Separation of Concerns: Experiences of the Crowd

Jacob Krüger
Harz University of Applied Sciences, Germany
Otto-von-Guericke-University Magdeburg, Germany
jkrueger@hs-harz.de

ABSTRACT

Separation of concerns is an important concept to define meaningful artifacts of a system. For software product lines (SPLs), concerns are described as features—that can be reused to configure and automatically derive variants of the system. Despite being extensively investigated, several fundamental questions regarding SPLs can only be answered vaguely. In particular, when and how developers separate a feature from a system. We aim to investigate this question by employing community question-answering (CQA) systems, which allow developers to share their experiences.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; • **General and reference** → *Empirical studies*;

KEYWORDS

Separation of concerns, product line, empirical study

ACM Reference Format:
Jacob Krüger. 2018. Separation of Concerns: Experiences of the Crowd. In *SAC 2018: Symposium on Applied Computing, April 9–13, 2018, Pau, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3167132.3167458

1 INTRODUCTION

Separation of concerns [17] is an important concept to structure systems into meaningful artifacts. In SPLs, the standard notion for concerns are *features* that describe common and variable functionalities [1]. While features are an established concept, there are still several uncertainties about them. For example, it is unclear what comprises a feature—resulting in different definitions [1–3]—or which granularities features have [6, 14]—potentially causing problems during refactoring [9, 12]. Separating and extracting a feature from an existing system promises several benefits but is risky and simultaneously costly [1, 4, 10, 11]. Therefore, an organization should know in advance if this is suitable for each particular feature. However, corresponding investigations on physical separation of features are rare and of limited size [5, 15].

SAC 2018, April 9–13, 2018, Pau, France

ACM ISBN 978-1-4503-5191-1/18/04.
https://doi.org/10.1145/3167132.3167458

Thus, from a practical point of view, several questions are of interest and currently remain unclear:

- RQ_1 Which pros and cons has physical separation of features in practice?
- RQ_2 Which types of source code do developers separate?
- RQ_3 Which criteria apply developers to decide when to separate a feature?

We aim to utilize CQA systems, such as Stack Overflow [16], to investigate these research questions in a large-scale study. In such systems, the crowd—comprising experienced developers from around the world—can ask and answer questions; thereby providing a large pool of available data and potential participants. To ensure a systematic methodology and improve the repeatability of our work, we will adopt guidelines for systematic literature reviews (SLRs) as proposed by Kitchenham and Charters [8]. However, conducting empirical studies in CQA systems is connected to several challenges [13], for example to identify whether the investigated topic is appropriately discussed, which may be problematic in the context of SPLs.

Answering our research questions helps to decide if features should be separated in specific scenarios. Thus, the practical applicability of SPLs for an organization can be scoped. We remark that we do not focus on a specific SPL implementation but consider any physical separation, for example, components and aspect-oriented programming (AOP) [7].

2 METHODOLOGY

For this research, we aim to conduct empirical studies based on CQA systems [13], complemented by additional surveys in industrial and open-source settings. We aim to consolidate experiences of developers from all over the world. As first step, we start crawling existing data in CQA systems that are related to our research questions. In addition, we will post questions ourselves to receive more precise responses. The validity and quality of the corresponding results depends on several factors we cannot influence [13, 16], wherefore we have to interpret them with caution.

In a second step, we aim to address this issue, using the derived findings and existing literature, for example, by Berger et al. [2], Liebig et al. [14], and Siegmund et al. [15], to design further empirical studies. We will use the investigated CQA systems, industrial partners, and open-source projects to identify participants for our studies that are experienced with SPLs and physical separation of concerns in general. For this, we aim to consider community members that answered to relevant questions in the previous step, complemented by sentiment analysis and expert identification [16]. This can

Table 1: Selected results of searching Stack Overflow.

Search String	Results		
	All	Questions	Answers
"separation of concerns"	6.362	1.090	2.067
"aspect-oriented programming"	1.734	306	431
"software product line"	20	9	3

help to scope and validate the usability of physical separation for different features by answering our research questions.

3 PRELIMINARY RESULTS

Currently, we develop a tool that enables us to crawl different CQA systems. We will use it to identify questions and answers that are related to our research. However, as CQA systems are not designed to perform empirical studies on them, several challenges arise [13]. Besides technical issues, we are facing the following four problems:

- Which CQA systems to crawl?
- Which search strings return suitable results?
- How can we automatically identify relevant results?
- How can we merge these results?

To address these questions, we start to manually analyze Stack Overflow, which is one of the most widely used CQA systems focusing on software development.

Adopting guidelines for SLRs [8], we first test different search strings. We summarize the outcome in Table 1, in which we distinguish between: All returned results, only questions, and only *accepted* answers (independent of the question). We remark that other metrics, for example highly voted answers (not only accepted ones), may be also—or even more—appropriate. Because not all results are connected to our research, scoping the search string, automatic extraction of data, and merging results are essential. To gain first insights, we analyze all questions that contain AOP as a term (306). We define as inclusion criteria that pros and cons are discussed, and that an answer is accepted (197).

Initial results indicate several pros and cons for AOP, contributing to our first research question (RQ_1):

Pro We identify several responses that are related to the expected benefits of AOP: It is an mechanism to facilitate the extension of multiple methods with the same functionality, and improves maintainability as well as understandability of code. In particular, the ideal (but also only seen) use case are orthogonal concerns that do not affect the remaining program.

Con Some negative responses are connected to the separation of concerns and automatic code weaving, resulting in the anti-pattern *action at a distance*. Thus, developers are not aware how the productive source code will look like. Other responses criticize missing tools and that the separation of concerns could break analyses, e.g., for test coverage. As any aspect can be implemented with object-orientation alone, some users also reject this idea completely.

Overall, we find that the pros and cons regarding AOP are addressing different issues in using this concept. Thus, further investigations on the usage scenarios and how to resolve problems can be based on our findings.

4 CONCLUSION

Separation of concerns is important to structure systems. Still, despite its importance, several uncertainties about assumed pros and cons remain. Previous works are often limited in size, which we aim to overcome by considering CQA systems. Furthermore, we combine qualitative and quantitative studies to achieve a more complete view. While several new challenges arise, first results are promising and we will extend our analysis significantly in future work.

ACKNOWLEDGMENTS

This research is supported by DFG grant LE 3382/2-1.

REFERENCES

[1] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines*. Springer.
[2] Thorsten Berger, Daniela Lettner, Julia Rubin, Paul Grünbacher, Adeline Silva, Martin Becker, Marsha Chechik, and Krzysztof Czarnecki. 2015. What is a Feature? A Qualitative Study of Features in Industrial Software Product Lines. In *SPLC*.
[3] Andreas Classen, Patrick Heymans, and Pierre-yves Schobbens. 2008. What's in a Feature: A Requirements Engineering Perspective. In *FASE*.
[4] Paul C. Clements and Charles W. Krueger. 2002. Point/Counterpoint: Being Proactive Pays Off/Eliminating the Adoption Barrier. *IEEE Softw.* 19, 4 (2002), 28–30.
[5] Janet Feigenspan, Christian Kästner, Sven Apel, and Thomas Leich. 2009. How to Compare Program Comprehension in FOSD Empirically - An Experience Report. In *FOSD*.
[6] Christian Kästner, Sven Apel, and Martin Kuhlemann. 2008. Granularity in Software Product Lines. In *ICSE*.
[7] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. 1997. Aspect-Oriented Programming. In *ECOOP*.
[8] Barbara A. Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE-2007-01.
[9] Jacob Krüger, Marcus Pinnecke, Andy Kenner, Christopher Kruczek, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. Composing Annotations Without Regret? Practical Experiences Using FeatureC. *Softw. Pract. Exper.* (2017).
[10] Jacob Krüger. 2017. Lost in Source Code: Physically Separating Features in Legacy Systems. In *ICSE*.
[11] Jacob Krüger, Wolfram Fenske, Jens Meinicke, Thomas Leich, and Gunter Saake. 2016. Extracting Software Product Lines: A Cost Estimation Perspective. In *SPLC*.
[12] Jacob Krüger, Louis Nell, Wolfram Fenske, Gunter Saake, and Thomas Leich. 2017. Finding Lost Features in Cloned Systems. In *SPLC*.
[13] Jacob Krüger, Ivonne Schröter, Andy Kenner, and Thomas Leich. 2017. Empirical Studies in Question-Answering Systems: A Discussion. In *CESI*.
[14] Jörg Liebig, Sven Apel, Christian Lengauer, Christian Kästner, and Michael Schulze. 2010. An Analysis of the Variability in Forty Preprocessor-Based Software Product Lines. In *ICSE*.
[15] Janet Siegmund, Christian Kästner, Jörg Liebig, and Sven Apel. 2012. Comparing Program Comprehension of Physically and Virtually Separated Concerns. In *FOSD*.
[16] Ivan Srba and Maria Bielikova. 2016. A Comprehensive Survey and Classification of Approaches for Community Question Answering. *ACM Trans. Web* 10, 3 (2016), 1–63.
[17] Peri Tarr, Harold Ossher, William Harrison, and Stanley M Sutton Jr. 1999. N Degrees of Separation: Multi-Dimensional Separation of Concerns. In *ICSE*.