MBSE and V&V: a tool-equipped method for combining various V&V strategies

Blazo Nastov, Vincent Chapurlat, François Pfister, Christophe Dony

▶ **To cite this version:**

Blazo Nastov, Vincent Chapurlat, François Pfister, Christophe Dony. MBSE and V&V: a tool-equipped method for combining various V&V strategies. 20th IFAC World Congress, Jul 2017, Toulouse, France. pp.10538-10543, 10.1016/j.ifacol.2017.08.1309 . hal-01930280

HAL Id: hal-01930280
https://hal.science/hal-01930280

Submitted on 21 May 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 10538–10543

MBSE and V&V: a tool-equipped method for combining various V&V strategies

B.Nastov[1], V. Chapurlat[1], F.Pfister[1], C.Dony[2]

1 - Laboratoire de Génie Informatique et d'Ingénierie de Production - LGI2P
site de l'école des mines d'Alès, Parc Scientifique Georges Besse, F30035 Nîmes cedex 5, France
(e-mail: surname.name@mines-ales.fr)

2 - Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier - LIRMM
161 rue ADA, 34000 Montpellier (e-mail: dony@lirmm.fr)

Abstract: Model-Based System engineering (MBSE) promotes Verification and Validation (V&V) as crucial activities to demonstrate, during the system design stage and based on models, that a system meets requirements defined by stakeholders and that it fulfills its intended purpose. Model V&V activities are defined through the following strategies: *Model Appraisal, Guided Modelling, Simulation* and *Formal Proof*. Regarding the objectives of each individual strategy, they are considered as complementary, therefore mutually beneficial when combined, for reaching the overall V&V objectives. Various techniques and tools permit nowadays the implementation of each strategy. However, the successful combination and implementation of all four strategies remains still difficult (difference of concepts), time consuming (transformation and dedicated modelling are often requested) and generally expensive. This paper introduces a tool-equipped method for the successful and eased combination and implementation of all four V&V strategies to provide stakeholders with a high level of confidence in decision-making based on models.

© 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: MBSE, V&V, Model Appraisal, Guided Modelling, Simulation, Formal Proof.

INTRODUCTION: PROBLEMATIC AND OBJECTIVES

Crossing System Engineering and Model Driven principles, Model-Based Systems Engineering (MBSE) is defined as *the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases* [1]. MBSE promotes the creation and the management of various models of a so-called System of Interest (SoI). A model is built with respect to designer's objectives and is dedicated to a particular modelling viewpoint, e.g., requirements, functional, logical, physical, or behavioral. Models are nowadays created by the means of modeling languages known as Domain Specific Modeling Languages (DSML). Within this context, model Verification and Validation (V&V) activities are considered as a crucial support for designers at least during the upstream phases of system lifecycle processes [2]. So we focus hereafter on requirements engineering and architectural design processes.

Different V&V strategies can be applied, classified into: *Model Appraisal*, *Guided Modelling*, *Simulation* and *Formal Proof* [3]. They are complementary because they are based on various techniques, methods and tools, but considered efficient even if they are used separately.

However (see Figure 1) current V&V strategies are performed by the means of specific modelling activities and/or of more or less automated model transformations that take into account specificities and expected input models for the V&V environment. The achieved V&V results must then be translated-back and interpreted for the initial SoI models. Finally, based on these V&V results, the SoI models are revisited for improvement. The process is iterative and repeated until the SoI models reach a certain level of quality and of relevance considering the initial stakeholder objectives. They become then a relying and accurate source of information about the SoI, allowing SE experts to make decisions with confidence.

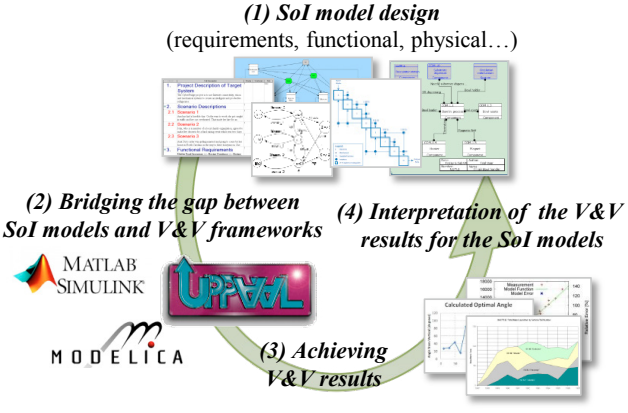

Figure 1: Current Model-Based V&V lifecycle

Despite the benefits of reusing already existing V&V methods and tools, several limitations remains still a subject of a debate. First, the use of different approaches,

2405-8963 © 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.1309

frameworks and tools is a relatively tedious and difficult task that requires huge learning, mastering and getting used to efforts. Second, the use of model-transformations might lead to information loss and misinterpretation of the V&V results. An accurate transformation of all SoI models into a single formal specification relative to the used framework is indeed difficult to define (e.g., considering that all the properties of the SoI are not, or cannot, be correctly mapped into the formal specification relative to the used framework) [4].

This paper introduces a tool-equipped method called *xviCore* (executable, verifiable and interoperable Core) for the successful and eased combination and implementation of all four V&V strategies to provide an overall better V&V for the achievement of more valid and less error-prone models. xviCore aims at bridging the gap between the designed SoI models and the existing V&V techniques and tools. The above highlighted issues are addressed by composing different research results relative to the V&V strategies into the expected method that aims to support *(1) guided modelling*, *(2) simulation* and *(3) formal proof*. It contributes also to the *model appraisal* strategy, by proposing SE experts to develop their own DSML and to implement the three strategies quoted above when performing their expertise.

The first issue concerns the variety of V&V techniques and tools for each strategy and by evidence the required learning, mastering and getting use to efforts. Of course, mastering xviCore and its techniques and tools is still required, but we argue that xviCore is optimized in terms of assisted and guided use and required technical skills to master, with respect to the current state of the art approaches discussed below. The second issue related to the use of model transformations is tackled by providing the means to directly manage SoI models, without transforming them to a third-party approach. So, after designing the SoI models by using our guided modeling approach, experts can execute models (separately of by combining various models) and formally check some of the expected SoI properties by using the same model or composition of models.

This paper is structured as follows. Section 2 introduces the general definitions for model verification and model validation, and details each of the above quoted V&V strategies with various references to existing works. Section 3 presents xviCore. It introduces first the needs that xviCore must address, before introducing its components (i.e., various techniques and tools) by quoting our works that can provide the reader with more details. Section 4 concludes the paper and states out research perspectives.

MODEL V&V: DEFINITIONS AND STRATEGIES

Within the MBSE context, models are used during the upstream phases of SoI development by experts to understand and argue about a given SoI (its structure and behavior), to perform various analyzes (e.g., analyses of SoI performance, safety, security, robustness, or resilience) and finally, to support them throughout decision making processes (e.g., an architectural choice about the SoI). These decisions impact as well the downstream phases of SoI development (e.g., an error detection based on a modelled SoI exponentially reduces the overall SoI costs, risks, safety or security, in comparison to late error detection - after the deployment of the real SoI). It is thus imperative, prior to any decision to assure that used models are error-free and valid, as much as possible, by performing V&V activities.

Model Verification aims to demonstrate that a model is *consistent* and *correct*, assuring the absence of modelling errors, mistakes and oversights, the respect to particular modelling rules (i.e., conformity to a metamodel, constraints and invariants verifications) and rules corresponding to domain expertise (i.e., good practices and patterns), and last the expected model behavior. *Model validation* aims to demonstrate that a model is *the right one and is trustworthy*, providing an accurate representation of a SoI in a viewpoint, as imagined by different stakeholders (i.e., the model respects the stakeholder and system requirements). The goal of V&V activities is to provide models of a SoI that can be used more easily and less costly to determine different solutions and to decide the Integration, Verification, Transition and Validation (IVTV) plan in the downstream phases of system lifecycle processes [2]. This must be done with respect to the needs and the strategy of an enterprise and its resource capabilities. V&V activities are performed considering SoI models, first separately and then together to provide more complete and suitable representation of a SoI, respecting the models' mutual coherence as well as their adequacy and global fidelity to the SoI.

There are four main strategies to implement V&V activities based on models: *Model Appraisal*, *Guided Modelling*, *Simulation* and *Formal Proof* [3]:

Model Appraisal involves *(1) human resources* (i.e., domain experts that have experience in the evaluation and the appraisal of models of a SoI relative to their domain of expertise) and *(2) technical resources* (i.e., different approaches, frameworks and tools for V&V might be requested to assist and help the experts during the process). This is an efficient method for determining the quality of a given model, but is relatively expensive, particularly in a multidisciplinary context where multiple V&V specialists with different domain expertise are required.

Guided Modelling consists in guiding and assisting model designers during the SoI modelling phase. We distinguish: *(1) pattern-based approaches, (2) boilerplate-based approaches* and *(3) feedback-based approaches*. The *pattern-based approaches* promote the use of modeling patterns, hints and frameworks for guiding experts during a design process. The goal of pattern-based approaches is to eliminate structural design errors by proposing possible and already validated solutions to a problem considered then to be good practices. For instance, a model-driven framework for guided design space exploration is proposed in [5]. The *boilerplate-based approaches* introduce template models that contain crucial, already validated information about a given domain. The goal of boilerplate-based approaches is to ease the work of designers by providing a solid starting point basis with pre-verified information. For instance, the European CESAR project [6] proposes boilerplates-based requirements specification language. The *feedback-based approaches* promote the reuse of models and examples that are considered to be, at least, verified and validated, or, at best, standardized in a given domain. The goal of feedback-based approaches is to share the domain experience (problems,

causes, and possible solutions) with designers of the same domain that attempt to solve similar problems. For instance, in [7], it is proposed to improve the automation in the model-driven engineering, based on examples.

Simulation consists in executing SoI models to observe their behavior, i.e., to simulate the behavior of a SoI. The simulation has numerous benefits and is for a long time recognized for its relevance within various industrial contexts. It is indeed generally cheaper, safer and faster than other strategies. A simulation can become more realistic if required, by increasing the number of parameters taken into account and the model hypothesis (discrete-events, continuous or hybrid). An example of a simulation framework is Simulink [8].

Formal Proof is based on the use of formal methods, languages and tools. Formal methods are mathematically based methods for the specification, development and verification of systems. They leverage the use of formal languages that have solid mathematical semantics. As a result, formal system specifications are unambiguous and can be used to perform mathematical analysis, contributing to the reliability and robustness of a design. Formal methods are based on two different approaches for formal verification: *(1) model-checking* or *(2) theorem proving*. *Model-checking* is an approach to check if a given behavioral specification of a system respects some properties. It consists first in specifying the system behavior through a formal specification and then the requirements to be verified as formal properties. Second, specified properties are verified based on a systematic and exhaustive exploration of the system specification (i.e., by exploring all possible states of this specification). An example of a model-checking framework is UPPAAL [9]. *Theorem proving* is a technique for formal verification that consists in generating a collection of mathematical proof obligations from a system specification. These obligations imply conformance of the system to its specification. They can be formally proven by using a theorem prover. An example of a theorem proving framework is Coq [10].

XVICORE: A METHOD FOR THE IMPLEMENTATION OF THE MODEL V&V STRATEGIES

Models are nowadays created and managed by using Domain Specific Modeling Language (DSML). So, DSML must first be designed and integrated in such a way so that they can be used for *(1) modelling that covers a total SoI representation* and for *(2) Verification and Validation (V&V)* of such representation, as suggested by the SE 2020-2025 challenges [11]. We argue in [12], [13] that the composition of a DSML dictates the overall quality of designed models in terms of representational (i.e., to cover a SoI) and V&V (i.e., to represent "correctly" and "accurately" a SoI) capabilities. For the first need of *modelling that covers a total SoI representation* by creating, integrating and graphically representing various models, the syntaxes of the requested DSML are the crucial component. DSML syntax can be divided into an *abstract syntax* (i.e., a metamodel that represents through a graph of classes, the concepts of a domain and their interactions) and a *concrete syntax* (i.e., the graphical or textual representation of each concept composing the DSML e.g. a function or a requirement) [14].

For the second need of *Verification and Validation (V&V)* covering the whole representation of a SoI, DSML must include and integrate *semantics*. Semantics is often neglected or, when needed, provided by means of translating the DSML into third-party formalisms [15]. This is a key limitation for implementing "direct" V&V activities without transforming into third party approaches [16]. According to [17], DSML semantics can be divided into a *static semantics*, representing concept meaning and different types of constraints (e.g., pre and post conditions, invariants, etc.) and a *dynamic semantics*, dealing with the way models behave.

However, on the one hand, building DSML remains currently difficult for SE experts with low level of software engineering skills [13]. To address this issue, we highlight the following needs. *(1) Guide the modelling towards models that are "well-constructed"* by providing the use of design patterns, anti-patterns and invariants or rules to be used in modelling, the reuse of tested and approved models, and the reuse of test sets that have already been used on other models and appear to be solid and valid. *(2) Guide the design of the graphical or textual representation of models* by assisting experts to design graphical of textual concrete syntaxes and by automating this process as much as possible. *(3) Ergonomics and Users' Autonomy*, i.e., to assist, ease and support the work of different users (DSML designers, DSML users or Model designers and Model users) throughout the whole modelling and V&V processes. The goal is to provide these users with a certain level of autonomy, minimizing the needs for external support. *(4) Collaborative multi-domain work for an efficient modelling and V&V,* i.e., the means for collaborative work to different users from different domains with different levels of expertise throughout the modelling processes for the successful design and integration of DSML and models, but also throughout the V&V activities to improve the quality of DSML and models.

On the other hand, DSML are usually limited in terms of modelling and V&V that covers simultaneously several views [13]. To address this issue, we highlight the following needs. *(5) Syntactical model integration.* When modeling a SoI, various interconnected viewpoint models are designed. Each model must be relevant even dedicated to the needs of different stakeholders involved in the design process. When all the viewpoint models are put together and integrated they form a "composite model", covering a more expressive, realistic and complete representation of a SoI. The syntactical integration aims to relate the structures and representations of different SoI viewpoints. *(6) Syntactical DSML integration*. Models can be integrated syntactically only if the integration points between different types of models are defined at a DSML level. This consists in integrating the syntaxes of the used DSML forming a "composite DSML", providing the means for a more expressive, realistic and complete representation of a SoI. *(7) Semantical model integration*. In a similar way, the whole behavior of a SoI can be represented by mixing or aggregating the behaviors of different viewpoints, even though these behaviors might be based on different functioning hypothesis (e.g., different level of details or different objectives). The semantical integration aims to relate the behavior of all SoI viewpoints so that all models are considered during V&V activities. *(8) Semantical*

DSML integration. The integration of the behavior of SoI viewpoints is also defined at DSML level by integrating the semantics (dynamic or executional) of the used DSML. *(9) Model and DSML Interoperability*. Both syntactical and semantical integrations are here-considered as sufficient to reach a certain level of model interoperability as it is proposed in other works such as [18], but also a certain level of DSML interoperability. *(10) Simulation and Formal proof covering total SoI representation*. When DSML are integrated syntactically and semantically, they provide the mean to create models that can also be syntactically and semantically integrated. The goal of such integration is to allow a SoI simulation based on a coordinated execution of all SoI viewpoint models and a SoI formal proof based on all SoI models. *(11) Trace V&V analyses*. A mechanism that provides an execution trace (i.e., a record about the SoI simulation), and a proof trace (i.e., a record about the satisfiability of a set of formal properties during a SoI simulation).

As a solution to these needs, we propose the tool-equipped method xviCore (executable, verifiable and interoperable Core) introduced in [12], [13], [15], [19], [20]. We discuss hereafter the different solutions proposed by xviCore with respect to the above needs:

- *Guiding the design of well-constructed models*. Using a design pattern approach is a way practitioners can represent invariant knowledge and experience in design. The goal is then to help designers to identify and solve various kinds of problems by drawing or imitating such knowledge and experience. The objectives are: (1) to improve performance (comprehensiveness, relevance) and reliability (proven solutions, justified and context-based); (2) to gain economic value (time savings); (3) to facilitate collaborative work by sharing design pattern repositories; (4) to improve by default the quality level of the model, being able to show the SoI achieves some well-known functionalities or characteristics. In [21], we promote a pattern metamodel that formalizes the application context, the problem to solve, the pattern structure, the benefits and drawbacks. We define three main problems to solve when defining Patterns. First the pattern model itself may allow the description of a promising solution model, thanks to a problem model arising in a given environment for reaching specific objectives and fulfilling given requirements. A Pattern may also be considered as an anti-pattern describing then a solution to avoid. Second, it is requested to establish a Patterns Repository and this needs a pattern modelling language (a particular DMSL) that allows system architects to translate their experience i.e. to formalize encountered problems, contexts, and adopted solutions. Finally, patterns may be embedded in existing should then be identified by pattern recognition mechanisms identified in [21].
- *Guiding the design of graphical concrete syntaxes*. For the design of DSML concrete syntaxes, we propose the use of our pattern-based approach that is introduced in [22], [23]. This approach includes concepts and mechanisms allowing to guide and to assist an expert from any engineering domain to define and formalize the concrete syntax of a graphical DSML considered as relevant in this domain. We define multiple classifications of the abstract syntax elements based both on the abstract syntax and on the concrete syntax. Grounded on these classifications, our approach for a given abstract syntax can generate automatically a concrete syntax, by a process that we refer to as a *graphical role election process*. The generated concrete syntax contains information about how the different instances of abstract syntax concepts can be embedded into each other and how they can be related to each other. More detailed information such as their size, color, shape, labels, etc., is not generated and thus engineers must furthermore manually complete the generated concrete syntax. But the generated concrete syntax is nonetheless operational and can be used for the basic graphical representations of models. For more details readers are invited to see [22], [23].
- *Design and Integration of DSML syntaxes*. For the syntactical design and integration of DSML we propose our approach Diagraph [24]. Diagraph is an agile method for the synthesis of graphical DSML leveraging, assisting and easing the use of the Eclipse Modelling Framework (EMF) and the Graphical Modeling Framework (GMF). It introduces a concurrent and iterative process for the simultaneous design and integration of abstract syntaxes based on the EMF and concrete syntaxes based on the GMF. Diagraph includes also the above discussed approch for an automatic generation of concrete syntaxes. For more details on the Diagraph approach, readers are encouraged to see [24], [25].
- *Design and Integration of DSML semantics.* For the semantical design and integration of DSML, the xviCore method put together the Diagraph approach for the design of DSML syntaxes (abstract and concrete), the UPSE-SL framework (a model verification framework for systems engineering) [16] for the design and verification of DSML static semantics, and the eISM (extended Interpreted Sequential Machine) [15] behavioral modelling language for the design, integration and verification of DSML dynamic semantics. The USE-SL framework promotes the CREI property modelling language for the design of static semantics as modelling and system properties based on integrated abstract syntaxes and dynamic semantics. The dynamic semantics created by the eISM language are provided as a set of integrated discrete-events behavioral models that have a formal underlying structure based on the Linear Temporal Logic (LTL) and thus can be formally verified. DSML designed by xviCore are denoted xviDSML (executable, verifiable and interoperable DSML) and they can be used to design executable, verifiable and interoperable models (xviModels). An xviDSML represent one integrated DSML in a composite DSML. For more details on xviCore, readers are invited to see [12], [15], [19], [20].
- *Mechanisms for Simulation and Formal Proof.* To put in use the semantics of an xviDSML for the V&V of xviModels, the xviCore approach provides a mechanism for simulation and a mechanism for formal proof. The here-considered simulation consists in using the dynamic semantics defined as sets of eISM behavioral models of one of several xviDSML to execute xviModels. These behavioral models require mechanisms for synchronization and centralized data and event exchange such that each step of the execution, all behavioral models from one or several xviDSML are synchronously executed based on data derived

from one or several xviModels. The data changes in the process, consequently changing the characteristics of the xviModels. Stakeholders observe these changes and judge about the relevance of the model vis-à-vis the expected reality. We propose a solution for such concurrent execution of behavioral models and centralized data and event exchange by applying the blackboard design pattern [26] and several original synchronization rules. For more details, readers are encouraged to see [15], [20]. The here-considered formal proof process consists in the verification of the system and modelling properties defined by the static semantics of an xviDSML. These properties are verified based on a formal specification that is extracted from the xviDSML's abstract syntax and dynamic semantics. First, the abstract syntaxes are defined as metamodels that naturally have an underlying structure based on an oriented graph that can be used for formal verification. Second, dynamic semantics designed by eISM have a formal underlying structures that can be written in the form of elementary valid formals (EVF) as proposed in [27]. As for the verification mechanism that will check the modelling and system properties based on these specifications, the UPSE-SL framework leverages the use of third-party model-checkers that can operate directly on these specifications without transforming them. For instance, the OCL interpreter [28] can be used to check properties directly based on a metamodel.

- *xviCore lifecycles for Users' Ergonomics, Autonomy and Collaborative multi-domain work.* xviCore introduces two formalized lifecycle to guide and assist the design, integration and use of xviDSML and xviModels. The first lifecycle, denoted "xviDSML and xviModel lifecycle" is composed of several phases and sub-phases. Each phase highlights the types of properties that need to be designed for each component of an xviDSML or an xviModel, the languages that need to be used and the V&V analyses that need to be performed. The goal of this lifecycle is to provide different experts with a certain level of autonomy, by assist and guide them as much as possible for the design and use of xviDSML and xviModels, minimizing the needs for external support. The second lifecycle, denoted "Composite DSML and model lifecycle", introduces several phases and sub-phased for the syntactical and semantical integration of xviDSML and xviModels. The goal of this lifecycle is to highlight, ease and assist the collaborative multi-domain work of different experts by allowing them to integrate their domain knowledge through the integration of their dedicated xviDSML and xviModels.

So, xviCore emphasize the guided modelling V&V strategy for the design of DSML syntaxes and models during the design phases of DSML and models. The simulation and formal verification V&V strategies are also emphasized to ensure the absence of modeling and design mistakes in DSML and models, throughout both model design and run time phases. xviCore contributes also for the model appraisal V&V strategy by guiding and assisting the design, integration, use, verification and validation of DSML and Models, and by proposing the previous three V&V strategies (guided modelling, simulation and formal proof).

Figure 2 shows an activity diagram that illustrates the interactions between different V&V strategies proposed by xviCore.

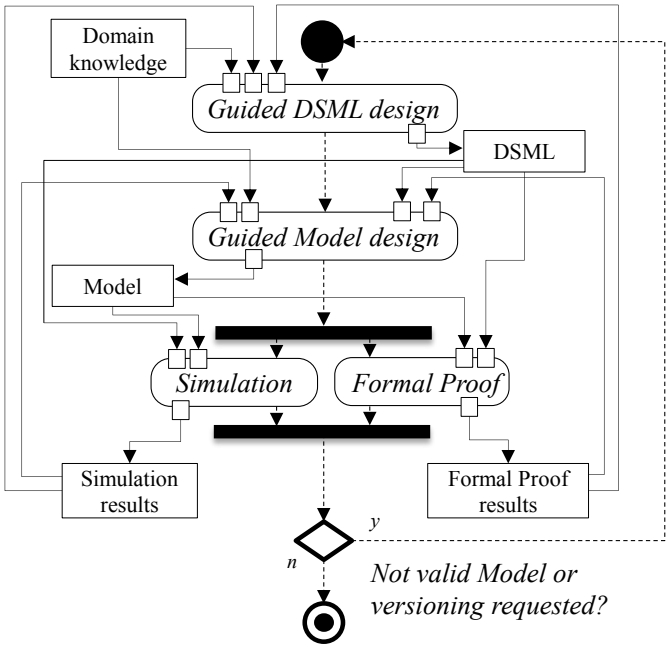

Figure 2: The interactions between different V&V strategies proposed by xviCore.

CONCLUSION

This article shows it is possible to combine several V&V strategies gaining time, quality and operational usage for designers involved in model based systems engineering activities. It aims to link different research results that we have proposed during the past five years into a tool-equipped method called *xviCore*. The goal here is not to demonstrate that our method xviCore is sufficient and will revolutionize V&V techniques and tools. However, we argue that *xviCore* is an interesting and beneficial alternative that can help designers to become able to stay autonomous and efficient, to model a system without huge efforts and doubts, to work with confidence, and to reach different V&V results.

The *xviCore* method is today partially equipped and has already been tested on several DSML largely used in MBSE. Our current primary goals are to complete the tooling and to adopt this method in research projects during which designers are more interested to evaluate and prove expertise when necessary and model without ambiguities some non-functional properties of a SoI and particularly here concerned resilience.

ACKNOWLEDGEMENTS

This research work is partially funded by the French MAIIEUTIC Project[1]. The authors acknowledge the French CARNOT M.I.N.E.S. and the French National Research Agency (ANR) who fund the MAIIEUTIC Project.

[1] http://maiieutic.mines-ales.fr/contenu/projet-maiieutic

REFERENCES

[1] INCOSE, "Systems Engineering Vision 2020," *INCOSE-TP-2004*, 2007.

[2] INCOSE, "Guide to the Systems Engineering Body of Knowledge (SEBoK)," *v. 1.6*, 2016. [Online]. Available: http://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK).

[3] V. Chapurlat, "Vérification et validation de modèles de systèmes complexes: application à la Modélisation d'Entreprise," University of Montpellier II [HDR in French], 2008.

[4] B. Nastov, V. Chapurlat, C. Dony, and F. Pfister, "A verification approach from MDE applied to model based systems engineering: XeFFBD dynamic semantics," in *Complex Systems Design and Management - Proceedings of the 5th International Conference on Complex Systems Design and Management (CSD&M 2014)*, 2015, pp. 225–235.

[5] Á. Hegedüs, Á. Horváth, and D. Varró, "A model-driven framework for guided design space exploration," *Autom. Softw. Eng.*, vol. 22, no. 3, pp. 399–436, Sep. 2015.

[6] CESAR, "Cost-efficient methods and processes for safety relevant embedded systems," 2012. [Online]. Available: http://www.cesarproject.eu/.

[7] M. Faunes Carvallo, "Improving automation in model-driven engineering using examples," University of Montréal, 2013.

[8] Mathworks, "Introduction to Simulink," *Matlab Simulink User's Guid. R2014b*, pp. 1–69, 2014.

[9] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *Int. J. Softw. Tools Technol. Transf.*, vol. 1, no. 1–2, pp. 134–152, 1997.

[10] Y. Bertot, "Coq in a Hurry," 2006.

[11] AFIS, *Ingénierie système: la vision AFIS pour les années 2020-2025*. AFIS (French Association for Systems Engineering) [in French], 2012.

[12] B. Nastov, V. Chapurlat, C. Dony, and F. Pfister, "A Tooled Approach for Designing Executable and Verifiable Modeling Languages," *INSIGHT Q. Mag. Int. Counc. Syst. Eng.*, vol. 18, no. 4, pp. 31–33, 2016.

[13] B. Nastov, "Contribution à une méthode outillée pour la conception de langages de modélisation métier interopérables, analysables et prouvables pour l'Ingénierie Système basée," University of Montpellier [Phd in English], 2016.

[14] A. G. Kleppe, "A language description is more than a metamodel," in *the 4th International Workshop on Software Language Engineering*, 2007.

[15] B. Nastov, V. Chapurlat, C. Dony, and F. Pfister, "Towards semantical DSMLs for complex or cyber-physical systems," in *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering (ENASE 2016)*, 2016.

[16] V. Chapurlat, "UPSL-SE: A model verification framework for Systems Engineering," *Comput. Ind.*, vol. 64, no. 5, pp. 581–597, Jun. 2013.

[17] B. Combemale, X. Crégut, P. L. Garoche, and X. Thirioux, "Essay on semantics definition in MDE: An instrumented approach for model verification," *J. Softw.*, vol. 4, no. 9, pp. 943–958, 2009.

[18] F. Belkadi, N. Troussier, B. Eynard, and E. Bonjour, "Collaboration based on product lifecycles interoperability for extended enterprise," *Int. J. Interact. Des. Manuf.*, vol. 4, no. 3, pp. 169–179, 2010.

[19] Nastov Blazo, "Contribution to model verification: operational semantic for System Engineering modeling languages," in *the 3th National Conference on Software Engineering (CIEL 2014)*, 2014, pp. 88–90.

[20] B. Nastov, V. Chapurlat, C. Dony, and F. Pfister, "Towards V&V suitable Domain Specific Modeling Languages for MBSE: A tooled approach," in *the 26th Annual INCOSE International Symposium (IS 2016)*, 2016.

[21] F. Pfister, V. Chapurlat, M. Huchard, C. Nebut, and J.-L. Wippler, "A proposed meta-model for formalizing systems engineering knowledge, based on functional architectural patterns," *Syst. Eng.*, vol. 15, no. 3, pp. 321–332, Sep. 2012.

[22] B. Nastov and F. Pfister, "Towards automatic graphical concrete syntax generation for domain specific modeling languages | Vers la génération des syntaxes concrétes graphiques pour les langages de modélisation métier," *Ing. des Syst. d'Information [in French]*, vol. 20, no. 2, pp. 67–91, 2015.

[23] B. Nastov and P. François, "Experimentation of a Graphical Concrete Syntax Generator for Domain Specific Modeling Languages," in *the 32th National Conference on INFormatique des ORganisation et Systèmes d'Information et de Décision (INFORSID 2014)*, 2014, pp. 197–213.

[24] F. Pfister, M. Huchard, and C. Nebut, "A framework for concurrent design of metamodels and diagrams towards an agile method for the synthesis of domain specific graphical modeling languages," in *Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS 2014)*, 2014, vol. 2, pp. 298–306.

[25] F. Pfister, "Contribution à la conception simultanée de syntaxes abstraites et de notations graphiques : application à la définition de langages pour l'Ingénierie Système," University of Montpellier [Phd in French], 2013.

[26] R. Engelmore and T. Morgan, *Blackboard systems*. Wesly publishing, 1988.

[27] M. Larnac, V. Chapurlat, J. Magnier, and B. Chenot, "Formal representation and proof of the interpreted sequential machine model," in *Computer Aided Systems Theory — EUROCAST'97*, 1995, pp. 93–107.

[28] OMG, "Object Constraint Language (OCL) Specification v2.4," 2014.