Reverse Engineering Architectural Feature Models

Mathieu Acher[1], Anthony Cleve[1], Philippe Collet[3],
Philippe Merle[2], Laurence Duchien[2], and Philippe Lahire[3]

[1] PReCISE Research Centre, University of Namur, Belgium
`{mac, acl}@info.fundp.ac.be`
[2] INRIA Lille-Nord Europe, Univ. Lille 1 - CNRS UMR 8022, France
`{philippe.merle,laurence.duchien}@inria.fr`
[3] Université de Nice Sophia Antipolis - I3S (CNRS UMR 6070), France
`{collet,lahire}@i3s.unice.fr`

Software product line (SPL) engineering aims at generating tailor-made software variants for the needs of particular customers or environments. SPL principles and techniques are gaining more and more attention as a means of efficiently producing and maintaining multiple similar software products, exploiting what they have in common and managing what varies among them.

It is not always feasible to design and implement a complete mass customization production line to support the full scope of products needed on the foreseeable horizon. In many cases, SPL practitioners rather have to deal with (legacy) software systems, that were not initially designed as SPLs. It is the case of FraSCAti, a large and highly configurable component and plugin based system, that have constantly evolved over time and now offers a large number of variants, with many configuration and extension points. The variability of such existing and feature rich systems should be properly modeled and managed.

A first and essential step is to explicitly identify and represent the variability of a system, including complex constraints between architectural elements. We rely on feature models that are widely used to model the variability of an SPL in terms of mandatory, optional and exclusive features as well as Boolean constraints over the features [5]. Feature models characterize the scope [3] of an SPL by specifying the set of combination of features (configurations) supported or not by an SPL. Reverse engineering the feature model of an existing system is a challenging activity [4]. The architect knowledge is essential to identify features and to explicit interactions or constraints between them. But the manual creation of feature models is both time-consuming and error-prone. On a large scale, it is very difficult for an architect to guarantee that the resulting feature model correctly represents the valid combination of features supported by the software system. The scope defined by the feature model should not be too large (otherwise some unsafe composition of the architectural elements are authorized) or too narrow (otherwise it is a symptom of unused flexibility of the architecture). Both automatic extraction from existing parts and the architect knowledge should be ideally combined to achieve this goal.

We present a comprehensive, tool supported process for reverse engineering architectural feature models [1]. At the starting point of the process, an intentional model of the variability – a feature model – is elaborated by the software architect. As the software architect feature model may contain errors, we develop automated techniques to extract and combine different variability descriptions of an architecture, namely a hierarchical software architecture model and a plugin dependencies model. Then, the extracted feature model and the software architect feature model are reconciled in order to reason about their differences. Advanced editing techniques are incrementally applied to integrate the software architect knowledge. The reverse engineering process is tool supported and made possible by the combined use of FAMILIAR [2] operators (aggregate, merge, slice, compare, etc.).

We illustrate the process when applied to a representative software system, FraSCAti. Our experience in the context of FraSCAti shows that the automated procedures produce both correct and useful results, thereby significantly reducing manual effort. First, the gap between the feature model extracted by the procedure and the feature model elaborated by the software architect appears to be manageable, due to an important similarity between the two feature models. However, it remains helpful to assist the software architect with automated support, in particular, to establish correspondences between features of the two feature models. The most time-consuming task was to reconcile the granularity levels of both feature models. For this specific activity, tool supported, advanced techniques, such as the safe removal of a feature, are not desirable but mandatory, since basic manual edits [5] of feature models are not sufficient. Second, the extraction procedure recovers most of the variability expressed by the software architect and encourages the software architect to correct his initial model. A manual checking of the five variability decisions imposed by the software architect shows that the extraction is not faulty. It correctly reproduces the information as described in the software artefacts of the project. Third, we learn that the software architect knowledge is required *i)* to scope the SPL architecture (e.g., by restricting the set of configurations of the extracted feature model), especially when software artefacts do not correctly document the variability of the system and *ii)* to control the accuracy of the automated procedure.

An open issue is to provide a mechanism and a systematic process to reuse the software architect knowledge, for example, for another evolution of the architectural feature model of FraSCAti.

References

1. Mathieu Acher, Anthony Cleve, Philippe Collet, Philippe Merle, Laurence Duchien, and Philippe Lahire. Reverse engineering architectural feature models. In *ECSA'11*, LNCS. Springer. material and experiments: `https://nyx.unice.fr/projects/familiar/wiki/ArchFm`.
2. Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert France. In *Symposium on Applied Computing (SAC)*, pages 1333–1340, Taiwan, March. Programming Languages Track, ACM.
3. Isabel John and Michael Eisenbarth. A decade of scoping: a survey. In *SPLC'09*, volume 446 of *ICPS*, pages 31–40. ACM, 2009.
4. S. She, R. Lotufo, T. Berger, A. Wasowski, and K. Czarnecki. Reverse engineering feature models. In *ICSE'11*, pages 461–470. ACM, 2011.
5. T. Thüm, D. Batory, and C. Kästner. Reasoning about edits to feature models. In *ICSE'09*, pages 254–264. IEEE, 2009.