Science of Computer Programming 77 (2012) 129–130

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Preface to the special issue on feature-oriented software development (FOSD 2009)

The idea of feature-oriented software development (FOSD) is to decompose a software system in terms of the features it provides. A feature is a unit of functionality that satisfies a requirement, represents a design decision, and provides a potential configuration option. Typically, from a set of features, many different software systems, called variants, can be generated that share common features and differ in other features. The complete set of variants is also called a software product line. Systematic software product-line development based on features has a number of benefits, among others, the ability to generate reliable and efficient software systems based on well-tested and verified software artifacts. So it is not surprising that the product-line paradigm has received considerable attention in research and industry.

The FOSD workshop series aims at exploring the foundations, applications, and implications of feature-oriented software development. The FOSD 2009 workshop was held in conjunction with MODELS, GPCE, and SLE in Denver, Colorado, USA, and the contributors were invited to submit subsequently extended versions to this special issue. All submissions received three reviews, to the usual standards of a journal publication. The following three papers passed the rigorous judgment of the reviewers.

A distinguishing property of FOSD, compared to other product-line engineering approaches, is the clean mapping between features (problem-space artifacts) and their implementations (solution-space artifacts), which is achieved by expressive module, composition, and annotation mechanisms. The article "Remodularizing Java Programs for Improved Locality of Feature Implementations in Source Code" by Andrzej Olszak and Bo N. Jørgensen presents an approach to restructuring Java programs such that features are made explicit as units of modularization (i.e., one feature is implemented by one module). The problem is that, often, it is not trivial to decide which code implements which feature — features tend to vanish in object-oriented programming. The authors' approach supports both forward and backward refactoring between feature-oriented and object-oriented decomposition. The authors report on two case studies that indicate that their approach requires moderate manual effort and reduces tangling and scattering of feature implementations in source code.

The article "rbFeatures: Feature-Oriented Programming with Ruby" by Sebastian Günther and Sagar Sunkle addresses a related problem: how should features be represented in a program? The authors propose to represent features as first-class entities of the programming language. They introduce rbFeatures, a feature-oriented programming language implemented on top of the dynamic programming language Ruby. With rbFeatures, programmers treat software product lines, variants, and features as first-class entities. This approach is very flexible as it allows programmers to modify features, variants, and product lines at run time, including the extension of the product line with new features and the handling of multiple variants in a single process. The article describes the implementation of rbFeatures and reports on its application to two case studies.

There are alternative approaches to representing features during the software life cycle. In tools such as AHEAD and FeatureHouse, features are represented as external entities such as transformations, not as first-class entities in the programming languages. In this setting, features statically add new elements to a program or refine existing elements. This raises the issue of visibility and access control in feature-oriented programming. The article"Access Control in Feature-Oriented Programming" by Sven Apel, Sergiy Kolesnikov, Jörg Liebig, Christian Kästner, Martin Kuhlemann, and Thomas Leich identifies some subtle issues that can occur in applying existing object-oriented access control mechanisms to features, and finds that in some cases the semantics are not well defined. The authors then propose a set of feature-specific access control modifiers. This yields a lattice of possible combinations of object-oriented and feature-oriented access control

0167-6423/$ – see front matter
doi:10.1016/j.scico.2011.02.002

modifiers. The article concludes by evaluating the applicability of the various elements of this lattice in a number of existing feature-oriented applications.

Finally, we would like to thank all the reviewers who helped with the article selection process, both for FOSD 2009 and for this special issue.

Sven Apel*
Christian Lengauer
University of Passau, Germany
E-mail addresses: apel@uni-passau.de (S. Apel), lengauer@uni-passau.de (C. Lengauer).

Julia Lawall
University of Copenhagen, Denmark
E-mail address: julia@diku.dk.

Available online 16 February 2011

* Corresponding editor.