

[P28] L. Lemazurier, V. Chapurlat, and A. Grossetête. 2017. An MBSE Approach to Pass from Requirements to Functional Architecture. *IFAC-PapersOnLine* 50, 1 (2017), 7260–7265. DOI:https://doi.org/10.1016/j.ifacol.2017.08.1376

[P29] Mustafa Al-Lail, Wuliang Sun, and Robert B. France. 2014. Analyzing Behavioral Aspects of UML Design Class Models against Temporal Properties. *IEEE*, 196–201. DOI:https://doi.org/10.1109/QSIC.2014.56

[P30] S.J.I. Herzig, R. Karban, G. Trancho, F.G. Dekens, N. Jankevificius, and M. Troy. 2017. Analyzing the operational behavior of the alignment and phasing system of the thirty meter telescope using SysML. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85049217630&partnerID=40&md5=b61fa6283e3cc4104311aef4158911ba

Just Accepted

[P31] S. Gebreyohannes, A. Karimoddini, and A. Homaifar. 2020. Applying model-based systems engineering to the development of a test and evaluation tool for unmanned autonomous systems. DOI:https://doi.org/10.1109/SysCon47679.2020.9275894

[P32] Gereon Weiss, Marc Zeller, Dirk Eilers, and Rudi Knorr. 2010. Approach for Iterative Validation of Automotive Embedded Systems. (2010), 15.

[P33] Awele I. Anyanhun, David N. Amanor, and William W. Edmonson. 2021. Architecting an MBSE Black-Box System Model for the Physical Layer of a Visible Light Intersatellite Communication System. *IEEE J. Miniat. Air Space Syst.* 2, 4 (2021), 168–178. DOI:https://doi.org/10.1109/JMASS.2021.3069826

[P34] Chih-Hong Cheng, Yassine Hamza, and Harald Ruess. 2017. Automated Analysis of Multi-View Software Architectures. *IEEE*, 725–730. DOI:https://doi.org/10.1109/APSEC.2017.93

[P35] Yilong Yang, Xiaoshan Li, Wei Ke, and Zhiming Liu. 2020. Automated Prototype Generation From Formal Requirements Model. *IEEE Trans. Rel.* 69, 2 (2020), 632–656. DOI:https://doi.org/10.1109/TR.2019.2934348

[P36] Maziar Mahani, Denise Rizzo, Chris Paredis, and Yue Wang. 2021. Automatic Formal Verification of SysML State Machine Diagrams for Vehicular Control Systems. 2021–01. DOI:https://doi.org/10.4271/2021-01-0260

[P37] Zhonglei Wang, Wolfgang Haberl, Stefan Kugele, and Michael Tautschnig. 2008. Automatic generation of SystemC models from component-based designs for early design validation and performance analysis. In *Proceedings of the 7th international workshop on software and performance,* 139–144.

[P38] Eiji Morinaga, Hidefumi Wakamatsu, Hijiri Abiru, and Eiji Arai. 2017. Behavior modeling method for functional verification of product considering ways of usage. *JAMDSM* 11, 5 (2017), JAMDSM0066–JAMDSM0066. DOI:https://doi.org/10.1299/jamdsm.2017jamdsm0066

[P39] Kui Zhang, Ji Wu, Chao Liu, Syed Sarmad Ali, and Jian Ren. 2019. Behavior Modeling on ARINC653 to Support the Temporal Verification of Conformed Application Design. *IEEE Access* 7, (2019), 23852–23863. DOI:https://doi.org/10.1109/ACCESS.2019.2895996

[P40] Nafiseh Kahani and James R. Cordy. 2020. Bounded Verification of State Machine Models. *ACM*, 23–32. DOI:https://doi.org/10.1145/3419804.3420263

[P41] Muhammad Waseem Anwar, Shumaila Qamar, Farooque Azam, Wasi Haider Butt, and Muhammad Rashid. 2020. Bridging the Gap between Design and Verification of Embedded Systems in Model Based System Engineering: A Meta-Model for Modeling Universal Verification Methodology (UVM) Test Benches. *Association for Computing Machinery*, 82–87. DOI:https://doi.org/10.1145/3408066.3408069

[P42] C. Duhil, J.-P. Babau, E. Lepicier, J.-L. Voirin, and J. Navas. 2020. Chaining model transformations for system model verification: Application to verify capella model with simulink. 279–286. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85082994857&partnerID=40&md5=8e14eb0ff803182ecd82623fdf5ff7ed

[P43] Sanford Friedenthal, Alan Moore, and Rick Steiner. 2008. Chapter 15 - Water Distiller Example Using Functional Analysis. *In The MK/OMG Press.* Morgan Kaufmann, Burlington, 357–396. DOI:https://doi.org/10.1016/B978-0-12-374379-4.00015-1

[P44] F. Bouffaron, P. Marange, and G. Morel. 2014. Checking models based on an iterative co-specification process of a critical system. 248–254. DOI:https://doi.org/10.1109/INDIN.2014.6945516

[P45] X. Zhang, S. Zhang, and J. Yan. 2021. Civil Aircraft Auto Brake System Development Using Model-Based Systems Engineering. 400–405. DOI:https://doi.org/10.1109/CAC53003.2021.9728152

[P46] X. Hai, S. Zhang, and X. Xu. 2017. Civil aircraft landing gear brake system development and evaluation using model based system engineering. 10192–10197. DOI:https://doi.org/10.23919/ChiCC.2017.8028981

[P47] Anjelika Votintseva, Petra Witschel, Nikolaus Regnat, and Philipp Emanuel Stelzig. 2012. Comparative Study of Model-Based and Multi-Domain System Engineering Approaches for Industrial Settings. *Springer-Verlag*, 20–31. DOI:https://doi.org/10.1007/978-3-642-31491-9_4

[P48] Y.-M Deng, G.A Britton, and S.B Tor. 2000. Constraint-based functional design verification for conceptual design. Computer-Aided Design 32, 14 (2000), 889–899. DOI:https://doi.org/10.1016/S0010-4485(00)00077-4

[P49] J. Liu, C. Zhuang, Z. Liu, and T. Miao. 2021. Construction method of shop-floor digital twin based on MBSE. *Journal of Manufacturing Systems* 60, (2021), 93–118. DOI:https://doi.org/10.1016/j.jmsy.2021.05.004

[P50] Salvador Trujillo, Jose Miguel Garate, Roberto Erick Lopez-Herrejon, Xabier Mendialdua, Albert Rosado, Alexander Egyed, Charles W. Krueger, and Josune de Sosa. 2010. Coping with Variability in Model-Based Systems Engineering: An Experience in Green Energy. *Springer-Verlag*, 293–304. DOI:https://doi.org/10.1007/978-3-642-13595-8_23

[P51] R. Karban, F.G. Dekens, S. Herzig, M. Elaasar, and N. Jankevicius. 2016. Creating system engineering products with executable models in a model-based engineering environment. DOI:https://doi.org/10.1117/12.2232785

[P52] Harald Bucher, Jürgen Becker, and Simon Kamm. 2019. Cross-Layer Behavioral Modeling and Simulation of E/E-Architectures Using Preevision and Ptolemy II. *Society for Computer Simulation International.*

[P53] Alfredo Garro, Vittorio Vaccaro, Stefan Dutré, and Jef Stegen. 2019. Cyber-Physical Systems Engineering: Model-Based Solutions. *Society for Computer Simulation International.*

[P54] Ronan Baduel, Jean-Michel Bruel, Iulian Ober, and Eddy Doba. 2018. Definition of states and modes as general concepts for system design and validation. (2018).

[P55] D. Kaslow, B. Ayres, P.T. Cahill, L. Hart, and R. Yntema. 2017. Developing a CubeSat Model-Based System Engineering (MBSE) reference model - Interim status #3. DOI:https://doi.org/10.1109/AERO.2017.7943691

Just Accepted

[P56] Francisco Durán, Manuel Roldán, Antonio Moreno, and José María Álvarez. 2014. Dynamic Validation of Maude Prototypes of UML Models. In *Specification, Algebra, and Software*. Springer Berlin Heidelberg, Berlin, Heidelberg, 212–228. Retrieved from http://link.springer.com/10.1007/978-3-642-54624-2_11

[P57] Manzoor Ahmad, Iulia Dragomir, Jean-Michel Bruel, Iulian Ober, and Nicolas Belloir. 2013. Early analysis of ambient systems sysml properties using omega2-ifx. In *SIMULTECH* 2013.

[P58] Christophe Duhil, Jean-Luc Voirin, Eric Lépicier, and Jean-Philippe Babau. 2020. Early Detection of Flaws in System Architecture Model by means of Model Simulation. *INCOSE International Symposium* 30, 1 (2020), 1758–1769. DOI:https://doi.org/10.1002/j.2334-5837.2020.00817.x

[P59] Veronika Brandstetter, Andreas Froese, Bastian Tenbergen, Andreas Vogelsang, Jan Christoph Wehrstedt, and Thorsten Weyer. 2015. Early Validation of Automation Plant Control Software using Simulation Based on Assumption Modeling and Validation Use Cases. *CSIMQ* 4 (2015). DOI:https://doi.org/10.7250/csimq.2015-4.04

[P60] Emmanouela Stachtiari, Anastasia Mavridou, Panagiotis Katsaros, Simon Bliudze, and Joseph Sifakis. 2018. Early validation of system requirements and design through correctness-by-construction. *Journal of Systems and Software* 145, (2018), 52–78. DOI:https://doi.org/10.1016/j.jss.2018.07.053

[P61] Joe Gregory, Lucy Berthoud, Theo Tryfonas, and Antonio Prezzavento. 2019. Early Validation of the Data Handling Unit of a Spacecraft Using MBSE. *IEEE*, 1–15. DOI:https://doi.org/10.1109/AERO.2019.8741767

[P62] Carlos A. González, Mojtaba Varmazyar, Shiva Nejati, Lionel C. Briand, and Yago Isasi. 2018. Enabling Model Testing of Cyber-Physical Systems. *Association for Computing Machinery*, 176–186. DOI:https://doi.org/10.1145/3239372.3239409

[P63] Robert Karban, Nerijus Jankevičius, and Maged Elaasar. 2016. ESEM: Automated Systems Analysis using Executable SysML Modeling Patterns. *INCOSE International Symposium* 26, 1 (2016), 1–24. DOI:https://doi.org/10.1002/j.2334-5837.2016.00142.x

[P64] Zhixue Wang, He hongyue, and Qingglong Wang. 2014. Executable Architecture Modeling and Simulation Based on fUML.

[P65] Renzhong Wang and Cihan H. Dagli. 2011. Executable system architecting using systems modeling language in conjunction with colored Petri nets in a model-driven systems development process. *Syst. Engin.* 14, 4 (2011), 383–409. DOI:https://doi.org/10.1002/sys.20184

[P66] Radoslaw Klimek and Piotr Szwed. 2010. Formal analysis of use case diagrams. Computer Science 11, (2010), 115–131.

[P67] Marco Bozzano, Roberto Cavada, Alessandro Cimatti, J-P Katoen, V Nguyen, Thomas Noll, and Xavier Olive. 2010. Formal verification and validation of AADL models. In *ERTS2 2010, Embedded Real Time Software & Systems*.

[P68] Luciano Baresi, Gundula Blohm, Dimitrios S. Kolovos, Nicholas Matragkas, Alfredo Motta, Richard F. Paige, Alek Radjenovic, and Matteo Rossi. 2015. Formal verification and validation of embedded systems: the UML-based MADES approach. *Softw Syst Model* 14, 1 (2015), 343–363. DOI:https://doi.org/10.1007/s10270-013-0330-z

[P69] Fernando Silvano Goncalves, David Pereira, Eduardo Tovar, and Leandro Buss Becker. 2017. Formal Verification of AADL Models Using UPPAAL. *IEEE*, 117–124. DOI:https://doi.org/10.1109/SBESC.2017.22

[P70] Habibi Husain Arifin, Yu Dong, Ho Kit Robert Ong, Yaoying Gu, Nasis Chimplee, and Wu Daphne. 2020. Hatley-Pirbhai Control Flow Diagram with SysML for Early Validation. *INCOSE International Symposium* 30, 1 (2020), 50–64. DOI:https://doi.org/10.1002/j.2334-5837.2020.00707.x

[P71] Messaoud Rahim, Malika Boukala-Ioualalen, and Ahmed Hammad. 2021. Hierarchical Colored Petri Nets for the Verification of SysML Designs- Activity-Based Slicing Approach. In *Advances in Computing Systems and Applications*. Springer International Publishing, Cham, 131–142. Retrieved from http://link.springer.com/10.1007/978-3-030-69418-0_12

[P72] R. Promyoo, S. Alai, and H. El-Mounayri. 2019. Innovative digital manufacturing curriculum for industry 4.0. 1043–1050. DOI:https://doi.org/10.1016/j.promfg.2019.06.092

[P73] D. Kaslow, G. Soremekun, H. Kim, and S. Spangelo. 2014. Integrated model-based systems engineering (MBSE) applied to the Simulation of a CubeSat mission. 1–14. DOI:https://doi.org/10.1109/AERO.2014.6836317

[P74] A.S. Dalvi, A. Razban, H. El-Mounyari, T. El-Mekkawy, and R. Promyoo. 2020. Integrated system model of district cooling for energy consumption optimization. 1722–1732. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85095751692&partnerID=40&md5=fc7d2333c806a0f81e28d3be0023caa5

[P75] Jörg Holtmann, Ruslan Bernijazov, Matthias Meyer, David Schmelter, and Christian Tschirner. 2015. Integrated Systems Engineering and Software Requirements Engineering for Technical Systems. *Association for Computing Machinery*, 57–66. DOI:https://doi.org/10.1145/2785592.2785597

[P76] Timo Vepsäläinen and Seppo Kuikka. 2014. Integrating Model-in-the-Loop Simulations to Model-Driven Development in Industrial Control. *Simulation* 90, 12 (2014), 1295–1311. DOI:https://doi.org/10.1177/0037549714553229

[P77] M. Friedl, A. Kellner, and L. Weingartner. 2017. Integration of domain-specific simulation models into descriptive system models by using SysML. 1–5. DOI:https://doi.org/10.1109/SysEng.2017.8088256

[P78] J. Ding, M. Reniers, J. Lu, G. Wang, L. Feng, and D. Kiritsis. 2021. Integration of modeling and verification for system model based on KARMA language. 41–50. DOI:https://doi.org/10.1145/3486603.3486775

[P79] Y. Cao, Y. Liu, and C.J.J. Paredis. 2010. Integration of system-level design and analysis models of mechatronic system behavior based on SysML and Simscape. 1099–1108. DOI:https://doi.org/10.1115/DETC2010-28213

Just Accepted

[P80] Vickram Singh and Gerrit Muller. 2013. Knowledge Capture, Cross Boundary Communication and Early Validation with Dynamic A3 Architectures. *INCOSE International Symposium* 23, 1 (2013), 84–97. DOI:https://doi.org/10.1002/j.2334-5837.2013.tb03005.x

[P81] Imran R. Quadri, Etienne Brosse, Ian Gray, Nicholas Matragkas, Leandro Soares Indrusiak, Matteo Rossi, Alessandra Bagnato, and Andrey Sadovykh. 2012. MADES FP7 EU project: Effective high level SysML/MARTE methodology for real-time and embedded avionics systems. *IEEE*, 1–8. DOI:https://doi.org/10.1109/ReCoSoC.2012.6322882

[P82] A.M. Madni. 2021. Mbse testbed for rapid, cost-effective prototyping and evaluation of system modeling approaches. *Applied Sciences (Switzerland)* 11, 5 (2021), 1–19. DOI:https://doi.org/10.3390/app11052321

[P83] Xiaopu Huang, Qingqing Sun, Jiangwei Li, and Tian Zhang. 2013. MDE-Based Verification of SysML State Machine Diagram by UPPAAL. In *Trustworthy Computing and Services*. Springer Berlin Heidelberg, Berlin, Heidelberg, 490–497. Retrieved from http://link.springer.com/10.1007/978-3-642-35795-4_62

[P84] U. Farooq. 2017. Model based verification of Electronic Control Unit (ECU) in high lift systems: Verification and validation. DOI:https://doi.org/10.1109/SysEng.2017.8088318

[P85] L. Petnga and M. Austin. 2016. Model-based design and formal verification processes for automated waterway system operations. *Systems* 4, 2 (2016). DOI:https://doi.org/10.3390/systems4020023

[P86] Muhammad Waseem Anwar, Muhammad Rashid, Farooque Azam, and Muhammad Kashif. 2017. Model-Based Design Verification for Embedded Systems through SVOCL: An OCL Extension for SystemVerilog. *Des. Autom. Embedded Syst.* 21, 1 (2017), 1–36. DOI:https://doi.org/10.1007/s10617-017-9182-z

[P87] Peter Munk and Arne Nordmann. 2020. Model-Based Safety Assessment with SysML and Component Fault Trees: Application and Lessons Learned. *Softw. Syst. Model.* 19, 4 (2020), 889–910. DOI:https://doi.org/10.1007/s10270-020-00782-w

[P88] N. C. W. M. Braspenning, E. M. Bortnik, J. M. van de Mortel-Fronczak, and J. E. Rooda. 2008. Model-Based System Analysis Using Chi and Uppaal: An Industrial Case Study. *Comput. Ind.* 59, 1 (2008), 41–54. DOI:https://doi.org/10.1016/j.compind.2007.06.002

[P89] Paolo Bocciarelli, Andrea D'Ambrogio, Andrea Giglio, and Emiliano Paglia. 2019. Model-Driven Distributed Simulation Engineering. *IEEE Press*, 75–89.

[P90] Wladimir Schamai, Peter Fritzson, Christiaan J. J. Paredis, and Philipp Helle. 2012. ModelicaML Value Bindings for Automated Model Composition. *Society for Computer Simulation International.*

[P91] A. Falcone, A. Garro, and A. Tundis. 2014. Modeling and simulation for the performance evaluation of the on-board communication system of a metro train. 20–29. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84912102514&partnerID=40&md5=107cdf8df7d7649aeb7b2de6dfe7a604

[P92] R. Baduel, I. Ober, and J.-M. Bruel. 2020. Modeling and verification method for an early evaluation of systems of systems interactions. 1798–1805. DOI:https://doi.org/10.1145/3341105.3373944

[P93] Olivia Penas, Rgis Plateaux, Stanislao Patalano, and Moncef Hammadi. 2017. Multi-Scale Approach from Mechatronic to Cyber-Physical Systems for the Design of Manufacturing Systems. *Comput. Ind.* 86, C (2017), 52–69. DOI:https://doi.org/10.1016/j.compind.2016.12.001

[P94] Jean-Francois Castet, Matthew L. Rozek, Michel D. Ingham, Nicolas F. Rouquette, Seung H. Chung, J. Steven Jenkins, David A. Wagner, and Daniel L. Dvorak. 2015. Ontology and Modeling Patterns for State-Based Behavior Representation. *American Institute of Aeronautics and Astronautics*. DOI:https://doi.org/10.2514/6.2015-1115

[P95] R. Chen, C.-H. Chen, Y. Liu, and X. Ye. 2020. Ontology-based requirement verification for complex systems. *Advanced Engineering Informatics* 46, (2020). DOI:https://doi.org/10.1016/j.aei.2020.101148

[P96] Alfredo Garro and Andrea Tundis. 2014. RAMSAS4Modelica: A Simulation-Driven Method for System Dependability Analysis Centered on the Modelica Language and Related Tools. *Society for Computer Simulation International.*

[P97] J. Holtkötter, J. Michael, C. Henke, A. Trächtler, M. Bockholt, A. Möhlenkamp, and M. Katter. 2018. Rapid-Control-Prototyping as part of Model-Based Development of Heat Pump Dryers. 235–242. DOI:https://doi.org/10.1016/j.promfg.2018.06.033

[P98] Messaoud Rahim, Ahmed Kheldoun, Malika Boukala-Ioualalen, and Ahmed Hammad. 2015. Recursive ECATNets-based approach for formally verifying System Modelling Language activity diagrams. *IET softw.* 9, 5 (2015), 119–128. DOI:https://doi.org/10.1049/iet-sen.2014.0087

[P99] Alvaro Miyazawa, Pedro Ribeiro, Wei Li, Ana Cavalcanti, Jon Timmis, and Jim Woodcock. 2019. RoboChart: modelling and verification of the functional behaviour of robotic applications. *Softw Syst Model* 18, 5 (2019), 3097–3149. DOI:https://doi.org/10.1007/s10270-018-00710-z

[P100] C. Kotronis, A. Tsadimas, G.-D. Kapos, V. Dalakas, M. Nikolaidou, and D. Anagnostopoulos. 2017. Simulating SysML transportation models. 1674–1679. DOI:https://doi.org/10.1109/SMC.2016.7844478

[P101] Matthew Hause and James Hummell. 2012. Simulation of an Electrical Network and Control System in SysML. *Society for Computer Simulation International.*

[P102] E. Palachi, C. Cohen, and S. Takashi. 2013. Simulation of cyber physical models using SysML and numerical solvers. 671–675. DOI:https://doi.org/10.1109/SysCon.2013.6549954

[P103] Ralph Weissnegger, Markus Schuss, Christian Kreiner, Markus Pistauer, Kay Römer, and Christian Steger. 2016. Simulation-based Verification of Automotive Safety-critical Systems Based on EAST-ADL. *Procedia Computer Science* 83, (2016), 245–252. DOI:https://doi.org/10.1016/j.procs.2016.04.122

Just Accepted

[P104] Parastoo Delgoshaei and Mark Austin. 2012. Software Patterns for Traceability of Requirements to Finite State Machine Behavior: Application to Rail Transit Systems Design and Management. *INCOSE International Symposium* 22, 1 (2012), 2141–2155. DOI:https://doi.org/10.1002/j.2334-5837.2012.tb01463.x

[P105] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Panagiotis Katsaros, Konstantinos Mokos, Viet Yen Nguyen, Thomas Noll, Bart Postma, and Marco Roveri. 2014. Spacecraft early design validation using formal methods. *Reliability Engineering & System Safety* 132, (2014), 20–35. DOI:https://doi.org/10.1016/j.ress.2014.07.003

[P106] Daniel Aceituna, Hyunsook Do, and Seok-Won Lee. 2010. SQ̂(2)E: An Approach to Requirements Validation with Scenario Question. *IEEE*, 33–42. DOI:https://doi.org/10.1109/APSEC.2010.14

[P107] Y. Wu, G. Xiao, and M. Wang. 2021. State-based safety analysis method for dynamic evaluation of failure effect. *Aerospace Systems* 4, 1 (2021), 49–65. DOI:https://doi.org/10.1007/s42401-020-00073-8

[P108] J. Kößler and K. Paetzold. 2015. Support of the system integration with automatically generated behaviour models. 21–30. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84979760363&partnerID=40&md5=e8a502cad2e8966022f01335b4d9d085

[P109] Ronan Baduel, Mohammad Chami, Jean-Michel Bruel, and Iulian Ober. 2018. SysML Models Verification and Validation in an Industrial Context: Challenges and Experimentation. In *Modelling Foundations and Applications*. Springer International Publishing, Cham, 132–146. Retrieved from http://link.springer.com/10.1007/978-3-319-92997-2_9

[P110] Ludovic Apvrille, Pierre De Saqui-Sannes, Oana Hotescu, and Alessandro Calvino. 2022. SysML Models Verification Relying on Dependency Graphs: SCITEPRESS - Science and Technology Publications, 174–181. DOI:https://doi.org/10.5220/0010792900003119

[P111] Maysam Zoor, Ludovic Apvrille, and Renaud Pacalet. 2020. SysML Models: Studying Safety and Security Measures Impact on Performance Using Graph Tainting. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. Association for Computing Machinery, New York, NY, USA. Retrieved from https://doi.org/10.1145/3417990.3419225

[P112] Yue Cao, Yusheng Liu, Hongri Fan, and Bo Fan. 2013. SysML-based uniform behavior modeling and automated mapping of design and simulation model for complex mechatronics. *Computer-Aided Design* 45, 3 (2013), 764–776. DOI:https://doi.org/10.1016/j.cad.2012.05.001

[P113] Y. Cao, Y. Liu, and C.J.J. Paredis. 2011. System-level model integration of design and simulation for mechatronic systems based on SysML. *Mechatronics* 21, 6 (2011), 1063–1075. DOI:https://doi.org/10.1016/j.mechatronics.2011.05.003

[P114] C.F. Claver, B.M. Selvy, G. Angeli, F. Delgado, G. Dubois-Felsmann, P. Hascall, P. Lotz, S. Marshall, G. Schumacher, and J. Sebag. 2014. Systems engineering in the large synoptic survey telescope project: An application of model based systems engineering. DOI:https://doi.org/10.1117/12.2056781

[P115] Aymen Louati, Kamel Barkaoui, and Chadlia Jerad. 2015. Temporal Properties Verification of Real-Time Systems Using UML/MARTE/OCL-RT. In Formalisms for Reuse and Systems Integration. *Springer International Publishing*, Cham, 133–147. Retrieved from http://link.springer.com/10.1007/978-3-319-16577-6_6

[P116] Daniel Knorreck, Ludovic Apvrille, and Pierre de Saqui-Sannes. 2011. TEPE: a SysML language for time-constrained property modeling and formal verification. *SIGSOFT Softw. Eng. Notes* 36, 1 (2011), 1–8. DOI:https://doi.org/10.1145/1921532.1921556

[P117] J. Gregory, L. Berthoud, T. Tryfonas, and L. Faure. 2020. There's no "I" in SEAM - An Interim Report on the "Spacecraft Early Analysis Model." DOI:https://doi.org/10.1109/AERO47225.2020.9172702

[P118] Mustapha Salim Ghitri, Mohamed Messabihi, and Abdelkrim Benamar. 2019. Tooled approach for formal verification of components interactions modeled in SysML. *IEEE*, 1–7. DOI:https://doi.org/10.1109/ICTAACS48474.2019.8988134

[P119] Jean-Marie Gauthier, Fabrice Bouquet, Ahmed Hammad, and Fabien Peureux. 2015. Tooled Process for Early Validation of SysML Models Using Modelica Simulation. In *Fundamentals of Software Engineering*. Springer International Publishing, Cham, 230–237. Retrieved from http://link.springer.com/10.1007/978-3-319-24644-4_16

[P120] A. Berrachedi, M. Ioualalen, and A. Hammad. 2021. Towards the formal modeling methodology of WSN through the transformation of SysML into DSPNs. 83–91. DOI:https://doi.org/10.5220/0010549200830091

[P121] J. Lee, Jiann-I Pan, Jong-Yih Kuo, Yong-Yi Fanjiang, and S. Yang. 2000. Towards the verification of scenarios with time Petri-nets. *IEEE Comput. Soc*, 503–508. DOI:https://doi.org/10.1109/CMPSAC.2000.884773

[P122] Wladimir Schamai, Peter Fritzson, Chris Paredis, and Adrian Pop. 2009. Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations. 612–621. DOI:https://doi.org/10.3384/ecp09430081

[P123] Michael E Shin, Alexander H Levis, and Lee W Wagenhals. 2003. Transformation of UML-based system model to design/CPN model for validating system behavior. In *Proc. of the 6th Int. Conf. on the UML/Workshop on Compositional Verification of the UML Models*, Citeseer.

[P124] Martin Gogolla, Fabian Büttner, and Mark Richters. 2007. USE: A UML-based specification environment for validating UML and OCL. *Science of Computer Programming* 69, 1–3 (2007), 27–34. DOI:https://doi.org/10.1016/j.scico.2007.01.013

[P125] W Damm, H Hungar, B Josko, T Peikenkamp, and I Stierand. 2011. Using contract-based component specifications for virtual integration testing and architecture design. *IEEE*, 1–6. DOI:https://doi.org/10.1109/DATE.2011.5763167

[P126] Matthias Bernaerts, Bentley Oakes, Ken Vanherpen, Bjorn Aelvoet, Hans Vangheluwe, and Joachim Denil. 2019. Validating Industrial Requirements with a Contract-Based Approach. *IEEE*, 18–27. DOI:https://doi.org/10.1109/MODELS-C.2019.00010

Just Accepted

[P127] Georg Kösters, Hans-Werner Six, and Mario Winter. 2001. Validation and verification of use cases and class models. In *7th International Workshop on Requirements Engineering: Foundations for Software Quality* (REFSQ'2001, Proc.).

[P128] Dan Li, Xiaoshan Li, Jicong Liu, and Zhiming Liu. 2008. Validation of requirement models by automatic prototyping. *Innovations Syst Softw Eng* 4, 3 (2008), 241–248. DOI:https://doi.org/10.1007/s11334-008-0062-3

[P129] Hongyu Li, Miao Wang, Gang Xiao, and Guoqing Wang. 2022. Verification and test case development method based on civil aircraft operation scenario. *AS* 5, 1 (2022), 65–74. DOI:https://doi.org/10.1007/s42401-021-00090-1

[P130] Myron Hecht and Jaron Chen. 2021. Verification and Validation of SysML Models. *INCOSE International Symposium* 31, 1 (2021), 599–613. DOI:https://doi.org/10.1002/j.2334-5837.2021.00857.x

[P131] R. Kawahara, D. Dotan, T. Sakairi, Kohichi Ono, Hiroaki Nakamura, A. Kirshin, Shinichi Hirose, and Hiroshi Ishikawa. 2009. Verification of embedded system's specification using collaborative simulation of SysML and simulink models. 21–28. DOI:https://doi.org/10.1109/MBSE.2009.5031716

[P132] Jinqiang Zhao and Zhenhua Duan. 2009. Verification of Use Case with Petri Nets in Requirement Analysis. In *Computational Science and Its Applications – ICCSA 2009*. Springer Berlin Heidelberg, Berlin, Heidelberg, 29–42. Retrieved from http://link.springer.com/10.1007/978-3-642-02457-3_3

[P133] Lucas Lima and Amaury Tavares. 2019. Verifying Deadlock and Nondeterminism in Activity Diagrams. *IEEE*, 764–768. DOI:https://doi.org/10.1109/MODELS-C.2019.00119

[P134] Edward Huang, Leon F. McGinnis, and Steven W. Mitchell. 2020. Verifying SysML activity diagrams using formal transformation to Petri nets. *Syst Eng* 23, 1 (2020), 118–135. DOI:https://doi.org/10.1002/sys.21524

[P135] Wladimir Schamai, Philipp Helle, Peter Fritzson, and Christiaan J. J. Paredis. 2011. Virtual Verification of System Designs against System Requirements. In *Models in Software Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 75–89. Retrieved from http://link.springer.com/10.1007/978-3-642-21210-9_8

[P136] Eduard Paul Enoiu, Raluca Marinescu, Cristina Seceleanu, and Paul Pettersson. 2012. ViTAL: A Verification Tool for EAST-ADL Models Using UPPAAL PORT. *IEEE*, 328–337. DOI:https://doi.org/10.1109/ICECCS20050.2012.6299228

[P137] Garazi Juez, Estíbaliz Amparan, Ray Lattarulo, Alejandra Ruíz, Joshué Pérez, and Huáscar Espinoza. 2017. Early safety assessment of automotive systems using sabotage simulation-based fault injection framework. In *International Conference on Computer Safety, Reliability, and Security*, Springer, 255–269.

[P138] Cong Liu, Junaid Babar, Isaac Amundson, Karl Hoech, Darren Cofer, and Eric Mercer. 2022. Assume-Guarantee Reasoning with Scheduled Components. In *NASA Formal Methods Symposium*, Springer, 355–372.

[P139] Rahul Krishnan and Shamsnaz Virani Bhada. 2022. Integrated System Design and Safety Framework for Model-Based Safety Assessment. *IEEE Access* 10, (2022), 79311–79334.

[P140] Orion Staskal, Josh Simac, Logan Swayne, and Kristin Y Rozier. 2022. Translating SysML Activity Diagrams for nuXmv Verification of an Autonomous Pancreas. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference* (COMPSAC), IEEE, 1637–1642.

[P141] Lin Zhang, Fei Ye, Kunyu Xie, Pengfei Gu, Xiaohan Wang, Yuanjun Laili, Chun Zhao, Xuesong Zhang, Minjie Chen, Tingyu Lin, and others. 2022. An Integrated Intelligent Modeling and Simulation Language for Model-based Systems Engineering. *Journal of Industrial Information Integration* 28, (2022), 100347.

[P142] Davide Basile, Maurice H ter Beek, Alessio Ferrari, and Axel Legay. 2022. Exploring the ERTMS/ETCS full moving block specification: an experience with formal methods. *International Journal on Software Tools for Technology Transfer* 24, 3 (2022), 351–370.

[P143] David King, David Jacques, Jeremy Gray, and Katherine Cheney. 2020. Design and simulation of a wide area search mission: an implementation of an autonomous systems reference architecture. In *2020 Winter Simulation Conference (WSC)*, IEEE, 540–551.

[P144] Xinwen Hu, Yi Zhuang, and Fuyuan Zhang. 2020. A security modeling and verification method of embedded software based on Z and MARTE. *Computers & Security* 88, (2020), 101615.

[P145] Micha Sende, Melanie Schranz, Gianluca Prato, Etienne Brosse, Omar Morando, and Martina Umlauft. 2021. Engineering Swarms of Cyber-Physical Systems with the CPSwarm Workbench. *Journal of Intelligent & Robotic Systems* 102, 4 (2021), 1–18.

[P146] Victor Romero, Romain Pinquié, and Frédéric Noël. 2022. A user-centric computer-aided verification process in a virtuality-reality continuum. *Computers in Industry* 140, (2022), 103678.

[P147] Fabian Giertzsch, Oliver C Eichmann, Hartmut Hintze, and Ralf God. 2022. An approach for a simulation-based analysis of business processes using the systems modeling language (SysML). In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 331–340.

[P148] Iulia Dragomir, Carlos Redondo, Tiago Jorge, Laura Gouveia, Iulian Ober, Ivan Kolesnikov, Marius Bozga, and Maxime Perrotin. 2022. Model-checking of space systems designed with TASTE/SDL. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 237–246.

[P149] Pedro Ângelo Vaz De Carvalho, André Ivo, Guilherme Venticinque, Gustavo Vicari Duarte, Matheus Miranda, and Fatima Mattiello-Francisco. 2022. Simplifying Operational Scenario Simulation for CubeSat Mission Analysis Purposes. In *Proceedings of the 11th Latin-American Symposium on Dependable Computing*, 125–130.

Just Accepted

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers whom, through their insightful comments, helped us in presenting the results of our research in an effective way. Moreover, this work was partly funded by the AIDOaRt project, an ECSEL Joint Undertaking (JU) under grant agreement No. 101007350, and by the SACSys Synergy project, an initiative of the Swedish Knowledge Foundation (KKS).

REFERENCES

[1] James L Adams. 2019. *Conceptual blockbusting: A guide to better ideas*. Basic Books.
[2] Tanwir Ahmad, Junaid Iqbal, Adnan Ashraf, Dragos Truscan, and Ivan Porres. 2019. Model-based testing using UML activity diagrams: A systematic mapping study. *Computer Science Review* 33 (2019), 98–112.
[3] Lissette Almonte, Esther Guerra, Iván Cantador, and Juan De Lara. 2022. Recommender systems in model-driven engineering. *Software and Systems Modeling* 21, 1 (2022), 249–280.
[4] Anne Angermann, Michael Beuschel, Martin Rau, and Ulrich Wohlfarth. 2020. Matlab–simulink–stateflow. In *MATLAB–Simulink–Stateflow*. De Gruyter Oldenbourg.
[5] Hugo Araujo, Mohammad Reza Mousavi, and Mahsa Varshosaz. 2022. Testing, Validation, and Verification of Robotic and Autonomous Systems: A Systematic Review. *ACM Transactions on Software Engineering and Methodology* (2022).
[6] Mah Noor Asmat, Saif Ur Rehman Khan, and Shahid Hussain. 2022. Uncertainty handling in cyber–physical systems: State-of-the-art approaches, tools, causes, and future directions. *Journal of Software: Evolution and Process* (2022), e2428.
[7] Johan Bergelin, Antonio Cicchetti, and Emil Lundin. 2022. Early validation of heterogeneous battery systems in the railway domain. In *2022 IEEE International Systems Conference (SysCon)*. IEEE, 1–8.
[8] Damir Bilic, Etienne Brosse, Andrey Sadovykh, Dragos Truscan, Hugo Bruneliere, and Uwe Ryssel. 2019. An integrated model-based tool chain for managing variability in complex system design. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 288–293.
[9] Benjamin S Blanchard, Wolter J Fabrycky, and Walter J Fabrycky. 1990. *Systems engineering and analysis*. Vol. 4. Prentice hall Englewood Cliffs, NJ.
[10] George EP Box and Norman R Draper. 1987. *Empirical model-building and response surfaces*. John Wiley & Sons.
[11] Antonio Bucchiarone, Jordi Cabot, Richard F Paige, and Alfonso Pierantonio. 2020. Grand challenges in model-driven engineering: an analysis of the state of the research. *Software and Systems Modeling* 19, 1 (2020), 5–13.
[12] Antonio Bucchiarone, Federico Ciccozzi, Leen Lambers, Alfonso Pierantonio, Matthias Tichy, Massimo Tisi, Andreas Wortmann, and Vadim Zaytsev. 2021. What is the future of modeling? *IEEE software* 38, 2 (2021), 119–127.
[13] Loli Burgueño, Alexandru Burdusel, Sébastien Gérard, and Manuel Wimmer. 2019. Preface to MDE intelligence 2019: 1st workshop on artificial intelligence and model-driven engineering. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 168–169.
[14] Jordi Cabot, Robert Clarisó, Marco Brambilla, and Sébastien Gérard. 2017. Cognifying model-driven software engineering. In *Federation of international conferences on software technologies: applications and foundations*. Springer, 154–160.
[15] Javier Cámara, Radu Calinescu, Betty HC Cheng, David Garlan, Bradley Schmerl, Javier Troya, and Antonio Vallecillo. 2022. Addressing the uncertainty interaction problem in software-intensive systems: challenges and desiderata. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*. 24–30.
[16] Kelly X Campo, Thomas Teper, Casey E Eaton, Anna M Shipman, Garima Bhatia, and Bryan Mesmer. 2022. Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature. *Systems Engineering* (2022).
[17] Edward Ralph Carroll and Robert Joseph Malins. 2016. Systematic Literature Review: How is Model-Based Systems Engineering Justified?. (2016).
[18] Johan Cederbladh and Jagadish Suryadevara. 2023. Towards a Unified Architecture Methodology for Product Service Systems. In *Asia Oceanic Systems Engineering Conference*. http://www.es.mdu.se/publications/6806-
[19] Mohammad Chami, Aiste Aleksandraviciene, Aurelijus Morkevicius, and Jean-Michel Bruel. 2018. Towards solving MBSE adoption challenges: the D3 MBSE adoption toolbox. In *INCOSE International Symposium*, Vol. 28. Wiley Online Library, 1463–1477.
[20] Mohammad Chami and Jean-Michel Bruel. 2018. A survey on MBSE adoption challenges. (2018).
[21] Jean-Charles Chaudemar and Pierre de Saqui-Sannes. 2021. Mbse and mdao for early validation of design decisions: a bibliography survey. In *2021 IEEE International Systems Conference (SysCon)*. IEEE, 1–8.
[22] Clayton M Christensen. 2013. *The innovator's dilemma: when new technologies cause great firms to fail*. Harvard Business Review Press.
[23] Benoit Combemale and Manuel Wimmer. 2020. Towards a model-based DevOps for cyber-physical systems. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 84–94.

Just Accepted

[24] Joseph D'Ambrosio and Grant Soremekun. 2017. Systems engineering challenges and MBSE opportunities for automotive system design. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2075–2080.
[25] Pierre De Saqui-Sannes, Rob A Vingerhoeds, Christophe Garion, and Xavier Thirioux. 2022. A taxonomy of MBSE approaches by languages, tools and methods. *IEEE Access* (2022).
[26] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. DevOps. *Ieee Software* 33, 3 (2016), 94–100.
[27] Jeff A Estefan et al. 2007. Survey of model-based systems engineering (MBSE) methodologies. *Incose MBSE Focus Group* 25, 8 (2007), 1–12.
[28] Philipp M Fischer, Daniel Lüdtke, Caroline Lange, F-C Roshani, Frank Dannemann, and Andreas Gerndt. 2017. Implementing model-based system engineering for the whole lifecycle of a spacecraft. *CEAS Space journal* 9, 3 (2017), 351–365.
[29] Amit Fisher, Mike Nolan, Sanford Friedenthal, Michael Loeffler, Mark Sampson, Manas Bajaj, Lonnie VanZandt, Krista Hovey, John Palmer, and Laura Hart. 2014. 3.1. 1 model lifecycle management for MBSE. In *INCOSE International Symposium*, Vol. 24. Wiley Online Library, 207–229.
[30] Frederick K Frantz. 1995. A taxonomy of model abstraction techniques. In *Proceedings of the 27th conference on Winter simulation*. 1413–1420.
[31] Sanford Friedenthal, Regina Griego, and Mark Sampson. 2007. INCOSE model based systems engineering (MBSE) initiative. In *INCOSE 2007 symposium*, Vol. 11. sn.
[32] Simon Fürst, Jürgen Mössinger, Stefan Bunzel, Thomas Weber, Frank Kirschke-Biller, Peter Heitkämper, Gerulf Kinkelin, Kenji Nishikawa, and Klaus Lange. 2009. AUTOSAR–A Worldwide Standard is on the Road. In *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*, Vol. 62. 5.
[33] Vahid Garousi, Kai Petersen, and Baris Ozkan. 2016. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. *Information and Software Technology* 79 (2016), 106–127.
[34] Sébastien Gérard, Cédric Dumoulin, Patrick Tessier, and Bran Selic. 2007. 19 Papyrus: A UML2 tool for domain-specific language modeling. In *Dagstuhl workshop on model-based engineering of embedded real-time systems*. Springer, 361–368.
[35] Iris Graessler and Julian Hentze. 2020. The new V-Model of VDI 2206 and its validation. *at-Automatisierungstechnik* 68, 5 (2020), 312–324.
[36] Joe Gregory, Lucy Berthoud, Theo Tryfonas, Alain Rossignol, and Ludovic Faure. 2020. The long and winding road: MBSE adoption for functional avionics of spacecraft. *Journal of Systems and Software* 160 (2020), 110453.
[37] Robert Hällqvist, Raghu Chaitanya Munjulury, Robert Braun, Magnus Eek, and Petter Krus. 2022. Realizing Interoperability between MBSE Domains in Aircraft System Development. *Electronics* 11, 18 (2022), 2901.
[38] Laura E Hart. 2015. Introduction to model-based system engineering (MBSE) and SysML. In *Delaware Valley INCOSE Chapter Meeting*, Vol. 30. Ramblewood Country Club Mount Laurel, New Jersey.
[39] Lars Ivar Hatledal, Arne Styve, Geir Hovland, and Houxiang Zhang. 2019. A language and platform independent co-simulation framework based on the functional mock-up interface. *IEEE Access* 7 (2019), 109328–109339.
[40] Kaitlin Henderson, Tom McDermott, Eileen Van Aken, and Alejandro Salado. 2023. Towards Developing Metrics to Evaluate Digital Engineering. *Systems Engineering* 26, 1 (2023), 3–31.
[41] Kaitlin Henderson and Alejandro Salado. 2021. Value and benefits of model-based systems engineering (MBSE): Evidence from the literature. *Systems Engineering* 24, 1 (2021), 51–66.
[42] Jon B Holladay, Jessica Knizhnik, Karen J Weiland, Amanda Stein, Terry Sanders, and Paul Schwindt. 2019. MBSE Infusion and Modernization Initiative (MIAMI):"Hot" benefits for real NASA applications. In *2019 IEEE Aerospace Conference*. IEEE, 1–14.
[43] Jiexiang Hu, Yang Yang, Qi Zhou, Ping Jiang, Xinyu Shao, Leshi Shu, and Yahui Zhang. 2018. Comparative studies of error metrics in variable fidelity model uncertainty quantification. *Journal of Engineering Design* 29, 8-9 (2018), 512–538.
[44] Jerome Hugues, Anton Hristosov, John J Hudak, and Joe Yankel. 2020. Twinops-devops meets model-based engineering and digital twins for the engineering of cps. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. 1–5.
[45] Tomas Huldt and Ivan Stenius. 2019. State-of-practice survey of model-based systems engineering. *Systems engineering* 22, 2 (2019), 134–145.
[46] Haider Naqvi Imran and Aziz Shazia. 2011. The impact of stakeholder communication on project outcome. *African Journal of Business Management* 5, 14 (2011), 5824–5832.
[47] Maikel Issermann, Fi-John Chang, and Pu-Yun Kow. 2021. Interactive urban building energy modelling with functional mockup interface of a local residential building stock. *Journal of Cleaner Production* 289 (2021), 125683.
[48] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. 2020. Characterising the Digital Twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology* 29 (2020), 36–52.
[49] Robbert Jongeling, Federico Ciccozzi, Jan Carlson, and Antonio Cicchetti. 2022. Consistency management in industrial continuous model-based development settings: a reality check. *Software and Systems Modeling* (2022), 1–20.
[50] Nafiseh Kahani, Mojtaba Bagherzadeh, Juergen Dingel, and James R Cordy. 2016. The problems with eclipse modeling tools: a topic analysis of eclipse forums. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and*

Just Accepted

Systems. 227–237.
[51] Barbara Kitchenham, Rialette Pretorius, David Budgen, O Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen Linkman. 2010. Systematic literature reviews in software engineering–a tertiary study. *Information and software technology* 52, 8 (2010), 792–805.
[52] Kathy Kotiadis and Stewart Robinson. 2008. Conceptual modelling: knowledge acquisition and model abstraction. In *2008 Winter Simulation Conference*. IEEE, 951–958.
[53] Christopher Laing, Pierre David, Eric Blanco, and Xavier Dorel. 2020. Questioning integration of verification in model-based systems engineering: an industrial perspective. *Computers in Industry* 114 (2020), 103163.
[54] Brian R Larson, Patrice Chalin, and John Hatcliff. 2013. BLESS: Formal specification and verification of behaviors for embedded systems with software. In *NASA Formal Methods Symposium*. Springer, 276–290.
[55] Thierrry Le Sergent, François-Xavier Dormoy, and Alain Le Guennec. 2016. Benefits of model based system engineering for avionics systems. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*.
[56] Edward A Lee and Marjan Sirjani. 2018. What good are models?. In *International Conference on Formal Aspects of Component Software*. Springer, 3–31.
[57] Yiping Li, Jianwen Chen, and Ling Feng. 2012. Dealing with uncertainty: A survey of theories and practices. *IEEE Transactions on Knowledge and Data Engineering* 25, 11 (2012), 2463–2482.
[58] Zihang Li, Jinzhi Lu, Guoxin Wang, Lei Feng, Didem Gurdur Broo, and Dimitris Kiritsis. 2021. A Bibliometric Analysis on Model-based Systems Engineering. In *2021 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 1–8.
[59] Margaret L Loper. 2015. *Modeling and simulation in the systems engineering life cycle: core concepts and accompanying lectures*. Springer.
[60] Junda Ma, Guoxin Wang, Jinzhi Lu, Hans Vangheluwe, Dimitris Kiritsis, and Yan Yan. 2022. Systematic Literature Review of MBSE Tool-Chains. *Applied Sciences* 12, 7 (2022), 3431.
[61] Azad M Madni and Michael Sievers. 2018. Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering* 21, 3 (2018), 172–190.
[62] John TJ Mathieson, Thomas Mazzuchi, and Shahram Sarkani. 2020. The systems engineering DevOps lemniscate and model-based system operations. *IEEE Systems Journal* 15, 3 (2020), 3980–3991.
[63] Yaroslav Menshenin, Carolina Moreno, Yana Brovar, and Clement Fortin. 2021. Integration of MBSE and PLM: complexity and uncertainty. *International Journal of Product Lifecycle Management* 13, 1 (2021), 66–88.
[64] William D MILLER. 2022. The Future of Systems Engineering: Realizing the Systems Engineering Vision 2035. (2022).
[65] Aurelijus Morkevicius, Aiste Aleksandraviciene, and Zilvinas Strolia. 2022. System Verification and Validation Approach Using the MagicGrid Framework. In *INCOSE International Symposium*, Vol. 32. Wiley Online Library, 767–781.
[66] Christian Nigischer, Sébastien Bougain, Rainer Riegler, Heinz Peter Stanek, and Manfred Grafinger. 2021. Multi-domain simulation utilizing SysML: state of the art and future perspectives. *Procedia CIRP* 100 (2021), 319–324.
[67] Mara Nikolaidou, George-Dimitrios Kapos, Anargyros Tsadimas, Vassilis Dalakas, and Dimosthenis Anagnostopoulos. 2016. Challenges in SysML model simulation. *Advances in Computer Science: an International Journal* 5, 4 (2016), 49–56.
[68] Knut Ovsthus, Lars M Kristensen, et al. 2014. An industrial perspective on wireless sensor networks—A survey of requirements, protocols, and challenges. *IEEE communications surveys & tutorials* 16, 3 (2014), 1391–1412.
[69] Gregory S Parnell, C Robert Kenley, Clifford A Whitcomb, and Karthikeyan Palanikumar. 2021. System design and engineering trade-off analytics: State of the published practice. *Systems Engineering* 24, 3 (2021), 125–143.
[70] Edith Parrott. 2016. The value of successful MBSE adoption. In *No Magic World Symposium 2016*.
[71] Sebastian Pilarski, Martin Staniszewski, Frederic Villeneuve, and Daniel Varro. 2019. On artificial intelligence for simulation and design space exploration in gas turbine design. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 170–174.
[72] Moritz Quandt, Benjamin Knoke, Christian Gorldt, Michael Freitag, and Klaus-Dieter Thoben. 2018. General requirements for industrial augmented reality applications. *Procedia Cirp* 72 (2018), 1130–1135.
[73] Muhammad Rashid, Muhammad Waseem Anwar, Farooque Azam, and Muhammad Kashif. 2016. Model-based requirements and properties specifications trends for early design verification of embedded systems. In *2016 11th System of Systems Engineering Conference (SoSE)*. IEEE, 1–7.
[74] Muhammad Rashid, Muhammad Waseem Anwar, and Aamir M Khan. 2015. Toward the tools selection in model based system engineering for embedded systems—A systematic literature review. *Journal of Systems and Software* 106 (2015), 150–163.
[75] Reema Sandhu. 2015. Model-Based Software Engineering (MBSE) and Its Various Approaches and Challenges. *Compusoft* 4, 6 (2015), 1841.
[76] Bill Schindel and Rick Dove. 2016. Introduction to the agile systems engineering life cycle MBSE pattern. In *INCOSE International Symposium*, Vol. 26. Wiley Online Library, 725–742.
[77] Douglas C Schmidt. 2006. Model-driven engineering. *Computer-IEEE Computer Society*- 39, 2 (2006), 25.
[78] Bran Selic. 2003. The Pragmatics of Model-Driven Development. *IEEE Software* 20, 5 (2003), 19–25.

Just Accepted

[79] Mary Shaw. 2002. What makes good research in software engineering? *International Journal on Software Tools for Technology Transfer* 4, 1 (2002), 1–7.
[80] Victor Singh and Karen E Willcox. 2018. Engineering design with digital thread. *AIAA Journal* 56, 11 (2018), 4515–4528.
[81] Nathan J Slegers, Ronald T Kadish, Gary E Payton, John Thomas, Michael D Griffin, and Dan Dumbacher. 2012. Learning from failure in systems engineering: A panel discussion. *Systems Engineering* 15, 1 (2012), 74–82.
[82] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. 2008. *EMF: eclipse modeling framework*. Pearson Education.
[83] Jagadish Suryadevara and Saurabh Tiwari. 2018. Adopting MBSE in construction equipment industry: An experience report. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 512–521.
[84] Koen Tange, Michele De Donno, Xenofon Fafoutis, and Nicola Dragoni. 2020. A systematic survey of industrial Internet of Things security: Requirements and fog computing opportunities. *IEEE Communications Surveys & Tutorials* 22, 4 (2020), 2489–2520.
[85] Tino Teige, Andreas Eggers, Karsten Scheibler, Matthias Stasch, Udo Brockmeyer, Hans J Holberg, and Tom Bienmüller. 2021. Two Decades of Formal Methods in Industrial Products at BTC Embedded Systems. In *International Symposium on Formal Methods*. Springer, 725–729.
[86] Naoum Tsioptsias, Antuela Tako, and Stewart Robinson. 2016. Model validation and testing in simulation: a literature review. In *5th Student Conference on Operational Research (SCOR 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[87] Andreas Vogelsang, Tiago Amorim, Florian Pudlitz, Peter Gersing, and Jan Philipps. 2017. Should I stay or should I go? On forces that drive and prevent MBSE adoption in the embedded systems industry. In *International Conference on Product-Focused Software Process Improvement*. Springer, 182–198.
[88] David D Walden, Garry J Roedler, and Kevin Forsberg. 2015. INCOSE systems engineering handbook version 4: updating the reference for practitioners. In *INCOSE International Symposium*, Vol. 25. Wiley Online Library, 678–686.
[89] Fabian Wilking, Benjamin Schleich, and Sandro Wartzack. 2020. MBSE along the Value Chain–An Approach for the Compensation of additional Effort. In *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*. IEEE, 61–66.
[90] James R Williams, Frank R Burton, Richard F Paige, and Fiona AC Polack. 2012. Sensitivity analysis in model-driven engineering. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 743–758.
[91] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. 1–10.
[92] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.
[93] Ying Yan, Bin Suo, and Ziwei Li. 2022. An area-based metrics to evaluate risk in failure mode and effects analysis under uncertainties. *IEEE Access* 10 (2022), 33969–33979.
[94] Bernard P Zeigler, Saurabh Mittal, and Mamadou Kaba Traore. 2018. MBSE with/out Simulation: State of the Art and Way Forward. *Systems* 6, 4 (2018), 40.

Just Accepted

A EXTRACTED TABLES

In this appendix the corresponding tables are presented for the graphs in Section 6, each table contain the cross-referencing of the publications. We note that in many cases several categories are true for a publication at the same time, for example several motivational reasons might apply for performing early V&V or several target properties are of interest from analysis. The tables are ordered in the same order as figures from Section 6.

A.1 Motivating reasons

Table 2 detail the author motivations for performing early V&V.

Table 2. Motivation for early V&V.

References	Motivating reason for early V&V
[P1, P120]	Improve quality of developed system before implementation
[P2, P6, P12, P34, P35, P43, P57, P58, P59, P60, P67, P78, P121, P123, P128, P149]	Decrease risk for incomplete, inconsistent or missing requirements earlier
[P3, P4, P6, P9, P11, P13, P14, P18, P19, P24, P30, P32, P33, P39, P44, P45, P48, P54, P55, P59, P61, P63, P68, P70, P78, P80, P84, P85, P86, P88, P89, P91, P95, P98, P99, P102, P104, P105, P106, P108, P112, P113, P114, P119, P120, P121, P124, P130, P131, P134, P135, P137, P139, P140, P141, P142, P148, P149]	V&V of design before implementation towards requirements
[P5, P39, P147]	Identify potential areas for design optimization
[P5, P7, P17, P18, P19, P20, P42, P51, P58, P60, P66, P70, P75, P83, P88, P92, P95, P103, P106, P109, P118, P119, P133, P134, P145]	Reduce risks of costs associated with late flaw detection
[P8, P10, P23, P31, P62, P65, P74, P82, P117, P143, P148]	Explore and test system behaviour before implementation
[P10, P52, P100]	System development decision making
[P15]	Raising quality of models
[P16, P21, P22, P29, P40, P56, P71, P90, P110, P115, P116, P127, P138]	Not formulated clearly
[P23, P38, P45, P46, P70, P79, P81, P83, P88, P89, P93, P111, P118, P119, P126, P129, P132, P146]	Reduce time to market/increase efficiency
[P25, P133]	To promote dependability
[P96, P105, P137, P142]	Dependability analysis
[P26, P146]	Validate intended behaviour as elaborate performance analysis is infeasible

Continued on next page

Just Accepted

Table 2 – continued from previous page

References	Motivating reason for early V&V
[P27, P73]	Understand component interactions
[P28, P91, P92]	Requirements traceability
[P28]	System architecture compliance with requirements
[P32]	To enable step-wise refinement
[P33]	Understanding the emergent behaviour
[P33, P69, P72, P104, P111, P123, P147, P149]	Performance understanding
[P35, P57]	Reduce uncertainty in requirements
[P36, P100]	Identification of hidden errors in specification
[P37]	Assist in hardware/software partitioning
[P41, P68, P143]	Increase level of abstraction
[P47]	Concept evaluation
[P49, P103, P109]	Improve re-use
[P49]	Improve scaleability
[P50]	Improve requirements management
[P53]	Product certification
[P53]	Product assurance
[P54]	Discovering unwanted behaviour
[P64, P66, P72, P136]	Analysis of architecture
[P67]	Improve requirements quality
[P73, P74, P76, P91, P94, P100, P102, P104]	Trade studies/impact of design
[P77, P80, P94, P97, P122, P126]	Improve common understanding between different domains/roles
[P87]	Increase understanding of system
[P92]	Verification of requirements before design
[P101]	Provide practical feedback for users earlier
[P107, P139]	Improving safety analysis coverage
[P125, P130, P136]	Reduce integration issues
[P137, P139]	Safety assessment
[P144]	Verification of security

A.2 Description language

Table 3 details the description languages in the extracted solutions.

Table 3. Description language for solutions.

References	Description language
[P1, P30, P60]	Informal/Natural language
[P1, P28, P48, P60, P80, P126, P99, P106, P3, P141]	Custom language

Continued on next page

Just Accepted

Table 3 – continued from previous page

References	Description language
[P2, P5, P6, P7, P8, P10, P11, P12, P13, P14, P20, P22, P25, P27, P30, P31, P33, P36, P41, P43, P44, P45, P46, P47, P49, P50, P51, P53, P54, P55, P57, P58, P59, P61, P62, P63, P65, P70, P71, P72, P73, P74, P77, P78, P79, P81, P82, P83, P85, P86, P87, P89, P91, P92, P93, P95, P98, P100, P101, P102, P104, P107, P108, P109, P110, P111, P112, P113, P114, P116, P117, P118, P119, P120, P122, P129, P130, P131, P134, P139, P140, P143, P145, P147]	SysML
[P4, P32, P103, P136]	EAST-ADL
[P5, P23, P24, P68, P81, P83, P115, P131, P144]	MARTE
[P6, P15, P20, P29, P40, P56, P86, P109, P124, P128, P11, P86]	OCL
[P9, P13, P14, P15, P23, P29, P34, P35, P41, P42, P52, P56, P57, P66, P68, P76, P86, P99, P115, P123, P124, P127, P128, P132, P133, P121]	UML
[P16, P17, P18, P67, P69, P105, P138]	AADL
[P19, P46, P84, P97, P108, P126, P137, P142, P146]	Simulink/MATLAB
[P20]	ALF
[P39]	Amola
[P21]	AltaRica 3.0
[P26]	Supremica
[P27, P65]	DoDAF
[P37]	Cola
[P38]	Language that supports petri net
[P39]	IMA architecture
[P52]	AUTOSAR
[P64]	fUML
[P71]	AcTRL
[P75]	CONSENS
[P76, P90, P96, P122, P135]	ModelicaML
[P78]	KARMA
[P78]	BPMN
[P78]	GOPPRR-E
[P88]	Chi
[P89]	HLA
[P94, P125]	Language agnostic
[P95, P104]	OWL
[P95]	SWRL
[P136]	FAA
[P138]	AGREE
[P144]	Z
[P146]	CAD
[P148]	SDL
[P149]	Lua

Just Accepted

A.3 Analysis language

Table 4 highlights the language used for model analysis.

Table 4. Analysis language in solutions.

References	Language for analysis
[P1, P4, P28, P59, P80, P141, P143]	Custom language
[P2, P8]	fUML
[P3, P82, P139]	Python
[P5, P6, P9, P38, P65, P71, P115, P120, P121, P123, P132, P134]	Petri net diagram
[P7, P19, P33, P39, P45, P46, P61, P62, P73, P84, P91, P102, P108, P109, P112, P113, P117, P126, P131, P79]	MATLAB/Simulink
[P8, P22, P27, P30, P31, P33, P46, P47, P50, P51, P55, P61, P63, P70, P75, P79, P107, P114, P117, P129, P130, P58, P101, P104, P147]	SysML
[P10]	DKF-based code
[P11, P14, P86, P41]	SystemVerilog RTL code
[P12, P53]	Not demonstrated/specified
[P13, P105, P67, P36, P140]	NuSMV
[P15]	Alloy
[P16, P47, P68, P74, P76, P81, P85, P90, P93, P96, P97, P102, P119, P122, P135]	Modelica
[P16, P17, P18]	AADL based
[P16]	Cheddar
[P17]	AGREE
[P17, P138]	Lustre
[P20]	B language
[P21]	AltaRica 3.0
[P23]	e hardware verification language
[P24]	Event-B
[P25]	COMPASS
[P27, P44, P69, P83, P85, P88, P116, P136, P118, P26, P142]	UPPAAL
[P32, P37, P103]	SystemC
[P34]	Promela
[P35, P40, P109, P124, P128]	OCL
[P42, P75, P40]	UML
[P43, P66]	Not clear
[P52]	Ptolemy II
[P56, P98]	Maude
[P57]	RELAXIFx
[P104]	OWL
[P60]	BIP
[P62]	C++
[P70]	HatleyPirbhai Control Flow Diagram

Continued on next page

Just Accepted

Table 4 – continued from previous page

References	Language for analysis
[P77, P94, P95, P125]	Language/Implementation agnostic
[P89, P100]	Java
[P89]	HLA
[P92]	R programming language
[P97]	LabVIEW
[P98]	Recursive ECATNets
[P106]	Prolog
[P137]	FARM
[P137]	Sabotage
[P138]	Scade
[P138]	SIGNAL
[P143]	Ardupilot Software in the loop
[P144]	ZMsec
[P145]	ROS
[P146]	C#
[P148]	IF language
[P149]	Lua

Table 5 details the formalisms employed for system analysis.

Table 5. Formalism used for analysis.

References	Formalism for analysis
[P28]	Modes and transitions
[P4, P14, P44, P83, P118, P88, P142, P144]	Timed Automata
[P5, P6, P9, P38, P65, P71, P115, P120, P121, P123, P132, P134]	Petri net variants
[P6, P14, P44, P60, P105, P67, P105, P144]	CTL
[P12, P53]	Not demonstrated/specified
[P13, P29]	Transition system
[P15, P132]	CSP
[P26, P67, P105]	Markov chains
[P35]	Contract based
[P36, P138, P144, P81]	Temporal logic formulas
[P41, P86]	UVM
[P48, P110]	Dependency graphs
[P48]	Constraint graphs
[P49, P72, P146]	CAD
[P1, P54, P58, P66, P67, P101, P104, P148, P149]	State based

Continued on next page

Just Accepted

Table 5 – continued from previous page

References	Formalism for analysis
[P59, P129]	Scenario based
[P59]	Differential equations
[P60]	Structured requirements
[P64, P99]	Process algebra
[P64]	Backus-Naur form
[P67, P105, P140]	LtL
[P68]	TRIO
[P69, P115]	TCTL
[P75]	MSD specifications
[P75]	Modal sequence diagrams
[P77, P94, P95, P125]	Language/Implementation agnostic
[P78]	Hybrid automata
[P80]	Functional state sequence diagrams
[P87]	Fault trees
[P87]	FMEA artefacts
[P95]	Ontology models
[P95]	Rule based
[P98]	Recursive ECATNets
[P99]	UTP
[P100]	DEVS
[P105]	Stochastic logic
[P111]	Directed graphs
[P111]	STA-GT algorithm
[P127]	Activity graphs
[P146]	VR
[P147]	Cumulative Distribution Functions

A.4 Method for analysis

Table 6 describes the reported overarching technique of the V&V.

Table 6. Method for V&V

References	Method for V&V
[P1, P12, P43, P53, P54, P55, P75, P80, P101, P109, P114, P124, P127, P130]	Review/Inspection of artefacts
[P106, P130]	Query engine analysis

Continued on next page

Just Accepted

Table 6 – continued from previous page

References	Method for V&V
[P2, P3, P4, P7, P8, P10, P11, P12, P14, P15, P18, P19, P21, P23, P27, P28, P30, P31, P32, P33, P36, P37, P38, P39, P41, P45, P46, P47, P49, P51, P52, P57, P58, P60, P61, P62, P63, P65, P67, P68, P70, P72, P73, P74, P75, P76, P77, P78, P79, P80, P81, P82, P83, P84, P85, P86, P88, P89, P90, P91, P93, P95, P96, P97, P99, P100, P102, P103, P104, P107, P108, P111, P112, P113, P117, P118, P119, P122, P123, P126, P131, P135, P136, P5, P9, P94, P101, P134, P25, P44, P59, P137, P139, P141, P143, P145, P146, P147, P149]	Simulation
[P4, P6, P13, P17, P18, P19, P25, P26, P34, P36, P44, P48, P60, P64, P66, P67, P68, P69, P71, P83, P85, P88, P98, P99, P105, P110, P115, P118, P134, P136, P5, P138, P140, P142, P144, P148]	Model checking
[P6]	Incremental verification
[P13]	Program analysis
[P16]	Schedulability analysis
[P16]	Syntax analysis
[P16]	Lexical analysis
[P17]	K-induction
[P17, P39, P84, P137, P139]	Fault-injection
[P17, P53]	Probabilistic analysis/reliability analysis
[P20, P99]	Theorem proving
[P22]	Requirements completeness analysis
[P24]	Proof obligations with event-B
[P25, P32, P65, P99, P133, P139, P145]	Refinement
[P29, P133]	Counter example search
[P30, P51, P63]	ESEM modeling
[P32, P42, P65, P67, P2]	Consistency check/analysis
[P35, P56]	OCL expression evaluation
[P35, P128]	Prototype execution
[P37, P39]	Timing and temporal analysis
[P40]	Execution path exploration
[P44, P121, P132]	Reachability analysis
[P45, P143]	Trade-off analysis
[P50]	Variability management
[P53, P87, P96]	Fault tree analysis
[P60]	Correctness by construction
[P66, P69, P85, P134]	State space exploration
[P72]	Computer aided engineering
[P76, P97, P146]	Model in the loop
[P82]	Rapid prototyping
[P87, P107]	Failure mode and effect analysis
[P92]	Graph theory

Continued on next page

Just Accepted

Table 6 – continued from previous page

References	Method for V&V
[P111]	Detailed latency analysis technique
[P116]	Model reasoning
[P125]	Contract-based virtual integration testing
[P129]	Value analysis
[P138]	Compositional verification
[P146]	Virtual Reality

A.5 Results of interest

Table 7 details the target properties of analysis reported by the authors.

Table 7. Result of interest / target properties

References	Result of interest
[P1, P32, P42, P82, P86, P109, P127, P15, P16, P46, P138, P140, P142, P145, P147]	Model correctness/validity/completeness
[P1, P9, P14, P24, P58, P69, P85, P98, P99, P110, P115, P116, P121, P132, P134, P136]	Reachability
[P2, P12, P19, P25, P28, P35, P42, P53, P58, P60, P65, P66, P67, P68, P80, P82, P117, P124, P126, P127, P128, P15, P140]	Inconsistencies/Contradictions
[P4, P5, P13, P24, P30, P39, P51, P69, P81, P85, P88, P111, P115, P131, P136]	Time related properties
[P4, P6, P22, P31, P35, P37, P51, P54, P55, P61, P63, P71, P72, P77, P78, P86, P114, P117, P135]	Functional requirements verification
[P6, P13, P25, P26, P34, P58, P60, P65, P67, P69, P83, P88, P99, P105, P115, P133]	Deadlock Freeness
[P6, P13, P14, P25, P26, P60, P65, P66, P69, P83, P85, P88, P98, P110, P116, P123, P132, P134, P136]	Liveness
[P7, P17, P21, P26, P39, P53, P87, P91, P96, P101, P107, P137, P139]	Reliability & failure probability
[P8, P11, P27, P38, P61, P62, P63, P64, P66, P74, P75, P76, P84, P90, P95, P100, P102, P103, P111, P117, P122, P131, P135, P32, P93, P119, P104, P137, P138, P141, P148]	Execution traces
[P8, P92, P50, P101, P105, P114, P125, P145]	Viable system configurations
[P9, P13, P14, P17, P24, P26, P60, P66, P67, P69, P77, P82, P83, P84, P85, P87, P88, P98, P134, P20, P137, P139, P140, P142, P144, P148]	Safety properties
[P10, P2, P12, P23, P37, P43, P45, P46, P52, P60, P61, P70, P80, P84, P89, P97, P136, P146, P149]	Intended functional behaviour
[P12, P28, P82]	Traceability
[P13, P46, P49]	Complexity & re-useability
[P16, P69, P116]	Schedulability
[P16, P19, P24, P59, P73, P78, P79, P82, P97, P100, P102, P108, P112, P113, P131]	Run-time behaviour

Continued on next page

Just Accepted

Table 7 – continued from previous page

References	Result of interest
[P17, P34, P36, P44, P81, P140, P144]	Viable counter-examples
[P3, P18, P29, P36, P40, P20, P56, P83, P118, P125]	Property constraint violations/satisfactions
[P5, P26, P30, P65, P73, P97, P120, P143, P147]	Performance
[P27, P33, P129]	Scenario validation
[P30, P45, P48, P51, P72, P73, P94, P143]	Trade-off analysis
[P41, P47, P55, P62, P130, P141]	Application dependant
[P57]	Viability of requirements with uncertainty
[P72, P76, P137]	Process/strategy validation
[P99, P133]	Determinism
[P107, P110]	Failure paths/effects
[P123]	Fairness
[P123, P132, P134]	Boundness
[P124]	Independence
[P132]	Reversability
[P144]	Security properties
[P146]	V&V in realistic VR environment

A.6 Tool categories

Table 8 details the tools categories of the tools used in solutions reported by authors.

Table 8. Tool categories of analysis tools.

References	Tools used for V&V
[P1, P20, P24, P40, P138]	Theorem provers
[P2, P4, P7, P8, P25, P34, P36, P42, P53, P58, P61, P65, P72, P74, P77, P84, P95, P117, P130, P147, P22, P23, P30, P49, P51, P55, P63, P70, P73, P79, P85, P100, P112, P113, P27, P44, P45, P46, P50, P57, P102, P80, P107, P114]	Integrated MBSE toolkit
[P6, P8, P10, P11, P14, P22, P25, P30, P33, P48, P51, P61, P63, P65, P68, P71, P73, P82, P86, P89, P103, P111, P123, P126, P139, P145, P147, P149]	Simulation toolkit
[P3, P4, P5, P13, P14, P17, P18, P21, P26, P29, P34, P36, P44, P60, P67, P68, P69, P81, P83, P85, P88, P99, P100, P105, P110, P111, P115, P116, P118, P121, P124, P132, P133, P142, P136, P138, P140, P141, P144]	Model-checker
[P7, P19, P32, P33, P39, P44, P45, P46, P52, P59, P62, P72, P73, P77, P79, P84, P91, P97, P100, P102, P108, P112, P113, P117, P129, P131, P137, P16, P68, P74, P76, P81, P85, P90, P93, P94, P96, P119, P122, P135, P141, P145]	Graphical programming/simulation environment
[P9, P10, P14, P16, P17, P18, P19, P23, P24, P28, P40, P41, P53, P56, P59, P75, P76, P78, P79, P81, P86, P89, P96, P99, P103, P112, P113, P118, P119, P120, P131, P134, P136, P148]	Modelling framework

Continued on next page

Just Accepted

Table 8 – continued from previous page

References	Tools used for V&V
[P12, P31, P37, P38, P43, P54, P66, P92, P98, P104]	Tool not presented
[P14, P87, P97, P127]	Integrated Design Environment
[P15]	SAT solver
[P16]	Scheduling simulator
[P16, P17, P18, P69, P123]	Syntax checker
[P87]	Probabilistic analysis engine
[P35, P60, P128]	Code Generator
[P47, P64, P76, P95, P101, P109, P125, P134]	Tool independent/agnostic
[P49, P72, P146]	CAD tool
[P61, P117]	Spreadsheet manipulator
[P95]	Ontology editor
[P97, P143, P146]	Hardware interface tool
[P106]	Scenario Question Query Engine

A.7 Tools used for V&V

Table 9 details the tools used in solutions reported by authors.

Table 9. Tools employed for V&V.

References	Tools used for V&V
[P1, P40]	Z3 solver
[P2, P8, P36, P61, P72, P74, P77, P95, P117, P130, P147]	Cameo systems modeler
[P8, P22, P30, P33, P51, P63, P73, P147]	Cameo simulation toolkit
[P3]	Sismic
[P4, P136]	ViTAL
[P4]	Papyrus UML
[P5]	INA Tool
[P6, P65, P71]	CPNTools
[P7]	Rational Rhapsody
[P7, P19, P33, P39, P44, P45, P46, P62, P73, P77, P79, P84, P91, P97, P102, P108, P112, P113, P117, P129, P131, P137]	MATLAB/Simulink
[P8, P84]	IBM DOORS
[P8]	ENOVIA
[P8]	3DEXPERIENCE
[P9, P10, P14, P24, P41, P56, P75, P86, P89, P103, P118, P119, P134, P136]	Papyrus
[P9, P10, P14, P16, P17, P18, P23, P24, P28, P41, P53, P56, P86, P89, P99, P103, P118, P119, P120, P131, P136]	Eclipse/EMF
[P10, P89]	HLA RTI
[P11, P14, P86]	QuestaSIM
[P12, P31, P37, P38, P43, P54, P66, P92, P98, P104]	Tool not presented
[P13]	ARTiSAN Real-time studio
[P14, P26, P44, P69, P83, P85, P88, P116, P118, P136, P142]	UPPAAL

Continued on next page

Just Accepted

Table 9 – continued from previous page

References	Tools used for V&V
[P14]	Xilinx Vivado
[P15]	MiniSat+
[P16, P68, P74, P76, P81, P85, P90, P93, P94, P96, P119, P122, P135]	OpenModelica/Generic Modelica tool
[P16]	Cheddar
[P16, P17, P18, P69]	OSATE
[P17]	JKind
[P18]	ABV
[P19]	Avionics systems test bench
[P19]	RTDS
[P19, P148]	TASTE
[P20]	Atelier B
[P20]	Kermeta
[P21]	AltaRica checker
[P22, P23, P30, P49, P51, P55, P63, P73, P79, P85, P100, P112, P113]	MagicDraw
[P24]	Rodin
[P25, P65]	Artisan studio
[P25]	Symphony
[P26, P67]	MRMC
[P27, P44, P45, P46, P50, P57, P102]	Rational rhapsody
[P27, P48, P68, P82, P87, P141, P144]	Custom tool/environment
[P27]	STK
[P27]	ModelLink
[P28, P40, P126]	Xtext
[P28, P41, P53]	Sirius
[P29, P124]	USE
[P32]	DynaSim
[P34, P80, P107, P114]	Enterprise architect
[P34]	Spin model checker
[P35]	RM2PT
[P36, P67, P105]	NuSMV
[P40, P81]	Epsilon
[P42, P53, P58]	Capella
[P44, P97]	Dymola
[P47, P64, P76, P95, P101, P109, P125, P134]	Tool independent/agnostic
[P49]	Creo
[P49, P146]	Solidworks
[P49]	UG
[P49]	Sketchup
[P49]	AutoCAD
[P49]	Unity3D
[P49]	MySQL Workbench
[P51]	OpenMBEE

Continued on next page

Just Accepted

Table 9 – continued from previous page

References	Tools used for V&V
[P51]	OpenCAE
[P52]	PREEvision
[P53]	Obeo designer
[P59]	AutoFocus3
[P59]	CosMOS
[P60]	BIP tools
[P60]	RERD
[P60]	DFinder
[P60, P140]	nuXmv
[P61, P117]	Excel
[P61]	Satellite tool kit
[P65]	Graphviz
[P65]	BRITNeY suite
[P67, P105]	COMPASS toolset
[P68, P81]	Zot
[P69]	ECPS verifier
[P70]	Cameo enterprise architecture
[P72]	Amesim
[P72]	NX 3D
[P72]	Star-CCM+
[P72]	Tecnomatix
[P72]	HEEDS MDO
[P73]	Systems tool kit
[P73]	ModelCenter
[P74]	ScenarioTools
[P76]	Topcased
[P76]	UML AP tool
[P77]	Siemens NX
[P78]	MetaGraph
[P79, P112, P113]	MOFLON
[P79, P112]	TiE
[P79, P112]	FUJABA tool-suit
[P82, P139]	CARLA
[P87]	JetBrains Meta Programming System
[P87]	APIS IQ-RM
[P93]	Jade
[P95]	Protoge
[P96]	OMEdit Tool
[P96]	GeNIe
[P97]	National instruments VeriStand
[P97]	TILSuite
[P97]	DIAdem

Continued on next page

Just Accepted

Table 9 – continued from previous page

References	Tools used for V&V
[P99]	RoboTool
[P99, P133]	FDR model checker
[P100, P141]	DEVS simulation framework
[P103]	SHARC
[P105]	Markov Reward modelchecker
[P106]	Scenario Question Query Engine
[P110, P111, P116]	TTool
[P111]	ProVerif
[P112]	MapleSim
[P115]	Romeo model checker
[P117]	AGI systems tool kit
[P120]	TineNET
[P121]	CASE tool
[P123]	Design/CPN
[P127]	Rational rose
[P128]	AutoPA3
[P132]	Pipe
[P133]	Astah modeling environment
[P137]	Dynacar
[P137]	Sabotage
[P138]	BiefCASE
[P138]	AGREE
[P143]	Ardupilot software in the loop simulator
[P145]	Gazebo
[P145]	Netlogo
[P146]	Unity3D
[P146]	Arduino tooling
[P149]	IF-toolset

A.8 Limitations

Table 10 detail the reported limitations of early V&V of the authors.

Just Accepted

Table 10. Identified limitations for early V&V.

References	Early V&V limitations
[P3, P6, P7, P8, P9, P10, P16, P17, P18, P27, P28, P31, P39, P42, P43, P46, P52, P53, P54, P55, P66, P72, P73, P74, P75, P77, P78, P81, P82, P83, P84, P85, P87, P89, P90, P91, P93, P94, P96, P97, P98, P100, P101, P102, P103, P104, P105, P107, P111, P112, P114, P115, P119, P121, P133, P137, P143, P144, P147]	None specified clearly
[P1]	Effort of introducing V&V methods in current practices
[P2, P4, P5, P6, P11, P12, P13, P14, P24, P29, P32, P33, P36, P40, P41, P60, P65, P70, P71, P76, P92, P99, P110, P118, P129, P131, P132]	Solution not fully developed
[P2]	Bias in interpreting the results
[P4, P61]	Resource usage not optimised
[P15, P19, P22, P47, P65, P106, P116, P140]	Limited automation
[P19, P67, P88, P109, P139]	Scalability issues
[P20, P22]	Lack of traceability between languages
[P21]	Lack of AI/ML
[P21, P30, P69]	Learning curve for beginners
[P23]	Not enough abstraction of models
[P23, P61, P109, P117, P139]	High complexity on process
[P25, P76, P80]	Lack of readability/understandability
[P30, P34, P70, P122, P128, P134, P135, P138, P148]	Expressability of descriptive language
[P26, P36, P47, P57, P62, P64, P86, P109, P123, P130, P134]	Integration issues between languages
[P35]	Tight coupling with application domain
[P37, P45, P50, P51, P58, P59, P60, P63, P68, P79, P92, P113, P120, P125, P126, P127, P136, P138, P146, P149]	Simplifications of analysis need to be taken into account
[P38, P44, P48, P95]	Method is too user dependent
[P49, P56, P108, P124, P132, P141]	Tools used
[P109]	Re-use of V&V
[P142, P145]	Management of abstraction
[P142]	Separation of concerns
[P148]	State-space explosion

Just Accepted