Characterizing behavioral modeling in Systems and Safety Model-Based Engineerings and their overlap for consistency checking

Stephen Creff

IRT SystemX, Palaiseau, France. E-mail: stephen.creff@irt-systemx.fr

Michel Batteux

IRT SystemX, Palaiseau, France. E-mail: michel.batteux@irt-systemx.fr

Due to nowadays systems complexity, the modeling of a system is multi-concerns and multi-viewpoints in its very essence. Systems Engineering and Safety Assessment are two engineering domains that currently follow model-based approaches to conceive the system at the same level of abstraction. The overall consistency between the different models contributing to the system design is a key element of the realization. Each concern must align with common assumptions. Models are made of two kinds of constructs: structural and behavioral ones. The structural consistency challenge between Model-based Systems Engineering and Safety Assessment has already been specifically addressed in a generic way. What about behavioral consistency now? Before considering any behavioral consistency checking, identifying the behavioral modeling characteristics of the behavioral representations in each domain and potential overlap must be performed. Therefore, in this article we propose to characterize the behavioral modeling in systems and safety model-based engineerings and to provide some keys to identify their overlap for a forthcoming consistency checking.

Keywords: Behavioral modeling, Model-Based Systems Engineering, MBSE, Model-Based Safety Assessment, MBSA, Overlap between models, Consistency checking.

1. Introduction

Due to nowadays systems complexity, the modeling of a system is multi-concerns and multi-viewpoints in its very essence ISO42010 (2011).

Each concern must align with common assumptions to avoid design iterations and late reworks. In order to offer an adequate tool support and to avoid making the coherence work rely on people-based documentary reviews, a work must be done on the design artifacts: the models. As reminded by Batteux et al. (2019), the integration of models coming from various engineering disciplines, such as system architecture, multi-physics simulation, automatic code generation as well as safety and performance analyses, is one of today's industrial challenges. Usually, in practice, the different disciplines (i.e., concerns) use their own tools, and therefore define their own models.

Models are made of two kinds of constructs: structural and behavioral ones. The notions of structural and behavioral consistency, cf. Van Der Straeten et al. (2003), then describe consistency among either the structural (e.g., missing components/parts) or the behavioral aspects of the models (incompatible behaviors). The structural consistency challenge between Model-Based Systems Engineering (MBSE) and Model-Based Safety Assessment (MBSA), two disciplines at opposite ends of the spectrum, has already been specifically addressed in a generic way by Batteux et al. (2019b). What about behavioral consistency now?

Before considering any verification of behavioral consistency between MBSE and MBSA (and thus defining a viable approach), it is necessary to identify the characteristics of behavioral modeling in each domain, and their overlaps. This is what we propose to do in this paper, by characterizing the models, illustrating them with different published case-studies, and thus providing some keys to identify their overlap, a prerequisite to any verification of behavioral consistency.

The remainder of the article is organized as follows. Section 2 re-contextualizes the work in

Proceedings of the 32nd European Safety and Reliability Conference (ESREL 2022).
Edited by Maria Chiara Leva, Edoardo Patelli, Luca Podofillini and Simon Wilson
doi: 10.3850/978-981-18-5183-4_R08-02-385-cd

the field of inconsistency management to highlight the importance of overlaps identification. Then, Section 3 characterizes behavioral modeling in both SE and SA domains. Next, Section 4 lists some difficulties to align behavioral models from MBSE and MBSA. After that, Section 5 provides some characteristics towards the overlap identification. Finally, the last section draws our conclusion and narrows down possible future works.

2. Inconsistency management and overlaps

Because of these different views, assumptions, and concerns, all of them being interrelated as they are related to the same system, overlaps may arise, cf. Muskens et al. (2005). As noticed by Spanoudakis and Zisman (2001), ultimately, the presence of such interrelations introduces the potential for inconsistencies.

Various domain and communities have found interest in (in-) consistency management, e.g. in requirement engineering with Van Lamsweerde et al. (1997) (late nineties), in software engineering with Nuseibeh et al. (2000) (early twenties), or in systems engineering with Herzig and Paredis (2014) (quite recently). An inconsistency is described by Spanoudakis and Zisman (2001) as "a state in which two or more overlapping elements of different [. . .] models make assertions about aspects of the system they describe which are not jointly satisfiable". The authors describe the inconsistency management process as consisting of a sequence of six activities, the first, important and non-trivial one, being the detection of overlaps between models. Going back to our specific problem, Fig. 1 represents in an abstract way the overlap between MBSE and MBSA.

3. Characterizing behavioral models in MBSE and MBSA

Both disciplines follow model-based approaches: e.g. SysML (2019) or Arcadia cf. Roques (2016) for SE; and e.g. AltaRica -Point and Rauzy (1999), Safety Analysis Modeling Language (SAML) - Gudemann and Ortmeier (2010), Figaro - Bouissou et al. (1991), etc. for SA.

The overall consistency between all models (thus including behavioral ones) is essential. The preliminary question is: What kind of behavior is generally modeled in MBSE and MBSA?

Models considered in MBSE and MBSA can be mainly characterized by Discrete Event Systems (DESs) modeling (cf. Fig 2), which are, as remained by Cassandras and Lafortune (2008), defined by two characteristics: i) the state spaces are discrete and potentially infinite, and ii) the dynamics are event driven as opposed to time driven. The set of events, is also a discrete set; the definition assumes that this set has finite cardinality.

3.1. *Usual behavioral MBSE models*

Model-Based Systems Engineering is seen by the Incose (2015) as "the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities". It relies on frameworks and notations like e.g. the standardized SysML (2019) or the dedicated Capella language. They allow to capture the requirements in the analysis phase, and to capture the architecture and design of a solution, in the design and implementation phase.

The behavior captured covers: what the system has to do to meet the requirements; the transformations of inputs to outputs (functional/activity models), the state/mode-based behavioral differences (state models), the responses to incoming requests for services (message models). Behavior diagrams in SysML (2019) and derived notations are: activity diagram, sequence diagram, state machine diagram and use-case diagram. These kinds of diagrams, with semantics variation points, are typically used in MBSE.

In practice, the expression of behavior is currently done at the margin in MBSE. Indeed, although formalisms are proposed, there are still obstacles to their wider adoption in the industry, like formal semantics definition (see Section 4.3), and tooling support. In fact, few Systems Engineering tools provides behavioral simulation features (e.g. Cameo or Sparx EA).

As previously said, modeled are mainly deterministic discrete event oriented, leaving aside algebraic equations, differential equations, continuous domain simulation etc.

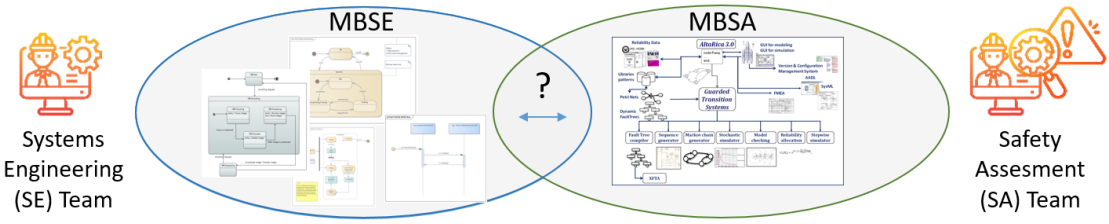

Fig. 1. MBSE – MBSA models overlap overview.

3.2. *MBSA behavioral models*

The safety analysis point of view focuses on degradations and failures of components, leading to dysfunctions of the system, meaning failure modes of functions of the system. Such degradations and failures are defined, more precisely their behaviors are abstracted, by their occurrences which are event based and stochastic.

Two kinds of models are considered. On the one hand, combinatorial (Boolean) models as fault trees, event trees, reliability block diagrams, etc., see for instance Rausand and Høyland (2004) for a reference book. On the other hand, states/events models such as Markov chains or stochastic Petri nets - Marsan et al. (1998). Even if the expressive power of the states/events models are higher than combinatorial ones, they are however not sufficient in themselves to reduce the distance between system specifications and safety models. The interested reader can see Batteux et al. (2019a) for explanations about this distinction between combinatorial and states/events models, their advantages and drawbacks.

The so-called MBSA approach models the system at higher level so to reduce the distance between systems specifications and models, without increasing the complexity of calculations. The AltaRica 3.0 modeling language, for instance, implements this MBSA approach -Batteux et al. (2019a). AltaRica 3.0 is an event based and object-oriented modeling language. Its mathematical framework, describing the behavioral part of the language, is based on Guarded Transition Systems (GTS) - Batteux et al. (2017), a specific implementation of stochastic discrete event systems. For the rest of the paper, we focus on this language for the models of SA.

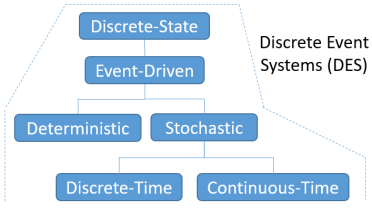

Fig. 2. Discrete Event Systems classification – cf. Cassandras and Lafortune (2008)

Summary: MBSE models are deterministic discrete event oriented, while MBSA models are stochastic ones.

4. Some difficulties to align MBSE and MBSA

As noted earlier, a common practice for managing its overwhelming complexity is to approach the study of the system from different viewpoints, perspectives and concerns, cf. ISO42010 (2011). These concerns are carried by different engineering departments/teams, and obviously, these viewpoints do not exist in isolation.

4.1. *MBSE and MBSA processes*

These processes generally evolve in parallel, with specific synchronization activities (reviews, models consistency checking, ...), respecting some independence between the concerns. Some fields of application, for example aeronautics, emphasize this point: the ARP4754A (*Guidelines For Development Of Civil Aircraft and Systems*, with an emphasis on safety aspects), defined the independence as: i) A concept that minimizes the likelihood of common mode errors and cascade failures between aircraft/system functions or items, ii) Separation of responsibilities that assures the

accomplishment of objective evaluation e.g. validation activities not performed solely by the developer of the requirement of a system or item.

These definitions include two concerns: the first one is related to the system and more specifically the architecture (thus models) of the system, the second one implies organizational constraints on the designing and development process. Therefore, process, methods and tools may be synchronized relatively lately.

Generic Difficulty: potential divergence and late synchronization.

4.2. *Systems criticality towards Safety*

From the safety point of view, there are two main kinds of systems:

A. Systems for which the main function is to provide a safety mechanism of a more general system. For instance, the cooling system of a nuclear power plant, the high integrity pressure production system (HIPPS) of a chemical plant or oil refinery, etc.
B. Other systems, the more general ones, meaning that the main function is not to provide a safety mechanism of a more general system. For instance, a nuclear power plant, a train, etc.

The design process of both two kinds of systems can be the same and follow the ones involved in SE. Nevertheless, for both SE and SA point of views, they consider the system with the same vision, in terms of level, environment, functions, components, behaviors, etc. for case A., and with two different visions for case B.

Generic Difficulty: dependent on the kind of systems and on the modeling choices.

4.3. *Issue with systems behavioral models*

In order to define overlaps between behavioral models in a meaningful way, it is necessary to assume that these models have a formal semantics (i.e. well-defined, explicit, and shared), which leads to the following issue. There is a diversity of notations, formalisms and tools around modeling in MBSE. The lack of semantics in the definition of modeling languages is highlighted by the software modeling community which revealed semantic variation points, e.g. , in the basis of SysML - Cuccuru et al. (2007), Cengarle et al. (2009), Latombe et al. (2015).

Popular behavioral specification languages such as Statecharts Harel and Naamad (1996) - or Message Sequence Charts (International Telecommunication Union - ITU - Recommendation Z.120), when incorporated into more complete notations such as SysML (2019) or Capella, have imprecise semantics - cf. Chauvel and Jézéquel (2005). The semantic variation points concern mainly 3 aspects: i) Time management (synchronous vs. asynchronous), ii) The event selection policy (Events can be internal/external or discrete/continuous, event pool: queue, stack, mail box ...), iii) The transition selection policy (source states, with transitions originating from deeper states having higher priority).

In contrast, other more formal behavioral specification languages – like the process algebras (Calculus of Communicating Systems - CCS), Milner (1989), and Communicating Sequential Processes (CSP), Hoare (1978) – provide a thoroughly defined semantics, but have not gained, and probably will never have, the same level of acceptance.

Specific Difficulty: lack of formalism in MBSE behavioral models.

4.4. *Synthesis*

To summarize, the following 3 difficulties are observed to align MBSE and MBSA and to detect inconsistencies. Some are generic, i.e. valid for structural and behavioral concerns, some are specific to the behavioral case. The generics are:

- Divergence during the realization and potentially late synchronization due to the process;
- Variability of the resulting models (scopes and details considered in MBSE and MBSA), depending on the kind of systems considered.

The specific is related to a MBSE lack of formal-

ism in the definition of behavioral models.

5. Some characteristics towards the overlap identification

This section describes some observations made on characteristics about the models "classically" produced in MBSE and MBSA, in order to provide a guideline to identify overlaps in a given project, or for a given situation. To avoid potential modeling biases introduced by our analysis, case studies published by other authors are selected to illustrate our remarks.

5.1. *MBSE and MBSA concerns*

SE and SA are two engineering domains that conceive the system at the same level of abstraction. Nevertheless, the concerns are not the same for these two stakeholders, and they do not consider the system of interest in the same way and with the same objectives. On the one hand, SE has interests in how the system works, with what means and for what missions: *the functional aspects*. Whereas SA, on the other hand, has interests in the *dysfunctions* of the system, their causes and consequences, and their possible mitigations (SA models reflect some functional aspects of the system).

Besides, depending on the kind of system (A. or B. in Section 4.2), the SE may include measures against dysfunctional consequences, and therefore include additional functions and components to check the validity of some inputs/outputs, some safety functions (voters…). For example, considering a function on the SE side with two inputs i_1 and i_2 and an output o; these elements can be defined by two properties, their value (within a domain) and their status (validity: ok/nok). The propagation of the validity status is modeled on the SE side. Table 1 sums up the considerations encountered in the modeling of the two types of systems A. and B.

SE and SA concerns are clearly different, but overlaps exist in their models between functional considerations, and between i) dysfunctional (SA) and ii) dysfunctional mitigation measures (SE).

Generic Difficulty: Distance between concerns.

Table 1. Functional and dysfunctional considerations in MBSE and MBSA models.

	MBSE	MBSA
Sys A.	Functional	(Functional) + Dysfunctional
Sys B.	Functional + Dysfunctional mitigation measures	(Functional) + Dysfunctional

Brackets represent options.

5.2. *Propagation of values and propagation of failures*

As said in the previous section, the SE models functional aspects of the system. Behavioral models used for this purpose describes functional chains (functional propagation), e.g. by mean of diagrams like activity and sequence diagrams in SysML, or functional data-flows and scenarios in Capella. On the MBSE side, a clear distinction is made between functional and physical levels, (which is not the case on the MBSA one). On the MBSA side, the propagation of failures depends on the physical description level (components).

Therefore, the sequencing of functions is not directly aligned with a sequencing of propagation of failures but has to consider the allocations to components. Structural consistency is a pre-requisite. Some diagrams may combine the activity flow and the allocation, e.g. Fig. 3 an example of an air compressor (cf. Friedenthal et al. (2008) for details on the case-study).

As a consequence, the overlap between SE functional propagation and SA dysfunctional one

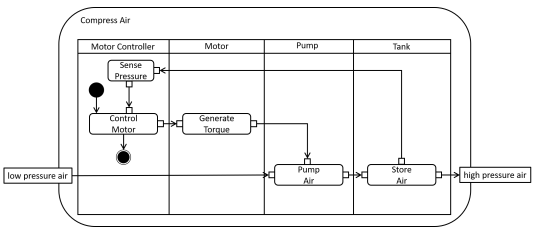

Fig. 3. SysML functions and allocations diagram – Air compressor example, source Friedenthal et al. (2008)

can only be made in regard with the allocation onto the physical architecture.

Specific Difficulty: SA models aggregate functional and physical levels; functional and dysfunctional propagations are not directly aligned.

5.3. *Scopes of the models*

The scopes of the SE and SA models are generally different. On the one side, SE models the system (functionally), its environment (focusing only on interactions), and provides details about the sub-systems (functionally). On the other side, the SA models the system, its environment by focusing on what is important for failures (not only direct interactions, but cascading failure propagations), and generally does not provide details on the components, only their failure in terms of events.

Modes and states are generally more developed on the SE side, covering different operational phases (from pre to post mission phases, initialization to maintenance and retirement). However, some modes and states may overlap between SE and SA, as illustrated by Fig. 4 on a case-study of a circuit breaker (details in Mortada et al. (2014)). Note that the MBSE defines a prescriptive model of what is considered and specified.

Let us consider a state-machine of a fuel pump, considered as a sub-system in a modeling, the details would be provided by the SE as illustrated in Fig. 5 (case-study of a fuel pump, details in Zdanis and Cloutier (2007)), but would only be considered on the SA side as its composing components that can fail.

Comparing state-machines, when their scope overlaps, under the hypothesis that the explicit states are the same or could be mapped at a given abstraction, deterministic event would have to be compared to stochastic ones.

Generic Difficulty: alignment of models with different scopes and abstractions (in different formalisms).

5.4. *Deterministic and Stochastic events*

There is a major difference between discrete event systems on the SE and SA sides. This difference is concerned with the kind of delays associated to events. A delay of an event defines the way, and the time instant, the event is fired. For instance, at a time instant t and for a delay done by a deterministic law (Dirac(d) with $d \geq 0$), it is scheduled to be fired at time instant $t+d$, if it stays enabled until $t+d$. For a delay done by a stochastic law (exponential(λ), with $\lambda > 0$), it is scheduled to be fired at a time instant in the interval $[t; +\infty[$ meaning an infinite number of dates.

On the one hand for the SE side, delays are deterministic. It (normally) means that there is a finite set of executions of the model. The differences occur with concurrent enabled transitions that can be fired at the same time instant. On the other hand, for the SA side, delays are deterministic or stochastic (mostly stochastic). It means that SA models provide an infinite number of executions with different order of fired events.

Thus, comparing such behaviors means comparing, on the one hand, a finite (and small)

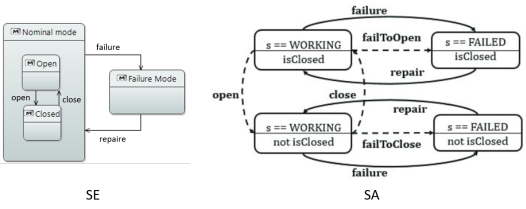

Fig. 4. Modes and states of a circuit breaker – example and SA source taken from Mortada et al. (2014)

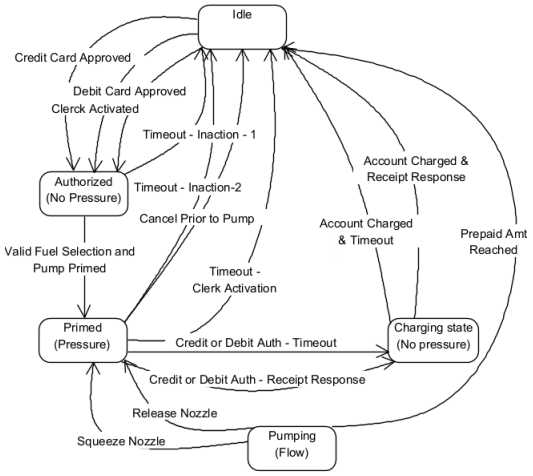

Fig. 5. State-machine of a sub-system, example of a Fuel pump taken, source Zdanis and Cloutier (2007)

number of executions, to, on the other hand, an infinite number of executions. Furthermore, with different order of fired events. It is impractical. Some recent works defining abstract executions of stochastic discrete event systems Batteux et al. (2022), could be a way to compare behaviors, but it needs additional research works, even in the case of comparing two version of a SA model.

Specific Difficulty: Comparing Deterministic and stochastic DESs.

5.5. *Synthesis*

To summarize, the following difficulties are observed in this section to align SE and SA models and detect inconsistencies. The generics are:

- The distance between SE and SA concerns that analyses the system in terms of functional and dysfunctional aspects;
- SE and SA models have different scopes, abstractions, and formalisms.

The specifics are:

- Functional and dysfunctional propagations are not directly aligned;
- Deterministic and stochastic models are hard to compare.

As previously said, SE and SA models depend on many factors, including the system under study. Genericity about identifying overlaps between MBSE and MBSA models is not possible. Table 2 provides initial keys to characterize the overlap between MBSE and MBSA models.

Table 2. Initial keys to identify overlaps between MBSE and MBSA behavioral models.

What	MBSE	MBSA
Nature	Functional + (Dysfunctional mitigation measures)	(Functional) + Dysfunctional
Scope	System + Sub-systems with details + (Environment)	System + Sub-systems without details + Environment
Behavioral phenomenon	Deterministic Discrete Events	Stochastic Discrete Events
Formalisms	So Many: Statecharts, I/O Automata, SyncCharts, EFFBD, petri nets, ...	Altarica Data flow, and guarded transition systems

Brackets represent options.

6. Discussions and conclusion

This paper proposes to characterize the behavioral modeling in systems and safety model-based engineerings and to provide some keys to identify their overlap for a future consistency checking. It highlights 4 difficulties that are not specific to behavioral modeling, and 3 that are. It provides a first guideline for identifying overlap between MBSE and MBSA behavioral models, in order to compare comparable elements.

A key observation is that the identification of overlap is not generic as it depends on the system under study and how the models are made. Aligning behavioral models between SE and SA is more difficult than aligning other engineering disciplines (e.g. simulation) with the SE, because the distance between the concepts being manipulated is greater (nature, phenomena, formalisms).

A basis on which behavioral models are built is the structural part of the models. The resolution of structural consistency seems to be less specific and therefore efforts seem more beneficial to address this aspect globally. Some dedicated behavioral consistency check can additionally be performed for given scenarios that may compare specific observers on the models, without trying to be generic. Future work will focus on the alignment of some MBSE and MBSA formalisms and extend the case-study from Batteux et al. (2019b) with behavioral models.

Acknowledgement

This research work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute.

References

Batteux, M., J.-Y. Choley, F. Mhenni, T. Prosvirnova, and A. Rauzy (2019). Synchronization of system

architecture and safety models: a proof of concept. In *2019 ISSE Proceedings*, pp. 1–8.

Batteux, M., T. Prosvirnova, and A. Rauzy (2017). AltaRica 3.0 assertions: the why and the wherefore. *Journal of Risk and Reliability 231*(6), 691–700.

Batteux, M., T. Prosvirnova, and A. Rauzy (2019a). AltaRica 3.0 in 10 modeling patterns. *International Journal of Critical Computer-Based Systems 9*(1–2), 133–165.

Batteux, M., T. Prosvirnova, and A. Rauzy (2019b). Model synchronization: A formal framework for the management of heterogeneous models. In *Model-Based Safety and Assessment*, pp. 157–172. Springer.

Batteux, M., T. Prosvirnova, and A. Rauzy (2022). Abstract executions of stochastic discrete event systems. *International Journal of Critical Computer-Based Systems 10*(3), 202–226.

Bouissou, M., H. Bouhadana, M. Bannelier, and N. Villatte (1991). Knowledge modelling and reliability processing: Presentation of the figaro language and associated tools. *IFAC Proceedings Volumes 24*(13), 69–75.

Cassandras, C. G. and S. Lafortune (Eds.) (2008). *Introduction to Discrete Event Systems*. Springer US.

Cengarle, M. V., H. Grönniger, and B. Rumpe (2009). Variability within modeling language definitions. In *Proceedings of MODELS*, Berlin, Heidelberg, pp. 670–684. Springer-Verlag.

Chauvel, F. and J.-M. Jézéquel (2005). Code generation from uml models with semantic variation points. In L. Briand and C. Williams (Eds.), *MODELS*, Berlin, Heidelberg, pp. 54–68. Springer Berlin Heidelberg.

Cuccuru, A., C. Mraidha, F. Terrier, and S. Gérard (2007). Templatable metamodels for semantic variation points. In D. H. Akehurst, R. Vogel, and R. F. Paige (Eds.), *Model Driven Architecture- Foundations and Applications*, Berlin, Heidelberg, pp. 68–82. Springer Berlin Heidelberg.

Friedenthal, S., A. Moore, and R. Steiner (2008). *A Practical Guide to SysML: Systems Modeling Language*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Gudemann, M. and F. Ortmeier (2010). A framework for qualitative and quantitative formal model-based safety analysis. In *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*, pp. 132–141.

Harel, D. and A. Naamad (1996, 10). The statemate semantics of statecharts. *ACM Trans. Softw. Eng. Methodol. 5*, 293–333.

Herzig, S. J. and C. J. Paredis (2014). A conceptual basis for inconsistency management in model-based systems engineering. *Procedia CIRP 21*, 52 – 57. 24th CIRP Design Conference.

Hoare, C. A. R. (1978, aug). Communicating sequential processes. *Commun. ACM 21*(8), 666–677.

Incose (2015). *Systems Engineering Handbook* (Fourth Edition ed.).

ISO42010 (2011). ISO/IEC/IEEE 42010:2011, Systems and software engineering — Architecture description. Document Number ICS/35/35.080.

Latombe, F., X. Crégut, J. Deantoni, M. Pantel, and B. Combemale (2015). Coping with Semantic Variation Points in Domain-Specific Modeling Languages. In *EXE'15, co-located with MODELS'15*, Ottawa, Canada. CEUR.

Marsan, M. A., G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis (1998, aug). Modelling with generalized stochastic petri nets. *SIGMETRICS Perform. Eval. Rev. 26*(2), 2.

Milner, R. (1989). *Communication and Concurrency*. USA: Prentice-Hall, Inc.

Mortada, H., T. Prosvirnova, and A. Rauzy (2014). Safety assessment of an electrical system with altarica 3.0. In F. Ortmeier and A. Rauzy (Eds.), *Model-Based Safety and Assessment*, Cham, pp. 181–194. Springer International Publishing.

Muskens, J., R. J. Bril, and M. R. V. Chaudron (2005). Generalizing consistency checking between software views. In *WICSA'05*, pp. 169–180.

Nuseibeh, B., S. Easterbrook, and A. Russo (2000, April). Leveraging inconsistency in software development. *Computer 33*(4), 24–29.

Point, G. and A. Rauzy (1999). Altarica : Constraint automata as a description language. *Journal européen des systèmes automatisés*.

Rausand, M. and A. Høyland (2004, January). *System Reliability Theory: Models, Statistical Methods, and Applications, 2nd Edition*. Hoboken, New Jersey, USA: Wiley-Blackwell.

Roques, P. (2016, January). MBSE with the ARCADIA Method and the Capella Tool. In *ERTS 2016*, Toulouse, France.

Spanoudakis, G. and A. Zisman (2001, dec). Inconsistency Management In Software Engineering: Survey And Open Research Issues.

SysML (2019). System Modeling Language Specification, Version 1.6. OMG Document Number formal/19-11-01.

Van Der Straeten, R., T. Mens, J. Simmonds, and V. Jonckers (2003). Using description logic to maintain consistency between uml models. In P. Stevens, J. Whittle, and G. Booch (Eds.), *Proceedings of UML 2003*, Berlin, Heidelberg, pp. 326–340. Springer Berlin Heidelberg.

Van Lamsweerde, A., E. Letier, and C. Ponsard (1997, 06). Leaving inconsistency.

Zdanis, L. and R. Cloutier (2007). The use of behavioral diagrams in sysml. In *IEEE Long Island Systems, Applications and Technology Conference*, pp. 1–1.