The Journal of Systems & Software 199 (2023) 111622

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Editorial

Systems and software product lines of the future

1. Introduction

For well over two decades now, researchers and practitioners have pushed forward the modelling, analysis and development of systems and software product lines rather than individual systems or software products. The distinguishing feature of the underlying engineering approach is to capture the commonalities and variability among individual variants explicitly and to manage variability-intensive systems and software throughout their entire life cycle, from requirements elicitation to runtime operation and maintenance.

The current Horizon Europe Programme targets the development of advanced technologies and methodologies for reliable and efficient distributed software applications in cloud-to-edge-to-IoT systems. Also the Software Product Line (SPL) community needs to keep up with the pace at which such applications evolve and provide adequate support for the software-intensive and variability-rich systems of the future, for example in challenging domains such as Smart Cities, Smart Health and Industry 4.0. This special issue offers a step forward to such systems and software product lines of the future.

2. Overview of the special issue

The theme of this special issue is "Systems and Software Product Lines of the Future". We solicited the submission of high-quality papers describing original and innovative research with a sound scientific or technological basis and validation as well as submissions of extended papers from the 25th International Systems and Software Product Line Conference (SPLC 2021). The call for papers attracted 15 submissions covering diverse relevant topics. Each submitted article was carefully evaluated by at least three experts in the field. After a rigorous peer review process, 11 high-quality research papers have been selected for the special issue.

In the paper titled "Variability modules", Damiani et al. introduce variability modules (VMs), a novel approach to implement Multi SPLs consisting of Java-like delta-oriented programs, in which different, possibly interdependent variants of the same SPL can co-exist and interoperate. Central to the design of VMs are simplicity and usability. The authors formalised the syntax and semantics of VMs as an extension of the ABS language called ABS-VM. This allowed them to implement an ABS-VM compiler as a front-end to the ABS tool chain, together with family-based checks for type uniformity and such. They evaluated the VM concept and their implementation both quantitatively and qualitatively by case studies that were partly taken from industrial code used in production.

In the paper titled "The language mutation problem: Leveraging language product lines for mutation testing of interpreters", Cazzola and Favalli present a sourceless approach to language implementation testing by leveraging on the properties of language product lines, language workbenches, separate compilations, and mutants generation. In this approach, the base language is taken as a black-box and mutated by means of mutation operators performed at language feature level to create a family of mutants of the base language. Each variant of the mutant family is created at runtime, without any access to the source code and without performing any additional compilation. The authors report an experiment that shows the applicability of the approach to a Javascript implementation written in the Neverlang language workbench by exploiting the Sputnik conformance test suite.

In the paper titled "Spectrum-based feature localization for families of systems", Michelon et al. present a novel feature localisation approach for families of systems by adapting the Spectrum-based fault localisation (SBFL) technique for single software systems known from the software debugging field, supporting both dynamic and static feature localisation. The latter does not use execution traces but the implementation elements and, typically, *n*-way comparison-based techniques for these elements. The proposed feature localisation approaches are evaluated using the consolidated ArgoUML SPL feature localisation benchmark. The results show that SBFL increases precision in families of systems even with only a few variants and that SBFL can be used in a static way to compare variants, providing competitive results with respect to the state of the art.

In the paper titled "Feature models to boost the vulnerability management process", Varela-Vaca et al. present AMADEUS-Exploit, a framework for the automatic generation of feature models from vulnerability and exploit repositories, which enables a vulnerability management process supported by automatic query and reasoning mechanisms. It provides vulnerability management experts with support for the identification of potentially vulnerable software and system configurations or to prioritise vulnerabilities to be assessed. The framework is compared with other vulnerability management tools, and it is evaluated on a synthetic scenario with 4000 vulnerabilities and 700 exploits as

https://doi.org/10.1016/j.jss.2023.111622
0164-1212/© 2023 Elsevier Inc. All rights reserved.

well as on a real-world scenario demonstrating the usability of reasoning operations to determine potential vulnerabilities.

In the paper titled "VIBE: Looking for Variability In amBiguous rEquirements", Fantechi et al. define VIBE, a tool-supported process for the identification of variability-related aspects in requirements documentation in natural language. The first step in the process concerns the application of a Natural Language Processing (NLP) tool for ambiguity detection, customised to identify variability. To this aim, the authors compared a set of two academic and four commercial NLP tools to understand which classes of ambiguity are detected by the different tools and whether they offer suitable customisation features, after which they used one of them, namely the Quality Analyser for Requirements Specifications (QuARS), to validate the adequacy of VIBE to identify variation points against a set of six large third-party requirements documents from varying domains.

In the paper titled "Automating Feature Model maintainability evaluation using machine learning techniques", Silva et al. evaluate the scalability of using white-box machine-learning (ML) algorithms to create ML models capable of classifying feature model maintainability from a set of measures, as well as that of a mechanisms that uses the results of the assessment made by the ML model to suggest feature model refactorings to improve the maintainability of the feature models. Moreover, the authors automated both the feature model assessment and the refactoring suggestion mechanism by implementing it in the DyMMer tool. This provides domain engineers with a feasible strategy to assess the maintainability and take corrective actions while editing the feature model.

In the paper titled "A Monte Carlo Tree Search Conceptual Framework for Feature Model Analyses", whose conference version was awarded the SPLC 2021 best paper award, Horcas et al. propose a conceptual framework that enables various automated analysis techniques for feature models by using Monte Carlo tree search methods rather than SAT solving or constraint programming. These methods incorporate probability into the analysis techniques as a means to solve problems that are typically difficult to tackle using deterministic approaches due to their large search space. Moreover, these methods can provide some decision-making capacity requiring very little domain-specific knowledge. The authors provide a Python implementation of the framework that shows the feasibility of their proposal, identify 11 lessons learned, and list several open challenges concerning the use of Monte Carlo methods in the software product line domain.

In the paper titled "Interaction detection in configurable systems — A formal approach featuring roles", Chrszon et al. present a formal approach to model and analyse role-oriented systems, inspired by the Compartment Role Object Model (CROM) that unifies most of the well-established views on roles from the literature and provides a graphical notation similar to UML class diagrams. The authors propose role-based automata (RBAs) to model the operational behaviour and introduce a light-weight modelling language to describe RBAs. They implemented an automated translation from their modelling language into the input language of the probabilistic model checker Prism, which allowed them to perform formal analysis to detect and quantify hierarchical interactions and active interplays with Prism.

In the paper titled "Configuring Mission-Specific Behavior in a Product Line of Collaborating Small Unmanned Aerial System", Al Islam et al. present Drone Response, an SPL for rapidly configuring and deploying a multi-role mission of autonomous small Unmanned Aerial Systems (sUAS), whilst guaranteeing a set of safety properties related to the sequencing of tasks within the mission. Individual sUAS behaviour is governed by an onboard state machine, combined with coordination handlers which are configured dynamically within seconds of launch and determine the sUAS' behaviour, decisions, and interactions with other sUAS, as well as human operators. To ensure the absence of common types of configuration failures and to promote safe deployments, the authors check vital properties of the dynamically generated sUAS specifications and coordination handlers before sUAS are assigned their missions. To validate their approach, the authors conducted end-to-end tests to demonstrate that the configuration process produces valid executable missions, and fault-based testing to check that valid specifications are accepted and invalid ones rejected.

In the paper titled "Test Scenario Generation for Feature-Based Context-Oriented Software Systems", Martou et al. explore the issue of testing feature-based context-oriented programming systems, which requires dedicated tool support due to their high runtime adaptivity. They do so by building upon a pairwise combinatorial interaction testing approach stemming from the SPL domain. Concretely, they implement an algorithm to automatically generate a small set of relevant test scenarios, which is ordered to minimise the number of context activations between tests, and study how the generated scenarios can be enhanced incrementally when the software evolves. They also study how useful the proposed testing approach is in practice by using it on a concrete case study.

In the paper titled "Colla-Config: a stakeholders preferences-based approach for product lines collaborative configuration", Ben Sassi et al. propose a collaborative product line configuration approach, called Colla-Config, that provides a preference-based conflict resolution method within a free-order configuration process according to which each stakeholder expresses her preferences through a set of predefined substitution rules and freely makes her configuration decisions towards the desired product, without being constrained by the configuration decisions made by other stakeholders. Based on the expressed preferences, the suitable minimal set of conflicting choices is computed and removed from the configuration. To evaluate their approach, the authors developed a software tool that implements the Colla-Config approach and used it to conduct a preliminary usability test, which provided promising results concerning the feasibility of the approach and the tool's ability to support collaborative SPL configuration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank all the authors for their high-quality contributions to the special issue. In addition, our appreciation is due to all the reviewers for their great effort and constructive comments. We are also grateful to the editors-in-chief (Paris Avgeriou and David Shepherd), the special issue managing guest editors (Laurence Duchien and Raffaela Mirandola), and the journal manager of JSS (Ankit Kapoor) for their support throughout the process of preparing the special issue.

Maurice ter Beek is a senior researcher at ISTI–CNR, Pisa, Italy, where he heads the Formal Methods and Tools lab. He obtained his BSc, MSc and PhD in Computer Science at Leiden University, The Netherlands. He works on formal methods and model-checking tools for the specification and verification of safety-critical software systems and communication protocols, focusing in particular on applications in service-oriented computing, software product line engineering and railway systems. He has co-authored over 150 peer-reviewed

papers, co-edited over 30 proceedings and special issues of journals, and serves on the editorial boards of the journals Formal Aspects of Computing: Applicable Formal Methods, International Journal on Software Tools for Technology Transfer, Journal of Logical and Algebraic Methods in Programming, PeerJ Computer Science, Science of Computer Programming and ERCIM News.

Ina Schaefer is a full professor of Software Engineering at Karlsruhe Institute of Technology (KIT), Germany. She received a PhD from Technische Universität Kaiserslautern in 2008. She was professor for Software Engineering and Automotive Informatics at TU Braunschweig from 2012–2022. Her main research interests are in the intersection of software engineering and formal methods, particularly focusing on correctness-by-construction development and quality assurance for software-intensive and variant-rich systems.

Maurice H. ter Beek *
ISTI–CNR, Pisa, Italy
E-mail address: maurice.terbeek@isti.cnr.it.

Ina Schaefer
KIT, Karlsruhe, Germany
E-mail address: ina.schaefer@kit.edu.

Available online 31 January 2023

* Corresponding editor.