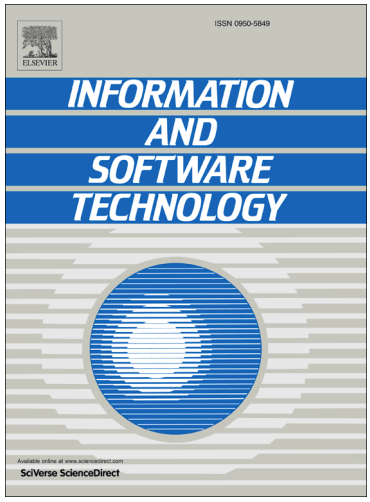

Variability mechanisms in software ecosystems

Thorsten Berger, Rolf-Helge Pfeiffer, Reinhard Tartler, Steffen Dienst,
Krzysztof Czarnecki, Andrzej Wąsowski, Steven She

PII: S0950-5849(14)00122-0
DOI: http://dx.doi.org/10.1016/j.infsof.2014.05.005
Reference: INFSOF 5462

To appear in: *Information and Software Technology*

Please cite this article as: T. Berger, R-H. Pfeiffer, R. Tartler, S. Dienst, K. Czarnecki, A. Wąsowski, S. She, Variability mechanisms in software ecosystems, *Information and Software Technology* (2014), doi: http://dx.doi.org/10.1016/j.infsof.2014.05.005

Variability Mechanisms in Software Ecosystems

Thorsten Berger[a], Rolf-Helge Pfeiffer[b], Reinhard Tartler[c], Steffen Dienst[d], Krzysztof Czarnecki[a], Andrzej Wąsowski[b], Steven She[a]

[a] *University of Waterloo, Canada*
[b] *ITU Copenhagen, Denmark*
[c] *University of Erlangen-Nuremberg, Germany*
[d] *University of Leipzig, Germany*

Abstract

Context: Software ecosystems are increasingly popular for their economic, strategic, and technical advantages. Application platforms such as Android or iOS allow users to highly customize a system by selecting desired functionality from a large variety of assets. This customization is achieved using variability mechanisms.
Objective: Variability mechanisms are well-researched in the context of software product lines. Although software ecosystems are often seen as their conceptual successors, such mechanisms in ecosystems are much less understood. Our objective is to improve empirical understanding of variability mechanisms used in successful software ecosystems.
Method: We analyze five ecosystems, ranging from the Linux kernel through Eclipse to Android. A qualitative analysis identifies and characterizes variability mechanisms together with their organizational context. This analysis leads to a conceptual framework that unifies ecosystem-specific aspects using a common terminology. A quantitative analysis investigates scales, growth rates, and—most importantly—dependency structures of the ecosystems.
Results: In all the studied ecosystems, we identify rich dependency languages and variability descriptions that declare many direct and indirect dependencies. Indirect dependencies to abstract capabilities, as opposed to concrete variability units, are used predominantly in fast-growing ecosystems. We also find that variability models—while providing system-wide abstractions over code—work best in centralized variability management and are, thus, absent in ecosystems with large free markets. These latter ecosystems tend to emphasize maintaining capabilities and common vocabularies, dynamic discovery, and binding with strong encapsulation of contributions, together with uniform distribution channels.
Conclusion: The use of specialized mechanisms in software ecosystems with large free markets, as opposed to software product lines, calls for recognition of a new discipline—variability encouragement.

Keywords: software ecosystems, empirical software engineering, software product lines, variability management, mining software repositories

1. Introduction

Large software ecosystems implement *variability*—the diversity of systems they offer—using radically different implementation techniques, also known as *variability mechanisms*. Consider the Linux kernel and the Android application platform for mobile devices. The former is a highly configurable system that implements variability in the source code, by conditionally compiling the desired functionality. The latter is a service-oriented architecture that encourages variability by letting users easily install new extensions (apps). Yet, both successfully facilitate, manage, and technically support a high degree of customization freedom for the intended recipients of the systems.

The Linux kernel and Android are examples of two major classes of large, highly successful software ecosystems. Linux manages variability centrally, carefully controlling the admission of new features into its official release. Android manages variability decentrally, embracing and encouraging variability within a free market of apps. Both systems rely on different mechanisms to achieve their goals.

The Linux kernel uses mechanisms known from *software product line engineering* (SPLE) [1, 2]. SPLE allows companies to efficiently create portfolios of systems in an application domain by leveraging the commonalities and carefully managing the variabilities among the systems [3]. For instance, the Linux kernel supplements conditional compilation with a *variability model*, which abstractly represents thousands of variabilities, such as drivers and processor architectures. Such models are popular means to manage variability. Despite using mechanisms of SPLE, the Linux kernel is more than a product line—it is a *software ecosystem* [4]. Around 7,800 developers from 800 companies have helped to more than double the code base from 6.6M to 15M lines of code (LOC) within seven years [5].

Android also manages huge variability, but in a more compositional and open way. Users derive a concrete system by selecting apps from online repositories using an