Understanding the Re-Engineering of Variant-Rich Systems: An Empirical Work on Economics, Knowledge, Traceability, and Practices

General Information

Reference
Jacob Krüger. Understanding the Re-Engineering of Variant-Rich Systems: An Empirical Work on Economics, Knowledge, Traceability, and Practices. Otto-von-Guericke University Magdeburg, 2021.
Reviewers

1. Prof. Dr. rer. nat. habil. Gunter Saake (Otto-von-Guericke University Magdeburg, Germany)
2. Prof. Dr.-Ing. Thomas Leich (Harz University Wernigerode, Germany)
3. Prof. dr. ir. Jan Bosch (Chalmers University of Technology Gothenburg, Sweden)
4. Univ.-Prof. Mag. Dr. rer. soc. oec. Rick Rabiser (Johannes Kepler University Linz, Austria)

Date of Defense
September 10, 2021
Publication DOI
http://dx.doi.org/10.25673/39349

Summary (English/Englisch)

Context: Most modern software systems exist in different variants to address a variety of requirements, such as customer requests, regulations, or hardware restrictions. To benefit from the high commonality between variants, such variant-rich systems (e.g., Linux kernel, automotive software, webservers) usually reuse existing artifacts (which implement so-called features). In fact, in many organizations, a variant-rich system establishes itself over time through developers using clone & own (i.e., copying and adapting a variant) as a reuse strategy. Typically, the maintenance burden of having numerous separated variants forces an organization to re-engineer its cloned variants into a reusable platform by adopting concepts of software product-line engineering. Despite the practical prevalence of this re-engineering scenario, most research has focused on the proactive adoption (i.e., starting from scratch) of platform engineering.

Objective: In this dissertation, we empirically study four closely related properties in the context of variant-rich systems, namely economics, knowledge, traceability, and practices. Note that, while we focus on the re-engineering of cloned variants into a platform, many of our findings are relevant for engineering any (variant-rich) software system. More precisely, we aim to contribute an empirics-based body-of-knowledge that can guide organizations in planning and monitoring their (re-)engineering projects. In parallel, our studies advance on educated guesses, which are widely used to reason on variant-rich systems. To this end, we aim to provide economical data that allows to compare and understand the differences between clone & own and platform engineering. Since our findings highlight the economical impact and close relation of knowledge and feature traceability, we further aim to provide a detailed understanding of these two properties in the context of re-engineering projects. Finally, we aim to synthesize all of our findings and connect them to contemporary software- engineering practices to derive processes and recommendations for planning, initiating, steering, and monitoring platform engineering.

Method: To address our objectives, we relied on a number of empirical research methods to collect data from various sources. In most cases, we built on eliciting qualitative data from the literature, which we identified through systematic literature reviews. To enrich that data, we conducted interview and online surveys, measurement and multi-case studies, as well as experiments; which we selected and employed based on their feasibility to address a certain objective. By synthesizing from different sources, we aimed to improve the validity of our data to provide reliable insights for researchers and practitioners.

Results: On an abstract level, we can summarize four key contributions. First, we contribute a rich dataset on the economics of (re-)engineering variant-rich systems, from which we derive the core insight that moving towards platform engineering (e.g., via more systematic clone management) is economically promising. Second, we contribute an understanding of developers memory and how to support their knowledge needs, leading to the core insight that expensive recovery activities can be mitigated by enforcing suitable documentation techniques (e.g., feature traceability). Third, we contribute insights on how different feature traces impact developers program comprehension, based on which our core insight is that feature traceability should ideally be independent of configurability. Finally, we contribute a process model and recommendations on how to (re-)engineer variant-rich systems, with our core insight being that carefully planning and periodically assessing a variant-rich system helps to exploit its full potential (e.g., in terms of cost savings).

Conclusion: Overall, we provide detailed insights into four important properties that help organizations as well as researchers understand and guide (re-)engineering projects for variant-rich systems. We

discuss these insights and their connections to each other as well as to contemporary software-engineering practices, enabling others to adopt them to different scenarios. So, our contributions involve the synthesis and considerable extension of the existing body-of-knowledge on (re-)engineering variant-rich systems.

Kurzfassung (German/Deutsch)

Kontext: Moderne Softwaresysteme werden in einer Vielzahl an Varianten angeboten um verschiedenste Anforderungen von Kunden, Regulierungen oder der Hardware zu bedienen. Da sich die einzelnen Varianten solcher variantenreichen Systeme (z.b. der Linux Kernel, Fahrzeugsoftware oder Webserver) sehr ähneln, können Entwickler die Wiederverwendung existierender Artefakte, die ein bestimmtes Feature implementieren, während der Entwicklung ausnutzen. Dabei werden in den meisten Fällen Varianten zuerst nur kopiert und an neue Anforderungen angepasst. Da solche kopierten Varianten voneinander losgelöst sind, wird deren Wartung in den meisten Fällen immer kostenintensiver, was oftmals dazu führt, dass die Entwickler eine wiederverwendbare Softwareplattform aus diesen extrahieren. Obwohl dieser extraktive Ansatz in der Praxis am verbreitetsten ist, fokussiert sich die Forschung meistens darauf, Entscheidungshilfen für die proaktive Neuentwicklung einer Plattform zu entwickeln.

Ziele: In dieser Dissertation werden empirische Methoden genutzt, um vier zusammenhängende Eigenschaften mit Bezug zu variantenreichen Systemen zu untersuchen: Kosten, Wissen, Nachvollziehbarkeit und Verfahren. Anzumerken ist, dass trotz unseres Fokus auf den extraktiven Ansatz, viele unserer Erkenntnisse für jegliche (variantenreiche) Softwaresysteme relevant sind. Wir sammeln empirische Daten um den wissenschaftlichen Stand in Bezug auf variantenreiche Systeme zu erweitern und Unternehmen bei der Planung und Beobachtung von Softwareprojekten zu unterstützen. Dazu vergleichen wir zuerst die Kosten um Varianten basierend auf Kopien oder einer Plattform zu entwickeln. Bei diesen Untersuchungen haben wir festgestellt, dass insbesondere das Wissen der Entwickler und dessen Nachvollziehbarkeit die Entwicklung variantenreicher Systeme beeinflussen. Nachdem wir diese beiden Eigenschaften detaillierter untersucht haben, kombinieren wir alle unsere Ergebnisse um Methodiken und Empfehlungen für die Planung, Initialisierung und Beobachtung von Softwareplattformen zu definieren.

Methodik: Unsere Forschung basiert auf verschiedenen empirischen Methoden um Daten aus unterschiedlichen Quellen zu erheben. Durch dieses Vorgehen konnten wir die Validität unserer Ergebnisse verbessern und liefern damit eine Alternative zu den begründeten Vermutungen auf denen sich Entscheidungen bezüglich variantenreicher Systeme bisher meist stützen. Wir haben meist systematische Literaturreviews genutzt um qualitative Daten aus existierenden Arbeiten zu erheben. Um diese Daten zu komplementieren, haben wir dem jeweiligen Ziel angepasste Methoden genutzt, beispielsweise Interviews, Umfragen, Fallstudien oder Experimente.

Ergebnisse: Stark zusammengefasst können wir vier Kernergebnisse definieren. Erstens liefern wir Daten zu den Kosten der Entwicklung variantenreicher Systeme, die zeigen, dass Unternehmen darauf hinarbeiten sollten, ihre Entwicklung in Richtung einer Plattform zu systematisieren. Zweitens bieten wir Einsichten dazu, welches Wissen Entwickler benötigen und aus welchen Quellen sie dieses gewinnen können, wobei unsere zentrale Einsicht die Notwendigkeit geeigneter Dokumentationen ist. Drittens haben wir die Nachvollziehbarkeit von Features als eine Art der Dokumentation untersucht und festgestellt, dass diese idealerweise nicht auf Techniken für die Konfiguration von Software aufbaut. Zuletzt haben wir Methodiken und Empfehlungen aus unseren Ergebnissen synthetisiert und die Einsicht gewonnen, dass die Planung und regelmäige Evaluierung variantenreicher Systeme hilft, deren Nutzen zu maximieren.

Fazit: Insgesamt beschreiben wir in dieser Dissertation detaillierte Einsichten zu vier essentiellen Eigenschaften für die Entwicklung und Extraktion von variantenreichen Systemen. Damit diese Eigenschaften an verschiedene Szenarien angepasst werden können, diskutieren wir ihre Zusammenhänge miteinander und mit aktuellen Methoden der Softwareentwicklung. Daraus resultiert eine Synthese und erhebliche Erweiterung des existierenden Wissensstandes bezüglich der Entwicklung variantenreicher Systeme.