FeatureCommander: Colorful #ifdef World

Janet Feigenspan, Maria Papendieck
University of Magdeburg

Christian Kästner
Philipps University Marburg

Mathias Frisch, Raimund Dachselt
University of Magdeburg

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*User interfaces*; D.2.3 [**Software Engineering**]: Coding Tools and Techniques—*Top-down programming*

Keywords

FeatureCommander, Preprocessor, Program Comprehension

Software product line engineering is a promising paradigm to create variable software. In practice, conditional compilation is often used to implement software product lines, such that code of features is annotated with *ifdef directives*. However, preprocessor usage can lead to obfuscated source code that is hard to understand and maintain. In the literature, *ifdef directives* are even referred to as "ifdef hell" [1, 5].

To support developing and maintaining preprocessor-based software, we implemented FeatureCommander. We show a screenshot in Figure 1. FeatureCommander is a prototype that makes consistent use of (a) background colors to highlight source code of features (i.e., code annotated with an ifdef directive) and (b) views to navigate features in the code base.

First, we highlight source-code fragments that belong to a feature *OPT_PRIOCPL* (wrapped with #ifdef `CONFIG_XENO_OPT_PRIOCPL`) with that feature's color. The benefit of colors is that humans process colors preattentively and, thus, considerably faster than text [3]. In addition to background colors, we illustrate *ifdef directives* and their nesting with vertical bars next to the code editor. Hence, locating and tracking (scattered, nested, and even very long) conditional code fragments in a file becomes easier.

Second, to navigate between features in different files, we provide additional views that use the same color metaphor. In the file browser (explorer view), we represent the amount of feature code using a small bar chart, again using the features respective colors. Vertical bars in this chart represent the relative amount of feature code, again using each feature's color. The same visualization is used for files and entire folders. In addition, we provide facilities of navigating the source tree by feature. This way, developers can get a quick overview of how features are distributed and navigate between them.

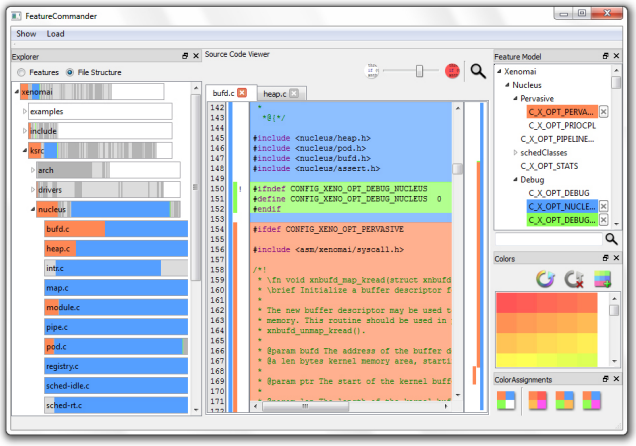

Figure 1: Screenshot of FeatureCommander, showing explorer view (left), source-code view (middle), and feature-model view (top right). Colors can be dragged from the color palette (bottom right) and dropped on a feature in any view.

To scale color usage to large software systems, we provide an as-needed mapping of colors to features. By default, each feature is assigned a shade of gray. That is, we see different shades as background colors and in the bar chart. Shades of gray are sufficient to recognize that there is some feature code and that different features are involved. To analyze a feature (or small set of features) in more detail, the developer assigns more distinguishable colors to individual features per drag and drop. This reflects that developers typically work with only few features at a time, whereas the major part of the source code does not need to be highlighted.

This way, background-color usage scales to large software projects with several hundred features. This is a major advantage of FeatureCommander compared to existing color concepts, for example as implemented in CIDE [4].

To evaluate whether the implemented background-color concepts work on a large scale, we conducted a controlled experiment based on Xenomai[1], a large program with over 160,000 lines of code and over 300 features [2]. We found that background colors speed up the comprehension process

[1] `http://www.xenomai.org`

and that subjects like the color idea. Details of the experiment, the prototype implementation, and a video demonstrating the core concepts are available at `http://fosd.net/fc`.

Acknowledgment

Feigenspan's and Frisch's work is supported by BMBF project 01IM08003C (ViERforES).

1. REFERENCES

[1] J. Favre. Understanding-In-The-Large. In *Proc. Int'l Workshop on Program Comprehension*, page 29. IEEE CS, 1997.

[2] J. Feigenspan, M. Schulze, M. Papendieck, C. Kästner, R. Dachselt, V. Köppen, and M. Frisch. Using Background Colors to Support Program Comprehension in Software Product Lines. In *Proc. Int'l Conf. Evaluation and Assessment in Software Engineering (EASE)*, pages 66–75. Institution of Engineering and Technology, 2011.

[3] B. Goldstein. *Sensation and Perception*. Cengage Learning Services, fifth edition, 2002.

[4] C. Kästner, S. Apel, and M. Kuhlemann. Granularity in Software Product Lines. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 311–320. ACM Press, 2008.

[5] D. Lohmann et al. A Quantitative Analysis of Aspects in the eCos Kernel. In *Proc. Europ. Conf. Computer Systems (EuroSys)*, pages 191–204. ACM Press, 2006.