Concurrent Editing for Multi-Model Environments

Benedikt Jutz
Karlsruhe Institute of Technology
Karlsruhe, Germany
benedikt.jutz@kit.edu

Abstract

Context: Complex systems, such as cyber-physical systems, are developed by multiple developers working on multiple models. To build the actual system, the models need to be consistent to each other. Automatically preserving consistency in such modeling environments introduces further conflicts that occur when developers concurrently edit their models. In addition, enforcing consistency at all times, when it only needs to be guaranteed at build time, may unnecessarily restrict developers. Objective: In our thesis, we plan to develop editing techniques for multi-model environments that provide strict consistency, as well as fast response times under relaxed consistency. Further, we plan to integrate suitable mechanisms for restoring consistency. Method: We adapt the transaction concept from databases, by formalizing consistency-preserving edits as transactions, and implementing transactional editing algorithms. We also incorporate consistency relaxations into the transaction model, and investigate the use of Conflict-Free Replicated Data Types (CRDTs). For restoring consistency, we investigate how to incorporate model repair techniques, as well as using suitable, inconsistency-showing views. Evaluation: We propose to evaluate the performance of the scheduling algorithms through creating a benchmark from existing graph models and edits. Additionally, we plan to evaluate the inconsistency views, and the switch between transactions and CRDTs, with case studies.

CCS Concepts

• **Software and its engineering → Collaboration in software development**; **Model-driven software engineering**; • **Information systems → Database transaction processing**.

Keywords

inter-model consistency, concurrent editing, conflict handling, transactionality

ACM Reference Format:
Benedikt Jutz. 2026. Concurrent Editing for Multi-Model Environments. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026), October 04–09, 2026, Málaga, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3837062.3839548

1 Introduction

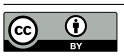
Context. The development of complex systems is inherently collaborative. Developers from multiple domains need to cooperate and work on the same system. Model-Driven Engineering (MDE) offers two ways to deal with this challenge, namely Multi-View Modeling (MVM) [7], and collaborative Model-Driven Software Engineering (MDSE) [9].

MVM is about providing different views and their viewpoints of a system under development. As different views may share the same concerns, consistency management becomes an issue. A number of different paradigms exists for MVM. In our research, we assume that: (i) views are constructed from underlying models, following a *projective approach* [2], (ii) the underlying models are kept separate [1], (iii) and that consistency is handled on the model level through checking for the fulfillment of consistency relations, and through automated *consistency preservation*, i.e. restoring consistency relations upon changes to the underlying models.

Collaborative MDSE combines MDSE with collaborative software engineering. One part thereof is how to concurrently edit models, and how to prevent or resolve conflicts between edits on models [9]. Industrial tools for MVM, e.g. for E/E-engineering in mobility domains [15], make use of these techniques.

Problems and Research Questions. When existing models are used to build domain-specific views, and consistency is automatically preserved, we encounter additional problems with concurrent edits. We demonstrate these problems on three example scenarios, derived from a multi-model system for the development of a brake system [16]. A brake system requires multiple components, e.g. a caliper, brake hoses, discs etc. These are specified by a *product manager*, who maintains the product specification model. A *construction engineer* designs a CAD model for the brake system, working on both the product specification, but also on personal expertise. A *simulation engineer* then creates test scenarios in e.g. SimuLink, that ensure that the brake meets safety requirements. Consistency checking rules have been defined between different models, e.g. to ensure safety requirements [25], and there are supposed to be preserved with Consistency Preservation Rules (CPRs), that produce consistency-preserving changes for input changes. Developers interact with the multi-model environment by submitting changes to the models they work on. Figure 1 shows a sketch of the brake system multi-model environment.

Scenario 1 - Conflicts between Concurrent Edits. Assume that the product manager and construction engineer edit the specification model and CAD model respectively, at the same time. The product manager changes the diameter of the brake disc, while the construction engineer also changes the diameter of the cylinder in the CAD model that corresponds to the disc. Assume now that CPRs are applied, which modify models in response to changes in other models, e.g. in the Vitruvius approach [24]. CPR application would add a consistency-preserving change to the edit of the product



This work is licensed under a Creative Commons Attribution 4.0 International License.
MODELS Companion 2026, Málaga, Spain
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2903-4/26/10
https://doi.org/10.1145/3837062.3839548

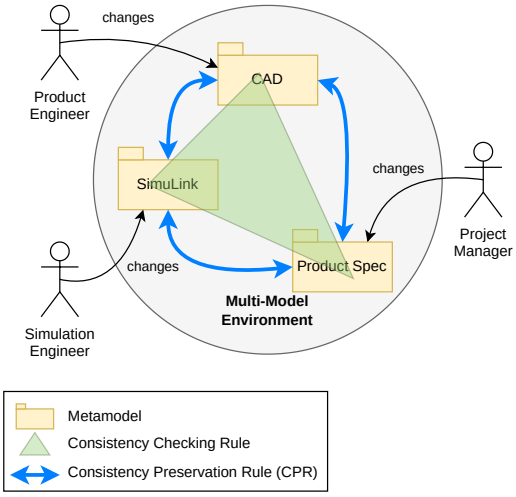


Figure 1: A Multi-Model Environment for development of a brake system. Based on [16].

manager, which would also modify the cylinder diameter, causing a conflict.

Even when user- and CPR-generated changes are not in direct conflict, their joint application can violate consistency constraints. Assume that the product manager and construction engineer execute two coordinated changes. The product manager reduces the number of pistons for the brake caliper, which reduces the caliper size in the CAD model through CPR application. Meanwhile, the construction engineer switches to a less heat-resistant material for the brake pad. Both changes do not directly conflict with each other, but they influence the brake disc surface temperature *temp*, which is derived from an constraint in the simulation model. When both changes are applied, *temp* now exceeds the temperature limit $temp_{max}$ set by the simulation engineer. One of the changes now has to be reverted, to maintain consistency.

Scenario 2 - Restrictions due to Strict Consistency. If we treat edits to multi-model environments as *transactions*, and guarantee the Atomicity, Consistency, Isolation, Durability (ACID) criteria for them, we prevent the problems above: *Atomicity* ensures that changes are applied in full or undone completely, *Consistency* constraints are always preserved (otherwise the transaction is undone), and *Isolation* means that conflicting edits are not permitted. However, these guarantees may cause further problems in turn.

For example, the product manager changes multiple components in the product catalog. One change, e.g. setting the brake disc diameter, conflicts with another change produced by CPRs from an edit originating from the construction engineer. Strict atomicity demands that the product manager has to abort their entire change, and change all components again. The same scenario can also be seen as a problem with strict isolation: Had both parties been aware of each others changes, this conflict could have been avoided. Finally, strict consistency cannot be enforced at all times. This may be because automated consistency preservation fails due to technical issues [23], or because consistency requires user input [18], which cannot be provided at the moment.

Scenario 3 - Resolving Inconsistencies. The problems mentioned for Scenario 2 motivate the relaxation of ACID guarantees. However, when inconsistencies are distributed across multiple models, presenting and resolving them becomes more challenging. For example, the consistency constraint about $temp_{max}$ is computed from information in the product specification and CAD models. When only using their domain-specific views, the stakeholders are unable to see the causes of the inconsistency. Additionally, consistency repair mechanisms now need to deal with cross-model consistency and automated consistency preservation.

From the exemplar scenarios we described above, we now derive our Research Questions (**RQs**):

- **RQ$_1$** How can a multi-model environment provide *ACID* when multiple developers edit its models?
- **RQ$_2$** When and how should consistency guarantees be *relaxed* in favor of permitting more concurrency?
- **RQ$_3$** What techniques can be applied to *eventually restore consistency relations* in the presence of relaxed consistency guarantees?

Feasability through CRC embedding. We conduct our research as part of the Collaborative Research Center (CRC) 1608 "Convide" [32]. The goal of the CRC is enable more functional/more complex Cyber-Physical Systems (CPSs) by enhancing CPS development with Views and Consistency Maintenance. Conceptually, the CRC is grounded in the idea of Virtual Single Underlying Models (V-SUMs), an instance of Model Federation [24]. Within the CRC, we conduct our research within the project B02, which investigates concurrent editing of multi-model environments while maintaining inter-model consistency [31]. In doing so, we adapt database concepts, most importantly the transaction concept. While transactions already provide the ACID guarantees we desire (see Scenario 1), transferring them into CPS development is non-trivial. Besides working with different data models and less complex consistency constraints in general [20], single- and multi-database systems do not handle consistency preservation across multiple models (see section 3). A defining feature of CRCs is cooperation across multiple projects and disciplines. As part of our project and for answering **RQ$_1$** and **RQ$_2$**, we work together with database researchers. We also work with other projects on the relaxation of ACID guarantees and repairing inconsistencies.

2 Expected Contributions

We plan to make three contributions:

- **C$_1$** We provide transactional guarantees for developers who work on multi-model environments with consistency preservation. Specifically, we devise ACID-guaranteeing editing algorithms on multi-model environments, whose correctness we show on a new formal model for transactions on these environments. The formal model provides richer semantics than traditional database models, and explicitly incorporates consistency relations and restoration.
- **C$_2$** We contribute different ways to relax ACID guarantees when making changes to multi-model environments, thus permitting more concurrency. By distinguishing between user-generated and consistency-preserving changes, we want to enable developers to relax atomicity. Through integrating

Conflict-Free Replicated Data Types (CRDTs), we relax the strict isolation requirement.

C3 We provide techniques that allow modelers to restore inter-model consistency relations in presence of inconsistencies. One part is to represent information about inter-model consistency violations with dynamically constructed views, which do not need to be specified eforehand. These show inconsistent model parts annotated with information about conflicts and inconsistencies. The second part are procedures to globally restore consistency, e.g. through automated planning.

3 Related Work

We see two areas of related work: concurrent editing techniques in other multi-model and multi-view environments, and existing model editing techniques.

Concurrent Editing in Multi-View and Multi-Model Environments. Openflexo, an implementation of the model federation approach [4], makes conflict handling an issue of its users. They write consistency and conflict detection rules in the FML modeling language [17]. DesignSpace supports a limited form of consistency preservation between attribute values [30], and collaborative consistency checking for one model type at a time [27]. Consistency relations across multiple models are not supported, however. Other approaches for multi-model development do not consider concurrent editing [35].

The MetaEdit+ language workbench combines fine-granular locking for conflict prevention with Version Control System (VCS) integration for versioning [22], but it does not provide consistency guarantees in the sense of declarative consistency. Jjodel is another web-based modeling workbench, providing tooling to customize viewpoints of one model, validate models using a OCL-like language, and enabling users to react to model changes with Event-Condition-Action (ECA) rules [6]. It does not handle multiple models, or inter-model consistency out of the box.

Dávid et al. have tackled the problem of inconsistency management with different modeling formalisms from a process perspective [11]. Building on a process model for describing development activities, they describe inconsistencies in terms of abstract model properties, and propose different patterns on how to manage them, i.e. prevention or reordering of activities Our work can be seen as orthogonal to this, as we focus on concrete editing techniques, rather than processes.

Other Model Editing Techniques. Within collaborative modeling in general, CRDTs are finding widespread usage, e.g. in the lowkey and MOIRAI frameworks [10, 29] Both approaches aim to guarantee *eventual consistency*, i.e. equality of models when receiving the same updates, and do not provide the consistency guarantees we desire.

Model Versioning is another technique for concurrent modeling. It is based on the development of specialized VCS for modeling artifacts with branching and merging capabilities. Typically, they work on the model structure instead of the text-based representation [5]. Sharbaf et al. proposes a taxonomy of model merging techniques [34], which highlights that merges across multiple models are not considered relevant right now. Fu et al. handle conflicts and violation of static constraints when merging changes in MVM [14]. Dam et al. have already implemented a three-way model merging algorithm that resolves inconsistencies introduced during the merge process, by way of a search algorithm [8]. Neither approach accounts for multiple, underlying models, or for three or more users merging models, which may occur if developers from so many domains need to combine their changes.

4 Proposed Approach

Overview. We achieve **C1** through adapting locking protocols such as Two-Phase Locking (2PL) [12] for multi-model environments. This adaption can be formally proven correct through with the use of a *transaction model* [36], which requires the definition of operations, their semantics, conflicts and reliability criteria for correct undo. We define such a model with support for consistency checking and CPRs application to preserve consistency in transactions. We also investigate the use of Copy-on Write (COW) data structures to concurrently apply CPRs to multi-model environments, which allows the parallel construction of transactions with consistency preservation.

For **C2**, we relax isolation through letting developers share views using CRDTs [33], providing strong eventual consistency instead of serializability through 2PL. To relax atomicity, we want to treat transactions with CPR applications as nested and flexible transactions. Nested transactions consist of subtransactions and permit some subtransactions to fail, without an abort of the entire transaction [28]. Flexible transactions further allow failing transactions to be replaced with alternative transactions to execute [38]. Additionally, in the case of failure, we want developers to decide which consistency-breaking changes still should be accepted, possibly on a parallel development branch.

For **C3**, we want to assign information about inconsistencies and conflicts to developer-facing views. Suitable views could be those that developers already work with, or be explicitly constructed, e.g. derived from consistency constraints. In addition, we are developing a transformation of multi-model environments with consistency checking and preservation rules into planning problems expressed in the Planning Domain Definition Language (PDDL) [13]. The produced plans from automated planners can then be translated back into operations applicable on the initial environment, thus restoring all consistency relations.

Technical Platform. We implement our approach in the Vitruvius framework, which implements the V-SUM idea [24]. It provides a change metametamodel that we reuse for the atomic operations in transactions, and supports both consistency checking and preservation through two dedicated Domain-Specific Languages (DSLs), namely Reactions and Vitruv-OCL. This forms the technical foundation for implementing transactions with strict and relaxed ACID guarantees as required for item **C1** and item **C2**.

Views of the V-SUM can be edited remotely through an already existing web editor Figure 2, which connects to an existing server component [19]. Besides identity views directly on the underlying models, custom views can be defined with the NeoJoin view definition language [26]. We use the editor to support CRDTs, and NeoJoin to construct views with appropriate inconsistency information.

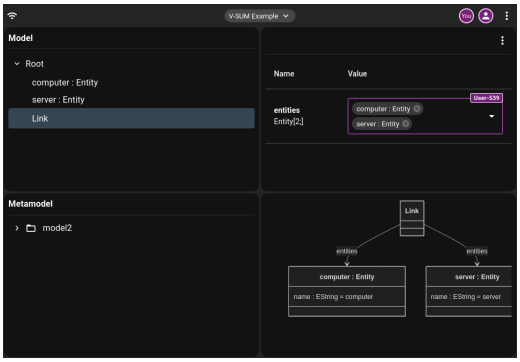


Figure 2: Screenshot of the Vitruvius Web Editor.

Risks and Ethical Issues. Possible ethical issues may occur when using human subjects for the evaluation. We assume that human participants are suitable to answer questions $Q_{2.2}$, and $Q_{3.1}$ that we define in our GQM-plan When planning user studies, we will contact the ethics committee at Karlsruhe Institute of Technology (KIT) [1], and request pre-study approval as required. We will also protect personal data in accordance with the General Data Protection Regulations (GDPR) of the European Union, following the guidelines provided by the Data Protection Staff Unit at KIT [2].

5 Evaluation Plan

Our evaluation is structured around this Goal-Question-Metric (GQM) [3] plan, where each goal contributes to one contribution:

Goal 1: Evaluate the Performance of Transactional Editing Algorithms for Developers concurrently working on Multi-Model Environments.

- $\mathbf{Q_{1.1}}$ How do different editing algorithms influence the throughput, latency, and acceptance rate of transactions applied on multi-model environments?
- $\mathbf{Q_{1.2}}$ How does applying CPRs on the server influence the performance?

Goal 2: Reduce delays in editing multi-model environments through editing algorithms with relaxed ACID guarantees.

- $\mathbf{Q_{2.1}}$ Compared to ACID transactions, by how much do relaxed transactions increase throughput and reduce latencies?
- $\mathbf{Q_{2.2}}$ Under which circumstances should developers share their views, instead of using transactions with isolation guarantees?

Goal 3: Resolve inconsistencies introduced through the relaxation of ACID guarantees.

- $\mathbf{Q_{3.1}}$ What information about inconsistencies aids developers in resolving them?
- $\mathbf{Q_{3.2}}$ What is the performance trade-off between attempting to resolve inconsistencies through consistency repair methods, and preventing them through ACID transactions?

[1] https://www.ethik.kit.edu/english/index.php
[2] https://www.dsb.kit.edu/english

Table 1: Timeline for tasks to execute for our thesis. Dates are given in Month/Year format.

Task	Start	End
Transaction Model ($\mathbf{C_1}$)	01/2025	07/2026
Transaction Manager ($\mathbf{C_1}$)	03/2026	12/2026
CRDT Integration ($\mathbf{C_2}$)	05/2026	03/2027
COW Semantics ($\mathbf{C_1}$)	01/2027	07/2027
Relaxing Atomicity ($\mathbf{C_2}$)	01/2027	09/2027
PDDL Translation ($\mathbf{C_3}$)	05/2026	05/2027
Inconsistency Views ($\mathbf{C_3}$)	09/2026	06/2027
Overall Evaluation ($\mathbf{C_1} - \mathbf{C_3}$)	07/2027	01/2028
Writing up	01/2028	07/2028

Validation Methods and Data Sources. To answer the questions we pose in our GQM plan, we will conduct case studies, and also construct a benchmark for applying transactions on multi-model environments. To evaluate the performance of different locking algorithms and relaxing atomicity ($Q_{1.1}, Q_{1.2}, Q_{2.1}$), we will develop a custom benchmark for applying transactions on multi-model environments. The underlying models and transactions are based on a graph and edit generator developed by Kegel et al. [21]. To determine the suitability of view sharing, and what inconsistency information should be provided ($Q_{2.2}, Q_{3.1}, Q_{3.2}$), we will conduct a series of case studies [37]. These case studies may also involve human subjects, i.e. our cooperation partners, or students with the appropriate knowledge.

In support of the evaluation, we will develop the required software components, such as prototypes of a transaction manager, that will be integrated into the Vitruvius Server component, and the web editor.

6 Current Status

At present, we have conducted the following work already. For $\mathbf{C_1}$, we have already defined the transaction model, and implemented a first transaction manager prototype that supports the Conservative Two-Phase Locking (C2PL) protocol, a variant of 2PL. For $\mathbf{C_2}$, we are also currently integrating CRDTs into the existing web editor. We have also started to translate inconsistent V-SUMs into automated planning problems for $\mathbf{C_3}$.

In the future, we plan to conduct the following research to complete our thesis. For $\mathbf{C_1}$, we are developing further locking algorithms based on 2PL, and integrate COW semantics into consistency preservation. For $\mathbf{C_2}$, we also still have to investigate the usage of nested and flexible transaction models to relax atomicity. For $\mathbf{C_3}$, we still have to derive specialized views for displaying inconsistency information. Table 1 shows our proposed timeline for finishing the remaining contributions and writing up the thesis.

Acknowledgments

Author B. Jutz is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1608 - 501798263.

References

[1] Moussa Amrani, Rakshit Mittal, Miguel Goulão, Vasco Amaral, Sylvain Guérin, Salvador Martínez, Dominique Blouin, Anish Bhobe, and Yara Hallak. 2024. A Survey of Federative Approaches for Model Management in MBSE. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems* (Linz Austria, 2024-09-22). ACM, 990–999. doi:10.1145/3652620.3688221

[2] Colin Atkinson, Christian Tunjic, and Torben Möller. 2015. Fundamental Realization Strategies for Multi-view Specification Environments. In *2015 IEEE 19th International Enterprise Distributed Object Computing Conference* (2015-09). 40–49. doi:10.1109/EDOC.2015.17

[3] Victor R. Basili. 1992. Software Modeling and Measurement: The Goal/Question/Metric Paradigm. hdl:1903/7538 http://hdl.handle.net/1903/7538

[4] Antoine Beugnard, Joël Champeau, Fabien Dagnat, Sylvain Guérin, and Salvador Martínez. 2024. 10 Years of Model Federation with Openflexo: Challenges and Lessons Learned. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems* (Linz Austria, 2024-09-22). ACM, 25–36. doi:10.1145/3640310.3674084

[5] Petra Brosch, Gerti Kappel, Philip Langer, Martina Seidl, Konrad Wieland, and Manuel Wimmer. 2012. An introduction to model versioning. In *International school on formal methods for the design of computer, communication and software systems*. Springer, 336–398.

[6] Antonio Bucchiarone, Juri Di Rocco, Damiano Di Vincenzo, and Alfonso Pierantonio. 2025. Modeling in Jjodel: Towards Bridging Complexity and Usability in Model-Driven Engineering. (2025). doi:10.1007/s10270-025-01324-y

[7] Antonio Cicchetti, Federico Ciccozzi, and Alfonso Pierantonio. 2019. Multi-View Approaches for Software and System Modelling: A Systematic Literature Review. 18, 6 (2019), 3207–3233. doi:10.1007/s10270-018-00713-w

[8] Hoa Khanh Dam, Alexander Egyed, Michael Winikoff, Alexander Reder, and Roberto E. Lopez-Herrejon. 2016. Consistent Merging of Model Versions. 112 (2016), 137–155. doi:10.1016/j.jss.2015.06.044

[9] Istvan David, Kousar Aslam, Sogol Faridmoayer, Ivano Malavolta, Eugene Syriani, and Patricia Lago. 2021. Collaborative Model-Driven Software Engineering: A Systematic Update. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, Fukuoka, Japan, 273–284. doi:10.1109/MODELS50736.2021.00035

[10] Istvan David and Eugene Syriani. 2023. Real-Time Collaborative Multi-Level Modeling by Conflict-Free Replicated Data Types. 22, 4 (2023), 1131–1150. doi:10.1007/S10270-022-01054-5

[11] István Dávid, Joachim Denil, Klaas Gadeyne, and Hans Vangheluwe. 2016. Engineering Process Transformation to Manage (In)Consistency. In *Proceedings of the 1st International Workshop on Collaborative Modelling in MDE (COMMitMDE 2016) Co-Located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2016), St. Malo, France, October 4, 2016* (2016) *(CEUR Workshop Proceedings, Vol. 1717)*, Henry Muccini, Ivano Malavolta, Sebastien Gerard, and Dimitris S. Kolovos (Eds.). CEUR-WS.org, 7–16. https://ceur-ws.org/Vol-1717/paper5.pdf

[12] K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger. 1976. The Notions of Consistency and Predicate Locks in a Database System. 19, 11 (1976), 624–633. doi:10.1145/360363.360369

[13] M. Fox and D. Long. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. 20 (2003), 61–124. doi:10.1613/jair.1129

[14] Yuhong Fu, Georg Grossmann, Karamjit Kaur, Matt Selway, and Markus Stumptner. 2026. Handling conflicts in consistency management of multi-level models in collaborative Digital Twins modelling environments. *Information and Software Technology* 198 (2026), 108237. doi:10.1016/j.infsof.2026.108237

[15] Vector Informatik GmbH. 2026. *PREEvision | Collaboration, Multi-User Operation, and Teamwork Features*. https://www.vector.com/int/en/products/products-a-z/software/preevision/collaboration-platform/

[16] 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, and Tobias Düser. 2025. Explainability in Automated Cross-Domain Model-Driven Brake System Development. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)* (2025). 204–211. doi:10.1109/ASEW67777.2025.00047

[17] Hiba Hnaini, Chahrazed Boudjemila, and Sylvain Guerin. 2025. Openflexo as a Model Management Platform for the MoM 2025 Satellite Configuration Challenge. In *2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 732–741. doi:10.1109/MODELS-C68889.2025.00116

[18] Bowen Jiang, Miriam Boss, Thomas Weber, Weixing Zhang, Mattias Ulbrich, and Anne Koziolek. 2026. CoCoPath: Concolic Exploration of Consistency-Preserving Paths. *Journal of Object Technology* 25, 3 (2026), 3:113–126. doi:10.5381/jot.2026.25.3.a9

[19] Benedikt Jutz, Thomas Weber, Arne Lange, Razieh Dehghani, Bowen Jiang, Martin Armbruster, Kevin Feichtinger, Nathan Hagel, Minakshi Kaushik, Lars König, Manar Mazkatli, Muhammad Asim Minhas, Dirk Neumann, Erik Burger, Anne Koziolek, and Ralf Reussner. 2025. Modeling the MoM 2025 Satellite Configuration Challenge with Vitruvius. In *2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (2025-10). 722–731. doi:10.1109/MODELS-C68889.2025.00115

[20] Benedikt Jutz, Zenon Zacouris, Thomas Weber, Maribel Acosta, Erik Burger, and Ralf Reussner. 2025. What Can We Learn from Other Domains – An Exploratory Process for the Extension of a Language Based on Cross-Domain Knowledge Transfer. In *2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (2025). IEEE, 755–761. doi:10.1109/MODELS-C68889.2025.00100

[21] Karl Kegel, Sebastian Götz, Ronny Marx, and Uwe Aßmann. 2024. A Variance-Based Drift Metric for Inconsistency Estimation in Model Variant Sets. *Journal of Object Technology* 23, 3 (2024), 1. doi:10.5381/jot.2024.23.3.a2

[22] Steven Kelly. 2018. Collaborative Modelling with Version Control. In *Software Technologies: Applications and Foundations*, Martina Seidl and Steffen Zschaler (Eds.). Springer International Publishing, Cham, 20–29. doi:10.1007/978-3-319-74730-9_3

[23] Heiko Klare. 2022. Building Transformation Networks for Consistent Evolution of Interrelated Models. doi:10.5445/KSP/1000138566

[24] Heiko Klare, Max E. Kramer, Michael Langhammer, Dominik Werle, Erik Burger, and Ralf Reussner. 2021. Enabling Consistency in View-Based System Development — The Vitruvius Approach. 171 (2021), 35. doi:10.1016/j.jss.2020.110815

[25] Tim Kräuter, Harald König, Adrian Rutle, Yngve Lamo, and Patrick Stünkel. 2023. Behavioral Consistency in Multi-Modeling. 22, 2 (2023), 2:1. doi:10.5381/jot.2023.22.2.a9

[26] Lars König, Tobias Stickling, Alexander Kocher, Hüseyin Kemâl Çakmak, Erik Burger, Veit Hagenmeyer, Anne Koziolek, and Ralf Reussner. 2026. Language Design of the NeoJoin View Definition Language. 25, 3 (2026), 14.

[27] Luciano Marchezan, Marcel Homolka, Andrei Blokhin, Wesley K. G. Assunção, Edvin Herac, and Alexander Egyed. 2024. A Tool for Collaborative Consistency Checking During Modeling. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24)*. ACM, New York, NY, USA, 655–659. doi:10.1145/3652620.3688558

[28] E.B. Moss. [n. d.]. *NESTED TRANSACTIONS: AN APPROACH TO RELIABLE DISTRIBUTED COMPUTING*. https://dl.acm.org/doi/book/10.5555/889868

[29] Léo Olivier, Marcos Didonet Del Fabro, and Sébastien Gérard. 2026. Hybrid Collaborative Modeling: Problem Analysis, Requirements, and Architectural Principles. *Journal of Object Technology* 25, 3 (2026), 3:351. doi:10.5381/jot.2026.25.3.a27

[30] Cosmina Cristina Rațiu, Wesley K. G. Assunção, Rainer Haas, and Alexander Egyed. 2022. Reactive links across multi-domain engineering models. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*. ACM, Montreal Quebec Canada, 76–86. doi:10.1145/3550355.3552446

[31] Ralf Reussner. 2025. *Convide – Research - Projects - Project Area B - Project B02*. https://www.sfb1608.kit.edu/316.php

[32] Ralf Reussner, Ina Schaefer, Bernhard Beckert, Anne Koziolek, and Erik Burger. 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)* (2023-10). 83–84. doi:10.1109/MODELS-C59198.2023.00026

[33] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. Conflict-Free Replicated Data Types. In *Stabilization, Safety, and Security of Distributed Systems*, Xavier Défago, Franck Petit, and Vincent Villain (Eds.). Vol. 6976. Springer, 386–400. doi:10.1007/978-3-642-24550-3_29

[34] Mohammadreza Sharbaf, Bahman Zamani, and Gerson Sunyé. 2023. Conflict Management Techniques for Model Merging: A Systematic Mapping Review. 22, 3 (2023), 1031–1079. doi:10.1007/s10270-022-01050-9

[35] Patrick Stünkel, Harald König, Yngve Lamo, and Adrian Rutle. 2021. Comprehensive Systems: A Formal Foundation for Multi-Model Consistency Management. 33, 6 (2021), 1067–1114. doi:10.1007/s00165-021-00555-2

[36] Gottfried Vossen. 1995. Database transaction models. In *Computer Science Today*, Gerhard Goos, Juris Hartmanis, and Jan Van Leeuwen (Eds.). Lecture Notes in Computer Science, Vol. 1000. Springer Berlin Heidelberg, Berlin, Heidelberg, 560–574. doi:10.1007/BFb0015267

[37] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Berlin Heidelberg. doi:10.1007/978-3-642-29044-2

[38] Aidong Zhang, Marian Nodine, Bharat Bhargava, and Omran Bukhres. 1994. Ensuring relaxed atomicity for flexible transactions in multidatabase systems. *ACM SIGMOD Record* 23, 2 (June 1994), 67–78. doi:10.1145/191843.191850