Towards Formalizing and Relating Different Notions of Consistency in Cyber-Physical Systems Engineering

Kevin Feichtinger
Karlsruhe Institute of Technology
Karlsruhe, Germany
kevin.feichtinger@kit.edu

Karl Kegel
Dresden University of Technology
Dresden, Germany
karl.kegel@tu-dresden.de

Romain Pascual
Karlsruhe Institute of Technology
Karlsruhe, Germany
romain.pascual@kit.edu

Uwe Aßmann
Dresden University of Technology
Dresden, Germany
uwe.assmann@tu-dresden.de

Bernhard Beckert
Karlsruhe Institute of Technology
Karlsruhe, Germany
beckert@kit.edu

Ralf Reussner
Karlsruhe Institute of Technology
Karlsruhe, Germany
ralf.reussner@kit.edu

Abstract

Cyber-Physical Systems (CPSs) are highly complex systems integrating computational and physical processes and consist of many interdependent and composed parts. Engineers from different domains, e.g., mechanical, electrical, and software engineering, cooperate to develop and deploy new CPSs. Engineers often only work on task-specific artifacts and models to reduce the complexity of the overall CPS. These models and artifacts form different views on the CPS, which must be kept consistent to enable development and system analysis. However, this inter-view consistency management lacks tool support and remains a tedious manual task. Hence, this results in late integration risks and may even lead to failed products during deployment. To manage consistency between models, artifacts, and views sufficiently, we need an understanding of the available notions of consistency and their properties. We need to identify, classify, formalize, and relate notions of consistency from different domains to derive a common definition for a consistency-aware, view-based development process for CPSs. This vision paper presents a set of existing notions of consistency we can build on and outline our vision towards consistency-aware CPS engineering.

CCS Concepts

• **Software and its engineering → Consistency**.

Keywords

cyber-physical systems engineering, notions of consistency, vision

ACM Reference Format:
Kevin Feichtinger, Karl Kegel, Romain Pascual, Uwe Aßmann, Bernhard Beckert, and Ralf Reussner. 2024. Towards Formalizing and Relating Different Notions of Consistency in Cyber-Physical Systems Engineering. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3652620.3688565

MODELS Companion '24, September 22–27, 2024, Linz, Austria

ACM ISBN 979-8-4007-0622-6/24/09
https://doi.org/10.1145/3652620.3688565

1 Introduction

Cyber-Physical Systems (CPSs) are systems that combine hardware and software components to integrate computational and physical processes [22]. This term summarizes many highly complex systems, from cars, trains, and aircraft to production and smart home systems. Engineers from different domains, e.g., mechanical, electrical, and software engineering, cooperate to fulfill various design, analysis, and quality assurance tasks to design and deploy new CPSs [9]. Engineers use specialized processes and tools during development to counter the high complexity of interdependent and composed parts to complete specific system synthesis and analysis tasks. These processes and tools are often domain-specific and capture domain and task-specific models, artifacts, and data of the CPS, increasing the overall complexity of CPS engineering [9]. As a result, developers of one discipline may not understand the impact of design decisions made by developers of another discipline. Thus, as the various models are related, the risk of inconsistencies among the different engineering artifacts increases, hindering the system's production, meaningful quality analyses or release.

Currently, researchers investigate the adoption of view-based approaches and tools to manage this complexity [26]. Engineers often only work on a smaller set of artifacts and models to complete their tasks. Hence, each artifact or model provides a task-specific view on the same CPS. During development, these task-specific and domain-specific views are usually not independent and have many dependencies among them. For instance, a component in one model depends on a module in another model to work correctly, or the behavior described in one model must fulfill another model's behavior specification. Different views are often involved when designing, i.e., synthesizing, analyzing, testing, and verifying a CPS. Thus, these views must be kept consistent to a certain extent during development to ensure various requirements and avoid delays caused by inconsistencies [1].

Preserving consistency among multiple models, artifacts, and views is difficult because consistency is often defined informally. Existing works on views, view types, meta-modeling, and the theory of a Single Underlying Model (SUM) [3, 12, 27, 30] are a good basis for dealing with the complexity and view consistency in the design of CPS. A SUM captures the portrayed information of all views on a system in a centralized full-domain-spanning model [3, 7, 8]. Similar ideas have been applied successfully in other domains. For instance, in database systems, it is common to have several views on

a single underlying database. Inconsistencies between the views are known as data anomalies [16], which are solved via data integration and repair [4] techniques. If the database views are consistent with the single underlying database, they are also consistent with each other. Transferring consistency management of views from database systems to CPSs remains challenging, as CPSs dependencies semantically go beyond key relationships between database tables. Also, previous studies showed that approaches utilizing centralized full-domain-spanning models, such as a SUM, do not scale to the necessary number of models and views in CPS engineering [24].

Therefore, we argue that the Virtual Single Underlying Model (V-SUM) approach [19], where several dependent models are coupled via consistency preservation rules, is better suited for cross-domain engineering processes and tools in CPS engineering [9]. A V-SUM allows engineers to develop CPS in different views and design spaces. Hence, the models within the V-SUM are not edited directly but through their respective views. To ensure that the models in a V-SUM remain free of contradictions, the V-SUM assists engineers in managing view consistency using consistency specifications [28]. These specifications are executed automatically and request user interaction where appropriate [19].

Currently, normative rule-based consistency specifications [28] are used for V-SUM construction [19, 21]. However, other notions of consistency and methods for specifying consistency relations exist in model-based engineering [6, 10, 19, 23]. Unfortunately, a comprehensive overview of the available notions of consistency, their formalization, and their properties is missing. Yet, such an overview and understanding is necessary to develop a single framework of consistency notions that can also be integrated into a view-based development process for CPSs. Thus, with our research, we aim to answer the following research questions:

RQ1 How can the different paradigms for specifying consistency relations be combined in a single formal framework of consistency notions?

RQ2 How can such a framework of consistency notions be applied in a V-SUM to enable consistency-aware, view-based development of CPS?

While projects in consistency-aware CPS engineering may start with the existing rule-based normative notion of consistency, we need to investigate other possible notions, such as those based on model theory and denotational semantics, where consistency amounts to the existence of an implementation realizing all specifications [6]. Thus, we first aim to comprehensively classify different notions of consistency and their properties. We then want to formalize and relate these different notions of consistency to enable an appropriate selection of consistency notions. Finally, we aim to implement our formal consistency framework with Vitruvius [19] and investigate its effects on achieving the needed consistency in CPS engineering.

In the remainder of the paper, we first motivate the need for consistency-aware CPS engineering using an example, highlighting possible inconsistencies, in Section 2. In Section 3, we describe consistency as a multidimensional problem and outline our first attempt at classifying consistency notions. We present our vision towards consistency-aware CPS engineering and our research agenda in Section 4. We conclude the paper and outline immediate next steps towards our goal in Section 5.

2 Illustrative Example

We use the engineering process of an electric car's braking system to illustrate and motivate different kinds of consistency relations in CPSs. We present an abstract representation of the braking system, as we cannot address all potential models, artifacts, parts, dependencies, and workflows of the different domains involved in the example. Figure 1 shows the abstracted view of the braking system.

The main brake of the electric car is a recuperating electrical brake system. This brake is coupled with a secondary mechanical brake system for safety reasons. Due to the multiplicity, complexity, and variety of subsystems, developing such a brake system requires the collaboration of multiple engineers from various disciplines [9]. Each engineer is responsible for a specific part of the braking system, developing their models or artifacts. We consider the following models, representing different views of the overall brake system. A mechanical engineer designs a Computer-aided design (CAD) model of the brake, including the physical dimensions, assembly, and materials used. An electrical engineer works on the E/E architecture, specifying the brake controller and its connection to the car's overall E/E topology, as well as cable dimensions and wiring layout. A software engineer programs the brake controller using a model-driven approach based on a control automaton for formal verification. The brake controller streams sensor data to the car's Central Monitoring Unit (CMU). In the later stages of development, a team of mechanics builds physical prototypes of the brake system. These prototypes are used for testing purposes according to the specifications of the Q/A expert. Concurrently, the management team advertises the new car model using renderings from the CAD models and functional claims from the test results. The multidisciplinary nature of the involved models, artifacts, and views makes keeping them consistent during development challenging.

With this simplified example, we illustrate multiple consistency relations. For instance, the Electrical Brake Controller (EBC) and the CMU share interfaces for communication that must be kept consistent. Both parts are developed independently, but changing the CMU's interface data format requires adapting the brake controller's sent sensor values and vice versa. Similarly, configuration values in the EBC determine the maximum brake force and, thus, the amount of electrical power the recuperation mechanism provides. A change in these values by the software engineer requires the electrical engineer to reevaluate the E/E architecture to ensure that the wires withstand the imposed electrical currents. These examples correspond to inter-model (and inter-domain) consistency relations, whose specifications should allow propagating changes within (the set of) artifacts and models involved to ensure consistency.

Models and artifacts may also form hierarchies that must be consistent. For instance, the CAD model of the electrical brake is part of the CAD model of the overall car, maintained in parallel by other engineers. A change in the brake system dimensions may corrupt this top-level assembly model. If this happens, engineering teams of related car parts need to consult on how to recover a sound assembly based on the new dimensions of the brake, i.e., to restore consistency between the various CAD models.

The renderings of the car's CAD models are used for communication by the management team. While the latest update is desirable

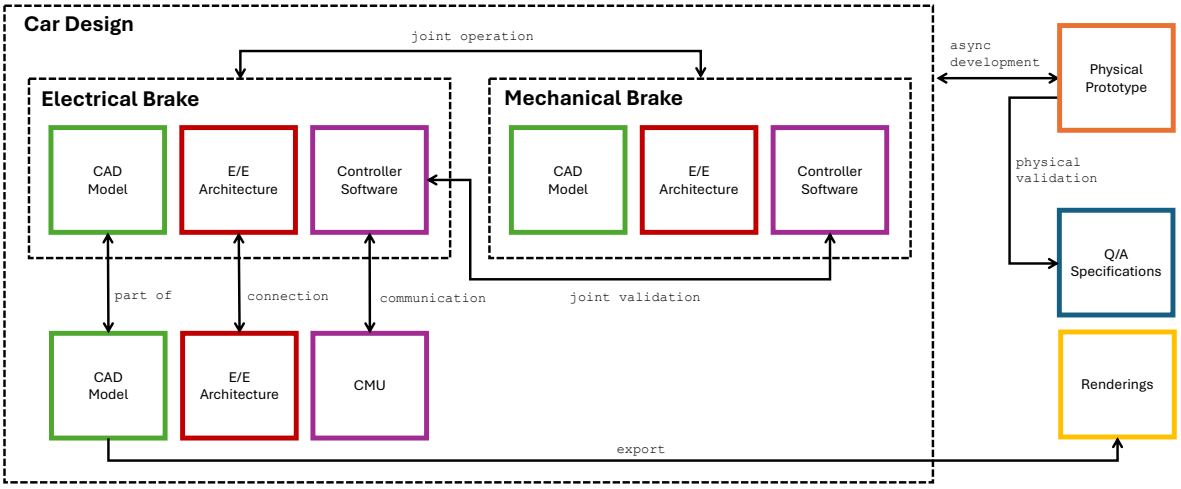

Figure 1: An electrical car braking system example illustrates the need for consistency-aware CPS engineering.

for communication, the car's renderings are usually only recomputed after significant modifications. As a result, inconsistencies might temporarily exist between the CAD models designed by engineers and the derived rendering used by the management team. While the previous examples were only between the structural properties of the models, consistency might also relate to the behavioral properties of the models. For instance, if the electrical brake reaches a certain threshold temperature, the mechanical brake must be activated as support. In this example, the assertion can only be verified by analyzing the possible traces describing the system's temporal evolution. This example illustrates that consistency also arises at the semantic level and not only at the syntactic one.

The shown consistency relations are only described abstractly, omitting most technical details. However, it already shows that consistency relations span different kinds of models, artifacts, and views, capturing different types of information. These consistency relations need to be unified to ensure that CPSs can be effectively developed. We argue that solving each consistency problem independently, without a common formal framework, is not feasible. Therefore, we aim to formalize and relate these multiple notions of consistency and their properties, ultimately deriving a unified definition of consistency for consistency-aware CPS engineering.

3 Multidimensional Consistency

The models within the V-SUM [19] are not edited directly but indirectly via the views engineers interact with. These models describe the system under development and must be free of contradictions to ensure its integrity. Thus, the primary goal is to ensure or preserve the consistency of a set of models within the V-SUM [19].

A view can be understood as a projection from certain models of the V-SUM. When a change is made in a view, it is first propagated to the models from which the view is derived. Then, additional modifications may need to be propagated across the models of the V-SUM to preserve their consistency. Since the engineers only edit views, the propagation of changes within the V-SUM must be automated. In particular, we need to know which models require modification and what specific changes should be applied. One plausible solution is to establish relations between models, such that whenever a model is modified, its related models can be kept in relation. Therefore, we are formally encoding consistency as a relation between models, thus dealing with intermodel consistency rather than intra-model consistency. Then, for a model m_1 of a metamodel M_1, consistency describe which other models m_2, m_3, …, m_n are allowed for the metamodels M_2, M_3, …, M_n.

Given this definition of consistency, we aim to develop a solution that can check and, if necessary, repair consistency when dealing with many large models. A prerequisite to building such an automated solution is a thorough understanding of the practicalities of a relational notion of consistency. When we claimed that we wanted to deal with intermodel consistency rather than intramodel consistency, we already hinted at the multiplicity of meanings attached to the notion of consistency. For instance, the V-SUM construction [19] rely on normative rule-based consistency specifications [28], which we can further classify as the use of syntactic qualitative binary normative consistency relations. More precisely, we argue that consistency is a multidimensional notion based on (at least) the following dichotomies: (1) syntax vs. semantics, (2) qualitative vs. quantitative, (3) binary vs. multi-ary, and (4) normative vs. descriptive.

Consistency can relate model elements, making syntactic connections between the models via their structures or values derived from computations based on the models. For instance, interface formats illustrated in Section 2 with the interfaces shared by the EBC and the CMU or the configuration and numerical values stored in the EBC yield syntactic consistency. Conversely, the consistency derived via refinement or bisimulation properties on the automata describing the mechanical and electrical brake systems is semantic. A more formal view on the relation between (syntactical) model consistency and semantics-based consistency is discussed in [25].

When consistency is viewed as a relation, then models are either consistent or inconsistent [1]. Thus, relational consistency is qualitative, i.e., a binary measure where consistency is either true or false. Then consistency consists of a quality that must be maintained, such as in the consistency problems from Section 2. However, at least temporarily living with consistency is unavoidable to prevent information loss [1, 11]. Therefore, qualitative consistency is desirable for repair or recovery from inconsistencies. Quantitative consistency allows for measuring and tracking inconsistencies, providing a spectrum or degree to which a set of models are consistent [17]. Quantitative consistency assumes that a pair of models can be gradually consistent rather than just consistent or inconsistent. Gradual consistency can be based on the number of constraint violations. Thus, consistency decreasing, maintaining, and improving operations can be classified [20].

Approaches based on model transformations such as triple graph grammars [2, 14, 15] or bidirectional transformation [5, 13] consider consistency as a binary relation and consistency needs to be maintained for pairs of models. Klare and Gleitze [18] proposed the commonalities language to handle multi-ary relations between models, formalized as comprehensive systems [29]. Consistency is normative when established as a standard without preexisting references, essentially *built out of thin air*, and claimed as a definition. However, consistency can also be related to or built upon an already existing normative definition. In this case, consistency is descriptive as it describes how consistency is achieved based on predefined standards.

While some dimensions might be missing to fully grasp the complexity of the various notions of consistency, we can already claim that we need a shift from syntactic qualitative binary normative consistency. For instance, while syntactic consistency is suited for a tool that needs to edit models efficiently, only semantic consistency can allow formal reasoning and remains the key to proving non-violation of consistency. Semantics refers to information not stored in the model but derived from it, which provides additional insights that cannot be grasped only via syntactical manipulations. Similarly, quantitative consistency acknowledges that consistency might not be absolute, thus allowing for repair based on increasing consistency. Real-world challenges in CPS engineering often require consistency to include elements from more than two models. For example, an Electronic Control Unit (ECU) has a logical view, a functional view, an electronic layout view, and a mechanical construction view for the electronic circuit. These views need to be kept consistent, as a logical change can lead to a different circuit layout, which is also constrained by possible physical dimensions of the ECU. In a sense, multi-ary consistency reduces circular dependencies induced by iterative propagation inherent to binary consistency. Only descriptive consistency enables the comparison of various definitions of consistency. Finally, some semantics-based model-theoretic foundations of consistency would allow for a shift towards descriptive specifications, enabling the study of consistency through the classical notion of correctness as behavior refinement.

As a result, we obtain a hypercube of consistency, where each dimension describes such a dichotomy. Each dichotomy yields an orthogonal dimension such that each alternative can be studied independently. Figure 2 shows a restriction of the hypercube to the three dimensions: normative-descriptive, syntax-semantics, and qualitative-quantitative. The cube illustrates the goal of shifting from syntactic qualitative normative consistency to semantic quantitative descriptive consistency. Each edge of the cube can be studied and solved separately, meaning that the green arrow can, in practice, be obtained through various walks on the cube.

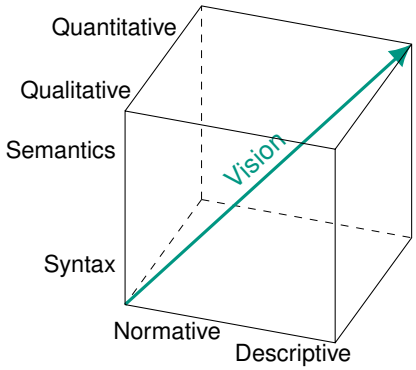

Figure 2: The hypercube of consistency notions

4 Vision of consistency-aware Cyber-Physical Systems engineering

Many software engineering solutions, e.g., model analysis, modeling version and variants, etc., can also be applied to meta-model-based artifacts of non-software artifacts. We envision consistency management for CPS engineering to be built on existing techniques from Model-Driven Software Engineering (MDSE), such as metamodeling, model transformations, and mapping semantic domains from models using formal methods. This transfer and application of MDSE methods to non-software artifacts form a new opportunity for both the discipline of software engineering and systems engineering. Hence, consistency-aware CPS engineering could be the basis of a new form of systems engineering where adopted software engineering-inspired methods are integrated and form a new generation of systems engineering methods, i.e., *Advanced Systems Engineering* [1].

Research Agenda: We must first understand the various approaches to defining consistency in order to design consistency management solutions. For instance, most approaches to consistency management approaches assume rule-based qualitative binary consistency relations. However, incorporating model-theoretic foundations (in the logical meaning of the word model) into the notion of consistency would shift normative specifications to descriptive ones and, therefore, relate models through their meaning, enabling classical correctness methods based on behavior refinement. Investigating the various notions of consistency means formalizing them to describe their properties and differences, which should lead to a comprehensive classification of the various notions.

While building this classification, we also plan to investigate some of its components further. In particular, semantic-based approaches to consistency, being very valuable for studying the correctness of model transformations, should be made intuitive and compositional. One goal consists of the possibility of providing

easy-to-understand counter-examples as witnesses for inconsistencies. We expect denotational semantics to improve the state of the art for reasoning about consistency. Model-theoretic semantics should become a first-class citizen in the modeling approach to ensure that consistency analyses are proven sound and complete. We plan to investigate if correct consistency preservation rules can be derived from model-theoretic definitions of consistency.

Similarly, when consistency is considered a qualitative property, then models are either consistent or inconsistent. However, when models are inconsistent, we need to quantify how inconsistent they are. Measuring inconsistency in terms of inconsistent syntactical or semantical units may not prove sufficient. Inconsistencies must be represented as a form of technical debt, allowing an a priori estimation of the difficulty of repair operations. To this end, we plan to investigate metrics assessing inconsistencies between models.

5 Conclusion

This paper outlines our vision towards consistency-aware Cyber-Physical Systems (CPSs) engineering. Engineers of different domains work on their specific domain and task-specific models and artifacts. We understand such models and artifacts as views on the overall CPSs, which must be kept consistent to ensure various requirements and avoid delays caused by inconsistencies.

We work towards a view-based development process for CPSs. We utilize the existing Virtual Single Underlying Model (V-SUM) approach, where the different models and artifacts are related via consistency relations. However, many different notions of consistency are available, and there is no understanding of their consequences for CPS engineering. Thus, we aim to establish a comprehensive, multidimensional classification of consistency notions, formalize and relate them and their properties, implement our formal consistency framework within a V-SUM, and investigate its effects on achieving consistency in CPS engineering. We envision that such a framework can offer a shared understanding of the results of currently separated communities, including software verification, model-driven systems engineering and formal semantics.

We presented a research agenda toward formalizing and relating different notions of consistency in CPS engineering. For immediate next steps, exploring existing properties of already established notions of consistency remains important. Thus, we first need to find existing notions of consistency by surveying the literature.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – CRC 1608 – 501798263.

References

[1] Albert Albers, Anne Koziolek, Thomas Alexander Völk, Monika Klippert, Felix Pfaff, Robert Stolpmann, and Stefan Eric Schwarz. 2024. Identification of Inconsistencies in Agile CPS Engineering with Formula Student. In *ISPIM Innovation Conference*.

[2] Anthony Anjorin, Sebastian Rose, Frederik Deckwerth, and Andy Schürr. 2014. Efficient Model Synchronization with View Triple Graph Grammars. In *Modelling Foundations and Applications (Lecture Notes in Computer Science)*. Springer International Publishing, Cham, 1–17.

[3] Colin Atkinson, Dietmar Stoll, and Philipp Bostan. 2010. Orthographic Software Modeling: A Practical Approach to View-Based Development. In *Evaluation of Novel Approaches to Software Engineering*. Springer, Berlin, Heidelberg.

[4] Leopoldo Bertossi. 2011. *Database Repairs and Consistent Query Answering*. Morgan & Claypool Publishers.

[5] Aaron Bohannon, J. Nathan Foster, Benjamin C. Pierce, Alexandre Pilkiewicz, and Alan Schmitt. 2008. Boomerang: resourceful lenses for string data. In *Proceedings of the 35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '08)*. Association for Computing Machinery.

[6] H. Bowman, M.W.A. Steen, E.A. Boiten, and J. Derrick. 2002. A Formal Framework for Viewpoint Consistency. *Formal Methods in System Design* 21, 2 (2002).

[7] Manfred Broy, Franz Huber, and Bernhard Schätz. 1999. AutoFocus– Ein Werkzeugprototyp zur Entwicklung eingebetteter Systeme. *Informatik Forschung und Entwicklung* 14, 3 (1999).

[8] Manfred Broy and Ralf Reussner. 2010. Architectural Concepts in Programming Languages. *Computer* 43, 10 (2010), 88–91.

[9] Kevin Feichtinger, Kristof Meixner, Felix Rinker, István Koren, Holger Eichelberger, Tonja Heinemann, Jörg Holtmann, Marco Konersmann, Judith Michael, Eva-Maria Neumann, Jérôme Pfeiffer, Rick Rabiser, Matthias Riebisch, and Klaus Schmid. 2022. Industry Voices on Software Engineering Challenges in Cyber-Physical Production Systems Engineering. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 1–8.

[10] Anthony Finkelstein. 2000. A Foolish Consistency: Technical Challenges in Consistency Management. In *Database and Expert Systems Applications*. Springer.

[11] A.C.W. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. 1994. Inconsistency handling in multiperspective specifications. *IEEE Transactions on Software Engineering* 20, 8 (Aug. 1994), 569–578.

[12] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. 1992. Viewpoints: a framework for integrating multiple perspectives in system development. *Int. J. Softw. Eng. Knowl. Eng.* (1992), 31–57.

[13] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. 2007. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems* 29, 3 (2007), 17–es.

[14] Lars Fritsche, Jens Kosiol, Adrian Möller, Andy Schürr, and Gabriele Taentzer. 2020. A precedence-driven approach for concurrent model synchronization scenarios using triple graph grammars. In *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2020)*. Association for Computing Machinery, New York, NY, USA, 39–55.

[15] Holger Giese, Stephan Hildebrandt, and Stefan Neumann. 2010. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. In *Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*. Springer, Berlin, Heidelberg, 555–579.

[16] Ihab F. Ilyas and Xu Chu. 2015. Trends in Cleaning Relational Data: Consistency and Deduplication. *Foundations and Trends® in Databases* 5, 4 (2015), 281–393.

[17] Karl Kegel, Sebastian Götz, Ronny Marx, and Uwe Aßmann. 2024. A Variance-Based Drift Metric for Inconsistency Estimation in Model Variant Sets. In *20th European Conference on Modelling Foundations and Applications*. JOT. In press.

[18] Heiko Klare and Joshua Gleitze. 2019. Commonalities for Preserving Consistency of Multiple Models. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 371–378.

[19] Heiko Klare, Max E. Kramer, Michael Langhammer, Dominik Werle, Erik Burger, and Ralf Reussner. 2021. Enabling consistency in view-based system development — The Vitruvius approach. *Journal of Systems and Software* 171 (2021), 110815.

[20] Jens Kosiol, Daniel Strüber, Gabriele Taentzer, and Steffen Zschaler. 2022. Sustaining and improving graduated graph consistency: A static analysis of graph transformations. *Science of Computer Programming* 214 (2022), 102729.

[21] Max Emanuel Kramer. 2019. *Specification languages for preserving consistency between models of different languages*. Vol. 24. KIT Scientific Publishing.

[22] Edward A. Lee. 2010. CPS foundations. In *Proceedings of the 47th Design Automation Conference (DAC '10)*. Association for Computing Machinery, 737–742.

[23] Francisco J. Lucas, Fernando Molina, and Ambrosio Toval. 2009. A systematic review of UML model consistency management. *IST* (2009).

[24] K.D. Muller-Glaser, G. Frick, E. Sax, and M. Kuhl. 2004. Multiparadigm modeling in embedded systems design. *IEEE Trans. Control Syst. Technol.* (2004), 279–292.

[25] Romain Pascual, Bernhard Beckert, Mattias Ulbrich, Michael Kirsten, and Wolfram Pfeifer. 2024. Formal Foundations of Consistency in Model-Driven Development. In *ISoLA*. to appear.

[26] J. Schallow, K. Magenheimer, J. Deuse, and G. Reinhart. 2012. Application Protocols for Standardising of Processes and Data in Digital Manufacturing. In *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, Berlin, Heidelberg, 648–653.

[27] Thomas Stahl, Markus Völter, and Krzysztof Czarnecki. 2006. *Model-driven software development: technology, engineering, management*. John Wiley & Sons, Inc.

[28] Perdita Stevens. 2020. Maintaining consistency in networks of models: bidirectional transformations in the large. *Software and Systems Modeling* 19 (2020).

[29] Patrick Stünkel, Harald König, Yngve Lamo, and Adrian Rutle. 2021. Comprehensive Systems: A formal foundation for Multi-Model Consistency Management. *Formal Aspects of Computing* 33, 6 (Dec. 2021), 1067–1114.

[30] M. Völter, S Benz, C Dietrich, B Engelmann, M Helander, LCL Kats, E Visser, and GH Wachsmuth. 2013. *DSL Engineering - Designing, implementing and using domain-specific languages*. M Volter / DSLBook.org. NEO.