Empirical Software Engineering (2021) 26: 72
https://doi.org/10.1007/s10664-021-09964-6

Foreword to the Special Issue on Configurable Systems

Laurence Duchien[1] · **Paul Grünbacher**[2] · **Thomas Thüm**[3]

Published online: 19 May 2021

Welcome to this special issue that includes empirical studies on configurable systems. This special issue in the Empirical Software Engineering journal is intended to provide the systems and software product lines community with a valuable collection of high-quality research articles that explore configurable systems with empirical studies. Particular attention was paid to research and industrial work carrying out experiments on configuration steps in the life cycle of system and software product lines. A configurable system is an artifact composed from instances of a set of predefined component types that can be composed and parameterized. The configuration step requires knowledge representation formalisms to capture variety and complexity of configurable products, but also acquisition methods and efficient reasoning algorithms for supporting solution search, to represent, and integrate user settings, personalization, and optimization. The configuration ends with the deployment and launches execution steps. This configuration can also change over time, because of a change of context. This is called reconfiguration.

The majority of articles extend research presented at SPLC, the 23rd International Systems and Software Product Lines Conference. The conference was held from September 9 to 13, 2019 in Paris, France (Berger et al. 2019).

We received seventeen articles for this special issue. The call was open, but the SPLC 2019 authors were encouraged to prepare a revised and substantially extended version, and to consider as possible extensions additional practical applications determined by case studies or experiences, empirical validations, systematic comparisons with other approaches, or

✉ Laurence Duchien
laurence.duchien@univ-lille.fr

Paul Grünbacher
paul.gruenbacher@jku.at

Thomas Thüm
thomas.thuem@uni-ulm.de

[1] Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

[2] Institute of Software Systems Engineering, Johannes Kepler University Linz, Linz, Austria

[3] Institute of Software Engineering and Programming Languages, University of Ulm, Ulm, Germany

sound theoretical foundations. The submitted manuscripts were each peer reviewed by three reviewers. Finally, ten papers were accepted for inclusion in this special issue and six of them were extended versions of the SPLC 2019 research papers.

Summary of the Papers In their paper titled “Automated Test Reuse for Highly Configurable Software”, S. Fischer, G. K. Michelon, R. Ramler, L. Linsbauer, and A. Egyed focus their research on the testing of configurable software. They report on an experiment about automatically reusing existing tests in configurable systems and show that they could directly reuse some tests for different configurations. Their automatically generated test variants generally yielded better results.

A. Vescan, A. Pintea, L. Linsbauer, and A. Egyed demonstrate that the genetic programming pipeline is a good fit for feature models and has been shown to produce good reverse engineering results. In their paper “Genetic Programming for Feature Model Synthesis: A Replication Study”, they replicate the results of such an existing approach with a larger set of feature models and investigate the effects of various genetic programming parameters and operators on the results.

C. D. Nascimento Damasceno, M. R. Mousavi, and A. Simao study family model learning that incorporates principles of feature model analysis and comparison of state-based models to efficiently merge product models into fresh family models and include new product behavior into existing models. In their paper “Learning by Sampling: Evaluating T-wise Sampling Criteria for Learning Family Models”, the authors argue that the exhaustive analysis of product families is usually infeasible due to the potentially exponential number of valid configurations. However, they extend their analysis upon FFSMDiff, an automated algorithm to learn family models, and report their experience on learning family models by sampling configurations using 105 products of six product lines expressed in terms of Mealy machines.

In their paper “Software Product-Line Evaluation in the Large”, R. Lindohf, E. Herzog, J. Krüger, and T. Berger study the design and implementation of product lines, which requires substantial upfront investment. Several measurement methods have been proposed in the literature, with the most prominent one being the Family Evaluation Framework (FEF). The authors present an experience report of applying the FEF to nine medium to large-scale product lines in the avionics domain. They discuss how they tailored and executed the FEF, together with the relevant adaptations and extensions they needed to perform. Specifically, they elicited the data for the FEF assessment with 27 interviews over a period of 11 months.

B. Ramos-Gutiérrez, A. J. Varela-Vaca, J. A. Galindo, M. T. Gómez-López, and D. Benavides present COnfiguration workfLOw proceSS (COLOSSI), an automated technique that can assist to determine the configuration workflow. This one better fits the configuration logs generated by user activities given a set of logs of previous configurations and a variability model. In their paper “Discovering Configuration Workflows From Existing Logs Using Process Mining”, the authors validate it on three different scenarios: an ERP configuration, a Smart Farming, and a Computer Configuration.

M. H. ter Beek, F. Damiani, M. Lienhardt, F. Mazzanti, and L. Paolini propose and empirically evaluate the efficiency of static analyses for featured transition systems. Transition systems have been extended several years ago to explicitly model features by means of featured transition systems. Similar to dead features in feature models and dead code in programs, such transition systems may contain anomalies. The empirical investigation demonstrates that dead locks, false-optional transitions, and dead transitions can be efficiently detected.

M. Cashman, J. Firestone, M. B. Cohen, T. Thianniwet, and W. Niu present their empirical investigation of organic software product lines. In their work, they apply product-line techniques to the field of synthetic biology. With synthetic biology, organisms are programmed to perform new functions by synthesizing their DNA. The authors reverse engineer the variability of an open-source repository with 45k reusable DNA parts. In their study, they evaluate the applicability of product-line concepts and give recommendations for other emerging application domains of software product lines.

P. Temple, G. Perrouin, M. Acher, B. Biggio, J.-M. Jézéquel, and F. Roli empirically assess the quality of adversarial configurations for software product lines. That is, the authors apply adversarial machine learning to product-line configurations to overcome the problem that exhaustively generating and testing all configurations is typically not feasible. While their empirical studies were applied to two product lines, a major technical contribution is how to encode the validity of configurations into the generation of adversarial configurations.

In the paper "Lightweight, Semi-Automatic Variability Extraction: A Case Study on Scientific Computing", A. Grebhahn, C. Kaltenecker, C. Engwer, N. Siegmund, and S. Apel investigate variability in an interesting domain: researchers in scientific computing frequently follow a clone-and-own approach when using frameworks to simulate physical, chemical, and biological processes. The paper proposes an API-based analysis approach that recommends appropriate artifacts as possible alternatives for replacing given artifacts. Such alternative artifacts can speed up performance of the simulation or make it amenable to other use cases, without modifying the overall structure of the simulation. The authors evaluate the practicality of their approach in a case study on the DUNE numerics framework and two simulations from the realm of physical simulations.

E. Kuiter, S. Krieter, J. Krüger, G. Saake, and T. Leich present "variED: An Editor for Collaborative, Real-Time Feature Modeling", which supports the scenario of collaborative and real-time editing of feature models. The article provides formal foundations of collaborative, real-time feature modeling and conflict resolution, including proofs of the formalization and an implementation of a tool prototype. The article further reports results of an empirical evaluation assessing the prototype's feasibility and showing that it is was perceived as helpful and valuable by users.

Acknowledgements We thank the authors for their excellent manuscripts. We are deeply grateful to the reviewers of this special section for their time and constructive feedback, which helped to shape the articles. We are also thankful to the Empirical Software Engineering journal and the Editors-in-Chief, Robert Feldt and Thomas Zimmermann, for their support throughout the process of preparing this special issue.

References

Berger T, Collet P, Duchien L, Fogdal T, Heymans P, Kehrer T, Martinez J, Mazo R, Montalvillo L, Salinesi C, Tërnava X, Thüm T, Ziadi T (eds) (2019) Proceedings of the 23rd international systems and software product line conference, SPLC 2019, volume a, paris, france, september 9-13, 2019. ACM, New York. https://dl.acm.org/citation.cfm?id=3336294