Towards Agile Model-Based Systems Engineering

Joachim Denil^{*†}, Rick Salay[‡], Chris Paredis[§], Hans Vangheluwe^{*†¶}

*University of Antwerp, Belgium

Email: {Joachim.Denil,Hans.Vangheluwe}@uantwerpen.be [†]Flanders Make, Belgium [‡]University of Toronto, Canada Email: rsalay@cs.toronto.edu [§]Georgia Institute of Technology, USA Email: Chris.Paredis@me.gatech.edu

¶Mcgill University,Canada

Email:hv@cs.mcgill.ca

Abstract—Engineering organisations following a traditional development process often suffer from under-specified requirements and from poor responsiveness to changes in those requirements during the course of a project. Furthermore, these organizations need to deliver highly dependable products and decrease time-tomarket. In the software engineering community, Agile methods have been proposed to address similar issues. Pilot projects that apply agile approaches in Cyber-Physical Systems (CPS) engineering have reported some success. This position paper studies the challenges faced when adopting an agile process to design CPS. These challenges are broken down into their essential components and solutions are proposed, both pertaining to model/simulation management and to processes.

I. INTRODUCTION

With today's fast pace of change and increasingly complex requirements, new ways must be found to engineer systems more quickly and with higher integrity. Traditional waterfalllike life-cycles such as the "V" process suffer from poor responsiveness to changes in the requirements of the system under construction. These changes are a consequence of legislative changes, changes in the market demand of the customer, and company policy. Often, changes stem from underspecified or even missing requirements. Rigid processes, such as the "V" process, freeze the requirements of the system throughout the design cycle. Consequently, wrong assumptions or underspecified requirements are discovered very late in the development cycle, for example, only once a full system prototype is demonstrated to the customer. This results in costly redesign.

In response to similar problems, the software engineering community has introduced more lightweight development or iterative methods such as Scrum [1]. These development methods are now commonly referred to as *Agile*. The principles and concerns for Agile practices in software engineering were published in the *Manifesto for Agile Software Development* [2]. The main principles of agility according to these authors are a preference for: (a) individuals and interactions over processes and tools, (b) working software over comprehensive documentation, (c) customer collaboration over contract negotiation, and (d) responding to change over following a plan.

There is evidence that agile methods are more successful than traditional methods. For example, a recent quantitative analysis of development projects showed that agile methods provide a statistically significant benefit to project success factors of efficiency, stakeholder satisfaction and perception of project performance [3]. Unfortunately, agile software methods cannot be transposed to the engineering context without adaptation. For example, having a working system (i.e., corresponding to "compiled, working code" in the software realm) amenable to end-user evaluation, after each iteration, is typically infeasible. A possible alternative is to have working *models* of the system [4]. 'Working model" means that stakeholders must be able to meaningfully evaluate the system by model-checking, (co-)simulating, etc. the model. Clearly, this can only be achieved with appropriate tool support.

In this position paper, we identify the ways in which agile methods must be adapted to fit the model-based engineering of Cyber-physical Systems (CPS) context and describe the methods, techniques and tools needed to support these adaptations.

Section II presents the state of the art in agile engineering. Most of the literature on agility in the mechatronic, software-intensive and CPS domains re-use the principles of agile software development with little change. (and hence, a software focus remains). We reflect on the original needs for agility: to reduce the risk of building a wrong system. Section III, identifies the main challenges of agile CPS For each of these challenges, section IV investigates appropriate model/simulation management and process solutions. Figure 1 serves as a map for the relationships between challenges and solutions described in this paper. Finally, section V concludes.

II. RELATED WORK

Woodcock adapted the agile manifesto for systems engineering [4]. Recently, Bruce Douglass [5] adaptated the IBM Harmony systems engineering methodology to be more agile. In his approach, models replace code as the artifact produced in an iteration and must be "executable." Klein and Reinhart provide mechatronic development companies with approaches to incorporate agile methods and techniques in their current processes [6]. Two specific processes, a vertical and horizontal

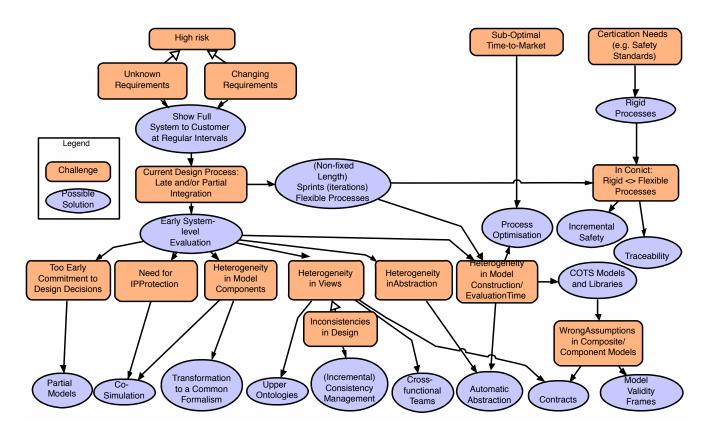


Fig. 1. Overview of challenges and possible solutions

process, combine techniques from agile with the traditional gated processes in mechatronic engineering.

III. CHALLENGES OF AGILE CPS DESIGN

In 2014, Tetra-pack undertook a pilot project to test the use of agile methods in engineering [7]. They combined a Scrumlike process during the development phase with a traditional V waterfall process for full system verification before releasing to the customer. A key motivation for trying out an agile method was to avoid losing customers due to delays and be able to respond more quickly to new trends. Overall, they observed better employee motivation and better work efficiency in the project.

[8] summarizes several studies on the use of agile methods in aircraft systems integration at Johns Hopkins. The Cube-Sat multi-mission bus demonstrator is engineered using five engineering teams. They experienced that co-located teams reduced communication complexity. Incremental test methods allowed them to find problems early. [9] reports on three case studies where agility is introduced in a non-software environment at three different companies: Marel GRB, Saab, and Andritz Hydro. The experience shows that it is possible to use agile principles for electro-mechanical systems design.

These approaches adapt agile processes without any extra methods, techniques and tool support. In this paper we identify several enabling techniques to help engineers being more agile in the design of CPS. In the following, we identify a number of challenges in the design of Cyber-Physical Systems. CPS are characterised by a tight integration and coordination between physical, computational and network components [10]. CPS can be seen as the natural evolution of software intensive and mechatronic systems to a higher complexity. Application domains include medical devices, advanced automotive systems, aeronautics, etc.

a) High Risk in Engineering: : Engineers want to reduce the risk of failure in both the product or the project [11]. A product fails because of problems such as reliability issues or working with wrong, incomplete, or changing requirements. The risk of changes in the requirements needs to be managed and is the main driver to include agile principles into a design process.

b) Dependability Challenges: CPS must be dependable. Dependability is a measure of the availability, reliability, durability, safety and security of a system. Dedicated standards address dependability in different domains. A well known example is the ISO26262 standard addressing the functional safety of an automotive system. Functional safety standards impose a *rigorous process* with different phases on the product developers.

c) Sub-optimal Time To Market: Time to market (TTM) is closely related to value creation as profit earned by a product

not only depends on engineering time and production cost but also on its launch date [12]. Most engineers do not take this into account when designing a system. It should be possible to optimize a design process to take the full product value, including TTM, into account.

d) Heterogeneity Challenges: Several heterogeneity challenges can be identified: (a) Due to the tight integration between different physical, computation and communication components, the design of CPS is heterogeneous as well. The engineers, as experts in their own domain, work in domainspecific "silos", and may have a different vocabulary. It is important that these different domain experts can communicate and understand the cross-functional design issues, and understand the interest of the other stakeholders in the system under design [13]. (b) The complexity of CPS also leads to the use of *multiple views* on the system under design. Multiple views address different concerns such as safety, energy consumption, desired function, etc. This results in sometimes unknown relations and dependencies between the different views [14]. (c) The problem is further exacerbated by the use of *different modelling formalisms*. Each engineer uses a dedicated set of most appropriate design languages, e.g. Simulink diagrams for control engineering, 3D CAD models for geometric engineering, etc. The dependencies between the different silos and views are rooted in dependencies and overlaps in the semantic domain of the models [15], [16]. (d) Another dimension of the heterogeneity is a consequence of the difference in time to create or evaluate a model. 3D models are much more labour-intensive to create compared to a 1D model of the physics. Furthermore, computational fluid dynamics simulations are much more computationally demanding compared to the simulation of a causal plant model in Simulink. (e) Finally, there can also be heterogeneity in the levels of abstraction between the different artifacts. When building a system, certain parts of the system may require more detailed models than others, to attain a desired level of fidelity. In case components are sourced from external suppliers, the level of model detail may be imposed extrinsically.

e) Intellectual Property: Internal and external suppliers have a vested interest in the knowledge they build up through defining models of their components. Therefore it is important that methods, techniques and tools are able to protect this intellectual property of the different stakeholders. Such protection may impede certain kinds of full system evaluation however.

IV. POSSIBLE SOLUTIONS

In this section we look at the possible scientific solutions for the presented challenges. Figure 1 serves as a map for the relationships between the challenges and solutions.

a) Show Working System to Customer at Regular Intervals - Early System-Level Evaluation: To reduce the risk of building a system with wrong assumptions or having to start over when the requirements change during the design process, the agile software community introduced the principle of continuous delivery. Working software is frequently produced and evaluated by the customer. This allows the customer to identify wrong or unknown requirements early in the design process. This is in contrast with the current design processes in the CPS domain. Mostly, engineers use rigid processes that only integrate the system very late in the design cycle. System engineers introduce some process steps to partially mend this issue. An example of this are the construction of (physical) system prototypes. However, building such a prototype is a costly and time-consuming activity. Furthermore, even when standardised physical test benches are available, system evaluation can be extremely expensive and time-consuming while not providing the needed insights into the unknown requirements and wrong assumptions [17]. A translations of this principle for systems engineering is early full system-level analysis. The analysis serves the same need as the "working software" but can be completely virtual using an evaluatable model. Because the evaluation occurs at system-level, the system level properties are being analysed. 'Evaluatable model" means that stakeholders must be able to meaningfully evaluate the system by model-checking, (co-)simulating, etc. the model [4]. Appropriate tool support is required to achieve this.

A particular challenge is dealing with the different types of heterogeneity during the design of CPS. Below we describe multiple of such possible solutions that deal with system-level evaluation and heterogeneity.

b) (Non-)fixed-length Sprints: Having "Evaluatable models" early on in the process is only one part of the agile solution. The other part is having these evaluations at regular intervals in the design process of the CPS. Agile software engineering introduces the concept of a fixed length "sprint". Within a sprint, a system fully realizing a set of selected requirements is constructed that can be evaluated by the different stake-holders.

Requirements are chosen for inclusion in a sprint, based on risk analysis. High-risk (critical to system operation, high uncertainty, low confidence, ...) requirements are best included as early as possible. Hidden and unknown dependencies may occur between different requirements. It is important that engineers have the methods, techniques and tools at their disposal to chart these dependencies and the associated risks. Two derived challenges arise. (a) Iteratively creating the system, without an up-front choice of requirements to include in each sprint, conflicts with the rigorous processes that safety and other dependability standards prescribe. A possible solution is to adhere to the safety process in the final sprint, linking all information using traceability, or, incrementally building the safety case(see later). (b) The time-heterogeneity in creating and evaluating parts of the system has to be taken into account during requirement selection and risk analysis. Process optimisation might provide guidance (see later).

c) Co-Simulation: CPS are designed using a multitude of languages and tools. The agile approach requires that these heterogeneous models are meaningfully combined for full-system evaluation at the end of every sprint. Cosimulation [18] is a technique that allows coupling of blackbox simulation units that do not expose all information in the models. This protects intellectual property of internal and external component suppliers. Co-simulation is an IP protecting technique that allows coupling of black-box simulation units. E.g. the Functional Mock-up Interface (FMI) is a standard for coupling continuous-time simulation components. However, FMI does not specify how to simulate the different mock-up units together. A so-called master algorithm needs to be used. A master algorithm reasons about the coordination/communication between the different FMUs and how to solve emerging global model artefacts such as algebraic loops. Generic co-simulation algorithms exist [19] as well as techniques to generate an optimised orchestration [20]. Coupling discrete-event models to continuous-time models is still needed for FMI. Different solutions have been proposed [21]. [22]. An overview of different methods and techniques for co-simulation can be found in [18].

There are a number of challenges for co-simulation, e.g. for proving the correctness of a co-simulation scenario or combining units with different levels of detail. A scenario can be correct in the computer science sense, the correct data types are coupled, etc. In the mathematical and numerical sense, correct numerical integration steps are selected. And finally, adhering to the laws of physics, e.g. energy and momentum are conserved. An example is the formal verification approach in [23]. A major challenge is dealing with multi-abstraction/purpose/precision components: how can we take advantage of an FMU that contains different abstraction levels in the black-box.

d) Transform to a Common Formalism: Another technique to couple heterogeneous models, is to transform all components to a common formalism [24] which support model evaluation. To bridge to cognitive gap between the original design model and the automatically generated model in the common formalism, back-annotation is required. Vangheluwe identified DEVS as a common denominator for hybrid modelling [25]. Another approach is to create a combination of multiple formalisms. This implies, as with co-simulation, that the coordination between the different formalisms is specified. Fragment-based composition of different languages at both the syntactic and semantic level remains a challenge.

e) Partial Modelling: In the agile model based systems engineering approach, we advocate early and regular fullsystem evaluation. However, to allow for simulation and analysis of early models, one may need to fix certain design choices while the most valuable action would be to defer this choice. A possible solution is to explicitly model the deferment or partiality of the design choice. This requires reasoning, not about a single design, but about a set of designs. This is called Partial Modeling [26] or Models with Holes. In [27], the authors formalise partial models that support four different annotations:(a) May partiality: the element may exist, (b) Abs partiality: we do not know whether it is a set of elements, (c) Var partiality: whether the element should be merged with another element and (d) OW partiality referring to the (in-)completeness of the model. Different operations needed for design have been "lifted" to partial models, such as verification [26] and model transformation [28]. The above techniques can be transposed to the architectural models used in current engineering approaches. Extending the partiality to other formalisms remains a challenge. E.g. in a model of the physics, a component parameter may be replaced by a distribution, resulting in a stochastic differential equation. All operations, such as simulation and transformation should be applied to a collection of models rather than to a single model. This can drastically affect performance. Furthermore, the trade-off between partial models and the implied computational overhead should be supported by guidelines.

f) Upper Ontologies: In a multi-view, multi-disciplinary context, it is crucial that engineers are aware of when they reason about identical concepts, even when they use their own, domain-specific terminology and formalisms, (Upper) ontologies allow for the high-level classification of real-world entities as well as of relations between these entities. New knowledge can be inferred by reasoning over these ontologies.

g) Incremental Consistency Management: Because of multi-view modelling approaches in the design of CPS, both syntactic and semantic overlaps occur. After each sprint, it is important to end up with a consistent system model. Therefore, dedicated consistency management approaches are needed. This can be supported by having a dedicated role in the team to manage inconsistencies. Model inconsistency is a well known problem in model-oriented approaches. Many solutions act on the syntactic level, e.g. [29] On the semantic level, Qamar creates a dependency modelling language to map the semantic dependencies between different artifacts in a project [30]. Herzig et al. create a Baysian inference mechanism to identify possible semantic overlaps in models [31]. Vanherpen et al. reason with upper ontologies to map the different semantic overlaps in a project [16]. In an agile context, lightweight solutions to consistency management are needed. The above methods can be incremental and combined with learning approaches and ontology-based reasoning.

h) Cross-functional Teams: Traditional engineering processes put individual engineering disciplines in distinct silos with communication structured along well-defined hand-offs. This discourages and interferes with interactions across disciples that could help detect and resolve conflicts early. An agile approach uses cross-functional teams to leverage the different skill sets and perspectives effectively during development. This works well in software development, as the different skills are closely aligned. Code is after all the common semantic domain. The challenge to adopting this approach in engineering is that the languages, tools and cultures of the different disciplines are quite distinct making collaboration difficult, even if the participants are all in the same room. Some form of mediation is required.

i) Automatic Abstraction: When combining component models to arrive at an evaluatable full-system model, there are likely to be mismatches in the abstraction levels of the different components. When evaluating such heterogeneous systems, automatic abstraction techniques can help in creating a finite model that can be checked using formal verification techniques

such as [32]. Commonly, some of the component models require costly computation as they can be used for evaluating many different properties. In certain cases, for example during design-space exploration, one is only interested in a subset of these properties. Model abstraction, surrogate modelling, or meta-modelling (not to be confused with the language engineering term) are techniques to derive a simplified model from a more complex model while trying to maintain fidelity with respect to a subset of the properties.

i) Validity Frames: To rapidly create virtual prototypes of a system, designers often use components from model libraries or of-the-shelf models that are provided by the supplier of the component. This virtual systems composition mimics its physical-world assembly counter-part. The topic of physical component reuse for system design is well known in engineering. Component catalogues shows all the properties and usage constraints of each component, enabling the engineer to make an informed decision. Reuse of models is treacherous however. During the act of modelling, the modeller makes a conscious choice about which properties are taken into account and which properties or phenomena are neglected. This depends on the modellers intent and what is of interest at that point in the design process. Reuse of a model thus not only depends on the model itself but also on the specific intents that the modeller had during the design process. Zeigler identified this problem in [33]. He introduced the Experimental Frame to model the context in which a model can be observed and experimented with. The experimental frame in Zeigler's book is further defined using three main components. (a) The generator describes the allowable inputs to the model, (b) The transducer to processes the output coming from the generator and the model, e.g. integrates the output or takes the mean. (c) The acceptor accepts or rejects the model. Note that a model can have multiple experimental frames and a frame can be valid for multiple models. Several authors have discussed the experimental frame and its relation to the M&S process, e.g. [34]. The concept of the experimental frame is however hard to implement, use as a contract for model selection, validate or calibrate models. In [35], the authors extended the experimental frames to the more general concept of purpose-drive frames where the validity and calibration frame are just two frames that specify the information needed for meaningful reuse of models and reproducibility of calibration and validation. Reasoning with these frames is still an open area of research. Furthermore, reusable models libraries have to be constructed taking these frames into account.

k) Contracts: To allow for concurrent design (co-design) of different parts of a system, interfaces and overlaps in the semantic domain such as resource consumption, need to be negotiated. Contract-based approaches in the context of CPS allow a formalisation of such co-design using assume and guarantee contracts. Under the assumed conditions, the component promises to fulfill the guaranteed properties. Benveniste et al. describe the mathematical foundations of refinement, composition and conjunction operators on contracts [36]. Vanherpen et al. extend the contract approach with upper ontologies (see earlier) to meaningfully reason about the content of contracts between different domains [37]. Examples of contracts between control engineers and deployment engineers can be found in [38]. Contracts are desirable as they simplify integration and verification. They might however introduce too much rigidity in the development process as they need to be formalised and negotiated. A challenge is thus to employ contracts without compromising agility.

l) Process Optimisation: To allow for shorter time-tomarket, processes should be optimised, for example performing independent tasks concurrently. This optimisation should take (in)consistency and the trade-off between value and optimised product into account. Queueing Network process performance can help to create optimal processes, by means of process simulation, similar to [39]. The calibration of these models requires historical data.

m) Incremental Safety: A way to allow for a more agile process with safety support is the incremental creation of the safety goals and corresponding cases. Incrementally creating this should deliver evidence after each sprint that the new safety goals are verified, Furthermore, it should also provide evidence that the safety cases corresponding to previously selected safety goals are still valid. Kokaly et al. provide a model management approach to allow for such reuse of assurance cases when a system evolves [40].

n) Traceability: Traceability keeps track of relationships between all the different artifacts, including requirements, design, tests, etc. during system development. Traceability is needed to create the assurance cases needed for safety and other dependability standards. It also gives the end-user feedback within the view/formalism he/she is familiar with. Trace management becomes increasingly important as the system design becomes more and more iterative with models at different levels of abstraction. Techniques such as impact analysis depend on the availability of traces.

V. CONCLUSIONS

In this position paper we analysed some of the different challenges related to reducing the risk of failure in the design process of Cyber-Physical Systems: changing and unknown requirements, dependability of the system, sub-optimal timeto-market, heterogeneity and intellectual property. Drawing inspiration from successful Agile approaches in software engineering, we proposed a scala of scientific solutions and, in turn, their related challenges.

REFERENCES

- [1] Pete Deemer. Gabrielle Benefield, Craig Larman, and Bas Vodde. The scrum primer v.1.2. Available at: http://goodagile.com/scrumprimer/scrumprimer.pdf [Accessed May 20171, 2010.
- [2] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. The agile manifesto. 2001.
- [3] Pedro Serrador and Jeffrey K Pinto. Does agile work? a quantitative analysis of agile project success. *International Journal of Project Management*, 33(5):1040–1051, 2015.
- [4] Hzael Woodcock. The agile manifesto reworked for systems engineering. Technical report, IBM, 2013.

- [5] Bruce Powel Douglass. Agile Systems Engineering. Morgan Kaufmann, 2015.
- [6] Thorsten P. Klein and Gunther Reinhart. Approaches for Integration of Agile Procedures into Mechatronic Engineering of Manufacturing Systems, pages 225–230. Springer International Publishing, 2014.
- [7] Bettina Buchel. How to build an agile beast. Technical report, IMD, 2017.
- [8] Robert Carlson and Richard Turner. Review of agile case studies for applicability to aircraft systems integration. *Procedia Computer Science*, 16:469–474, 2013.
- [9] Thórdís Reynisdóttir. Scrum in mechanical product development: Case study of a mechanical product development team using scrum. Technical report, Chalmers University of Technology, Sweden, 2013.
- [10] Edward A Lee. Cyber physical systems: Design challenges. In 2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pages 363–369. IEEE, 2008.
- [11] Lynne P Cooper. A research agenda to reduce risk in new product development through knowledge management: a practitioner perspective. *Journal of Engineering and Technology Management*, 20(1–2):117 – 140, 2003.
- [12] Benjamin D Lee and Christiaan JJ Paredis. Accounting for the duration of analyses in design process decisions. SAE International Journal of Materials and Manufacturing, 3(2010-01-0908):512–522, 2010.
- [13] Estelle Frey, Egon Ostrosi, Lionel Roucoules, and Samuel Gomes. Multi-domain product modelling: from requirements to CAD and simulation tools. In *International Conference on Engineering Design*, *ICED'09*, 2009.
- [14] Ahsan Qamar, Christiaan JJ Paredis, Jan Wikander, and Carl During. Dependency modeling and model management in mechatronic design. *Journal of Computing and Information Science in Engineering*, 12(4):041009, 2012.
- [15] Magnus Persson, Martin Törngren, Ahsan Qamar, Jonas Westman, Matthias Biehl, Stavros Tripakis, Hans Vangheluwe, and Joachim Denil. A characterization of integrated multi-view modeling in the context of embedded and cyber-physical systems. In *Proceedings of the Eleventh ACM International Conference on Embedded Software*, page 10. IEEE Press, 2013.
- [16] Ken Vanherpen, Joachim Denil, István Dávid, Paul De Meulenaere, Pieter J Mosterman, Martin Torngren, Ahsan Qamar, and Hans Vangheluwe. Ontological reasoning for consistency in the design of cyber-physical systems. In *Cyber-Physical Production Systems*, *Workshop on*, pages 1–8. IEEE, 2016.
- [17] N. Pedersen, K. Lausdahl, E. Sanchez, P. Larsen, and J. Madsen. Distributed co-simulation of embedded control software with exhaust gas recirculation water handling system using into-cps. Technical report, under review, 2017.
- [18] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art. CoRR, abs/1702.00686, 2017.
- [19] Jens Bastian, Christop Clauß, Susann Wolf, and Peter Schneider. Master for co-simulation using fmi. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical University; Dresden; Germany*, number 63, pages 115–120. Linköping University Electronic Press, 2011.
- [20] Bert Van Acker, Joachim Denil, Hans Vangheluwe, and Paul De Meulenaere. Generation of an optimised master algorithm for fmi cosimulation. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, DEVS '15, pages 205–212. SCS, 2015.
- [21] David Broman, Christopher Brooks, Lev Greenberg, Edward A Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of fmus for co-simulation. In *Proc.* 11th ACM International Conference on Embedded Software, page 2. IEEE, 2013.
- [22] Joachim Denil, Bart Meyers, Paul De Meulenaere, and Hans Vangheluwe. Explicit semantic adaptation of hybrid formalisms for fmi co-simulation. In Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, DEVS '15, pages 99–106. SCS, 2015.
- [23] Luiza Gheorghe, Faouzi Bouchhima, Gabriela Nicolescu, and Hanifa Boucheneb. A formalization of global simulation models for continuous/discrete systems. In *Proceedings of the 2007 Summer Computer Simulation Conference*, pages 559–566. SCS, 2007.

- [24] Pieter J. Mosterman and Hans Vangheluwe. Guest editorial: Special issue on computer automated multi-paradigm modeling. ACM Trans. Model. Comput. Simul., 12(4):249–255, October 2002.
- [25] Hans Vangheluwe. Devs as a common denominator for multi-formalism hybrid systems modelling. In IEEE International Symposium on Computer-Aided Control System Design (ed. Andras Varga), IEEE Computer Society Press, Anchorage, Alaska, sept. 2000, pages 129–134, 2000.
- [26] Michais Famelis, Rick Salay, and Marsha Chechik. Partial models: Towards modeling and reasoning with uncertainty. In *Software Engineering* (*ICSE*), 2012 34th International Conference on, pages 573–583. IEEE, 2012.
- [27] Rick Salay, Michalis Famelis, and Marsha Chechik. Language Independent Refinement Using Partial Modeling, pages 224–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [28] Michalis Famelis, Rick Salay, Alessio Di Sandro, and Marsha Chechik. *Transformation of Models Containing Uncertainty*, pages 673–689. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [29] Alexander Egyed. Instant consistency checking for the uml. In Proceedings of the 28th international conference on Software engineering, pages 381–390. ACM, 2006.
- [30] Ahsan Qamar. Model and dependency management in mechatronic design. PhD thesis, KTH, 2013.
- [31] Sebastian JI Herzig and Christiaan JJ Paredis. Bayesian reasoning over models. In *MoDeVVa@MoDELS*, pages 69–78, 2014.
- [32] Robert A Thacker, Kevin R Jones, Chris J Myers, and Hao Zheng. Automatic abstraction for verification of cyber-physical systems. In *International Conference on Cyber-Physical Systems*, pages 12–21. ACM, 2010.
- [33] B. P. Zeigler, H. Praehofer, and T. G. Kim. Theory of Modelling and Simulation, Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, 2 edition, 2000.
- [34] Mamadou K. Traoré and Alexandre Muzy. Capturing the dual relationship between simulation models and their context. *Simulation Modelling Practice and Theory*, 14(2):126–142, 2006.
- [35] Joachim Denil, Stefan Klikovits, Pieter Mosterman, Antonio Vallecillo, and Hans Vangheluwe. The experimental model and the validity frame in M&S. In *Proceedings of the Symposium on Theory of Modelling and Simulation*, 2017.
- [36] Albert Benveniste, Benoit Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas Henzinger, and Kim G. Larsen. Contracts for System Design. Research Report RR-8147, INRIA, November 2012.
- [37] Ken Vanherpen, Joachim Denil, Paul De Meulenaere, and Hans Vangheluwe. Ontological reasoning as an enabler of contract-based codesign. In *International Workshop on Design, Modeling, and Evaluation* of Cyber Physical Systems, pages 101–115. Springer, 2016.
- [38] Patricia Derler, Edward A Lee, Martin Törngren, and Stavros Tripakis. Cyber-physical system design contracts. In *Cyber-Physical Systems* (ICCPS), International Conference on, pages 109–118. IEEE, 2013.
- [39] István Dávid, Joachim Denil, Klaas Gadeyne, and Hans Vangheluwe. Engineering process transformation to manage (in)consistency. In *COMMitMDE*), pages 7–16, 2016.
- [40] Sahar Kokaly, Rick Salay, Valentin Cassano, Tom Maibaum, and Marsha Chechik. A Model Management Approach for Assurance Case Reuse due to System Evolution. In *Proc. of MODELS'16*, 2016.