Component based and Model Driven Development for Mobile Product Line

Yvette E. Gelogo and Haeng-Kon Kim

School of Information Technology, Catholic University of Daegu, Korea
*yvette@cu.ac.kr, hangkon@cu.ac.kr

Abstract. There are many proposed mechanism to deal with the constraints of the implementation and reuse of the mobile application. In this paper, we tried to study the recent development in this area and formulate our own method that will enhance the mobile product line. In this study a combination of concept form Model Driven Development (MDD) and Component Based Development (CBD) is applied. We have discussed and compare (MDD) and Component Based Development. In general we can say that CBD is superior to MDD when it comes to code reuse. In this paper we also propose some solution for model reusable components and effectively classify them according to the attributes.

Keywords: MDD, CBD, Mobile Product Line

1 Introduction

The mobile device software applications play an important role in ubiquitous environment. More and more applications are needed to satisfy the needs of users. Almost every day of our lives we benefit with the use of mobile devices to make our living easier. With the rapid advancement in technology, more and more demands for ubiquitous computing services are expected. Mobile devices software also must meet the demand of the users. In this scenario, Mobile Product-line take into account to satisfy the requirements, with this, there is an easy to reuse and implement mechanisms.

There are many proposed mechanism to deal with the constraints of the implementation and reuse of the mobile application. Some of these are e.g. component libraries and domain specific languages. These mechanisms are easy to implement but introduces accidental complexity in application and are remain expensive to maintain. The requirements for mobile product-line should be easy to introduce, easy to optimize and easy to maintain.

In Software Engineering, mobile software development is one of the areas where variety of development platform is considered. Thus, a lot of development architectures are being tested, which one is better and suits to the kind of market that the software company is planning to have. Now, the latest research suggested to move into component based development (CBD), this is in connection to the requirements of Software Product Line concept. CBD architecture is being used nowadays and the
research on how to make it more efficient is the focus of this study. A component re-used is one of the most convenient ways for the fast software production. There have been many methods on how to do this and it does involve more technical and detailed view. In this paper, we tried to study the recent development in this area and formulate our own method that will enhance the software product line mobile development architecture. In this study a combination of concept form Model Driven Development (MDD) and Component Based Development (CBD) is applied.

2 Background

In this section, we discuss the involved terminologies. These are the Software Product Line, Model Driven Development (MDD) and Component Based Development.

2.1 Software Product Line [1]

The challenges in ubiquitous mobile computing is going higher and along with this, the need to build and organize functionality, seamless migration of logic among devices and different environment, devices software modeling and scalability must be meet. The common functionality can be automatically injected in particular mobile device application. Figure 1 shows the overview of the Mobile Device Application.

Fig. 1. The overview of the Mobile Device Application.

The objectives of the mobile device software product-line require the software technology to support iterative product-line development. These means that the mechanism should satisfy the following: (a) easy to introduce and maintain (low cots)
Component based and Model Driven Development for Mobile Product Line

(b) flexible to support the design experimentation (c) clear separation of common functionality among application.

2.2 Model Driven Development (MDD) [2]

Model Driven Development (MDD) is a software development approach based on the design and the transformation of models. In MDD, models are systematically translated to other models and to a source code. Model transformation plays a key role in MDD [3].

Several model transformation languages have been launched lately, aiming to facilitate the translation of input models to output models. The employment of such languages in practical contexts has succeed, although quite often those languages cannot be directly applied to a particular type of model transformation, called refinement. Figure 2 is the overview of Model Driven Development Transformation.

![Fig. 2. Overview of Model Driven Development Transformation](image)

2.3 Component Based Development

Component-Based Development claims to offer a radically new approach to the design, construction, implementation and evolution of software applications. Software applications are assembled from components from a variety of sources; the components themselves may be written in several different programming languages and run on several different platforms.
2.4 Mobile Development Architecture

Mobile application development is the process of developing a software application for mobile phones and other handheld devices such as PDAs. These applications can be pre-installed on phones during manufacturing, downloaded by customers from various mobile software distribution platforms, or delivered as web applications using server-side or client-side processing. Application software developers also have to consider a lengthy array of screen sizes, hardware specifications and configurations because of intense competition in mobile software and changes within each of the platforms [4] [6].

![Diagram of Mobile Development Architecture](image)

Fig. 3. Overview of Mobile Development Architecture

3. Challenges

In the previous section we discussed the two architectures applicable for Mobile product-line. Model Driven Development (MDD) and Component Based Development (CBD) have the same purpose but different in the process in the way it develops the mobile application.

There have been a lot of challenges in software product-line. These challenges are
need to be overcome by providing effective mechanisms. The first challenge is to find out what is the better architecture for the mobile product-line. It is important to know what are the advantages and disadvantages of both MDD and CBD. There is a need to find out if these two architectures can be integrated and would come up with the better mechanism. The next challenge is to meet the objectives of the mobile product-line and that is to introduce a mechanism that is able organize common domain functionality in mobile product line that enables reuse of the domain functionality in mobile application. A structured way to interpret attributes modularized attribute-driven transformation and the use of container abstraction to organize the domain assets of mobile application product line.

4 Propose Solution

There are many variability mechanisms that have been proposed over the years to deal with the organization of the common domain functionality in a product-line. Some of these mechanisms, e.g., component libraries, are easy to implement, but introduce a lot of accidental complexity in an application. Other variability mechanisms, such as, code generation based on visual models, support a more abstract representation of a domain, but do not preserve the architecture of the domain abstractions in the application code. The variability mechanisms for mobile product-lines should be declarative, easy to introduce, and enable various domains-specific optimizations [5]. Based on the challenges that we have discussed on the previous section, we tried to find the solution for each challenges.

4.1 Comparison of CBD and MDD

In this section we briefly discuss the comparison between CBD and MDD [5] in terms of Design and Architecture Evaluation, Design Reuse, Required Tools and Technology and Code Reuse.

<table>
<thead>
<tr>
<th>Category</th>
<th>CBD</th>
<th>MDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design and Architecture Evaluation</td>
<td>Evaluation is an important part of component based design because it is consist of the needed components which are based on selection. It can be considered that the design and architecture of components based development is a result of evaluation itself.</td>
<td>MDD provides the evaluation of some models based on existing theoretical views. As an example an Entity-Relationship model can be evaluated by checking if it complies with some of the relational data base normal forms</td>
</tr>
</tbody>
</table>
Design Reuse

CBD is focused on component reuse. In order to reuse a component, component design must be done first. If architectural design is reused, a greater components reuse can be achieved. But unfortunately, many components based technologies define their own formalities for architecture; hence it is not easy to do a straightforward reuse for both design components.

Required Tools and Technology

The only kind of tool needed - component framework. Currently CBD tools are considerably mature, though some theoretical achievements have not been widely incorporated yet. Vendor lock-in can be avoided by using implementations of public component model specifications.

Code Reuse

The major purpose of CBD is code reuse. However, following a CBD approach does not mean developing without coding. Components that are business critical or unique to a specific system are typically developed by the system developers. Also the quality or the price of existing components that provide needed functionality may drive for custom development of alternatives.

MDD is not focused on source code reuse per se. The central notion in MDD is the model and thus it is the main goal for reuse. Models can be successfully reused across multiple systems. As a side effect this may lead to reuse of the code correspondent to these models.

The central items in MDD are the models and thus reusing them can be very beneficial. Obviously, if models can successfully be reused then architecture or a design can also be reused since in MDD these are represented as sets of interrelated models. A major problem before the reuse of models is the abundance of modeling formalisms. A model defined in a given formalism is hard to be reused in another system which uses others.
4.2 Reuse of the domain functionality in mobile application

Figure 5 shows the proposed method on how to model reusable components and effectively classify them according to the attributes. The concept of this study is to refine the model, transform, classify them and finally put in container software according to its classification and common attributes. In this way, it is easy to map them and get the right component for reuse.

Fig. 4. Overview of Mobile Development Architecture

a. Model Refinement
This study comprises a PIM-into-PSM model transformation and refers to an endogenous refinement of models based on the UML meta-model. In a model refinement most elements from the source model (PIM) are copied to the target model (PSM), while other elements must be changed in order to incorporate platform-specific aspects. [2].

b. Components
Software engineers regard components as part of the starting platform for service-orientation. Components play this role, for example, in web services, and more recently, in service-oriented architectures (SOA), whereby a component is converted by the web service into a service and subsequently inherits further characteristics beyond that of an ordinary component.

c. Transformation and Classification
Model transformation plays a key role in MDD. Several model transformation languages have been launched lately, aiming to facilitate the translation of input models to output models. The employment of such languages in practical contexts has succeed, although quite often those languages cannot be directly applied to a particular type of model transformation, called refinement.

d. Container
Container is where the transformed components should be stored after transformation. A container is an architectural abstraction that can be used to organize a product-line development process.
for a domain, by factoring out the common domain behavior into a set of services provided by the container.

5 Conclusion

In this paper we discussed the issues in Software Product Line and several related backgrounds. We have discussed and compared the Model Driven Development (MDD) and Component Based Development (CBD). In general we can say that CBD is superior to MDD when it comes to code reuse. It is identified that typically using MDD results in more traceable, maintainable and resilient to changes systems than using CBD. Both have strengths and weakness. But when it comes to the consideration of the Product Line objectives, that is the reuse of the code or components, it is favorable to use the CBD. In this paper we also propose some solution for model reusable components and effectively classify them according to the attributes.

References

3. Andrew Carton, Cormac Driver, Andrew Jackson and Siobhan Clarke, “Model-Driven Theme/UML”, LNCS, Springer, 10.1007/978-3-642-03764-1_7, ISSN: 0302-9743