The Journal of Systems and Software 242 (2026) 113037

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Steel does not Sprint (or does it?): Enabling agility for cyber–physical systems through consistency preservation ☆, ☆☆

Albert Albers [a], Anne Koziolek [b], Thomas Alexander Voelk [a,*], Razieh Dehghani [b], Thomas Weber [b], Tilman Pfersdorff [a], Frederik Beck [a], Maximilian Beck [c], Katharina Bause [a], Tobias Düser [a]

[a] IPEK - Institute of Product Engineering, Karlsruhe Institute of Technology, Kaiserstr. 10, Karlsruhe, 76131, Baden-Württemberg, Germany
[b] KASTEL - Institute of Information Security and Dependability, Am Fasanengarten 5, Karlsruhe, 76131, Baden-Württemberg, Germany
[c] Institut für Technik der Informationsverarbeitung (ITIV), Karlsruhe Institute of Technology, Engesserstr. 5, Karlsruhe, 76131, Baden-Württemberg, Germany

ARTICLE INFO

Keywords:
Cyber–physical systems
Consistency preservation
Agile adaption
Systems engineering
Inconsistency management

ABSTRACT

Agile development promises responsiveness and faster feedback, yet its applicability beyond the development of software to cyber–physical systems (CPS) remains limited by physical constraints, cross-domain dependencies, and the need for planning stability. While previous work frequently suggests that consistency mechanisms support agility in such settings, this relationship has largely remained implicit and insufficiently explained. This paper explicates how consistency preservation and inconsistency management influence agility in CPS engineering. Drawing on a design science research approach, we derive an interdisciplinary network of influencing factors that makes explicit the dependencies between consistency mechanisms, organizational and technical conditions, and key agility outcomes such as responsiveness, transparency, and the realizability of development increments. To illustrate the explanatory power of this model, we apply model-based consistency preservation to an industry-motivated inconsistency situation from the development of a hybrid braking system using consistency preservation tools. By systematically maintaining the alignment of interdependent development artifacts across engineering disciplines such as software, electrical, and mechanical engineering (reflected in artifacts like SysML diagrams), cross-disciplinary dependencies in CPS development can be explicitly managed. This enables agile practices to be applied more effectively in physically constrained and interdisciplinary development environments. The case demonstrates how increased transparency of the current consistency state and automated change propagation can support faster feedback cycles and cross-domain coordination without undermining necessary planning stability. The results show that consistency preservation is neither an obstacle to agility nor a universal remedy, but a context-dependent enabling mechanism whose effectiveness depends on domain maturity, recurring dependencies, and organizational context.

1. Introduction

Agility in development projects is not a phenomenon confined to software engineering. Classic engineering domains such as mechanical engineering have long investigated agile or agile-like practices as responses to product development environments characterized by volatility, uncertainty, complexity, and ambiguity (VUCA) (Atzberger et al., 2019). In these contexts, agility is understood less as a specific method and more as a conceptual capability that enables development organizations to adapt to rapidly changing and increasingly interdependent conditions. One of the biggest threats to quickly developing such systems is complexity (Dumitrescu et al., 2021): Besides the internal complexity of advanced mechatronic systems and cyber–physical systems (CPS), there is additionally the external complexity in development organization of, for example, the tight integration of multiple disciplines, technologies, and abstraction levels. This complexity is further compounded by ambiguity, because design decisions in one domain may affect not only the solution space but also the decision-making capacity of other domains. Local optima achieved within individual disciplines may therefore obstruct the global optimum of the system as a whole. While agility seems to be one answer to this challenge, transferring agility into hardware-intensive and cross-domain

☆ This article is part of a Special issue entitled: 'Agile Reloaded' published in The Journal of Systems & Software.
☆☆ Editor: Raffaela Mirandola.
* Corresponding author.
E-mail address: thomas.voelk@kit.edu (T.A. Voelk).

https://doi.org/10.1016/j.jss.2026.113037
Received 15 December 2025; Received in revised form 2 July 2026; Accepted 13 July 2026
Available online 18 July 2026

engineering contexts such as CPS development has proven challenging (Heimicke, 2025). It seems that when developing with steel (or less provocatively, with hardware in general), engineering has difficulties deploying agility.

That brings the concept of consistency into focus. Consistency describes a system state in which development artifacts with overlapping semantics do not contradict one another (Spanoudakis and Zisman, 2001a; Albers and Lohmeyer, 2012). While consistency preservation is manageable for simple products, consistency becomes harder to achieve as system complexity increase and as coupling between domains becomes tighter. To sustain reactive and agile development under these conditions, approaches for preserving and managing consistency are required (Reussner et al., 2023). These approaches must encompass formalized techniques supported by tools and processes and methods that effectively support engineers in practice (Cicchetti, 2016).

Traditionally, consistency management and agile development have been treated as separate concerns. Nevertheless, there is good reason to consider the two phenomena, instead, as one joined phenomenon. Consistency management seeks to ensure correctness by preventing or resolving inconsistencies in development artifacts and processes. Yet preserving consistency has become increasingly difficult not only because systems grow more complex but also because development processes involve larger and more heterogeneous collaborations across organizational and disciplinary boundaries (Reussner et al., 2023; Dumitrescu et al., 2021). Existing consistency management solutions are therefore typically restricted to specific application contexts and exhibit notable limitations even within those contexts (Hebig et al., 2016; Rose et al., 2009). The same applies to agility. Agile methods are applied to cope with problems raised by complexity in engineered systems and engineering processes by rapidly increasing the maturity of developed systems (Dumitrescu et al., 2021; Weiss et al., 2023). However, agile methods are likewise limited by organizational complexity and cross-disciplinary challenges (Weiss et al., 2023; Atzberger and Paetzold, 2019; Michalides et al., 2023b).

These parallel challenges indicate a convergence between the problems faced by agility and those faced by consistency management. Existing literature frequently asserts (on the basis of expert experience or isolated empirical observations) that preserving consistency may support agility in cross-domain system development (Cicchetti, 2016; Reussner et al., 2023; Feldmann and Vogel-Heuser, 2024). The missing element here is an explicit theoretical explanation of how and why consistency preservation can contribute to agility, as a phenomenon, but also as an engineering practice; further, it is also important to learn which influencing factors are responsible for this lack of theoretical sophistication.

In this paper, it is our goal to provide an in-depth description of consistency and agility as a jointly understood phenomenon. We provide this insight by first discussing related work on consistency practice and how it affects agility in Section 2. Based on a interdisciplinary derived research methodology explained in Section 3, we present two major results in Section 4: an initial explanation of the influencing factors on the phenomenon of consistency in agile development, as well as a case study from industrial practice to highlight the applicability of the given description approach on applying consistency preservation in a CPS use case. The findings are discussed in Section 5.

2. State of research and related work

Here we present related work on maintaining consistency in multi-domain engineering 2.1, on consistency preservation 2.2, and on the potential for application of consistency preservation in agile development 2.3.

To begin, we clarify the meaning of consistency, especially for the model-driven community, then discuss present approaches to deal with consistency in different engineering disciplines, describe previous notions of the phenomenon of consistency and agility, and finally conclude with a discussion of related work already (partially) addressing the given problem. Consistency can be defined considering the negative and positive sides of inconsistency (Spanoudakis and Zisman, 2001b). Some researchers focus on aligning different representations of reality, referred to as multi-model consistency (Klare, 2018; Klare et al., 2021), while other researchers focus on the alignment between models and the real-world phenomena represented. In the latter research direction, a representation denotes any artifact that captures aspects of reality, including documentation. However, regardless of the research focus, there may always be multiple perspectives or views originating from different stakeholder viewpoints; therefore, model-to-model and model-to-reality consistency can only be maintained when all views of the stakeholders remain mutually consistent. For this reason, it is equally important that we clarify the meaning of view. A view can be understood as a customized representation that comprises selected parts of a single model or a combination of projections derived from multiple models (Atkinson et al., 2008). Views are constructed by extracting and composing relevant elements from the projections to address specific stakeholder concerns. That technique of views construction demands that both the underlying models and also the constructed views and their represented reality be kept consistent (Klare et al., 2021).

2.1. Consistency preservation and inconsistency management in software and hardware engineering practice

Two practices in particular are used to achieve consistency: the first is to ensure consistency by continuously preserving it, and the second is to manage inconsistency such that a consistent state can be recovered at any moment in the development. For example, architectural documentation can be built which helps to prevent inconsistency between various documents (Clements et al., 2003). Hence, consistency between views is as relevant to software engineering processes as it is to architectural documentation (Clements et al., 2003) and again, as it is to CPS development overall (Bhave et al., 2011).

Consistency preservation has been conceptualized from multiple perspectives in the literature. In the context of Model-Driven Development, consistency has received particular attention because models originating from different engineering disciplines represent only partial knowledge, some of which is usually shared. These models are typically sufficient to describe the whole system in isolation; therefore, maintaining consistency among them requires domain-specific knowledge that spans disciplinary boundaries. For example, Jadoon et al. (2025) define consistency with respect to property preservation, but Pascual et al. (2024) emphasize semantic agreement concerning system-level goals. Other approaches interpret consistency in terms of compliance with predefined constraints that must not be violated across interrelated artifacts (Braga et al., 2014).

On the other hand, inconsistency management has emerged as a line of research which is complementary to consistency preservation. In inconsistency management, temporary inconsistencies are not prevented but instead tolerated within defined limits, specifically, as long as the inconsistencies remain diagnosable and manageable throughout development. Jongeling et al. (2022), for example, argue for short feedback cycles in which violations of predefined relations are continuously detected and mitigated, often requiring human-driven decision-making. The handling of incomplete changes leading to temporarily accepted inconsistencies is likewise recognized in various domains (Kretschmer et al., 2021). Inside of the general area of inconsistency management, repair constitutes an important subarea. Macedo et al. (2016) distinguish between delta-based and state-based repair techniques, and Klare et al. (2021) propose a delta-based framework that identifies changes between model states and propagates necessary updates to restore consistency. More recent work also highlights the need to tolerate inconsistencies under uncertainty, so that instead of enforcing a single corrective action, the researchers acknowledge how multiple valid resolution strategies may coexist (Weidmann et al., 2021).

2.2. Previous notions of the phenomenon of consistency and agility

Although the Agile Manifesto (Beck et al., 2001) does not explicitly address consistency, previous work relates agility to different forms of consistency in development processes. Three major categories of work can be identified in this regard: (1) consistency between agile requirements specifications and system models, e.g., the maintenance of consistency between user stories and system representations (Wanderley et al., 2014); (2) rule-based consistency between development artifacts, e.g., architectural documentation which supports traceability through correspondence relations using, for example, extensions to the ISO/IEC/IEEE 42010 metamodel (Wanderley et al., 2014); and (3) consistency in development routines, i.e., the reliable application of agile methods within teams (Clear and Clear, 2018). Although none of this work explicitly targets Model-Driven Development, their use of abstraction may share a conceptual overlap with agile and model-driven paradigms; in particular, development artifacts produced within agile processes can be interpreted as models at different levels of abstraction (Bézivin, 2005).

The relationship between agile and model-driven abstraction becomes particularly relevant in CPS engineering, where heterogeneous artifacts, tools, and disciplinary viewpoints increase the likelihood of contradictions between evolving system representations (Feldmann and Vogel-Heuser, 2024; Yeman and Malaiya, 2024). Such contradictions are often detected only at system integration, thereby impeding iterative development and reducing planning stability. Consequently, agility in CPS development depends strongly on how cross-domain dependencies are handled, either by continuously preserving consistency or by explicitly managing inconsistencies until resolution becomes feasible (Rinker, 2025; Albers et al., 2024b).

In this context, CPS disciplines typically employ domain-specific practices, methods, and tools, while cross-domain dependencies must be coordinated across heterogeneous artifacts and models. We refer to this coordination as *engineering orchestration*. Based on the industrial study by Jongeling et al. (2022), orchestration in practice focuses on recurring areas of consistency concern between key artifact types, including relationships between design models and implementation, between models and tests, between architecture models and behavioral implementations, as well as between multi-artifact dependencies. This notion of engineering orchestration differs from deployment orchestration (e.g., Kubernetes), which operationalizes consistency as continuous reconciliation between a declared desired state and the observed runtime state (Burns, 2024). In contrast, engineering orchestration targets cross-domain consistency among discipline-specific representations and their underlying assumptions.

2.3. Implementation of consistency- and agility-related application

Related work on inter-model synchronization in safety- and mission-critical embedded systems emphasizes tool-supported detection and resolution mechanisms also for preserving consistency across evolving artifacts. For example, AutoFOCUS 3 provides an integrated model-based tool chain (including simulation, verification, and code generation) that supports continuous analysis of system designs (Aravantinos et al., 2015). Similarly, proof-based engineering approaches built around Architecture Analysis & Design Language (AADL) models enable automated consistency assurance across different phases (Hugues et al., 2008). Some AADL-centered tool chains further support automated code generation, allowing early verification of architectural assumptions prior to late-stage system integration (Lasnier et al., 2009). Such tool chains frequently rely on formal back-ends for contradiction detection (e.g., Satisfiability Modulo Theories (SMT) solvers for satisfiability checking De Moura and Bjørner, 2008 or component frameworks such as Behavior, Interaction, Priority (BIP) for preserving system-level properties across design steps Basu et al., 2011). From the perspective of consistency engineering, these tools function as mechanisms operating across abstraction levels to reduce the likelihood of contradictions remaining undetected until costly integration phases (Lasnier et al., 2009; Voelk et al., 2025a).

Model-driven approaches further operationalize consistency preservation through mechanisms such as the Virtual Single Underlying Model (V-SUM), which connects heterogeneous metamodels via explicitly defined consistency specifications (Kramer et al., 2013; Atkinson et al., 2008). Approaches such as Vitruvius (Klare et al., 2021) use Consistency Preservation Rules (CPRs) to encode cross-domain agreements and enable semi-automatic restoration after local changes (Klare et al., 2021; Reussner et al., 2023). This is particularly relevant in CPS contexts, where parallel work across domains frequently leads to diverging assumptions (e.g., when software iterations modify actuator requirements without corresponding mechanical design adjustments). Literature suggests that automated preservation of such dependencies can enable parallel iteration across disciplines while reducing integration rework (Albers and Lohmeyer, 2012; Reussner et al., 2023). However, preservation typically requires upfront agreement on shared semantics and therefore yields the greatest benefit when cross-domain dependencies are recurring and sufficiently understood (Klare et al., 2021; Reussner et al., 2023).

Conversely, inconsistency management tolerates temporary contradictions provided that mechanisms for diagnosis, visualization, and resolution are in place (Feldmann and Vogel-Heuser, 2024). Practitioners often report ambiguity in describing inconsistency situations, which complicates ad-hoc analysis and motivates structured support (Albers et al., 2024b; Voelk et al., 2025a). Hence, inconsistency management is increasingly regarded as an inherent aspect of daily engineering activity (Albers et al., 2024b). From an agility perspective, temporary tolerance may facilitate local progress when the shared understanding of cross-domain semantics is still evolving or when immediate propagation is impractical due to tooling boundaries or organizational separation (Feldmann and Vogel-Heuser, 2024; Albers et al., 2024b). However, unmanaged accumulation of inconsistencies can cause disruptive integration issues that undermine iterative delivery (Voelk et al., 2025a). Boundary conditions such as regulatory requirements, documentation scope, system complexity, and variant diversity further influence the frequency and resolution effort associated with inconsistencies (Albers et al., 2025a). Empirical analyses identify recurring patterns, including manual data transfer, tool incompatibilities, communication gaps, and misalignment between agile and sequential processes, that contribute to inconsistency emergence (Albers et al., 2025a; Voelk et al., 2025a; Feldmann and Vogel-Heuser, 2024).

2.4. Summary

The literature indicates that agile CPS development benefits from a combined strategy: consistency preservation is particularly advantageous for frequent and well-understood dependencies, enabling shorter feedback cycles and parallel work, whereas inconsistency management supports iteration under uncertainty by allowing controlled deviations that can later be resolved (Klare et al., 2021; Reussner et al., 2023; Feldmann and Vogel-Heuser, 2024). Additionally, where correctness constraints are critical, tool-supported detection and formal verification mechanisms can complement both strategies (Aravantinos et al., 2015; Albers et al., 2024a; Lasnier et al., 2009).

3. Research methodology

The description of the joined phenomenon of agility and consistency requires knowledge and practices originating inside but also outside the discipline of software engineering. Hence, our description will necessarily be interdisciplinary by nature. We aim to ensure that our description achieves transparency of the underlying causes and effects of the phenomenon; we also seek to achieve with our description a high level of acceptance beyond any single domain perspective. To that end,

we describe agile development of CPS according to a design research methodology

A typical research problem in design research entails two sides, a human side and an artifact side (Hevner and Chatterjee, 2010b). Likewise, in this study, we also investigate the human side of consistency problems when we investigate agile development of CPS; on the other hand, since we provide a means to communicate the phenomenon of consistency both in software and in hardware engineering research, we also create artifacts and in that way, contribute to the body of scientific evidence. There are different approaches to design research, all of which are viable and interrelated even (Hevner and Chatterjee, 2010a). We have chosen the design science research methodology (DSRM) of Peffers et al. (2007) because it is seen as an established design research methodology in both software and hardware engineering research (Hevner and Chatterjee, 2010a).

DSRM comprises these six core activities:

1. Problem identification and motivation
2. Define the objectives for a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication

This work addresses the activities 1 through 4. We aim to provide a strong foundation to describe the phenomenon of consistency in agile development. To that end, our influencing factors describe the phenomenon and the underlying cause-and-effect-chains derived from the literature; moreover, we also show the viability of that network of influencing factors to highlight consistency problems and its resolution through engineering solutions. To the best of our knowledge, our description of the intertwinement of consistency and agile development is the first of its kind; for that reason, we have decided to aim for precision and comprehensiveness in our description. One of the tradeoffs of this decision is that we can foreshadow activities 5 and 6. While this limits the generalizability achievable by this work, it nonetheless motivates further research in the field. Hence, the goal of this study (and therefore our claim to validity) is not to provide a complete solution for consistency preservation and agile development but instead to lay the foundations for evaluation in future work.

For the DSRM activity 1, we showed in Section 2 that agility and consistency can be understood as a joined phenomenon only when investigating complex system development such as the development of CPS.

For the DSRM activity 2, to operationalize our goal of laying a foundation for evaluation, the following objectives were chosen by the authors to evaluate the outcome of our work. These objectives (and therefore the result for activity 2 of the DSRM) are as follows:

1. The generated result of this study should be understandable across research disciplines.
2. The generated result of this study should explain detailed context on the discussed phenomenon.
3. The generated result of this study must be validated through a design practice example showing applicability on an industry use case.
4. The generated result of this study must enable future work to discuss contributions based on the research phenomenon discussed.

To conduct activity 3, we pose our first research question:

RQ1: Which influencing factors describe the phenomena of consistency preservation, inconsistency management, and agile development of CPS and how are they interconnected?

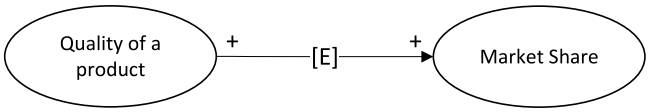

Fig. 1. Notion example for a network of influencing factors based on Blessing and Chakrabarti (2009).

To answer RQ1, we depict the current situation of agile development in engineering environments by mapping out a conceptual model that represents a network of influencing factors (hereafter referred to as the "network of influencing factors") based on the DRM — Design Research Methodology by Blessing and Chakrabarti (2009). This approach will enable researchers to identify the factors influencing the agility of a development process and also the role played by consistency in that phenomenon. Based on this knowledge, targeted measures can be addressed and validated. To operationalize this task, influencing factors are derived from multiple sources such as literature, assumptions, experiments, hypotheses, but also based on experience and research goals. An example of an influencing factor is the quality of a product and how that may affect market share. Every influencing factor comprises an attribute and its corresponding element; that is, each attribute corresponds to a quality of some element in the problem definition. Accordingly, when an influencing factor can be shown to link to others, then those links are depicted as arrows, where the direction of an arrow indicates the direction of factor influence and the value assigned the arrow (+, –, 0) indicates the type of change. Moreover, every arrow indicates, as well, the origin of information, as follows: published results with reference number x [X], assumptions [A], Experience [E], own investigations [O], and hypothetical links [?]. An example for such a notation is given in Fig. 1, using the dependency between product quality and effect in the market share.

Finally, to conduct activity 4, we pose our second research question:

RQ2 How can consistency preservation be applied to an inconsistency situation in agile development of CPS?

We show that consistency preservation can have a positive effect on successful agile implementation in the development of CPS by presenting the effect, in an agile development environment, of resolving an industrial inconsistency situation (i.e., where a lack of consistency causes a problem). Our industrial inconsistency situation is an anonymized data sample from industry (Albers et al., 2025c). To operationalize the situation, we transferred it to a cyber–physical brake system which is now an established research platform used by domain specialists from software engineering, mechanical engineering, and electrical engineering (Gesmann et al., 2025). The effectiveness of our consistency preservation for CPS development is demonstrated on the Vitruvius framework. We discuss implications based on the network of influencing factors derived in Section 4.1.

Again, activities 5 and 6 lie outside the scope of this study. Nevertheless, we discuss these activities as future work (Section 5–7), detailing the current limitations, the next development steps, and the implications for industrial practice.

4. Results

4.1. Consistency as a phenomenon of agile product engineering

Here we obtain the results to answer our RQ1. We do this in two steps. First, we derive the influencing factors on an agile development of CPS; and second, we use those factors as a basis for visualizing the current understanding of consistency and agility in CPS development practices.

Our derivation of influencing factors (Blessing and Chakrabarti, 2009) is based on an extended review of the literature about the groups

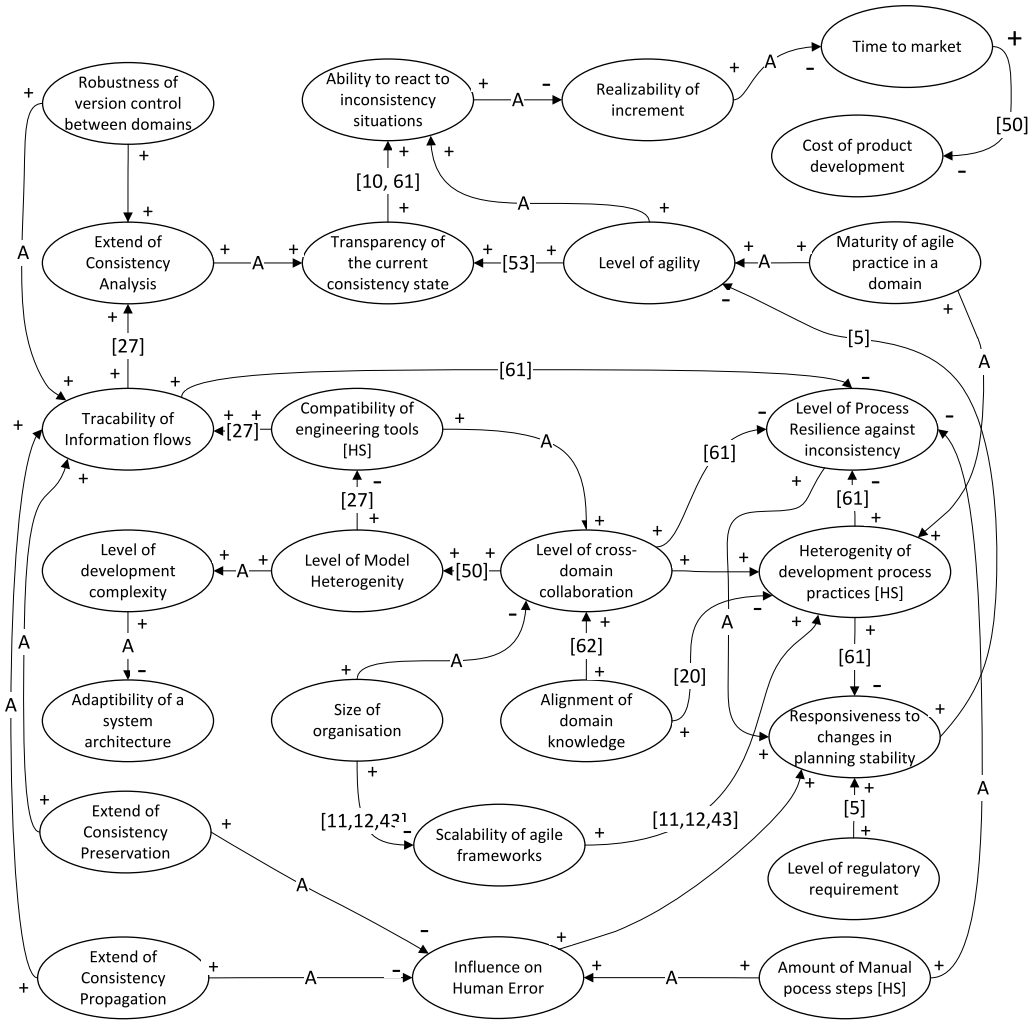

Fig. 2. Network of influencing factors (Blessing and Chakrabarti, 2009) to describe the impact of consistency preservation on agility in CPS engineering.

of topics introduced in Section 2; additionally, our derivation is also based on our own observations of current engineering practice and on our own assumptions about interconnections with other factors. Hence, this set of influencing factors has no claim to completeness; nevertheless, we do claim that it provides a coherent and integrative overview of current insights into agile development in CPS design. In that way, this set of influencing factors serves as a solid foundation for future investigation.

To begin, we explain briefly our formatting below. We have placed in the heading of each influencing factor the citations which provide those particular insights. To maintain readability and also to conform to the method of visualization (Blessing and Chakrabarti, 2009), we omit the literature from Fig. 2. Lastly, on each arrow in the figure, we indicate the argumentative basis for that connection, that is, whether the connection is based on literature (indicated by the citation or citations) or on our own assumptions (indicated by A).

4.1.1. Influencing factors on consistency preservation

Transparency of the current consistency state (Dumitrescu et al., 2021; Reussner et al., 2023; Schwaber, 2004)

Transparency is significantly enhanced through approaches like Model-Based Systems Engineering (MBSE), where unified models allow actors to represent knowledge explicitly and trace technical contexts. This concept is extended by introducing virtual single underlying (meta)-models for consistency preservation with the goal to enhance agility. Digital consistency and the networking of IT systems provide transparency through traceability, which is a primary expected benefit for development teams. Additionally, agile frameworks like Scrum aim for improved transparency in planning and documenting procedures to enable continuous improvement.

Extent of consistency analysis (Reussner et al., 2023; Feldmann and Vogel-Heuser, 2024; Spanoudakis and Zisman, 2001a; Albers et al., 2026)

Inconsistency management requires a systematic diagnosis of the current consistency state of a system in development, which involves localizing the inconsistency, identifying its root cause, and classifying it. Current research emphasizes the need for an in-depth analysis of (in-)consistency and its propagation across real-world projects to establish a unified understanding among domain experts.

Extent of consistency preservation (Reussner et al., 2023; Albers et al., 2024b)

Preserving consistency among different stakeholders in CPS engineering requires distinct tools and methods. A central vision for future engineering is the implementation of (semi-)automatic consistency preservation to reduce the high manual effort and coordination currently required to keep models consistent.

Extent of consistency propagation (Reussner et al., 2023; Albers et al., 2026)

Inconsistencies propagate through subsequent activities in engineering design, often crossing the boundaries of objectives, engineering artifacts and development activities that transform them in a continuous interplay. Tracing these cause-and-effect chains is essential for

understanding how a change in one domain (e.g., software) impacts a more "mature" (in the sense of "readiness" for integration) engineering state in another domain (e.g., mechanics).

4.1.2. Influencing factors on agility

Time to market (Reussner et al., 2023; Albers et al., 2019; Atzberger et al., 2019; Dumitrescu et al., 2021)

Shortening the time-to-market is considered a "hard" factor or key performance indicator (KPI) and is a primary motive for companies adopting agile development methods. Companies expect agile iterations to lead to faster product development compared to traditional plan-driven approaches.

Cost of product development (Atzberger et al., 2019; Dumitrescu et al., 2021)

Product development cost is a critical KPI often overestimated in its reduction through agility. However, early identification of errors through improved system understanding leads to massive savings in both development and manufacturing costs. Simultaneously, designing complex interdisciplinary systems involves high costs for the administration and orchestration of software tools.

Level of agility (Albers et al., 2019; Michalides et al., 2023b; Paetzold-Byhain et al., 2026)

Agility is characterized as the capability to react and adopt to both expected and unexpected changes constantly and quickly. Within synchronization processes, there are different levels of "agile pervasiveness". It can be conceptualized both as an attribute (behavioral pattern of the organization) and a construct (the underlying structure of elements like roles, activities, and artifacts).

Responsiveness to changes in planning stability (Albers et al., 2019)

There is a direct correlation between entropy and planning stability; as entropy (lack of knowledge) increases, planning stability decreases. In complex operative contexts, plans must be adjusted frequently to react to emerging insights, while at the same time especially physical deployment contexts (i.e. tooling machine manufacturing, or sourcing of parts) need a more stable, long-term planning foundation.

Level of development complexity (Dumitrescu et al., 2021; Schwaber, 2004)

Developmental complexity is driven by the interdisciplinary nature of modern systems and their networking with other systems (especially in the context of systems-of-systems (SoS)) that may be unknown during the initial design phase. It is defined by three dimensions: requirements, technology, and people. **Size of organization** (Michalides et al., 2023a,b; Schwaber, 2004)

Scaling agile is primarily relevant for corporations and large enterprises that develop complex systems, as these organizations have often reached a transition level where single teams can no longer manage the entire product development process. Regardless of the overall organizational size, individual team productivity is observed to decline once a team exceeds seven people (plus or minus two); beyond this size, miscommunications increase and the need for formal documentation grows.

Realization of increment (Schwaber, 2004; Weiss et al., 2023; Paetzold-Byhain et al., 2026)

An increment is defined as the product functionality developed by a team during a single Sprint. For it to be "potentially shippable", it must be thoroughly tested, well-written, and integrated into a state that the customer could use immediately. While software increments consist of digital source code, the realization of increments for physical products is a significant challenge due to the tangible nature of the components and longer manufacturing lead times. Because regular delivery of a final physical product is often impossible in short iterations, hardware increments are frequently reinterpreted as working prototypes, simulation results, or independent components that can be validated by the customer. In complex mechatronic environments, declaring a working prototype as the only measure of progress is often unrealistic; instead, progress is validated through engineering generations and a mix of virtual and physical artifacts.

Adaptability of a system architecture (Dumitrescu et al., 2021)

Because constraints and boundary conditions in development objectives change with every iteration, the design of the system architecture and its technical implementation must be adapted continuously. Formal modeling is used to describe and master this increasing systemic complexity.

Maturity of agile practice in a domain (Müller et al., 2024; Atzberger et al., 2019; Michalides et al., 2023a)

The maturity level of agile teams is often inconsistent in product development, which creates obstacles for cross-domain synchronization. Especially the gap between software- and hardware engineering are considerable.

Scalability of agile frameworks (Michalides et al., 2023b,a; Voelk et al., 2025b; Atzberger and Paetzold, 2019)

Scaling agile methods requires an appropriate infrastructure and mechanisms for frequent synchronization between multiple teams. A major challenge in the industry is the lack of systematic models for introducing agile processes that satisfy scalability requirements across many departments.

4.1.3. Influencing factors on inconsistency management

Ability to react to inconsistency situations (Schwaber, 2004; Albers et al., 2024b, 2026)

Agile methodologies facilitate a quick response to inconsistencies through frequent inspect-and-adapt cycles that allow for on-the-spot adjustments when deviations are detected. Although these situations are often identified retrospectively when a conflict becomes imminent, regular retrospectives help teams resolve them by uncovering root causes within the design process.

Robustness of version control between domains (Voelk et al., 2025b)

Merging different versions of files into one consistent state is a significant challenge in multi-domain development. Patterns of inconsistency are frequently linked to poor version control, leading to the use of outdated models and missing validation steps.

Traceability of information flows (Paetzold, 2022; Voelk et al., 2025b)

Traceability is a core component of digital continuity, linking model elements, requirements, and test cases. In sectors like medical or automotive technology, traceability and information availability are current challenges in available software tools.

Compatibility of engineering tools (Feldmann and Vogel-Heuser, 2024; Reussner et al., 2023; Voelk et al., 2025b)

Views on a system are often isolated due to incompatibilities like proprietary file standards, or vendor lock-in. Discontinuities are common because of a lack of standardized exchange formats or domain-specific requirements and preferences.

Level of model heterogeneity (Voelk et al., 2025b)

A variety of actors from different disciplines use heterogeneous modeling approaches, formalisms, and software tools. This creates a model landscape characterized by overlapping and redundant information, which is a major source of inconsistency when requirements change.

Influence of human error (Voelk et al., 2025b)

Maintaining relationships across domain-specific models is a tedious and error-prone task. The "Human Factor", including time pressure and forgetfulness in manual updates, is identified as a pattern that worsens inconsistency occurrence.

Level of process resilience against inconsistency (Albers et al., 2026)

When working in interdisciplinary engineering teams and, results must have enough stability so that the probability and cost of a change is lower than the potential cost of an inconsistent state. Resilience is

also a central requirement for the integration of computational and physical processes in modern CPS.

Amount of manual process steps (Voelk et al., 2025b)

Many inconsistencies are caused by manual intervention, manual data transfers, and a lack of automated synchronization across different tools. High manual effort is required to maintain consistency between isolated views in separate modeling languages.

Heterogeneity of development process practices (Choi and Pak, 2006; Voelk et al., 2025b; Albers et al., 2024b)

Inconsistency arises when agile and sequential processes coexist without proper alignment. This mismatch is amplified by the different development cycles of hardware (long cycles) versus software (short iterations).

Level of regulatory requirement (Albers et al., 2025b)

High requirements of the development process are often driven by regulatory aspects like the Medical Products Act or liability for autonomous systems. These regulations necessitate strict documentation and verification phases.

Alignment of domain knowledge (Voelk et al., 2025d,b)

Collaboration requires a common, multidisciplinary development language and common mental models to bridge "epistemic barriers" rooted in domain-specific terminology. Misalignment occurs when different domains have differing interpretations of the same terminology or priorities.

Level of cross-domain collaboration (Albers, 2023)

Complex systems like CPS can be developed only through cross-disciplinary approaches that go beyond knowledge silos. Current mechatronic development is often hindered by collaboration that is limited to the exchange of "strictly separated partial results" rather than integrated results.

4.1.4. Visualizations of influencing factors on consistency as a phenomenon of agile product engineering

After deriving the necessary influencing factors, we now interconnect them and thereby answer RQ1. Fig. 2 visualizes our influencing factors as a network (based on Blessing and Chakrabarti (2009)).

4.2. Application of consistency preservation on a problem case study from industry

The core line of argumentation which emerges from Fig. 2 is the following: Agility and consistency are made mutually beneficial by the "ability to react to inconsistency situations", and also, this same ability increases (1) as an organization's "level of agility" increases, and (2) as the "transparency of the current consistency state" increases too.

We now turn to providing an initial validation of the above core finding, thereby answering our RQ2. This validation is our key contribution to theoretical future work, namely, a first explanation *for* as well as first prevention *of* an inconsistency which hindered the agile development of an organization. To begin, we provide information on the inconsistency situation to be resolved. To strike a balance between industry relevance and transparency of inconsistency information, we use a real inconsistency situation from industry stemming from anonymized case descriptions (Albers et al., 2025a). As those cases do not contain precise information on models, architectures, or even data, we have transferred the cases to an explainable and modelable situation: Our inconsistency situation is based on the example of a cyber–physical brake system as an established research platform case (Gesmann et al., 2025).

In Section 4.2.1, we describe the context in which inconsistencies occur, present the inconsistency situation based on standardized SysML models (Object Management Group, 2025), and outline the problem using our influencing factor network. Section 4.2.3 provides a conceptual explanation of the chosen approach to realize transparency of the current consistency state, as proposed in Klare et al. (2021) within the Vitruvius framework through the Virtual Single Underlying Model (V-SUM) concept. In Section 4.2.4, we present the implementation of the consistency preservation rules and illustrate their effect on the resulting consistency state.

4.2.1. Description of an inconsistency situation in the development of a cyber–physical brake system

In the development of automotive braking systems, the transition from conventional hydraulic architectures to hydraulic and recuperative electrical (and therefore hybrid) braking systems introduces substantial cross-domain complexity. In brake systems from the previous generation of a generic car described by the notation G_{n-1}, braking functionality is realized through a purely hydraulic–mechanical chain: pedal input is transformed via a brake booster into hydraulic pressure, which is modulated by an anti-lock braking system (ABS) based on wheel slip and subsequently applied to the wheel brakes to generate frictional torque, as given in Fig. 3. This infrastructure has to be kept for the next development generation due to regulatory requirements.

In the next (and currently developed) generation of the vehicles series described by the notation G_n, this architecture is extended by an electromechanical braking path enabling regenerative braking for hybrid-electrical cars. A central brake controller interprets pedal input as a braking request and distributes it across both domains: the hydraulic braking system and the new electric system components as illustrated in Fig. 4. Through an inverter, the electric motor operates in generator mode, contributing a negative torque while recovering energy. Additionally, the system design introduces an active coupling between the controller and the brake booster via a modulation interface to the vacuum pump, allowing dynamic adjustment of brake force amplification to coordinate overall braking behavior.

The synthetical derived inconsistency arises due to misaligned assumptions across engineering domains. While control and system design explicitly rely on an actively controllable brake booster, the mechanical domain treats the booster and vacuum pump as off-the-shelf components without external control interfaces. As a result, the intended control strategy is not supported by the physical system architecture due to system specification, leading to an inconsistency between system-level design and domain-specific implementation. In our industry case, the inconsistency itself only got detected during system integration activities: All simulations relied on the own intended use-cases and assumed the other domains result comply to the own system architecture.

At first glance, such an inconsistency situation could be expected to be resolved easily with given tool support. However, the presented system model primarily constitutes a meta-model of the system structure rather than an integrated representation of consistent engineering data. The underlying artifacts originating from heterogeneous tools and domain-specific processes are only abstracted within the model and are not inherently synchronized. Consequently, consistency is manageable at the model level but not guaranteed at the domain-specific level.

While this abstraction is manageable for comparatively small systems such as our simplified braking example with a unified notion, limitations arise when focusing on centralized system notions (or Single-Underlying Models): Such approaches do not scale to larger, highly integrated systems like full vehicle architectures, production plants or electricity networks (and on the same time remain maintainable). Also, scalability is not enabled for more detailed and heterogeneous descriptions of the models introduced, as all representations have to be kept consistent as well. Increasing system size and domain and modeling approach heterogeneity introduce a rapid growth in domain interactions, artifact heterogeneity, and interdependencies, making it infeasible to maintain complete and up-to-date consistency relations within a centralized model. This creates a gap between the perceived consistency in system models and the actual consistency of the engineered system.

4.2.2. Using the network of influencing factors to explain the consistency problem and its effect on agility

This gap directly relates to the research phenomenon addressed by the network of influencing factors displayed in Fig. 2. In agile development environments, the realizability of increments critically depends on the ability to detect and resolve inconsistencies in a timely

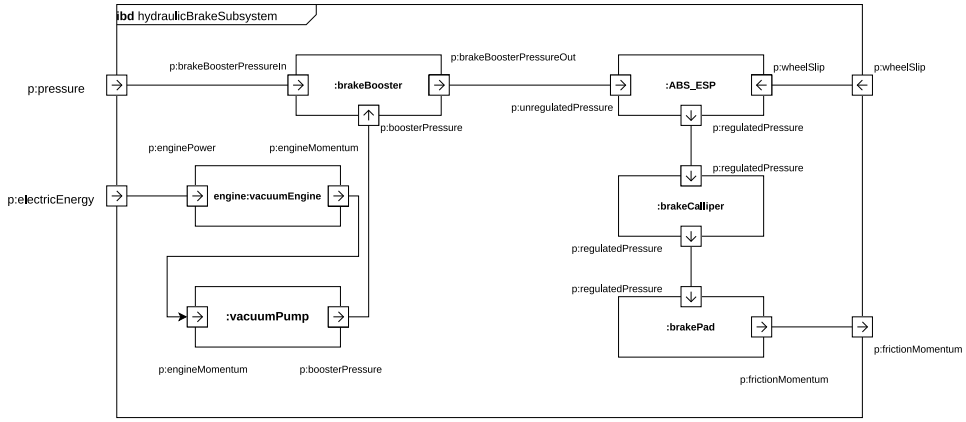

Fig. 3. Internal Block Diagram (ibd) of the hydraulic (and therefore mechanical) brake system generation G_{n-1}.

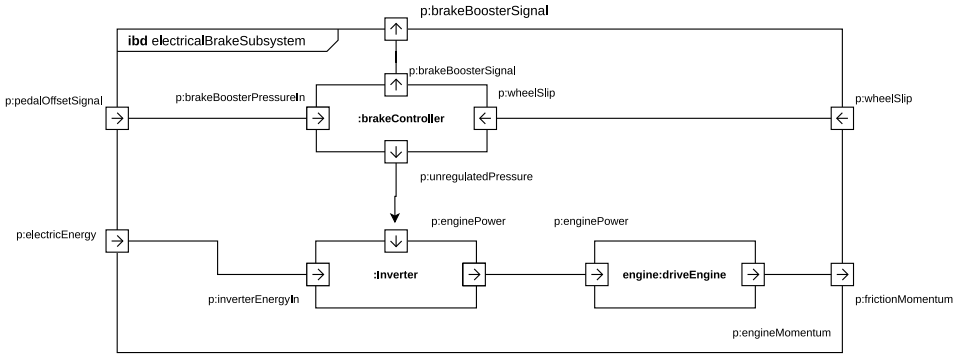

Fig. 4. Internal Block Diagram (ibd) of the electrical components of the second brake system generation G_n.

manner. This ability to react to inconsistencies is strongly dependent on the transparency of the current consistency state. However, in our case several systemic factors negatively influence this transparency. As organizational size increases by adding an entirely new engineering department (like an electrical engineering department in our case), the overall ability for cross-domain collaboration typically decreases, leading to a rise in the heterogeneity of development process practices. This heterogeneity manifests in divergent modeling approaches (and also system architectures as presented in the case), toolchains, and engineering workflows. As a consequence, traceability across domains deteriorates, tool interoperability decreases, and dependencies become less visible. These effects reduce both responsiveness to change and planning stability, as changes require extensive coordination and their system-wide impact becomes harder to assess.

The combined effect is a significant reduction in transparency regarding system consistency. As inconsistencies become harder to detect and evaluate, the ability of the development organization to react to them is diminished. Ultimately, this reduction in agility directly impacts the realizability of development increments. Increments can no longer be reliably implemented within iteration boundaries due to unresolved or late-detected inconsistencies.

The presented braking system example serves as a minimal yet representative case to illustrate this phenomenon. Despite its limited scope, it captures essential characteristics of modern system development: cross-domain dependencies, heterogeneous engineering artifacts, and implicit assumptions between teams. As such, it provides a controlled environment in which the emergence and impact of inconsistencies can be clearly demonstrated, while still being sufficiently complex to reflect challenges observed in large-scale system development.

4.2.3. Conceptual Basis for Consistency State Transparency in Vitruvius

Vitruvius is a framework that provides tool support for consistency preservation between interrelated models (Vitruv Team, 2024). The framework is built upon EMF (Steinberg et al., 2008) and is based on the Virtual Single Underlying Model (V-SUM) concept, which aims at maintaining multiple models in a virtually synchronized state representing a single system. Vitruvius distinguishes between two primary roles: methodologists, who define metamodels and their semantic relations using domain knowledge, and developers, who develop the system under construction by instantiating the V-SUM specified by the methodologist. To operationalize these semantic relations, the framework provides a Domain-Specific Language (DSL), called “Reactions”, through which developers specify Consistency Preservation Rules (CPRs) linking elements of different metamodels. A Reaction defines a trigger (a change in the source model) and an associated effect (the corresponding updates required in the target model) to restore and maintain consistency. In addition to enabling automated propagation of model changes, CPRs contribute to improving transparency of the current consistency state by making cross-model dependencies explicit and traceable. This is achieved through the “Reactions” language, which allows developers to specify which elements of which models must remain synchronized, how consistency should be restored, and when synchronization should occur. By explicitly defining these conditions for change propagation, the synchronization process becomes both transparent and documented. In this way, Vitruvius supports developers in understanding how changes in one domain affect related representations in another domain and allows them to assess whether models still jointly reflect a consistent state of the system. For instance, it ensures that hydraulic (mechanical) and electrical representations of

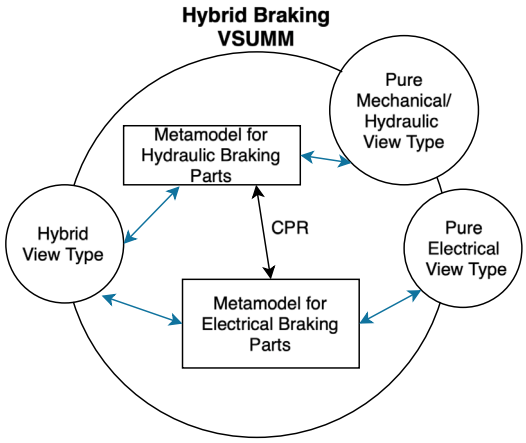

Fig. 5. V-SUMM for hybrid braking.

the brake system remain aligned while collectively representing a single underlying system through the V-SUM. As illustrated in Fig. 5, a hybrid view type integrates components from both mechanical and electrical domains into a unified representation.

4.2.4. Implementation of the V-SUM concept using VITRUVIUS *on the cyber–physical brake system*

We applied the VITRUVIUS framework to the inconsistency situation described in Section 4.2.1 by first designing two metamodels that represent the relevant modeling elements (Figs. 6 and 7) and subsequently specifying how consistency between them should be maintained. As explained in Section 4.2.3, using the V-SUM concept proposed by Klare et al. (2021), consistency between the corresponding models must be preserved such that they remain virtually aligned and jointly represent a single underlying system, despite being maintained as separate but interrelated artifacts. Based on these specifications, CPRs were defined to enforce synchronization between the models. An example illustrating the use of Reactions is provided in Listing 1. As shown, changes to the measured rotational speed on the mechanical side are automatically propagated to the corresponding value on the electrical side to restore and maintain cross-model consistency.

Listing 1: Reaction to a change in the value of the *currentRPM* attribute in the brake booster engine

```
reaction RPMChange {
    after attribute replaced at hb::
        brakeBoosterEngine[currentRPM]
    call changeRPM(affectedEObject)
}
routine changeRPM(hb::brakeBoosterEngine bbe) {
    match {
        val eb_bbe = retrieve eb::
            brakeBoosterController corresponding to
            bbe
    }
    update{
        eb_bbe.rpmSetPoint = bbe.currentRPM
    }
}
```

5. Discussion and future work

As indicated in Section 4, this work does not seek to provide a fully developed solution for an engineering problem, but aims, instead, to provide a consensus scientific description to make the phenomenon of agility in the context of consistency preservation accessible. To that end, we discuss the implications of our theoretical work and practical evaluation on future research. First, we discuss hypotheses that can be derived from analyzing the network of influencing factors given in Section 4.1. Based on those hypotheses, we discuss how future work should approach the research phenomenon to provide more empirical evaluation and also to provide necessary proof to the proposed items.

5.1. Discussion of the result for RQ1

To investigate the influence of consistency preservation and inconsistency management on an agile development of CPS, Section 4.1 introduces a network of influencing factors. This network enables cause-and-effect-argumentations, that enable the discussion of certain statements on the influence of consistency practice and agility:

1. Consistency Preservation, Propagation and Analysis contribute to transparency of developed increments, leading to a faster ability to react to inconsistency and therefore to a better realizability of an engineering increment.
2. Consistency Preservation and Propagation lessen the influence of human error, providing more flexibility regarding the required planning stability of CPS development processes.
3. To increase the responsiveness to changes in planning stability (leading to a higher level of agility), the resilience of processes against inconsistency has to be increased.
4. A core barrier to increase the responsiveness to changes in planning stability is collaboration between domains, manifested in heterogeneous development practices, heterogeneous models and artifacts, or compatibility of tools.

The case study in Section 4.2 shows that the application of the network of influencing factors enables a discussion on the effects of an inconsistency to agile practices. Also, this application provides initial evidence for the hypotheses. Still, empirical research has to be conducted to investigate those cause-and-effect-chains through implementation of engineering support of processes, methods, and tools.

Especially the perspective of the effect of tool-based consistency preservation enables future work on a new perspective to typical engineering processes of cyber–physical systems and complex mechatronic systems. Currently, most companies implement a form of “design freeze” (Eger et al., 2005) as milestones in their process. Those “frozen” system states enable the assumption of consistent reference states, implementing transparency of the current consistency state in current engineering practice. In those states, changes are not possible after a given point, enabling a good planning stability for design and integration of subsystems during the development process. Such elements lead to a high level of planning stability (i.e. to develop production systems suiting for a product) but make a company unresponsive to changes in planning stability. This rigid structure has a core weakness: By assuming inconsistency is resolved in a design freeze, unforeseen inconsistency propagates through all involved engineering domains, potentially breaking a majority of subsequent engineering artifacts.

By overcoming design freezes as the current state of the art of transparency mechanism for consistency assessment, tool-based approaches for consistency preservation might enable a new kind of agility in the multi-disciplinary development of cyber–physical systems. Continuous assessment of the current consistency state helps engineers rating their design decisions, even if propagation is not possible yet. The information that something *may* potentially become inconsistent by deploying a change makes this inconsistency visible, enabling communication between developers.

To integrate the found approaches into actual development processes, a completely new understanding of the integration activities in agile and sequential process models has to be found. A single process model most likely will not cover the domain-specific challenges resulting in the different engineering practices, as otherwise a best practice like Scrum would have already dominated the market (which is already the case only taking organizational benefits into

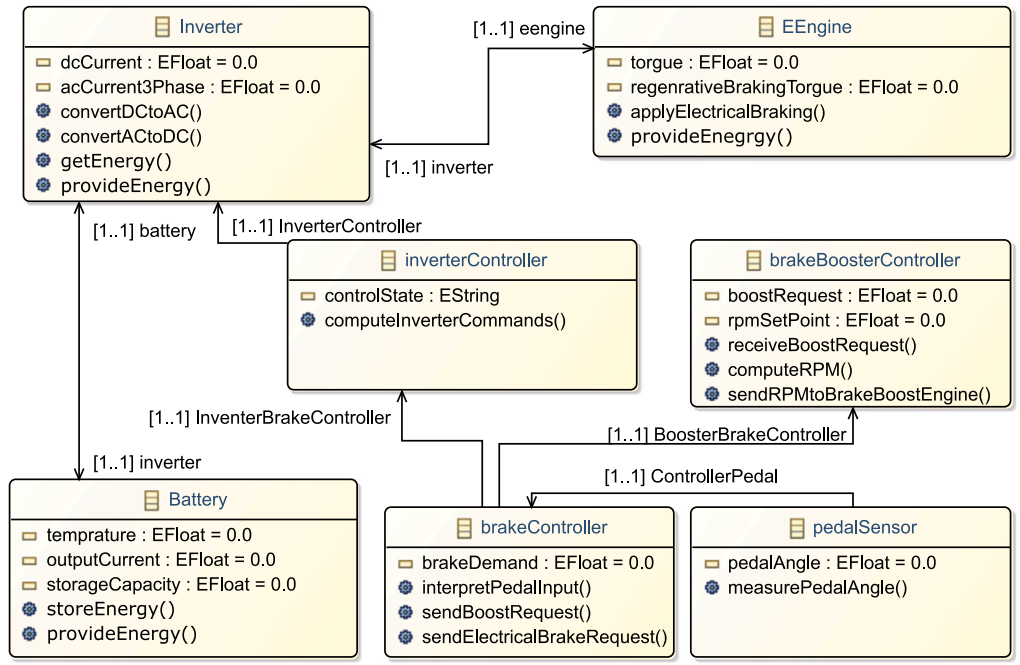

Fig. 6. Metamodel for electrical parts of a hybrid brake.

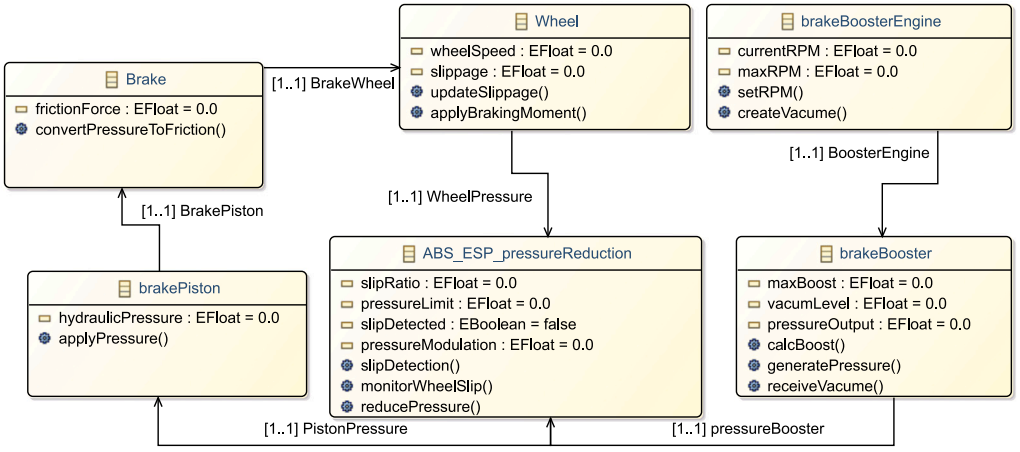

Fig. 7. Metamodel for hydraulic parts of a hybrid brake.

account Michalides et al., 2023b). Therefore, a meta-perspective based the function of integration activities in engineering has to be found, building on established models for development processes, e.g., the internationally renowned standards for hardware development processes VDI2206 (Verein Deutscher Ingenieure e. V., 2021) or VDI 2221 (Verein Deutscher Ingenieure e. V., 2019), and the insurance of consistency (building on established events, e.g. Scrums Sprint Review). Such a meta-perspective should indeed not just be "yet another Scrum", but should be incorporable both into established frameworks and individual development practice in industry.

5.2. Discussion of the result for RQ2

Section 4.2 shows that using Vitruvius supports change-based consistency preservation of models, therefore showing that the application of the framework can support agility in the case of an inconsistency situation. The framework enables quick feedback cycles, as changes can be propagated iteratively and rapidly, indicating the validity of the found hypotheses. However, sufficient time needs to be invested in building models, which creates a shared understanding and facilitates interdisciplinary collaboration. Also, defining CPRs is time-consuming, especially for non-standard models that require substantial upfront effort, potentially undermining the possible time-related benefits associated with agility. The specification and use of multiple views (tailored to specific domains and the aspects needed for the developers in these domains) allows tailoring the information to the needs of the developers, improving focus on only the relevant parts of a system for a specific task.

In our case study (which applied the Vitruvius framework), mechanical and electrical engineers reported that the integration of consistency preservation tooling with their existing inconsistency management processes enhanced development agility. Specifically, it facilitated faster propagation of updates across related models and supported more effective incremental model evolution.

It should be noted that new functionalities are going to be added to Vitruvius to broaden its support for blended development practices.

First, it will allow developers to customize the change propagation decision, taking into account the maturity of the modifications and the likelihood that a propagation will later be rolled back. Second, Vitruvius will permit temporary "experiment" runs that resemble agile prototyping: the effects of a propagation can be observed and then either continued or aborted. To enable this, a branching mechanism is being introduced so that changes can be propagated on non-main branches without disturbing the primary development line. These extensions give Vitruvius a range of integration points that combine the flexibility of agile methods with the rigor of traditional model-driven engineering.

Additionally, further extensions are planned in line with adaptability and incrementality. Incremental propagation enables selective updates by modifying only the affected model elements, rather than regenerating the entire model. In parallel, adaptive MDD approaches leverage feedback gathered throughout the development process to dynamically update models, ensuring their alignment with evolving requirements and changing conditions. Together, these incremental and adaptive capabilities promise to enable a new level of agility in engineering systems.

6. Threats to validity

We base our discussion here on the validity threats outlined by Wohlin et al. (2012).

- External Validity: Regarding the application in Vitruvius, the conclusions drawn about overlaps may not be universally applicable since the metamodels are designed for specific purposes. It should be noted that extension of metamodels is possible to support more general scenarios.
  The derived reference model and the discussion of implications mainly refer to academic publications. As Voelk et al. (2025c) pointed out, only a limited amount of industrial practice is available in peer-reviewed literature. To derive a more representative understanding of agility in the context of CPS, more industrial cases have to be analyzed. The same situation applies to inconsistency situations: without large-scale industry background data, the generalization of found results remains challenging.
- Construct Validity: Regarding the application in Vitruvius, inappropriate attributes may be used to synchronize overlap in the analysis of semantic overlaps, and the overlaps themselves may be misinterpreted. To mitigate this risk, expert interviews were conducted.
  The derived reference model and the discussion of implications include many assumptions, especially about cross-disciplinary collaboration and the extent of consistency preservation activities. To verify the individual claims and therefore to validate the reference model, further empirical research has to be conducted so that the given factorial dependencies are evaluated.

7. Conclusion

This paper explored how consistency preservation and inconsistency management influence the applicability of agile development practices in cyber–physical systems (CPS) engineering. While agility is well established in software engineering, its transfer to CPS remains challenging due to physical constraints, heterogeneous engineering domains, and the need for planning stability. Although prior work frequently suggests that consistency mechanisms support agility, this relationship has largely remained implicit and insufficiently theorized.

To address this gap, we developed a theoretically grounded explanation of the interplay between consistency and agility by explicating a network of influencing factors. Rather than proposing a prescriptive process or reference architecture, this network makes explicit the dependencies and trade-offs that shape agile CPS development under conditions of cross-domain complexity answered by consistency mechanisms. By visualizing these relationships, the contribution of this work lies in clarifying how consistency preservation, inconsistency management, human involvement, tooling, and organizational characteristics jointly influence responsiveness, planning stability, and the realizability of development increments. These conceptual findings were complemented by an industry-motivated case study of a hybrid braking system. The case illustrates how model-based consistency preservation can improve transparency of the current consistency state and support faster feedback cycles without compromising necessary stability. At the same time, the case highlights that these benefits are associated with significant upfront modeling and rule-specification effort. This confirms that consistency preservation is not a universal remedy for agility challenges, but a mechanism whose effectiveness depends on domain maturity, recurring cross-domain dependencies, and organizational context.

Our work contributes (1) by making the relationship between consistency mechanisms and agility explicit through an interdisciplinary explanatory model, (2) by positioning CPS development as a stress test for studying agility in complex systems engineering, and (3) by providing a structured foundation for future empirical research. Our conceptual model of influencing factors does not claim completeness; further, our case study serves as an illustrative validation rather than causal proof. Future work should therefore investigate the identified relationships across larger and more diverse industrial contexts.

Overall, in this paper, we have argued that consistency preservation should be understood neither as an obstacle to agility nor as an end in itself; instead, we make the case that consistency preservation (when appropriately combined with inconsistency management) is a key mechanism to enable agile development in cyber–physical systems. By making the underlying relationships explicit, this work provides a basis for more informed design decisions and opens avenues for systematically studying agility in complex, interdisciplinary engineering environments.

CRediT authorship contribution statement

Albert Albers: Writing – review & editing, Supervision, Resources, Project administration, Investigation, Funding acquisition, Conceptualization. **Anne Koziolek:** Writing – review & editing, Supervision, Resources, Project administration, Investigation, Funding acquisition, Conceptualization. **Thomas Alexander Voelk:** Writing – review & editing, Writing – original draft, Visualization, Validation, Project administration, Methodology, Data curation, Conceptualization. **Razieh Dehghani:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Conceptualization. **Thomas Weber:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Formal analysis, Data curation. **Tilman Pfersdorff:** Writing – original draft, Visualization, Validation, Methodology, Data curation. **Frederik Beck:** Writing – original draft, Validation, Methodology, Data curation. **Maximilian Beck:** Writing – review & editing, Validation. **Katharina Bause:** Writing – review & editing, Validation. **Tobias Düser:** Writing – review & editing, Supervision, Resources, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Thomas Alexander Voelk reports financial support was provided by German Research Foundation. Razieh Dehghani reports financial support was provided by German Research Foundation. Thomas Weber reports financial support was provided by German Research Foundation. Maximilian Beck reports financial support was provided by German Research Foundation. Katharina Bause reports financial support was provided by German Research Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1608 – 501798263. This research was edited by our textician, Daniel Shea.

Data availability

Data will be made available on request.

References

Albers, Albert (Ed.), 2023. Rethinking and Reframing Engineering. Challenges, Application Scenarios and the New Advanced Systems Engineering Model. Munich, http://dx.doi.org/10.48669/aca_2023-21.

Albers, Albert, Lohmeyer, Quentin, 2012. Advanced systems engineering–towards a model-based and human-centered methodology. In: 9th International Symposium on Tools and Methods of Competitive Engineering TMCE. pp. 407–416.

Albers, Albert, et al., 2019. Eine systematik zur situationsadäquaten mechatroniksystementwicklung durch ASD - agile systems design. http://dx.doi.org/10.5445/IR/1000091847.

Albers, Albert, et al., 2024a. Consistency in cross-generational engineering of cyber-physical systems. In: DS 133: Proceedings of the 35th Symposium Design for X. DFX2024, pp. 085–094.

Albers, Albert, et al., 2024b. Identification of inconsistencies in agile CPS engineering with formula student. In: Bitran, Iain, Conn, Steffen, Mitsis, Alex, Ritala, Paavo, Torkkeli, Marko, Trabelsi, Meriam (Eds.), Proceedings of the XXXV ISPIM Innovation Conference. ISBN: 978-952-65069-6-8, pp. 1–15. http://dx.doi.org/10.5445/IR/1000173128.

Albers, Albert, et al., 2025a. Inconsistency Situations in Engineering of Cyber-Physical Systems. Zenodo, http://dx.doi.org/10.5281/zenodo.13968934, URL: https://zenodo.org/records/15267521.

Albers, Albert, et al., 2025b. How to "measure" a mirage: Assessing inconsistency in mechatronic and cyber-physical development projects. In: 2025 IEEE International Symposium on Systems Engineering. ISSE, IEEE, pp. 1–8. http://dx.doi.org/10.1109/ISSE65546.2025.11370063.

Albers, Albert, et al., 2025c. Inconsistency situations in engineering of cyber- physical systems. http://dx.doi.org/10.5281/zenodo.15267521.

Albers, Albert, et al., 2026. Chasing symptoms, missing causes: Modelling parallel engineering activities in CPS development with the advanced system triple approach. Procedia CIRP (ISSN: 22128271) http://dx.doi.org/10.1016/j.procir.2026.05.344.

Aravantinos, Vincent, et al., 2015. AutoFOCUS 3: Tooling concepts for seamless, model-based development of embedded systems. In: ACES-MB&wuCOR@ MoDELS. Vol. 1508, pp. 19–26.

Atkinson, Colin, Stoll, Dietmar, Bostan, Philipp, 2008. Orthographic software modeling: a practical approach to view-based development. In: International Conference on Evaluation of Novel Approaches To Software Engineering. Springer, pp. 206–219.

Atzberger, Alexander, Paetzold, Kristin, 2019. Current challenges of agile hardware development: What are still the pain points nowadays? Proc. Des. Soc.: Int. Conf. Eng. Des. 1 (1), 2209–2218. http://dx.doi.org/10.1017/dsi.2019.227.

Atzberger, Alexander, et al., 2019. Evolution of the hype around agile hardware development. In: 2019 IEEE International Conference on Engineering, Technology and Innovation. ICE/ITMC, IEEE, ISBN: 978-1-7281-3401-7, pp. 1–8. http://dx.doi.org/10.1109/ICE.2019.8792637.

Basu, Ananda, et al., 2011. Rigorous component-based system design using the BIP framework. IEEE Softw. 28 (3), 41–48.

Beck, Kent, et al., 2001. Manifesto for agile software development. URL: http://agilemanifesto.org/.

Bézivin, Jean, 2005. On the unification power of models. Softw. Syst. Model. 4 (2), 171–188.

Bhave, Ajinkya, et al., 2011. View consistency in architectures for cyber-physical systems. In: 2011 IEEE/ACM Second International Conference on Cyber-Physical Systems. IEEE, pp. 151–160.

Blessing, Lucienne T.M., Chakrabarti, Amaresh, 2009. DRM, a Design Research Methodology. Springer Dordrecht, Heidelberg, London, New York, ISBN: 978-1-84882-586-4, http://dx.doi.org/10.1007/978-1-84882-587-1.

Braga, Christiano, Santos, Cássio, da Silva, Viviane Torres, 2014. Consistency of model transformation contracts. Sci. Comput. Program. 92, 86–104.

Burns, Brendan, 2024. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Systems Using Kubernetes. " O'Reilly Media, Inc.".

Choi, Bernard C.K., Pak, Anita W.P., 2006. Multidisciplinarity, interdisciplinarity and transdisciplinarity in health research, services, education and policy: 1. Definitions, objectives, and evidence of effectiveness. Clin. Investig. Med. Med. Clin. Exp. (ISSN: 0147-958X) 29 (6), 351–364, URL: https://pubmed.ncbi.nlm.nih.gov/17330451/.

Cicchetti, Antonio, 2016. Consistency and uncertainty in the development of cyber-physical systems. In: Vangheluwe, Hans, et al. (Eds.), MPM4CPS: Multi-Paradigm Modelling for Cyber-Physical Systems. pp. 56–61, URL: https://riuma.uma.es/xmlui/bitstream/handle/10630/12412/mpm4cps-gdanskp-2016.pdf?sequence=3&isallowed=y#page=62.

Clear, James, Clear, James, 2018. Atomic Habits: Tiny Changes, Remarkable Results. Penguin Publishing Group.

Clements, Paul, et al., 2003. Documenting software architectures: views and beyond. In: 25th International Conference on Software Engineering, 2003. Proceedings. IEEE, pp. 740–741.

De Moura, Leonardo, Bjørner, Nikolaj, 2008. Z3: An efficient SMT solver. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, pp. 337–340.

Dumitrescu, Roman, et al., 2021. Engineering in Germany - the status quo in business and science, a contribution to advanced systems engineering. URL: www.advanced-systems-engineering.de.

Eger, Tido, Eckert, Claudia, Clarkson, P. John, 2005. The role of design freeze in product development. In: DS 35: Proceedings ICED 05, the 15th International Conference on Engineering Design, Melbourne, Australia, 15.-18.08. 2005.

Feldmann, Stefan, Vogel-Heuser, Birgit, 2024. Diagnose von inkonsistenzen in heterogenen engineeringdaten. In: Vogel-Heuser, Birgit, ten Hompel, Michael, Bauernhansl, Thomas (Eds.), Handbuch Industrie 4.0. In: Springer Reference, Springer Vieweg, Berlin, ISBN: 978-3-662-58527-6, pp. 909–928. http://dx.doi.org/10.1007/978-3-662-58528-3_91.

Gesmann, Lars, et al., 2025. Ensuring consistency in CPS engineering: Braking systems as a research platform. In: 11th Int. Symposium on Development Methodology. in press.

Hebig, Regina, Khelladi, Djamel Eddine, Bendraou, Reda, 2016. Approaches to co-evolution of metamodels and models: A survey. IEEE Trans. Softw. Eng. 43 (5), 396–414.

Heimicke, Jonas, 2025. Eine Methodik zur Entwicklung agil-strukturierter Prozesslösungen für die Einführung des ASD - Agile Systems Design (German. Ph.D. thesis). Karlsruher Institut für Technologie (KIT), p. 229. http://dx.doi.org/10.5445/IR/1000181477.

Hevner, Alan, Chatterjee, Samir, 2010a. Design science research frameworks. In: Design Research in Information Systems: Theory and Practice. Springer US, Boston, MA, ISBN: 978-1-4419-5653-8, pp. 23–31. http://dx.doi.org/10.1007/978-1-4419-5653-8_3.

Hevner, Alan, Chatterjee, Samir, 2010b. Introduction to design science research. In: Design Research in Information Systems: Theory and Practice. Springer US, Boston, MA, ISBN: 978-1-4419-5653-8, pp. 1–8. http://dx.doi.org/10.1007/978-1-4419-5653-8_1.

Hugues, J, et al., 2008. Using AADL to build critical real-time systems: Experiments in the IST-ASSERT project. In: Embedded Real Time Software and Systems. ERTS2008.

Jadoon, Gullelala, ter Beek, Maurice H., Ferrari, Alessio, 2025. Model transformation and property preservation in rigorous software development: A systematic literature review. J. Syst. Softw. 230, 112508.

Jongeling, Robbert, et al., 2022. Consistency management in industrial continuous model-based development settings: a reality check: R. Jongeling et al. Softw. Syst. Model. 21 (4), 1511–1530.

Klare, Heiko, 2018. Multi-model consistency preservation. In: Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. pp. 156–161.

Klare, Heiko, et al., 2021. Enabling consistency in view-based system development—the VITRUVIUS approach. J. Syst. Softw. 171, 110815.

Kramer, Max E., Burger, Erik, Langhammer, Michael, 2013. View-centric engineering with synchronized heterogeneous models. In: Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling. pp. 1–6.

Kretschmer, Roland, et al., 2021. Consistent change propagation within models. Softw. Syst. Model. 20 (2), 539–555.

Lasnier, Gilles, et al., 2009. Ocarina: An environment for aadl models analysis and automatic code generation for high integrity applications. In: International Conference on Reliable Software Technologies. Springer, pp. 237–250.

Macedo, Nuno, Jorge, Tiago, Cunha, Alcino, 2016. A feature-based classification of model repair approaches. IEEE Trans. Softw. Eng. 43 (7), 615–640.

Michalides, Marvin, et al., 2023a. Analyzing current challenges on scaled agile development of physical products. Procedia CIRP (ISSN: 22128271) 119, 1188–1197. http://dx.doi.org/10.1016/j.procir.2023.02.188.

Michalides, Marvin, et al., 2023b. Why companies scale agile development of physical products: An empirical study. In: Chakrabarti, Amaresh, Singh, Vishal (Eds.), Design in the Era of Industry 4.0, Volume 3. In: Smart Innovation, Systems and Technologies, vol. 346, Springer Nature Singapore, Singapore, ISBN: 978-981-99-0427-3, pp. 1163–1174. http://dx.doi.org/10.1007/978-981-99-0428-0_95.

Müller, Johannes, et al., 2024. Success factors for measuring agile process changes and their metrics. In: Bitran, Iain, Conn, Steffen, Mitsis, Alex, Ritala, Paavo, Torkkeli, Marko, Trabelsi, Meriam (Eds.), Proceedings of the XXXV ISPIM Innovation Conference. ISBN: 978-952-65069-6-8, pp. 1–18.

Object Management Group, 2025. OMG systems modeling language (SysML). URL: https://www.omg.org/sysml/. (Accessed 17 April 2026).

Paetzold, Kristin, 2022. Data and information flow design in product development. In: Krause, Dieter, Heyden, Emil (Eds.), Design Methodology for Future Products: Data Driven, Agile and Flexible. Springer International Publishing, Cham, ISBN: 978-3-030-78368-6, pp. 201–218. http://dx.doi.org/10.1007/978-3-030-78368-6_11.

Paetzold-Byhain, Kristin, Michalides, Marvin, Weiss, Stefan, 2026. A conceptualization of agility: Utilization and future research for the development of mechatronic systems. Systems (ISSN: 2079-8954) 14 (1), http://dx.doi.org/10.3390/systems14010028.

Pascual, Romain, et al., 2024. Formal foundations of consistency in model-driven development. In: International Symposium on Leveraging Applications of Formal Methods. Springer, pp. 178–200.

Peffers, Ken, et al., 2007. A design science research methodology for information systems research. J. Manage. Inf. Syst. 24 (3), 45–77.

Reussner, Ralf, et al., 2023. Consistency in the view-based development of cyber-physical systems (convide). In: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion. MODELS-C, IEEE, ISBN: 979-8-3503-2498-3, pp. 83–84. http://dx.doi.org/10.1109/MODELS-C59198.2023.00026.

Rinker, Felix, 2025. Agile Multi-Domain Model Integration Management in Cyber-Physical Production Systems Engineering (Ph.D. thesis). Technische Universität Wien.

Rose, Louis M, et al., 2009. An analysis of approaches to model migration. In: Proc. Joint MoDSE-MCCM Workshop. pp. 6–15.

Schwaber, Ken, 2004. Agile Project Management with Scrum. Microsoft Press and Safari Books Online, Redmond, Wash. and Boston, Mass., ISBN: 073561993X, URL: https://learning.oreilly.com/library/view/-/9780735619937/?ar.

Spanoudakis, George, Zisman, Andrea, 2001a. Iconsistency management in software eengineering: Survey and open research issues. In: Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing Co Pte Ltd, pp. 329–380. http://dx.doi.org/10.1142/9789812389718_0015.

Spanoudakis, George, Zisman, Andrea, 2001b. Inconsistency management in software engineering: Survey and open research issues. In: Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals. World Scientific, pp. 329–380.

Steinberg, Dave, et al., 2008. EMF: Eclipse Modeling Framework. Pearson Education.

Verein Deutscher Ingenieure e. V., 2019. Design of Technical Products and Systems - Part 1: Model of Product Design. Berlin.

Verein Deutscher Ingenieure e. V., 2021. Development of Mechatronic and Cyber-Physical Systems. Berlin.

Vitruv Team, 2024. Vitruv: View-based engineering framework. https://github.com/vitruv-tools. (Accessed 28 November 2025).

Voelk, Thomas Alexander, Dehghani, Razieh, Klippert, Monika, Pfaff, Felix, Stolpmann, Robert, Nowak, Konstantin, Koziolek, Anne, Albers, Albert, 2025a. Structuring inconsistency situations in engineering of cyber-physical systems - a description template proposal. Procedia CIRP (ISSN: 22128271) 136, 49–54. http://dx.doi.org/10.1016/j.procir.2025.08.011.

Voelk, Thomas Alexander, et al., 2025b. Consistently inconsistent: AI-analyzed patterns in inconsistency-situations of cyber-physical systems development. In: 2025 IEEE International Symposium on Systems Engineering. ISSE, IEEE, http://dx.doi.org/10.1109/ISSE65546.2025.11370100.

Voelk, Thomas Alexander, et al., 2025c. Developing cyber-physical systems as entrepreneurs: Agile development in startups and SMEs. Eur. Conf. Innov. Entrep. (ISSN: 2049-1050) 20 (1), 785–793. http://dx.doi.org/10.34190/ecie.20.1.3804.

Voelk, Thomas Alexander, et al., 2025d. Misunderstand me correctly - comprehensibility in interdisciplinary collaboration. Proc. Des. Soc. (ISSN: 2732-527X) 5, 2441–2450. http://dx.doi.org/10.1017/pds.2025.10258.

Wanderley, Fernando, Silva, António, Araujo, João, Silveira, Denis S, 2014. SnapMind: A framework to support consistency and validation of model-based requirements in agile development. In: 2014 IEEE 4th International Model-Driven Requirements Engineering Workshop. MoDRE, IEEE, pp. 47–56.

Weidmann, Nils, Kannan, Suganya, Anjorin, Anthony, 2021. Tolerance in model-driven engineering: A systematic literature review with model-driven tool support. arXiv preprint arXiv:2106.01063.

Weiss, Stefan, Paetzold-Byhain, Kristin, Michalides, Marvin, Pendzik, Martin, Scharold, Franziska, Stoiber, Lino, 2023. Agile Entwicklung physischer Produkte 2023. Technische Universität Dresden, http://dx.doi.org/10.25368/2023.213.

Wohlin, Claes, Runeson, Per, Höst, Martin, Ohlsson, Magnus C, Regnell, Björn, Wesslén, Anders, et al., 2012. Experimentation in Software Engineering, vol. 236, Springer.

Yeman, Robin J., Malaiya, Yashwant K., 2024. A systematic literature review of the application of agile applied to large-scale, safety-critical, cyber-physical systems. In: 2024 IEEE International Conference on Data and Software Engineering. ICoDSE, IEEE, pp. 13–18.