

Do Not Disturb

Proposing Approaches Towards Motivated Human-in-the-Loop for Less Disruptive Cross-Domain Interactive Repair

Miriam Boss

KASTEL- Institute of Information Security and
Dependability
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
boss@kit.edu

Ralf Reussner

KASTEL- Institute of Information Security and
Dependability
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
reussner@kit.edu

Friedrich Schadow

IAR - Institute for Anthropomatics und Robotics
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
friedrich.schadow@kit.edu

Kathrin Gerling

IAR - Institute for Anthropomatics und Robotics
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
kathrin.gerling@kit.edu

Abstract

In multi-domain systems development, resolving consistency conflicts often reaches an impasse: the resolution cannot be fully automated, but neither can an interactive consistency repair be completed with knowledge from one domain only. Unfortunately, many approaches forthcoming from research are disruptive to the workflow of the developers of the system, primarily because it is typically assumed that inconsistencies should be resolved by the domain causing the inconsistency. In this vision paper, we make a first effort to identify resolution processes that are better suited to support users with this complex task. Our method is to first classify interactive repairs along different design categories, so that we can explore existing interactive repair approaches and how they cover our proposed categories. Our results show that the current frameworks implement a naive approach likely leading to suboptimal results; therefore, we propose a collaborative and an uncertainty-based approach for performing interactive repairs, and measure these novel approaches for coverage against our design categories. Our results indicate potential to make interactive repair less disruptive, both to the resolving users and to other users of the system. We intend to implement these two new framework types and combine them in future work to evaluate their impact.

CCS Concepts

• **Software and its engineering** → **Consistency; Model-driven software engineering**; • **Human-centered computing** → *Computer supported cooperative work; HCI theory, concepts and models.*

Keywords

Model-driven engineering (MDE), Self Determination Theory (SDT), Cross-model consistency repair, Interactive resolution, User interaction, Human-in-the-loop

ACM Reference Format:

Miriam Boss, Friedrich Schadow, Ralf Reussner, and Kathrin Gerling. 2026. Do Not Disturb: Proposing Approaches Towards Motivated Human-in-the-Loop for Less Disruptive Cross-Domain Interactive Repair. 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/10.1145/3837062.3838682>

1 Introduction

During Cyber-Physical System (CPS) development of multi-view systems, the resolution of consistency conflicts unavoidably demands manual effort (e.g., due to cross-model refinement relations [19]) [15]. Moreover, in multi-disciplinary cooperations, such conflicts are likely to span domains, and where this is the case, the resolution will require knowledge from the source domain and from the target domain (cf. [31]). That presents a severe disruption in the development workflows of all parties involved, as currently all work is blocked until one person resolves this conflict, either with incomplete knowledge (risking errors) or by first acquiring the required knowledge, which either takes time of experts or is highly likely to be incomplete, as the experts were not consulted. This hinders the effective use of models for model-driven development because usability and user experience is a central factor [20].

In this vision paper, we explore how these conflicts are currently resolved in literature and propose novel types of approaches (cf. Table 1): one based on collaboration, and the other based on uncertainty (specifically, whether the resolution of a current inconsistency will change an affected value). We devise design categories for the resolution of these interactions and use these to compare current approaches, the category's impact on user experience, and finally predicting the impact of the newly proposed approaches. To



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/10.1145/3837062.3838682>

methodically discuss the impact, we use Self-Determination Theory (SDT) [24], which has been widely applied in Human Computer Interaction (e.g., see [2] and [28]), but also shows promising first results in software design [33]. The contribution of this paper is the novel application of SDT to resolve complex problems in software design, as well as the concrete solution approaches that lead to a more seamless integration of the human into the loop along with a baseline approach for comparison.

We believe that there is untapped potential in the design of the workflow to perform interactive repair, and SDT and our categories will help to guide us to creating less disruptive approaches that are optimized for the needs of the users. Overall, we expect good tool design with regard to interactive repair to be the key to foster closer collaboration and a shared understanding of design decisions that cross domain boundaries, leading to more motivated developers and engineers, fewer design errors, and faster project implementations, also as an enabler for more agile CPS processes.

2 Design Decisions in Existing Approaches for Interactive Repair

In this section, we compare existing approaches for interactive model repair, focusing on their suitability for interdisciplinary collaboration. We define interactive (model) repair as any repair that involves a human in the loop, where repair actions (i.e., concrete changes to restore consistency) are presented. A repair action can comprise multiple changes; we only prescribe that it leads to a consistent state. We consider inconsistency detection and the derivation of repair actions as orthogonal to the design of the inconsistency resolution process, which we want to examine in this paper. Therefore, for this comparison, we assume a detected inconsistency and multiple repair actions as a given.

Our comparisons are made along these six categories of design decisions: (1) resolver of interaction, (2) representation of repair options at the time of the decision, (3) possible delay of repair decision, (4) context represented for repair decision, (5) transparency of rationale for previous repair decisions, and (6) granularity of possible repair actions. The categories were arrived at through discussion among all coauthors; we agreed that these categories fundamentally shape the solution space for interactive repair. In contrast to the taxonomy by Macedo et al. [17], our categories place greater emphasis on central factors relevant to the user experience. *These categories form an initial working hypothesis and will be refined based on empirical results; therefore, we do not claim completeness.*

We identified three relevant fields in which interactive repair can play a role: Co-Evolution, Collaborative Modeling, and Federated Models, and used recent reputable surveys (or comparable) in the field to find relevant contributions on inconsistency resolution processes, which we reviewed and included here for relevance:

In *co-evolution of metamodels* with models (survey in [27]), the concrete syntax of models can get out of sync when the abstract syntax is changed. However, because the dependency is unidirectional, the resolution of the inconsistency can be delegated to the model maintainer, who then acts as the resolver, as demonstrated in [13]. In [13], the outdated element is marked as obsolete and is read-only. It is up to the maintainer to devise an alternative solution, i.e., there is no guidance on the repair options, and the repair can

be delayed arbitrarily. Similarly, categories 4-6 do not apply. The same platform also offers collaborative modeling, but uses locking mechanisms to prevent conflicts [12].

In *collaborative modeling*, conflicts are likely to occur (if not prevented by design). As the changes were made concurrently, the changes have to be consolidated, which requires understanding both changes. As shown in [7], the field mostly assumes a shared domain; therefore, we assume a stronger shared foundation of knowledge, which lessens the disruption as the resolver is likely capable of resolving the conflict competently on their own. This is also reflected in the resolution processes here: if one is presented, it mentions the direct presentation of possible repair actions to users, who then have to decide. However, this does not mean that distributed knowledge is not a problem in collaborative modeling: [8] expressly states the need for more communication in tools, which, to us, indicates the need for more support for the contextualization of developments in the model. This is further supported by [4], which states that the resolution of a merge conflict is more likely to be buggy than commits that were not involved in a merge conflict. In [4], this is linked to the disruptiveness of the merge resolution process. In [25], we encountered a further mention of a voting mechanism to decide conflicts in feature diagrams [16]. While this approach is collaborative and more flexible in the resolver category, it is missing the in-tool support to discuss options before deciding on a vote, which hinders the exchange of knowledge. Furthermore, the evaluation is performed with 19 participants, and the user experience is recorded coarsely-grained by asking for tool satisfaction. Therefore, this study indicates that collaborative resolution might be promising, but does not provide stronger evidence.

Federated Models [1] are usually employed when heterogeneous domains are collaborating. This field reflects the gap we intend to fill the most, but research on repair mainly focuses on the technical realization of consistency between the models, which is already quite complex. VITRUVIUS [14] offers the implementation to connect multiple metamodels with Consistency Preservation Rules (CPRs), which define the mapping of changes between them. These CPRs can propagate changes across models using the *reactions* language, and those reactions allow users to decide on a repair action interactively [15]. In relation to our categories, delay in VITRUVIUS is not possible, because interactions have to be resolved immediately. The resolver may only be the role that started the change propagation. The representation of a solution as well as the context are both determined once when the CPR is defined, and the granularity of a repair interaction is determined by the reaction. Transparency cannot be compared for VITRUVIUS because the approach does not preserve the rationale behind decisions. We found no other concrete descriptions of interactive repair workflows, although tools such as DesignSpace [18] also do interactive repair.

In summary, all three fields require one person to repair inconsistencies on their own, either with no context or a predefined context. If no unidirectional dependency can be assumed between the source domain and the target domain, then the resolver is the person who caused the conflict. The rationale for deciding on a specific repair action is not expressly preserved; some approaches include the option for messages in the version control system. The impact of the repair action is not presented to the resolver before the decision, which could be relevant when a decision might have implications

for multiple models. The design of the resolution process tends not to be the focus of publications, although there are indications of its importance; therefore, we want to further explore the solution space spanned by our categories.

3 Self-Determination Theory

To guide our exploration of alternative resolution processes, we use Self-Determination Theory (SDT) as a predictor of the user’s experience of tools for said conflict resolution.

SDT is a macro theory [24] concerned with human motivation, the inherent capacity for psychological growth and engagement, and the broader biological, organizational, and social conditions that support or thwart human well-being [23]. SDT consists of a variety of mini-theories that provide various perspectives on human motivation, with Basic Psychological Needs theory (BPNT) being one of the most prominent ones [30]. BPNT states that humans have an innate need to experience and satisfy the three basic psychological needs of autonomy, competence, and relatedness [24]. The need for autonomy refers to the experience of acting with volition and with self-endorsement of one’s actions, which is supported by the alignment of an individual’s values with their actions and acting outside of external control [9]. Competence, i.e., the experience of being effective and capable, is fostered by meaningful interaction with the environment, as well as access to informational feedback on one’s effectiveness and impact within the interaction [22]. Relatedness, or the need to belong and forge meaningful connections with others, is promoted by contributing to a common goal and a feeling of being integrated in a social context [9]. Within Human-Computer Interaction research, SDT is broadly applied in various domains that involve interaction with computing systems [2], for example, user experience design [3], development of persuasive health technologies [26], and software development [21]. SDT is also applied in research on human factors of motivation and well-being in the workplace [26, 29]. In a recent study by Wong et al. [32], SDT was applied to understand human factors of well-being and motivation in software development processes within companies at the organizational and technical levels.

Here, we argue that the development of complex software systems and CPS relies on cross-disciplinary communication and relationship building and maintenance [11], as well as complex interaction with computing systems and interfaces to ensure consistency of these CPS. Integrating an SDT perspective into the development of tools for consistency maintenance thus enables future design decisions toward human-centered interfaces that prioritize human motivation and well-being, contributing to effective work practices in the highly complex domain of CPS development.

We apply the three SDT needs of autonomy, competence, and relatedness to our design-decision categories in order to interrelate design decision-making and user experience of interactive model repair. Our aim here is to go beyond UI design so as to understand design decisions on a more fundamental level and also to discover new approaches for interactive model repair.

Please note that, unless indicated otherwise, the correlation between categories and needs is not empirically validated, although we tried to hypothesize within reason and in line with the mentioned research on SDT. This also applies to Section 4.

Competence can be affected by different categories: If the resolver has insufficient knowledge, their competence is negatively affected; similarly, if the representation of the solutions does not strike the correct balance between abstraction and relevance. It can also be positively affected with a suitable representation of solutions, by adding relevant knowledge in the context of the interaction, or by being able to access previous decisions and the reasoning in these cases (transparency). *Relatedness* can be fostered by successful interactions with other users, e.g., by providing additional relevant context for the interactions or by sourcing domain knowledge from previous decisions that are transparently documented. On the other hand, the relatedness of the resolver is negatively affected if they are asked to make a decision they cannot competently make. Furthermore, the delay of the resolution may affect either the resolver or the rest of the users, as a disruption may cause resentment in the resolver, whereas a long delay that affects other users affects their relatedness negatively. *Autonomy* is strengthened when an inconsistency can be resolved with minimal delay by a competent resolver without having to wait for bureaucratic overhead. Furthermore, an adequate granularity of solution allows for the desired degrees of freedom that supports the resolver’s autonomy. On the other hand, autonomy can be negatively affected if the work is impeded by repair interactions, e.g., by blocking changes (long delay), blocking resolution (wrong resolver), or disrupting work (short delay), but also by forcing the resolver to do additional work to document the decision (transparency).

4 Proposed Approaches

We now take a first step toward realizing our research vision of an optimized approach for interactive model repair. Here we sketch three different types of approach for interactive model repair, in order to explore (a) feasibility, but also crucially (b) user motivation, particularly under the three SDT needs of autonomy, competence, and relatedness. Table 1 aligns our three proposed approach types with our categories of design decisions (cf. Section 2).

Our baseline approach type is rooted in the state of the art. The resolver is the user who causes the initial change, **asserting their autonomy**. The context as well as all possible repair actions are determined during design time of the meta models; as a result, no repair action is specific to any instance of a repair. The interactive repair needs to be performed immediately, and until it is performed, no other changes can be propagated, and any possibly affected model elements are locked, thereby possibly **thwarting the autonomy as well as relatedness** of other users within development teams. Decisions are non-transparent because no further rationale for a decision is provided. The granularity of a repair action spans just one decision, and each decision spans only one model to another.

Our collaborative type of approach is inspired by reviews of Pull Requests (PRs) in the software development domain. In a PR, a developer asks to pull changes made in their own branch (upon review) into the main branch. We observe that interactive model repair resembles the PR’s review process in that the repair can be performed collaboratively by anyone: All collaborators can, as resolvers, contribute their knowledge and also weigh the repair options or ask for relevant context, similar to the review of a pull

Table 1: Features of our proposed approach types in comparison to our design decision categories

	Baseline	Collaborative	Uncertainty
Resolver	Affecting user	Collaborative	Affected users
Repr. of Options	Textual description	Outcomes & description	Most likely outcome
Delay	None	Dynamic	Maximum
Context	Predefined at design time	Conversations possible	None
Transparency	Rationale not preserved	Decision artifacts part of VCS	Rationale not preserved
Granularity	Model to model	Dynamic	Decision Points

request, in which the proposed changes can be openly discussed before deciding on the integration of the code. Ideally, this system is self-regulating, and knowledge exchange can be elicited exactly where necessary. In such a work environment, **relatedness is fostered while autonomy remains uninhibited**, i.e., the degree of involvement of different domains in the repair can be tailored to the specific instance. Furthermore, widespread familiarity with PRs increases the likelihood of user satisfaction on account of pre-existing competence (cf. findings from development of Jjodel [5]). However, such discussion before making a decision might cause delays. On a different note, dynamic exchange of domain-specific knowledge will likely enrich competence. The results of a repair action can be simulated and derived in such a way that the outcome of the decision can be presented along with a high-concept textual description; this, of course, makes for a representation of options. The delay until consistency is restored is dynamic, but peer pressure may be exerted to enforce quicker decision-making. In this case, we assume a Git-like version control, meaning incoming changes can be merged with the repair actions. Transparency is high, because Version Control Systems (VCS) can also persist the discussion and make it findable for future reference. Granularity of decisions is dynamic, because different users can contribute to the final decision based on their expertise.

Our uncertainty type of approach focuses on mitigating the disruptiveness arising prior to the interactive repair: if an inconsistency cannot be fully resolved, the elements possibly affected are tagged with an uncertainty property, as they are part of an epistemic uncertainty. For example, an element that is changed in six out of ten possible paths receives a 60% uncertainty on account of its potential for change once the inconsistency is repaired. Of course, other distributions of probability may be specified in the CPR, and if a probability is not equally distributed, the most probable solution could be propagated through the models such that the repair action only needs to be reviewed. (The technical feasibility of propagation of uncertainty annotations, albeit for a different purpose, is demonstrated in [10].) The possible delay in our uncertainty type of approach is maximal, the decisions can be made directly in the model by its maintainers (i.e., our resolvers), and the most likely outcome is directly presented in the model (i.e., the approach has a representation of options). On the other hand, this approach type does not provide context in the models directly, nor does it exhibit transparency rationales. The granularity of the repair action is decision-point based. Our uncertainty type of approach does not block ongoing work; for that reason, the approach type is intended to improve the experience of users who

are affected by pending repair actions. As a result, the approach type **mitigates negative impacts on autonomy**; moreover, the support in an actual resolution of inconsistency is comparable to our baseline type.

The collaborative approach strongly supports knowledge exchange, while the uncertainty approach mitigates the disruption caused by the inconsistency present in the system. Therefore, we expect that, in most scenarios, the combination of the two approaches is the most promising: once a conflict is introduced, all elements that could be implicated in a repair are annotated with uncertainty to prevent the introduction of further inconsistencies. In parallel, the inconsistencies and their repair actions are listed on a collaborative platform, where all involved parties can discuss the proposed repairs and exchange the knowledge required to make the decision. Because we assume collective ownership of the overall model, we expect review latency to correlate with the urgency of resolving the inconsistency. However, the need for additional measures to mitigate review latency should be addressed in an empirical evaluation. In collaborative modeling [6], asynchronous communication has been identified as more suitable for distributed teams, which we assume for interdisciplinary collaboration, so we expect the problem of review latency to be manageable in future work.

5 Conclusion and Research Road Ahead

In this vision paper, we identified a critical gap in state-of-the-art implementations of interactive model repair. Most approaches fail to consider such, and where an approach does, the interaction is chosen naively, without any consideration for human-in-the-loop factors. Therefore, we see an untapped potential in the design of interactive model repair. We explored ways of tapping into that potential by leveraging SDT to establish a set of initial working categories on decision-making in design. We examined how design decisions likely affect the satisfaction of SDT basic user needs of autonomy, competence, and relatedness. Based on our findings, we proposed two new types of approaches: the collaborative type and the uncertainty type. Importantly, we recognized the possible advantages of combining those approach types.

The road ahead for this new direction in the research will involve, for example,

- Implementation of the proposed approaches
- Validation of our assumptions on disruptiveness by (a) considering functional requirements (e.g., performance and applicability), but also by (b) conducting qualitative user studies

to determine satisfaction of autonomy, competence, and relatedness, by designing a repeatable “design challenge” for light-weight evaluation

- Refinement of the design decision categories based on the newly gained qualitative insights

Acknowledgments

This work was supported by the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF) and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1608 - 501798263. This paper was edited by our textician, Daniel Shea.

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 (MODELS Companion '24)*. Association for Computing Machinery, New York, NY, USA, 990–999. doi:10.1145/3652620.3688221
- [2] Nick Ballou, Sebastian Deterding, April Tyack, Elisa D Mekler, Rafael A Calvo, Dorian Peters, Gabriela Villalobos-Zúñiga, and Selen Turkay. 2022. Self-Determination Theory in HCI: Shaping a Research Agenda. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. ACM, New Orleans LA USA, 1–6. doi:10.1145/3491101.3503702
- [3] Daniel Bennett and Elisa D. Mekler. 2024. Beyond Intrinsic Motivation: The Role of Autonomous Motivation in User Experience. *ACM Transactions on Computer-Human Interaction* 31, 5 (Oct. 2024), 1–41. doi:10.1145/3689044
- [4] Caius Brindescu, Iftekhar Ahmed, Carlos Jensen, and Anita Sarma. 2020. An empirical investigation into merge conflicts and their effect on software quality. *Empirical Software Engineering* 25, 1 (Jan. 2020), 562–590. doi:10.1007/s10664-019-09735-4
- [5] Antonio Bucchiarone, Juri Di Rocco, Damiano Di Vincenzo, and Alfonso Pierantonio. 2025. Modeling in Jjodel: towards bridging complexity and usability in model-driven engineering. *Software and Systems Modeling* (Oct. 2025). doi:10.1007/s10270-025-01324-y
- [6] Abhishek Choudhury, Ivano Malavolta, Federico Ciccozzi, Kousar Aslam, and Patricia Lago. 2025. The technological landscape of collaborative model-driven software engineering. *Software and Systems Modeling* 24, 5 (Oct. 2025), 1595–1619. doi:10.1007/s10270-025-01274-5
- [7] 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
- [8] Istvan David, Kousar Aslam, Ivano Malavolta, and Patricia Lago. 2023. Collaborative Model-Driven Software Engineering — A systematic survey of practices and needs in industry. *Journal of Systems and Software* 199 (May 2023), 111626. doi:10.1016/j.jss.2023.111626
- [9] Edward L. Deci, Anja H. Olafsen, and Richard M. Ryan. 2017. Self-Determination Theory in Work Organizations: The State of a Science. *Annual Review of Organizational Psychology and Organizational Behavior* 4, 1 (March 2017), 19–43. doi:10.1146/annurev-orgpsych-032516-113108
- [10] Nathan Hagel, Johannes Mäkelburg, Claus Hammann, Maximilian Hummel, Ralf Reussner, and Maribel Acosta. 2026. UnCertaGator: Propagation of Uncertainty Information in Development Processes Using Consistency Automation. In *2026 IEEE 23rd International Conference on Software Architecture (ICSA)*. Amsterdam, Netherlands, 315–322. doi:10.1109/ICSA66085.2026.00038
- [11] Adarsh Kumar Satindarlal Kakar. 2018. Engendering cohesive software development teams: Should we focus on interdependence or autonomy? *International Journal of Human-Computer Studies* 111 (March 2018), 1–11. doi:10.1016/j.ijhcs.2017.11.001
- [12] Steven Kelly and Juha-Pekka Tolvanen. 2021. Collaborative modelling and meta-modelling with MetaEdit+. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, IEEE, Fukuoka, Japan, 27–34. doi:10.1109/MODELS-C53483.2021.00012
- [13] Steven Kelly and Juha-Pekka Tolvanen. 2023. How MetaEdit+ supports co-evolution of modeling languages, tools and models. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, IEEE, Västerås, Sweden, 9–13.
- [14] 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. *The journal of systems and software* 171 (2021), Article no: 110815. doi:10.1016/j.jss.2020.110815
- [15] Max Emanuel Kramer. 2019. *Specification Languages for Preserving Consistency between Models of Different Languages*. KIT Scientific Publishing, Karlsruhe. 476 pages. doi:10.5445/KSP/1000081446
- [16] Elias Kuitert, Sebastian Krieter, Jacob Krüger, Gunter Saake, and Thomas Leich. 2021. variED: an editor for collaborative, real-time feature modeling. *Empirical Software Engineering* 26, 2 (March 2021), 24. doi:10.1007/s10664-020-09892-x
- [17] Nuno Macedo, Tiago Jorge, and Alcino Cunha. 2017. A Feature-Based Classification of Model Repair Approaches. *IEEE Transactions on Software Engineering* 43, 7 (July 2017), 615–640. doi:10.1109/TSE.2016.2620145
- [18] Luciano Marchezan, Wesley K. G. Assuncao, Roland Kretschmer, and Alexander Egyed. 2022. Change-Oriented Repair Propagation. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering (Pittsburgh, PA, USA) (ICSSP '22)*. Association for Computing Machinery, New York, NY, USA, 82–92. doi:10.1145/3529320.3529330
- [19] Manar Mazkati, Erik Burger, Anne Koziolek, and Ralf H. Reussner. 2017. Automotive Systems Modelling with Vitruvius. *INFORMATIK 2017*, 1487–1498 pages. doi:10.18420/in2017_148
- [20] Alfonso Pierantonio, Simone Gozzano, and Monica Mazza. 2026. The cognitive science of modeling tools. *Software and Systems Modeling* 25, 2 (April 2026), 319–325. doi:10.1007/s10270-026-01359-9
- [21] Daniel Russo, Paul H. P. Hanel, Seraphina Altnickel, and Niels van Berkel. 2023. Satisfaction and performance of software developers during enforced work from home in the COVID-19 pandemic. *Empirical Software Engineering* 28, 2 (March 2023), 53. doi:10.1007/s10664-023-10293-z
- [22] Richard M. Ryan and Edward L. Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist* 55, 1 (2000), 68–78. doi:10.1037/0003-066X.55.1.68
- [23] Richard M. Ryan and Edward L. Deci. 2018. *Self-determination theory: basic psychological needs in motivation, development, and wellness* (paperback ed.). The Guilford Press, New York London.
- [24] Richard M. Ryan and Maarten Vansteenkiste. 2023. Self-Determination Theory: Metatheory, Methods, and Meaning. In *The Oxford Handbook of Self-Determination Theory* (1 ed.), Richard M. Ryan (Ed.). Oxford University Press, 3–30. doi:10.1093/oxfordhb/9780197600047.013.2
- [25] Mohammadreza Sharbaf, Bahman Zamani, and Gerson Sunyé. 2023. Conflict management techniques for model merging: a systematic mapping review. *Software and Systems Modeling* 22, 3 (June 2023), 1031–1079. doi:10.1007/s10270-022-01050-9
- [26] Gavin R. Slemp and Dianne A. Vella-Brodrick. 2014. Optimising Employee Mental Health: The Relationship Between Intrinsic Need Satisfaction, Job Crafting, and Employee Well-Being. *Journal of Happiness Studies* 15, 4 (Aug. 2014), 957–977. doi:10.1007/s10902-013-9458-3
- [27] Juha-Pekka Tolvanen, Steven Kelly, Juri Di Rocco, Alfonso Pierantonio, and Giordano Tinella. 2025. A framework for evaluating tool support for co-evolution of modeling languages, tools and models. *Software and Systems Modeling* 24, 2 (April 2025), 311–338. doi:10.1007/s10270-024-01218-5
- [28] April Tyack and Elisa D. Mekler. 2024. Self-Determination Theory and HCI Games Research: Unfulfilled Promises and Unquestioned Paradigms. *ACM Transactions on Computer-Human Interaction* 31, 3 (June 2024), 1–74. doi:10.1145/3673230
- [29] Anja Van den Broeck, D. Lance Ferris, Chu-Hsiang Chang, and Christopher C. Rosen. 2016. A Review of Self-Determination Theory’s Basic Psychological Needs at Work. *Journal of Management* 42, 5 (July 2016), 1195–1229. doi:10.1177/0149206316632058
- [30] Maarten Vansteenkiste, Richard M. Ryan, and Bart Soenens. 2020. Basic psychological need theory: Advancements, critical themes, and future directions. *Motivation and Emotion* 44, 1 (Feb. 2020), 1–31. doi:10.1007/s11031-019-09818-1
- [31] Thomas Alexander Voelk, Lars Gesmann, Stefan Götz, Kevin Feichtinger, Stefan Eric Schwarz, Tobias Düser, Ralf Reussner, and Albert Albers. 2025. Misunderstand me correctly - comprehensibility in interdisciplinary collaboration. *Proceedings of the Design Society* 5 (Aug. 2025), 2441–2450. doi:10.1017/pds.2025.10258
- [32] Novia Wong, Nai-Yu Cheng, Bruna Oewel, Katherine E Genuario, SarahElizabeth Stoeckl, Stephen M. Schueller, Iftekhar Ahmed, André Van Der Hoek, and Madhu Reddy. 2025. ‘It’s a spectrum’: Exploring Autonomy, Competence, and Relatedness in Software Development Processes and Tools. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–19. doi:10.1145/3706598.3713250
- [33] Ming Wu, Xiaohui Shen, Lekai Zhang, Xingyu Liu, Hailong Zhang, and Yaqi Hu. 2024. A heuristic design toolkit for web conferencing software based on self-determination theory. *Computer Standards & Interfaces* 89 (April 2024), 103831. doi:10.1016/j.csi.2023.103831