Leviathan: SPL Support on Filesystem Level

Wanja Hofer[1], Christoph Elsner[1,2], Frank Blendinger[1],
Wolfgang Schröder-Preikschat[1], and Daniel Lohmann[1]

[1] Friedrich–Alexander University Erlangen–Nuremberg, Germany
[2] Siemens Corporate Research & Technologies, Erlangen, Germany
{hofer,elsner,wosch,lohmann}@cs.fau.de

A lot of configurable software, especially in the domain of embedded and operating systems, is configured with a source preprocessor, mostly to avoid run-time overhead. However, developers then have to face a myriad of preprocessor directives and the corresponding complexity in the source code, even when they might only be working on the implementation of a single feature at a time. Thus, it has long been recognized that tool support is needed to cope with this '#ifdef hell'. Current approaches, which assist the software developer by providing preprocessed views, are all bound to a special integrated development environment. This eliminates them from being used both in industry settings (where domain-specific toolchains are often mandated) and in open-source projects (where diverse sets of editors and tools are being used).

We therefore propose to tackle the problem at a lower level by providing variant views via a *virtual filesystem*. Our Leviathan filesystem allows the developer to mount one or more concrete product variants to work on. A variant is hereby defined as a set of enabled and disabled features (possibly output by a feature modeling tool to ensure correctness). A mounted variant essentially provides virtual files and directories, which are in fact slices of the original configurable code base, preprocessed by Leviathan's corresponding preprocessor component (e.g., CPP or M4). This technique enables the use of arbitrary file-based tools and tasks on that view. Operation includes read-only tasks such as reasoning about variants by viewing the differences between, or feeding them into syntax validators or code metric analysis tools. Furthermore, Leviathan's virtual files are also *editable* with arbitrary tools; Leviathan will merge back the changes into the configurable code base transparently in the background.

Leviathan's write-back support enables the developer to make changes directly on the mounted view. For instance, he can directly debug a mounted variant and modify the variant's code to get rid of a bug, eventually saving the changes in his editor. In the background, the write request will be handled by Leviathan's write-back engine, which is responsible for mapping *source configuration blocks* (enclosed in preprocessor directives) to *variant blocks* (actually visible in the view). This mapping and merging is either done heuristically (if the source configuration blocks are rather large in a given SPL) or with the help of markers, which Leviathan inserts as language-dependent comments in the mounted view to make the write-back process completely unambiguous.

By providing toolchain-independent views on preprocessor-configured code bases, Leviathan introduces SPL support in unprecedented domains like open-source projects (e.g., Linux) and industry projects.

J. Bosch and J. Lee (Eds.): SPLC 2010, LNCS 6287, p. 491, 2010.