EXtracting Product Lines from vAriaNTs (EXPLANT)

Jacob Krüger
Otto-von-Guericke University, Germany

Sebastian Krieter
Harz Unviersity & Otto-von-Guericke University, Germany

Gunter Saake
Otto-von-Guericke University, Germany

Thomas Leich
Harz University, Germany

ABSTRACT

The project EXtracting Product Lines from vAriaNTs (EXPLANT) funded by the German Research Foundation (DFG) is currently in its second phase. In this project, we are concerned with the stepwise migration of cloned variants into a software product line (i.e., the extractive approach of adopting systematic software reuse). While the extractive approach is the most common one in practice, many of its characteristics (e.g., processes, costs, best practices) are still unclear and tool support is limited (e.g., for feature location, refactoring, quality assurance). Within this extended abstract, we report on a selection of results we achieved in EXPLANT so far, and highlight our goals as well as opportunities for research collaborations in the second phase.

CCS CONCEPTS

• **Software and its engineering → Software product lines**; **Software reverse engineering**.

KEYWORDS

Software product line, Reverse engineering, Software evolution, Extractive approach, Clone and own

ACM Reference Format:
Jacob Krüger, Sebastian Krieter, Gunter Saake, and Thomas Leich. 2020. EXtracting Product Lines from vAriaNTs (EXPLANT). In *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS '20), February 5–7, 2020, Magdeburg, Germany.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377024.3377046

1 INTRODUCTION

Software product lines enable developers to systematically manage and reuse software artifacts, which contribute to software *features*, based on an integrated platform [1, 25]. Various techniques (e.g., feature modeling [3, 24]) and tools (e.g., FeatureIDE [23]) have emerged that enable developers to cope with the complexity of software product lines. For instance, feature modeling enables developers to define a set of features and their interdependencies in a feature model. Based on this model, a customer can configure a customized software system that fulfills their specific requirements. In an automated process, developers can then derive a software product from the software platform and the given configuration.

Despite the benefits of software product lines, such as reduced development and maintenance costs [1, 25], they are rarely applied from scratch. Instead, practitioners often reuse software in an ad hoc style by cloning an existing system and adapting it to new requirements (*clone & own*) [4]. With an increasing number of clones (resulting in a *variant-rich system*), this reuse strategy becomes expensive and error prone, mainly because changes must be propagated between the individual clones [20, 26]. As a result, many organizations decide to migrate towards a software product line by extracting it from the cloned systems [7], which has become the most common adoption strategy in practice [2, 9].

While being a common adoption strategy, the extractive approach still lacks advanced tooling and an empirical understanding of its characteristics. To tackle these issues, we investigate the automated, incremental integration of cloned systems into a software product line based on code-clone detection and refactorings. We argued that an extraction into a composition-based—in contrast to an annotation-based—software product line makes it immediately usable in practice, improves maintainability due to physically separated features, and guarantees to preserve semantics. However, we deviated from our initial plan, due to our findings contradicting common believes in research and the benefits we argued to achieve. In particular, our results show that an automatic extraction based on code-clone detection does not result in meaningful features [5, 17]. Also, surprisingly, annotations do not seem to be as problematic as often assumed in research [6, 13]. Consequently, we adapted our plan during the first and for the second phase of EXPLANT. Within the remaining paper, we summarize our key findings (cf. Section 2) and our research goals for the remaining project (cf. Section 3).

2 PROGRESS & CONTRIBUTIONS

In the following, we exemplify some of our main findings.

Feature Location. Program comprehension and especially feature location are core problems of evolving any software system. We aimed to understand how developers handle these problems based on empirical and meta studies. For example, we conducted an experiment on developers' memory, which showed that especially authoring and repeatedly editing code facilitate remembering [19]. Moreover, we performed a literature review on manual feature location that highlights the importance of involving, supporting, and understanding developers' activities [12].

Automated Extraction. One of our goals was to automate the extraction of a software product line from cloned systems. For this purpose, we proposed a set of refactorings that employ code-clone detection [5]. However, neither existing feature-location techniques nor code-clone detection can fully automate the extraction process. We conducted an additional case study that highlighted the differences between automated and manual feature location, indicating that full automation does not result in reasonable features [17].

VaMoS '20, February 5–7, 2020, Magdeburg, Germany
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7501-6/20/02.
https://doi.org/10.1145/3377024.3377046

Separation of Features. Researchers often argue that annotations are a poor design choice for managing variability (e.g., `#ifdef`-hell [22]). In contrast, we found that they do not impact change proneness [6], are suitable for tracing features [13], and are preferred by developers to avoid scattering of fine-grained features [8, 15]. Due to these insights, we adapted our work plan to focus less on composition-based variability mechanisms.

Incremental Integration. We conducted case studies on analyzing legacy systems [16], extracting software product lines [17, 20], and combining variability mechanisms [18]. A particular result of these studies are defined processes for developing features and migrating source code. Our findings indicate that the initially intended horizontal extraction (extracting the most common parts into features) is not suitable, as clones and differences do not necessarily represent reasonable features. Consequently, we argued in favor of, and employed, a vertical extraction (incrementally integrating new variants into a common code base) [17].

3 GOALS FOR THE SECOND PHASE

In this section, we exemplify our new research goals.

We aim to tackle the automated migration and evolution of test suites to assure the quality and to facilitate testing of a software product line [10]. Besides the actual source code, most software projects comprise various additional artifacts, for example, models, documentation, and test suites. To facilitate the extraction and evolution of a software product line, automated analyses and syntheses of such artifacts are highly valuable, too. While some techniques have been proposed, for example, natural language processing of requirements documentation [21], they are often in an initial state and require more research to be applicable in practice. Particular problems that occur for the extraction are the matching of artifacts to each other and the mapping of artifacts during the migration process.

We aim to conduct and synthesize empirical studies, including analyses of version control systems (VCSs), to measure activities and assess their impact on costs [11]. It is important to understand the costs related to extracting a software product line [14]. Only with reliable empirical insights, we can initiate collaborations and show organizations that an extraction is valuable, and not an unprofitable investment. Currently, we started with empirical studies and collaborations with practitioners to gain insights into these costs, but further analyses are needed.

We aim to adapt analysis techniques for VCSs, including feature location, variability analysis, and architecture synthesis. VCSs (e.g., Git) and software-hosting platforms (e.g., GitHub) are becoming the major tools to implement and manage cloned systems. Consequently, further analyzing such systems and adapting our current techniques to their specifics is an important research goal. In contrast to completely separated clones, clones of VCSs are often lightly connected (e.g., through linked forks) and comprise additional information sources (e.g., commit messages, timestamps, diffs, and pull requests) [16].

4 CONCLUSION

The extraction and evolution of variant-rich systems, and especially software product lines, are challenging tasks. Consequently, they provide various research opportunities of which we address some within the EXPLANT project. In this extended abstract, we exemplified core contributions that we achieved in the first phase of this project, and highlighted some of our new research goals for the second phase.

Acknowledgments. This research is supported by the DFG project EXPLANT (LE 3382/2-3, SA 465/49-3). We thank all our past and current collaborators for their contributions to this research project.

REFERENCES

[1] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. 2016. *Feature-Oriented Software Product Lines.* Springer.
[2] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A Survey of Variability Modeling in Industrial Practice. In *VaMoS.*
[3] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *VaMoS.*
[4] Yael Dubinsky, Julia Rubin, Thorsten Berger, Slawomir Duszynski, Martin Becker, and Krzysztof Czarnecki. 2013. An Exploratory Study of Cloning in Industrial Software Product Lines. In *CSMR.*
[5] Wolfram Fenske, Jens Meinicke, Sandro Schulze, Steffen Schulze, and Gunter Saake. 2017. Variant-Preserving Refactorings for Migrating Cloned Products to a Product Line. In *SANER.*
[6] Wolfram Fenske, Sandro Schulze, and Gunter Saake. 2017. How Preprocessor Annotations (do not) Affect Maintainability: A Case Study on Change-Proneness. In *GPCE.*
[7] Charles Krueger. 2001. Easing the Transition to Software Mass Customization. In *PFE.*
[8] Jacob Krüger. 2018. Separation of Concerns: Experiences of the Crowd. In *SAC.*
[9] Jacob Krüger. 2019. Are You Talking about Software Product Lines? An Analysis of Developer Communities. In *VaMoS.*
[10] Jacob Krüger, Mustafa Al-Hajjaji, Sandro Schulze, Gunter Saake, and Thomas Leich. 2018. Towards Automated Test Refactoring for Software Product Lines. In *SPLC.*
[11] Jacob Krüger and Thorsten Berger. 2020. Activities and Costs of Re-Engineering Cloned Variants into an Integrated Platform. In *VaMoS.* Accepted.
[12] Jacob Krüger, Thorsten Berger, and Thomas Leich. 2019. Features and How to Find Them: A Survey of Manual Feature Location. In *SEVIS.*
[13] Jacob Krüger, Gül Çalikli, Thorsten Berger, Thomas Leich, and Gunter Saake. 2019. Effects of Explicit Feature Traceability on Program Comprehension. In *ESEC/FSE.*
[14] Jacob Krüger, Wolfram Fenske, Jens Meinicke, Thomas Leich, and Gunter Saake. 2016. Extracting Software Product Lines: A Cost Estimation Perspective. In *SPLC.*
[15] Jacob Krüger, Kai Ludwig, Bernhard Zimmermann, and Thomas Leich. 2018. Physical Separation of Features: A Survey with CPP Developers. In *SAC.*
[16] Jacob Krüger, Mukelabai Mukelabai, Wanzi Gu, Hui Shen, Regina Hebig, and Thorsten Berger. 2019. Where is my Feature and What is it About? A Case Study on Recovering Feature Facets. *J Syst Softw* 152 (2019).
[17] Jacob Krüger, Louis Nell, Wolfram Fenske, Gunter Saake, and Thomas Leich. 2017. Finding Lost Features in Cloned Systems. In *SPLC.*
[18] Jacob Krüger, Marcus Pinnecke, Andy Kenner, Christopher Kruczek, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2018. Composing Annotations Without Regret? Practical Experiences Using FeatureC. *Softw Pract Exper* 48, 3 (2018).
[19] Jacob Krüger, Jens Wiemann, Wolfram Fenske, Gunter Saake, and Thomas Leich. 2018. Do you Remember this Source Code?. In *ICSE.*
[20] Elias Kuiter, Jacob Krüger, Sebastian Krieter, Thomas Leich, and Gunter Saake. 2018. Getting Rid of Clone-and-Own: Moving to a Software Product Line for Temperature Monitoring. In *SPLC.*
[21] Yang Li, Sandro Schulze, and Gunter Saake. 2017. Reverse Engineering Variability from Natural Language Documents: A Systematic Literature Review. In *SPLC.*
[22] Daniel Lohmann, Fabian Scheler, Reinhard Tartler, Olaf Spinczyk, and Wolfgang Schröder-Preikschat. 2006. A Quantitative Analysis of Aspects in the eCos Kernel. In *EuroSys.*
[23] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability with FeatureIDE.* Springer.
[24] Damir Nešić, Jacob Krüger, Ştefan Stănciulescu, and Thorsten Berger. 2019. Principles of Feature Modeling. In *ESEC/FSE.*
[25] Frank Van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering.* Springer.
[26] Kentaro Yoshimura, Dharmalingam Ganesan, and Dirk Muthig. 2006. Assessing Merge Potential of Existing Engine Control Systems into a Product Line. In *SEAS.*