Unifying Software and Product Configuration: A Research Roadmap

Arnaud Hubaux[1] and **Dietmar Jannach**[2] and **Conrad Drescher**[3] and **Leonardo Murta**[4] and **Tomi Männistö**[5] and **Krzysztof Czarnecki**[6] and **Patrick Heymans**[7] and **Tien Nguyen**[8] and **Markus Zanker**[9]

Abstract.
For more than 30 years, knowledge-based product configuration systems have been successfully applied in many industrial domains. Correspondingly, a large number of advanced techniques and algorithms have been developed in academia and industry to support different aspects of configuration reasoning. While traditional research in the field focused on the configuration of physical artefacts, recognition of the business value of customizable software products led to the emergence of software product line engineering. Despite the significant overlap in research interests, the two fields mainly evolved in isolation. Only limited attempts were made at combining the approaches developed in the different fields. In this paper, we first aim to give an overview of commonalities and differences between software product line engineering and product configuration. We then identify opportunities for cross-fertilization between these fields and finally develop a research agenda to combine their respective techniques. Ultimately, this should lead to a unified configuration approach.

1 Introduction

Customizable products are an integral part of most B2B and B2C markets. Mass-customization strategies have been applied to tangible products (e.g., cars and mobile phones) as well as intangible products like software (e.g., operating systems and ERPs) and services (e.g., insurance). To this end, companies use software *configurators* that provide automated support to tailor products to the requirements of specific customers or market segments.

[1] PReCISE Research Center, University of Namur, Belgium, ahu@info.fundp.ac.be
[2] Department of Computer Science, TU Dortmund, Germany, dietmar.jannach@tu-dortmund.de
[3] Computing Laboratory, University of Oxford, UK, Conrad.Drescher@comlab.ox.ac.uk
[4] Fluminense Federal University, Niterói, Brazil, leomurta@ic.uff.br
[5] Aalto University School of Science, Finland, Tomi.Mannisto@aalto.fi
[6] Generative Software Development Lab, University of Waterloo, Canada, kczarnec@gsd.uwaterloo.ca
[7] PReCISE Research Center, University of Namur, Belgium, phe@info.fundp.ac.be
[8] Electrical and Computer Engineering Department, Iowa State University, USA, tien@iastate.edu
[9] Alpen-Adria-Universität Klagenfurt, Austria, markus.zanker@aau.at

Compared to the long history of computer-supported configuration of *products*, research on the configuration of parametrizable *software* is rather new. *Product configuration* (PC) is the umbrella activity of assembling and customizing physical artefacts (e.g. technical equipment, cars or muesli) or services. Historically, PC has been a subfield of *artificial intelligence* (AI), focusing on knowledge representation and reasoning techniques to support configuration. Mostly independent of PC, the field of *software product line engineering* (SPLE) emerged in the *software engineering* community. SPLE deals with the design and implementation of software components that can be adapted and parametrized according to customer requirements and business or technical constraints [47]. As in PC approaches, the goal is to save costs by assembling individualized systems from reusable components [41]. Typical application domains for SPLE include embedded systems, device drivers, and operating systems.

Interestingly, research in these two fields has been carried out so far mostly independently. Except in rare cases (e.g. [28, 5]), researchers in both fields are often unaware of approaches that have been developed in the other community. Further, even though a lot of PC work has focused on configuring technical equipment, such equipment increasingly contains software. At the same time, SPLE increasingly targets software-intensive systems that also include computing and other types of equipment. Based on these observations, our hypothesis is that both the PC and SPLE communities have produced results that are applicable in the other domain.

The remainder of this paper explores further the opportunity for cross-fertilization (Section 2) and proposes a research roadmap (Section 3) to systematically compare the two domains and foster efforts towards unifying both fields. In particular, we hope to find innovative approaches to questions that are largely open in one or the other community such as the reconfiguration of deployed systems, better interactive configuration support (e.g., in case of unsatisfiable requirements), methods for full lifecycle support and the evolution of models and knowledge bases. For this last question, we will also explore how techniques from software configuration management (CM) can be integrated.

2 Motivation

Questions of knowledge acquisition, knowledge representation as well as different types of reasoning support have been in-

Wolfgang Mayer and Patrick Albert, Editors.
Proceedings of the Workshop on Configuration at ECAI 2012 (ConfWS'12),
August 27, 2012, Montpellier, France.

vestigated for many years in PC and SPLE. We highlight some key results in both fields to show their commonalities and differences. These preliminary observations motivate our endeavour to study, compare, and eventually unify research on configuration. We split the introduction of our motivations along five dimensions. For each dimension, we also formulate the research questions whose answers should expose opportunities for cross-fertilization.

2.1 Knowledge acquisition and modelling

Research on PC has used a wide range of knowledge modeling approaches (based, e.g., on UML [22] or description logic [40]), involving different types of logics and constraints. While a few SPLE approaches also used UML to capture aspects of configuration knowledge (e.g. [60, 25]), most results build upon the seminal work on *feature-oriented domain analysis* (FODA) initiated in 1990 by Kang et al. [37], which today is converging with decision models [17]. The cornerstone of FODA are *feature models* (FMs), a graphical notation to capture and express the commonalities and variabilities of a product family. FMs are menu-like hierarchies of mandatory, optional, and alternative features, with cross-hierarchy relationships to express dependencies and incompatibilities. This initial FM notation has been gradually extended to support, for example, multiple instances [18, 44] or the configuration process [29].

Compared to configuration modelling ontologies used in PC (e.g., [22] or [53]), the expressiveness of FMs (even extended ones) appears too limited compared to more complex PC ontologies. Examples of advanced PC problems include connecting components via ports (i.e., inferring complex topologies), finding optimal or at least good configurations, integrating iteratively new components, and distributing knowledge over different agents or business entities [32]. Some work exists in SPLE on component connection and integration (e.g. [5]) and optimization (e.g. [58]). This motivated the creation of more expressive languages (e.g., [8]). PC appears to offer a richer body of work in this area, though.

Opportunities for cross-fertilization: Some authors have already acknowledged the bond between configuration in PC and in SPLE through feature-based configuration. Günter et al. [27] recognize concept hierarchies (similar to FMs) as a fundamental concept in their survey of knowledge-based configuration methods. According to Junker's classification of known configuration problems [34], feature-based configuration falls in the option selection or shopping list problems. To systematically identify such synergies, our research agenda should answer the following questions:

RQ1 *What classes of configuration problems exist?*
RQ2 *How are these problems modelled?*

2.2 Automated reasoning

Generally, with respect to modeling and knowledge representation, the AI-rooted PC community is usually interested in "executable" models that can be directly translated into a representation processable by a reasoning engine. The formal basis of most knowledge modelling languages lays the foundation for advanced configuration reasoning techniques (e.g., checking for consistency of configurations, completing partial configurations, or supporting interactive configuration processes). In contrast, the SPLE community only started recently to develop a formal foundation of FMs (e.g. [9, 49]) and their analyses (e.g. [5, 43, 33, 59]). Based on precise formal problem characterizations, additional automations for SPLE become feasible. An example is the automated analysis of FMs; see [10] for an overview. Furthermore, Benavides et al. [11] propose to translate FMs into a Constraint Satisfaction Problem (CSP) and apply Reiter's model-based diagnosis (MBD) approach to detect problems in the models. Xiong et al. [59] combine MBD and Satisfiability Modulo Theory (SMT) solvers to generate range fixes in software configuration. Mendonca et al. [43] report on experiments with a SAT-encoding of FMs. Finally, Bagheri et al. [7], support hard and soft requirements in the configuration process.

The SPLE community sometimes reinvents techniques which have been developed previously in PC. Encoding configuration problems in some logic or as CSPs has a long history in the AI community [34]. The PC community was also the first to apply SAT solvers to configuration problems [51]. New CSP representations such as Dynamic, Composite or Generative CSPs [45, 48, 54, 32] as well as logics [26, 4] were partially inspired by the challenges observed in PC. This latter pool of techniques addresses the problem of conditionally including multiple instances of a certain component type. The PC community was also first to use MBD for configuration, e.g., for detecting problems in configuration knowledge bases [23]. Regarding soft constraints and preferences, there is abundant literature in constraint programming (e.g. [36, 46]). Finally, binary decision diagrams (BDDs) have also been used for building fast interactive configurators (trading time vs space from a complexity point of view) [3]. That latter approach has been explored in SPLE as well (e.g. [42]), but it turned intractable on large FMs.

Opportunities for cross-fertilization: In contrast to physical components, software components are represented completely as computer artifacts. While physical components need to be specified explicitly in the computer to check cross-component compatibility, software configuration can analyze variability models and the actual configurable artifacts at the same time. This opens up new possibilities for configuration, where the compatibility of components can be checked on the fly during configuration without going back to the design phase and modifying configuration knowledge. To identify overlaps and differences between SPLE and PC, we include the questions:

RQ3 *What automated tasks are supported (e.g., completion, repair, and optimization of configurations)?*
RQ4 *How are these automated tasks implemented?*

2.3 Complexity

The computational complexity is an indicator of the amount of resources needed to solve a given problem. In PC, reasoning is usually achieved by encoding the problem in formalisms such as CSP, SAT, answer set programming or description logics, all of which are being supported by mature reasoners. Both SAT and CSP are well-known to be NP complete [15, 38]. Some extensions of CSPs (dynamic, composite) polynomially reduce to classical CSP [56] whereas the decidability

of Generative CSP has yet to be established. Ground answer set programs are NP complete (Σ_2^p complete in the case of optimization); if programs contain uninstantiated variables we obtain NEXPTIME completeness [50, 19]. Description logics are typically decidable fragments of first order logic [6]; the DLs used in PC range from polynomial over PSPACE complete to undecidable [35, 40, 12]. Let us emphasize that the aforementioned complexity results are only upper bounds: Precise complexity results for classes of configuration problems are still too rare (e.g., [52]).

The same symptom can be observed in SPLE where only a tiny fraction of the papers study complexity aspects (e.g. [49]). While some experimental results exist, e.g., [43], theoretical results are largely missing.

Opportunities for cross-fertilization. Although to a different extent, both PC and SPLE do not fully cover questions related to the complexity of the automated tasks they support for different classes of configuration problems. To follow up on **RQ1** and **RQ3**, we propose these new questions that study their complexities:

RQ5 *What is the complexity of automated tasks for relevant classes of configuration problems?*

RQ6 *What reasoning frameworks can be used to build scalable tools for each class of configuration problems?*

2.4 Life cycle coverage

SPLE suffers from a certain lack of homogeneity across the modelling artefacts used throughout the engineering life cycle. From requirements engineering down to code generation, a myriad of alternative techniques exist which are used and combined differently depending on the application domain and project context. Therefore, there is no standard view on how they should be integrated. As for PC, configuration tasks can range from bill-of-material configuration over cement factory design to t-shirt customization. These tasks call for very different methods and techniques whose applications have been insufficiently studied. Additionally, the creation of feedback loops from productive use back to variability design decisions is rather explored in the more business- or management-centric literature without transfer to PC.

Configurator engineering is a more mature discipline in PC than in SPLE, aiming at the co-design of the configurator and the configurable artefact. According to Hvam et al. [30], the creation, implementation and operation of a configurator is a seven-phase procedure. The first phase identifies the product specification process, used for analyzing customer needs, creating a customized product, and prescribing other related activities, such as purchasing, delivery, servicing, and recycling. The specification process also defines the configuration system that supports the activities composing it. The second phase deals with the definition of the product portfolio. Phases three to six deal with the modelling and implementation of the configuration system. The seventh phase focuses on maintenance.

Finally, the commercial side of configuration is also important. PC has to deal typically with sales, consumer goods, and engineers, wheras SPLE is more geared towards software engineers and other technical experts. Although stakeholder profiles vary from one case to the other, some configuration tasks overlap and techniques could be shared.

Opportunities for cross-fertilization. To better pinpoint overlaps, we split the problem into three questions:

RQ7 *What are the configuration tasks?*

RQ8 *How is a configurable product engineered?*

RQ9 *How is a configurator engineered?*

2.5 Knowledge evolution

The main concern of the software configuration management (CM) discipline is controlling and tracking the evolution of products in response to changes. To do so, it introduces the concept of *versions* that represent instances of products and its parts over time. In CM, there are two main types of versions [14]: *revisions* and *variants*. Revisions are versions that supersede other versions due to bug fixes or addition of new functionalities. Variants are versions intended to coexist through time to satisfy different user or platform needs. Variants are well known in the PC and SPLE fields. However, the management of revisions and the interaction of both is still a weakness of PC and SPLE, especially when the product and its parts evolve frequently. The need to deploy change management techniques in SPLE is recognized [13, 47], and some researchers have started working in this direction (e.g. [2, 57]). Although promising, these results are still incomplete, and need to be extended and consolidated.

Problems related to the evolution of the configuration knowledge and system (e.g. knowledge base, database, and product instances) are also known in PC [21]. PC research has addressed some of these evolution-related aspects in the context of *reconfiguration* problems (e.g. [55, 24]), which consist in changing an already existing or deployed configuration to accommodate new or changed customer requirements or constraints in the knowledge base. Männistö et al. [39] discuss the issue of the evolution of configuration knowledge and instances, proposing a framework to address it. The key idea is to accept the independence of these evolutions, capture the evolution in the models, and then do reconfiguration. Problems similar to reconfiguration have been addressed in the software domain: given a component-based software installation (e.g. Linux, Eclipse), the component dependencies and an (un-)install request, compute a best new installation [1].

Another practical challenge both in PC and SPLE is the constantly growing number of components that can be part of a configuration, be they semi-conductors, switches, or software plug-ins. The corresponding knowledge bases soon become hard to manage because they describe how older components have to be replaced by newer ones or which component versions are compatible.

In CM, some have tried to provide a unified model for software CM and product data management (e.g. [20, 16]). Those models stop at the conceptual level without providing operational solutions, however.

Opportunities for cross-fertilization. Since the 1970s, research in CM focused on change management. Mature solutions are now available and could be applicable to PC. Furthermore, researchers in SPLE and PC could join forces to develop scalable techniques to address the explosion of the number of components and their evolution. Those results could then be contributed back to CM. The resulting questions are:

RQ10 *How can CM techniques be applied to PC and SPLE?*

RQ11 *How to scale up with growing revision knowledge?*

3 Research Roadmap

The first step of our project was the composition of an heterogeneous panel of experts from the different communities and the creation of a knowledge exchange portal. Once populated with our initial results, our intention is to open this shared portal to a wider community and invite collaborations. Our research roadmap has five phases.

1. **Literature survey.** The first phase focuses on a literature survey that should answer **RQ1-9**. The part of the survey related to **RQ1-4** is already underway. The body of knowledge gathered during this initial phase will be the foundation upon which the panel will start its unification endeavour.
2. **Domain understanding and classification.** The second phase paves the road toward a unified theory of configuration. Its objective is to analyze the material collected in the first phase, to classify the application domains, study their differences and commonalities, and to identify the possible bridges between these domains.
3. **Unified theory definition.** The foundation for this theory will be a mathematical model of the classes of configuration problems and their properties. These classes of configuration problems will be characterized by the expressiveness needed to solve the problems. Analogous to the classification of description logics [6], this will enable the study of the theoretical complexity of the associated computational problems. Problem frames [31] could be used as the macro-structure for this classification.
4. **Unified theory operationalization.** Upon this definition, we will build two layers: modelling languages and reasoning techniques. In the modelling layer, we will first sort existing languages according to the configuration problems they address. We will then work toward their unification, possibly by mapping into a common core language. The reasoning layer will gather the configuration tasks (e.g., consistency check, repair, and completion) identified in the previous phases. For each task and a class of configuration problems captured by a language, we will study alternative reasoning techniques and assess their applicability (see next phase). Finally, we will investigate how version management techniques from CM can be adapted to support configuration evolution. Based on these elements, we will answer **RQ6, 10** and **11**.
5. **Unified theory validation.** Configuration can be applied in a myriad of application domains to support many different usage scenarios. Each situation will need its own benchmarks and analyses. The objective of the panel will be to validate the results of the theory operationalization on a few industry-size models. In particular, it will seek to better understand the respective merits and limitations of the various reasoning techniques. Benchmarks will be needed to assess their applicability and tractability on various classes of problems. This pilot validation will be a first step in setting up a framework in which parallel efforts of the community could commence.

This roadmap is not intended to be executed in a waterfall fashion: we expect several iterations through phases 2-5.

4 Conclusion

Configuration, both of software and other types of products, continues to be a timely business strategy as customers consistently strive for affordable tailor-made products. Yet, research in product and software configuration progresses on different and rarely intersecting paths. This paper (1) motivated the need for bridging the current gap between these two domains, and (2) presented a roadmap to build such a bridge and set cross-fertilization in motion.

Our initial observations show that the contribution of the research in product configuration to software product configuration is rather glaring. The inverse is, however, less obvious. As possible contributions, we see methodological aspects as well as modelling techniques. Once laid upon more formal foundations, some of these models could further improve product configuration.

Finally, the *evolution* problem is still under-explored in both software and product configuration. They both have a lot to gain from the techniques promoted in configuration management. Conversely, the formal treatment of configuration problems and automated reasoning could enhance existing work in configuration management.

REFERENCES

[1] P. Abate, R. Di Cosmo, R. Treinen, and S. Zacchiroli, 'Dependency solving: a separate concern in component evolution management', *Systems and Software*, (2012). To appear.

[2] M. Anastasopoulos, D. Muthig, T. H. B. de Oliveira, E. S. Almeida, and S. R. de Lemos Meira, 'Evolving a software product line reuse infrastructure: A configuration management solution', in *Proc. VaMoS'09*, Sevilla, (2009).

[3] H. R. Andersen, T. Hadzic, and D. Pisinger, 'Interactive cost configuration over decision diagrams', *Artificial Intelligence Research (JAIR)*, **37**, 99–139, (2010).

[4] M. Aschinger, C. Drescher, and H. Vollmer, 'LoCo - A Logic for Configuration Problems', in *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI), Montpellier, France*, (2012).

[5] T. Asikainen, T. Soininen, and T. Männistö, 'A Koala-based approach for modelling and deploying configurable software product families', in *Porc PLE'03*, Springer LNCS 3014, pp. 225–249, (2003).

[6] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Cambridge University Press, 2003.

[7] E. Bagheri, T. Di Noia, A. Ragone, and D. Gasevic, 'Configuring software product line feature models based on stakeholders' soft and hard requirements', in *SPLC'10*, pp. 16–31, (2010).

[8] K. Bak, K. Czarnecki, and Andrzej W., 'Feature and meta-models in clafer: Mixed, specialized, and coupled', in *Proc. SLE'10*, pp. 102–122, Eindhoven, The Netherlands, (2010). Springer-Verlag.

[9] D. Batory, 'Feature models, grammars, and propositional formulas', in *SPLC'05*, pp. 7–20, Rennes, France, (2005).

[10] D. Benavides, S. Segura, and A. Ruiz-Cortes, 'Automated analysis of feature models 20 years later: a literature review', *Information Systems*, **35**(6), 615 – 636, (2010).

[11] D. Benavides, P. Trinidad, and Ruiz-Cortez A., 'Automated reasoning on feature models', in *Proc. CAiSE'05*, pp. 491–503, Porto, Portugal, (2005). Springer.

[12] M. Buchheit, R. Klein, and W. Nutt, 'Constructive Problem Solving: A Model Construction Approach towards Configuration', Technical Report TM-95-01, DFKI, (1995).

[13] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2001.

Wolfgang Mayer and Patrick Albert, Editors.
Proceedings of the Workshop on Configuration at ECAI 2012 (ConfWS'12),
August 27, 2012, Montpellier, France.

[14] R. Conradi and B. Westfechtel, 'Version models for software configuration management', *ACM Computing Surveys*, **30**, 232–282, (June 1998).

[15] S. A. Cook, 'The complexity of theorem-proving procedures', in *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, (1971).

[16] I. Crnkovic, U. Asklund, and A. P. Dahlqvist, *Implementing and integrating product data management and software configuration management*, Artech House Publishers, 2003.

[17] K. Czarnecki, P. Grunbacher, R. Rabiser, K. Schmid, and A. Wasowski, 'Cool features and tough decisions: Two decades of variability modeling', in *Proc. VaMoS'12*, Leipzig, Germany, (2012). ACM Press.

[18] K. Czarnecki, S. Helsen, and U. W. Eisenecker, 'Staged configuration through specialization and multi-level configuration of feature models', *Software Process: Improvement and Practice*, **10**(2), 143–169, (2005).

[19] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, 'Complexity and expressive power of logic programming', *ACM Computing Surveys*, **33**(3), 374–425, (2001).

[20] J. Estublier, J.-M. Favre, and P. Morat, 'Toward scm / pdm integration?', in *Proc. SCM'98*, pp. 75–94, London, UK, UK, (1998). Springer-Verlag.

[21] A. Falkner and A. Haselböck, 'Challenges of knowledge evolution in practice', in *ECAI 2010 IKBET Wks*, pp. 1–5, (2010).

[22] A. Felfernig, G. Friedrich, and D. Jannach, 'UML as domain specific language for the construction of knowledge-based configuration systems', in *Proc. SEKE 99*, pp. 337–345, Kaiserslautern, Germany, (1999).

[23] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, 'Consistency-based diagnosis of configuration knowledge bases', *Artificial Intelligence*, **152**, 213–234, (2004).

[24] G. Friedrich, A. Ryabokon, A. Falkner, A. Haselböck, G. Schenner, and H. Schreiner, '(Re)configuration based on model generation', in *Proc. LoCoCo Workshop*, EPTCS, pp. 26–35, (2011).

[25] H. Gomaa and M. Eonsuk Shin, 'Multiple-view modelling and meta-modelling of software product lines', *IET Software*, **2**(2), 94–122, (2008).

[26] G. Gottlob, G. Greco, and T. Mancini, 'Conditional Constraint Satisfaction: Logical Foundations and Complexity', in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India*, (2007).

[27] A. Günter and C. Kühn, 'Knowledge-based configuration: Survey and future directions', in *Proc. XPS'99*, pp. 47–66, London, (1999).

[28] L. Hotz, K. Wolter, and T. Krebs, *Configuration in Industrial Product Families: The ConIPF Methodology*, IOS Press, Inc., 2006.

[29] A. Hubaux, A. Classen, and P. Heymans, 'Formal modelling of feature configuration workflow', in *Proc. SPLC'09*, pp. 221–230, San Francisco, (2009).

[30] L. Hvam, N. Henrik Mortensen, and J. Riis, *Product Customization*, Springer-Verlag Berlin Heidelberg, 2008.

[31] M. Jackson, *Problem frames: analyzing and structuring software development problems*, Addison-Wesley, 2001.

[32] D. Jannach and M. Zanker, 'Modeling and solving distributed configuration problems: A CSP-based approach', *IEEE TKDE*, http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.236.

[33] M. Janota, 'Do SAT solvers make good configurators?', in *Proc. ASPL'08*, pp. 191–195, Limerick, Ireland, (2008).

[34] U. Junker, *Handbook of Constraint Programming*, chapter Configuration, 837–873, Elsevier, 2006.

[35] U. Junker and D. Mailharro, 'The logic of ILOG (J)Configurator: Combining constraint programming with a description logic', in *Proc. IJCAI-03 Configuration Workshop*, (2003).

[36] U. Junker and D. Mailharro, 'Preference programming: Advanced problem solving for configuration', *AIEDAM*, **17**(1), 13–29, (2003).

[37] K. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, 'Feature-Oriented Domain Analysis (FODA) Feasibility Study', Technical report, SEI, CMU, (1990).

[38] A. K. Mackworth, 'Consistency in networks of relations', *Artificial Intelligence*, **8**, 99–118, (1977).

[39] T. Männistö, *A Conceptual Modelling Approach to Product Families and Their Evolution*, Ph.D. dissertation, Helsinki University of Technology, Finland, 2000.

[40] D. L. McGuinness and J. R. Wright, 'Conceptual modelling for configuration: A description logic-based approach', *Artif. Intell. Eng. Des. Anal. Manuf.*, **12**(4), 333–344, (1998).

[41] M. D. McIlroy, 'Mass produced software components', in *Proc. Software Engineering Concepts and Techniques*, pp. 138–150. NATO Science Committee, (1968).

[42] M. Mendonça, *Efficient Reasoning Techniques for Large Scale Feature Models*, Ph.D. dissertation, U Waterloo, 2009.

[43] M. Mendonça, A. Wasowski, and K. Czarnecki, 'SAT-based analysis of feature models is easy', in *Proc. SPLC'09*, pp. 231–240, San Francisco, (2009). Carnegie Mellon University.

[44] R. Michel, A. Classen, A. Hubaux, and Q. Boucher, 'A formal semantics for feature cardinalities in feature diagrams', in *Proc. Wks. VaMoS'11*, pp. 82–89, Namur, BE, (2011).

[45] S. Mittal and B. Falkenhainer, 'Dynamic constraint satisfaction problems', in *AAAI'90*, pp. 25–32, Boston, (1990).

[46] M. Pasanen, *Warnings and Pre-selection Packages in a Weight Constraint Rule Based Configurator*, Master's thesis, Helsinki University of Technology, Department of Computer Science and Engineering, Finland, 2003.

[47] K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer, 2005.

[48] D. Sabin and E. C. Freuder, 'Configuration as Composite Constraint Satisfaction', in *Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop (AIMRP), Albuquerque, New Mexico*, (1996).

[49] P-Y. Schobbens, P. Heymans, J-C. Trigaux, and Yves Bontemps, 'Generic semantics of feature diagrams', *Comput. Netw.*, **51**(2), 456–479, (2007).

[50] P. Simons, I. Niemelä, and T. Soininen, 'Extending and implementing the stable model semantics', *Artificial Intelligence*, **138**(1–2), 181–234, (2002).

[51] C. Sinz, A. Kaiser, and W. Küchlin, 'SAT-based consistency checking of automotive electronic product data', in *ECAI Workshop*, (2000).

[52] T. Soininen, *An Approach to Knowledge Representation and Reasoning for Product Configuration Tasks*, Ph.D. dissertation, 2000.

[53] T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen, 'Towards a general ontology of configuration', *AIEDAM*, **12**, 357–372, (1998).

[54] M. Stumptner, A. Haselböck, and G. Friedrich, 'Generative Constraint-based Configuration of Large Technical Systems', *AI EDAM*, **12**(4), 307–320, (1998).

[55] M. Stumptner and F. Wotawa, 'Model-based reconfiguration', in *Proc. AID'98*, pp. 45–64, (1998).

[56] E. Thorstensen, 'Capturing Configuration', in *Doctoral Program at the 16th International Conference on Principles and Practice of Constraint Programming (CP), St. Andrews, Scotland*, (2010).

[57] T. Thum, D. Batory, and C. Kastner, 'Reasoning about edits to feature models', in *Proc. ICSE'09*, pp. 254–264, Washington, DC, USA, (2009). IEEE Computer Society.

[58] T. T. Tun, Q. Boucher, A. Classen, A. Hubaux, and P. Heymans, 'Relating requirements and feature configurations: A systematic approach', in *Proc. SPLC'09*, pp. 201–210, San Francisco, CA, USA, (2009). ACM Press.

[59] Y. Xiong, A. Hubaux, and K. Czarnecki, 'Generating range fixes for software configuration', in *Proc. ICSE'12*, Zurich, Switzerland, (2012). IEEE Computer Society.

[60] T. Ziadi, L. Helouet, and J.-M. Jezequel, 'Towards a UML profile for software product lines', in *Software Product-Family Engineering*, Springer LNCS 3014, 129–139, (2004).

Wolfgang Mayer and Patrick Albert, Editors.
Proceedings of the Workshop on Configuration at ECAI 2012 (ConfWS'12),
August 27, 2012, Montpellier, France.