Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Introduction

Quantitative Variability Modelling and Analysis

Maurice H. ter Beek[1], **Axel Legay**[2]

[1] ISTI–CNR, Pisa, Italy
e-mail: maurice.terbeek@isti.cnr.it

[2] Université Catholique de Louvain, Belgium
e-mail: axel.legay@uclouvain.be

Received: date / Revised version: date

Abstract Over the last decade, the explicit introduction and management of variability in the development cycle of (software) systems has led to a plethora of modelling and analysis techniques tailored to deal with such highly configurable (variational) systems. Most of the work on variability modelling and analysis, however, focusses on qualitative (i.e. functional) requirements. This Special Section of the Foundations for Mastering Change (FoMaC) track of the International Journal on Software Tools for Technology Transfer (STTT) on Quantitative Variability Modelling and Analysis (QSPL) aims to provide a comprehensive overview of the latest approaches to variability modelling and analysis techniques which specifically take quantities into account. In this paper, we first introduce Quantitative Variability Modelling and Analysis, after which we briefly describe the contents of the six papers that constitute this Special Section.

Key words: Variability · Quantitative modeling · Quantitative analysis · QSPL

1 Introduction

Software Product Line Engineering (SPLE) aims to develop and manage, in a cost-effective and time-efficient manner, a family of highly configurable software systems [1–3]. The key point is to manage the *variability* among the products (or variants) of a Software Product Line (SPL). This variability is expressed in terms of features, which are (user-visible) increments in functionality between software products. Individual products (a.k.a. variants) thus share a common set of features, but differ with respect to specific features, implying that a product can be viewed as a set of features. This is typically captured in an overall reference variability model of the family, also called a feature model, whose purpose is to define which combinations of features (i.e., which products) are valid.

The explicit introduction and management of feature-based variability in the software development cycle causes complexity in the *modelling and analysis* of SPLs. An important example concerns behavioural validation, i.e., guaranteeing that each product of the SPL satisfies a series of behavioural requirements. Variants of this problem include computing a set of products that do not satisfy the requirements together with a justification. A common approach to assess these requirements consists of building the system and testing it. However, if the system should fail to meet the requirements, then costly interactions are needed to improve it. This problem is amplified in SPLE, where the number of products typically grows exponentially with the number of features. In fact, it is generally acknowledged that two major challenges with respect to behavioural validation of SPLs are i) to offer a compact and efficient way of modelling the behaviour of products that share common features, and ii) to offer efficient analysis algorithms to exploit such models [4].

Over the last decade, we have witnessed a lot of efforts on lifting well-known formal specification languages and formal verification techniques from (single system) software engineering to SPLE or to configurable systems in general (cf., e.g., [5–24]). However, the vast majority of work on variability modelling and analysis focusses on qualitative (i.e., functional) requirements. Only recently, we have been witnessing a growing interest in modelling and analysis techniques that explicitly consider *quantitative* (i.e. non-functional) requirements, such as dependability, which encompasses attributes like availability and reliability, but also cost and security [25–32]. Today's software is embedded in a wide variety of critical systems (e.g., aircraft, railways, automotive, and medical devices) that run in environments where events occur

randomly and affect the system (e.g., think of failures). For these reasons, quantitative modelling and analysis (e.g., through probabilistic systems and probabilistic or statistical model checking) is nowadays receiving a lot of attention. In the specific setting of variational systems, this leads to modelling and analysis techniques that need to cope with the complexity of systems stemming from behaviour, variability, and randomness.

This Special Section is dedicated to what we coined Quantitative Variability Modelling and Analysis [33]. It contains a total of six papers, which were selected by means of a thorough reviewing process after an open call for papers on the following, non-exhaustive, topics of interest:

- Quantitative specification and verification techniques for systems with variability;
- Modelling and analysis of real-time, hybrid, or probabilistic systems with variability;
- Analysis of safety, security, or dependability properties of systems with variability;
- Modelling and analysis of dynamic, adaptive, and (runtime) reconfigurable systems.

In the remainder of this introductory paper, we contextualize the contributions contained in this Special Section. First we provide an overview of quantitative variability modelling languages and quantitative variability analysis techniques from the recent literature, after which we briefly describe the contents of the six papers that together constitute this Special Section.

2 Quantitative Variability Modelling

Arguably the most well-known behavioural SPL model is a Featured Transition System (FTS) [14]. An FTS models a family of behavioural product models in the form of Labelled Transition Systems (LTSs), which can be obtained by projecting on the feature expressions (Boolean formulae defined over the set of features) that decorate the transitions. Each transition whose feature expression is not satisfied by the specific product's set of features is abstracted from. In [25], Featured Timed Automata (FTA) are introduced along the lines of the aforementioned generalisation of LTSs into FTSs, to model the behaviour of real-time SPLs, whereas FTSs are generalised to cater for quantitative feature attributes in [26].

In [27], so-called Variable Sequence Diagrams are introduced. These are UML Sequence Diagrams enriched with variation points and annotated with quality parameters, catering for high-level behavioural modelling of SPLs. These models can automatically be transformed into DTMCs or, appropriately parametrized, into parametric DTMCs (with Rewards) for reliability (and energy) analysis, which can be fed into off-the-shelf probabilistic or parametric model checkers for the analysis of non-functional properties of different configurations of an SPL.

In [28], performance properties of SPLs are modelled by annotating UML Activity Diagrams (with, e.g., the duration to execute an activity of an activity node) and by interpreting them as Continuous-Time Markov Chains (CTMCs). Combined with the well-known delta-oriented SPL modelling approach of automated product derivation for SPLs, which is based on so-called deltas that specify the changes to be applied incrementally to a core product (cf. e.g. [8]), this approach constitutes the first attempt to efficient performance modelling of SPLs.

The high-level domain-specific language QFLan, presented in [31], allows the specification of an SPL in terms of a feature model with quantitative attributes and probabilistic behaviour subject to quantitative constraints. The language's behavioural part enables the dynamic installation, removal and replacement of features. QFLan comes equipped with a Discrete-Time Markov Chain (DTMC) semantics, permitting quantitative variability analyses (cf. Sect. 3).

3 Quantitative Variability Analysis

The tool suite ProVeLines [13] supports the analysis of discrete as well as real-time SPL models, various types of computations, and advanced feature notions. All tool variants share the same common high-level specification language fPROMELA, an feature-based dialect of SPIN's input language, to be used on top of FTSs. It includes the well-known SPL model checker SNIP [12] for the verification of fLTL (feature LTL) properties over FTSs (cf. Sect. 2) with dedicated SPL model-checking algorithms, treating features as first-class citizens and with built-in support for feature models. In [26], SMT solving is implemented on top of SNIP (with Z3), i.e., behavioural SPL models written in fPromela with additional quantitative constraints.

In [29], a feature-aware extension of DTMCs, so-called Featured Discrete-Time Markov Chains (FDTMCs), are introduced to capture stochastic behaviour of SPLs by associating each transition with the probability of a it being executed in a product. Verification of dependability (i.e., the probability to reach a success state) is formulated in PCTL (Probabilistic CTL). Furthermore, three dedicated (family-based) SPL model-checking techniques are defined to verify stochastic SPL behaviour modelled as FDTMCs, derived from annotated UML Sequence Diagrams as in [27] (cf. Sect. 2).

ProFeat [30], built on top of the probabilistic model checker PRISM, caters for the analysis of feature-aware probabilistic models. It provides a guarded-command language for modelling families of probabilistic systems, as well as an automatic translation of such SPL models to the (feature-less) input language of PRISM. ProFeat can deal with probabilistic (dynamic) SPLs by offering

dynamic feature switching (i.e., the activation and deactivation of features at runtime) and with feature attributes.

The tool QFLan, presented in [32], is a quantitative modelling and verification environment for highly (re)configurable systems, such as SPLs. It offers a modern Eclipse-based integrated development environment for the homonym probabilistic feature-oriented language (cf. Sect. 2) and advanced statistical model-checking analyses, such as the likelihood of specific behaviour or the expected average value of quantitative aspects related to feature attributes.

4 Contributions

Sampling Strategies for Product Lines with Unbounded Parametric Real-time Constraints. In [34], Luthmann et al. propose a novel sampling technique for effectively testing behavioural SPL models from a configuration space that is infinite due to the presence of real-time parameters and constraints. They build on their earlier work in which FTA were extended with freely configurable parameters, resulting in so-called Configurable Parametric Timed Automata (CoPTA), and an efficient family-based methodology to generate test suites from CoPTA models was defined. They propose a white-box testing approach, which employs solution-space knowledge in terms of configurations obtained from CoPTA models to generate samples to cover critical best-/worst-case execution-time behaviour. The approach has been implemented in a toolset and evaluated in terms of computation time and generated sample size.

Verification and Abstraction of Real-Time Variability-Intensive Systems. In [35], Cordy and Legay extend their previous work on FTA (cf. Sect. 2) by transferring existing work to the domain of timed SPL systems. First they nicely illustrate the benefits and need of feature-based extensions also for timed SPL systems, namely timing behaviour can crucially depend on the given feature combination, i.e., on the considered product. Furthermore, family-based variability modelling is more sustainable and better maintainable, while family-based variability analysis is an efficient means of verifying an entire SPL at once rather than for each product separately. Next they define a method based on anti-chains to decide and efficiently compute reachability in FTA, and they discuss how to apply standard abstraction techniques to FTA, including a CEGAR-based approach to optimize the automated reachability analysis.

Configuration of Inter-Process Communication with Probabilistic Model Checking. In [36], Herrmann et al. address the application of parametric variants of probabilistic model checking-based parameter synthesis for Markov models to configure inter-process communication protocols that are affected by errors caused by randomly occurring bit-flips. They provide a tool to automatically construct such Markov models from a high-level description of the processes and their communication structure, after which they consider a particular system instantiation: a Markov chain with a limited number of states, but for which certain transition probabilities (e.g., of restarting the system periodically and of an error occurring) are very small. The restart and error detection probabilities are under the control of the system designer, i.e., these are considered as tunable parameters. The configuration criterion is to maximize the system's availability, i.e., the steady-state probability of neither performing a restart nor having a silent data corruption. The authors then use several (parametric) probabilistic model-checking approaches to explore the dependencies between the restart and error detection probabilities. This complements their earlier work by focussing on experimental evaluation and by providing a detailed discussion of insights and lessons learned from the results of the experiments.

Quantitative Properties of Featured Automata. In [37], Fahrenberg and Legay introduce featured weighted automata, whose transitions are annotated with functions from feature expressions to weights, as abstract models of the quantitative behaviour of SPLs. This is achieved by combining FTSs and (semiring-) weighted automata, thus extending their previous work to applications in non-idempotent settings by considering general semirings rather than $*$-continuous Kleene algebras. They show how to lift verification techniques from weighted automata to featured weighted automata and they develop algorithms to compute quantitative properties (e.g., minimum reachability and feature-based energy properties) of featured weighted automata for all sets of features at once (i.e., family-based analysis). This is based on the main insight of this paper, which is that checking properties on a featured weighted automaton is equivalent to checking properties on a weighted automaton with a semiring that is a mapping from products to weights.

Applying Supervisory Control Synthesis to Priced Featured Automata and Energy Problems. In [38], Basile applies supervisory control synthesis to Priced Featured Automata (PFA), a.k.a. weighted featured automata, by translating a given PFA to an Extended Finite-state Automaton (EFA), which includes data variables and distinguishes between controllable and uncontrollable transitions. This enables the use of state-of-the-art synthesis algorithms from supervisory control theory to solve game-based problems (e.g., (featured) energy problems). The proposed method has the drawback that a priori, a most permissive controller needs to be synthesized for every product of the SPL, but the author shows how to use three-valued logic and partial-order reduction based on the SPL structure to greatly reduce the number of controllers required (in the best case, a number of con-

trollers that is linear in the number of features of the SPL).

Automatic Refactoring of Delta-Oriented SPLs to Remove-free Form and Replace-free Form. In [39], Damiani et al. introduce remove-free and replace-free normal forms for delta-oriented SPLs and explain how to refactor a specific form of delta-oriented SPLs of Java-like programs into these normal forms. Refactorings are program transformations that alter a program's internal structure without changing its observable behaviour, which is generally acknowledged to improve program comprehensibility by reducing complexity while improving readability. The refactoring algorithms developed by the authors are applicable to Imperative Featherweight Delta Java, which is a core calculus for delta-oriented SPLs, and they can be applied without requiring interaction with the developers of the SPL. Finally, delta-oriented SPLs in remove-free or replace-free normal form are expected to improve the efficiency of dynamic reconfiguration of such SPLs.

5 Conclusion

We have provided an overview of recent approaches to quantitative variability modelling languages and quantitative variability analysis techniques, followed by a presentation of the papers constituting this Special Section of the Foundations for Mastering Change (FoMaC) track of the International Journal on Software Tools for Technology Transfer (STTT) on Quantitative Variability Modelling and Analysis (QSPL). The fact that QSPL is a currently a hot topic is witnessed by the panel on Quantitative Variability Modeling and Analysis that was held at the 13th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'19) in Leuven, Belgium [33]. This Special Section on QSPL complements earlier collections on variability modelling and analysis focussing instead on qualitative requirements (cf., e.g., [40–42]).

In the future, ideally, the variability modelling and analysis approaches described in the papers constituting this Special Section should be applied to SPLs from industry. To be effectively applicable to system design in practice, the presented formal modelling languages and analysis techniques typically require further development and implementation such that they encompass both qualitative and quantitative aspects at an appropriate level of abstraction (cf., e.g., [43,44]). To this aim, the authors of [34] plan to extract CoPTA models directly from source code (e.g., C code) and to generalize their approach to incorporate also non-functional properties. The authors of [35] intend to evaluate their abstraction methods on challenging case studies that involve heavy timed constraints as well as large variability, which may very well require them to consider additional abstraction heuristics to achieve acceptable computation times. The authors of [36] want to extend their tool support to consider also the occurrence of errors in processes and messages. The authors of [37] are interested in extending the setting of their approach to enable feature-based treatment of properties like limit-average cost, which would allow an extension to the real-time SPLs of [25]. The author of [38] foresees to implement his approach as a plugin to allow the tools for PFA and FTSs mentioned above to interact with tools for Supervisory Control Theory such as CIF 3 [45,46]. The authors of [39] would like to adapt their refactoring algorithms to so-called delta-oriented SPLs of FSL statecharts and to integrate them into the HyVar toolchain [47] and into the ABS toolchain (`http://abs-models.org/`), which is based on a delta-oriented modelling language that has been successfully used in industry. This would allow to apply their refactorings on concrete industrial case studies to evaluate whether they can indeed improve the comprehensibility of SPLs.

Acknowledgements. We would like to thank all authors for their contributions and the reviewers of this Special Section for their reviews.

References

1. P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.
2. K. Pohl, G. Böckle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.
3. S. Apel, D. S. Batory, C. Kästner, and G. Saake. *Feature-Oriented Software Product Lines: Concepts and Implementation*. Springer, 2013.
4. A. Legay and G. Perrouin. On Quantitative Requirements for Product Lines. In M. H. ter Beek, N. Siegmund, and I. Schaefer, editors, *Proceedings of the 11th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'17)*, pages 2–4. ACM, 2017.
5. K. G. Larsen, U. Nyman, and A. Wąsowski. Modal I/O Automata for Interface and Product Line Theories. In R. De Nicola, editor, *Proceedings of the 16th European Symposium on Programming (ESOP'07)*, volume 4421 of *LNCS*, pages 64–79. Springer, 2007.
6. A. Gruler, M. Leucker, and K. D. Scheidemann. Modeling and Model Checking Software Product Lines. In G. Barthe and F. S. de Boer, editors, *Proceedings of the 10th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'08)*, volume 5051 of *LNCS*, pages 113–131. Springer, 2008.
7. K. Lauenroth, K. Pohl, and S. Töhning. Model Checking of Domain Artifacts in Product Line Engineering. In *Proceedings of the 24th International Conference on Automated Software Engineering (ASE'09)*, pages 269–280. IEEE, 2009.
8. D. Clarke, M. Helvensteijn, and I. Schaefer. Abstract Delta Modeling. *ACM SIGPLAN Not.*, 46(2):13–22, Oct 2010.

9. P. Asirelli, M. H. ter Beek, A. Fantechi, and S. Gnesi. A Model-Checking Tool for Families of Services. In R. Bruni and J. Dingel, editors, *Proceedings of the 13th International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'11) and 31st International Conference on FORmal TEchniques for Networked and Distributed Systems (FORTE'11)*, volume 6722 of *LNCS*, pages 44–58. Springer, 2011.
10. M. Erwig and E. Walkingshaw. The Choice Calculus: A Representation for Software Variation. *ACM Trans. Softw. Eng. Methodol.*, 21(1), 2011.
11. M. H. ter Beek, F. Mazzanti, and A. Sulova. VMC: A Tool for Product Variability Analysis. In D. Giannakopoulou and D. Méry, editors, *Proceedings of the 18th International Symposium on Formal Methods (FM'12)*, volume 7436 of *LNCS*, pages 450–454. Springer, 2012.
12. A. Classen, M. Cordy, P. Heymans, A. Legay, and P.-Y. Schobbens. Model checking software product lines with SNIP. *Int. J. Softw. Tools Technol. Transf.*, 14(5):589–612, 2012.
13. M. Cordy, A. Classen, P. Heymans, P.-Y. Schobbens, and A. Legay. ProVeLines: a product line of verifiers for software product lines. In *Proceedings of the 17th International Software Product Line Conference (SPLC'13)*, volume 2, pages 141–146. ACM, 2013.
14. A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin. Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. *IEEE Trans. Softw. Eng.*, 39(8):1069–1089, 2013.
15. M. H. ter Beek and E. P. de Vink. Using mCRL2 for the Analysis of Software Product Lines. In S. Gnesi and N. Plat, editors, *Proceedings of the 2nd FME Workshop on Formal Methods in Software Engineering (FormaliSE@ICSE'14)*, pages 31–37. ACM, 2014.
16. T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Comput. Surv.*, 47(1), 2014.
17. M. Tribastone. Behavioral Relations in a Process Algebra for Variants. In *Proceedings of the 18th International Software Product Line Conference (SPLC'14)*, pages 82–91. ACM, 2014.
18. M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. Modelling and analysing variability in product families: Model checking of modal transition systems with variability constraints. *J. Log. Algebr. Meth. Program.*, 85(2):287–315, 2016.
19. M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. Statistical Model Checking for Product Lines. In T. Margaria and B. Steffen, editors, *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*, volume 9952 of *LNCS*, pages 114–133. Springer, 2016.
20. A. S. Dimovski, A. S. Al-Sibahi, C. Brabrand, and A. Wąsowski. Efficient family-based model checking via variability abstractions. *Int. J. Softw. Tools Technol. Transf.*, 5(19):585–603, 2017.
21. M. Lochau, S. Mennicke, H. Baller, and L. Ribbeck. Incremental model checking of delta-oriented software product lines. *J. Log. Algebr. Meth. Program.*, 85(1):245–267, 2016.
22. R. Muschevici, J. Proença, and D. Clarke. Feature Nets: behavioural modelling of software product lines. *Softw. Sys. Model.*, 15(4):1181–1206, 2016.
23. M. H. ter Beek, E. P. de Vink, and T. A. C. Willemse. Family-Based Model Checking with mCRL2. In M. Huisman and J. Rubin, editors, *Proceedings of the 20th International Conference on Fundamental Approaches to Software Engineering (FASE'17)*, volume 10202 of *LNCS*, pages 387–405. Springer, 2017.
24. M. H. ter Beek, F. Damiani, M. Lienhardt, F. Mazzanti, and L. Paolini. Static Analysis of Featured Transition Systems. In *Proceedings of the 23rd International Systems and Software Product Line Conference (SPLC'19)*. ACM, 2019.
25. M. Cordy, P.-Y. Schobbens, P. Heymans, and A. Legay. Behavioural Modelling and Verification of Real-Time Software Product Lines. In *Proceedings of the 16th International Software Product Line Conference (SPLC'12)*, pages 66–75. ACM, 2012.
26. M. Cordy, P.-Y. Schobbens, P. Heymans, and A. Legay. Beyond Boolean Product-Line Model Checking: Dealing with Feature Attributes and Multi-features. In *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*, pages 472–481. IEEE, 2013.
27. C. Ghezzi and A. Molzam Sharifloo. Model-based verification of quantitative non-functional properties for software product lines. *Inform. Softw. Technol.*, 55(3):508–524, 2013.
28. M. Kowal, I. Schaefer, and M. Tribastone. Family-based performance analysis of variant-rich software systems. In S. Gnesi and A. Rensink, editors, *Proceedings of the 17th International Conference on Fundamental Approaches to Software Engineering (FASE'14)*, volume 8411 of *LNCS*, pages 94–108. Springer, 2014.
29. G. N. Rodrigues, V. Alves, V. Nunes, A. Lanna, M. Cordy, P.-Y. Schobbens, A. Molzam Sharifloo, and A. Legay. Modeling and Verification for Probabilistic Properties in Software Product Lines. In *Proceedings of the 16th International Symposium on High-Assurance Systems Engineering (HASE'15)*, pages 173–180. IEEE, 2015.
30. P. Chrszon, C. Dubslaff, S. Klüppelholz, and C. Baier. ProFeat: feature-oriented engineering for family-based probabilistic model checking. *Form. Asp. Comp.*, 30(1):45–75, 2018.
31. M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. A framework for quantitative modeling and analysis of highly (re)configurable systems. *IEEE Trans. Softw. Eng.*, 2018.
32. A. Vandin, M. H. ter Beek, A. Legay, and A. Lluch Lafuente. QFLan: A Tool for the Quantitative Analysis of Highly Reconfigurable Systems. In K. Havelund, J. Peleska, B. Roscoe, and E. P. de Vink, editors, *Proceedings of the 22nd International Symposium on Formal Methods (FM'18)*, volume 10951 of *LNCS*, pages 329–337. Springer, 2018.
33. M. H. ter Beek and A. Legay. Quantitative Variability Modeling and Analysis. In G. Perrouin and D. Weyns, editors, *Proceedings of the 13th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'19)*, pages 13:1–13:2. ACM, 2019.
34. L. Luthmann, T. Gerecht, and M. Lochau. Sampling Strategies for Product Lines with Unbounded Parametric

Real-time Constraints. *Trans. Found. Mastering Chang.*, 2, 2018. In this volume.
35. M. Cordy and A. Legay. Verification and Abstraction of Real-Time Variability-Intensive Systems. *Trans. Found. Mastering Chang.*, 2, 2018. In this volume.
36. L. Herrmann, M. Küttler, T. Stumpf, C. Baier, H. Härtig, and S. Klüppelholz. Configuration of Inter-Process Communication with Probabilistic Model Checking. *Trans. Found. Mastering Chang.*, 2, 2018. In this volume.
37. U. Fahrenberg and A. Legay. Quantitative Properties of Featured Automata. *Trans. Found. Mastering Chang.*, 2, 2018. In this volume.
38. D. Basile. Applying Supervisory Control Synthesis to Priced Featured Automata and Energy Problems. *Trans. Found. Mastering Chang.*, 2, 2018. In this volume.
39. F. Damiani, M. Lienhardt, and L. Paolini. Automatic Refactoring of Delta-Oriented SPLs to Remove-free Form and Replace-free Form. *Trans. Found. Mastering Chang.*, 2, 2018. In this volume.
40. P. Borba, M. B. Cohen, A. Legay, and A. Wąsowski. Analysis, Test and Verification in The Presence of Variability (Dagstuhl Seminar 13091). *Dagstuhl Reports*, 3(2):144–170, 2013.
41. I. Schaefer and M. H. ter Beek. Fomal Methods and Analyses in Software Product Line Engineering. In T. Margaria and B. Steffen, editors, *Proceedings of the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Technologies for Mastering Change (ISoLA'14)*, volume 8802 of *LNCS*, pages 253–256. Springer, 2014.
42. M. H. ter Beek, D. Clarke, and I. Schaefer. Editorial preface for the JLAMP Special Issue on Formal Methods for Software Product Line Engineering. *J. Log. Algebr. Meth. Program.*, 85(1):123–124, 2016.
43. M. H. ter Beek, S. Gnesi, D. Latella, and M. Massink. Towards Automatic Decision Support for Bike-Sharing System Design. In D. Bianculli, R. Calinescu, and B. Rumpe, editors, *Software Engineering and Formal Methods — Revised Selected Papers of the SEFM 2015 Collocated Workshops: ATSE, HOFM, MoKMaSD, and VERY*SCART*, volume 9509 of *LNCS*, pages 266–280. Springer, 2015.
44. A. Nouri, M. Bozga, A. Molnos, A. Legay, and S. Bensalem. ASTROLABE: A Rigorous Approach for System-Level Performance Modeling and Analysis. *ACM Trans. Embed. Comput. Syst.*, 15(2):31:1–31:26, 2016.
45. D. A. van Beek, W. J. Fokkink, D. Hendriks, A. Hofkamp, J. Markovski, J. M. van de Mortel-Fronczak, and M. A. Reniers. CIF 3: Model-Based Engineering of Supervisory Controllers. In E. Ábrahám and K. Havelund, editors, *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *LNCS*, pages 575–580. Springer, 2014.
46. M. H. ter Beek, M. A. Reniers, and E. P. de Vink. Supervisory Controller Synthesis for Product Lines Using CIF 3. In T. Margaria and B. Steffen, editors, *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*, volume 9952 of *LNCS*, pages 856–873. Springer, 2016.
47. C. Chesta, F. Damiani, L. Dobriakova, M. Guernieri, S. Martini, M. Nieke, V. Rodrigues, and S. Schuster. A Toolchain for Delta-Oriented Modeling of Software Product Lines. In T. Margaria and B. Steffen, editors, *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications (ISoLA'16)*, volume 9953 of *LNCS*, pages 497–511. Springer, 2016.