Explainability in Automated Cross-Domain Model-Driven Brake System Development

Nathan Hagel*, Johannes Mäkelburg‡, Claus Hammann*, Thomas Weber*,
Thomas Alexander Völk†, Francesco P. Urbano†, Patrick Grycz†, Katharina Bause†,
Minakshi Kaushik*, Vincenzo Scotti*, Akhila Bairy*, Maike Schwammberger *,
Maribel Acosta‡, Albert Albers†, Anne Koziolek*, Tobias Düser†
*KASTEL, Karlsruhe Institute of Technology, Karlsruhe, Germany
†IPEK, Karlsruhe Institute of Technology, Karlsruhe, Germany
{firstname.lastname}@kit.edu
‡Data Engineering, Technische Universität München, München, Germany
{firstname.lastname}@tum.de

***Abstract*—Modern engineering systems often require collaboration across multiple domains, each using different models and tools. By implementing automated processes for keeping consistency, changes propagate across these models and tools, affecting the work of various stakeholders and teams. However, an explanation of these automatically propagated changes is often required. This paper presents a case study using heterogeneous models in the development and validation of a cross-domain automotive brake system. Our case study comprises a Brake Specification Model, a Computer-Aided Design Model, a Simulation Model as well as their corresponding metamodels, and a model of a real-world test bench for brake validation. We also present a set of change and explainability scenarios that occur during the development and validation process of a cyber-physical brake system. We use these scenarios to highlight the explainability requirements and challenges which should be addressed by any explainability solutions and approaches for cross-domain engineering and validation processes in the cyber-physical system context. By providing such models and metamodels, as well as the change and explainability scenarios that use them, our work will aid researchers in validating their approaches and in investigating those development challenges which arise from cross-domain consistent cyber-physical systems development.**

***Index Terms*—explainability, automation, modeling, consistency, metamodel, view-based modeling, consistency preservation**

I. Introduction

Cyber-Physical System (CPS) development is characterized by cross-domain collaboration, where multiple models and artifacts are used to describe systems which need to be at least partly explainable to stakeholders from different domains. Therefore, CPS development faces the challenge of explainability, just as do all domain involving collaborative development.

The success of cross-domain CPS collaboration depends on multiple teams and different roles, ranging from requirement engineers and product managers to software developers and mechanical or electrical engineers. Interestingly, the different domains involved here are not orthogonal. For example, the specification of a CPS under development will exhibit semantic overlap because different models and artifacts used all actually represent the same information, just likely in different ways. By the same token, wherever one stakeholder makes a change to a model or artifact, all other models or artifacts representing the same information must also be changed, to keep everything consistent and to prevent contradictions. This challenge remains one of the major activities in engineering complex systems [1], and it is here that automation can help. By automatically applying changes to models or artifacts, we can reduce the need for communication and increase the efficiency of the development process. However, automating consistency to solve the inefficiency challenge, as e.g., in the VITRUVIUS approach [2], just raises another challenge of keeping the consistency explainable. In other words, explainability must be ensured not only for the changes that were automatically applied to keep the models consistent but also after this propagation, for the original change itself for other stakeholders and teams.

In this paper, we present a case study to facilitate research on cross-domain collaboration. We extracted and refined our case study from the context of the *CONVIDE* [3] research project, the domains of computer science, mechanical engineering, and electrical engineering join forces to investigate challenges in model-based engineering arising from consistency, i.e., the absence of contradictions between the different models [3]. Our case study revolves around the development of a brake system, focusing on the testing phase. The engineering challenges involved in building a brake system are solved, evident by the existence of functioning brakes. However, the collaboration between the different engineering disciplines, also involving explainability engineers, is an open and worthwhile challenge. We provide an example from state of the art on what needs to be improved [3]. We here propose our case study to the community, to facilitate discussion and the progressive

This work is funded by the Topic Engineering Secure Systems of the Helmholtz Association (HGF), by KASTEL Security Research Labs, Karlsruhe, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1608 - 501798263, and by the Innovation Campus for Future Mobility (www.icm-bw.de). Thanks to our textician.

development of approaches and tools, aiming at improving this collaboration. We describe the different engineering domains involved, as well as the models, techniques, and workflows employed, and how we integrated them in our case study.

The remainder of this paper is organized as follows: In section II, we describe the context of our case study, i.e., the brake system development, focusing on the testing process and the roles involved therein. Section III details the case study by presenting the involved models, techniques, and workflows employed to develop and especially test a brake system. In Section IV we describe the scenario and extract the explainability requirements. Afterwards, we discuss the case study and outline planned future work in Section V, outline existing approaches to include explainability for CPS development in Section VI, and provide information about the replication package in Section VII. We conclude this paper in Section VIII.

II. Brake system Development and Testing Process

The development and testing of brake systems involves several roles, c.f. Figure 1, which contribute with their domain expertise to build safe brake systems. These roles are typically not limited to a single person, but rather each role is composed of a team of specialists. These specialists each work within their own epistemic framework, relying primarily on their implicit knowledge and expertise [4]. Making this knowledge accessible is a current challenge in engineering [1] in general, and is also present in brake system development [5], posing a direct requirement to explainability mechanisms to the process.

Engineering complex cyber-physical systems, therefore, requires collaboration both within and between different roles. Regarding testing, development-accompanying validation is needed, as well as traditional release testing, which must also be conducted and documented across domains [6]. Furthermore, it is not guaranteed that the roles or teams belong to the same company, leading to further challenges regarding fast feedback cycles and change propagation across these organizational boundaries [7]. When a change is propagated or automatically applied by consistency mechanisms, and also when a new change is applied, explainability challenges arise due to information needs from other domains [8]. The state-of-practice involves complex supply chains, only on whose end the original equipment manufacturer integrates the different parts. However, as they all work on the same product (in our example, a brake system for a vehicle), there is a semantic overlap of information between the views that each role uses. To efficiently make engineering decisions, the different roles often require knowledge from other domains, posing a challenge to explainability in the development processes.

In the following paragraphs, we describe a simplified version of this process and the roles that participate in it to highlight the challenges regarding explainability that must be addressed through improvements in processes and tools. While the description here focuses on key roles for clarity, it is essential to note that in a real-world setting, this process involves a multitude of additional stakeholders, each requiring a different kind of explanation [9]. These include, but are not limited to, project managers, procurement and sales departments, quality assurance, production system development and various domain experts. Their interactions and dependencies significantly influence how requirements are interpreted, refined, and ultimately modeled.

a) Product Manager: The *product manager* is responsible for product features and specifications that are relevant to both external and internal stakeholders. Examples include, for instance, the number of mounting holes in a brake disc and their diameters. The vehicle manufacturer may define the requirement of this information, so an external stakeholder with a requirement specification in, e.g., a simple textual format or a table, can satisfy it. The brake system product manager translates these requirements into a *product specification model*. Information in this *product specification model* is directly relevant, although it may not always be presented in the most technical manner, for both the construction engineer and the validation engineer.

b) Construction Engineer: The *Construction Engineer* is concerned with the design of the brake system. This includes, in particular, 3D modeling using tools like Creo Parametric [10]. The construction engineer makes engineering decisions to match requirements from the product specification model, but also from external stakeholders, regulations, or quality attributes. Some of the parameters managed by the product manager in the product specification model are therefore also represented in the construction engineer's domain. However, the 3D model usually includes more parameters than given by the product manager and also might include relations between parameters to automatically adjust the 3D shape of the brake system and its components.

c) Simulation Engineer: This role involves implementing simulation scenarios to test the brake system and engineering decisions made by the construction engineer in various scenarios. Simulation tools, such as Dyna4 [11] or Simulink [12], are used here. The simulation runs are based on scenarios like a full braking event of the brake system, e.g., from 60 km/h down to a standstill at 0km/h. Based on the desired insights expected to be gained, different real-world scenarios and parameters are defined to test the brake system in various environments and scenarios. The characteristics of the simulated brake system are mainly influenced by the 3D model and the materials defined by the construction engineer. Parameters like the environment, activities, or vibrations of the vehicle, however, are solely present in the simulation engineer's models.

d) Validation Engineer: The main task of the validation engineer is to validate engineering decisions made by the construction engineer and simulated by the simulation engineer in a real-world or close-to-real-world scenario. In this case study, the validation engineer uses a specialized test bench and tries to replicate or extend scenarios from the simulation engineer. The results of these tests are then compared with the simulation results and the requirements of the brake system. The test bench configuration is therefore based on the simulation models, while the brake system that is tested represents the construction engineer's models.

During the development of a brake system, the different roles interact to keep the information that semantically overlaps of the models consistent. However, without well-designed modeling tools and automated workflows, e.g., for consistency preservation and explainability, developing like that is inefficient and may lead to delayed projects or even failure when decisions are made on false assumptions or missing information. For that reason, we propose in the following the (simplified) metamodels of such an engineering and validation process together with a set of scenarios, which highlight challenges and requirements for explanability approaches. These resources and scenarios can be used to validate new approaches.

III. Brake System Case Study

The development process of a cyber-physical system such as a brake system can undergo different phases following different development activities in its lifecycle, eventually leading to the testing and validation of the previously only digital design on a physical test bench [13] [14]. Most of the models involved in the engineering process are only represented in standalone applications that are mostly connected by the implicit knowledge unique to each of the involved stakeholders. Changes in design parameters are not easy to manage, as the effect of a design change is not trivial to translate to its change in functional behavior [15]. Currently, engineers are relying on their implicit knowledge on objects, e.g., solid structures in Computer Aided Design (CAD), and how they affect changes in other parts of the overall system. While this approach works in ad-hoc settings in practice, it impedes the development when the complexity of the system to build increases. Additionally, right now, engineers typically do not model their generated objects, as for single-domain systems with limited complexity, implicit knowledge management is feasible. Yet, this approach lacks scalability, as different domain experts do not share their own knowledge even if they are closely collaborating [1], impeding the development of more complex systems.

To highlight challenges in cyber-physical systems development, we created simplified metamodels to share with the community, based on the actual models and tools used in brake system development and validation. Additionally, in Section IV, we present a set of example scenarios that highlight explainability challenges currently limiting the efficiency of development processes.

We deducted the following three metamodels, which are available in the replication package in a PDF format, as well as an Eclipse Modeling Framework (EMF) [16] metamodel:

A. Brake Specification Metamodel

The brake specification metamodel in Figure 2 is used to create a product model by product managers and procurement managers. Its semantics are also the basis for any stakeholder communication. It describes a *Brakesystem* as a set of *Brake-Components* with their specific product features and attributes. Some of these values are also represented in other models of the development and validation process, like the 3D model or the simulation model, and must therefore be managed.

B. CAD Metamodel

For The metamodel for 3D modeling we oriented us at known CAD tools like Creo Parametric [10] as a basis. See Figure 3 for a concrete 3D model of a braking system from the scenario of this case study, created in Creo Parametric. It defines the concepts and meta-classes required to create 3D shapes. The base class, which groups the 3D information, is the *Part* metaclass. It is composed of a *Sketch*, which is a set of simple *SketchElements*. To make 2D sketches 3-dimensional, a *Feature* is a pipeline of *Features* concatenated, such as extrusion. This metamodel, compatible with modeling frameworks like EMF, serves as a basis for further analysis and validation of solutions for improving explainability in cross-domain engineering. A model from a concrete CAD tool can also be transformed into a syntax, which is directly deducted from this metamodel.

C. Simulation Metamodel

For this case study, we simplified the simulation models into a concrete syntax that shows workflow runs similar to an activity diagram. We based our simplification on tools like Dyna4 [11] or MATLAB / Simulink [12]. A model from a concrete scenario is illustrated in Figure 5. A simulation starts with a start block. Then three different node types can be selected: Activity, Parameter Setting, and Parameter Measurement. Parameter settings are used to, e.g., define the environment. Measurement nodes define which parameters of the system must be measured during the simulation. Activities describe actions like accelerating or slowing down, i.e., braking, the system.

In addition to the metamodels described above, we present simplified, concrete example models from the scenarios of this case study. These example models are used to further specify problem scenarios, as defined in Section IV. The example models are part of the replication package or can be easily created using the presented metamodels.

a) Brake system model: The concrete brake system model is based on a validated brake system from the test bench and the 3D models. However, some of the product-related information is synthetic. It features three elements, a brake disc, brake pads, and the brake caliper, with product-relevant information, c.f. Figure 6.

b) 3D model: The 3D model is taken from Creo Parametric [10], c.f. Figure 4 and Figure 7. It is composed of three main parts, as in the brake system model. Depending on the concrete use case, it can be modeled using the proposed metamodel.

c) Simulation Model: The last model we modeled is the Simulation Model. It is represented as an extended UML Activity Diagram and represents the execution logic of a

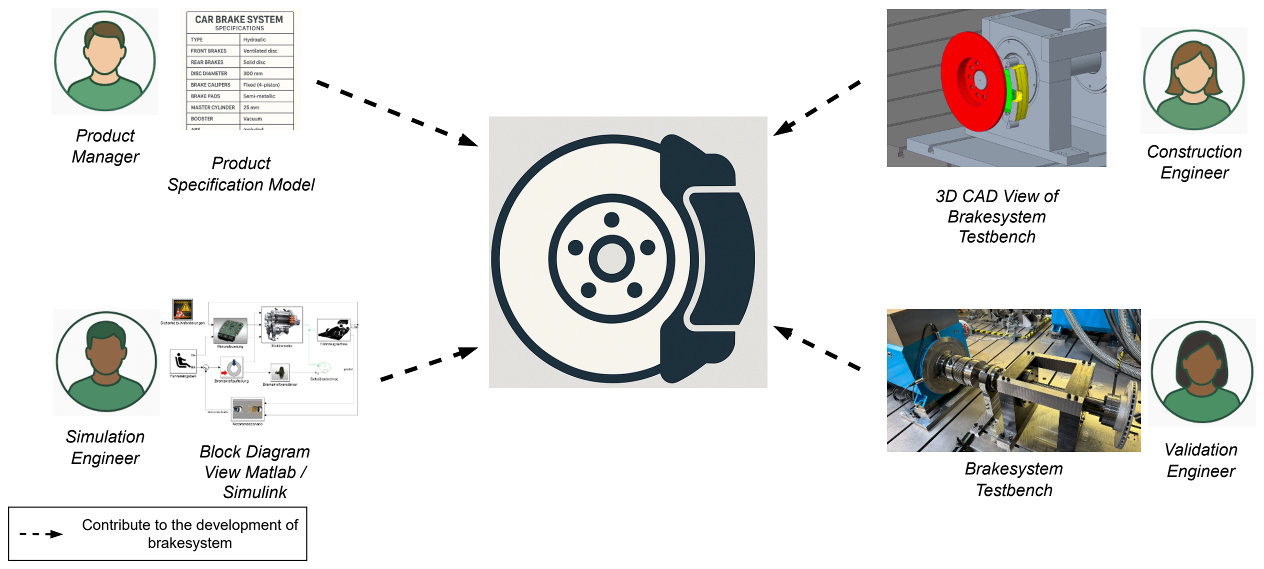
Product Manager
Product Specification Model
CAR BRAKE SYSTEM
3D CAD View of Brakesystem Testbench
Construction Engineer
Simulation Engineer
Block Diagram View Matlab / Simulink
Brakesystem Testbench
Validation Engineer
Contribute to the development of brakesystem

Fig. 1. Brake system development and testing process (simplified) - involved roles and views on the brake system and respective models.

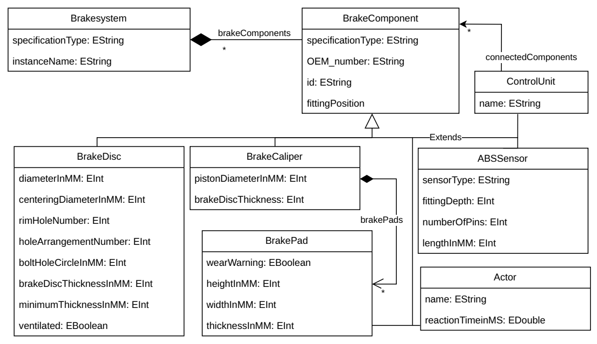

Fig. 2. Metamodel for brake specification model (simplified)

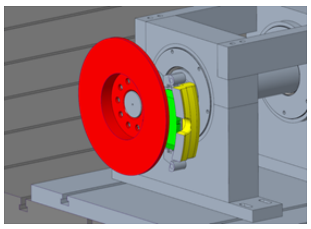

Fig. 3. Front view, smaller brake disc (previous generation).

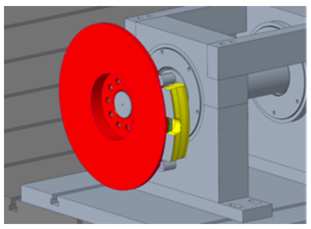

Fig. 4. Front view, larger brake disc (new generation).

full braking scenario on a test bench. The model begins by activating the sensor system to measure the vehicle speed and to configure the test environment, including setting the ambient temperature to -15 degrees Celsius. After the environment has been initialized, the test bench engine accelerates the simulated vehicle to a constant speed of 40 km/h. At this steady speed, the Simulation Model sets motor vibrations to 35μm and triggers the measurement of brake torque. The scenario concludes with the brake controller applying a brake pressure of 100 bar, thereby simulating a full brake application under defined environmental and mechanical conditions.

IV. Scenario Description

In the proposed scenarios, which we extracted from the work on our research project *CONVIDE* [3] and which build upon the roles and models defined beforehand and are also present in current industry case studies [17], we look at design changes in a brake disc.

A. *Inconsistency Caused by Partial Design Reuse*

During development, a design engineer decides to test a new brake disc of a newly developed generation. In the descriptive notation of the model of systems generation engineering (SGE) [18], most of the system was carried over from a previous generation of the same disc. While the main principle of the brake disc should be kept, only the attribute of the diameter should be changed in the CAD model (c.f. Figure 3 and Figure 4). This example could apply to the adoption of an established brake system to a slightly bigger vehicle than originally planned. In our example, the brake caliper and brake pad are not changed in the Product Specification Model, as they are currently under development for the new generation. Still, a first assessment of the new brake setup should be provided to generate new knowledge for the current system generation.

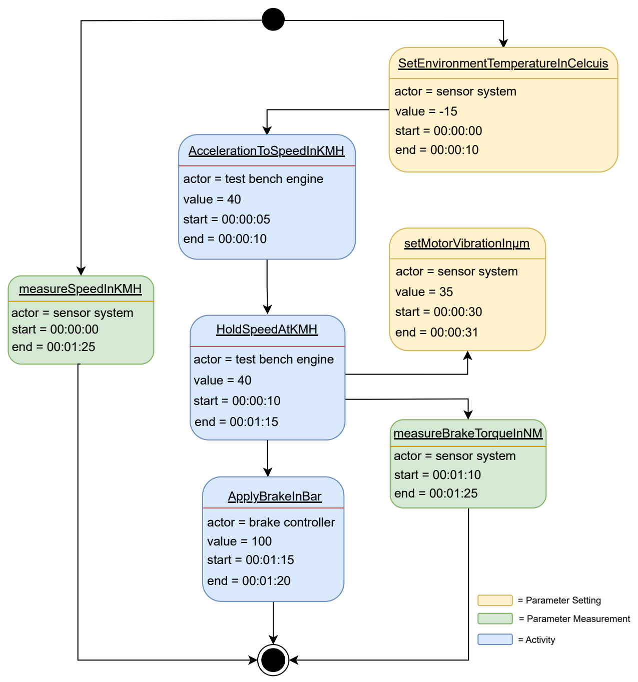

Fig. 5. Simulation Model (simplified).

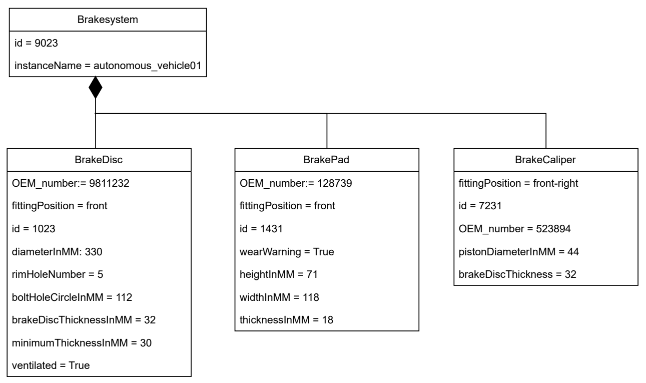

Fig. 6. Concrete Brake specification model managed by Product Manager

A validation engineer is given the task of adjusting the current test bench configuration model for the last system generation (c.f. Figure 9 and Figure 10). He changes the brake disc according to the old test bench. By changing the attribute of the old brake disc, the previously sufficient construction space becomes insufficient, resulting in an intersection of the brake disc and the caliper (c.f. Figure 7 and Figure 8). If this is not recognized in the virtual assembly (or not correctly displayed), this problem is only realized in the physical assembly of the test bench.

These kinds of problems are typically manageable by a validation engineer, who follows a step-by-step workflow to adjust

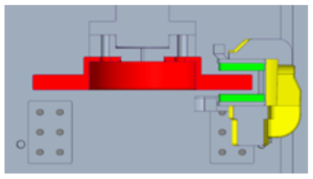

Fig. 7. Top view, smaller brake disc (previous generation).

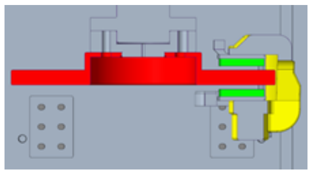

Fig. 8. Top view, larger brake disc (new generation).

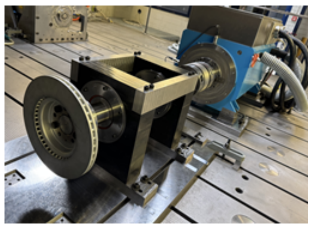

Fig. 9. Side View Testbench.

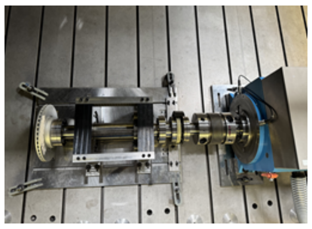

Fig. 10. Top View Testbench.

the designed elements and prevent unwanted design changes. However, in highly connected systems, such changes might be overlooked, especially in time-critical situations. Alternatively, if the change is not overlooked, other engineers could question the necessity of this change. Without automated explainability mechanisms, two teams could diverge as the necessity for a strictly needed design change was not adequately explained, propagated, or even available.

B. Parameter Inconsistency Between Specification and Simulation

During the development, the product manager updates the product specification model to account for a new generation of brake pads with a slightly increased thickness of 18 mm, as specified in Figure 6. The intention of this change was to improve the vibration behavior by enhancing the structural properties of the brake pads. However, the Simulation Model, created from the previous product specification, was reused without adapting its parameters. In particular, the motor vibrations were again set to 35μm while performing a full brake to measure the brake torque (c.f. Figure 5). As the Simulation Model does not explicitly reference brake pad properties, the impact of the geometry remained unnoticed. Therefore, the simulated behaviour appeared valid, but during the physical validation, the measured brake torque differed from the expectations.

This scenario highlights how reusing simulation artifacts without verifying compatibility with updated system parameters can lead to inconsistencies that manifest only during late validation phases. However, the manifestation does not necessarily result in an understandable explanation for why this problem occurred, despite the information being present; it was not properly propagated and made available.

C. *Untracked Material Substitution in 3D and Simulation*

To address supply chain issues in brake pad procurement, the product manager decides to switch the brake pad material from cast iron to a carbon-ceramic compound. This change is appropriately communicated to the construction engineer, who adjusts these changes in the 3D material metadata of the brake pad. However, the Simulation Model (c.f. Figure 5) remained unchanged, and the simulation engineer is unaware of this substitution. Therefore, the simulation engineer continues to use the old brake pad material in the simulations. As a result, the simulations yield an acceptable result. Once the new brake pads were provided, they were directly sent to the validation engineer's test bench, where new physical tests were performed. The results of these tests show that the brake torque and heat dissipation tests failed due to the changed material.

This scenario highlights the challenge of managing changes that affect multiple disciplines when the consistency between models is maintained manually.

D. *Requirements*

Following the framework by Bersani et al. [19], we elicit and categorize the following explainability requirements from the scenarios, the case study itself, and interviews with stakeholders working on this process from mechanical engineering, validation, and software engineering. The framework foresees a classification on four levels (L1 - no explainability, L2 - recognition-of-explainability, L3 - local-explainability and L4 - global explainability):

Change Origin Tracing (L3): The system must show the original change (e.g., design engineer changes diameter) and the automatically applied propagated changes in dependent models.

Propagation Path Transparency (L3): Engineers need a step-by-step explanation of which consistency rules triggered which downstream modifications (e.g., increased disc diameter - insufficient caliper space - test bench model adjusted).

Counterfactual Explanations (L3/L4): The system should show what would happen if the change had not been propagated (e.g., "If the diameter change was ignored, simulation torque results would mismatch physical validation.").

Provenance Tracking (L2): Every property (e.g., brake pad thickness = 18 mm) should be traceable back to its source: product manager's specification, regulatory requirement, or test data.

Reasoning Tracking (L2/L3): The system must record the justification of parameter settings (e.g., "Thickness increased to improve vibration behavior").

Responsibility Mapping (L2): The model should explicitly indicate the role responsible for a parameter (product manager, development engineer, simulation engineer).

Stakeholder Specific Explainability (L4): The system must show explanations customized for each role in the development process (i.e., using a view-based approach).

V. Discussion

The case study in this paper highlights a set of challenges that must be addressed by development processes and modeling tools to improve cross-domain model-based development and engineering, emerging from research, but also prevalent in industry.

a) Automated Cross-Model Impact Analysis: During attribute changes in any of the models with semantic overlap with other models in the development process, the effects of a change must potentially be reflected across several other models with heterogeneous owners, such as a product manager or a development engineer. However, as often standalone non-connected tools are used, without model management and explainability capabilities across domains, these changes cannot be appropriately reflected, or the original changing role is not aware of the impact of this change. Therefore, tools should address this challenge, e.g., by implementing automated change-model impact analysis as a foundation for explainability analysis and features.

b) Configuration Synchronization Across Physical and Virtual Systems: In this case study, the real-world test bench can evolve, as can the virtual models or code of the brake system. This challenge is very relevant in the development of cyber-physical systems. Especially on virtual models, solutions exist to notice changes and propagate them appropriately [2]. However, for changes, e.g., to a physical test bench, different solutions are needed to ensure that the evolution of the physical systems always reflects the virtual models as closely as possible, while notifying the virtual models of changes to the physical system. Development processes and tools should be aware of this potential discrepancy and reflect that in explainability mechanisms to increase awareness of it.

c) Organizational and Tooling Barriers: The proposed scenarios expose organizational silos. In cross-domain engineering processes, where different teams collaborate, communication, comprehension and awareness must be enhanced. However, the specialized software that is used is often standalone and not part of a larger tool suite [17]. This makes the integration into cross-domain or cross-organization models and artifact repositories more difficult. Tools and processes must address this challenge by providing interfaces to synchronize models and model data formats, e.g., shared repositories, to improve consistency preservation and provide an interface to ensure explainability approaches can access and observe the required information.

VI. Explainability approaches and solutions – Opportunities

Our case study shows that consistency maintenance alone is not sufficient, and must be complemented with explainability; Through explanations stakeholders can be enabled to understand *why* the changes occurred and *how* they propagated. One way of doing that would be through **provenance tracking**, ensuring that each artifact carries its origin, rationale, and responsible stakeholder. Such information enables engineers to trace back how a parameter value was introduced and why it was changed. Provenance has been shown to increase accountability [20].

Existing approaches, such as VITRUVIUS, maintain consistency across models through rule-based synchronization and they rarely expose the reasoning behind automated updates. However, logging **rule firings** and showing **propagation paths** would make adaptations transparent, e.g., when disc diameter changes cause interference in CAD models. It would allow the stakeholders to see not only *that* a change was applied but also *why*. The MAB-EX (Monitor, Analyse, Build, and Explain) framework [21] facilitates this by monitoring changes, analyzing their effects, and generating explanations of both rule firings and propagation paths.

In addition, **counterfactual reasoning** is needed to support design decisions. Models should allow "what-if" comparisons between alternatives, such as different pad materials or dimensions. One approach to generating counterfactual explanations is the self-explanation model developed by Fey et al. [22].

Moreover, explanations need to be tailored to the perspectives of different stakeholders. Prior research indicates that contrastive, **user-aligned explanations** are more effective [23], which in our context means requirement-oriented justifications for product managers, parameter-level differences for simulation engineers, and so on. Schwammberger and Klös provide a method to extract user-specific explanations from system models [9]. Tailoring explanations towards diverse stakeholders also means to consider the optimal **timing of explanations**. Bairy and Fränzle [24] show that for different users and different situational contexts, explanations provided at specific time points can improve explanation understanding significantly.

VII. Replication Package

For this case study, we provide all data in our replication package [25]. The replication package contains all metamodels in EMF format and as a PDF. We additionally provide concrete example models from the scenarios.

VIII. Conclusion

We present a case study on explainability scenarios in cross-domain engineering with consistency keeping. The presented work describes a collaboration with other domains involved in our research project CONVIDE [3]. Within the work at hand, we present a first investigation into explanation issues emerging from (semi-)automatic consistency keeping to propose a starting point for the community. From the perspective of mechanical engineering, the situation represented through our case study is self-evident and the existing solutions work, but show a lack of scalability due to the lack of automation.

The automation introduced by (semi)-automated consistency keeping introduces other issues, namely the challenge of explaining changes. Therefore, this change management is difficult to understand and model sufficiently enough so that all stakeholder needs for information and explainability align during development, which we support by providing this case study.

References

[1] A. Albers, A. Koziołek, T. A. Völk, M. Klippert, F. Pfaff, R. Stolpmann, and S. E. Schwarz, "Identification of inconsistencies in agile cps engineering with formula student," in *ISPIM Innovation Symposium*. The International Society for Professional Innovation Management (ISPIM), 2024, pp. 1–15.

[2] H. Klare, M. E. Kramer, M. Langhammer, D. Werle, E. Burger, and R. Reussner, "Enabling consistency in view-based system development — The Vitruvius approach," *Journal of Systems and Software*, vol. 171, p. 35, Jan. 2021.

[3] R. Reussner, I. Schaefer, B. Beckert, A. Koziolek, and E. Burger, "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, 2023, pp. 83–84.

[4] E.-M. Grote, C. Koldewey, S. E. Schwarz, R. Dumitrescu, and A. Albers, "Enabling better systems through better teams: 27 role profiles for engineering advanced systems," in *2025 IEEE International Conference on Engineering, Technology, and Innovation (ICE/ITMC)*. IEEE, 2025, pp. 1–9.

[5] M. W. Spekker, K. Vlajic, Y. Cacheux, W. Liang, C. Thümmel, M. Kuebler, A. Albers, and T. Düşer, "Using xr for validation in cross-domain, agile product development," in *ISPIM Innovation Conference*. The International Society for Professional Innovation Management (ISPIM), 2025, pp. 1–19.

[6] J. Cederbladh, A. Cicchetti, and J. Suryadevara, "Early validation and verification of system behaviour in model-based systems engineering: A systematic literature review," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 3, pp. 1–67, 2024.

[7] T. Weber and S. Weber, "Model everything but with intellectual property protection - the deltachain approach," in *MODELS '24: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*. Association for Computing Machinery (ACM), 2024, p. 49–56, 46.23.03; LK 01.

[8] T. Weber, N. D. Kuder, T. A. Völk, J. Schneider, S. Weber, and A. Koziolek, "Enhancing production workflows by leveraging bpmn to model inconsistencies — an experience report," *The Journal of Object Technology*, vol. 24, no. 2, pp. 1–14, 2025, 46.23.01; LK 01.

[9] M. Schwammberger and V. Klös, "From specification models to explanation models: An extraction and refinement process for timed automata," in *Proceedings Fourth International Workshop on Formal Methods for Autonomous Systems (FMAS) and Fourth International Workshop on Automated and verifiable Software sYstem DEvelopment (ASYDE), FMAS/ASYDE@SEFM 2022, and Fourth International Workshop on Automated and verifiable Software sYstem DEvelopment (ASYDE)Berlin, Germany, 26th and 27th of September 2022*, ser. EPTCS, M. Luckcuck and M. Farrell, Eds., vol. 371, 2022, pp. 20–37. [Online]. Available: https://doi.org/10.4204/EPTCS.371.2

[10] P. Inc., "Creo Parametric," https://www.ptc.com/en/products/creo/parametric, accessed: 2025-07-10.

[11] Vector Informatik GmbH, "Dyna4," https://www.vector.com/gb/en/products/products-a-z/software/dyna4/, accessed: 2025-08-26.

[12] The MathWorks, Inc., "Simulink," https://mathworks.com/products/simulink.html, accessed: 2025-08-26.

[13] Verein Deutscher Ingenieure e. V., "Design of technical products and systems: Model of product design," Berlin, 11.2019.

[14] ——, "Development of mechatronic and cyber-physical systems," Berlin, 2021.

[15] S. Matthiesen, C. Zimmerer, L. Pähler, and P. Grauberger, *Wissen in der Produktentwicklung – Grundlagen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2024, pp. 1–56. [Online]. Available: https://doi.org/10.1007/978-3-662-68986-8_1

[16] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: eclipse modeling framework*. Pearson Education, 2008.

[17] T. A. Voelk, K. Nowak, F. Pfaff, R. Dehghani, T. Düser, A. Koziolek, and A. Albers, "Consistently inconsistent: Ai-analyzed patterns in inconsistency-situations of cyber-physical systems development," in *2025 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2025, in press.

[18] A. Albers and S. Rapp, *Model of SGE: System Generation Engineering as Basis for Structured Planning and Management of Development*. Cham: Springer International Publishing, 2022, pp. 27–46. [Online]. Available: https://doi.org/10.1007/978-3-030-78368-6_2

[19] M. M. Bersani, M. Camilli, L. Lestingi, R. Mirandola, M. Rossi, and P. Scandurra, “A conceptual framework for explainability requirements in software-intensive systems,” in *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, 2023, pp. 309–315.

[20] J. Singh, J. Cobbe, and C. Norval, “Decision provenance: Harnessing data flow for accountable systems,” *IEEE Access*, vol. 7, pp. 6562–6574, 2019.

[21] M. Blumreiter, J. Greenyer, F. J. C. Garcia, V. Klös, M. Schwammberger, C. Sommer, A. Vogelsang, and A. Wortmann, “Towards self-explainable cyber-physical systems,” in *22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019, Munich, Germany, September 15-20, 2019*, L. Burgueño, A. Pretschner, S. Voss, M. Chaudron, J. Kienzle, M. Völter, S. Gérard, M. Zahedi, E. Bousse, A. Rensink, F. Polack, G. Engels, and G. Kappel, Eds. IEEE, 2019, pp. 543–548. [Online]. Available: https://doi.org/10.1109/MODELS-C.2019.00084

[22] G. Fey, M. Fränzle, and R. Drechsler, “Self-explanation in systems of systems,” in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, 2022, pp. 85–91.

[23] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences,” *Artificial Intelligence*, vol. 267, pp. 1–38, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370218305988

[24] A. Bairy and M. Fränzle, “Enhancing multi-user experience: Optimizing explanation timing through game theory,” in *Intelligent Technology for Future Transportation*, A. Razminia and D. H. Nguyen, Eds. Cham: Springer Nature Switzerland, 2025, pp. 106–117.

[25] N. Hagel, J. Mäkelburg, C. Hammann, T. Weber, T. Völk, F. Urbano, K. Bause, P. Grycz, M. Kaushik, V. Scotti, A. Bairy, M. Acosta, A. Albers, A. Koziolek, and T. Düser, “Supplementary material for Explainability in Automated Cross-Domain Model-Driven Brake System Development,” 2025. [Online]. Available: https://github.com/vitruv-tools/Brake-System-V-SUM-Case-Study