Using Product Lines to Manage Variability in Mobile Context-Aware Applications

Dean Kramer

School of Computing and Technology
Thames Valley University, London, UK, W5 5RF
dean.kramer@tvu.ac.uk

1 Introduction and Motivation

Today, the penetration of modern smart phones is vastly increasing with over 172 million smart phones shipped worldwide in 2009 [5]. It is quickly becoming predictable that these smart phones will contain sensors such as Global Positioning System (GPS) receivers, accelerometers, close proximity sensors and digital compasses. These components make smart phones a good candidate for pervasive computing and context-awareness. Context-awareness can be described as a systems ability to adapt and alter its behavior based on the situation it is current aware of [9]. Pervasive computing and context-awareness are becoming a highly interested area as it allows for applications to become more intelligent and more usable, unlike traditional software applications which would only respond to direct user input [6].

People and clients using software systems are becoming increasingly demanding regarding configuration and system features, thus requiring customization. In software development, re-developing software from scratch for each difference is purely uneconomical. This can be solved using Software Product Lines (SPL), creating many similar products from a single set of core assets [12]. Systems can be customized using variation points within its design, thus allowing for variants to shape the final product based on the configuration. This variability can be modeled using commonly used feature-modeling, which can help express different required features and variations of features. Because of the fragmentation of hardware/firmware, firmware/hardware constrained features should be modeled identifying this dependency which currently is not supported in feature modeling.

Though a SPL approach offers a viable approach to handling variability, deriving products from SPLs can be a challenging task. To compliment SPLs, the use of Model-Driven Development (MDD) and Domain Specific Languages can help allow for model transformations.

Domain Specific Languages (DSL) allow the developer to raise abstraction away from the software implementation making development easier. DSLs can be used within SPL engineering [14], and used to help provide model transformations. These transformations include horizontal (model to model), and more importantly in my case, vertical (model to code).

In this research, I propose to help integrate the modeling of context, features and their dependencies on hardware and firmware platforms. To compliment this, a DSL will be defined as method of expressing the model, which can then provide model-code transformations providing large amounts of application structure and features based on the domain feature implementations and variations.

2 Related Work

Developing for mobile platforms vastly differs to desktop and larger systems. Firstly, there are many different constraints that need to be considered. These constraints include lack of screen size, processing power, memory, and power consumption [8]. Though this idea may look out-dated, the expediential growth in software complexities and sizes causes this to remain true. Furthermore single application development is becoming increasing rare, with the trend more on a set of similar applications. Because of this, and increasing software complexities, the management of this re-usable software needs to be made simpler. The use of product lines is already evident in mobile gaming [10].

2.1 Context-Awareness

Much work in the past for aiding the production of context-ware applications has been through the developments of frameworks. The Java Context-Aware Framework [1] is a lightweight Java-based framework for creating context-aware applications. The feasibility of using this framework is help backed by its use in medical scenarios [2]. An issue with using this approach is the dependence on an external context-service, which for smart phone applications is not a desired method and can also bring security issues when transmitting sensitive data. Context-reasoning and challenges of mobility have been addressed with the use of a sentient object model [3].

MDA approaches have been proposed for context-aware software development. CAMEL (Context Awareness Modeling Language) [13], a DSL design for the initial stage of software development provides a method for modeling context dependent behaviors. Authors pointed out that currently the language does not handle model transformations to executable code.

2.2 Software Product Lines

UbiFEX [7] has been presented as a feature model notation to context-aware SPLs. This notation allows for context rule specific product configuration, but does not complete the need for modeling the complete system combining features and contexts.

Modeling context and its adaptation requires modelling within a product line, to help indicate how it relates to each feature. Parra et al. [11] creates a composition of assets binding context adaptation to features. This was later used with the FraSCAti platform[1], an open source implementation of the Service Component Architecture standard. This approach is similar to my plans, but there is a lack of concern for issues relating to how hardware/firmware may constrict different contexts being monitored and how features within the application may/may not be compatible with differing requirements. Furthermore, though the use of dynamic product line derivation is a useful approach for larger systems, because of different issues relating to mobile development a less dynamic derivation may prove more suitable. Firstly if a high amount of variation is in the application, this may 'bloat' the application unnecessarily, taking up already limited space on the device.

Despite the research being carried out into making product lines easier to implement, the use of SPLs for asset re-use is still having low adoption rates. This is because it affects the whole software life cycle, compared to other methods including Model Driven Development and Service Oriented Architecture [4]. Reasons for the lack of adoption include hardware/software integration issues, unpredictability caused by the recession and lack of SPL experts and low cost training.

3 Proposed Research

The primary aim of this research is to experiement and explore with a DSL to help support product derivation in context-aware mobile software product lines. This product-derivation will need to handle platform version differences, hardware differences and context differences. The proposed research will be a combination of action research and case-study based research. Within the research, two case studies will be developed, creating differing product-lines. The first of these case studies will be on a sports application, with the second more focusing on a mobile enterprise application.

3.1 Research Objectives

1. Investigate how you express the context aware environment and it's impact on application development. Modeling and defining the contexts for the product line and how it relates to the main application logic will be required. Metamodels for context, platform and application will be developed which will form the basis for the DSL.
2. Research into how to manage variability with the different contexts that will be supported within the application. Also, variability will need to be mapped and modeled to deal with different smart-phone hardware components and firmware versions.
3. Acquirement and analysis of DSL requirements, primarily driven through the development of two software product lines.

[1] http://frascati.ow2.org

4. Define and develop a DSL to help enable model-code transformations using Domain Specific Modelling (DSM) and implement over several iterations to suit case studies.
5. Use or develop an analysis framework for validating the proposed DSL. This validation will help ensure that the language meets all the requirements of a language.

3.2 Product Line Approach

The approach I propose for the product line is shown in Figure 1. This approach is made up of a product line (PL) meta model, and the DSL to specialise and deviate products into specialised applications (App). To help gain requirements of the DSL, meta-models will be developed including:

– Context Metamodel, a model indicating the different conditions that will constitute a context including user and environmental conditions.
– Platform Metamodel, a model showing the similarities and variability within the platforms. This will include hardware and software differences.
– Application Metamodel, a model describing the application. This will include different components and features included, with emphasis on core and additional/alternative parts of the application.

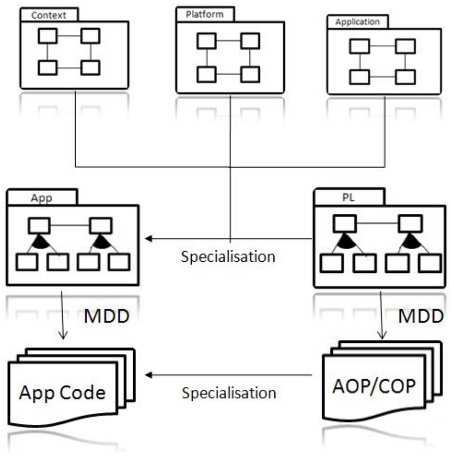

Fig. 1. Proposed Product Deviation Approach

For code generation, the PL will have software assets produced by MDD using a mix of Aspect-Oriented Programming (AOP) and Component-Oriented Programming (COP). Using aspects can easily seperate application concerns, in this case different platforms and/or contexts. These assets are then specialised using the DSL to produce partial application specific code, to which further implementation can be added.

References

1. Bardram, J.E.: The java context awareness framework (jcaf) - a service infrastructure and programming framework for context-aware applications. In: Pervasive Computing. 2005: Munchen. pp. 98–115 (2005)
2. Bardram, J.E., Nørskov, N.: A context-aware patient safety system for the operating room. In: UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing. pp. 272–281. ACM, New York, NY, USA (2008)
3. Biegel, G., Cahill, V.: A framework for developing mobile, context-aware applications. In: PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04). p. 361. IEEE Computer Society, Washington, DC, USA (2004)
4. Catal, C.: Barriers to the adoption of software product line engineering. SIGSOFT Softw. Eng. Notes 34(6), 1–4 (2009)
5. De La Vergne, H.J., Milanesi, C., Zimmermann, A., Cozza, R., Nguyen, T.H., Gupta, A., Lu, C.: Competitive landscape: Mobile devices, worldwide, 4q09 and 2009. Tech. rep., Gartner (2010)
6. Du, W., Wang, L.: Context-aware application programming for mobile devices. In: C3S2E '08: Proceedings of the 2008 C3S2E conference. pp. 215–227. ACM, New York, NY, USA (2008)
7. Fernandes, P., Werner, C., Murta, L.: Feature modeling for context-aware software product lines. In: SEKE. pp. 758–768 (2008)
8. Gaedke, M., Beigl, M., Gellersen, H.W., Segor, C.: Web content delivery to heterogeneous mobile platforms. In: ER '98: Proceedings of the Workshops on Data Warehousing and Data Mining. pp. 205–217. Springer-Verlag, London, UK (1999)
9. Häkkilä, J., Schmidt, A., Mäntyjärvi, J., Sahami, A., Åkerman, P., Dey, A.K.: Context-aware mobile media and social networks. In: MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. pp. 1–3. ACM, New York, NY, USA (2009)
10. Nascimento, L.M., Almeida, E.S.d., Meira, S.R.d.L.: A case study in software product lines - the case of the mobile game domain. In: SEAA '08: Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications. pp. 43–50. IEEE Computer Society, Washington, DC, USA (2008)
11. Parra, C., Blanc, X., Duchien, L.: Context awareness for dynamic service-oriented product lines. In: SPLC '09: Proceedings of the 13th International Software Product Line Conference. pp. 131–140. Carnegie Mellon University, Pittsburgh, PA, USA (2009)
12. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005)
13. Sindico, A., Grassi, V.: Model driven development of context aware software systems. In: COP '09: International Workshop on Context-Oriented Programming. pp. 1–5. ACM, New York, NY, USA (2009)
14. Voelter, M.: Using domain specific languages for product line engineering. In: SPLC '09: Proceedings of the 13th International Software Product Line Conference. pp. 329–329. Carnegie Mellon University, Pittsburgh, PA, USA (2009)