Software and Systems Modeling (2021) 20:897–916
https://doi.org/10.1007/s10270-020-00834-1

REGULAR PAPER

A systematic literature review of cross-domain model consistency checking by model management tools

Weslley Torres[1] · **Mark G. J. van den Brand**[1] · **Alexander Serebrenik**[1]

Received: 24 September 2019 / Revised: 15 July 2020 / Accepted: 30 September 2020 / Published online: 21 October 2020

Abstract

Objective The goal of this study is to identify gaps and challenges related to cross-domain model management focusing on consistency checking. *Method* We conducted a systematic literature review. We used the keyword-based search on Google Scholar, and we identified 618 potentially relevant studies; after applying inclusion and exclusion criteria, 96 papers were selected for further analysis. *Results* The main findings/contributions are: (i) a list of available tools used to support model management; (ii) 40% of the tools can provide consistency checking on models of different domains and 25% on models of the same domain, and 35% do not provide any consistency checking; (iii) available strategies to keep the consistency between models of different domains are not mature enough; (iv) most of the tools that provide consistency checking on models of different domains can only capture up to two inconsistency types; (v) the main challenges associated with tools that manage models on different domains are related to *interoperability* between tools and the *consistency maintenance*. *Conclusion* The results presented in this study can be used to guide new research on maintaining the consistency between models of different domains. Example of further research is to investigate how to capture the *Behavioral* and *Refinement* inconsistency types. This study also indicates that the tools should be improved in order to address, for example, more kinds of consistency check.

Keywords Model management · Systems engineering · Model-based systems engineering

1 Introduction

Inconsistencies can cause catastrophic events: e.g., the NASA unmanned MARS Climate Orbiter [101] was destroyed in 1999 due to use of inconsistent metric units by design teams, and Airbus had 6 billion dollar loss in 2006 due to use of inconsistent specifications in different versions of design tools [114].

Inconsistencies can be found in several stages of the system development life cycle. In earlier stages, when engineers are eliciting requirements, they might misunderstand the stakeholders' needs. Thus, the stakeholders' needs might be modeled wrongly, resulting in a product that does not match their expectations. Another inconsistency can arise when the models (e.g., class diagram, activity diagram) are correct but the software developers misunderstand them, resulting in a source code that does not represent the design intention. The crucial point here is that the earlier the inconsistency is found, the less it will cost [64] to fix the inconsistency. In the previous examples, only one domain was involved, i.e., the software engineering domain. In these scenarios, identifying and managing inconsistencies is already difficult.

Furthermore, it is known that systems are becoming increasingly complex to develop, especially when these systems are heterogeneous and there is a need to combine models created by engineers from different expertise and different domains [6,7,40,125,144]. One example of such a complex system is a mechatronic component: to develop it, one might need to combine expertise from different engineering domains such as mechanics, electronics and software [124].

Formally, we say that models are from the same domain, if they are created by engineers from the same engineering discipline, e.g., software engineering or mechanical engineering. Models of the same domain can be created using

Communicated by Lionel Briand.

✉ Weslley Torres
w.silva.torres@tue.nl

Mark G. J. van den Brand
m.g.j.v.d.brand@tue.nl

Alexander Serebrenik
a.serebrenik@tue.nl

[1] Eindhoven University of Technology, Eindhoven, Netherlands

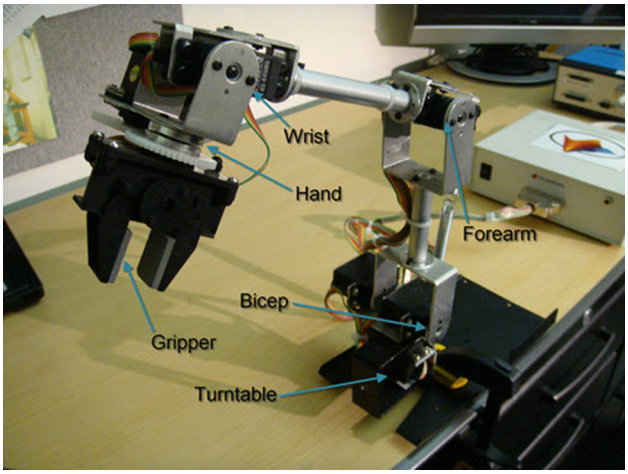


Fig. 1 Six degree-of-freedom robotic arm. Reproduced with permission of MathWorks. Copyright (2020) [46]

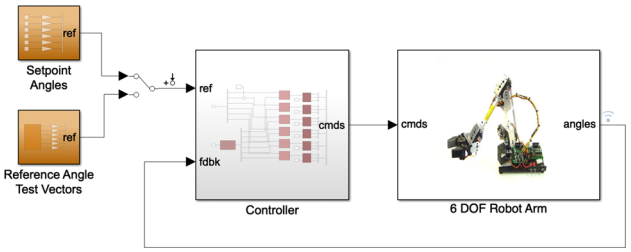


Fig. 2 A Simulink model of the electrical and mechanical components of the robotic arm. Reproduced with permission of MathWorks. Copyright (2020) [46]

different modeling tools: e.g., one UML model can be created using Lucidchart.[1] and another one using StarUML.[2] We say that models are from different domains if they are created by engineers from different engineering disciplines that might be using the same or different modeling tools. For example, engineers can use Simulink[3] to design both the electrical and mechanical components of a six degree-of-freedom robotic arm in Fig. 1. Models of electrical and mechanical components are shown in Fig. 2.

Due to the sheer complexity of modern systems and the presence of multiple authors, inconsistencies between the models might be inadvertently introduced, e.g., one model might assume the presence of a certain feature, while another one might assume its absence. This problem might be further amplified by differences in terminology used in different domains: e.g., for a software engineer, a feature is a functionality provided by the system, but for system engineer, a feature is an aspect of the system, like the color. This kind of misunderstanding can affect the consistency of the models. Therefore, the terms have to be well described in order to simplify the process of maintaining consistency between models.

Maintaining consistency between models is known to be a challenging task, especially because it is difficult to predict the effects of changes introduced in one model on other models [113]. While maintaining consistency between models is imperative [117], in practice, it can never be fully ensured [63], and the system engineer is responsible to define what has to be consistent and when. The process of managing these models can be expensive. Thus, we believe that the consistency should only be managed when the costs to maintain the consistency are lower than the costs that an eventual inconsistency can cause.

In order to understand industrial practices and academical approaches, aimed at checking and keeping consistency of models of different domains, we defined four Research Questions (RQ)s:

- RQ1: How do model life cycle management tools address consistency between models from different domains?
  - Motivation: Maintaining the consistency between models of different domains is a challenging task. Thus, we investigate how tools support this task.
  - Answer: We identify 80 tools, but the majority of them do not check consistency between models from different domains.
- RQ2: What inconsistency types are addressed by the model life cycle management tools?
  - Motivation: In order to be able to indicate gaps related to the inconsistencies types addressed by the tools, we investigate which inconsistencies types are captured or not.
  - Answer: The inconsistency types addressed by the tools are: *Behavioral, Information, Interaction, Interface, Refinement, and Requirement*. *Interface* is the most popular. *Behavioral* and *Refinement* are the least popular. We believe that these inconsistency types, previously identified in the research literature, are not addressed by the tools due to the complexity of capturing them. Surprisingly, these tools do not advertise that they can capture the *Name* inconsistency type. We conjecture that since *Name* inconsistency can be easily captured, and it is also an *Interface* inconsistency (but not vice versa), the tool builders prefer to advertise the later option.
- RQ3: Which strategies have been used to keep the consistency between models of different domains?

[1] https://www.lucidchart.com.

[2] http://staruml.io.

[3] https://www.mathworks.com/products/simulink.html.

  - Motivation: Identify the drawbacks of the technologies and approaches used to keep consistency between models of different domains.
  - Answer: The following strategies have been used to keep the consistency between models from different domains: Interoperability, Inconsistency Patterns, Modeling dependencies explicitly, Parameters or constraints management, Ontology, STEP, and KCModel. Some of these strategies are based on prototypes or approaches having the following main drawbacks: they are time-consuming, they cause data loss, and they are tool dependent.
- RQ4: What are the challenges to manage models of different domains?
  - Motivation: to identify the main challenges in order to identify directions for future work.
  - Answer: Due to the heterogeneous environment this topic belongs, the main cited challenges are interoperability, maintaining consistency, dependency management, and traceability.

To achieve this goal, we have conducted a systematic literature review [77]. Taking into consideration the fact that scientific publications do not always reflect industrial practices, we have decided to include white papers, such as technical reports. Thus, we have covered both industrial and academic sources.

Answering RQ1–RQ3 is useful both for researchers willing to develop new approaches and for tool vendors willing to add new features or improve the features presented in current tools. Answering RQ4 is useful for those researchers that are willing to study model management, to organize the study knowing the challenges they will face.

This study is an extension and revision of our previous work [139]. In this extension, we updated the number of papers causing minor alterations on the first three research questions. Research question four was added for this extension. The **main contributions** of this study are:

- List of model management tools;
- Classification of the inconsistency management approaches;
- Identification of gaps such as the need to improve the current tools to address more kinds of consistency check, and direction for future work indicates that further research should be done on *Interoperability*, *Maintaining Consistency*, *Dependency Management*, and how to capture the *Behavioral*, *Refinement*, and *Requirement* inconsistency types.

2 Related work

To the best of our knowledge, no systematic literature review has considered model management tools focusing on cross-domain model consistency. However, a number of studies consider consistency checking of models and model management focusing on the software engineering domain. All the studies reviewed below focus on models from the same domain, usually the software engineering domain: as opposed to this line of work, we consider cross-domain consistency checking.

Cicchetti et al. [33] conducted a systematic literature review on existing solutions for multi-view modeling of software and systems. The authors further investigated the support for consistency management provided by multi-view modeling solutions. They identified consistency management as one of the most common limitations and recognized importance of a lack of support for semantic consistency management. Franzago et al. [51] conducted a systematic mapping study of collaborative model-driven software engineering approaches from a researcher's viewpoint. The authors decomposed the collaborative Model-Driven Software Engineering (MDSE) approaches into three main dimensions, with model management being one of them. Franzago et al. presented characteristics of the model management infrastructure focusing on the supported artifact, modeling language, multi-view, editor, and application domain.

Bharadwaj et al. [21] conducted a survey of model management literature within the mathematical modeling domain. The authors identified three approaches to support model management on the mathematical modeling domain and categorized various modeling systems based on features they provided.

Santos et al. [122] conducted a systematic mapping study to investigate existing inconsistency management approaches within Software Product Lines. The authors conclude that the existing approaches should provide faster feedback, support co-evolution of the artifacts, and handle the inconsistencies.

Muran et al. [98] conducted a systematic literature review on software behavior model consistency checking. As conclusion the authors suggested that future research should focus on tool support for consistency checking, tool integration, and better strategies for inconsistency handling.

Spanoudakis and Zisman [130] conducted a literature review to investigate techniques and methods that support the management of inconsistencies on models within the software engineering domain. Usman et al. [141] conducted an informal literature review on consistency checking techniques for UML models focusing on five consistency types (inter-/intra-model, evolution, semantic, syntactic). They concluded that almost all techniques provide consistency

rules to validate consistency between UML models. It is worth noting that these studies did not follow a strict literature review protocol.

Lucas et al. [90] conducted a systematic literature review to identify and evaluate the current approaches for model consistency management between UML models. They also briefly proposed a solution to overcome the limitations they found. Ahmad and Nadeem [3] conducted a survey to evaluate Description Logics-based approaches to consistency checking also focusing on UML models.

Torre et al. [138] conducted a systematic mapping study on UML consistency rules and observed that there is limited tool support for checking these rules. Later, Torre et al. [137] conducted a survey in order to understand how model consistency between UML models is addressed in academia and industry.

3 Background: model management

INCOSE[4] defines **model-based systems engineering (MBSE)** as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." [70]. According to Friedenthal et al. [52], MBSE was proposed to facilitate systems engineering activities: following MBSE the system engineers would use models instead of documents, and this is expected to improve quality of system specification and design, as well as the communication among the development team.

A **model** is a representation of reality, an abstraction of something relevant to the stakeholder described using well-defined and unambiguous languages [48].

Model management emerged with the need to organize and maintain models, ensuring consistency. Franzago et al. [51] stated that the infrastructure for the model management may include a model repository and modeling tools. This infrastructure is responsible for managing the life cycle of the models such as creating, editing, and deleting. The focus on this study lies on one aspect of model management: the consistency checking on models from different domains.

Product lifecycle management (PLM) [56,57,118,132,134] is an environment, infrastructure, a system of methods, processes, and practices that cover the entire product lifecycle, from requirements definition, design, to late stages such as maintenance and recycling of the product. While model management is focused on the models of the product, PLM includes every artifact related to the product. Teamcenter[5] is an example of a PLM software that can also provide model management capabilities. Since PLM can include model management features, we have decided to include it in our literature review.

4 Methodology

4.1 Selecting the literature review technique

In order to answer the Research Questions (RQ1–RQ4), we conducted a literature review. Several literature review techniques have been proposed in the scientific literature, e.g., snowballing [145,148], systematic literature review (SLR) [77], and systematic mapping review (SMR) [107].

We opted for SLR because of the SLR characteristics that identify, analyze and interpret the data related to specific RQs. In contrast, SMR aims to answer general research questions and snowballing can be labor intensive. Thus, we believe that this approach is the most appropriate to answer our RQs. To circumvent the inherent SLR limitations implied by the choice of the search strings, we have combined different keywords obtaining 600 different search strings. This process is more extensively explained in the next section.

The SLR consists of the creation of research questions (RQs), the queries on electronic sources having the RQs as a guide, and the use of pre-determined criteria for eligibility and relevance to form the set of accepted papers to be used in the study. As the data source, Kitchenham and Charters [77] recommend the use of a search engine that offers a wide coverage of sources. Thus, we have chosen Google Scholar: it offers a wide coverage of electronic sources of different research areas, and it has been used in multiple software engineering studies [53,71,83,94,100,148].

4.2 Data extraction

Since we are mainly interested in **tools** (product life cycle management tools, and model management tools), we create search strings to query Google Scholar, based on **PICO** [79]. Thus, we have selected and organized keywords into four categories: process supported by tools, model, consistency, and multiple domains. Figure 3 presents the overview of the selection of the keywords. For each category, we have selected keywords related to the Research Questions, as presented in Table 1.

The first two categories (process supported by tools and model) are the base for answering all RQs, and the remaining two categories (consistency and multiple domains) are more

[4] The International Council on Systems Engineering (INCOSE) is a nonprofit membership organization dedicated to the advancement of systems engineering and to raise the professional stature of systems engineers https://www.incose.org.

[5] https://www.plm.automation.siemens.com/teamcenter.

Table 1 Keywords

Process supported by tools	Model	Consistency	Multiple domains
Model management	MBSE	Consistency	Multidomain model integration
Model lifecycle management	Model-based systems engineering	Inconsistency	Multi domain
Product lifecycle management	Model based systems engineering	Dependency	Multiple domains
Systems lifecycle management	Model-based system engineering		Domains
	Model based system engineering		Heterogeneous
	Product modeling		
	Model-driven system engineering		
	Model-driven system engineering		
	Model-driven systems engineering		
	Model-driven systems engineering		

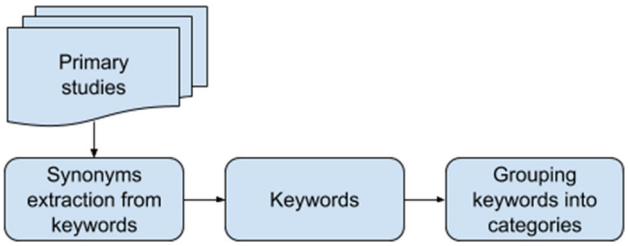


Fig. 3 Overview of the selection of the keywords

specific for answering RQ2 and RQ3. For example, the reasoning behind choosing the keyword "Dependency" in the category "Consistency" was that in our initial research we found that dependency modeling has been used to maintain consistency. Thus, this keyword could help find more results that could answer RQ3.

We have combined the keywords from different categories to create queries to be executed [6] in Google Scholar. For instance, for the first query we have used the following keywords: "Model Management, MBSE, Consistency, Multidomain Model Integration." For the second query we used "Model Management, MBSE, Consistency, Multi Domains," and so on. In total we have $600 = 4 \times 10 \times 3 \times 5$ combinations.

Due to the similarity of the queries, some papers have been retrieved multiple times. We have automatically excluded these duplicates prior to the manual inspection. In total we have obtained 4293 hits, but only 618 of them were unique.

4.3 Manual inspection

The selection criteria were defined in order to avoid bias and to reduce subjectivity. The inclusion (I) and exclusion (E) criteria were designed to answer the RQs, as proposed by Kuhrmann et al. [81]. The following are the inclusion and exclusion criteria we used:

- I1: Studies written in English and available in full text.
- I2: Studies review or proposal of a new technique, approach, method, or tool (prototype[7]) that support model management.
- I3: Studies mention tools related to Product Lifecycle Management (PLM) and Model lifecycle management.
- E1: Studies do not mention model (in)consistency.
- E2: CVs, PhD and Master theses, and books or book chapters. Although we excluded all PhD theses, we considered publications related to the PhD theses and applied the inclusion and exclusion criteria to them. We decided to check for derived papers because we chose to be as conservative as possible, and we did not want to exclude PhD theses without checking for derived papers. In the end of this process, we included 32 derived papers.

In order to identify the relevance of the paper, we read the title, abstract, and conclusion of 618 papers.

Relevance assessment was performed iteratively. At each iteration, we used 15 papers randomly selected from the list of papers we had downloaded. The first and the last authors of this paper individually read the title, abstract, and conclusion to label the relevance of the paper. Both of the raters were software engineering researchers having at least a Master's degree in Computer Science. At the end of each iteration, we computed Cohen's κ [34] to measure the agreement between the raters and discussed the disagreements. According to Cohen [34] and Landis et al. [82], the Kappa coefficient in the range of 0.61–0.80 is interpreted as substantial agreement and this range was used in previous studies [16,22,69].

As presented in Table 2, four review rounds were needed to reach the κ value greater than 0.6. In the first review round,

[6] Although the queries were executed on 20/12/2018, it was possible to collect papers that will be published in 2019.

[7] We consider prototypes those tools that are categorized as such by the authors of the papers.

Table 2 The κ value obtained in each iteration

Iteration	Cohen's kappa value
First review round	0.22
Second review round	0.29
Third review round	0.33
Fourth review round	0.62

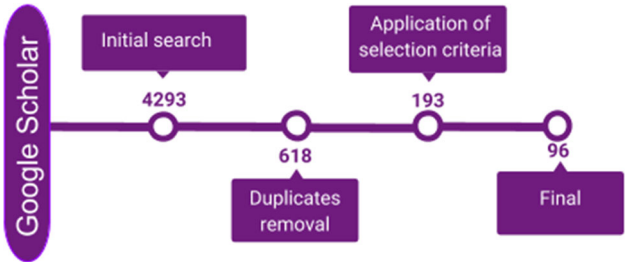


Fig. 4 Search and selection process. We obtained 4293 hits in the initial search. After removing all duplicated hits, we obtained 618 unique hits. We obtained 193 papers when the selection criteria was applied, and we conclude the process with 96 papers

we obtained the lowest κ value, because the interpretation of the inclusion and exclusion criteria was not clear among the raters. We improved the agreement level in every subsequent review round. In the second and third review rounds, we obtained 0.29 and 0.33, respectively. We finalized the fourth round of the process by reading 20 papers instead of 15, and we obtained the agreement level of 0.61.

When we reached the acceptable agreement level, the first author continued the selection procedure independently. After reading the title, abstract, and conclusion of 618 papers, we labeled 193 as possibly relevant. To finalize, the first author completely read these papers and selected 96 papers to answer the Research Questions. Figure 4 presents the summary of the process to select the papers.

4.4 Gathering information about tools

The previous section describes how to identify relevant papers supporting us to find the answers for the RQs. Although the papers provide a list of tools, the information regarding the tools is not necessarily well presented or detailed. As a consequence, we decided to use additional data sources, for instance the website of each tool. One option to gather information about the tools would be to install and to try all the tools. However, this option was not feasible, mainly because of the need to know how to use them, but also because most of the tools are commercial, requiring the license to try them.

We selected the closed card sorting technique [131] to categorize the type of consistency the selected tools address. Taylor et al. [133] describe consistency as *"an internal property of an architectural model, which is intended to ensure that different elements of that model do not contradict one another"* and distinguish the following five inconsistency types.

Name inconsistencies happen when components, connectors or services have the same name. In most programming languages, this kind of inconsistency is trivial to be captured. However, there are cases in which capturing this inconsistency is not trivial. Taylor et al. exemplify that large systems may have two or more similarly named GUI-rendering components. Identifying the misuse of these components can be a difficult task.

Interface inconsistencies happen when connected interface elements have mismatching values, terminologies, or schemes [66]. Name inconsistencies are interface inconsistencies but not vice versa. Taylor et al. [133] explain that *"A component's required service may have the same name as another component's provided service, but their parameter lists, as well as parameter and return types, may differ."*. They exemplify that this inconsistency can be presented in a case where there are methods with the same name but different parameters and the connector between the client and server components is a direct procedure call.

Behavioral inconsistencies Taylor et al. [133] explain that these inconsistencies *"occur between components that request and provide services whose names and interfaces match, but whose behaviors do not."* This kind of inconsistency can happen when the behavior of the element is not the expected one. An example of behavioral inconsistency would be if the service provider assumes that the distance is expressed in kilometers and the requester assumes it to be in miles.

Interaction inconsistencies this kind of inconsistencies can *"occur when a component's provided operations are accessed in a manner that violates certain interaction constraints, such as the order in which the component's operations are to be accessed."* [133]. To exemplify the occurrence of this inconsistency, assuming that there is a *Queue* component (server) that stores a list of elements. This component demands to not be empty before an attempt to remove an element. In case the client component does not respect this constraint, then an interaction inconsistency will happen.

Refinement inconsistencies occur between models of different abstraction levels due to the fact some elements are suppressed/inserted to fit the corresponding abstraction level. Taylor et al. [133] explain that *"a very high-level model of the architecture may only represent the major subsystems and their dependencies, while a lower-level model may elaborate on many details of those subsystem and dependencies."*.

We used the consistency types identified by Taylor et al. [133], to label the selected tools. We opted to use the consistency types provided by the selected tools, in those cases which it was not possible to match the consistency types

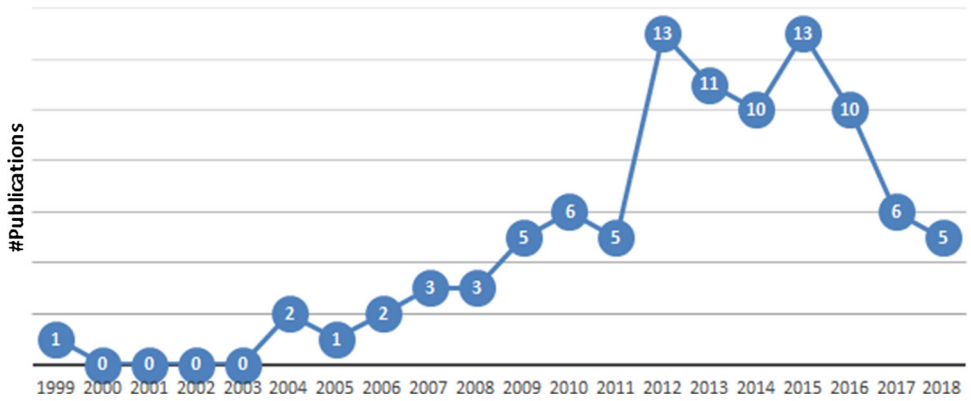


Fig. 5 Selected publications organized by year

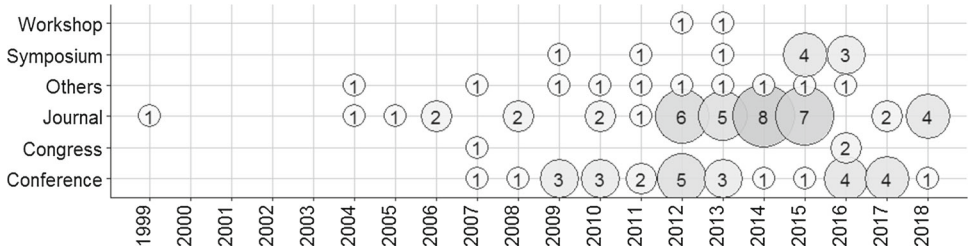


Fig. 6 Distribution of selected publications organize by type and year

identified by Taylor et al. with the description of the consistency type provided by the tools. We labeled DNF (data not found), those tools that we did not find information about the consistency type they address.

Additionally, we conducted a survey designed following the recommendations of Kitchenham and Pfleeger [78]. We contacted the responsible for each tool inquiring whether the tool can check consistency between models from different domains. In case of a positive answer we ask them the consistency types the tool can address. The survey was conducted either via email, via a question and answer form on the official website, or via the official forum of each tool. Not all tools provide contact information, such as an email address. Consequently, we could not contact all of them. We sent 58 messages (15 emails failed to deliver), and we received 24 replies. Our response rate is ≈50%, much higher than response rates commonly reported in the software engineering literature [17,116].

4.5 Categorizing challenges and future work

During the full reading of the selected papers, we collected key sentences that summarize the challenges that are faced by the authors. Then, we applied the open card sorting technique [131] to categorize these challenges.

Complementary to the list of challenges faced by the authors, we also organized the list indicating the direction for future work. In order to organize this list, we followed the same methodology described before. The categories are basically the same, the only category not present is simulation. Those studies that do not explicitly state the future work are grouped in the category "Not Applicable."

5 Data description

We organize the selected publications based on the type (Symposium, Conference, Journal, Congress, Workshop, and Others), venue, and year. It is important to make it clear that these publications do not represent all publications about model management, but only those that are relevant according to our exclusion criteria.

Figure 5 presents the distribution of the publications on model management. The first study was published in 1999 and only after the hiatus of 4 years the number of publications had increased. More precisely, the average number of publications between 1999 and 2011 is less than 3 studies per year, whereas between 2012 and 2018 this number is ≈10 studies per year.

Table 3 presents the publication venues that hosted more than two publications. There are publications on 56 different venues spread on different research areas such as software and system engineering, aerospace engineering, and information engineering. This is an indication that model management is not specific for a research area but a set of different research domains.

The venue with more publications is INCOSE having six publications spread in 2012 (one publication), 2015 (two publications), and 2016 (three publications). Journal papers are the majority representing 43.75% of the selected publications, followed by conference papers 32.29%. Figure 6

Table 3 Publication venues that have more than two selected publications

Publication venue	Domain	Type	#Publications
INCOSE International Symposium	Systems engineering	Symposium	6
Advanced engineering informatics	Engineering informatics	Journal	3
Aerospace conference	Aerospace engineering	Conference	3
Computer-aided design	Applied engineering	Journal	3
Computers in industry	Software engineering	Journal	3
International conference on model driven engineering languages and systems (MODELS)	Software and systems engineering	Conference	3
International conference on product lifecycle management	Systems engineering	Conference	3
Journal of computing and information science in engineering	Information engineering	Journal	3
Procedia CIRP	Production Engineering	Journal	3
Procedia computer science	Computer Engineering	Journal	3
Others	–	–	63
Total		–	96

presents the distributions of papers per type of venue and per year. The type "others" represent the white papers.

6 Results

6.1 RQ1: How do model life cycle management tools address consistency between models from different domains?

We analyzed the descriptions of the tools that were mentioned in the selected papers and we organized them into three categories as described in Table 4.

1. Provide consistency model checking on models of **different** domains: we identified 32 tools that claim they can perform model checking on models of different domains.[8] This number represents 40% of the total amount of tools we found.
2. Provide consistency model checking but only on models of the **same** domain: we identified 20 tools that fit into this category, representing 25% of the total of the tools.
3. Do **not** provide any consistency checking: We assume that tools that do not explicitly claim that they provide consistency check, do not have this functionality. Thus, we identified that 28 tools, 35%, fit into this category. We did not expect to identify this amount of tools, since we use keywords related to multiple domains to restrict the results.

Take away message

We observe that 40% of the tools can provide consistency model checking on models of different domains, 25% on the same domain, and 35% do not provide any consistency model checking.

6.2 RQ2: What consistency types are addressed by the model life cycle management tools?

In order to answer RQ2, we classified the consistency types addressed by the tools identified in the previous subsection.

We focus on those tools that could provide model consistency check at some level, more specifically regarding to what type of consistency those tools provide. However, it is not possible to find the description of the kind and level of consistency check in ≈50% of these tools. For those tools that provide this description, we observe that half of them only address one kind of consistency check. Table 5 presents the list of tools and the consistency types they address.

In Subsection 4.4 we present a list of consistency types identified by Taylor et al. [133]. Additional consistency types are presented below:

- Requirement Consistency—It checks whether the requirements from a requirement list are related to some model element and if this relationship is valid.[9]
- Information Consistency—It checks if the data that can be presented on different media, remain the same regardless of how they are presented [36]. Example of Information inconsistency would be when the distance is presented in different units without respecting the conversion calculation.

Interface and *Interaction* are the two most popular consistency types addressed by the tools, and *Behavioral*, and *Refinement* are the least popular. The complexity in capturing *Behavioral*, and *Refinement* inconsistencies might be the main reason for the low amount of tools implementing these inconsistencies. *Name* inconsistency type can be easily identified. However, we observed that the tool builders do not advertise that their tools address it. Therefore, we conjecture one possible reason: since capturing *Name* inconsistency is trivial and it is also an *Interface* inconsistency (but not vice versa), the tool builders advertise the latter option.

Take away message

The majority of the tools address up to two inconsistency types. *Interface* inconsistency type is the most popular consistency type addressed by the tools

6.3 RQ3: Which strategies have been used to keep the consistency between models of different domains?

We have selected papers that cited tools that manage consistency between models of different domains. We selected 56 papers; however, only half described how they check and keep the consistency between models. We organize the papers into categories according to the approach they use to keep consistency between models of different domains.

Interoperability This approach is defined as *"the ability of two or more software components to cooperate despite differences in language, interface, and execution platform"* [146]. Qamar et al. [110] present the need to manage inconsistency through interoperability between tools such as

[8] These tools can also check consistency between models from the same domain. For the sake of simplicity, they are present only in the different domain category.

[9] https://nl.mathworks.com/help/slrequirements/ug/requirements-consistency-checks.html.

Table 4 Tools organized into three categories (RQ1)

Categories		
Tools that check consistency on models from…		Do not provide any consistency checking
Different domains	Same domain	
Acceleo* [103], ADES [8,106], CADOM [119], Capella (Arcadia) [26,72], CATIA v5 [30,31,55,88,127], CoDeMo [127], COLIBRI [8], Dassault Systèmes PLM platform [5,19, 29,30,55,88,99,126,135,147], Dymola [35,55,66], Epsilon* [103], IntePLM [86], Isight [99,111], LOTAR [88], modeFRONTIER* [111], OntoSTEP (Protégé plug-in) [13,13,27,31,80], OpenModelica*[73], PDES* [88], Pronoia* [42,60], ProSTEP [88], Siemens PLM Tools [5,30, 31,110,112,115,126,136,147], Simulink [35], Syndeia (SLIM)*[10,11,14,54], SysDice [29], SysML4Modelica* [49,74,117], Trails [150], VE4PD [151], VIATRA* [39,103], as well as the academic prototypes or frameworks of Konstantin Kernschmidt [75], Tim Hjertberg [67], István Dávid [38–40], Diana Penciuc [106], and Alfred Sadlauer [121]	Agile PLM (Oracle) [5], Artisan Studio [14,15], Cameo [115], ControlBuild [35], EPLAN [74], EUCLID [30], Magic draw [10,14,15,47,93,99,110,113,115, 117,143], Mechasoft [49], Mechatronics concept designer (Siemens) [29], Melody* [54,102], ModelCenter (Phoenix Integration)* [99,111], MOFLON [15,49,103,125], Windchill (PTC) [1,5,30,106], ParaMagic* [14,102], ParaSolver* [102], Rational rhapsody [14,15,99], SCADE [35], SimMoLib [43], Simscape [29,49], SolidWorks 2010 [31]	Athena Project [106], ATL [103], ATOM3 [103], Autodesk vault*[30], Comet workbench [111], E2KS tool [8], EA Parametrics (EntEterprise Architect)—Sparx systems [102], EAST-ADL2 [124], Entime [2], FUJABA [49,103], GAM framework [120], GReAT*[103], IBM's Jazz collaborative Lifecycle management (CLM) suite [142], i-FEST project [111], Interdisciplinary communication medium (ICM) [119], KerMeta [103], ModelHel'X framework20*[93], Modelisar consortium/FMI: functional mock-up interface [19,50], OBIIS [62], OntoPLM framework [106], Open service for lifecycle collaboration (OSLC) [29], Pro/Engineer Wildfire 4.0 [31], Product design graphics system (PD-GS) [30], Rosetta [93], SAP product life cycle management [126], Share-A-space (Eurostep)* [5], Vitech CORE* [99], VW Surf [30]

The asterisk indicates the tool's authors confirmed the classification by answering the survey

MagicDraw, TeamCenter, and Simulink. On the one hand, standard file formats as Mcad-ecad [32] and XML [37,89] are used to maintain interoperability in engineering and software domains. On the other hand, the use of these standard files to maintaining consistency could be problematic due the data loss [12], since the data would be transiting between different tools and domains.

Inconsistency Patterns This approach recommends selecting the appropriate technique from an extensible catalogue of inconsistency patterns, and apply it in an unmanaged process to achieve a managed one [38,40].

Modeling dependencies explicitly in order to manage inconsistency, some researchers such as [86,111,124,136] believe that making the inter-/intra-model dependencies explicit will facilitate the model management. The main drawback of this approach is, as any other modeling task, it can be time-consuming. Such dependencies can be identified between properties or between structural elements of two models, in such a way that the properties or elements can affect each other. This dependency modeling can be done using any technology that explicitly maps dependencies [110,111,113,115,136,140]. Design Structure Matrix (DSM) is an example of such technology. DSM is a representation of the components and their relations, in order to make the shared information more precise and less ambiguous [86,124,128]. DSM consists of a matrix with properties mapped horizontally and vertically. Each marked box inside a cell of a DSM indicates a dependency between the corresponding properties. A dependency loop occurs when there is a dependency marked above the main diagonal on the DSM. In order to avoid these loops, a reorganization of the DSM is needed [111].

Parameters or constraints management This approach proposes using parameters or constraints to check the model consistency within a multi-disciplinary development team. If these parameters or constraints are violated, the inconsistency can be detected and managed. According to Weingartner et al. [147], to implement this approach, it is necessary to have a well-designed data model of the models one wants to manage [10,119,120,135,147,151].

Ontology is an explicit specification of a conceptualization of properties and relations of one or more domains. A conceptualization is the set of objects, concepts and other entities that are assumed to exist in some area of interest

Table 5 Tools and kind of consistency (RQ2)

Tool	Consistency type checking						DNF
	Behavioral	Information	Interface	Interaction	Refinement	Requirement	
Acceleo							X
ADES			X				
CADOM				X			
Capella (Arcadia)							X
CATIA v5			X				
CoDeMo							X
COLIBRI				X			
Dassault systèmes PLM platform							X
Dymola							X
Epsilon	X	X	X	X	X	X	
IntePLM							X
Isight							X
Lotar							X
ModeFRONTIER	X	X	X	X	X	X	
OntoSTEP (Protégé plug-in)	X			X			
OpenModelica		X	X	X	X	X	
Pdes							X
Pronoia			X	X		X	
ProSTEP							X
Siemens PLM tools							X
Simulink			X			X	
Syndeia			X				
SysML4Modelica	X	X	X		X		
Trails							X
VE4PD		X					
VIATRA	X	X	X	X	X	X	
Prototype [40]							X
Prototype [67]							X
Prototype [75]							X
Prototype [106]			X				
Prototype [120]				X			
Prototype [121]							X

Data not found (DNF) means that, although these tools claim to provide consistency check, they do not describe what kind of consistency check they provide and at which level

together with the relationships that hold among them. A conceptualization is an abstract simplified view of the world to be represented for some purpose [59]. This approach allows engineers to independently develop partial descriptions of the same product and check consistency when the descriptions are combined [24,96,105]. However, creation of an ontology can be a time-consuming task [59].

STEP Standard for the Exchange of Product model data (ISO 10303)[109]. STEP consists of a number of components, called application protocols (APs), which define data models on which translators for CAD data exchange are based. The International Organization for Standardization (ISO) developed STEP in order to cover a wide range of application areas, such as automotive, aerospace, and architecture [32]. In our systematic literature review, we have not found papers that only use STEP to check consistency between models of different domains; instead, they use an extension of STEP, or a combination of STEP and other technologies [13,31,32,80].

KCModel This approach is organized basically into "Information Core Entity" (ICE) and "Configuration Entity" (CE). The former is the smallest information entity used, responsible for storing parameters and rules, and represents a generic multi-domain baseline. In order to use the parameters

and rules in a specific context (3D, thermal calculations, excel files, etc.), it is necessary to create a Configuration Entity instantiating ICE. This approach allows engineers to create their own models, trace parameters and rules, and check consistency. [8,9,18,106]

We identified seven strategies to keep the consistency between models of different domains. Although, some of these strategies are commonly used in the industry, we believe that these strategies are not mature enough because they might cause data loss, they are tool dependent, time-consuming, and they do not (individually) fully support co-evolution of the models.

Take away message

Strategies to keep the consistency between models of different domains are not mature enough: they might cause data loss, are tool-dependent, and do not (individually) fully support co-evolution of the models

6.4 RQ4: What are the challenges to manage models of different domains?

We identify nine main challenges encountered by authors of the selected studies (Table 6). It is possible that the challenges of the studies belong to more than one category. In this case, the study is present in more than one category. The last right column gives examples of each category. Some of these studies do not explicitly state the challenges faced. In this case, these studies are in the category "Not Applicable." The most cited challenges are related to **Interoperability, Maintaining consistency, Dependency Management** and **Traceability**. These challenges are presented in 29, 23, 16, and 10 selected studies, respectively.

Interoperability and **maintaining consistency** are cited as the main challenges. A heterogeneous setting where engineers of different areas of expetise use different design tools can easily cause synchronization issues such as problems with respect to data exchange. Thus, interoperability and maintaining consistency represent important challenges to be faced.

As an extension of the interoperability, the **dependency management** also represents a challenge. It is because the models are created using different technologies not always known by all engineers, making it difficult for them to identify the dependency and relations between these models by themselves. Once the relations are defined, the **traceability** has to be done in order to track affected models due to changes. Questions that arise from this challenge can be "How to create the trace automatically?" or "How to trace the impact of one change?"

Additionally to the list of challenges faced by the authors, we also organized the list indicating the direction for future work (Table 7). As expected, the direction for future work follows the challenges faced by the authors. **Interoperability, Maintaining Consistency,** and **Dependency Management** are also the topics for further research. One additional finding is that almost 40% of the selected studies do not explicitly state the direction for future work, which is surprising, since we did not expect that this number could be this high.

Take away message

The main challenges to manage models on different domains lay on the interoperability between tools and the consistency maintenance.

7 Discussion and future work

The results described in this paper can serve as a starting point for future research on model management topics. We provide a list of available tools used to support model management. We group them according to the functionality they offer related to the consistency model checking on models of different or same domains. We observe that 40% of the tools we found can provide consistency model checking on models from different domains, 25% on the models of the same domain, and 35% do not provide any consistency model checking.

Regarding commercial tools, we have found that they do not fully describe the kind of inconsistency they can address (Sect. 6.2). We conducted a survey to overcome this problem. While ca. 50% response rate is better than one is accustomed to in software engineering surveys, this also means that half of the tool builders did not respond. This lack of information makes it difficult to map the inconsistencies these tools can handle, since these tools are commercial and we would need the licenses and the expertise to use them. Further evaluation on commercial tools is necessary, and this should be done with the help of specialists of each tool or at least a full description of all features should be provided. For those tools that describe the consistency type they address, we have found that the majority of them can perform the *Interface* consistency check, checking whether the connected interface elements have mismatching values. We expected that these tools could address more than one kind of consistency type. However, this was not what we observed. We observed that most of them can address up to two consistency types.

Due to the fact that *Name* inconsistency can be easily captured, we expected that all tools could address this consistency type. However, this was not what we found. We conjecture that the reasons might be that this functionality is indeed implemented; however, the tool builders do not advertise that their tools address it because it is trivial. The

second reason can be due to the fact that *Name* inconsistency is also an *Interface* inconsistency (but not vice versa), and thus, they advertise the later option. *Behavioral*, and *Refinement* are the least addressed consistency types. It might be due to the complexity of capturing them. Thus, for future work, we believe the researchers should investigate how to capture these inconsistency types and tool builders should improve their tools to address more kinds of inconsistency type.

Our study (Sect. 6.3) reveals seven strategies to keep the consistency between models of different domains. These strategies are based on prototypes or approaches having the following main drawbacks: time-consuming, data loss, and they are tool dependent.

According to Qamar et al. [111], explicit dependency modeling between models is not commonly used in the industry. However, this is regarded as a requirement by the academic research if one wishes to manage inconsistency between models from different domains. They claim that *"Capturing dependencies formally and explicitly is currently not supported by available methods and tools in MBSE, and having no explicit knowledge of dependencies is a main cause of inconsistencies and potential failures"*. The explicit dependency modeling between models can be done using design structure matrix and ontology, and it can be followed by the use of standards as STEP.

Reichwein et al. [117] state that *"Due to the wide variety of disciplines and modeling tools that are used in mechatronic design, there is currently no established solution that allows engineers to efficiently and formally define dependencies between different models. Therefore, maintaining consistency between different models is often a manual, time-consuming, and error-prone process."*. Interoperability between tools was also used as a strategy to keep the consistency between models of different domains, specially due the fact that engineers would not need to stop using the tools they are familiarized with.

As stated in Sect. 6.4, there is still room for more research. The main challenges faced by the authors, as well as the proposed directions for future work, belong to the same research topics. The most cited research topics are: interoperability, maintaining consistency, and dependency management. Thus, for future work, we strongly believe the researchers should focus not only on ways of modeling dependency between models of different domains, but also on how to manage these dependencies. The management can be done using a tool agnostic infrastructure that stores in a database all the relationships between models, notifies the owners of those affected models due to a change, and that infers new relationships by analyzing the stored relations. Facilitating the capture of all inconsistency types can be a direction for future work.

8 Threats to validity

Wohlin et al. [149] provide a list of possible threats that researchers can face during a scientific research. In this section, we describe the actions we took in order to increase the validity and decrease the threats.

External validity concerns how the results and findings can be generalized. We only accepted studies written in English, and this can represent a threat despite the fact that English is the most widely used language for scientific papers. As one of the goals of this study is to understand what the industrial practices are we decided to accept gray literature (white papers and technical reports).

The fact that we could not try all the tools and we could not find the full description of the kind of inconsistency they address can represent a threat. We believe that this threat can be minimized with the help of specialists of each tool or at least a full description of all features should be provided.

Internal validity: Google scholar continuously indexes new papers. Hence running the queries at different moments of time might lead to different results. However, it is not possible to run all queries simultaneously due to limitations of Google scholar. We do not think that a considerable amount of papers was missed, since all queries were similar to each other and more than half of our query results were duplicated hits. textbfConstruct validity concerns how the selected studies represent the real population of studies to answer the research questions. To mitigate this concern, in the construction of the search string, we performed an informal literature review that helped us in the selection of the appropriated keywords, and we used different variations of the same keyword. To such a degree, we are confident that our queries are broad enough that all relevant papers were found in our automatic search. In order to mitigate possible bias that could be present in the manual inspection, we strictly followed the inclusion and exclusion criteria to select the relevant papers. Additional to it, the relevance assessment was performed iteratively. At the end of each iteration, we measured the inter-researcher agreement level, and we obtained the Cohen's κ coefficient of 0.61, which is interpreted as substantial agreement.

Conclusion validity concerns the relations between the conclusions that we draw and the analyzed data. In order to mitigate this concern, we followed well known systematic research methods, and we described all decisions we made. Thus, this study can be replicated by other researchers. Of course, the gross number of papers can change, because new papers can be published or some papers might not be available online anymore, but we believe that the final conclusion will not deviate from ours.

Table 6 The main challenges encountered by authors of the selected papers to manage models of different domains (RQ4)

Category	#Studies	Studies	Examples (paper fragments)
Dependency management	16	[2,20,42,60,67,76,91,104,112,113,115,120,127,128, 136,150]	"…it becomes even more challenging and difficult to address the above reported challenges while performing mechatronic design, especially due to cross-domain dependencies" [113]
Design conflict	7	[30,44,45,61,75,121,154]	"…minimize the appearance of early design conflicts and to solve the rest of them…" [45]
Interoperability	29	[1,5,11,13–15,26,27,29,31,32,49,54,55,62,72,73,80, 103,106,108,111,118,119,123,126,129,147,152]	"…the challenge is that models exist in different tools, i.e., there is a tool-integration problem…." [111]
Lack of language	3	[10,105,117]	"…no currently available modeling language can represent all aspects of a system…" [10]
Maintain consistency	23	[8,24,38–40,47,50,61,64– 66,74,86,87,91,93,95,97,111,113,125,128,140]	"…ensuring consistency across models is a challenging issue…" [128]
Simulation	5	[20,54,68,124,135]	"…have efficient simulations involving large hybrid models of complex multi-systems architectures…" [135]
Standardization	6	[25,28,41,96,151,153]	"…lack of standards for information exchange and presentation…" [151]
Traceability	10	[20,35,43,54,61,86,91,113,142,143]	"…the core operational challenges of searching, traceability, accessibility and visibility…" [35]
Visualization	2	[85,110]	"…another challenge is to visualize the information in the model for different stakeholders…" [110]
Not applicable	7	[4,23,84,88,92,99,102]	-

Not Applicable means that these studies do not explicitly state the challenges faced

Table 7 The direction of the future work organized by categories (RQ4)

Category	#Studies	Studies
Dependency management	13	[2,42,44,60,61,65,104,111,113,115,127,128,136]
Design conflict	2	[45,120]
Interoperability	18	[10,13,27,29–32,43,74,80,106,108,110,123,125,126,147,152]
Lack of language	5	[49,61,72,105,117]
Maintain consistency	14	[8,11,15,24,38–40,47,64,66,68,87,95,140]
Standardization	3	[5,25,129]
Traceability	3	[86,112,150]
Visualization	2	[67,85]
Not applicable	37	[1,4,14,20,23,26,28,35,41,50,54,55,58,62,73,75,76,84,88,91–93,96,97,99,102,103,118,119,121,124,135,142,143,151,153,154]

Not applicable means that these studies do not explicitly state the direction of the future work

9 Conclusions

We presented a systematic literature review intending to give an overview of industrial practices and academic approaches to cross-domain model management. We started with 618 potentially relevant studies, and after a rigorous selection criteria we concluded the process with 96 papers.

We provide a list of available tools used to support model management. We observed that 40% of the tools can provide consistency model checking on models of different domains, 25% on the same domain, and 35% do not provide any consistency model checking.

Our study reveals that the strategies to keep the consistency between models of different domains are not mature enough because they might cause data loss, are tool dependent, and do not (individually) fully support co-evolution of the models. Moreover, the majority of the tools address no more than two kinds of consistency.

Due to the lack of details about the kind of inconsistency that commercial tools address, we suggest that a further evaluation on commercial tools is needed. This should be done with the help of specialists of each tool or at least a full description of all features should be provided. We believe that future work should be towards the creation of tool agnostic infrastructure to manage the relationships between models of different domains.

To conclude, we observe that more research has to be done to improve the quality of the approaches and tools used to ensure consistency. There is no silver bullet, but at least we have a set of strategies that together can provide consistency.

Acknowledgements This work has been supported by the European Commission through the ENABLE-S3 project that has received funding from the ECSEL Joint Undertaking under grant agreement No 692455.

References

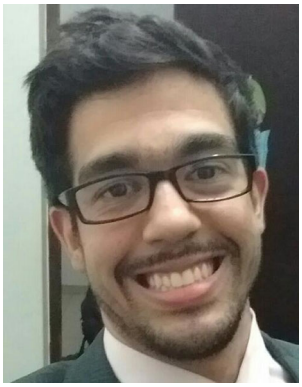
1. Abid, H., Pernelle, P., Noterman, D., Campagne, J.P., Ben Amar, C.: SysML approach for the integration of mechatronics system within PLM systems. Int. J. Comput. Integr. Manuf. **28**(9), 972–987 (2015)
2. Abrishamchian, F., Trächtler, A.: Configuration of mechatronic systems using feature models. Proc. Technol. **15**, 27–34 (2014)
3. Ahmad, M.A., Nadeem, A.: Consistency checking of UML models using description logics: a critical review. In: 2010 6th International Conference on Emerging Technologies (ICET), pp. 310–315. IEEE (2010)
4. Alatalo, P., Hyysalo, J., Mettovaara, V., Salonpää, P., Kuvaja, P., Heinonen, S., Kääriäinen, J., Soininen, S., Tihinen, M.: Merlin white paper
5. Aram, S., Eastman, C.: Integration of PLM solutions and BIM systems for the AEC industry. In: ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, vol. 30, p. 1. Vilnius Gediminas Technical University, Department of Construction Economics and Property (2013)
6. Babur, Ö., Cleophas, L., van den Brand, M., Tekinerdogan, B., Aksit, M.: Models, more models, and then a lot more. In: Federation of International Conferences on Software Technologies: Applications and Foundations, pp. 129–135. Springer (2017)
7. Babur, O.: Model Analytics and Management. Ph.D. Thesis, Department of Mathematics and Computer Science. Proefschrift (2019)
8. Badin, J., Monticolo, D., Chamoret, D., Gomes, S.: Knowledge configuration management for product design and numerical simulation. In: International Conference on Engineering Design—Impacting Society Through Engineering Design, vol. 6 (2011)
9. Badin, J., Monticolo, D., Chamoret, D., Gomes, S.: Using the knowledge configuration model to manage knowledge in config-

uration for upstream phases of the design process. Int. J. Interact. Des. Manuf. (IJIDeM) **5**(3), 171 (2011)
10. Bajaj, M., Zwemer, D., Peak, R., Phung, A., Scott, A.G., Wilson, M.: Slim: collaborative model-based systems engineering workspace for next-generation complex systems. In: Aerospace Conference, pp. 1–15. IEEE (2011)
11. Bajaj, M., Zwemer, D., Yntema, R., Phung, A., Kumar, A., Dwivedi, A., Waikar, M.: Mbse++–foundations for extended model-based systems engineering across system lifecycle. In: INCOSE International Symposium, vol. 26, pp. 2429–2445. Wiley Online Library (2016)
12. Ball, A., Ding, L., Patel, M.: An approach to accessing product data across system and software revisions. Adv. Eng. Inform. **22**(2), 222–235 (2008)
13. Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., Sriram, R.D.: Ontostep: enriching product model data using ontologies. Comput. Aided Des. **44**(6), 575–590 (2012)
14. Barbieri, G., Fantuzzi, C., Borsari, R.: Tools for the development of a design methodology for mechatronic systems. In: IEEE 18th Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–4. IEEE (2013)
15. Barbieri, G., Fantuzzi, C., Borsari, R.: A model-based design methodology for the development of mechatronic systems. Mechatronics **24**(7), 833–843 (2014)
16. Barbosa, L., Feng, J.: Robust sentiment detection on twitter from biased and noisy data. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pp. 36–44. Association for Computational Linguistics (2010)
17. Begel, A., Zimmermann, T.: Analyze this! 145 questions for data scientists in software engineering. In: P. Jalote, L.C. Briand, A. van der Hoek (eds.) 36th International Conference on Software Engineering, ICSE '14, Hyderabad, India—May 31–June 07, 2014, pp. 12–23. ACM (2014). https://doi.org/10.1145/2568225.2568233
18. Belkadi, F., Dremont, N., Notin, A., Troussier, N., Messadia, M.: A meta-modelling framework for knowledge consistency in collaborative design. Annu. Rev. Control **36**(2), 346–358 (2012)
19. Belloncle, G., Chombart, P., Clark, B.: An integrated approach to developing automotive climate control systems. In: Complex Systems Design and Management, pp. 209–226. Springer (2013)
20. Belloncle, G., Chombart, P., Clark, B.: An integrated approach to developing automotive climate control systems. In: Aiguier, M., Caseau, Y., Krob, D., Rauzy, A. (eds.) Complex Systems Design and Management, pp. 209–226. Springer, Berlin (2013)
21. Bharadwaj, A., Choobineh, J., Lo, A., Shetty, B.: Model management systems: a survey. Ann. Oper. Res. **38**(1), 17–67 (1992)
22. Bjørnson, F.O., Dingsøyr, T.: Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used. Inf. Softw. Technol. **50**(11), 1055–1068 (2008)
23. Blondelle, G., Bordeleau, F., Exertier, D.: Polarsys: a new collaborative ecosystem for open source solutions for systems engineering driven by major industry players. INSIGHT **18**(2), 35–38 (2015)
24. Bock, C., Zha, X., Suh, H., Lee, J.H.: Ontological product modeling for collaborative design. Adv. Eng. Inform. **24**(4), 510–524 (2010)
25. Bogusch, R., Ehrich, S., Scherer, R., Sorg, T., Wöhler, R.: A lean systems engineering approach for the development of safety-critical avionic systems. In: 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016) (2016)
26. Bonnet, S., Voirin, J.L., Normand, V., Exertier, D.: Implementing the mbse cultural change: Organization, coaching and lessons learned. In: INCOSE International Symposium, vol. 25, pp. 508–523. Wiley Online Library (2015)
27. Borsato, M.: Bridging the gap between product lifecycle management and sustainability in manufacturing through ontology building. Comput. Ind. **65**(2), 258–269 (2014)
28. Brusa, E., Ferretto, D., Calà, A.: Integration of heterogeneous functional-vs-physical simulation within the industrial system design activity. In: 2015 IEEE International Symposium on Systems Engineering (ISSE), pp. 303–310. IEEE (2015)
29. Chami, M., Bruel, J.M.: Towards an integrated conceptual design evaluation of mechatronic systems: the sysdice approach. Proc. Comput. Sci. **51**, 650–659 (2015)
30. Chandrasegaran, S.K., Ramani, K., Sriram, R.D., Horváth, I., Bernard, A., Harik, R.F., Gao, W.: The evolution, challenges, and future of knowledge representation in product design systems. Comput. Aided Des. **45**(2), 204–228 (2013)
31. Chaparala, R.T., Hartman, N.W., Springer, J.: Examining cad interoperability through the use of ontologies. Comput. Aided Des. Appl. **10**(1), 83–96 (2013)
32. Chen, K., Schaefer, D.: MCAD-ECAD integration: overview and future research perspectives. In: ASME 2007 International Mechanical Engineering Congress and Exposition, pp. 123–132. American Society of Mechanical Engineers (2007)
33. Cicchetti, A., Ciccozzi, F., Pierantonio, A.: Multi-view approaches for software and system modelling: a systematic literature review. Softw. Syst. Model. **18**(6), 3207–3233 (2019)
34. Cohen, J.: A coefficient of agreement for nominal scales. Educ. Psychol. Meas. **20**(1), 37–46 (1960)
35. Corbier, F., Soodeen, M., Loembe, S., Thurston, G.: Creating A Systems Simulation Framework and Roadmap. Technical Reports on SAE Technical Paper (2013)
36. Costello, J., Canestraro, D.S., Gil-Garcia, J.R., Werthmuller, D.: Using XML for Web Site Management: Lessons Learned Report. Center for Technology in Government University at Albany, Suny (2007)
37. Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, vol. 45, pp. 1–17 (2003)
38. Dávid, I., Denil, J., Gadeyne, K., Vangheluwe, H.: Engineering process transformation to manage (in)consistency. In: COMMitMDE@MoDELS (2016)
39. Dávid, I., Meyers, B., Vanherpen, K., Tendeloo, Y.V., Berx, K., Vangheluwe, H.: Modeling and enactment support for early detection of inconsistencies in engineering processes. In: MODELS (2017)
40. Dávid, I.: A multi-paradigm modeling foundation for collaborative multi-view model/system development. In: SRC@MoDELS (2016)
41. de Moura, A.F.C.S., Szejka, A.L., Junior, O.C., Annunziato, R.C., et al.: A discussion on current issues for semantic interoperability in an integrated product development process. In: IFIP International Conference on Product Lifecycle Management, pp. 117–125. Springer (2018)
42. Demoly, F., Matsokis, A., Kiritsis, D.: A mereotopological product relationship description approach for assembly oriented design. Robot. Comput. Integr. Manuf. **28**(6), 681–693 (2012)
43. Deshmukh, M., Schwarz, R., Braukhane, A., Lopez, R.P., Gerndt, A.: Model linking to improve visibility and reusability of models during space system development. In: Aerospace Conference, pp. 1–11. IEEE (2014)
44. Dutra, M., da Silva, C.F., Ghodous, P., Gonçalves, R.: Using an inference engine to detect conflicts in collaborative design. In: IEEE International Technology Management Conference (ICE), pp. 1–8. IEEE (2008)
45. Dutra, M., Ghodous, P.: A reasoning approach for conflict dealing in collaborative design. In: Complex Systems Concurrent Engineering, pp. 495–502. Springer (2007)

46. Example Robotarm: https://www.mathworks.com/help/control/ug/multi-loop-pid-control-of-a-robot-arm.html. Accessed 09 June 2020
47. Feldmann, S., Herzig, S.J., Kernschmidt, K., Wolfenstetter, T., Kammerl, D., Qamar, A., Lindemann, U., Krcmar, H., Paredis, C.J., Vogel-Heuser, B.: Towards effective management of inconsistencies in model-based engineering of automated production systems. IFAC-PapersOnLine **48**(3), 916–923 (2015)
48. Fisher, A., Nolan, M., Friedenthal, S., Loeffler, M., Sampson, M., Bajaj, M., VanZandt, L., Hovey, K., Palmer, J., Hart, L.: Model lifecycle management for MBSE. In: INCOSE International Symposium, vol. 24, pp. 207–229. Wiley Online Library (2014)
49. Fotso, A.B., Rettberg, A.: State of the art for mechatronic design concepts. In: IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications (MESA), pp. 232–240. IEEE (2012)
50. Fourgeau, E., Gomez, E., Hagege, M.: Managing the embedded systems development process with product lifecycle management. In: Complex Systems Design and Management, pp. 147–158. Springer (2016)
51. Franzago, M., Ruscio, D.D., Malavolta, I., Muccini, H.: Collaborative model-driven software engineering: a classification framework and a research map. IEEE Trans. Softw. Eng. **44**, 1146–1175 (2017)
52. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann, Burlington (2014)
53. Gerpheide, C.M., Schiffelers, R.R.H., Serebrenik, A.: Assessing and improving quality of qvto model transformations. Softw. Qual. J. **24**(3), 797–834 (2016). https://doi.org/10.1007/s11219-015-9280-8
54. Gondhalekar, A., Kale, S., Vidap, A.: Tool-agnostic framework for systems engineering implementation. In: INCOSE International Symposium, vol. 26, pp. 1–10. Wiley Online Library (2016)
55. Graignic, P., Vosgien, T., Jankovic, M., Tuloup, V., Berquet, J., Troussier, N.: Complex system simulation: proposition of a MBSE framework for design-analysis integration. Proc. Comput. Sci. **16**, 59–68 (2013)
56. Grieves, M.: Virtually Perfect: Driving Innovative and Lean Products Through Product Lifecycle Management. Space Coast Press (2011)
57. Grieves, M.: Product Lifecycle Management: Driving the Next Generation of Lean Thinking: Driving the Next Generation of Lean Thinking. McGraw Hill Professional, New York (2005)
58. Gross, J., Reichwein, A., Bock, D., Laufer, R., Rudolph, S.: An executable unified product model based on UML to support satellite design. In: AIAA Space 2009 Conference and Exposition, p. 6642 (2009)
59. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (1993)
60. Gruhier, E., Demoly, F., Dutartre, O., Abboudi, S., Gomes, S.: A formal ontology-based spatiotemporal mereotopology for integrated product design and assembly sequence planning. Adv. Eng. Inform. **29**(3), 495–512 (2015)
61. Guychard, C., Guerin, S., Koudri, A., Beugnard, A., Dagnat, F.: Conceptual interoperability through models federation. In: Semantic Information Federation Community Workshop (2013)
62. He, L., Ming, X., Ni, Y., Li, M., Zheng, M., Xu, Z.: Ontology-based information integration and sharing for collaborative part and tooling development. Concurr. Eng. **23**(3), 199–212 (2015)
63. Herzig, S.J., Qamar, A., Reichwein, A., Paredis, C.J.: A conceptual framework for consistency management in model-based systems engineering. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 1329–1339. American Society of Mechanical Engineers (2011)
64. Herzig, S.J., Paredis, C.J.: A conceptual basis for inconsistency management in model-based systems engineering. Proc. CIRP **21**, 52–57 (2014)
65. Herzig, S.J., Qamar, A., Paredis, C.J.: An approach to identifying inconsistencies in model-based systems engineering. Proc. Comput. Sci. **28**, 354–362 (2014)
66. Hisarciklilar, O., Rahmani, K., Thomson, V.: A conflict detection approach for collaborative management of product interfaces. In: ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 555–563. American Society of Mechanical Engineers (2010)
67. Hjertberg, T., Stolt, R., Elgh, F.: A tool for obtaining transparency and traceability in heterogeneous design automation environments. Comput. Aided Des. Appl. **15**(4), 488–500 (2018)
68. Horváth, L., Rudas, I.J.: Knowledge engineering for modeling and simulation in virtual product development. In: 5th International Symposium on Computational Intelligence and Intelligent Informatics (ISCIII), pp. 111–116. IEEE (2011)
69. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. **51**, 915–929 (2015)
70. INCOSE: Systems Engineering Vision 2020. Technical Reports on INCOSE-TP-2004-004-02 (2007)
71. Jalali, S., Wohlin, C.: Systematic literature studies: Database searches vs. backward snowballing. In: International Symposium on Empirical Software Engineering and Measurement, pp. 29–38 (2012)
72. Jinzhi, L., Chen, D., Törngren, M., Loiret, F.: A model-driven and tool-integration framework for whole vehicle co-simulation environments. In: European Congress on Embedded Real Time Software and Systems. No (2016)
73. Johnson, T., Kerzhner, A., Paredis, C.J., Burkhart, R.: Integrating models and simulations of continuous dynamics into SysML. J. Comput. Inf. Sci. Eng. **12**(1), 011–002 (2012)
74. Kernschmidt, K., Barbieri, G., Fantuzzi, C., Vogel-Heuser, B.: Possibilities and challenges of an integrated development using a combined SysML-model and corresponding domain specific models. In: MIM, pp. 1465–1470 (2013)
75. Kernschmidt, K., Feldmann, S., Vogel-Heuser, B.: A model-based framework for increasing the interdisciplinary design of mechatronic production systems. J. Eng. Des. **29**(11), 617–643 (2018)
76. Khan, M., Madisetti, V.K.: Multi-domain model based system engineering using SysML for networked systems
77. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Engineering **2**, 1051 (2007)
78. Kitchenham, B.A., Pfleeger, S.L.: Principles of survey research part 2: designing a survey. SIGSOFT Softw. Eng. Notes **27**(1), 18–20 (2002). https://doi.org/10.1145/566493.566495
79. Kitchenham, B.A., Mendes, E., Travassos, G.H.: Cross versus within-company cost estimation studies: a systematic review. IEEE Trans. Softw. Eng. **5**, 316–329 (2007)
80. Krima, S., Barbau, R., Fiorentini, X., Sudarsan, R., Sriram, R.D.: Ontostep: OWL-DL Ontology for Step, vol. 7561. National Institute of Standards and Technology (NISTIR), Gaithersburg (2009)
81. Kuhrmann, M., Fernández, D.M., Daneva, M.: On the pragmatic design of literature studies in software engineering: An experience-based guideline. CoRR arXiv:1612.03583 (2016)
82. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics 159–174 (1977)
83. Landman, D., Serebrenik, A., Vinju, J.J.: Challenges for static analysis of Java reflection-literature review and empirical study. In: International Conference on Software Engineering, pp. 507–518. IEEE (2017)

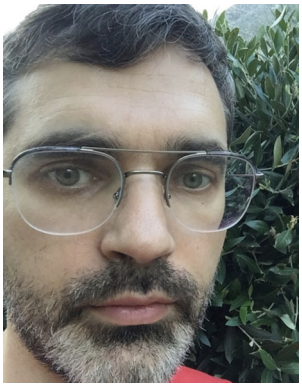
84. Lee, J., Fenves, S., Bock, C., Suh, H., Rachuri, S., Fiorentini, X., Sriram, R.: A Semantic Product Modeling Framework and Language for Behavior Evaluation, vol. 7681. National Institute of Standards and Technology, Gaithersburg (2010)
85. Lee, J.H., Fenves, S.J., Bock, C., Suh, H.W., Rachuri, S., Fiorentini, X., Sriram, R.D.: A semantic product modeling framework and its application to behavior evaluation. IEEE Trans. Autom. Sci. Eng. **9**(1), 110–123 (2012)
86. Li, Y., Wan, L., Xiong, T.: Product data model for PLM system. Int. J. Adv. Manuf. Technol. **55**(9–12), 1149–1158 (2011)
87. Liang, C., Guodong, J.: Product modeling for multidisciplinary collaborative design. Int. J. Adv. Manuf. Technol. **30**(7–8), 589–600 (2006)
88. Lubell, J., Frechette, S.P., Lipman, R.R., Proctor, F.M., Horst, J.A., Carlisle, M., Huang, P.J.: Model-Based Enterprise Summit Report. Technical Reports on Army Research Lab Aberdeen Proving Ground MD Weapons and Materials Research Directorate (2014)
89. Lubell, J.: From model to markup: Xml representation of product data (2002)
90. Lucas, F.J., Molina, F., Toval, A.: A systematic review of UML model consistency management. Inf. Softw. Technol. **51**(12), 1631–1645 (2009)
91. Ma, Y.S., Chen, G., Thimm, G.: Paradigm shift: unified and associative feature-based concurrent and collaborative engineering. J. Intell. Manuf. **19**(6), 625–641 (2008)
92. Madsen, J., Munck, A.: A systematic and practical method for selecting systems engineering tools. In: IEEE International Systems Conference (SysCon), pp. 1–8. IEEE (2017)
93. Mckelvin Jr., M.L., Castillo, R., Bonanne, K., Bonnici, M., Cox, B., Gibson, C., Leon, J.P., Gomez-Mustafa, J., Jimenez, A., Madni, A.M.: A principled approach to the specification of system architectures for space missions. In: AIAA Space 2015 Conference and Exposition, p. 4462 (2015)
94. Moghaddam, F.A., Lago, P., Grosso, P.: Energy-efficient networking solutions in cloud-based environments: a systematic literature review. ACM Comput. Surv. **47**(4), 64:1–64:32 (2015)
95. Monticolo, D., Badin, J., Gomes, S., Bonjour, E., Chamoret, D.: A meta-model for knowledge configuration management to support collaborative engineering. Comput. Ind. **66**(C), 11–20 (2015)
96. Mostefai, S., Bouras, A., Batouche, M.: Effective collaboration in product development via a common sharable ontology. Int. J. Comput. Intell. **2**(4), 206–212 (2005)
97. Müller, P.: Configuration management-a core competence for successful through-life systems engineering and engineering services. Proc. CIRP **11**, 187–192 (2013)
98. Muram, F., Tran, H., Zdun, U.: Systematic review of software behavioral model consistency checking. ACM Comput. Surv. (CSUR) **50**(2), 1–39 (2017)
99. Murray, J.: Model based systems engineering (MBSE) media study. INCOSE International Symposium (2012)
100. Muske, T., Serebrenik, A.: Survey of approaches for handling static analysis alarms. In: IEEE International Working Conference on Source Code Analysis and Manipulation, pp. 157–166 (2016)
101. NASA: Report on Project Management in NASA: Phase II of the Mars Climate Orbiter Mishap Report, March 2000. Technical Reports on Mars Climate Orbiter, Mishap Investigation Board (2000)
102. O'Brien, S., Peak, R., Alldredge, P., Warden, L., Fortune, J., Cimtalay, S., Scott, A., Wilson, M., Aikens, B., Martin, D.: Verification, Validation and Accreditation Using AADL. Technical Reports on Systems Engineering Research Center, Hoboken (2011)
103. Paige, R.F., Varró, D.: Lessons learned from building model-driven development tools. Softw. Syst. Model. **11**(4), 527–539 (2012)
104. Peak, R.S.: Capturing Design Process Information and Rationale to Support Knowledge-Based Design and Analysis Integration. Technical Reports on Georgia Institute of Technology (2004)
105. Penas, O., Plateaux, R., Patalano, S., Hammadi, M.: Multi-scale approach from mechatronic to cyber-physical systems for the design of manufacturing systems. Comput. Ind. **86**, 52–69 (2017)
106. Penciuc, D., Durupt, A., Belkadi, F., Eynard, B., Rowson, H.: Towards a plm interoperability for a collaborative design support system. Proc. CIRP **25**, 369–376 (2014)
107. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. EASE **8**, 68–77 (2008)
108. Plateaux, R., Penas, O., Hehenberger, P., Hammadi, M., Mhenni, F., Warniez, A., Choley, J.Y.: Needs for a 3d enriched ontology for mechatronic systems design. In: IEEE International Symposium on Systems Engineering (ISSE), pp. 253–260. IEEE (2015)
109. Pratt, M.: ISO 10303, the step standard for product data exchange, and its PLM capabilities **1** (2005)
110. Qamar, A., Meinhart, M., Walley, G.: Model based systems engineering to support failure mode avoidance for driver-assistance systems. In: Aerospace Conference, pp. 1–9. IEEE (2017)
111. Qamar, A., Paredis, C.J., Wikander, J., During, C.: Dependency modeling and model management in mechatronic design. Journal of Computing and Information Science in Engineering **12**(4), 041,009 (2012)
112. Qamar, A., Törngren, M., Wikander, J., During, C.: Integrating multi-domain models for the design and development of mechatronic systems. In: 7th European Systems Engineering Conference EuSEC 2010. INCOSE (2010)
113. Qamar, A., Wikander, J., During, C.: Overcoming current mechatronic design challenges: a discussion. In: 13th Mechatronics Forum International Conference (2012)
114. Qamar, A.: Model and Dependency Management in Mechatronic Design. Ph.D. Thesis, KTH Royal Institute of Technology (2013)
115. Qamar, A., Wikander, J., During, C.: Managing dependencies in mechatronic design: a case study on dependency management between mechanical design and system design. Eng. Comput. **31**(3), 631–646 (2015)
116. Qiu, H.S., Nolte, A., Brown, A., Serebrenik, A., Vasilescu, B.: Going farther together: the impact of social capital on sustained participation in open source. In: J.M. Atlee, T. Bultan, J. Whittle (eds.) Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25–31, pp. 688–699. IEEE/ACM (2019). https://doi.org/10.1109/ICSE.2019.00078
117. Reichwein, A., Paredis, C.J., Canedo, A., Witschel, P., Stelzig, P.E., Votintseva, A., Wasgint, R.: Maintaining consistency between system architecture and dynamic system models with sysml4modelica. In: Proceedings of the 6th International Workshop on Multi-Paradigm Modeling, pp. 43–48. ACM (2012)
118. Rickover, A.H.G.: Product lifecycle management: the salvation of systems engineering (2015)
119. Rosenman, M., Wang, F.: Cadom: a component agent-based design-oriented model for collaborative design. Res. Eng. Des. **11**(4), 193–205 (1999)
120. Sadeghi, M., Noel, F., Hadj-Hamou, K.: Development of control mechanisms to support coherency of product model during cooperative design process. J. Intell. Manuf. **21**(4), 539–554 (2010)
121. Sadlauer, A., Riedl-Ehrenleitner, M., Hehenberger, P., Demuth, A., Egyed, A.: The practical use of inconsistency information in engineering design tasks-first observations. Int. J. Prod. Lifecycle Manag. **10**(2), 171–190 (2017)
122. Santos, A.R., de Oliveira, R.P., de Almeida, E.S.: Strategies for consistency checking on software product lines: a mapping study. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–14 (2015)

123. Sarigecili, M.I., Roy, U., Rachuri, S.: Interpreting the semantics of GD&T specifications of a product for tolerance analysis. Comput. Aided Des. **47**, 72–84 (2014)
124. Sellgren, U., Törngren, M., Malvius, D., Biehl, M.: PLM for mechatronics integration. In: Proceedings of the 6th International Product Lifecycle Management Conference (2009)
125. Shah, A.A., Schaefer, D., Paredis, C.: Enabling multi-view modeling with sysml profiles and model transformations. In: The 6th International Conference on Product Lifecycle Management, pp. 527–538. University of Bath (2009)
126. Shaout, A., Arora, M., Awad, S.: Automotive software development and management. In: International Computer Engineering Conference (ICENCO), pp. 9–15. IEEE (2010)
127. Sharon, A., Dori, D., De Weck, O.: Model-based design structure matrix: deriving a DSM from an object-process model. In: Second International Symposium on Engineering Systems, pp. 1–12 (2009)
128. Sirin, G., Yannou, B., Coatanéa, E., Landel, E.: Analyze of the simulation system in an automotive development project (2012)
129. Song, H., Roucoules, L., Eynard, B., Lafon, P.: Interoperability between a cooperative design modeler and a cad system: Software integration versus data exchange. J. Manuf. Sci. Prod. **7**(2), 139–149 (2006)
130. Spanoudakis, G., Zisman, A.: Inconsistency management in software engineering: survey and open research issues. In: Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals, pp. 329–380. World Scientific (2001)
131. Spencer, D.: Card Sorting: Designing Usable Categories. Rosenfeld Media, New York (2009)
132. Stark, J.: Product lifecycle management. In: Product Lifecycle Management (Volume 1), pp. 1–29. Springer (2015)
133. Taylor, R., Medvidovic, N., Dashofy, E.: Software Architecture: Foundations, Theory, and Practice. Wiley, Hoboken (2009)
134. Terzi, S., Bouras, A., Dutta, D., Garetti, M., Kiritsis, D.: Product lifecycle management:from its history to its new role. Int. J. Prod. Lifecycle Manag. **4**(4), 360–389 (2010)
135. Thomas, E., Ravachol, M., Quincy, J.B., Malmheden, M.: Collaborative complex system design applied to an aircraft system. In: Proceedings of the 9th International MODELICA Conference, 076, pp. 855–866. Linköping University Electronic Press (2012)
136. Törngren, M., Qamar, A., Biehl, M., Loiret, F., El-Khoury, J.: Integrating viewpoints in the development of mechatronic products. Mechatronics **24**(7), 745–762 (2014)
137. Torre, D., Genero, M., Labiche, Y., Elaasar, M.: How consistency is handled in model driven software engineering and UML: a survey of experts in academia and industry (2018)
138. Torre, D., Labiche, Y., Genero, M.: Uml consistency rules: a systematic mapping study. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–10 (2014)
139. Torres, W.S., van den Brand, M.G.J., Serebrenik, A.: Model management tools for models of different domains: a systematic literature review. In: The 13th Annual IEEE International Systems Conference (2019)
140. Tristl, C., Karcher, A.: Integrating systems and mechanical/electrical engineering-how model-based interface management supports multi-domain collaboration. In: International Design Conference (2012)
141. Usman, M., Nadeem, A., Kim, T.h., Cho, E.s.: A survey of consistency checking techniques for UML models. In: 2008 Advanced Software Engineering and Its Applications, pp. 57–62. IEEE (2008)
142. VanZandt, L.: Engineering lifecycle management. what a bunch of rhetoric. In: INCOSE International Symposium, vol. 26, pp. 1905–1921. Wiley Online Library (2016)
143. Vileiniskis, T., Skersys, T., Pavalkis, S., Butleris, R., Butkiene, R.: Lightweight approach to model traceability in a case tool. In: AIP Conference Proceedings, vol. 1863, p. 330008. AIP Publishing (2017)
144. Vosgien, T., Van, T.N., Jankovic, M., Eynard, B., Bocquet, J.C.: Towards model-based system engineering for simulation-based design in product data management systems. In: IFIP International Conference on Product Lifecycle Management, pp. 612–622. Springer (2012)
145. Webster, J., Watson, R.T.: Analyzing the past to prepare for the future: writing a literature review. MIS Q 8–23 (2002)
146. Wegner, P.: Interoperability. ACM Comput. Surv. (CSUR) **28**(1), 285–287 (1996)
147. Weingartner, L., Hehenberger, P., Friedl, M., Kellner, A., Boschert, S., Rosen, R.: A lightweight approach to manage engineering parameters in mechatronic design processes. In: IFIP International Conference on Product Lifecycle Management, pp. 79–88. Springer (2016)
148. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, p. 38. ACM (2014)
149. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Berlin (2012)
150. Wolfenstetter, T., Basirati, M.R., Böhm, M., Krcmar, H.: Introducing trails: a tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development. J. Syst. Softw. **144**, 342–355 (2018)
151. Wu, T., Xie, N., Blackhurst, J.: Design and implementation of a distributed information system for collaborative product development. J. Comput. Inf. Sci. Eng. **4**(4), 281–293 (2004)
152. Yang, W., Xie, S., Ai, Q., Zhou, Z.: Recent development on product modelling: a review. Int. J. Prod. Res. **46**(21), 6055–6085 (2008)
153. Zaletelj, V., Hozdić, E., Butala, P., et al.: A foundational ontology for the modelling of manufacturing systems. Adv. Eng. Inform. **38**, 129–141 (2018)
154. Zhang, X., Wang, K., Wang, H., Xie, Z.: Integrated information modeling of engineering digital prototyping for satellite design. In: IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 1314–1318. IEEE (2013)



Weslley Torres is a PhD Student at Eindhoven University of Technology, The Netherlands. His M.Sc. degree is in Computer Science from the Center of Informatics (CIn) at Federal University of Pernambuco, Brazil. He is currently researching effective ways of managing models from different engineering domains. His current research activities are on software engineering, model-driven engineering, and model management.

Mark van den Brand is a full professor of Software Engineering and Technology in the Department of Mathematics and Computer Science, and a visiting professor at Royal Holloway, University of London. His current research activities are on model driven engineering, domain-specific languages, meta-modeling, model management, digital twins, and automotive software engineering. He edited in 2019 a Springer book on Automotive Systems and Software Engineering. His research is industry inspired; he works with most of the high-tech companies in the Eindhoven (The Netherlands) region. He has been an invited lecturer and keynote speaker at various conferences, workshops, and doctoral schools. He was and is member of PCs on workshops and conferences related to software engineering language engineering, rewriting, reverse engineering, and software maintenance. He initiated the special issues of Science of Computer Programming devoted to academic software development (Experimental Software and Toolkits) and since 2007 has been guest editor of six of these. He is on the editorial board of the journals Science of Computer Programming, Open Computer Science and Computer Languages (COLA). He is Editor-in-Chief of the Journal on Automotive Software Engineering. He is associate Editor-in-Chief of the Software Section of the Science of Computer Programming. He is deputy Editor-in-Chief of platinum open access journal JOT.



Alexander Serebrenik (PhD KU Leuven, Belgium 2003) is a Full Professor social software engineering at Eindhoven University of Technology, The Netherlands. His goal is to facilitate evolution of software by taking into account social aspects of software development. His work tends to involve theories and methods both from within computer science (e.g., theory of socio-technical coordination; methods from natural language processing, machine learning) and from outside of computer science (e.g., organizational psychology). He has co-authored a book "Evolving Software Systems" (Springer Verlag, 2014), and more than 100 scientific papers and articles. He has won several distinguished paper and distinguished review awards.