Towards Change Impact Prediction for Consistent Multi-Domain Evolution

Andreas Domanowski[1,*], Dirk Neumann[2], Philip Ochs[2] and Uwe Aßmann[1]

[1]Technical University Dresden, Chair of Software Technology, 01187 Dresden, Germany
[2]Karlsruhe Institute of Technology, Test, Validation and Analysis of Software-Intensive Systems, 76131 Karlsruhe, Germany

Abstract

The development of cyber-physical systems involves heterogeneous engineering domains. This requires development artifacts to be consistent across the involved domains to prevent contradictory artifacts threatening system reliability. Although reactive consistency restoration mechanisms address this issue, they lack transparency towards developers regarding the cross-artifact impact of their changes, i.e., whether their changes trigger consistency-restoring operations or result in ripple effects like restoration cascades or even loops. The lack of knowledge regarding the impact of changes on consistency renders planning the system's evolution unpredictable, as the consequences of modifications cannot be reliably anticipated, thereby decreasing development efficiency. In this work, we envision a concept for impact analysis of change operations before such changes are actually applied. Therefore, we analyze consistency specifications, automatic repair operations, and (meta-)models to classify change operations, laying the groundwork for predictive impact analysis regarding the impact on consistency.

Keywords

Consistency, Model Evolution, Change Impact Control, Change Impact Analysis

1. On Change Impact Awareness in Heterogeneous Systems

Cyber-physical systems involve heterogeneous models and multi-team engineering environments. Therefore, they require explicit consistency management to maintain coherent and non-contradictory system representations. Reactive consistency restoration mechanisms (RCRMs) [1, 2, 3] address this by automatically applying transformations that restore consistency in heterogeneous models in reaction to specific changes. Although this effectively solves consistency maintenance, it hides the impact of repairing operations from stakeholders like developers, since RCRM act reactively and do not inform developers about the impact of their changes proactively before such changes are applied.

Therefore, planning the evolution of a cyber-physical system becomes increasingly challenging. Stakeholders of the development process have knowledge of neither the quality nor the quantity of the impact of their changes on other domains. They do not know whether changes trigger none, single, multiple cascading, or even circular repair operations. However, the teams working in different domains usually have different strategies to include repairing operations within their development process models. Since developers cannot anticipate the effect their changes have on other domains at all, every change bears the risk of forcing other teams to react, possibly faster or differently than their preferred workflow implements. When such changes occur in an unplanned way, the planning and communication overhead decreases development efficiency, therefore leading to greater organizational costs. We therefore argue that consistency-impact prediction is required, informing developers about the impact of their changes prior to applying them.

This work contributes an assessment concept for the relevance of changes to consistency *prior to their actual application*. With this assessment, we solve the lack of knowledge regarding the impact of change operations in this regard. We classify the relevance of each type of change to consistency by

Joint Proceedings of the STAF 2026 Workshops: AgileMDE, GCM, ICMM, LLM4SE, TTC. Rennes, France, June 29-July 3, 2026
*Corresponding author.
andreas.domanowski@tu-dresden.de (A. Domanowski); dirk.neumann@kit.edu (D. Neumann); philip.ochs@kit.edu (P. Ochs); uwe.assmann@tu-dresden.de (U. Aßmann)
0000-0003-3479-661X (A. Domanowski); 0000-0003-1999-639X (D. Neumann); 0009-0009-4224-3530 (P. Ochs); 0000-0002-3513-6448 (U. Aßmann)
© 2026 Copyright © 2026 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-26

CEUR-WS.org/Vol-4338/ICMM-short1.pdf

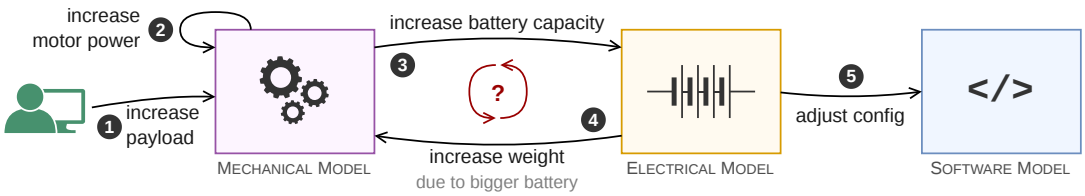


Figure 1: Development of a drone with cross-domain reactive consistency restoration rules.

analyzing (i) reactive consistency restoration specifications, since they describe cross-domain relations; (ii) metamodels to derive types of change operations to their instances, and (iii) model instances, since consistency restoration can depend on the current model state. Additionally, we analyze relevant changes further to predict whether their application itself might potentially trigger automatic consistency restoration or bears the risk of circular consistency restoration cascades. With such a classification of changes and knowledge about possible cascades of changes, stakeholders of the development process are able to plan the evolution of a cyber-physical system in a controlled way, taking cross-domain repairing operations into account. Our approach enables predictive impact analysis of consistency repairs across heterogeneous models even before changes are applied. Teams are now able to prioritize changes, e.g., according to minimal impact on other domains or by incorporating the capabilities or resources of other teams. The knowledge gained leads to more efficient planning, less friction regarding integration efforts between development teams, and therefore to a more efficient development process.

2. Problem Illustration with Cross-Domain Drone Development

As a motivating example, consider the development of a drone, as illustrated in Figure 1. There is one team responsible for each of the mechanical model (CAD), the electric/electronics model, and the control software. While the mechanical team works in longer design iterations due to CAD models requiring proper validation and prototypical manufacturing, the electrical team operates in shorter iterations due to only having to simulate circuits. Finally, the software team employs agile methodology. In general, the cyber-physical system requires all the development artifacts to be consistent to form a sound and coherent description of the system without contradictions. For instance, the motors need to be powerful enough for the drone and its payload to take off and fly with a battery strong enough to power the drone for a given time frame, while the software components send appropriate control signals suited for the hardware. In this scenario, several reactive consistency restoration rules were identified, e.g, increasing the power of the required motor in reaction to an increased total mass; increasing the minimum battery capacity if motor power increases; and increasing configuration parameters like applied thrust required for takeoff in the software model based on mass.

Changes in one domain trigger operations to restore consistency across others. Consider the drone design being close to start-of-production, and a mechanical engineer proposes increasing the payload. From the perspective of the mechanical engineer, this appears to be a minor modification. However, this evolution step requires higher motor power, which in turn demands increased battery capacity, possibly resulting in a heavier battery. The increased battery mass further increases total weight, potentially triggering additional motor power adjustments and, in turn, adjustment of the battery again. Finally, the software model must adapt: the thrust required for takeoff is updated based on the new mass and electrical power properties. Without impact awareness, this seemingly small change may require extensive consistency restoration in other domains and delay the start of production. In contrast, modifications such as changing the color of the drone frame in the CAD model do not trigger consistency repairs and therefore do not affect cross-domain consistency. Therefore, this change could be applied without cross-team coordination.

With predictive change-impact awareness, developers gain early insight into whether a modification will remain local to their domain or propagate across the system. This allows teams to schedule consistency-relevant changes deliberately and coordinate during critical development phases.

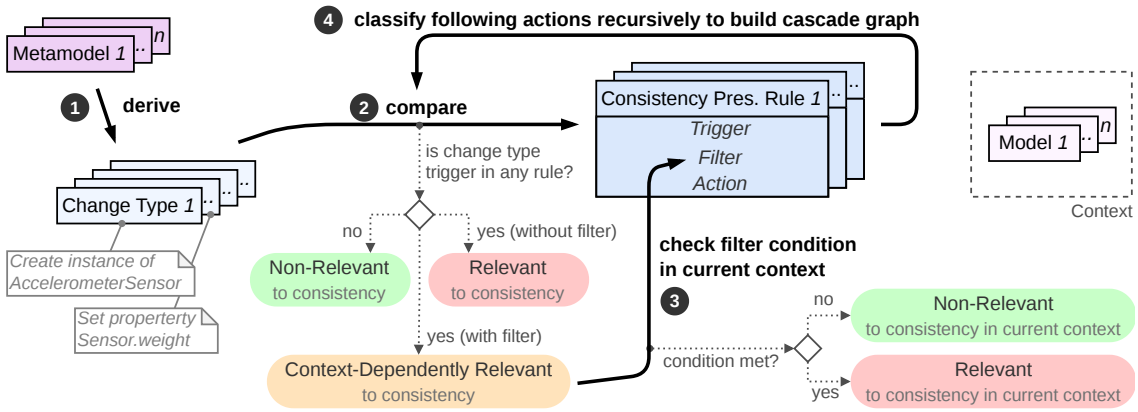


Figure 2: Classification of change types in regard to relevance to cross-domain consistency.

3. Change Impact Analysis in RCRM: State of the Art

Cyber-physical systems spanning multiple domains are defined by different, heterogeneous metamodels. Their semantic or partially dependent overlap makes consistency maintenance paramount. Reactive consistency restoration mechanisms [1, 2, 3] and their realizations, e.g., with Triple Graph Grammars (TGG) [4] or bidirectional transformations [5], specify the required repair operations for restoring consistency and execute them automatically in reaction to changes. Nevertheless, none of these approaches provides predictive feedback about the impact of individual changes prior to their execution.

Furthermore, this work is based on concepts from the field of change impact analysis (CIA). For an overview, we refer to general [6] or code-based [7] systematic literature studies. CIA aims at determining the consequences of planned or actual modifications to development artifacts. Approaches for multi-domain CIA either describe the change impact after the actual change happens or prior to its application. There are approaches for multi-domain CIA that, after actually executing changes in one model, inform about required changes to other models [8, 9]. Several approaches aim at the definition of dependencies or rules, for checking what other models or elements are actually affected by certain changes [10, 11], or reason about the impact of specific repair operations, not the actual initial change [12]. Although the latter computes impacted artifacts based on dependencies [12], it neither classifies the actual changes nor quantifies possible ripple effects.

In summary, existing work focuses on either repairing inconsistencies after they occur or analyzing the (possibly transitive) dependencies of changes. However, none of the approaches realizes transitive multi-domain CIA prior to the execution of changes. In turn, our envisioned approach enables the prediction and quantification of possibly transitive impacts of consistency repairing operations across multiple models prior to change execution.

4. Predictive Assessment of Change Impact on Consistency

This section presents our approach for predictive impact analysis with respect to consistency implications. It is based on the analysis of reactive consistency restoration specifications and repair operations, metamodels, and model instances. We analyze each metamodel that describes the different domains and derive the possible classes of atomic changes. In the following, we classify whether a type of change operation is relevant to consistency as specified in consistency preservation rules, is not relevant, or is relevant depending on the context. Moreover, we derive whether repairing operations themselves trigger cascading operations or even circular cascades. Figure 2 illustrates our approach.

To calculate the impact of change operations, knowledge about the set of possible change operations for each domain is crucial. In a model-based environment, metamodels describe domains or their parts. Research on the types of changes that can be applied to EMOF-based models [1] shows that all changes can be expressed by a small set of atomic operations, e.g., the creation and deletion (and

possibly replacement) of model elements, attributes, or references. In a first step, we therefore analyze which classes of change are actually possible for models conforming to each given metamodel, e.g., the instantiation of a metaclass or manipulation of an attribute (Step ① in Figure 2). While composite changes usually carry more intent of change, we nevertheless focus on atomic changes in a first step in this work to provide a minimal, domain-independent foundation for the analysis.

Accordingly, we relate these classes of change to the consistency specification (Step ②), e.g., the reactive consistency restoration rules [1]. We identified that such rules, as employed by approaches discussed above, consist of three major building blocks: the *trigger* they react to (i.e., the type of change as described above), optional boolean application conditions (*filters*), and finally an *action* segment that describes how consistency is restored automatically. Whenever a type of change is not a trigger for any of the consistency restoration specifications, we can already classify it as *non-relevant* to consistency. Furthermore, whenever a type of change is specified as a trigger, we classify it as *context-dependently relevant* to consistency if further filters are described, and *relevant* to consistency if there are no filters.

Types of change that trigger conditionally need to be analyzed further (Step ③). This analysis requires *context*-sensitivity, i.e., taking filter conditions of consistency restoration rules into account as well as the current state of the models. These conditions usually consist of queries and Boolean checks like the existence or nonexistence of certain structures in the models themselves or between them, e.g., correspondences between elements that trace consistency relations. Examples for such filters are *checks* and *matches* in the Vitruvius approach [1] or *constraints* and *application conditions* in TGGs [4]. After executing and evaluating the queries, we can finally classify whether the consistency restoration rule in question is relevant to consistency *in the current context* or not, depending on the context-sensitive evaluation results. Note that filters may be arbitrarily complex and therefore challenging to compute. In this work, we only consider decidable and terminating computations.

In the previous steps, we computed the quality of the impact, i.e., whether the operation affects consistency or not. By further examination of the consistency restoring *actions* (Step ④), we are able to gain further insights regarding the impact of (possibly circular) repair cascades. Such actions consist of model repair operations and possibly further computations. While the further computations can be challenging or even infeasible to analyze, the repair operations are, in turn, instances of the types of changes already identified and therefore analyzable with our process. In general, we can deduce which of the specified repair operations are triggers for further consistency restoration rules. This relation captures that a certain initial change not only triggers one or multiple consistency restorations, but that they, in turn, trigger or context-dependently might trigger following repairs. We can construct a cascade graph based on these relations to visualize and further reason about the potential impact. Such a graph requires every rule to be present at most once to ensure the termination of the construction process due to the possibility of cycles. Cycles can exist between the actions of a rule and the invocation of further rules. This needs to be communicated to developers to raise awareness of this impactful situation. While it is not possible to analyze the context of the resulting consistent models after application without simulating the changes, we are still able to reason about *potential* chains of impact to other models. With our analysis, we can provide information regarding the possible depth of the cascading operations, i.e., how many other consistency restoration rules might be triggered. Moreover, we can identify possible circular reaction chains, cf. Figure 1, by checking whether a trigger itself is in the chain of triggered repairing operations. Besides knowing about how many consistency-restoring operations might be required in how many other models, knowledge about such circular chains is crucial to enable stakeholders to adequately plan the evolution of a system.

5. Conclusion and Future Work

We presented a vision of a concept for calculating the quality and possible quantity of the impact that the consistency repairing operations and rules of RCRM have on models. Based on the possible types of changes for a metamodel and analysis of rules defined by the RCRM, we compute the individual impact of such rules by classifying whether they actually have an impact, and if so, how deeply they possibly

cascade and whether there are possible circular reaction chains. In future work, we will evaluate our approach with a comparison to the virtualization of changes, i.e., simulation. The partial virtualization of changes and impact observation for gaining exact knowledge regarding the context-sensitive impact, as well as the application of other predictive change impact techniques like traditional static analysis or traceability analysis, promise to be well-suited complementary enhancements of our approach. Lastly, we will investigate the applicability of our approach in consistency-aware multi-domain variability realized with delta modeling [13], due to the large conceptual overlap.

Acknowledgments

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

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] H. Klare, M. E. Kramer, M. Langhammer, D. Werle, E. Burger, R. Reussner, Enabling consistency in view-based system development — the vitruvius approach, Journal of Systems and Software 171 (2021).
[2] J.-C. Bach, A. Beugnard, J. Champeau, F. Dagnat, S. Guérin, S. Martínez, 10 years of model federation with openflexo: Challenges and lessons learned, MODELS '24, 2024.
[3] C.-C. RaŢiu, W. K. G. Assunção, E. Herac, R. Haas, C. Lauwerys, A. Egyed, Using reactive links to propagate changes across engineering models, Software and Systems Modeling 24 (2025).
[4] A. Anjorin, E. Leblebici, A. Schürr, 20 years of triple graph grammars: A roadmap for future research, Electronic Communications of the EASST 73 (2016).
[5] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, J. F. Terwilliger, Bidirectional transformations: A cross-discipline perspective, in: Proceedings of the 2nd International Conference on Theory and Practice of Model Transformations, ICMT '09, 2009.
[6] M. Kretsou, E.-M. Arvanitou, A. Ampatzoglou, I. Deligiannis, V. C. Gerogiannis, Change impact analysis: A systematic mapping study, Journal of Systems and Software 174 (2021).
[7] B. Li, X. Sun, H. Leung, S. Zhang, A survey of code-based change impact analysis techniques, Software Testing, Verification and Reliability 23 (2013).
[8] K. Müller, B. Rumpe, A model-based approach to impact analysis using model differencing, Electronic Communications of the EASST 65 (2014).
[9] A. Demuth, R. Kretschmer, A. Egyed, D. Maes, Introducing traceability and consistency checking for change impact analysis across engineering tools in an automation solution company: An experience report, in: ICSME, 2016.
[10] T. Pfeiffer, C. Apostolov, P. Savarino, T. Dickopf, Concept and implementation path for model-based impact analysis in product development (2023). The 33rd CIRP Design Conference.
[11] S. Nejati, M. Sabetzadeh, C. Arora, L. C. Briand, F. Mandoux, Automated change impact analysis between sysml models of requirements and design, in: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Association for Computing Machinery, 2016.
[12] A. Keller, H. Schippers, S. Demeyer, Supporting inconsistency resolution through predictive change impact analysis, in: Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation, Association for Computing Machinery, 2009.
[13] D. Neumann, T. Pett, I. Schaefer, Towards consistency preservation for multi-domain variability modeling, SPLC-A '25, Association for Computing Machinery, 2025.