WORK IN PROGRESS

Preserving Non-Functional Requirements in Goal Models Using Meta-models of the Software Product Lines

GULLELALA JADOON, University of Florence, Florence, FI, Italy

Published: 02 September 2024

Citation in BibTeX format

SPLC '24: 28th ACM International Systems and Software Product Line Conference
September 2 - 6, 2024
Dommeldange, Luxembourg

Conference Sponsors:
SIGSOFT

Preserving Non-Functional Requirements in Goal Models Using Meta-models of the Software Product Lines

Gullelala Jadoon
gullelala.jadoon@isti.cnr.it
University of Florence-ISTI CNR
Pisa, Italy

ABSTRACT

Non-functional requirements (NFRs) play a critical role in software product line (SPL) engineering, ensuring products meet essential criteria beyond mere functionality. However, preserving NFRs across product variants induces considerable challenges, particularly in goal-oriented SPLE where goals guide product derivation. This research proposes a novel framework to preserve NFRs in goal models using meta-models of SPLs and manage inconsistent NFRs. The framework utilizes product and domain meta-models to accurately capture and represent NFRs, addressing construct validity concerns. This research aims to enhance the credibility and generalizability of findings in SPL engineering, contributing to the advancement of goal-oriented modeling and NFR preservation practices.

CCS CONCEPTS

• **Software and its engineering → Software product lines**.

KEYWORDS

Software Product Line Engineering, Property preservation, Goal models, Meta-models, Non-functional requirements

ACM Reference Format:
Gullelala Jadoon. 2024. Preserving Non-Functional Requirements in Goal Models Using Meta-models of the Software Product Lines. In *28th ACM International Systems and Software Product Line Conference (SPLC '24), September 02–06, 2024, Dommeldange, Luxembourg*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3646548.3676541

1 INTRODUCTION AND MOTIVATION

An integral aspect of Software Product Line Engineering (SPLE) is the effective management of requirements, encompassing both functional and non-functional facets. While the former defines what a system should do, non-functional requirements (NFRs) specify how it should behave and perform. NFRs are essential to ensure the overall quality, usability, and performance of software products, making their consideration imperative for the success of SPLE [2, 4].

Goal-oriented SPLE involves the systematic development of software products that are tailored to meet specific goals or objectives. In this approach, goals serve as the primary drivers for product derivation, guiding the selection and configuration of features and functionalities across multiple product variants within the SPL [3, 12]. Goal SPLs facilitate the systematic management of variability by aligning product features with stakeholder requirements and objectives, thereby enhancing the flexibility, adaptability, and reusability of software assets across the product line.

Traditionally, attention has concentrated on modeling functional requirements in SPLE, with well-established techniques and methodologies available for this purpose [9]. However, the modeling of NFRs in SPLE presents unique challenges and complexities. Unlike functional requirements, which can often be directly derived from user needs or system functionalities, NFRs may span multiple products within an SPL, requiring careful analysis and management [1].

Moreover, NFRs exhibit inherent variability, with different SPL products potentially having different sets of NFRs or varying degrees of emphasis on specific quality attributes [8]. This introduces additional complexity, necessitating specialized modeling techniques and tools to effectively capture and manage NFRs across the SPL.

Tools often focus on facilitating SPL development, verification, and management, aiming to assess their capabilities, features, and effectiveness [7, 13, 25, 26]. These papers provide insights into the current state of tool support for SPLs, identifying strengths, weaknesses, and areas for improvement. Their empirical findings contribute to a better understanding of the SPL tooling ecosystem.

Problem Identification. Despite growing recognition of the importance of NFRs in SPLE, a notable gap remains regarding the modeling and management of these requirements in goal models. Existing approaches often lack clarity on meta-models and strategies for handling contradictory NFRs, hindering their practical applicability and effectiveness in goal-oriented SPLE projects.

Motivation. In light of these challenges, this research aims to address the gaps in modeling NFRs in goal-oriented SPLs through the application of Model-Driven Software Engineering (MDSE) approaches. By developing a systematic methodology for modeling domain non-functional requirements (DNFRs) from individual product non-functional requirements (PNFRs) and refining metamodels for enhanced clarity and consistency, this research seeks to advance the state-of-the-art in goal-oriented SPL requirements engineering.

Contribution. Through empirical evaluation and investigation of strategies for handling NFRs and contradictory NFRs, our research aims to provide practical insights and recommendations for improving NFR modeling in goal-oriented SPL. By bridging the gap between theory and practice, this research contributes to the broader pursuit of enhancing the quality, reliability, and adaptability of SPL in diverse application domains.

Research Questions (RQs). We focus on the following main RQs:

SPLC '24, September 02–06, 2024, Dommeldange, Luxembourg

ACM ISBN 979-8-4007-0593-9/24/09
https://doi.org/10.1145/3646548.3676541

RQ1 What is the significance of property preservation in goal-oriented SPL?
RQ2 What methods can be employed to utilize domain metamodels effectively to preserve properties outlined in goal models?
RQ3 What strategies can be implemented to effectively handle and resolve contradictory requirements in goal-oriented SPL?

2 RELATED WORK

A lot of research has been carried out to focus on goal-oriented SPL and its variability management, including surveys and literature reviews exploring approaches for modeling NFRs by systematically analyzing existing research in the field [1, 5, 11, 15, 22, 24]. Noyer et al. [21] discuss the importance of traceability and interfacing between requirements engineering and UML domains, highlighting the use of the standardized Requirements Interchange Format (ReqIF) as a means to achieve traceability and seamless integration between these domains. On the other hand, Sree-Kumar et al. [23] focus on natural language specifications for extracting features of SPLs.

Loniewski et al. [17] introduce a novel methodology that integrates architecture-oriented principles with model-driven requirements engineering techniques. This approach aims to enhance the traceability, consistency, and comprehensibility of requirements throughout the software development lifecycle. Jadoon et al. [14] address the traceability challenges faced by small and medium-sized software industries by proposing a model-oriented approach. By leveraging models as central artifacts in the traceability process, their framework offers a systematic method to establish and maintain trace links between requirements and other development artifacts.

Kroher, Daniel, Myllarniemi, and Munoz et al. [8, 16, 19, 20] focus on integrating quality considerations in the artifacts of SPL variability. They propose methodologies to embed quality attributes directly in SPL goal models, enabling better management of quality concerns across product variants. They address the challenge of managing variability changes over the SPL lifecycle, which is crucial to ensure the adaptability and maintainability of product line assets.

Derakhshanmanesh and Buttling et al. [6, 10] present component-based approaches to model software systems. The authors propose a methodology that allows for the integration of diverse goal models at different abstraction levels, facilitating the development of complex software systems. By leveraging component-based techniques, the approach enables modularization and reuse of system components, enhancing maintainability and scalability.

Gwasem and Ayala et al. [3, 12] focused on the variability of goal-oriented models and introduced a methodology for acquiring testable NFRs to enhance requirements analysis in goal-driven SPL. The authors propose a method for analyzing and optimizing SPL products based on these models, aiming to align product features with stakeholder objectives and requirements.

To conclude, these studies identify, categorize, and analyze relevant work related to goal models of SPLs, including common practices, challenges, methodologies, and tools. The novelty of our proposed approach lies in its use of meta-models to preserve NFRs in goal models and effectively manage inconsistent NFRs. This approach ensures accurate NFR representation and addresses construct validity concerns, enhancing the credibility and generalizability of findings in goal-oriented SPLE.

3 METHODOLOGY AND APPROACH

Our methodology adopts a design science approach comprising the identification of the problem, the proposed solution, and its evaluation. By using model-driven techniques and domain-specific modeling, our methodology aims to enhance the traceability, consistency, and completeness of NFRs within SPLs, thereby facilitating more robust and reliable product development processes. In the following, we outline each step of the methodology, explaining the rationale, transformation, and management of NFRs employed to achieve our research objectives effectively as shown in Figure 1. NFRs are extracted from each SPL and are used to create the product and domain metamodels. These metamodels result in a product variability model (PVM) and domain variability model (DVM). Both models are mapped with the goal model (GM) to verify their completeness. Contradictory NFRs are recorded from product and domain VMs and are processed to be incorporated into the transformations.

Extraction of NFRs. At first, a detailed examination of individual product requirements is undertaken. This analysis aims to identify and extract NFRs embedded within each product variant. Each requirement is given a unique identifier ('ID') and an associated description ('Description'). Utilizing domain knowledge, stakeholder insights, and existing documentation, we attain a thorough understanding of the diverse NFR landscape across product variants. Through this initial phase, a rich repository of product-specific NFRs is established, serving as the foundational dataset for subsequent modeling and analysis.

Creation of Product NFRs Metamodel. With the extracted NFRs at hand, we proceed to formalize the representation of product-level NFRs through the development of a dedicated metamodel. This metamodel serves as a structured artifact for encapsulating the essential attributes, relationships, and constraints associated with the product. By defining standardized entities, properties, variability, and commonality within the metamodel, the model ensures coherence and consistency in the representation of NFRs across the product models. An NFR is categorized into a commonality or variability based on a Commonality Variability ratio (CV), which is defined as the ratio of the number of systems sharing a common feature to the total number of systems [18].

Transformation to PVM. Using model-driven techniques, we undertake the transformation of individual product models into a cohesive variability model called PVM. This transformation process comprises the aggregation and configuration of product-specific NFRs into a unified representation that constitutes the collective non-functional requirements across the product variants in the SPL. With automated model transformations using QVT (Query View Transformation) and tools such as EMF ecore, this process is automated while ensuring the preservation of NFR integrity and fidelity throughout the transformation. At this stage, the features of PVM are mapped with the GM. The GM comprises the soft and hard goals agreed upon by the stakeholders.

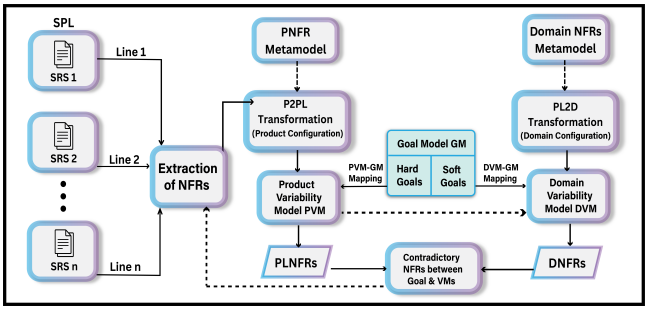

Figure 1: Methodology to preserve NFRs in goal-oriented SPL

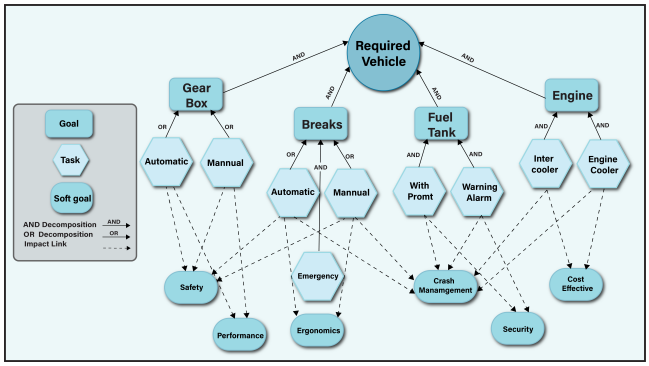

Figure 2: Goal model of required Vehicle

Creation of Domain NFRs Metamodel. In parallel with the extraction of domain NFRs, we extend the existing metamodel to accommodate the representation of domain-specific NFRs. This extended metamodel incorporates additional entities, relationships, and constraints necessary for capturing the features of domain-level NFRs within the software product line. Through iterative refinement and validation, the domain NFRs metamodel undergoes scrutiny to ensure completeness, correctness, and alignment with domain architecture principles and objectives. This metamodel results in a DVM that captures the domain NFRs.

Transformation to DVM. Based on the domain configuration, DVM is transformed. The features are mapped with the GM to ensure completeness and consistency with the required characteristics.

Extraction of Domain NFRs. Building upon the transformed DVM, we derive domain-specific NFRs that encapsulate commonalities and variabilities across product variants within the software product line. Through a frequent analysis of the DVM, domain-specific NFRs are identified and extracted, aligning with the domain model and objectives.

Identification of Contradictory NFRs. Following the derivation of product line and domain-specific NFRs, we identify and moderate any inconsistencies, conflicts, or contradictions that may arise between the two sets of requirements. The resolved requirements are sent back to the product and domain transformations for inclusion in the product line and domain variability models.

4 EVALUATION AND INITIAL RESULTS

In this section, we provide a generic evaluation of the proposed methodology using a running example from the automotive domain, focusing on the main features of a Vehicle.

As a running example, we consider the development of a Vehicle with four main features. Each feature is associated with specific NFRs that are critical for the overall performance, safety, and usability of the Vehicle as shown in the GM of Figure 2.

Extraction of NFRs. Initially, we conduct a detailed examination of individual product requirements to extract the embedded NFRs within each feature of the Vehicle. We identify NFRs associated with the Gearbox, Brakes, Fuel Tank, and Engine. These NFRs include:

(1) Gearbox: Safety, Performance
(2) Brakes: Safety, Ergonomics, Crash Management
(3) Fuel Tank: Security, Cost Effectiveness, Crash Management
(4) Engine: Performance, Safety, Crash Management, Cost Effectiveness

These NFRs span various aspects such as safety, performance, ergonomics, crash management, security, and cost-effectiveness, reflecting the quality aspects of requirements in automotive systems.

Creation of Product and Domain Metamodels. With the extracted NFRs, we proceed to formalize their representation through the development of dedicated Ecore metamodels as shown in Figure 3. Each product is associated with a unique 'id'. Additionally, we extend the metamodeling to accommodate domain-specific NFRs, ensuring coherence and consistency in their representation across product variants.

Transformation and Verification. We transform individual product models into cohesive variability models: PVM and DVM. These transformations aggregate and configure product- and domain-specific NFRs into unified representations, ensuring the preservation of NFR integrity and adherence throughout the process. Then the features of the PVM and DVM are mapped with the GM, comprising soft and hard goals agreed upon by stakeholders, to verify the completeness and consistency of the variability models.

Identification of Contradictory NFRs. Following the derivation of product line and domain-specific NFRs, we identify and moderate any inconsistencies or conflicts that may arise between the two sets of requirements. For example, a requirement for maximum safety through reinforced steel doors might conflict with a requirement for fuel efficiency. Reinforced steel doors increase the car's weight, which can reduce fuel efficiency by requiring more energy to move the vehicle.

Through this demonstration, we showcase the application of the proposed methodology in systematically addressing the complexities of modeling NFRs within the context of goal-oriented SPLE. The threats to the validity of our research include construct validity concerns regarding the accurate representation of NFRs in metamodels, internal validity challenges related to attributing observed effects to interventions, and external validity limitations concerning the generalizability of findings beyond the automotive domain. Addressing these threats requires thorough attention to detail in

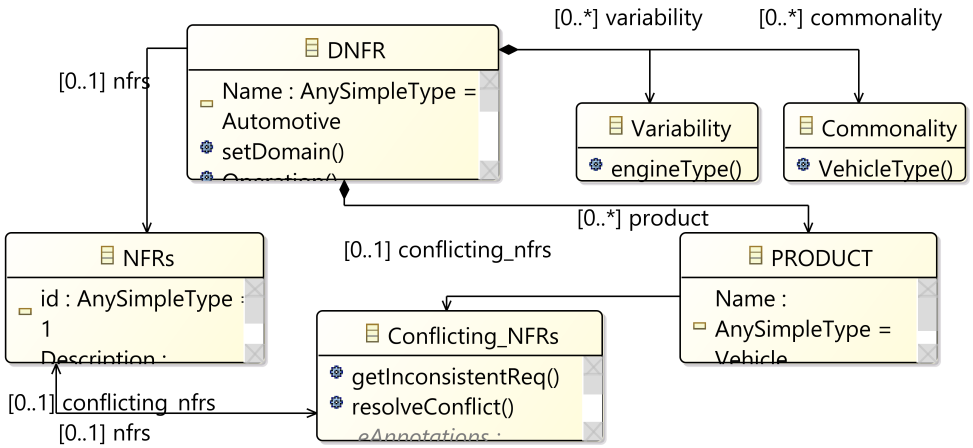

Figure 3: Domain metamodel in Ecore representing the NFRs, commonalities, and variabilities

conceptualization, execution, and interpretation to enhance the credibility and applicability of our research findings.

5 WORK PLAN

The following work plan will be followed.

Thesis Structure.

Introduction: Introduce the research problem, objectives, and significance, emphasizing the need for property preservation in rigorous software development.

Literature Review: Review existing literature on model transformation and property preservation in rigorous software development, focusing on relevant methodologies and techniques in preserving NFRs in goal SPLs.

Methodology: Detail the proposed framework for preserving non-functional requirements in goal models using meta-models of software product lines (SPLs).

Demonstration and Evaluation: Demonstrate the effectiveness of the framework through case studies or rigorous analysis using more detailed NFRs and evaluate its impact on property preservation in goal models.

Discussion: Analyze findings, discuss implications for SPL development, and propose future research directions.

Conclusion: Summarize key findings, contributions, and implications, reiterating the importance of property preservation in goal models.

References: Include all cited references in the thesis.

Work Accomplished.

Systematic Literature Review: Conducted a comprehensive review of literature on model transformation and property preservation in rigorous software development, identifying gaps and challenges.

Framework Proposal: Developed and submitted a framework proposal in the doctoral symposium, for preserving non-functional requirements in goal models using meta-models of SPLs, based on insights from the literature review.

Work Remaining.

Framework Refinement: Further refine and develop the proposed framework based on feedback from the literature review and symposium proposal.

Case Studies: Design and conduct case studies to demonstrate the framework's effectiveness in preserving non-functional requirements in goal models within SPLs.

Evaluation: Evaluate the impact of the framework on property preservation, comparing outcomes before and after its implementation in SPL development processes.

Thesis Writing: Complete the remaining chapters of the thesis, including the discussion, conclusion, and final revisions.

Publication Plan.

Research Paper SLR: Prepare the systematic literature review findings for publication, highlighting insights and implications for model transformation and property preservation in rigorous software development.

Research Paper Framework: Draft a research paper detailing the proposed framework, including its development, application, and evaluation results, for submission to a relevant conference or journal.

Work Plan for the Next 12 Months.

Months 1-3: Refine and further develop the framework based on feedback from the literature review and proposal submission.

Months 4-6: Design and conduct case studies to demonstrate the framework's effectiveness in real-world SPL scenarios.

Months 7-9: Evaluate the impact of the framework on property preservation in goal models within SPLs, analyzing outcomes and identifying areas for improvement.

Months 10-11: Complete writing the remaining chapters of the thesis, including discussion and conclusion, integrating findings from the case studies and evaluation.

Month 12: Finalize thesis, prepare for defense, and submit for examination.

REFERENCES

[1] Vander Alves, Nan Niu, Carina Alves, and George Valença. 2010. Requirements engineering for software product lines: A systematic literature review. *Information and Software Technology* 52, 8 (2010), 806–820. https://doi.org/10.1016/j.infsof.2010.03.014

[2] David Ameller, Xavier Franch, Cristina Gómez, Silverio Martínez-Fernández, João Araújo, Stefan Biffl, Jordi Cabot, Vittorio Cortellessa, Daniel Méndez Fernández, Ana Moreira, Henry Muccini, Antonio Vallecillo, Manuel Wimmer, Vasco Amaral, Wolfgang Böhm, Hugo Bruneliere, Loli Burgueño, Miguel Goulão, Sabine Teufl, and Luca Berardinelli. 2019. Dealing with Non-Functional Requirements in Model-Driven Development: A Survey. *IEEE Transactions on Software Engineering* 47, 4 (2019), 818–835. https://doi.org/10.1109/TSE.2019.2904476

[3] Inmaculada Ayala, Mercedes Amor, and Lidia Fuentes. 2023. Analysis and optimization of SPL products using goal models. In *Proceedings of the 31st International Requirements Engineering Conference (RE'23)*. IEEE, 89–99. https://doi.org/10.1109/RE57278.2023.00018

[4] Felix Bachmann and Paul C. Clements. 2005. *Variability in software product lines.* Technical Report CMU/SEI-2005-TR-012. Carnegie Mellon University. https://insights.sei.cmu.edu/library/variability-in-software-product-lines/

[5] Anissa Benlarabi, Amal Khtira, and Bouchra El Asri. 2023. Evaluation of User Experience in Software Product Lines Derivation Process. In *Proceedings of the 3rd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET'23)*. IEEE, 1–7. https://doi.org/10.1109/IRASET57153.2023.10152989

[6] Arvid Butting, Robert Eikermann, Oliver Kautz, Bernhard Rumpe, and Andreas Wortmann. 2018. Modeling language variability with reusable language components. In *Proceedings of the 22nd International Systems and Software Product Line Conference (SPLC'18)*. ACM, 65–75. https://doi.org/10.1145/3233027.3233037

[7] Asma Charfi, Shuai Li, Thomas Payret, Patrick Tessier, Chokri Mraidha, and Sébastien Gérard. 2019. A Model Driven Tool for Requirements and Hardware Engineering. In *Proceedings of the 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C'19)*. 769–773. https://doi.org/10.1109/MODELS-C.2019.00120

[8] Mworia Daniel, Nderu Lawrence, and Kimwele Michael. 2021. Embedding Quality into Software Product Line Variability Artifacts. *International Journal of Software Engineering & Applications* 12, 2/3 (2021). https://doi.org/10.5121/ijsea.2021.12302

[9] Bert de Brock. 2018. Towards Pattern-Driven Requirements Engineering: Development Patterns for Functional Requirements. In *Proceedings of the 8th International Model-Driven Requirements Engineering Workshop (MoDRE'18)*. IEEE, 73–78. https://doi.org/10.1109/MoDRE.2018.00016

[10] Mahdi Derakhshanmanesh, Jürgen Ebert, Marvin Grieger, and Gregor Engels. 2019. Model-integrating development of software systems: a flexible component-based approach. *Software and Systems Modeling* 18 (2019), 2557–2586. https://doi.org/10.1007/s10270-018-0682-5

[11] Sascha El-Sharkawy, Nozomi Yamagishi-Eichler, and Klaus Schmid. 2019. Metrics for analyzing variability and its implementation in software product lines: A systematic literature review. *Information and Software Technology* 106 (2019), 1–30. https://doi.org/10.1016/j.infsof.2018.08.015

[12] Ibtesam Gwasem, Weichang Du, and Andrew McAllister. 2023. Acquiring Testable NFRs Utilizing Goal Models Enhancing Application Requirements Analysis in Goal-Driven Software Product Lines. In *Proceedings of the 8th International Conference on Engineering Technologies and Applied Sciences (ICETAS'23)*. IEEE, 1–8. https://doi.org/10.1109/ICETAS59148.2023.10346316

[13] José Miguel Horcas, Mónica Pinto, and Lidia Fuentes. 2023. Empirical analysis of the tool support for software product lines. *Software and Systems Modeling* 22, 1 (2023), 377–414. https://doi.org/10.1007/s10270-022-01011-2

[14] Gullelala Jadoon, Muhammad Shafi, and Sadaqat Jan. 2019. A Model-Oriented Requirements Traceability Framework for Small and Medium Software Industries. In *Proceedings of the 20th International Arab Conference on Information Technology (ACIT'19)*. IEEE, 91–96. https://doi.org/10.1109/ACIT47987.2019.8991116

[15] Fatima Khalique, Wasi Haider Butt, and Shoab Ahmad Khan. 2017. Creating Domain Non-functional Requirements Software Product Line Engineering Using Model Transformations. In *Proceedings of the 15th International Conference on Frontiers of Information Technology (FIT'17)*. IEEE, 41–45. https://doi.org/10.1109/FIT.2017.00015

[16] Christian Kröher, Lea Gerling, and Klaus Schmid. 2018. Identifying the intensity of variability changes in software product line evolution. In *Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1* (Gothenburg, Sweden) *(SPLC '18)*. Association for Computing Machinery, New York, NY, USA, 54–64. https://doi.org/10.1145/3233027.3233032

[17] Grzogorz Loniewski, Ausias Armesto, and Emilio Insfran. 2011. An architecture-oriented model-driven requirements engineering approach. In *Proceedings of the 1st Model-Driven Requirements Engineering Workshop (MoDRE'11)*. IEEE, 31–38. https://doi.org/10.1109/MoDRE.2011.6045364

[18] Mikyeong Moon, Keunhyuk Yeom, and Heung Seok Chae. 2005. An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. *IEEE Transactions on Software Engineering* 31, 7 (2005), 551–569. https://doi.org/10.1109/TSE.2005.76

[19] Daniel-Jesus Munoz, Dilian Gurov, Monica Pinto, and Lidia Fuentes. 2021. Category Theory Framework for Variability Models with Non-functional Requirements. In *International Conference on Advanced Information Systems Engineering*, Marcello La Rosa, Shazia W. Sadiq, and Ernest Teniente (Eds.), Vol. 12751. Springer, 397–413.

[20] Varvana Myllärniemi, Juha Savolainen, Mikko Raatikainen, and Tomi Männistö. 2016. Performance variability in software product lines: proposing theories from a case study. *Empirical Software Engineering* 21 (2016), 1623–1669. https://doi.org/10.1007/s10664-014-9359-z

[21] Arne Noyer, Padma Iyenghar, Elke Pulvermueller, Florian Pramme, and Gert Bikker. 2015. Traceability and interfacing between requirements engineering and UML domains using the standardized ReqIF format. In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD'15)*. IEEE, 1–6.

[22] Monique Snoeck and Yves Wautelet. 2022. Agile MERODE: a model-driven software engineering method for user-centric and value-based development. *Software and Systems Modeling* 21, 4 (2022), 1469–1494. https://doi.org/10.1007/s10270-022-01015-y

[23] Anjali Sree-Kumar, Elena Planas, and Robert Clarisó. 2018. Extracting software product line feature models from natural language specifications. In *Proceedings of the 22nd International Systems and Software Product Line Conference (SPLC'18)*. ACM, 43–53. https://doi.org/10.1145/3233027.3233029

[24] Saman Tariq and Sehrish Munawar Cheema. 2021. Approaches for non-functional requirement modeling: a literature survey. In *Proceedings of the 4th International Conference on Computing & Information Sciences (ICCIS'21)*. IEEE, 1–6. https://doi.org/10.1109/ICCIS54243.2021.9676398

[25] Maurice H. ter Beek, Erik P. de Vink, and Tim A. C. Willemse. 2017. Family-Based Model Checking with mCRL2. In *Proceedings of the 20th International Conference on Fundamental Approaches to Software Engineering (FASE'17) (LNCS, Vol. 10202)*, Marieke Huisman and Julia Rubin (Eds.). Springer, 387–405. https://doi.org/10.1007/978-3-662-54494-5_23

[26] Maurice H. ter Beek, Sjef van Loo, Erik P. de Vink, and Tim A. C. Willemse. 2020. Family-Based SPL Model Checking Using Parity Games with Variability. In *Proceedings of the 23rd International Conference on Fundamental Approaches to Software Engineering (FASE'20) (LNCS, Vol. 12076)*, Heike Wehrheim and Jordi Cabot (Eds.). Springer, 245–265. https://doi.org/10.1007/978-3-030-45234-6_12