Feature-to-Code Mapping in Two Large Product Lines

Thorsten Berger[1], Steven She[2], Rafael Lotufo[2],
Krzysztof Czarnecki[2], and Andrzej Wąsowski[3]

[1] University of Leipzig, Germany berger@informatik.uni-leipzig.de
[2] University of Waterloo, Canada {shshe, rlotufo, kczarnec}@gsd.uwaterloo.ca
[3] IT University of Copenhagen, Denmark wasowski@itu.dk

Large software product lines have complex *build systems* that enable compiling the source code into different products that make up the product line. Unfortunately, the dependencies among the available build options, which we refer to as *features* and their mapping to the source code they control, are implicit in complex imperative build-related logic. As a result, reasoning about dependencies is difficult; this not only makes maintenance of the variability harder, but also hinders development of support tools such as feature-oriented traceability support, debuggers for variability models, variability-aware code analyzers, or test schedulers for the product line. Thus, we advocate the use of explicit *variability models*, consisting of a *feature model* specifying the available features and their dependencies and a *mapping* between product specifications conforming to the feature model and the implementation assets.

Previously, we extracted feature models from the Linux kernel and the Ecos embedded operating system. The Ecos model directly embeds the feature-to-code mapping. However, this is not the case for Linux. Now, we extract the feature-to-code mapping from the build systems of the prominent operating systems Linux and FreeBSD.

We represent the mapping as *presence conditions*. A presence condition (PC) is an expression on an implementation artifact written in terms of features. If a PC evaluates to *true* for a configuration, then the corresponding artifact is included in the product.

We show that the extraction of mappings from build systems is feasible. We extracted 10,155 PCs, each controlling the inclusion of a source file in a build. The PCs reference the total of 4,774 features and affect about 8M lines of code. We publish[4] the PCs for Linux and FreeBSD as they constitute a highly realistic benchmark for researchers and tool designers.

In the poster, we describe the build systems of the Linux and FreeBSD kernel as well as our approach to transforming the imperative build-system logic into a large Abstract Syntax Tree (AST) and to deriving presence conditions. Furthermore, we expand on basic characteristics of the resulting expressions. We hope our work deepens understanding of variability in build systems and that the insights will eventually lead to extracting complete variability models encompassing the feature model and the mapping from features to code.

[4] http://code.google.com/p/variability