SOFTWARE ARCHITECTURE AND PRODUCT LINES FOR MOBILE APPLICATIONS

Tandon Surbhi
Department of Computer Science
Illinois Institute Of Technology
standon3@hawk.iit.edu
+1 (312) 478-0239

Poste Prateek
Department of Computer Science
Illinois Institute Of Technology
pposte@hawk.iit.edu
+1 (312) 478-0837

Nathani Chanki
Department of Computer Science
Illinois Institute Of Technology
cnathan1@hawk.iit.edu
+1 (312) 478-0832

Arora Nikhil
Department of Computer Science
Illinois Institute Of Technology
narora2@hawk.iit.edu
+1 (312) 532-8420

Abstract

The paper provides an overview of software engineering issues related to software architecture and product line of mobile application. Mobile application will generally be structured as a multi-layered application consisting of user experience, business, and data layers. The purpose of this software architecture is to ensure that an application is up to the job it is designed for and meets the requirements that are common to all good software: 1) Ease of use; 2) Quality; 3) Performance; and 4) Security. The structural elements, interfaces, behavior, functionality have a considerable impact on the overall user experience, usability and success of the application. Software product lines, or software product line development, refers to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. The process of creating software systems from existing software rather than building them from scratch.

Keywords

Index Terms— Development, Framework, SPL (Software Product Line) Mobile, Object oriented SPL (OOSPL), Model Driven SPL, MDE (Model Driven Engineering), DSML (Domain Specific modeling language).

Introduction

There has been exponential growth in Mobile Application development where you will find a wide range of mobile software applications to run. If we look this in the perception of technology, mobility emerges the computing infrastructure paradigm is changing from homogenous computers to heterogeneous, resource-constrained grasp devices.

Reusability of software has been one of the major ambitions of the software engineering industry (Software product lines). Those who achieved component-based software development assumes a market divided into module developers, component users and an app store. Instead, this comes to be most ambitious for most types of application. To answer that, there has been a change from world- wide reuse of modules to company-wide reuse. Apart from this, the importance of an explicit design and representation of the architecture of a software system has become incrementally recognized. The combination of this leads to the definition of software product lines. A software product line consists of product line architecture, a set of reusable components and a set of products derived from the shared assets. This can be better explained by understanding the example of reusability of template in different products. Let’s say we build an application

that consists a login and log out feature including the sessions of web module. There we have written a piece of code and any other application related to that feature can share a same family of artifact in that organization. Hence, we can apply the software product lines in this type of scenario.
A recent survey revealed that developers are facing problem while creating a family of an app i.e. one app for different models of phones, which support compatible hardware and configurations, and such incompatibility is in network, memory, and device resolution, processing power, CPU, camera, sensors and storage capacity. Therefore, that app must be having the features according to the software and hardware compatibility to match the characteristics of each device.

A product-line architecture (PLA) is a blue-print for creating families of related applications. PLAs acknowledge the fact that developers build artifacts of families of closely related products that can be used to embed in different mobile device configurations. So to restrained cost of software prints PLAs are used.

A Software Product lines is a set of software-intensive systems that is used to share a common, managed set of features to reuse the specific needs of a particular family of applications from set of codes for same family in a prescribed way.

Therefore this paper enlightens the following contributions:

1. An analysis of recent mobile application software engineering area.
2. An explanation of the issues or less materialistic related mobile application software engineering research.

This paper can make better understand the analysis in current domain and the open issues for Software architecture and product lines in mobile application run.

Different Types of mobile platform environments

There are different types of programming environments available for mobile applications: iOS for Apple, Android Framework for android phones, Windows Phone, Blackberry, Symbian etc. To develop android applications we need android studio or adt plugin for eclipse. To develop iOS apps we need a mac computer running OS X mountain lion or later, Xcode package and iOS sdk. For windows phone, an application can be developed using different programming languages such as C++ with XAML or C# or C++ with DirectX.

Basic Requirements

[1] Mobile application has some extra requirements which are not commonly found in existing software applications these requirements are defined below:

1. Power consumption - It is the one of the major requirement for building mobile applications as cellphone as restricted to power usage.
2. User Interface - There must exist an equal level between securing the data and user friendliness. For example asking complex password every time from the user makes the user experience awful. So the interface must be attractive and simple.
3. Hardware and Software Platform Families: mobile application must be programmed in such a way that once it is written it must run on all the versions of supporting operating system.

4. Sensor Handling: now a day's majority of the cellphone contains different types of sensors such as proximity, accelerometer, gyroscope, gps, barometer etc. application must be able to make use of such type of sensors wherever applicable.
5. Potential interaction with other applications.

Approach Followed

The approach followed for mobile application development is an agile approach known as scrum approach. It is an iterative and incremental approach for managing product development. It defines "a flexible product development strategy where a development team works as a unit to reach a common goal".
[8]Scrum adopts an empirical approach—focuses on maximizing the team's ability to deliver quickly and respond to important requirements. In Scrum, projects are divided into sprints, which are typically one to three weeks in duration. At the end of each sprint, stakeholders and team members meet to see the progress of the project. The next steps of the project are also discussed in the sprint meeting. There are three roles of scrum, namely: Product Owner, Scrum Master, and team member. The Product owner represents the customer's interests through requirements. The Product owner has the authority of the three roles. She/he is the single individual who must face the situation when a project goes awry. The Scrum Master works to remove any impediments that are obstructing the team from achieving its sprint goals. The team is responsible for completing work. The teams consist of seven cross- functional members, plus or minus two individuals. The team is responsible for determining how it will accomplish the work to be completed. This grants teams a great deal of autonomy, but, similar to the Product Owner's situation, that freedom is accompanied by a responsibility to meet the goals of the sprint.

Research Approach for Mobile Software Engineering

The purpose of the research is to raise the understanding of current trends in mobile application engineering research and focusing on the differences between mobile devices and software applications.

1. The User-Experience: operating a mobile device is different from operating a desktop application. While desktop has windows, menus, icons, pointers etc to interact with the users. This is totally different in case of mobile applications. Mobile application has widgets, touch, gestures. Also they have different types of sensors to get the user input or provide user an output. They increase the interaction with the user. Also cellphones has small screens which requires lot of effort to adjust everything into that small screen.

2. Non-functional Requirements: it covers how an application will be designed such that it consumes less battery and makes maximum usage of allocated resources. Also it raises question how will the application respond to particular situation. For a web based application a light version of webpage is made visible if the internet speed is slow.

3. Processes, Tools and Architecture: as explained earlier user experience plays a crucial role for success of an app. Different prototypes of user interface of an app must be created if it supports multiple devices.

4. Portability/Code Reusability: In today's world there are five major mobile platforms available (android, iOS, windows phone, Symbian, Blackberry) and developing an app for a single platform will be expensive. So an app must be written in such a way that majority of the code must be reused to develop applications for other platforms.

Outline of Mobile Architecture

There are three main components in mobile architecture that we are going to use for classification which are as follows

1. Existing System
2. Middleware Application
3. Handheld Applications

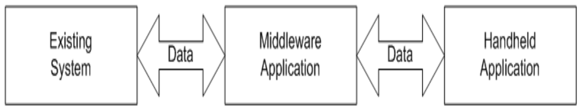

Figure1: Basic Mobile Architecture

Above figure has three main components, middleware application is used for applying the business logic, data transformation. This can act as a central point of communication for mobile devices. If the business system is started from the scratch or it is re-implemented then there is no need for middleware.
We can try to build a universal logic which can be integrated into system to handle communication start to different types of devices. In most of the cases, systems are not written most often because it is not economically acceptable to rewrite them for mobile devices.

Different Challenges faced in Mobile Architecture

There are different challenges which are faced in most of real time projects. Some of them are explained below

1. **Device Configuration**: It is the responsibility of the driver to take care of the hardware configuration and drivers of the latest mobile device for creating an application. There are chances that device may have different device configurations and can have different device drivers which are propriety of it. This is important at the time of app creation.

2. **Upgrades and Software Development**: if we are deploying a software which is compatible for different devices will be a great challenge. There are different software package which deals with the software configuration and device configuration. There are particular packages which must be written for management applications which explains the software and configuration file to initiate. An application must carry a mechanism for automatic updating if there is no management package specified.

3. **Performance**: Now a day's performance is not a major issue in normal day computing. But it is not normal in case of handheld held devices. With complex user interfaces, small screen, CPU processing intensive algorithm and data processing makes an application running slower thus reducing the performance.

4. **Memory Management**: Now a day's memory is available at very cheaper price. Many of the mobile devices are coming with fixed storage capacity. They do not have the ability to upgrade. Hardware choice will be restricted if there is high requirement of memory in a device. The optimal way of storing data is necessary and interface must be designed in such a way that they occupy less memory based in accordance to the particular mobile device.

5. **Security**: Security is most important concern in many systems. In fact, in mobile devices it is one of the major challenge faced by developers as mobile storage is full of confidential/personal storage. Proper authentication and security measures must be taken to provide one safer access to the data.

6. **Interfacing Dissimilar Technologies**: Most of the mobile development includes the involvement of different type of technologies because of different types of platform limitations and constraints. To make connection with different web technologies, one can make use of different services such HTTP, TCP/IP, sockets to overcome such problems.

7. **User Interface**: making a user interface graphical is a tough task as mobile devices have small screens and which sometimes is frustrating for some users while entering data. An application must be made interactive by using different objects like list, checkboxes and radio buttons to avoid text entry from users.

Cellular Phone Architecture

This system was developed for logistics companies. Truck drivers carry cellular phones that have the mobile application running on them. This application keeps the drivers of new jobs informed, general messaging and allows entry of pickups and deliveries. This architecture addresses the following high-level requirements:

1. Message delivery should be guaranteed in this system.
2. When we are out of network the mobile application should allow data entry.
3. When the mobile device is out of network area then the system should be able to store messages.

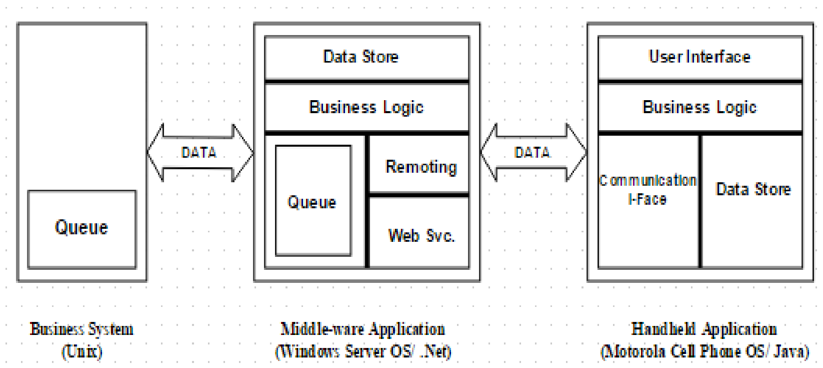

Figure 2: cellular phone architecture

Business System

The business system runs a big logistics application on IBM hardware. This application is a customized package application that uses DB2 as the backend database. Messaging to this system must occur via MQ-Series.

Middleware Application

The middleware application uses Windows service for configuring the remoting infrastructure. Web service used by the mobile application accesses the application's business classes via the remoting infrastructure. The middleware is responsible for:

1. Message is received from mobile application.
2. Combining messages for a mobile device into a single message using a message envelope.
3. Sending and receiving messages from the business system.
4. Processing all the messages from the mobile application in order.
5. Creating a message envelope that contains all messages for delivery to a mobile device.
6. Storage of messages from the business system for a device until that device gets connected.

Due to restrictions in the cellular provider's network, the middleware cannot "push" any messages to a device. A web service was chosen for receiving messages from the devices because of this reason, and also as the mobile application is written in Java programming language. To enable the web service to pass the message packet from the mobile application to business layer remoting is used. The business layer will be processing the message and it returns a message packet to the device via remoting and web service always.

Handheld Application

The Java 2 Mobile Edition (J2ME) technology was used in the development of the mobile application. This application is responsible for:

1. When the user interface is not in use, it should always run.
2. Sending messages to middleware with occasional "ping" messages.
3. Re-sending any messages when exceptions or errors occur. When exception or error occur, it should resend any message.
4. After the message is received it should automatically notify the user.
5. As needed store the application data on device.
6. The mobile application is multithreaded. First thread runs to send and receive messages always and another thread runs the user interface when invoked. For data storage J2ME record management facilities are used.

Software Product Guidelines:

[6] Software product line engineering helps in reuse by developing many applications that share a common set of requirements yet it differ among each other according to a set of variable requirements. Mobile applications are software programs for mobile device operating systems (such as Android, Blackberry OS, iOS, or Windows Phone OS) that can collect, use, and transfer users personal information from a mobile device. In terms of the technology, mobile environment is changing the global computing infrastructure from static computers to dynamic, resource-constrained handheld devices. Mobile computing imposes several restrictions to software development because of diversities in network connectivity, platform capability and resource availability.
Product Lines Architecture is software architecture for supporting a whole product family, it reflects similarities as well as variabilities among the various products. Product-line variant is created by including re-used device-specific software parts.

For Software product line, Domain analysis is substitute technology to requirements analysis. By making Domain analysis of software products, we decided to create a common architecture and for making this architecture we need to study about common and variable features in the requirements.

Organizational Benefits: The organizations that we know have achieved remarkable benefits because of using software product line approach which also provides options to future market opportunities.
Product lines enhance quality of the product. Each new system takes advantage of all the defect elimination in its forebears; developer and customer confidence both rise with each new instantiation. The more complicated the system, the higher the payoff for solving the vexing performance, distribution, reliability, and other engineering issues once for the entire family.

1. Large-scale productivity gains
2. Increased product quality
3. Decreased product risk
4. Increased market agility
5. Increased customer satisfaction
6. More efficient use of human resources

Example of software product line:
[4] Many mobile companies like Nokia, Samsung, Motorola, apple, blackberry develop a wide variety of mobile phones and maintain them. So, these companies apply product line approach to their mobile phones and its software development.
The software product line consists of two levels:-

1. **The top level platform:** It is developed and maintained by a top-level infrastructure group, and consists of a product line architecture and a set of components, that are shared by all mobile phone products and ported to different hardware platforms.

2. **A specialized groups** exist that develop, especially, components specific for the members in the subset. These domain engineering units have frequent contact and exchange considerable amount of information, but are organized as independent units.

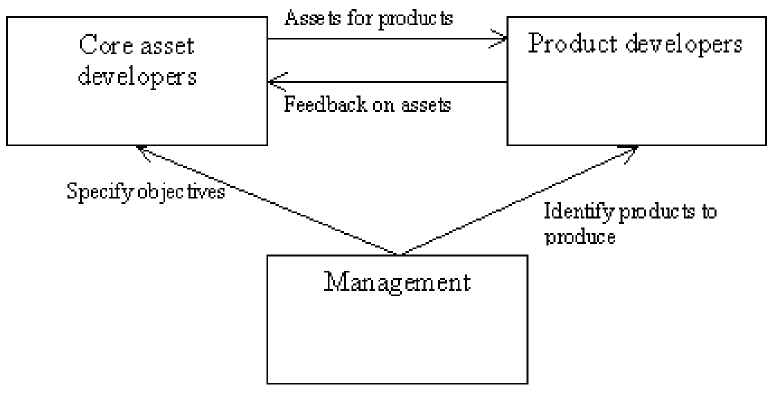

Figure 3: SPL overview

Here, we are describing 2 kinds of Product lines which are suitable in mobile application development

1. Model Driven SPL.
2. Aspect Oriented SPL.

Model Driven Software Product Line: Model-driven software product lines are combination of both abstraction capability of Model Driven Software Development and the variability management capability of Software Product Line Engineering. MDE has been used by many organizations to effectively manage software product lines. An entire software product line can be expressed and created from a single configurable model. Models is used show both the common and different parts of the SPL design and they are the key design which are used to implement artifacts through the application life cycle. Using MDE technology, SPLs can be planned, specified, processed, and maintained on a higher abstraction level.
It is a design approach that explain the essential characteristics of a problem in a way that is isolated from particular solution space details such as dependence on particular programming language, OS, middleware.
MDE minimize gap between problem and solution domain by combining below components:

1. Models and meta-modelling interpreters to create domain-specific modeling languages (DSMLs)
2. CVA and object-oriented extensibility capabilities to create domain-specific component frameworks.

Domain-Specific Modeling helps to raise the abstraction level beyond coding by specifying programs directly using domain concepts. The most important reason to use DSM is simply the increased productivity. It raises the level of abstraction and shortens the gap between the problem and the solution space of software-intensive systems by applying the following techniques:

1. **Domain-specific Modeling Languages (DSMLs):** It facilitate the model-based design, development, and analysis of vertical application domains, such as industrial process control, telecommunications, and avionics mission computing. It includes model interpreters and meta models. Model interpreters follow a procedure to read, traverse, and analyze the models and to create the executable system which are based on these models.

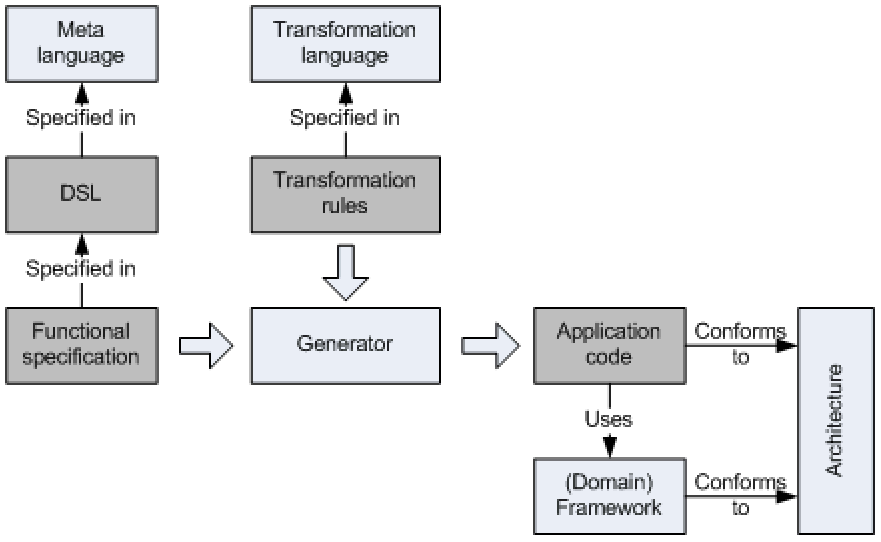

Figure 4: Model Driven Approach

2. Domain-specific Component Frameworks (DSCF)–DSCF is composed of a component model and the tool support which permit assembling, deploying and executing demanded applications . Moreover, such component model defines the relevant architectural concepts, called domain-specific concepts, according to the requirements of the targeted application domain.

[7] Challenges with model-driven software product-line architectures

1. **Capturing the new requirements into existing MDD based software product lines for DRE systems:** MDD helps to improve productivity of mobile applications, still it is hard to use new requirements.

2. **Migrating the existing domain Model with Model driven development with MDD based PLA Evolution:** If the Meta model is changed, all models that were defined using that Meta model may require maintenance to adapt to the semantics that represent the computer based system correctly. Without ensuring the correctness of the domain models after a change to the domain, the benefits of MDD will be lost.

Aspect Oriented SPL

[10] Aspect-orientated programming is a paradigm for programming concerned with bifurcation of crosscutting concerns and converting them into non-crosscutting ones. Cross-cutting concerns does this by putting additional behavior without modifying the code itself, rather defines which part of code or code is modified via a “**pointcut”** specification, such as "log all function calls when the function's name begins with 'set'". It uses the modified Propagation Probability to assess the software architecture and the quality of design. AOP is used to obtain the better changeability and **modularity** of product lines than the old techniques.

Further explaining the cross-cutting term with the help of a diagram as shown below:
There are 3 classes which describe a city i.e. University, Hotel and Amusement park. All the classes have defined their own methods used inter-class and the object-oriented works correct for this case. But if we want a notification whenever an update occurs in the system. It will comprise of all the update methods present in each of the classes. This will keep track of inter-class methods and hence, this express well what cross-cutting concerns means.

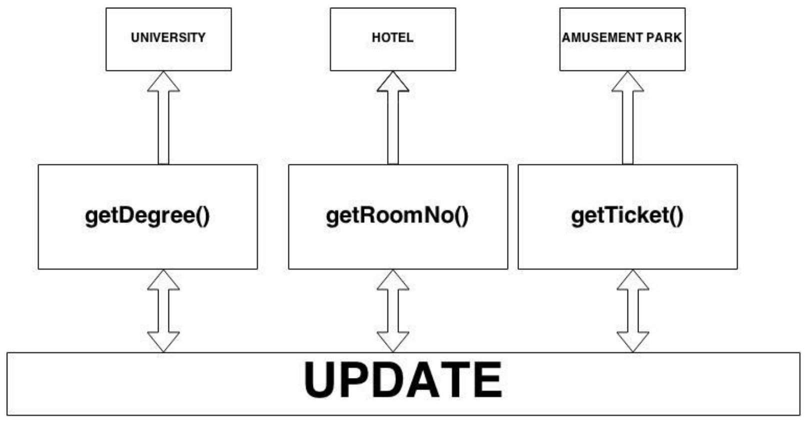

Figure 5: Aspect Oriented SPL

Conclusion

In order to get advantages from two of the SPL categories MDE and AOP must be combined together. Different artifacts can be expressed as model. AOP overcomes the drawback of Model driven architecture that is we need to define different domain models. Aspects can be easily added to increase the characteristics of object used in object oriented approach. After joining these models we can get the best out of the model.
For majority of mobile applications software development process is well known but still there exists some issues which requires in depth research and analysis. With the advancement in mobile application development, much is known about how to build mobile application. Developers are well aware about what applications are of daily use, the usage of application and user's experience.
The paper explains in brief different challenges faced for mobile application software engineering which are being faced at present: designing user interface, making application context aware, balancing agility. This review was a step to show some of the challenges identified during research in context to mobile application software engineering.
One of the most challenging factor was dealing with the multiple platform as there are different mobile platforms available and when developers try to develop an application, they treat every platform separate entity. Testing is done mainly to ensure the functional correctness. Also testing an app is another major concern because we need to perform manual testing as most of the tool available are still weak.

References

[1] Software Engineering Issues for Mobile Application Development by Anthony I. Wasserman

[2] https://riunet.upv.es/bitstream/handle/10251/15075/tesisUPV3762.pdf?sequence=1

[3] http://gp.uwaterloo.ca/sites/default/files/2005-czarnecki-model-driven-software-product-lines.pdf

[4] Example of Software Product line in mobile companies http://www.methodsandtools.com/archive/archive.php?id=49

[5] http://www.dei.isep.ipp.pt/~alex/publico/025_Product_Line_Case_UML_Pulse_2004_GoPhone.pdf

[6] Software Product Line https://www.sei.cmu.edu/productlines/start/

[7] Challenges in model driven http://www1.cse.wustl.edu/~schmidt/PDF/MDD-PLA.pdf

[8] http://scrummethodology.com/

[9] The Role of Aspect-Oriented Programming in OMG's Model-Driven Architecture http://polyglotprogramming.com/papers/AOP%20and%20MDA.pdf

[10] Aspect oriented SPL http://en.wikipedia.org/wiki/Aspect-oriented_programming