Assessing Advanced Composition Mechanisms for the Integration of Software Product Lines

Alessandro Gurgel, Francisco Dantas, Alessandro Garcia

OPUS Research Group – Software Engineering Lab, Department of Informatics, PUC – RJ- Brazil

{agurgel, fneto, afgarcia}@inf.puc-rio.br

1. ABSTRACT

Software product lines (SPLs) are increasingly being built from product lines within the same domain. In this context, the integration of multiple SPLs provides the reuse of previously-implemented features across SPLs in order to promote faster time-to-market. The integration of a SPL feature into other SPLs is supported by the underlying composition mechanisms of the programming language. However, if such mechanisms require significant change effort to realize a feature integration, there is not much payoff for the reuse investment being made.

To integrate features from different SPLs, many modifications may be required in the code. The integration of a feature into other product lines may not be a trivial task even when its code is confined to a single module [1]. In this context, many advanced programming techniques, such as Aspect-Oriented Programming (AOP) [2] and Feature-Oriented Programming (FOP) [4] have emerged as viable techniques for building stable and reusable SPLs [4]. Their composition mechanisms are intended to both explicitly capture the ways features interact and improve the reuse of existing modules. Recent studies analyzed if a single SPL can be evolved with aspects or other means to smoothly accommodate changes (e.g. [4][5]). However, they do not evaluate how advanced composition mechanisms can support on-demand integrations of previously-designed SPLs.

Our research accomplishes an exploratory, non-controlled study of on-demand integration of multiple product lines. We compare the degree of reuse and stability achieved with AOP and FOP in this context. The AspectJ [6] and CaesarJ [7] languages were chosen to support the stepwise integration of several features across three SPLs in the board game domain. CaesarJ was chosen as representative of FOP as there are reports of their successful adoption in industrial projects. AspectJ was chosen due to its popularity in the AOP community.

The target product lines contain a series of (non-)crosscutting mandatory, optional and alternative features [8]. All these diverse feature characteristics exposed the techniques to a wide range of feature integrations. A complete description of the study procedures, metrics, the target product lines, and the source code of each integration scenario are available at our accompanying website [9].

Our analysis provided evidence that the use of FOP appears as a promising solution to implement functional and non-functional heterogeneous features. This happens mainly because the use of CaesarJ provides better support for refinement of collaborations which tend to increase the degree of reuse of modules and decrease the number of required modifications in comparison to AspectJ. However, we found signs that the use of FOP support for late-binding of types [1] might negatively affect the stability of the integrated SPLs in relation to persistence features that stores virtual classes [7]. When an object is loaded from a file, its type is not known as it may be stored as a specialized type by the mixin composition of modules [7]. Our study also confirms the expectation raised from previous studies [8] that AOP implementations require less effort in implementing homogenous crosscutting features.

REFERENCES