AOP-Driven Variability in Product Lines of Pervasive Computing Applications

Vander Alves, Ayla Dantas, Paulo Borba
Informatics Center
Federal University of Pernambuco – Brazil
{vra,add,phmb}@cin.ufpe.br

Position Paper

The demand for pervasive computing applications has increased. Additionally, the large number of different devices and features per device make it difficult to quickly respond to such demand and to comply with ever-increasing quality and reuse requirements. A product line approach in this domain is essential in order to meet these goals. In this context, a central issue that must be addressed by the development process is variability management.

Product Line variability is usually managed directly at the architectural level. For example, a class diagram has stereotypes in certain classes denoting variation points. The variations themselves are related to this class by some Object-Oriented variation mechanism, such as inheritance. However, this approach may lead to unnecessary complexity when used for parameterizing certain kinds of features, namely aspects. For instance, there may be either duplication of feature implementation across different classes or a tight coupling of these elements.

We propose switching this focus from the architectural level to the feature modeling level for early detection of variations. These are more abstractly represented by feature diagrams, which provide a mechanism-independent view. Such view allows a more effective configurability analysis, through which we map variations into aspects in the architectural level, where the latter provide a simpler representation of variability.

Accordingly, our approach tailors the Framework for Product Lines [1] as follows. In Core Asset Development, Feature Modeling defines the features of the product line members, identifies common features among all products as well as the variable ones for the members. Next, the product line architecture is designed according to the mandatory features. In this activity, architectural and design patterns are used to achieve a flexible architecture. Finally, variations previously identified in Feature Modeling are modeled as aspects. In particular, an aspect stereotype is created to represent a variable feature as an aspect and this latter is shown together with any auxiliary classes in the variation view, which crosscuts the architecture view and is maintained independently from it. Use of design patterns [2][3] is also recommended in the variation view.

In Product Development, we collect the aspects corresponding to the desired variable features, the auxiliary classes they use, the product line architecture, and compose them

using AOP [4] weavers. Additionally, the code is processed, the application is packaged, installed, and tested. We do not address any specific adaptation for the Management cycle of the Product Line framework.

Currently, we have run this adapted product line process in the domain of pervasive computing applications concerned with word translation. A flexible architecture was devised according to some mandatory features such as the word translation itself. Furthermore, variable features related to dynamic customization, language selection, and resource monitoring were modeled as aspects. Their composition by using AOP weavers simplified product generation and configurability significantly. We have also tried to model the same aspects using OO approaches for separation of concerns, but they were not as effective as AOP.

Future work includes deriving more products and experimenting with different domains such as games and multimedia applications. Furthermore, the development team will also be scaled to a larger number of members.

References

[1] Software Engineering Institute. *A Framework for Software Product Line Practice*. http://www.sei.cmu.edu/plp/framework.html

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[3] Jan Hannemann and Gregor Kiczales. *Design pattern implementation in Java and AspectJ*. In Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications, pages 161–173. ACM Press, 2002.

[4] Tzilla Elrad, Rovert E. Filman, and Atef *Bader*. Aspect-Oriented Programming. Communications of the ACM, 44(10):29–32, 2001.