Software Product Line Libraries for Mobile Based Applications

1. Dutta, Raj Sekhar (A20380630)
2. Kulpe, Mohit (A20372614)
3. Mallu, Sowjanya (A20385564)
4. Pimpalgaonkar, Abhinav (A20387324)
5. Sistla, Chaitanya (A20373561)

ABSTRACT

Mobile Applications are rapidly emerging as a convenient medium for using a variety of services. Over time and with the high penetration of smart phones in society, self-adaptation has become an essential capability requirement for the mobile application users. Software Product Lines (SPL), creates many similar products from a single set of core assets and that assets are reused. The design of a mobile application is a tedious task owing to its intrinsic variability. Software designers need to take into account in the development process the versatility of available platforms (e.g. Android, iPhone, tablets). There are variety of existing devices with divergence introducing a further layer of complexity to the development process. In addition, at runtime, many potential situations have to be considered.

Keywords – Software Product Lines (SPL), Software Product Line Libraries, Mobile Applications, Mobile computing, Architectural Assets

I. INTRODUCTION

What is Software Product Line?

Software Product line is a designing method for making an arrangement of comparable programming frameworks from a common arrangement of programming resources. Programming items are created utilizing a similar application and information designs utilizing a typical center of reusable programming segments. A product offering offers an arrangement of benefits that incorporate prerequisites, engineering, plan designs, reusable segments, test cases, and other work items. A product offering permit in the advancement of numerous items that are built by profiting by the shared characteristic among all items inside the product offering. Inspirations for Programming Product offering are as underneath

1. Mobile Applications now-a-days are more complex and performs high level task
2. Software Development from scratch for each differences is purely uneconomical
3. Reuse is becoming imperative.

Software Line Product Libraries

Software product line assume a key part in programming reuse. Center Resource Library development is a noteworthy assignment on software product line. Software line Resource comprises of part instruments and related documentation. All advantages are overseen by core asset library.

Core Asset Library

Core asset library deals with all product resources. It comprises of software resources comprises of parts, devices and related documentation. It additionally contains devices identifying with resources library incorporate advancement apparatuses for segment combination, core asset library administration instruments, test devices. Parts of Core asset Library are as below:

1. The core asset library system comprises core asset, core asset management tool, code component development tool and code component test tool.
2. Core Asset Development Tool:
3. The development staff develops a new component on the basis of requirements through develop tools for component integration.

Contribution of libraries to the Market

Mobile applications are software applications developed for mobile devices, such as smart phones and tablets. Applications are available via online distribution channels called app stores, such as the Apple App Store, and Google Play Store. Across these stores, the mobile app market grew from $18 billion in 2012 to $26 billion of revenue in 2013. However, approximately 91% of the apps in such app stores are free-to-download. Hence, many app developers use non-traditional revenue models to legalize their apps. One of the most popular revenue models is the advertisement model, where ads are displayed within an app, and developers earn money every time users click on such ads. To serve ads, developers must embed specialized libraries – ad libraries – in their apps. Previously from the studies it is estimated that around 51–60% of free Android apps have at least one ad or analytics library. Gartner projects that the revenue from mobile ads (both mobile apps and websites) will reach $18 billion in 2014. App Annie and IDC found that revenue from in app advertising has gone up by 56% in 2013. Given the highly competitive nature of the ad industry, add libraries are updated on a regular basis to include new ad serving models, or for legal and financial reasons (e.g., merging of advertising companies). In that case, app developers need to decide if they wish to integrate the updated libraries into their app. While an ad library would not usually affect the core functionality of an app (hence most library updates can be safely ignored), such an update to an ad library might have a big impact on the revenue of an app. However, releasing a new version of the app primarily because of an updated ad library could be a hard sell (figuratively) to existing users. It costs money to download the new update and users also need to worry about whether it would be safe to update to the new version, i.e., does it break existing behavior or corrupt existing data? . Software definition or requirements analysis, top-level design, detailed design, coding, unit testing, integration, and delivery were the

sequential stages a project followed. Activities of each stage were generally completed before the subsequent stage was begun. The primary results of each stage were documents and formal reviews, particularly early in the life cycle. Design focused on the stepwise refinement of subsystems into successively smaller components ending with source-code units. Code units were integrated once all units were implemented—late in the life cycle. The result was late discovery of significant problems leading to a scenario of late delivery, cost overruns, reduced functionality, reduced quality, and dissatisfied customers. In the early 1980s, Celsius Tech developed new systems by re-using developers who modified existing systems. Previous structures formed the basis of a new system, developers used code from previous systems with modifications as needed. In time, a new system would inherit structures, design decisions and constraints, and code, potentially from several parent systems. While this approach was conceptually similar to a product line approach, there are important differences. Each working system was designed and implemented without consideration of future uses and needs. With each successive descendant, a potentially increasing amount of the parent system required redevelopment. Maintenance was needed for each individual system rather than being leveraged across all scions. Reuse was ad hoc and purely opportunistic. Although not product line development, this early development strategy laid the foundation for the subsequent product line paradigm.

II. SOFTWARE PRODUCT LINES ESSENTIALS

The product lines includes three process that are Core Asset Development, Product Development and Management. Center resources are utilized to construct new items, or Core resources are removed utilizing existing items. Figure 1.1 gives insights about these three angles.

Each rotating circle speaks to one fundamental movement. The endless circle bolts indicates how they are associated with each other in constant way. It speaks to that the request of exercises is irregular request (remove) and not successive. The turning bolts not just shows that the center resources are utilized to build up the new items additionally reexamine the current core assets.

The figure 3.1 gives unbiased thought of which part is executed first. Now and again existing items are mined to get the non-specific resources like necessity determination, a design or programming parts. These bland resources are then moved to the product offering base resources.

On advancement of the new items the core assets are adjusted. After the changes the core assets are followed and the yields are offered back to core assets advancement action. The result is, core assets are made more non-specific in view of the new items.

The Three Essential Activities for Software Product Lines

A.Core Asset Development

Fundamental point of center resource improvement is the foundation of creation abilities for the recently created items. The generation action of core assets can change its specific circumstance. Creation limitation, for example, fast amassing of items may confine the yield that is the product offering design. This confinements highlights the officially existing resources which can be reused or utilized for mining. Just within the sight of predefined imperatives and assets the Core asset advancement is effectively accomplished. Four essential requirements components for core asset improvement are as below:

Production Constraints

Taking after are a few inquiries which should be offered an explanation to choose whether to put time and cash in programmed code and setup general condition or depend on the manual code work. The inquiries to be addressed are – What must be the length to get the new item the market, a month or a year? What organization related guidelines must be taken after for the item improvement? Will's identity proprietor of the item module improvement and what condition setup is utilized? Answers to these inquiries will prompt a choice about what contrasts to be made in center resources and what advancement process will be utilized for improvement of item.

Product Constraints

It contains below inquiries what comparative thing and contrasts exist in the product offering? What highlights identified with conduct they give? What models in contrast with military or business are connected to the items? What limit of execution they should give? These limitations might be mined from existing items which will shape premise of product offering, or they might be made recently or both approach could be taken after. Creation Strategy: What approach of product offering building will be utilized? Proactive or responsive (proactive is set of center resource from which item is created or receptive means summing up the center resources from examination of existing item). Will the general segments will be obtained from market or will it be manufactured inside? The creation system gives the cause of engineering and in addition its parts and the guide of its development.

Pre-existing Assets

The segments of the product line engineering can be used from some very much planned inheritance frameworks. The genuine question here is are there any current calculations or libraries or parts that can be reused from the inheritance framework. Officially existing resources are those by inheritance product offering, as well as the ones which are outside like web administrations, open source items and etc. The output of core asset development phase is explained below:

Product Line Scope

It is the list of the items and its likenesses and contrasts. This in turn incorporates execution they give or the components they have. For product offering achievement this degree is extremely vital. On the off chance that the degree is wide then center resources can't oblige the varieties, creation economies will be lost. Then again if the degree is too little then it can't fabricate more blend core assets if future development is considered which will stop the product offering development. Product offering extension advances with change in economic situations, or change in association arrange.

Core asset base

In the product line all the core assets utilized for building up the item is incorporated into the core asset base. It's not compulsory that all the core asset will be utilized as a part of the considerable number of items. Any ongoing execution models, design assessment comes about, prerequisite determinations, space models, web administrations and COTS (business off the rack programming) all constitute to the core asset base. Core assets should have the procedure joined to it which clarifies how it is utilized as a part of real item improvement. It ought to incorporate below steps:

- Product line requirements to be used as baseline.
- Variation requirement to be specified
- Additional requirements other than product line requirements to be added.
- Verify whether the additional components and variations can be incorporated by the architecture.
- Production plan: It depicts the arrangement how the core assets are used for building up the items. It has two primary parts as takes after. 1) Production process: It incorporates ventures of procedure to be utilized for building the item. The correct procedure to be utilized that gives the varieties in the items is portrayed in this creation process.2) It clarifies the points of interest of venture which helps in process execution and administration. The subtle elements clarified resemble measurements, materials bills and the timetable.

B.Product Development phase

This action is subject to the 3 yields of the Core Asset Development – Product line scope, center resource base and the creation arrange. This stage additionally relies on upon the item depiction of every item. The info steps are clarified to sum things up: The product offering degree shows if the item is legitimate to be considered in the product line.

- The core asset base is having all assets that is to be used for product development.
- The production plan provides the process of using the core assets to build the project.
- The Product Description provides the unique details of that product which varies from the generic product description.

Production plan and the core assets are used to develop the product by the product builders. It is also the moral imperative of the product builders to provide good or bad feedback on any problems or issues in the core assets so that core asset base remains updated and without any deficiencies.

C. Management Phase

Administration at both the specialized and association level is very required for the product line. Undertakings must be given assets for its execution and after that it ought to be administered for effective consummation inside indicated day and age. Association administration ought to make legitimate structure which is endorsed by the endeavor and ought to ensure that every association unit get enough assets and in addition redress assets. Authoritative administration is viewed as a definitive component for the accomplishment of the product line. A subsidizing model is produced by the hierarchical administration that guarantees core assets development and deals with the assets appropriately. Authoritative administration likewise sorts out the specialized exercises in vital exercises of core assets and item improvement. It mitigates the hazard that turns into the barricade in accomplishment of the product line. The product line create great association with the association clients and additionally providers, which is to be kept up productively by the administration.

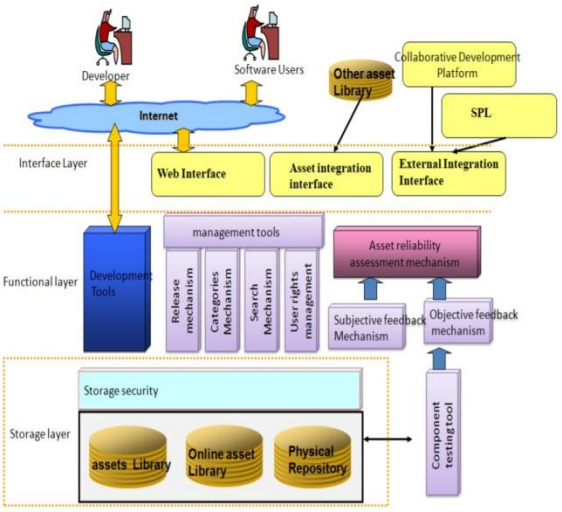
Specialized administration manages the genuine core assets advancement and item improvement exercises. This administration group guarantees that assets appointed to finish the exercises are working up to the stamp and furthermore guarantee whether they take after conferred forms for product line. To track the advance this group gathers the intermittent information.

III. PRODUCT LINES FOR MOBILE APPLICATIONS

The design of a mobile application is a monotonous task owing to its natural variability. Software designers need to take into account in the development process the adaptability of available platforms for e.g. android, iPhone, tablets. They are variety of existing devices with dissimilarity in introducing a further layer of complexity to the development process. In addition, at runtime, many possible situations have to be considered.

System Structure Design

Below figure displays the architecture of core asset library. The core asset library system comprises core asset, core asset management tool, code component development tool and code component test tool.



Structure of Core Asset Library

Code Component Development Tools:
Newly-built components and functions of editing, compiling and unit test are mainly provided by the code component development tools. Data processing, data interaction between components on the software product line platform is mainly through XML. Therefore the input and output parameters of methods that the component provides should be in XML. For general use of components, XML string data defines the input and output parameters. Coming to components each method, it normally contains 3 tasks: to parse XML input parameters, to execute the business logic and to encapsulate XML results. Test for simple units should be performed after new compilation. Various properties that can be configured to satisfy all kinds of requirements in product line assembly should be possessed by the components.

Code Component Test Tools :
Test tools for product lines are code component test tools. They provide objective evaluation of assets' trustworthiness adopting the B/S structure. They also combine with core asset library management systems and can supply black box test function for all code components inside the system for the purpose of code component inside the system. The test tools can graphically edit test data and process xml test data. The main functions of code component test tools are as follows:

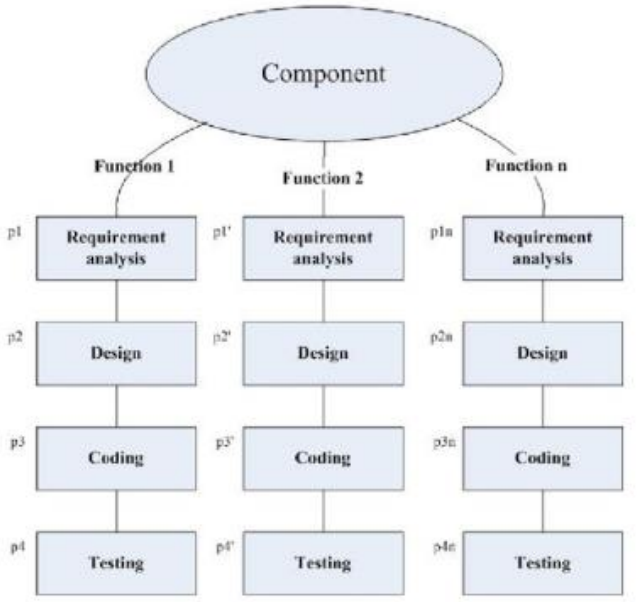
1. To display component list for choice.
2. To display the function list of appointed components;
3. Users can enter XML data as function input;
4. To display notation of the selected function or description of input & output;
5. To call selected function based on user input used as entry parameters;
6. After that, present output of the function;
7. Provide description for safety test, performance test and reliability test.

Core Asset
Core assets can be considered as reusable codes, libraries, design documents, testing documents and others components that can further used to build a new product with some or more similar functionality.

Core Asset Management tool

Asset management is defined as maintaining a desired level of service at the lowest life cycle cost. Asset management provides a framework to make data driven decisions about how to operate, maintain, repair, rehabilitate, and replace assets. The five core components of asset management are: current state of the assets, level of service, criticality, life cycle costing, and long term funding.

The Trustworthy Assurance Mechanism
The trustworthy software possesses trustworthy properties with the domain requirements. Data processing domain component should bear the following dependable features [5, 6, 7, 8]:



Correctness: It is the probability of realization of user expected functions. This is the basic and lowest requirement for software systems.
Reliability: In a specified period of time in a specified environment it is the probability of failure-free software operation.
Security: It is the probability of software to provide and safeguard privacy, integrity, availability and validity.
Real time: It is the probability of software to response and output within a prescribed time.
Maintainability: It is the probability that the software product can be modified. The modification may mean to correct or improve the software.
Survivability: It is the probability of software to continue to function during a natural or man-made disturbance or during the occurrence of failures and to restore full service within the prescribed time. In order to realize the measurement of trustworthy properties, we suggest the above trustworthy assurance framework demonstrated as the trustworthiness tree model.
There are four major links calling for trustworthy proofs during software development. They are the requirement phase, the design phase, the encoding phase and the testing phase. By evaluating the trustworthiness value at every stage the final trustworthiness value is calculated.

SPL Practice in the Mobile Industry:

There are additional cost and risks associated with establishment of Software product line development in the organization. An organization that attempts to encompass a product line approach without being aware of the costs are likely to withdraw the approach before seeing it through. Some of the costs and risks that is associated with the software product line implementation are listed below:
It is believed that the development of the core assets is costlier, it usually takes two of these products to be built as a family for this type of development process.
The training of the individual in the new approach for the business is considered costly, the individual should not be only trained in the software engineering domain but in the corporate procedures as well, this should be done to ensure that the product line practice would be used in accordance of the current compliance. Furthermore the new individual must be trained specifically for the new changes in the product line and the proper documentation of the training process should be done.
There is a potential risk in harmonizing the new product line approach and it causes a repulsion to the individual in doing the business in the new define procedure as the per the new product line. This kind of repulsion is often found in the middle level management and it may require the individuals to be moved to the other tasks.
An organization does not always require a level of process maturity in the process of the introduction of the product line development, if there is a hope for a successful launch of the procedure. It doesn't mean that the organization must comply with CCM level 5 to be successful in the approach, but for sure some level 2 and 3 practices are necessary to make it in the accordance.

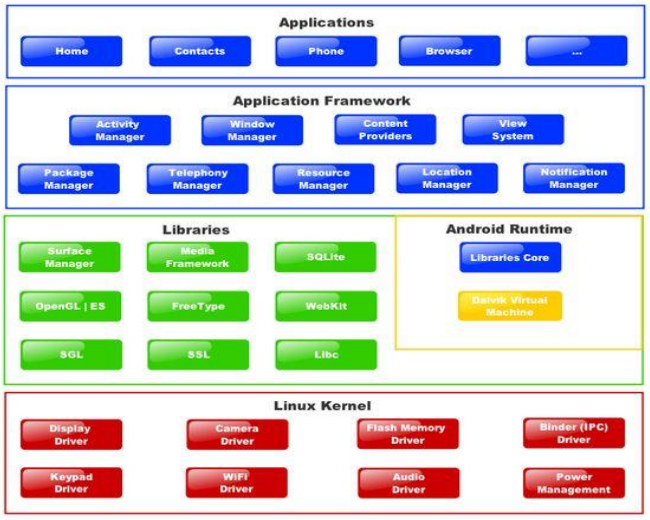
General Product Line Practices
The mobile computer industry has only recently become a popular platform for custom application development with the advancement in the reduced cost and wireless technology of micro hardware. As a result, product line techniques in the mobile space are not well established or implemented. The software development environment for this technology has some significant differences compared to desktop computing which means typical product line approaches are not necessarily effective. There are a variety of product line implementation techniques that have been used with different degrees of success outside the mobile application domain. Some are used in isolation; others can be used in combination with different product line approaches or compatible component and framework methodologies. General product line International Journal of Hybrid Information Technology practices that could be considered for mobile application development is represented in this section. In the cases where these general techniques have been tried with mobile development, we comment on their applicability to the device API displacement problem

Object-Oriented Frameworks and Patterns
An effective way of building re-usable components and sometimes form the basis of product line development practices are Object- oriented frameworks and design patterns. Some of these object-oriented frameworks have component models added to them that allow plug-n-play behavior for software applications with many optional features. This makes it easy to mix and match different applications with unique features to form a complete product instance.

Internet of Things in Mobile Device Product Line
The important characteristics of Mobile Device Product Line Security is no delay to react in front of disturbances due to detecting events in real time with the help of new technologies, mainly IoT. This is a new concept which allows any object communicates with others objects through Internet, and provide information in real time with new technologies, like RFID and sensors, to facilitate the exchange of goods and services in global supply chain networks Internet of Things is structured in three levels.

SPL Costs and Potential Risks
There are additional cost and risks associated with establishment of Software product line development in the organization. An organization that attempts to encompass a product line approach without being aware of the costs are likely to withdraw the approach before seeing it through. Some of the costs and risks that is associated with the software product line implementation are listed below:
• It is believed that the development of the core assets is costlier, it usually takes two of these products to be built as a family for this type of development process.
• The training of the individual in the new approach for the business is considered costly, the individual should not be only trained in the software engineering domain but in the corporate procedures as well, this should be done to ensure that the product line practice would be used in accordance of the current compliance. Furthermore the new individual must be trained specifically for the new changes in the product line and the proper documentation of the training process should be done.
• There is a potential risk in harmonizing the new product line approach and it causes a repulsion to the individual in doing the business in the new define procedure as the per the new product line. This kind of repulsion is often found in the middle level management and it may require the individuals to be moved to the other tasks.
• An organization does not always require a level of process maturity in the process of the introduction of the product line development, if there is a hope for a successful launch of the procedure. It doesn't mean that the organization must comply with CMM level 5 to be successful in the approach, but for sure some level 2 and 3 practices are necessary to make it in the accordance.

Android Case Study:
A library is a collection of functions / objects that serves one particular purpose. A library can be used in a variety of projects. A framework which is a collection of patterns and libraries is used to ease building an application. An API is an interface for other programs to interact with your program without having direct access. In another way, library as apiece of an application, a framework as the skeleton of the application, and an API as an outward-facing part of said app.



Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown above in the architecture diagram.

Linux kernel:

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

Libraries:

There are a set of libraries on top of Linux kernel including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Libraries:

This category encompasses those Java-based libraries that are specific to Android development.

It is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack, as we have covered Java-based core libraries in the Android run time.

A summary of some key core Android libraries available to the Android developer is as follows − Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access.

Android Runtime:

This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

Android developers will write android applications using standard java programming language which is enabled by a set of core libraries which are also provided by the android runtime.

Application Framework

In the form of java classes the Application Framework layer provides many higher-level services to applications. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services:

Activity Manager − All aspects of the application lifecycle and activity stack are controlled.

Content Providers – It allows to publish and share data with other applications through applications.

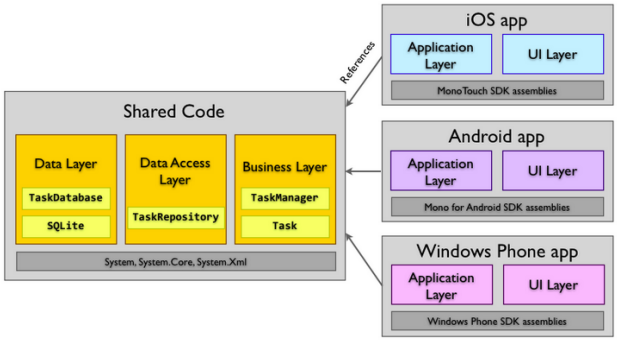
Resource Manager – Access is provided to non-code embedded resources such as strings, color settings and user interface layouts.

Notifications Manager – It allows to display alerts and notifications to the user by applications.

View System − For creating user interface applications an extensible set of views are used.

Applications

In the top layer we all will find Android application.



Examples of such applications are Contacts Books, Browser etc.

Tasky Case Study

Tasky portable is a simple to-do list application. In Tasky Portable sample application the principles of Building Cross-Platform Applications have been applied.

Common Code

A common project that contains re-usable code to store the task data. This reusable code is stored in core asset library.

Tasky Portable uses the Portable Class Library strategy, which is a library for sharing common code. All common code, including the data access layer, database code and contracts, is placed in the library project. All of the code in the portable library is compatible with each targeted platform.

Platform-specific Code

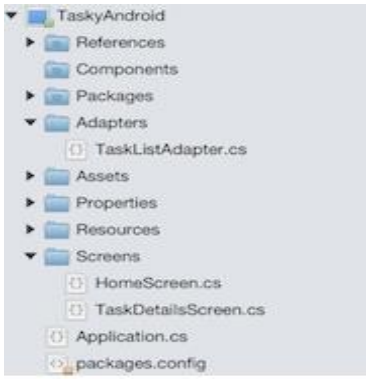
Platform-specific projects that implement a native UI for each operating system utilizing the common code.

Once the common code has been written, the user interface must be built to collect and display the data exposed by it. The TaskItem Manager class implements the Façade pattern to provide a simple API for the application code to access. The

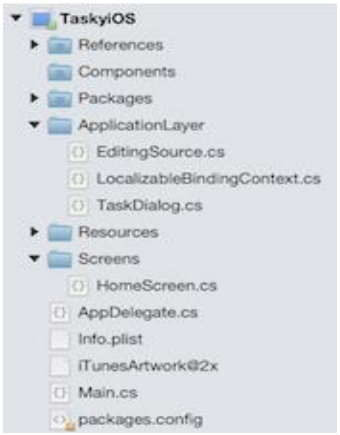
code written in each platform-specific project will generally be tightly coupled to the native SDK of that device, and only access the common code using the API defined by the TaskItem Manager. This includes the methods and business classes it exposes, such as TaskItem.

Coding Libraries of Android and IOS are shown in below figure

Android APP: Xamarin.Android



IOS App: Xamarin.iOS



Some Popular Libraries

Android:

Gson: Gson is a Java library used for serializing and deserializing Java objects from and into JSON.

Retrofit: It's an elegant solution for organizing API calls in a project. The request method and relative URL are added with an annotation, which makes code clean and simple.

EventBus: EventBus is a library that simplifies communication between different parts of your application.

ActiveAndroid: ActiveAndroid is an ORM for Android. It's an abstraction over SQLite which allows you to communicate with a database on a device without writing SQL statements.

Universal Image Loader:

UIL is a library which provides asynchronous, out of the box loading and caching of images

IOS Libraries:

AFNetworking: this library makes every developer's life a whole lot easier. AFNetworking is a light-weight and fast networking library that uses blocks and GCD (Grand Central Dispatch).

JSONModel: JSON Model is an open-source library that helps you with parsing and initializing your model classes with JSON response from the server.

MagicalRecord: Magical Record, a wrapper around Core Data that simplifies managing your persistence storage.

SDWebImage: SDWebImage specializes in image downloading and caching, and offers an extensive features.

ReactiveCocoa: A major advantage of ReactiveCocoa is that it provides a way to deal with asynchronous behaviors, including delegate methods, callback blocks, target-action mechanisms, notifications and KVO, simply by using signals.

Benefits of SPL

Software Product Libraries play very important role. The benefits of SPL are given below:

1. Large-scale productivity gains
2. Decreased time to market
3. Increased product risk
4. Increased market agility
5. Increased customer satisfaction
6. More efficient mass customization
7. Ability to maintain market presence
8. Ability to sustain unprecedented growth.

IV. Conclusion

Organizations can gain considerable business benefits in terms of reduced costs, shortened time-to market, increased product quality, etc. by applying product line development. There is however risks and costs associated to the adoption of business to product line development that must be evaluated before embarking in a reuse initiative of this kind. Organizations have beard witness of cancellation of several major projects to free resources for core assets development, the reassignment of employees that could not adapt to the new way of doing business and the suspension of product delivery while putting the new way of business on the road. This implies considerable costs to the organizations, however all those organizations still agreed that the return on investment well made up for the additional costs (SEI PLP, 2003). Considering the benefits gained by organizations who have successfully adopted a software product line approach, the final conclusion to be drawn from this paper is that any organization developing related products as single systems, that has good domain knowledge and a reasonably high level of process maturity, should at least build the business case to evaluate if product line development might be right for them.

References

1. https://www.researchgate.net/publication/304011170_A_Case_Study_Using_AOP_and_Components_to_Build_Software_Product_Lines_in_Android_Platform
2. http://www.sei.cmu.edu/productlines/
3. http://www.sersc.org/journals/IJHIT/vol9_no7_2016/37.pdf
4. https://ticsw.uniandes.edu.co/software-product-lines/a-software-product-lines-for-mobile-applications
5. http://tuprints.ulb.tu-darmstadt.de/epda/000630/cepa-thesis.pdf
6. https://www.tutorialspoint.com/android/android_architecture.htm
7. http://www.sersc.org/journals/IJHIT/vol9_no7_2016/37.pdf
8. https://ticsw.uniandes.edu.co/software-product-lines/a-software-product-lines-for-mobile-applications
9. http://tuprints.ulb.tu-darmstadt.de/epda/000630/cepa-thesis.pdf
10. https://www.cqse.eu/publications/2012-effective-and-efficient-reuse-with-software-libraries.pdf
11. http://ecc.ibm.com/case-study/us-en/ECCF-CDC12386USEN
12. https://ercim-news.ercim.eu/en88/special/capucine-context-aware-service-oriented-product-line-for-mobile-apps
13. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=12587
14. https://link.springer.com/chapter/10.1007/978-3-540-88030-1_18
15. https://android-arsenal.com/
16. https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/case_study-tasky/
17. http://square.github.io/#android
18. https://infinum.co/the-capsized-eight/top-5-android-libraries-every-android-developer-should-know-about
19. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjf9_725sbTAhUq2IMKHZsADx0QFghJMAA&url=https%3A%2F%2Finfinum.co%2Fthe-capsized-eight%2Ftop-5-ios-libraries-every-ios-developer-should-know-about&usg=AFQjCNG7mRqhKZaxTd4ojjIwcEAKCNTYgA&sig2=Lz1qslAO1bdAKy2CbQvNAA
20. https://books.google.com/books?id=SFZoKgTxJ1EC&pg=PA324&lpg=PA324&dq=structure+of+core+asset+library&source=bl&ots=uX9lVr0sBD&sig=o32BJrBf5uXW4wTd9k5C5szkBik&hl=en&sa=X&ved=0ahUKEwjkgJj_vMbTAhVG4oMKHQitBpAQ6AEIMzAC#v=onepage&q=structure%20of%20core%20asset%20library&f=false