Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing

Andrew Carton, Siobhán Clarke, Aline Senart, Vinny Cahill
Distributed System Group,
Department of Computer Science,
Trinity College, Dublin.
{cartona, sclarke, senarta, vjcahill}@cs.tcd.ie

Abstract

The development of applications for pervasive computing presents a number of challenges to the software engineer. In particular, application adaptation based on context such as environmental factors, device limitations and connectivity, requires the programmer to handle a complex combination of factors that manifest themselves throughout the application. This position paper presents an approach to managing such complexity based on a combination of aspect-oriented development techniques, and model-driven development.

1. Introduction

Pervasive computing is a vision where mobile devices integrate seamlessly and unobtrusively into the environment in order for people to be able to interact to a greater degree with computers and technology [1]. One of the requirements for pervasive applications is to be able to dynamically adapt based on a number of factors, including environment state, device limitations, user mobility and connectivity. Two significant observations can be made from a software engineering perspective. First, these factors are likely to have an impact across the application as a whole (in other words, they are *crosscutting*). We have previously illustrated that the crosscutting nature of adaptation means that it cannot be effectively modularised by conventional programming formalisms [6]. In effect, developers are forced to design and implement such concerns repetitively throughout the application. This reduces the maintainability, extensibility and comprehensibility of the design. The second observation is that it is difficult for the software engineer to reason about the application semantics separately from the technical specifics of the implementation. This is, of course, true for any application, but the potential ranges of interacting heterogeneous devices in play in pervasive applications and the potential for different target deployment environments exacerbate the difficulties.

In this position paper, we discuss the potential to address both observations by combining aspect-oriented software development (AOSD) with model-driven development (MDD). The AOSD paradigm provides a set of techniques to modularise crosscutting behaviour. In Section 2 we summarise previous findings related to the aspect-oriented modularisation of context. MDD is an approach to support developers reasoning about applications using domain-specific modelling techniques and that also supports the automated transformation of those models to different target platforms. In Section 3 we introduce MDD in the context of pervasive computing and in Section 4, we introduce our approach to combining aspect-oriented modularisation with model-driven transformations. Section 5 briefly outlines some future work.

2. Modularising Pervasive Concerns

Through a study of a number of mobile, context-aware applications, and of related literature, it emerged that there are a number of concerns that are crosscutting in such applications. Schmidt et al. categorise context types into eight sub-categories: device, location, user, social, environmental, system, temporal, and application-specific context [7]. Using the aspect-oriented paradigm, we could design and implement concerns within these categories independently, while separately stating where in base applications adaptation should occur. We designed the concerns using Theme/UML [4], which is an extension to standard UML that provides constructs to support

First International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments (SEPCASE'07)
0-7695-2970-4/07 $20.00 © 2007 IEEE

the specification of crosscutting behaviour relative to abstract template triggers, with a corresponding means to specify the base application elements that bind to the template triggers. We implemented a selection of the concerns in AspectJ [8], which is an aspect-oriented extension to Java that provides programming constructs to modularise "aspects" (or crosscutting concerns), and to specify where and when in base code aspects apply. We compared the AspectJ implementation with a corresponding standard object-oriented approach in Java, and an approach using the Context Toolkit [9]. Results indicated that the aspect-oriented approach fared well when measured against comprehensibility, maintainability and manageability metrics [6].

3. Model-Driven Development

However, modularising crosscutting concerns does not help the software engineer with coping with the wide range of technical platforms that might be in play for pervasive applications. For example, many pervasive devices are small and embedded, with limited and varying capabilities and requirements. Designing for a single device may require low-level embedded technology knowledge together with domain expertise. Building applications for these devices is inaccessible to programmers without both these knowledge and expertise. Model-Driven Development (MDD) attempts to solve this problem by allowing developers to design the application according to a domain-specific, platform-independent specification with transformations generating the design for the platform-specific requirements of each device. In this way, a single design can also accommodate multiple platform-specific implementations that are required by different devices.

Model-Driven Architecture (MDA) [2] is essentially MDD as standardised by the OMG and emphasises the use of models throughout the development lifecycle. The MDA approach allows a platform-independent model (PIM) and multiple platform-specific models (PSM) to be defined. In the case of ubiquitous computing, a PSM will target a specific device, technology (e.g., J2ME [11], .NET Compact [12]) or framework (e.g., Context Toolkit [9], MoCoA [14]). As devices differ in size, capability and technology, we can separate the high level design (PIM) from the implementation and platform-specific details of each device (PSMs).

Transformations generate PSMs from PIMs and are a central concept in MDA. They allow different layers of abstraction to be bridged in an automated, formal manner. In our example, a platform-specific meta-model represents a unique device, while mappings specify the transformations to and from instances of this meta-model.

4. Our Approach

Our approach consists of integrating aspect-oriented development techniques with model-driven technologies in order to gain benefits from both.

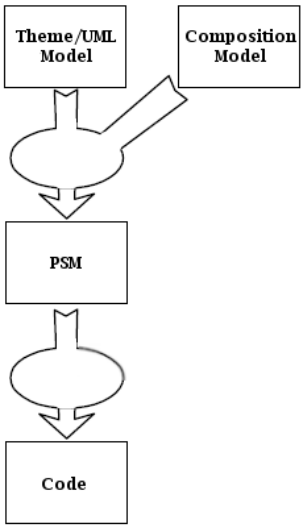

Figure 1 – Process Overview

The designer begins by modelling the pervasive application in Theme/UML (c.f. Figure 1). In this way, domain-specific context-aware concerns that exhibit crosscutting behaviour can be separated and reasoned about independently from the base application. The designer then specifies where and how these concerns are composed with each other. Using model-driven transformations, the designer gains the additional benefits of platform and technology independence. Following, we demonstrate the process with a location context module.

4.1. Modularised PIM

The acquisition, management and notification of location in a pervasive application are complex tasks [6]. Location context can be acquired by heterogeneous sources with different levels of accuracy. By acquiring and fusing together low-level context sources, a more accurate higher-level context can be inferred. Figure 2 demonstrates a small, location-based example of how a designer using our

approach can capture and reason about a contextual concern. The requirement is to ensure that the current user location is valid (e.g., the user is not deemed to be inside a wall!). The current location is checked against a list of safe locations and the concern design (or *theme*) decides whether it is safe to continue executing. If the location is unsafe, an *InvalidLocationException* is thrown. Theme/UML defines templates as placeholders for real elements in the code. In this example, the *request(..)* parameter is the placeholder for triggering the location-checking behaviour. In this way, the designer can capture and reason about location checking separately from other concerns.

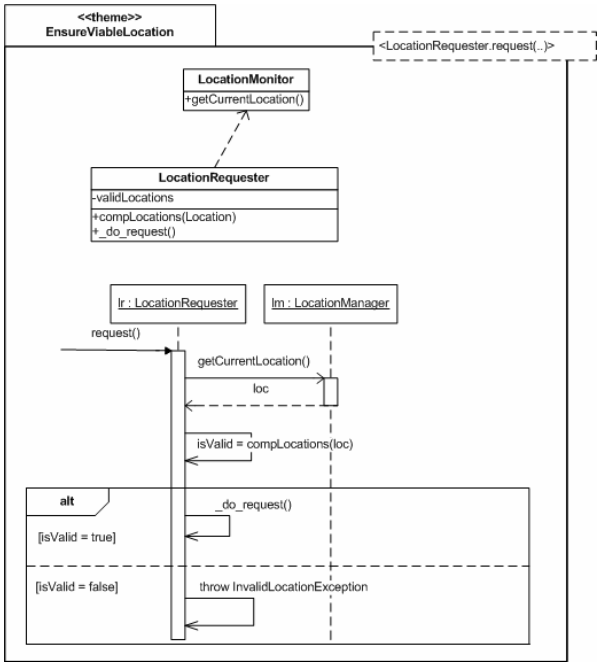

Figure 2 – Ensure Viable Location Theme

Figure 3 illustrates the specification of how location checking should be composed with the application code. The `MapViewer` class displays a map to the user showing his current location. The location is retrieved through the `LocationMonitor` class. The `MapViewer` wishes to receive only correct information about the location of the user. To ensure that the location is correct, the `EnsureViableLocation` theme is applied. To apply the crosscutting concern to the theme, a `bind` relationship is used, that specifies the appropriate operation in the base application that should be adapted with the location validation behaviour.

4.2. Implementation

The application of model-driven technologies to Theme/UML will allow us to fully realise the potential of both AOSD and MDA. Figure 4 details the architecture, including the meta-models, the mappings between them and the transformation process to be developed using the Eclipse Modelling Framework (EMF) technologies [3].

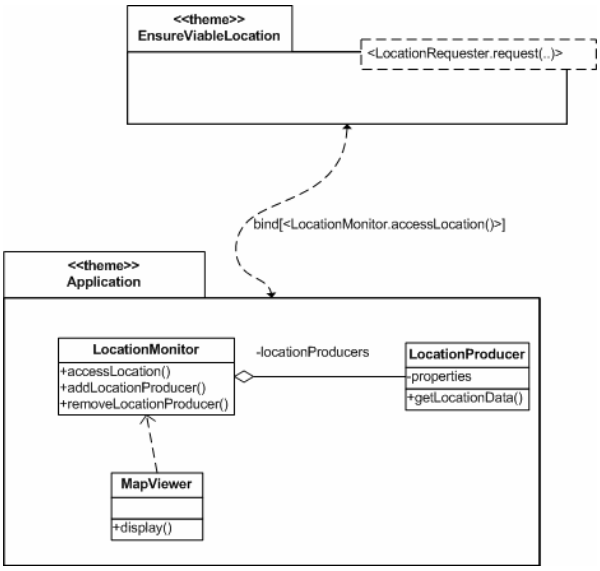

Figure 3 – Theme Composition

The EMF provides a Java/XML-based modelling and code generation framework to enable the construction of tools based on a common, structured model. Ecore is a meta-meta-model that describes the structure of any EMF-based meta-model including the UML2 library (a realisation of the UML 2.0 specification [5]). Theme/UML will profile or extend the UML 2.0 along with a meta-model defined in Ecore to represent the structure and meaning of its composition. Together, these represent the domain-specific, platform-independent meta-model that the developer uses to design a pervasive application as a PIM. The benefit of the combination of AOSD with MDD is highlighted here as the composition semantics may be kept separate from the main model, easing the transformation process. The Object Constraint Language (OCL) [16] provides application developers a means to place constraints on their models. Modellers can also use this language to query models, which is helpful in the transformation process.

A transformation specification is a set of rules that map one meta-model to another. In this case, we need to map the Theme/UML platform-independent meta-model to a platform specific meta-model. Figure 4 illustrates this approach using the platform-specific

technology, J2ME – a Java-based approach for mobile devices. Taking the PIM designed by the developer, and the transformation specification, an equivalent representation (PSM) will be produced that conforms to the platform-specific meta-model. Using the Java Emitter Templates (JET), another EMF-based technology, the PSMs can be transformed into code.

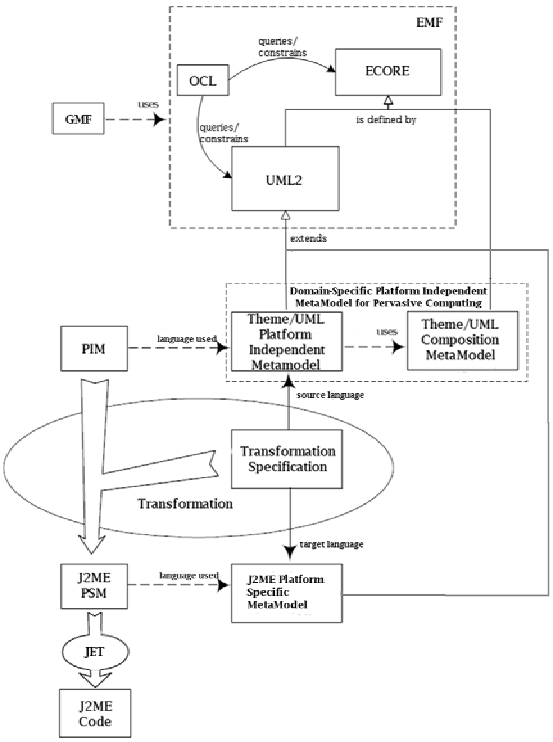

Figure 4 – Architecture

In order to provide a suitable visual representation, the Graphical Modelling Framework (GMF) will be used [10]. The GMF uses the EMF, and provides a means to develop graphical editors with domain-specific visual extensions.

5. Future Work

There are a number of possible extensions to this work once completed. A repository of mappings and platform-specific meta-models could be developed to allow designers to be able to target the required pervasive devices and technologies. The current work-in-progress is focused on one platform-specific meta-model as a proof of concept, but more would prove invaluable to the model-driven designer.

UML has a rich set of diagrams to aid the user to visualise software from many viewpoints. In particular, state diagrams have found to be useful in the aspect-oriented modelling of reactive systems in the domains of both embedded software design [15] and the telecom infrastructure software industry [13]. It would be worth investigating if these approaches could be beneficial to the mobile, context-aware domain.

The full potential of MDD has yet to be realised. The tool support and technologies are still maturing and the development of the transformation process is one of the major drawbacks. The exploration of improved methods for transformation is a worthwhile objective.

References

[1] M. Weiser. The Computer for the 21st Century. Scientific American, 265(3):94-104, September 1991.

[2] http://www.omg.org/mda/

[3] http://www.eclipse.org/emf/

[4] S. Clarke and E. Baniassad. "Aspect-Oriented Analysis and Design the Theme Approach", ISBN: 0321246748, 2005 Addison-Wesley.

[5] UML Superspec p107-115, http://www.omg.org/, 2004.

[6] J. Munnelly, S. Fritsch and S. Clarke. An Aspect-Oriented Approach to the Modularisation of Context". To appear in the Proceedings of the 5th International Conference on Pervasive Computing and Communications (PerCom), 2007.

[7] A. Schmidt, M. Beigl and H.-W. Gellerson. There's more to context than location. In Computers and Graphics, vol. 23, no. 6, pp. 893-901, 1999.

[8] http://www.eclipse.org/aspectj/

[9] D. Salber, A. Dey and G. Abowd. The context toolkit: aiding the development of context-enabled applications. In Proceedings of International Conference on Human Factors in Computing Systems (CHI),1999

[10] http://www.eclipse.org/gmf/

[11] http://java.sun.com/javame

[12] http://msdn.microsoft.com/mobility/netcf/

[13] T. Cottenier, A. van den Berg and T. Elrad. The Motorola WEAVR : Model Weaving in a Large Industrial Context. Industry Track paper to be presented at AOSD'07.

[14] A. Senart, R. Cunningham, M. Bouroche, N. O'Connor, V. Reynolds and V. Cahill. MoCoA: Customisable Middleware for Context-aware Mobile Applications in Proceedings of the 8th International Symposium o Distributed Objects and Applications (DOA'2006), pages 1722-1738, Montpellier, France, November 2006.

[15] N. Noda and T. Kishi. An Aspect-Oriented Modeling Mechanism Based on State Diagrams in Proceedings of the 9th International Workshop on Aspect-Oriented Modeling.

[16] http://www.omg.org