Global Decision Making Over Deep Variability in Feedback-Driven Software Development

Jörg Kienzle, Benoit Combemale, Gunter Mussbacher, Omar Alam, Francis Bordeleau, Loli Burgueño, Gregor Engels, Jessie Galasso, Jean-Marc Jézéquel, Bettina Kemme, et al.

Global Decision Making Over Deep Variability in Feedback-Driven Software Development

Jörg Kienzle
McGill University
Canada
Joerg.Kienzle@mcgill.ca

Benoit Combemale
University of Rennes
France
benoit.combemale@irisa.fr

Gunter Mussbacher
McGill University
Canada
Gunter.Mussbacher@mcgill.ca

Omar Alam
Trent University
Canada
omaralam@trentu.ca

Francis Bordeleau
Ecole de technologie superieure
Canada
francis.bordeleau@etsmtl.ca

Lola Burgueño
Open University of Catalonia
Spain
lburguenoc@uoc.edu

Gregor Engels
Paderborn University
Germany
engels@upb.de

Jessie Galasso
Université de Montréal
Canada
jessie.galasso-carbonnel@umontreal.ca

Jean-Marc Jezequel
University of Rennes
France
jezequel@irisa.fr

Bettina Kemme
McGill University
Canada
kemme@cs.mcgill.ca

Sebastien Mosser
McMaster University
Canada
mossers@mcmaster.ca

Houari Sahraoui
DIRO, Université de Montréal
Canada
sahraouh@iro.umontreal.ca

Max Schiedermeier
McGill University
Canada
max.schiedermeier@mcgill.ca

Eugene Syriani
University of Montreal
Canada
syriani@iro.umontreal.ca

ABSTRACT

To succeed with the development of modern software, organizations must have the agility to adapt faster to constantly evolving environments to deliver more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, business, development, and IT. However, stakeholders do not have sufficient automated support for global decision making, considering the increasing variability of the solution space, the frequent lack of explicit representation of its associated variability and decision points, and the uncertainty of the impact of decisions on stakeholders and the solution space. This leads to an ad-hoc decision making process that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. The Multi-Plane Models and Data (MP-MODA) framework explicitly represents and manages variability, impacts, and decision points. It enables automation and tool support in aid of a multi-criteria decision making process involving different stakeholders within a feedback-driven software development process where feedback cycles aim to reduce uncertainty. We present the conceptual structure of the framework, discuss its potential benefits, and enumerate key challenges related to tool supported automation and analysis within MP-MODA.

CCS CONCEPTS

• **Software and its engineering → Collaboration in software development**.

KEYWORDS

MODA, Iterative Software Development, Feedback Loop

ASE '22, October 10–14, 2022, Rochester, MI, USA

ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3559551

ACM Reference Format:
Jörg Kienzle, Benoit Combemale, Gunter Mussbacher, Omar Alam, Francis Bordeleau, Lola Burgueño, Gregor Engels, Jessie Galasso, Jean-Marc Jezequel, Bettina Kemme, Sebastien Mosser, Houari Sahraoui, Max Schiedermeier, and Eugene Syriani. 2022. Global Decision Making Over Deep Variability in Feedback-Driven Software Development. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3551349.3559551

1 INTRODUCTION

The software industry is experiencing revolutionary changes on how software is built and how its functionalities are offered to users [28]. Contemporary software systems are highly complex and heterogeneous, and their development involves a tremendous amount of knowledge from different domains. Furthermore, the unpredictability of an ever changing environment, and the unprecedented pace of emergence and decay of technologies require deep changes on how we develop, deploy, and evolve software [1]. To succeed with the development of modern software, organizations must have the agility to adapt faster to constantly evolving environments and to deliver more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, business, development, and IT.

For certain software, e.g., for smart phone applications, it has become easy to frequently release and deploy new versions, e.g., to address bug fixes, add support for new technology or introduce new features [25]. Because of this ease of updating, it is nowadays often *expected* that such software continuously evolves and improves.

One way to deal with the uncertainty of how an application must evolve is to explicitly plan for feedback-driven software evolution. The idea is to gather telemetry data from or about the running system [22], e.g., to measure the effects of the heterogeneity of execution environments, or to capture user preferences and expectations [5]. The data is then analyzed and the gained knowledge is used to guide the development of the next software release.

Over time, such evolution of software requires expertise and knowledge from many domains and levels of abstraction, ranging from low-level technical skills to legal and business know-how, and decisions made on one level and in one domain can have an impact on other domains as well. While DevOps [11] aims at structuring the teams and the system architecture to enable agility and autonomy to take decisions and make modifications locally, some decisions and modifications still have a more cross-cutting impact. Unfortunately, stakeholders currently do not have sufficient support for global decision making, considering the increasing variability of the solution space, the frequent lack of explicit representation of its associated variability and decision points, and the uncertainty of the impact of decisions. This leads to an ad-hoc decision making process that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives.

In this new idea paper, we outline a framework called Multi-Plane MODA, as an extension of the existing *Models and Data* (MODA) framework [8], that explicitly represents and manages variability of the problem and solution space, impacts, and decision points to enable automation and tool support in aid of a multi-criteria decision making process involving different stakeholders within a feedback-driven software development process. In Section 2, we present a motivating example and relevant background information. Then we describe the conceptual structure of the MP-MODA framework, discuss its potential benefits, and revisit the proposed motivating example in Section 3. We list key challenges of MP-MODA related to tool supported automation and analysis in Section 4.

2 MOTIVATION AND BACKGROUND

Motivating Example. Let us consider a company developing a new version of a software for a Heating, Ventilation and Air Conditioning (HVAC) system, which comprises both software and hardware components such as humidity and temperature sensors, coordinated processing units for data analysis, and actuators. Numerous stakeholders are involved and deal with different decisions. For instance, the business team decides on the functionality to be offered whereas the development team determines the infrastructure of the solution (whether to host it in the cloud or on premise). In addition, a decision may impact other decisions, e.g., deciding to use a public cloud rather than a private one requires deciding on the provider and services. Documenting the different available alternatives and decisions made by the different stakeholders as well as their impact on each other is essential to help manage this complexity.

Iterative development and evolving environments require making decisions throughout the whole development life cycle of the product as well as delaying some of them due to uncertainties in early iterations. For instance, when the development team lacks details about the amount of data that the client's sensors will provide per second, they will not be able to determine the server requirements necessary to process data in real-time. Furthermore, the HVAC system is subject to a process of continuous improvement, which means that the solution will be monitored and the stakeholders will use that information to revisit/refine their decisions. For instance, during the development of the HVAC, insufficient performance in extended analytics may indicate the need to update the cloud storage. In all these situations, introducing or changing previously made decisions is of vital importance.

MODA. In order to support an informed decision-making process in building and maintaining complex solutions, all relevant data, used models and particularly their interplay have to be understood and managed. The *Models and Data* (MODA) framework provides a conceptual structure, where different types of telemetry data (input, output, measured) and external data, as well as different roles of models (*descriptive*, *predictive*, *prescriptive*) are distinguished and where their role in a closed feedback loop are defined [8]. Hereby, a model in a *descriptive* role describes aspects of an existing, running system, in a *predictive* role aspects of future behaviour of a system, e.g., by means of simulation or extrapolation, and in a *prescriptive* role desired aspects of a system to be built.

Variability and Impact Modelling. Feature models (FMs) [16] are typically used to expose the available variability of an artifact that a stakeholder can choose from when building an application. A feature model expresses different features that an artifact encapsulates and describes their *optional*, *mandatory*, *include*, and *exclude* relationships. Impact models are used to reason about trade-offs when the developer chooses between alternative solutions. When developing a system, a developer would typically select the variant(s) with the best impact on relevant stakeholder goals and system qualities. These impacts can be specified using, e.g., goal models with GRL, which is part of the User Requirements Notation (URN) standard [15], or the NFR framework [7], i* [33], and KAOS [10], but any other technology for multi-criteria decision-making could be used for impact models. There are approaches that allow for automated reasoning of impacts when selecting features [3, 29]. Feature

and impact models are being used to support reuse and making decisions during the software development process [18, 27]. For example, when reusing the *Authentication* artifact in an application, the developer can use its feature model to select the appropriate variant, e.g., a *biometric-based* solution, or a *password-based* solution. The impact model would allow to make a trade-off analysis based on different impacts on high-level goals such as *level of security* or *user convenience*. Software Product Lines (SPL) [23] and self-adaptive systems technology [6, 17] help in that regard, and there exist already several multi-objective optimization tools for SPLs [14, 19, 29].

3 MULTI-PLANE MODA

The new idea put forward in this paper is to use a novel conceptual framework entitled *Multi-Plane MODA* (MP-MODA) to explicitly represent and manage variability of the problem and solution space, impacts of the different variants, and decision points. As such, MP-MODA enables automation and tool support in aid of a multi-criteria decision making process involving different stakeholders within a feedback-driven software development process where feedback cycles aim to reduce uncertainty. Figure 1 presents an overview of MP-MODA.

3.1 MODA Planes

At the center of the framework lies the notion of a *MODA plane*, which serves as a unit of modularization that addresses a development concern of interest related to the application under development. MODA planes, in contrast to other typical modularization units used in software development such as components [31], are significantly bigger units that encapsulate several variants or alternative ways of addressing a development issue and include one or multiple feedback loops. For example, typical MODA planes might encapsulate technologies such as different operating systems and ways of configuring them, or different cloud providers and service architectures, or different ways of authentication and variants of how to deal with unsuccessful authentication attempts and expiring credentials. However, MODA planes address business objectives just as they address technical objectives. Hence, a MODA plane may also describe business objectives (e.g., market share) and include features for various business opportunities.

The main elements of a MODA plane are shown on the left hand side of Figure 1 (red colors). To render decision making explicit, a MODA plane must define a *Variability Model* (cf. VM in Figure 1) that exposes the set of variants / configuration options encapsulated within in form of *features*. The VM should also make feature dependencies explicit, e.g., when one feature requires the presence of another one, or when one feature can not co-exist with another one. A MODA plane must also provide an *Impact Model* (IM) that lists the non-functional *Plane Properties and Qualities* of the encapsulated variants and how the different features affect those qualities to allow trade-off analysis when making selections.

A plane encapsulates not only executable code, but also other development artefacts such as models, documentation and sometimes even hardware. In line with Model-Driven Engineering practices [26] and as explained by the MODA framework reviewed in Section 2, the models encapsulated within a MODA plane can play a *descriptive*, *predictive* or *prescriptive* role. Furthermore, each plane defines potentially several feedback loops that define how to monitor the running system and gather *Telemetry Data* in order to gain insight into the properties of the current configuration of the plane. The data can be used to create descriptive models of the current version of the application for analysis purpose or to update the impact models.

Finally, each MODA plane also comes with *Plane Experts*, i.e., stakeholders that understand the concern that a plane addresses, that have knowledge about the variants the plane offers and know the involved trade-offs of selecting one variant over another. These experts are depicted as stick men to the left of the planes in Figure 1.

While there will always be MODA planes that are specific to the application that is being developed, we imagine that very quickly a set of standard MODA planes will emerge, encapsulating typical execution environments and platforms, or frameworks. For example, there could be a MODA plane for Java-based development with the Spring framework, or one for Unity-based development for computer games. These MODA planes basically encapsulate libraries and technologies that are already heavily being reused today, but augment them with a variability model, impact model and feedback loop. The standard MODA planes will themselves evolve over time, with new features being added and outdated ones retired, or when new measures are put in place to gather additional telemetry data to improve the feedback loops.

3.2 Inter-Plane Dependencies

To develop an application, multiple planes must be used in combination as shown by the stack of planes with different colors in Figure 1. Each plane exposes its features and configuration options in its variability model, and the consequences of choosing a feature for its stakeholders in the impact model. Each plane gathers telemetry data which drives one or several feedback loops.

When multiple planes are used in conjunction to develop an application, a *Decision Space* is formed in MP-MODA, visualized by the grey models and grey feedback loop in Figure 1.

The experts and developers of each plane have already expressed feature dependencies internal to the plane, e.g., when a feature requires some other feature, or when features are mutually exclusive. Within a decision space, though, there can be additional cross-plane feature constraints, which MP-MODA captures with explicit *inter-plane feature dependencies* (cf. black lines between VMs of different planes in Figure 1) as well as automatically discovered ones.

Furthermore, there can be indirect feature dependencies, caused by the fact that features in different planes impact the same properties and qualities of the software under development. To this aim, MP-MODA allows developers to define *Decision Space Properties* (cf. D.S. Props in Figure 1) and express mappings or conversions between the different plane properties and the decision space properties (cf. black lines from IMs to D.S. Props).

Inter-plane feature dependencies, decision space properties and property conversions fuse all variability and impact models together to enable global decision making.

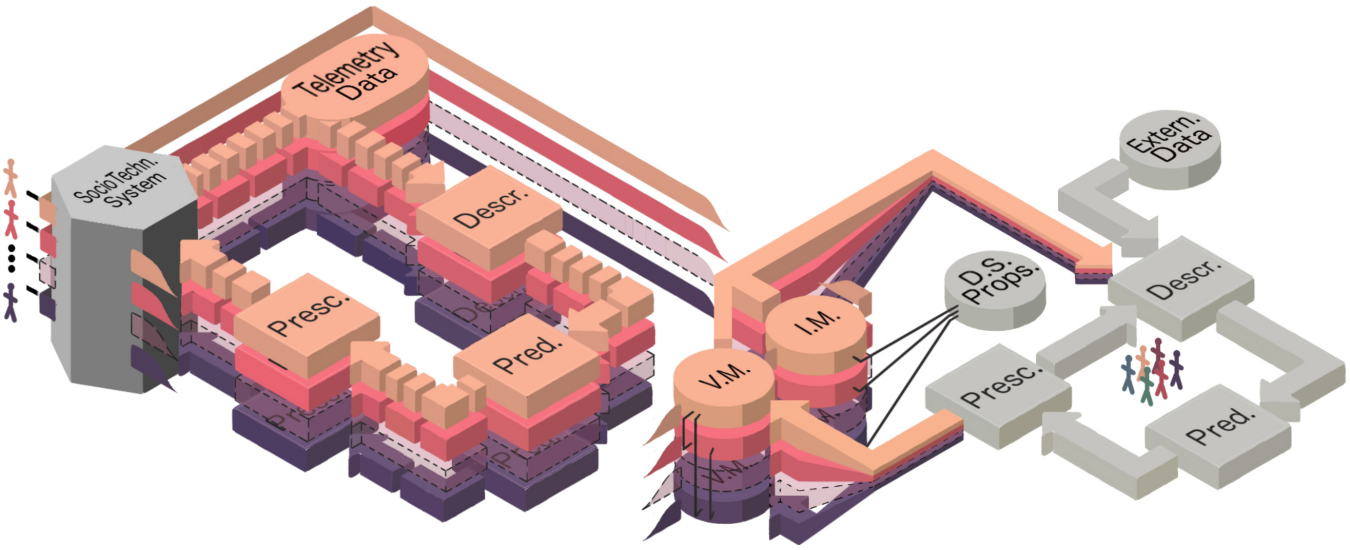

Figure 1: Overview of the MP-MODA Framework

3.3 Making Decisions

The goal of the decision space is to run a global feedback loop in which decisions are made with respect to the current or subsequent software versions by considering feedback from one or several planes, involving experts and stakeholders from all relevant planes.

The descriptive model of the decision space is essentially composed of the union of all variability models of the contained planes and their current configurations. When gathered telemetry data or external data (e.g., gathered knowledge from environment) points to a situation that needs improvement, the fused variability models and impact models of the different planes can be used to explore different configurations and determine whether any of them would result in an improvement. The inter-plane dependencies prevent incorrect configurations from being explored.

When developing a software, a set of objectives is imposed to the development team to satisfy the many requirements imposed by stakeholders as well as legal and business constraints. When decisions are made it is of paramount importance that the objectives with hard constraints are still fulfilled, and the soft objectives are optimized. This is where MP-MODA shines. Whenever feedback from one or several planes suggests to update a plane's configuration, the inter-plane feature dependencies will determine whether this change requires other changes.

Furthermore, thanks to the connections between features, plane properties and decision space properties, MP-MODA can detect indirect dependencies that can be just as important to consider for decision making as direct dependencies. For example, the hard objectives of the software might require to guarantee a certain response time for a service. Based on gathered feedback, an expert of a plane might suggest to enable a new feature that uses an algorithm that slows down performance and would hence endanger the response time. However, other changes could be made to save computation time somewhere else and still meet the required response time objective. To make the decision, MP-MODA can determine all the impacts the proposed feature change has, find the properties and ultimately the objectives that are impacted by that change, and involve all of the associated stakeholders and experts in the decision making process if practicable. MP-MODA can also suggest configuration changes to compensate for the negative impacts of a feature by suggesting alternative features that positively affect the impacted properties. In that case, the plane experts associated with the suggested features are involved in the decision process as well.

To summarize: Every time decisions need to be made when working on the next software release, the decision space in MP-MODA can assist in determining the global consequences of a planned change, and involve the right stakeholders and experts in the decision making process.

3.4 Motivating Example Revisited

Let us consider how MP-MODA could help in our HVAC motivating example by exploring how decisions made by one stakeholder could impact the remaining stakeholders.

The HVAC company decided to define 3 planes addressing business, software development and operations concerns, respectively. The experts of each plane make decisions among the available options within their attributed plane, and thanks to the documented dependencies between decisions (intra- and inter-planes), they can also assess the impact of their decisions on the decisions made in other planes. For example, the operations experts may realize that the current CPUs are most of the time at 90% usage and consider replacing them with more powerful processing units. While examining the potential options captured in their variability model, MP-MODA alerts them that changing the processors can impact choices made by the experts of the development plane, as it may cause incompatibilities with some of the used software libraries. Thanks to the input provided by MP-MODA, the operations experts decide to meet with the development experts. The former experts were already aware of the computing issue and had a plan to introduce software optimizations to fix the computational cost.

When the development team assesses whether it is better to proceed with their planned optimizations or replace the hardware and therefore update their libraries, MP-MODA also helps them make the best decision by pointing to their intra-plane dependencies. E.g., MP-MODA may point to the fact that updating libraries has an impact on the architecture of the solution. Based on knowledge gathered from previous iterations provided through the feedback loop, they estimate that adapting the architecture can introduce important delays in the delivery of the product, which can further impact decisions made by the business experts about release dates.

By being aware of these inter- and intra-plane dependencies, the development and operations experts were able to make an informed global decision and agree that the best solution is not to update the hardware but to prioritize the work of the development team on the software optimizations.

4 AUTOMATION AND TOOL CHALLENGES

MP-MODA requires significant automation and tool support to be effective in practice. This section briefly points to the most important challenges we identified, and we call on the ASE and SE community at large to intensify research efforts in this direction.

Support Within and Across Planes. Automation and tool support needed within a plane to realize MP-MODA exists for the most part already. For example, many execution platforms or frameworks provide logging support that can be used to gather telemetry data at runtime. Monitoring and data aggregation technology exists that makes it possible to reduce uncertainty or deal with the overwhelming amount of gathered data to focus on the essential information [32]. Techniques such as those promoted by the models@runtime community allow to reason about the current state of the system at a higher level of abstraction [4]. All this should help to capture and quantify the impact of features offered by the plane. Finally, tool support for configuring planes and enacting new configurations, as well as customizing planes with open variability to specific application context is required. While dedicated approaches exist (*e.g.,* CVL [13]), they need to be integrated with the concept of planes.

Across planes the challenge is to manage dependencies between planes. SPL approaches such as staged configurations [9], multi product lines [12] and delaying decisions [18] are a good starting point. The sheer number of dependencies however makes it unlikely or even infeasible to manually specify all relevant ones. Sophisticated support is hence needed to discover missing direct and indirect dependencies. While quantifying impacts is already a challenge within a plane, aligning of impacts across planes adds additional complexity. Planes deal with development concerns from different domains, and hence the way of measuring and quantifying impacts will likely differ, even for impacts that are easily quantifiable, e.g., performance. Ontology alignment technology can most likely help here. Finally, standardization of plane interfaces and protocols could alleviate this alignment challenge in a long run.

Decision Making Support. Automation and tool support for decision making is central to MP-MODA. The automation of the required multi-objective optimization for making informed decisions faces of course a significant scalability challenge. Techniques such as managing diversity in the decision space exploration can speed up the convergence [2].

First, the framework has to determine the stakeholders and experts that have to be involved in a decision that needs to be made. Sophisticated support is hence needed to discover and keep track of key stakeholders and experts for individual features. Then, tool support enabling effective human-computer interaction is crucial. For example, advanced visualization techniques to help understand configurations and dependencies are needed [20, 30]. Intuitive exploration of what-if scenarios generated using predictive models is highly important [21]. But most importantly, MP-MODA tools need to communicate with all the stakeholders to inform them of the proposed modifications and decisions that need to be made, get their input, engage them in discussions and allow them to approve the final decision. Functionality offered nowadays by modern version control software and collaborative platforms, e.g., GIT pull requests, could be adapted for that purpose. Chat-bots might also be an interesting direction to explore [24], especially for stakeholders from non-technical communities.

REFERENCES

[1] Mathieu Acher. 2022. Reproducible Science and Deep Software Variability. In *VaMoS '22: 16th International Working Conference on Variability Modelling of Software-Intensive Systems, Florence, Italy, February 23 - 25, 2022*, Paolo Arcaini, Xavier Devroey, and Alessandro Fantechi (Eds.). ACM, 1:1–1:2. https://doi.org/10.1145/3510466.3510481

[2] Edouard R Batot and Houari Sahraoui. 2022. Promoting social diversity for the automated learning of complex MDE artifacts. *Software and Systems Modeling* (2022), 1–20.

[3] David Benavides, Pablo Trinidad, and Antonio Ruiz-Cortés. 2005. Automated Reasoning on Feature Models. In *Proceedings of the 17th International Conference on Advanced Information Systems Engineering* (Porto, Portugal) *(CAiSE'05)*. Springer-Verlag, Berlin, Heidelberg, 491–503.

[4] Gordon Blair, Nelly Bencomo, and Robert B France. 2009. Models@ run. time. *Computer* 42, 10 (2009), 22–27.

[5] Petra Bosch-Sijtsema and Jan Bosch. 2015. User Involvement throughout the Innovation Process in High-Tech Industries. *Journal of Product Innovation Management* 32, 5 (2015), 793–807. https://doi.org/10.1111/jpim.12233 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/jpim.12233

[6] Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Di Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi, Gabor Karsai, Holger M. Kienle, Jeff Kramer, Marin Litoiu, Sam Malek, Raffaela Mirandola, Hausi A. Müller, Sooyong Park, Mary Shaw, Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle. 2009. *Software Engineering for Self-Adaptive Systems: A Research Roadmap.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–26. https://doi.org/10.1007/978-3-642-02161-9_1

[7] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. 2000. *Non-Functional Requirements in Software Engineering.* Springer. 476 pages.

[8] Benoit Combemale, Jorg Kienzle, Gunter Mussbacher, Hyacinth Ali, Daniel Amyot, Mojtaba Bagherzadeh, Edouard Batot, Nelly Bencomo, Benjamin Benni, Jean-Michel Bruel, Jordi Cabot, Betty H.C. Cheng, Philippe Collet, Gregor Engels, Robert Heinrich, Jean-Marc Jezequel, Anne Koziolek, Sebastien Mosser, Ralf Reussner, Houari Sahraoui, Rijul Saini, June Sallou, Serge Stinckwich, Eugene Syriani, and Manuel Wimmer. 2021. A Hitchhiker's Guide to Model-Driven Engineering for Data-Centric Systems. *IEEE Software* 38, 4 (2021), 71–84.

[9] Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. 2005. Staged configuration through specialization and multilevel configuration of feature models. *Software Process: Improvement and Practice* 10, 2 (2005), 143–169.

[10] A. Dardenne, A. van Lamsweerde, and S. Fickas. 1993. Goal-directed Requirements Acquisition. *Science of Computer Programming* 20 (1993), 3–50.

[11] Jennifer Davis and Ryn Daniels. 2016. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale.* O'Reilly. 410 pages.

[12] Deepak Dhungana, Dominik Seichter, Goetz Botterweck, Rick Rabiser, Paul Grunbacher, David Benavides, and Jose A. Galindo. 2011. Configuration of Multi Product Lines by Bridging Heterogeneous Variability Modeling Approaches. In *2011 15th International Software Product Line Conference.* 120–129. https://doi.org/10.1109/SPLC.2011.22

[13] Øystein Haugen, Andrzej Wąsowski, and Krzysztof Czarnecki. 2013. CVL: Common Variability Language. In *Proceedings of the 17th International Software Product Line Conference* (Tokyo, Japan) *(SPLC '13)*. Association for Computing Machinery, New York, NY, USA, 277. https://doi.org/10.1145/2491627.2493899

[14] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. 2015. Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 517–528. https://doi.org/10.1109/ICSE.2015.69

[15] International Telecommunication Union (ITU-T). approved October 2012. Recommendation Z.151 (10/12): User Requirements Notation (URN) - Language Definition.

[16] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Technical Report CMU/SEI-90-TR-21. Software Engineering Institute, Carnegie Mellon University.

[17] J. O. Kephart and D. M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (Jan 2003), 41–50. https://doi.org/10.1109/MC.2003.1160055

[18] Jörg Kienzle, Gunter Mussbacher, Philippe Collet, and Omar Alam. 2016. Delaying decisions in variable concern hierarchies. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*. 93–103.

[19] Jabier Martinez, Daniel Strüber, Jose Miguel Horcas, Alex Burdusel, and Steffen Zschaler. 2022. *Acapulco: An extensible tool for identifying optimal and consistent feature model configurations.*

[20] Johann Mortara, Philippe Collet, and Anne-Marie Dery-Pinna. 2021. Visualization of Object-Oriented Variability Implementations as Cities. In *2021 Working Conference on Software Visualization (VISSOFT)*. 76–87.

[21] Gunter Mussbacher, Daniel Amyot, Ruth Breu, Jean-Michel Bruel, Betty H. C. Cheng, Philippe Collet, Benoît Combemale, Robert B. France, Rogardt Heldal, James H. Hill, Jörg Kienzle, Matthias Schöttle, Friedrich Steimann, Dave R. Stikkolorum, and Jon Whittle. 2014. The Relevance of Model-Driven Engineering Thirty Years from Now. In *Model-Driven Engineering Languages and Systems - 17th International Conference, MODELS 2014, Valencia, Spain, September 28 - October 3, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8767)*, Jürgen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfrán (Eds.). Springer, 183–200. https://doi.org/10.1007/978-3-319-11653-2_12

[22] Helena Holmström Olsson and Jan Bosch. 2014. *The HYPEX Model: From Opinions to Data-Driven Software Development.* Springer International Publishing, Cham, 155–164. https://doi.org/10.1007/978-3-319-11283-1_13

[23] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer-Verlag, Secaucus, NJ, USA.

[24] Ranci Ren, John W. Castro, Adrián Santos, Sara Pérez-Soler, Silvia Teresita Acuña, and Juan de Lara. 2020. Collaborative Modelling: Chatbots or On-Line Tools? An Experimental Study. In *EASE '20: Evaluation and Assessment in Software Engineering, Trondheim, Norway, April 15-17, 2020*, Jingyue Li, Letizia Jaccheri, Torgeir Dingsøyr, and Ruzanna Chitchyan (Eds.). ACM, 260–269. https://doi.org/10.1145/3383219.3383246

[25] Tony Savor, Mitchell Douglas, Michael Gentili, Laurie Williams, Kent Beck, and Michael Stumm. 2016. Continuous Deployment at Facebook and OANDA. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 21–30.

[26] D.C. Schmidt. 2006. Guest Editor's Introduction: Model-Driven Engineering. *Computer* 39, 2 (feb 2006), 25–31.

[27] Matthias Schöttle, Nishanth Thimmegowda, Omar Alam, Jörg Kienzle, and Gunter Mussbacher. 2015. Feature modelling and traceability for concern-driven software development with TouchCORE. In *Companion Proceedings of the 14th International Conference on Modularity*. 11–14.

[28] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. 2017. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access* 5 (2017), 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

[29] Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, Sven Apel, and Gunter Saake. 2012. SPL Conqueror: Toward Optimization of Non-functional Properties in Software Product Lines. *Software Quality Control* 20, 3-4 (Sept. 2012), 487–517.

[30] Margaret-Anne D Storey, Davor Čubranić, and Daniel M German. 2005. On the use of visualization to support awareness of human activities in software development: a survey and a framework. In *Proceedings of the 2005 ACM symposium on Software visualization*. 193–202.

[31] Clemens Szyperski, Dominik Gruntz, and Stephan Murer. 2002. *Component Software: Beyond Object-Oriented Programming* (2 ed.). ACM Press and Addison-Wesley, New York, NY.

[32] James Turnbull. 2014. *The art of monitoring.* James Turnbull.

[33] Eric Yu. 1995. *Modelling strategic relationships for process reengineering.* Ph.D. Dissertation. Department of Computer Science, University of Toronto.