UNIVERSITY OF GOTHENBURG

Architectural Support for Openness in Mobile Software Platforms

Mohsen Anvaari

Master's Thesis in Software Engineering and Management

Report No. 2010:064
ISSN: 1651-479

University of Gothenburg
Department of Applied Information Technology
Gothenburg, Sweden, May 2010

Table of Contents

Abstract

Introduction: The answer to the frequently asked question "how open is a software platform" is not binary; especially when it comes to software platforms for mobile devices. The openness of these platforms is determined by the openness strategy of a software producing organization. The decision to open up a platform, however, determines the degree of freedom for third parties to adopt the platform for commercial opportunities.

Objective: The aim of this thesis is identification of the openness strategies of the main mobile platforms based on their architecture.

Methodology: The openness strategies are uncovered using literature review and several qualitative interviews with mobile application developers.

Results: An architectural openness model, several architectural openness factors and identification of openness strategies in the main mobile platforms are results of this thesis.

Conclusions: The proposed architectural openness model shows how the openness strategies of mobile platform suppliers affect the software architecture of the platforms. Architectural openness factors demonstrate how open the mobile software platforms are. Finally based on the model and the factors, the openness degree of five main mobile platforms is indentified.

Audience: Researchers of the mobile software community, mobile software platform suppliers, application developers and architects could benefit from using the results of this thesis.

Keywords: Mobile Software Platforms, Openness Strategy, Platform Architecture, Platform Accessibility, Literature Review, Qualitative Interview

Note: A paper from this thesis is submitted to the 2nd Workshop on Software Ecosystems in conjunction with 4th European Conference on Software Architecture 2010, Copenhagen, Denmark.

1. Introduction

Open source or proprietary; which of them is more successful? Which strategy has received more attention from the developers? Which does lead to more innovative applications? These are some among several frequently asked questions in the software community. The discussion was stimulated by Raymond's paper “The Cathedral and the Bazaar” in which open source community is compared to a bazaar while proprietary community to a cathedral (Raymond, 2001). One can say the open source strategy is more successful due to its creativities and inventions and on the other hand, one can believe that the proprietary strategy is more prospering because it will lead to more qualified products due to strong control. But the reality is not that black and white, especially when it comes to the software on the smartphones called mobile software platform. Considering a platform as open or closed is rarely a binary decision and generally is a question of “how open” (Maxwell, 2006). The answer of the question deals with openness strategy of the platform.

Openness strategy is the degree to which a platform approaches to open attributes which depend on different technical and commercial aspects such as platform architecture, platform accessibility, licensing state, marketing policy, etc. Mobile software platform means the overall structure of the software on the mobile devices (Cho and Jeon, 2007). The openness strategy is different among various mobile software platforms of smartphone ecosystem. Generally, a software ecosystem is “a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them” (Jansen et al., 2009). When the definition comes to the smartphones area software platform suppliers, device manufacturers, operators, application developers, device customers, etc are considered as the actors and participants of the ecosystem. In the current smartphone ecosystem Symbian, Windows Mobile, iPhone, BlackBerry, and Android are the main software platforms competing for superiority (Canalys research, 2010). Some of these mobile platforms are more successful than others and many differences between these platforms exist. Some are available for any hardware such as the Android platform, whereas others are only available on limited hardware such as the iPhone platform. To implement the openness strategy that platform suppliers have made, the architecture of their platform should support proper platform accessibility for their application developers and device manufacturers.

This thesis studies the openness strategy of the different mobile platforms. The scientific contribution of this thesis lies in the identification of the openness strategy of the main mobile platforms based on their architecture. The intended audience of this thesis is researchers of the mobile software community, mobile software platform suppliers, application developers and architects.

Figure 1 shows the domain model of the research. The platform suppliers define their openness strategy based on their business model by considering the architectural aspects of the platform. The strategy they choose affects device users, application developers, device manufacturers and operators. In this research, the architectural aspects of the platforms and some licensing aspects related to platform accessibility are studied. Platform architecture is the structure of the software platform compromises its components and the relationship between them (Bass et. al, 2003) and platform accessibility means the methods and points that developers can use to extend or modify the platform. Since other aspects of the business model such as marketing are not in the scope of *software engineering and management* they are not studied in this thesis. To validate the openness strategy in different mobile platforms, only application developers are chosen to be interviewed. This is due to time limitation, although the device manufacturers are also in the scope of this research.

The remainder of this report is structured as follows: In part 2, the research questions of the thesis are clarified and a summary of the research methods is presented. Before describing the activities of the study, a summary of related works is presented in the part 3, and then in part 4, results of the activities of thesis, which are literature study and some qualitative interviews, are described. Part 5 and 6 discuss the research results and argue how the research methods and the research progressed. Part 7 summarizes all sub-questions and answers, and provides some pointers for future research.

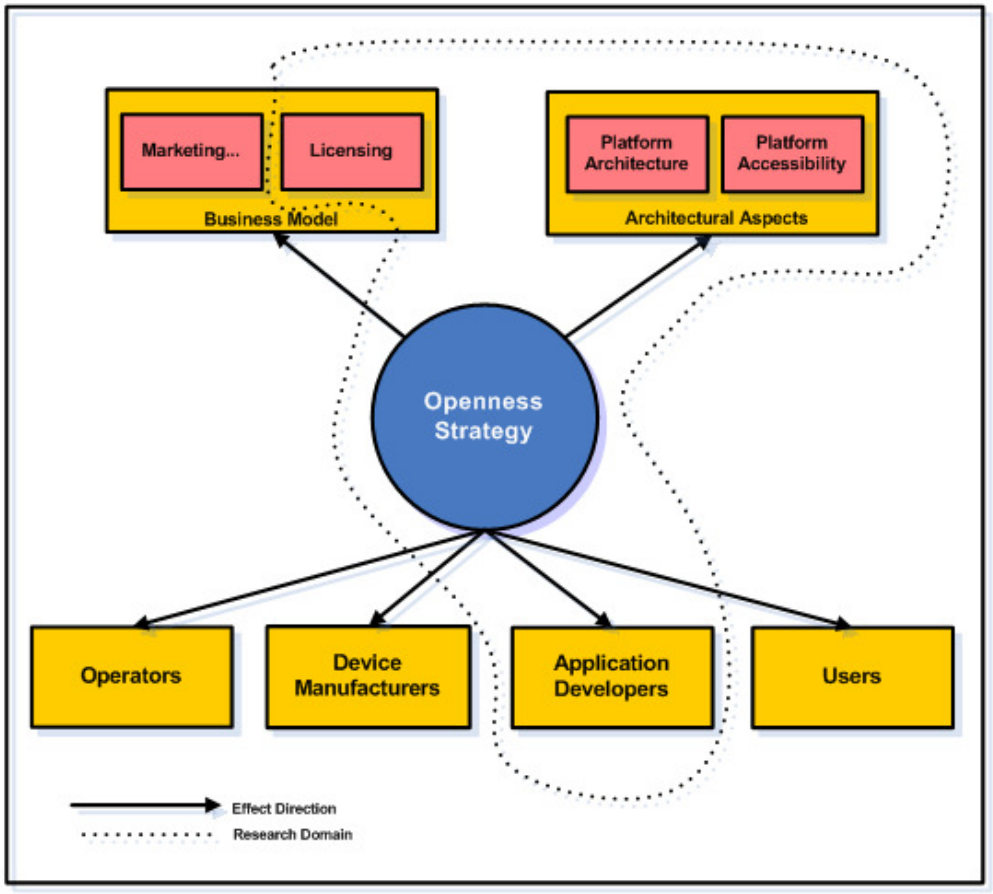

Figure 1. Domain model of the research

2. Research Approach

As mentioned before, the aim of this research is identifying the openness strategies in the five main mobile platforms based on their architecture. The research question and sub-questions are defined as follows:

RQ: How does the software architecture expose the openness strategy of mobile software platforms?

- SQ1: Can a model be developed that describes the architectural openness of mobile platforms?
- SQ2: How would the licensing aspects of the platforms relate to such a model?
- SQ3: How open are the five main mobile platforms?

In the following section the methods chosen for this research are discussed.

2.1. Research Methods

To answer above questions, two research methods are conducted:

- Literature review for finding the relationship between openness strategies and the software architecture of mobile platforms and finding out the platform accessibilities their application developers have to extend the platform.
- Several qualitative interviews (semi-structured interviews) with mobile application developers in the Netherlands to verify the platform accessibility in the five main platforms that allows application developers to extend the platforms, which previously has been captured from the literature and platform documents.

In order to address the questions, the literature and also the documents of the mobile platforms are studied to find the openness strategy in the platforms, the software architecture of the platforms and the platform accessibilities. A reference model is defined to make a connection between the openness strategy and the software architecture in mobile software platforms. The model is made based on common architecture of mobile software platforms and common platform extension ways application developers use to extend the mobile platforms. The extension ways deal with the openness strategy of platforms.

In order to verify the answer of the third sub-question captured from the literature, several interviews with the mobile application developers are conducted. Based on the reference model and the results from the interviews, the discussion and analysis about the control and strategy of the platforms are argued and some suggestions are mentioned. The objectives that should be achieved stepwise during this research are:

- Building a model to describe the architectural openness of mobile platforms.
- Defining architectural openness factors by considering the licensing aspects of mobile platforms in the model.
- Looking at main mobile platforms by the lens of developed model and factors to determine how open the platforms are.
- Conducting some qualitative interviews with application developers of the platforms to confirm the results of previous step.

The activity diagram demonstrated in Figure 2 shows the execution process followed in order to conduct the research.

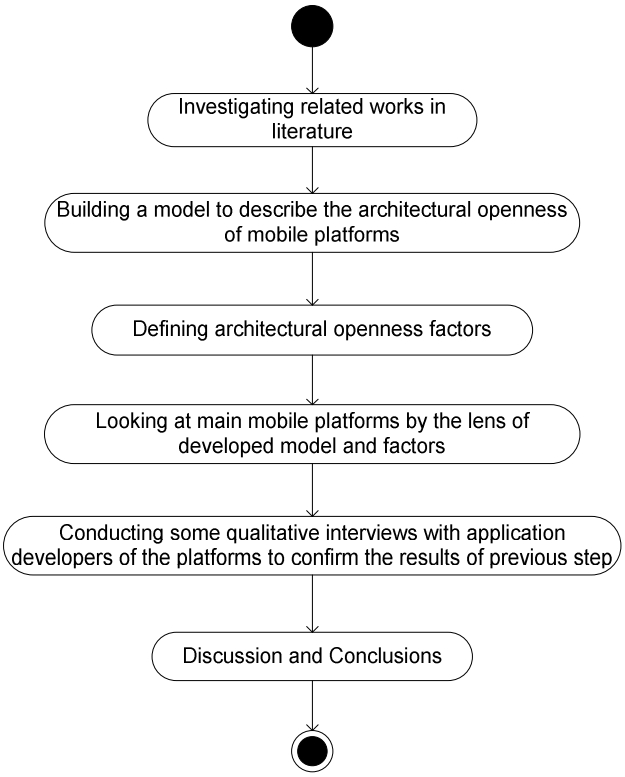

Figure 2. Execution Process of the Study

In the following sub-sections, more details about the literature review and qualitative interviews are presented.

2.2.1. Literature Review

> A review of prior, relevant literature is an essential feature of any academic project. An effective review creates a firm foundation for advancing knowledge. It facilitates theory development, closes areas where a plethora of research exists, and uncovers areas where research is needed (Webster and Watson, 2002).

To conducting a scientific and high quality literature review, "systematic literature review" is one of the main suggested methods among researchers. Planning the review, conducting the review and reporting and dissemination are the key stages of the process of a systematic review (Tranfield et al., 2003). Nevertheless, as Bryman and Bell mention, this approach contains some parts that cannot be easily applied in a student research project (like a master thesis) due to limitations of time and resources (Bryman and Bell, 2007). However, they believe there are some aspects of the approach can be used in a student research (*ibid*). In this research a similar but customized approach to the

systematic literature review is conducted. In the plan stage, according to the research objectives, some keywords were chosen to search the literature and due to access permissions, some relevant online databases were selected. To conduct the review, after finding preliminary results, the backward and forward approach suggested by Webster and Watson was applied to find more relevant resources. Going backward by reviewing the references of the articles and going forward by reviewing articles citing the preliminary founded articles is the main instruction of the approach (Webster and Watson, 2002). After reading and filtering all the founded articles by reviewing the abstract (filter 1) and the whole article (filter 2), several relevant articles were chosen to be summarized and cited in the research. Finally, to summarize the relevant resources, a similar approach to the concept matrix (Salipante et el., 1982) was conducted. For each main topic, the relevant resources are compiled in a table. For each resource as a row, the table includes a column for the covered keywords and one for the relevant quotes. Table 1 shows the summary of literature review.

Table 1. Summary of literature review

Topic	Keywords	Number of relevant articles	Sources
Conducting literature review	literature review, systematic literature review, structure literature review	**4**	IEEE, ACM, Springer Link (databases) and Engineering Village, ISI Web of Science, Google Scholar (search engines)
Related works	architectural openness model, software architecture and openness, open mobile platforms, ecosystem of mobile platforms	**9**	
Software openness and software extension mechanisms	open software, software openness, open source, closed source, open platforms, open standards, open foundation, open architecture, software extension, platform extension, platform API, architecture and openness, open and closed architecture, architecture and software extension, platform architecture	**11**	
Mobile software platforms	open mobile operating system, open mobile platforms, android architecture, android openness, iphone architecture, iphone openness, windows mobile … , blackberry …, symbian …	**37**	

2.2.2. Qualitative Interviews

> The qualitative research interview is a construction site of knowledge. An interview is literally an *inter view,* an interchange of views between two persons conversing about a theme of mutual interests. (Kvale, 1996)

There are three types of research interviews: *structured* interview which is usually conducted by a structured questionnaire, *semi-structured* interview which is conducted on the basis of a loose structure consisting of open ended questions and *unstructured* (in depth) interview which is less structured and usually covers one or two issues in great details (Britten, 1995). Considering the context of interview, *more than one interviewee* (group interviews or focus groups), *more than one interviewer* and *interviews by telephone* are another types of interviews (Bryman and Bell, 2007). In qualitative researches, the two main types of interviews are semi-structured interview and unstructured interview. "Researchers sometimes employ the term qualitative interview to encapsulate these two types of interviews" (*ibid*).

For this research semi-structured interview is chosen. In this type of interview the researcher has a list of questions on specific topics to be covered, but the interviewee has a great deal of leeway in how to reply. Questions may not follow in the way outlined on the questionnaire. Questions that are not included in the questionnaire may be asked as the interviewer picks up on things said by interviewees. But all the questions will be asked and a similar wording will be used from interviewee to interviewee (*ibid*). The main objective of interviews of this research is validating the openness strategy of mobile platforms by gaining experiences of mobile platform developers. To prepare the questionnaire, some general questions are created based on the objectives of interviews. Some more specific questions about favorite mobile platform of interviewees are added to the list of questions. The questionnaire is presented in Appendix. In the same time, several application developers are contacted to see whether they are interested to be interviewed or not. The initial plan was to find one application developer for each platform. However, for Android no developer was found available for a face to face interview, therefore an online discussion with developers of Android group in LinkedIn website is chosen as the secondary plan. After finalizing the questionnaire and making appointments with interviewees, the interviews are conducted. The interviews are recorded by a voice recorder, so transcription is the next step of the interview process. Finally for analyzing the interview data, the proposed stage-by-stage method by Burnard

is applied. The aim of this method is “to produce a detailed and systematic recording of the themes and issues addressed in the interviews and to link the themes and interviews together under a reasonably exhaustive category system” (Burnard, 1991).

3. Related Works

The most recently published article in the area of smartphone ecosystem states that few attempts have been made to demonstrate the comparison of smartphone OSs (Lin and Ye, 2009). It is also true particularly about comparing the openness strategies in different mobile software platforms. Lin and Ye, themselves, have compared the ecosystem of iPhobe, Symbian, Windows Mobile and Blackberry by discussing about the value chain (they call it “food web”) of the ecosystems, but they have not discussed the openness strategy of the platforms based on the architectural aspects. Further, Android is not in their comparison. Cho and Joen have discussed and compared Symbian, Linux and Rex (Cho and Joen, 2007), which, except Symbian, are not the current major platforms. Besides that, although they have compared the architecture of the platforms and also the openness strategy of the platforms, but not a connection between two subjects has been settled. Yamakami has covered Android, Symbian, iPhone and Windows Mobile in his competitive analysis (Yamakami, 2009). But the factors he has considered in his analysis are governance, ease of maintenance, interoperability, ecosystem, cost reduction and reliability, which are not related to architectural aspects. Constantinou has discussed open source in mobile platforms and shows the current state of openness in software stack of mobile platforms (Constantinou, 2008). It is very general and not applied to any specific platform.

Beyond the software platforms on mobile devices, in the software area in general, there is again a lack of works that discuss how software architecture is treated in open source projects (Nakagawa et al., 2008). Nakagawa et al. have shown that software architecture has an important role in open source projects via a case study, but they have not discussed how the openness strategy affects the architecture. Prehn has concluded in his article that one characteristic that is shared by the largest and most successful open source projects is a software architecture and it has to be modular (Prehn, 2007). Arief et al. in a similar conclusion shows that for open source projects the architecture must be modularized (Arief et al., 2001). Both are not applicable results for the case of comparing the openness strategy in different mobile software platforms.

4. The Results of Research Activities

As mentioned earlier, the research activities of this thesis are studying the literature and conducting some qualitative interviews. The structure of the activities and methods was described in part 2. In this section the results of the activities are presented.

4.1. Software Openness and Software Extension Mechanisms

Openness is the "quality or condition of being open" (Oxford English dictionary, 2010). The software openness, in the scope of this research, means the degree to which a software platform approaches to open characteristics which depend on accessibility and licensing as key aspects. There might be other aspects which determine openness in a software platform but only these two are researched here as explained in the research domain. Although openness affects many aspects of computing besides degree of accessibility to view and modify source code (Lamothe, 2006) but when accessibility aspect is considered, degree of freedom to modify components of a software system will be a key factor to define the openness of software platforms. On the other hand, software extension mechanisms let people modify the software's functionality or architecture (Scacchi, 2004). So the terms "software openness" and "software extension mechanisms" are related and therefore are discussed in this part together. First openness in software, open-source software, closed-source software, proprietary software and other related issues are discussed based on the literature and then software extension mechanisms are presented.

To clarify the meaning of the openness, Maxwell in his article starts with this question that what is the technical and philosophical meaning of openness? He answers that many definitions are possible and it is not simple to say a work or process is open or closed. He compares the openness concept to a spectrum and discusses if a person creates a work but does not share it with anyone, the work is closed and on the other end of the spectrum are works made available to and modifiable by all. Then he explains that most works stand between two extremes (Maxwell, 2006). Brown and Booch use open term for further phenomenon in software area by defining *open software, open collaboration, open process, open release, open deployment* and *open environment* (Brown and Booch, 2002) and in all of them, the openness should not be considered as a binary characteristic. West in his article called "how open is open enough" expands the traditional open-close view

to a more flexible range. He assigns *proprietary*, *opening parts*, *partly open* and *open source* to the openness degree of a software platform. Proprietary platform consists of an architectural standards controlled by one or more sponsoring firms where architectural standards typically encompass a processor, operating system (OS), and associated peripherals. In opening parts platforms, suppliers wave control of commodity layers of the platform, while retain full control of other layers that presumably provide greater opportunities for differentiation. In partly open platforms, suppliers disclose technology under such restrictions that it provides value to customers while making it difficult for it to be directly employed by competitors. Open source platforms are those that the ability to create and modify software products is governed by the access to the source code (West, 2003). Asplaugh et al. in their research mention several software elements included in common software architectures that affect the openness of architecture. The elements are software source code components, executable components, application program interfaces (APIs) and configured system or sub-system architectures (Asplaugh et al., 2009). The ways that architectural elements of a software system affect the openness of the system are being expressed as software extension mechanisms and approaches. Klatt defines software extension mechanism based on extension points. An extension point is the definition of the provided interface for extensions. An extension itself is an implementation according to an extension point. And an extension mechanism includes everything about an explicit extensible part of a software" (Klatt, 2008). Jansen et al. mention several mechanisms in their article that extend software functionality. The mechanisms are component calls, service calls, source code inclusion, and shared data objects (Jansen et al., 2008). To analysis more specifically the extension in mobile software platforms, a discussion of extension mechanisms on operating systems is needed since software platforms on mobile devices are expressed as operating system of the devices (Verkasalo, 2009). Alexandrov et al. present different approaches for extending the operating system. The ways they have mentioned are extension or modification of different levels of an operating system from the lower levels to the higher levels. The approaches are changing operating system (kernel) itself, modifying device drivers, installing a network server, adding user level Plug-Ins, making changes to user level libraries, applications specific modifications and intercept system calls (Alexandrov et al., 1997). Their point of view about extension approaches is employed to develop the architectural openness model for this research and is discussed in the following chapters.

4.2. Mobile Software Platforms and their Architecture

In the last decade, mobile phones have become programmable handheld computers which have internet connectivity, computing power and open application programming interfaces (APIs) providing prospective platforms for an infinite set of new mobile services and applications (Verkasalo, 2009). The new mobile phones which are usually called smartphones have both hardware and software parts the same as all computing systems. The software part is called mobile software platform. Some authors like Verkasalo use software platform as a synonym to operating system of mobile devices (*ibid*). Some others like Cho and Jeon believe that software platform of a system means the overall structure of the software on the system and operating system is a part of it (Cho and Jeon, 2007). Cho and Jeon consider a layered architecture for the software platform of typical mobile devices consists of operating system layer, middleware layer and applications layer (ibid). Layered architecture is an architectural pattern helps to structure systems that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction (Buschmann et al., 1996). The software platform of mobile devices is such a system, therefore the layered architecture is applicable to the architecture of mobile software platforms. The suggested model for architecture of mobile software platforms for this research is the same as proposed architecture by Cho and Jeon with some modifications. In their model, they have not separated the default applications which are set by device manufacturer from applications developed by third-party community and installed by users. Since third-party applications have a main role in defining the openness of a platform, the proposed model for this research has two different sub-layers in the applications layer: native applications and extended applications. Native applications are those developed by device manufacturers and in some platforms are not modifiable. Extended applications are those developed by application developers and installed by device users, so these applications extend the applications layer of the platform. Middleware layer consists of main libraries and services of the platform like data storage, virtual machine, multimedia libraries, etc. When application developers create extended application, they usually call this layer in their applications instead of calling the core libraries of the platform. The kernel layer, which in Cho and Jeon's model is called operating system layer, is the core the platform. It consists of the lower level components of the platform such as device drivers, power

management framework, security framework, etc. Figure 3 shows the proposed architecture for mobile software platforms.

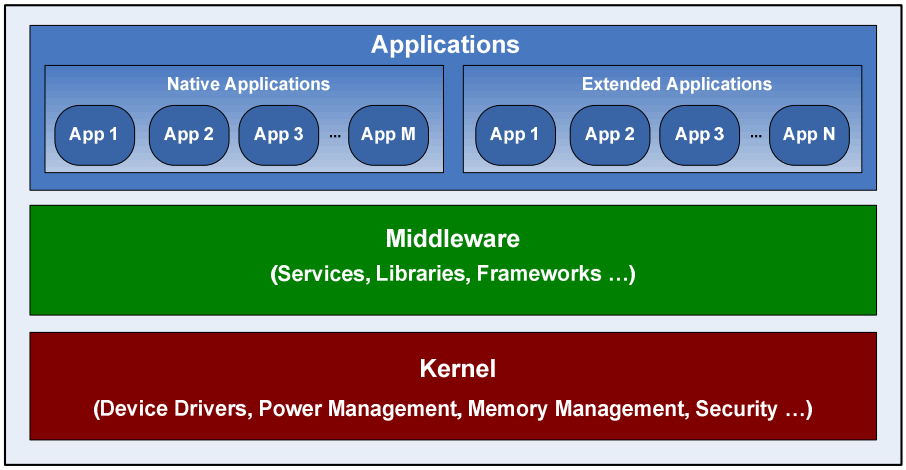

Figure 3. Architecture of Mobile Software Platforms

The architecture of main mobile platforms include Android, iPhone, Symbian, Blackberry and Windows Mobile is looked by the lens of this architecture and the results are presented in the following parts. In the next section, the architectural openness model which is built based on the proposed architecture is discussed.

4.3. Architectural Openness Model and Openness Factors for Mobile Software Platforms

As presented in the previous part, a mobile software platform like other software systems has an architecture which is the structure of the platform. A proposed mobile software architecture that is a general model and can be applied to different mobile platforms was shown. To discuss the openness strategy of mobile platforms based on their architecture, the proposed model is not sufficient and it should be expanded to an "architectural openness model" to demonstrate platform extension mechanisms and platform accessibility since these notions have a connection with the openness concept as discussed before.

The architectural openness model to accommodate the platform accessibility and platform extension methods and in a higher view the platform openness, should illustrate how much and under which conditions the platform extenders (application developers, device makers, customers…) can access to different layers and components of the

platform and extend its functionality. Two online resources of Google Android have used and defined *integrate, extend* and *modify* concepts to clarify the openness notion in the architecture of Android platforms (Chen, 2008 and Live from Google I/O – Android, 2008). Sim et al. mention similar meanings when consider *integration* and *customization* as issues in mobile operating systems (Sim et al., 2006). To expand the "mobile software architecture" presented in the previous part to "architectural openness model", the same terms and definitions are used here:

Integrate a layer: To use the existing components of a layer in a mobile application via API, Service Call, source code inclusion, shared data object and other software extensions mechanisms.

Extend a layer: To enhance the functionality of the components of a layer. The application uses the built-in Google map application and adds its own functionality on top of Maps is an example.

Modify a layer: To replace or change the components of a layer. Writing your own device driver is an example.

The architectural openness model to support the openness strategy in mobile software platforms is illustrated in the Figure 4.

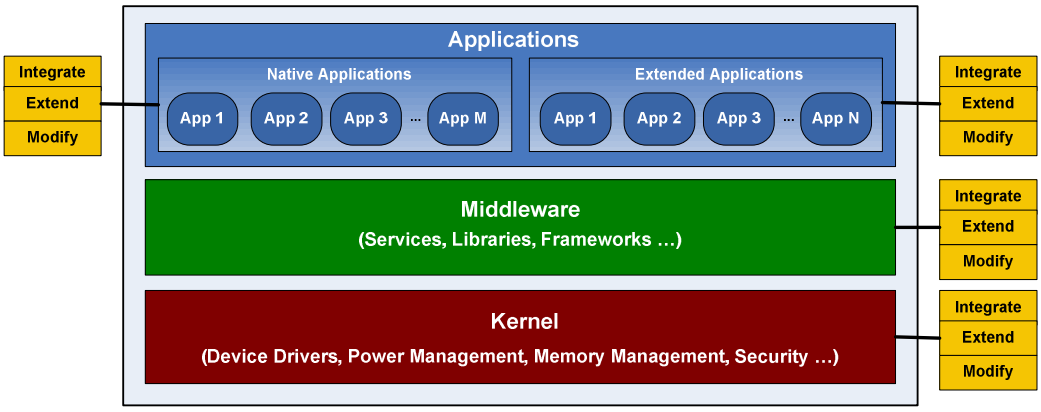

Figure 4. Architectural Openness Model for Mobile Software Platforms

Although the model shows the platform access and extension methods in different levels, but to demonstrate the openness degree of a platform, licensing aspects of mobile platforms should also be considered. Different mobile platforms have various licensing considerations. Besides the possibility to integrate, extend or modify the components of a platform, the permission of these activities depends on the platform supplier's licensing policy. In some platforms you do not need any permission to extend the platform whereas

in others you need to submit the change to the platform supplier first and if they accept, the changes will be confirmed.

Based on the proposed architectural openness model and possible states of licensing, the following table shows the architectural openness factors for mobile software platforms. Applying this table to a particular mobile platform, the result for instance for middleware layer of the platform can be like this: Integrate the components of the layer is allowed and doesn't need any permission. Extend the components of the layer is allowed but needs permission from the platform supplier. And modify the layer is allowed only for some components and needs permission.

Table 2. Architectural openness factors for mobile software platforms

Layer	Factor	Possibility statuses	If possible→Licensing statuses
Extended applications	Integrate extended applications	Possible/ Possible for some components/ Not possible	Permission is not needed/ In some situation permission is needed/ Permission is always needed
	Extend extended applications		
	Modify extended applications		
Native applications	Integrate native applications		
	Extend native applications		
	Modify extended applications		
Middleware	Integrate middleware		
	Extend middleware		
	Modify middleware		
Kernel	Integrate kernel		
	Extend kernel		
	Modify kernel		

These factors and their statuses provide a continuum to determine how much a platform is open based on the architecture of the platform. A platform is considered entirely open if in all the architectural layers of the platform integrate, extend and modify the components are possible without any permission from the platform supplier. On the other end, a platform is totally closed if within all layers of the platform integrate, extend or modify the components are not possible. Other platforms are situated between these two boundaries. In the following part, the situation of five main mobile platforms being Android, iPhone, Symbian, Blackberry and Windows Mobile in the openness continuum based on the architectural openness model and the openness factors is discussed.

4.4. Openness in Main Mobile Platforms

In this section, based on the architectural openness factors presented in the previous part, the openness strategy in five main mobile platforms being Android, iPhone, Symbian, Windows Mobile and Blackberry is discussed. In the section of each platform before

arguing about the architecture and openness of the platform, an introduction about the platform, its history and current situation is presented.

4.4.1. Android

Introduction: In November 2007, Google formed a group of mobile and technology companies called The Open Handset Alliance. The aim of this conglomerate is improving mobile phones to change the mobile experience for customers by increasing the openness in the mobile ecosystem. Their first joint project is Android which has been released officially on October 2008 (Open Handset Alliance, 2007). Android is a Linux-based OS that's geared to run on lightweight devices like mobile phones (Childers, 2009) and has recently launched on TVs (Fleming, 2010). At the time of releasing the platform, Android Market was also made available to users as a store where developers can publish and distribute their applications to users of Android-compatible phones for free (Android Market, 2008).According to Canalys research, the proportion of smart phones running Android OS in 2009 was almost 5% only after less than two years of releasing (Canalys research, 2010).

Architecture: Software architecture of Android platform is a layered architecture described in the official website of the platform (What Is Android?, 2010). Looking at the architecture through the "architectural openness model" lens, the result will look like Figure 5. In the application layer, there are some native applications including an email client, SMS program, calendar, maps, browser, etc which are written in Java (*ibid*). Developers can extend this layer by adding their own applications. In the middleware layer an application framework, a virtual machine and several libraries are provided to offer developers the ability to build rich and innovative applications (*ibid*). Kernel of Android is built on the Linux kernel, but does not include the full set of standard Linux utilities. It relies on Linux for core system services such as process management, memory management, security, etc (*ibid*).

Openness Factors: When Android was released, one of its major architectural goals was to allow applications to reuse components from one another. This reuse applies to services, data and UI. As a result, the Android Platform has some architectural features that keep this openness a reality (Hashimi and Komatineni, 2009). To see how open the Android architecture is, the openness factors presented in the section 4.3 are discussed for Android as follows:

Applications: According to the official website of Android, "any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user" (What Is Android?, 2010). It means that all integrate, extend and modify factors are supported by the platform for any application including native and extended one.

Middleware: Since all source code of Android platform is officially available, and all components of the platform can be called through APIs, therefore middleware layer of Android as a part of the whole platform is allowed to be used by developers. So integrate factor is supported for this layer. In the official documents of Android, there is no statement about the possibility of extend and modify factors for the middleware layer in general. But there are some examples show at least some components of this layer are customizable. For instance in Dalvik, which is the virtual machine of Android "it is possible to customize the set of optimized instructions for your platform" (Dalvik, 2010). To see how much other services and libraries of this layer such as windows manager or media framework are open to be extended or modified by developers there is lack of literature. So it is evaluated via an interview with an Android developer and the results are presented in the section 4.5.

Kernel: The same as other parts of the platform, Android kernel can be integrated directly in developer's application via APIs. After building default kernel, developers can begin to modify some components of the kernel such as device drivers to be compatible with their target devices. They can write their own drivers or customize the default drivers (Bring Up, 2010). So it means device drivers of this layer are allowed to be integrated, extended and modified. But it is not true about whole the layer. Power management is an example. "User space native libraries … should never call into Android Power Management directly. … Bypassing the power management policy in the Android runtime will destabilize the system. All calls into Power Management should go through the Android runtime PowerManager APIs" (Android Power Management, 2010). So it means power management can be just integrated and is not accessible for developers and users to be extended or modified. Conclusion is that kernel layer is totally allowed to be integrated by developers and partly allowed to be extended and modified.

Licensing status: Although "developers do not need permission to ship an application" (Chen, 2008) but to identify the author of applications, all Android applications must be

digitally signed with a certificate whose private key is held by the application's developer. This certificate establishes trust relationships between applications and is not used to control which application the user can install. It does not need to be signed by a certificate authority and Android developers can use self-signed certificates (Signing Your Application, 2010). As a conclusion, in the application layer of Android including native and extended applications integrate, extend and modify the components are allowed without permission but any change needs to be digitally signed before installing on the platform. It is not defined in the documents of the platform whether a person needs to sign the change of components in the middleware or kernel layer for his personal purpose not for using in an application. This question is answered via interview and presented in section 4.5. Figure 5 shows the architectural openness model in Android based on the results of this study.

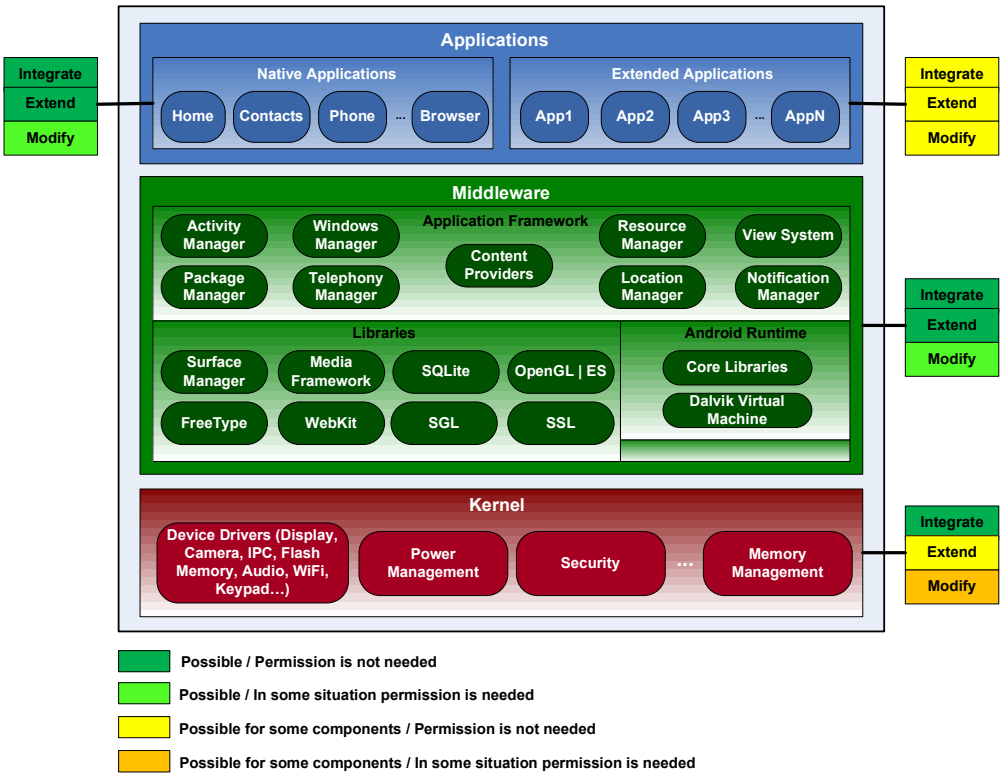

Figure 5. Architectural Openness Model in Android

4.4.2. iPhone

Introduction: In January of 2007, Apple announced iPhone at the MacWorld expo in San Francisco (Hall and Anderson, 2009). iPhone was created to be the operating system at the heart of iPhone smartphones and iPod touch devices and recently has been

launched on iPad tablet computers (Smith and Evans, 2010). The iPhone OS platform was built on Mac OS X. Despite its similarity to Mac OS X, there are some technologies available only on iPhone OS such as Multi-Touch interface (iPhone OS Overview, 2009). In July 2008, Apple launched its iPhone App Store as a marketplace for third-party developers to sell iPhone applications or offer them for free (Medford, 2008). According to Canalys research, in 2009 just two years after iPhone's unveiling, its share in the smartphone market was about 15% more than Windows Mobile share (Canalys research, 2010).

Architecture: iPhone software platform is built on a layered architecture described in the official website of Apple (iPhone OS Technology Overview, 2009). The architecture in the view of "architectural openness model" looks like Figure 6. In the application layer, there are some native applications such as calendar, photos, compass, etc settled by the platform supplier. Customers can install extended applications that are written by third-party developers and confirmed by platform supplier. In the middleware layer of iPhone, there are three sub-layers: Cocoa Touch, Media and Core Services (*ibid*). Cocoa Touch layer consists the key frameworks that provide infrastructure developers need to implement applications. The platform supplier advises developers to always start with this layer and drop-down to lower layers only as needed. In the Media layer there are graphic, audio and video frameworks for creating multimedia applications. Core Services layer provides the fundamental system services that all applications use either directly or indirectly via high-level frameworks. Address Book Framework, Core Data Framework, SQLite Library and Core Location Framework are some of the components of this layer. One layer downer than middleware layer is kernel layer of Core OS of iPhone, which acts as an intermediary between the iPhone hardware, and the higher level layers of the platform. It compromises Device Drivers, Security Framework, Accessory Framework, etc (*ibid*).

Openness Factors: The iPhone is a proprietary smartphone running a proprietary OS meaning Apple has the full control on its software and does not license iPhone OS to other device manufacturers. "Apple uses its OS to gain control over its product, and it sees iPhone and iPhone OS as a package in the smartphone competition" (Lin and Ye, 2009). In the following parts, the openness factors are discussed for the iPhone to see how much open its architecture is in more details.

Extended Applications: The iPhone software development kit (SDK) supports the creation of foreground applications that appear on the device's Home screen (iPhone OS Technology Overview, 2009) It means adding applications written by developers to the platform or in other words extending the "extended applications" layer is allowed. About integrating or modifying these applications the documents of the platform have not presented an explicit statement and it is evaluated via interviews presented in section 4.5.

Native Applications: The iPhone does not support the creation of background applications (*ibid*). It means that developers are not allowed to extend or modify the native applications. About integrating these applications there is no clear statement in the documents of platform and it is also evaluated via interviews.

Middleware: The same as the native applications, modifying the components of middleware layer is not supported by the iPhone. But developers can integrate the components of this layer by linking the code of the component statically in their application's executable files when building their project (*ibid*). About extending this layer the documents of iPhone do not present a clear statement and it is evaluated via interviews.

Kernel: Although the iPhone has suggested developers to use high-level frameworks over low-level frameworks, "the lower-level frameworks are still available for developers who prefer using them or who want to use aspects of those frameworks that are not exposed at the higher level. … iPhone OS provides a set of interfaces for accessing many low-level features of the operating system" (*ibid*). The term "many" means developers can not integrate all components of this layer in their applications and due to security purposes, "access to the kernel and drivers is restricted to a limited set of system frameworks and applications" (*ibid*). About extending or modifying the components of this layer there is lack of documentation. So it is evaluated by interviews as well.

Licensing status: All Applications must be signed with a certificate in order to be installed on registered devices. If a developer makes any changes to an application after submission to Apple, he must resubmit the application to Apple. After submission or resubmission of an application, it might be selected and digitally signed for distribution via the App Store or be determined that does not meet all or any part of the documentation or program requirements, or in the third case be rejected for distribution for any reason even if it meets the requirements (iPhone Developer Program License Agreement, 2009). An example for rejecting an application by Apple is calling a private

API in the application. “Private APIs refer to frameworks that are not accessible by Xcode” (Sadun, 2008). Xcode is an integrated development environment (IDE) provided by Apple. So as a conclusion, the licensing status of iPhone is very restricted and therefore if integrate, extend or modify of any of the platform layers is allowed, should be controlled and certificated before distributing.

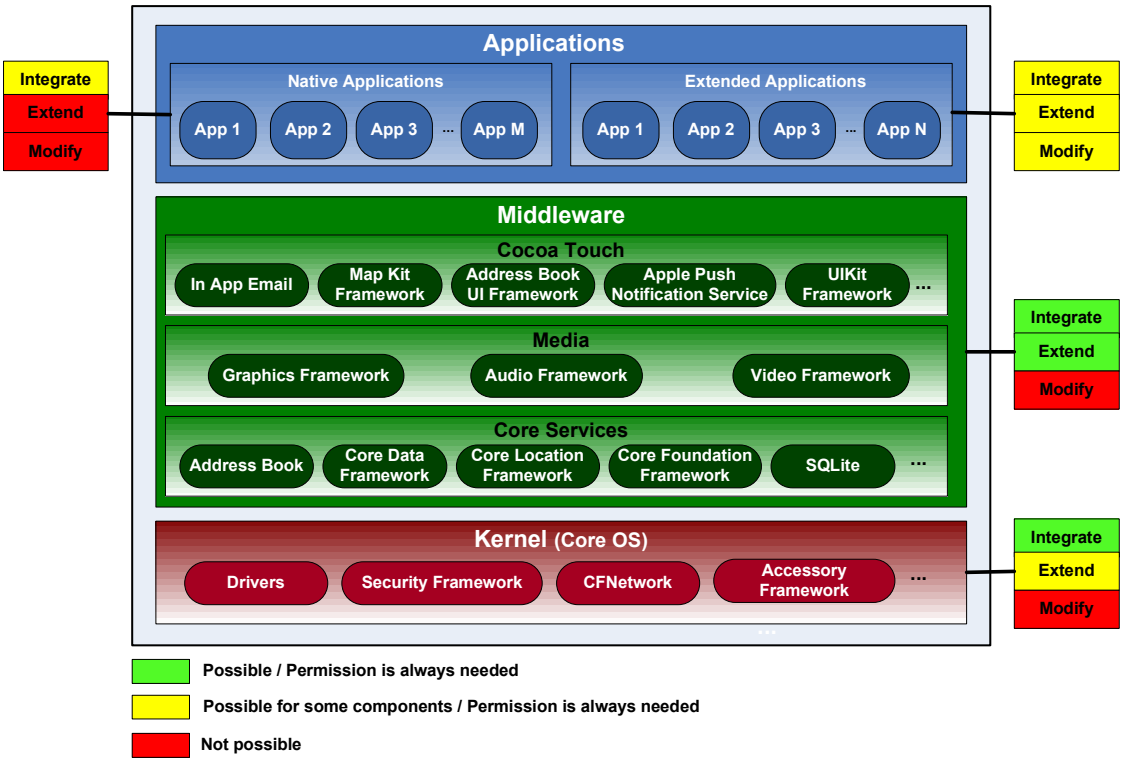

Figure 6. Architectural Openness Model in iPhone

4.4.3. Symbian

Introduction: Symbian OS is still at the highest position in the worldwide smartphone market (Canalys research, 2010). Its creator is Symbian Limited founded in 1998 by Ericsson, Nokia, Motorola and Psion. But in 2008, Nokia acquired all the remaining shares of Symbian Limited (Lin and Ye, 2009). Cho and Jeon in 2007 before releasing the Android believed that Symbain software platform is more open than other existing mobile software platforms (Cho and Jeon, 2007). After releasing the Android, Nokia wanted to compete with it in the openness degree (Hall and Anderson, 2009), therefore has formed Symbian Foundation in 2008. “One of the goals of the Symbian Foundation was to make the source code for the entire Symbian platform available to everyone, free of charge” (Symbian is Open Source, 2010). This goal has been reached on February

2010 by completion of releasing the entire platform as open source under Eclipse Public License (Symbian Foundation, 2010). The same as Android and despite iPhone, any device manufacturer can launch Symbian on its devices.

Architecture: Symbian is a layered software platform. According to the official website of Symbian, there are currently three layers within the main platform and 158 packages within the layers. The layers are application layer, middleware layer and OS layer (Symbian System Model, 2010). All 158 packages within three layers are represented in the official website of Symbian developer community (Symbian Foundation System Model, 2009). Each package has its specific owner which is responsible for the quality of the package and releasing the package to the Symbian Foundation source code (Foundation Builds, 2010). The three layers are similar to layers described in the architectural openness model. The result of looking at the platform by the architectural openness model is shown in Figure 7. In the native applications sub-layer there are the applications available as part of the Symbian platform, such as the organizer application suite, multimedia applications, telephony and IP applications, and applications for controlling device settings and so on (Symbian Foundation System Model, 2009). Within the extended applications sub-layer the users can install applications developed by third-party community. The middleware layer represents the functionality that is independent of applications. It provides services to applications and other higher-level programs by application programming interfaces (API). Services and frameworks in this layer can be specific application technology such as multimedia, or more general device services such as web services, security, IP services and so on. Finally the kernel layer includes the hardware adaptation layer (HAL) required to support a specific hardware platform and the Symbian kernel including physical and logical device drivers (*ibid*). There is also another layer called hardware adaptation which is outside the platform but depends on kernel layer (Software Structuring Principle, 2010).

Openness Factors: Before launching the Symbian Foundation, Symbian OS was not open source and only the licensees of the OS could get the source code (Sukanen, 2004). In that time the Symbian OS offered programming interfaces for developers to access different subsystems of the device (Sonera MediaLab, 2003) and it means, only the integration of components of the platform was allowed. After launching Symbian Foundation and therefore releasing the source code of the platform, the openness degree of the platform has been risen up and due to the claim of the official website of the Foundation the Symbian platform is now a free open-source software platform for mobile

devices. From the third Symbian platform release (Symbian^3) onwards developers can get any of the source, modify it, and contribute back the changes (Platform Opening/Get Started, 2010). To find out more about the current situation of openness in the platform, the openness factors for Symbian should be discussed for all three layers of the platform.

Extended applications, native applications, middleware, and kernel: The documents claim that you can download the whole source code of the platform or any individual package, and rebuild the platform or packages. Even in the kernel layer, there are some samples that show when a developer wants to build a package, he should implement some classes which are an abstract class. One example is the "power controller" class in "power management" framework inside the kernel layer of the platform (Symbian Power Manegement, 2010). It means that integrate, extend and modify of all three layers in the Symbian platform are allowed. To validate this statement a qualitative interview is conducted with a Symbian application developer and the results are presented in the section 4.5. The licensing status of the platform is presented concretely in the documents and discussed as follows.

Licensing status: "Source code is the life-blood of the Symbian platform, so we love source code contributions. We'd like to give everyone the freedom to check-in source code, but to ensure that the Symbian platform continues to be great for a long time in the future and to make sure we make best use of our valuable community, some checks and balances are required" (Contribution Process, 2010). This statement from the official website of the Symbian Foundation shows that the developers and users of the platform are free to do change in the source code of the platform but under some conditions and controls which are set by the platform supplier. They have categorized the contributions and the source code solutions to four types based on the size and complexity of the solutions. The categories are *fix* a defect, *enhance* the platform by a minor change, *extend* the source code by a major modification and *invent* a community based project on the platform. More complexity level of the contribution leads to additional process of controlling and therefore longer control time (*ibid*). It means that although the platform is an open source mobile platform, but control and licensing issues are restricted by the platform supplier.

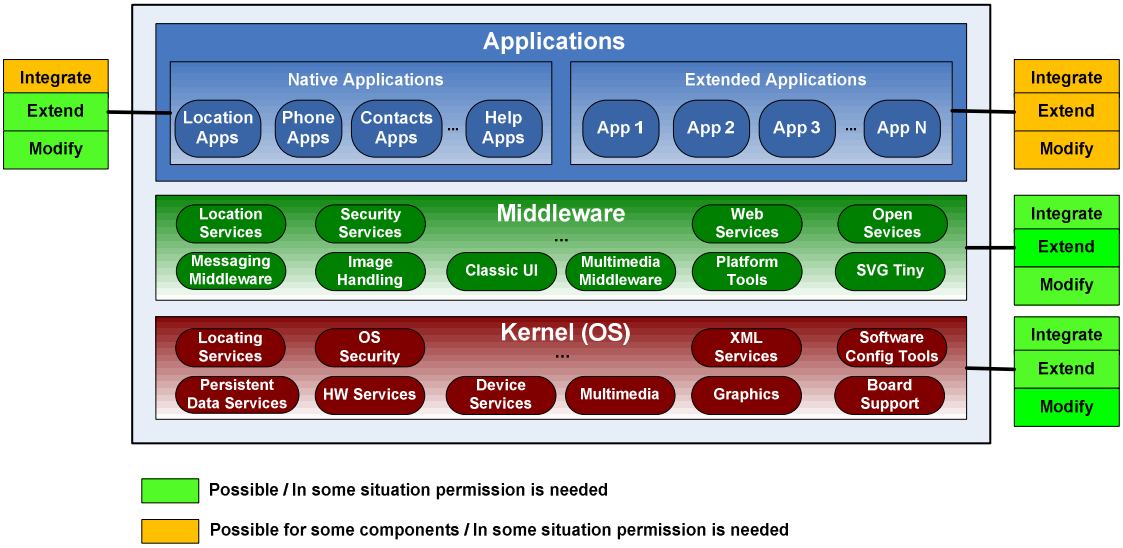

Figure 7. Architectural Openness Model in Symbian

4.4.4. Blackberry

Introduction:

"In 1999, Research In Motion (RIM) introduced the Blackberry which started as a simple two-way pager, but quickly became one of the most widespread of mobile computing devices … The device became so widely adopted, that PC magazine ranked it the 14th most important gadget invented in the past 50 years" (Hall and Anderson). Blackberry's share in the smartphone market in 2009 was almost 21% (Canalys research, 2010). The same as iPhone for Apple, Blackberry OS is also a proprietary OS for Blackberry phone devices and RIM does not not give license to any other device manufacturer to use the OS on their phone device. Blackberry App World is the official store for Blackberry applications which is governed by RIM.

Architecture:

There is no significant information about the structure of the platform in either in literature or in official documents of the platform. So the architecture of the platform is get from the interview with a Blackberry application developer. The same as other mobile platforms, Blackberry software platform has a layered architecture. In native applications layer, there are default applications set by RIM such as email, calendar, contacts, maps, browser, etc. The platform is open for users to install extended applications which are developed by third-party developers. In the middleware layer there are libraries and services developers can integrate them in their applications such as Java Virtual Machine.

Finally in the kernel layer there are core services like device drivers which are only accessible for middleware components.

Openness Factors:

Openness strategy of the platform is not discussed clearly in the documents of the platform. So the results for this part is gained from interview and presented in the interview section. Figure 8 demonstrates the architectural openness model of the platform based on the explanations about the architecture of the platform and the results of the interview.

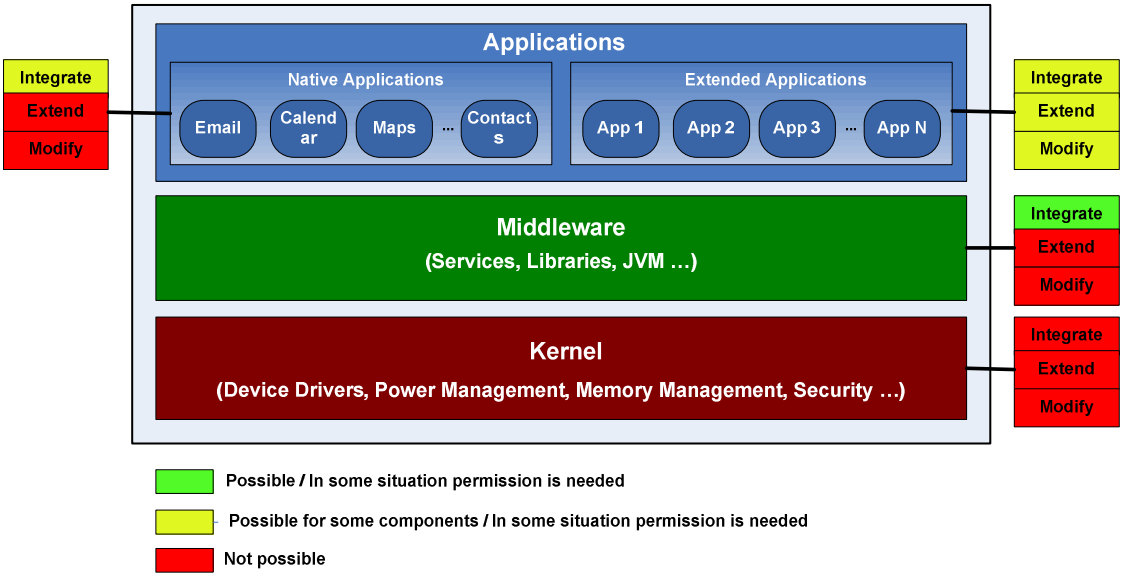

Figure 8. Architectural Openness Model in Blackberry

4.4.5. Windows Mobile

Introduction: Around the same time the Blackberry was gaining traction, Microsoft released its first OS targeted at the mobile device market (Hall and Anderson, 2009). Microsoft licenses Windows Mobile to any mobile phone maker who is interested in launching Windows Mobile on its device (Lin and Ye, 2009), therefore it is not a proprietary OS. The first version of the OS was under the name Pocket PC 2000 and current name of the OS which is planned to be released after Windows Mobile 6.5.5 and has been announced in February 2010 is Windows Phone 7 (Koh, 2010). The same as other main platforms, users can install applications of third-party on the Windows Mobile. Applications can be purchased from the Windows Marketplace for Mobile. For Windows Phone OS, Marketplace will be the only way to get applications (Patel, 2010).

According to Canalys research, Windows Mobile share in smartphone market is still decreasing and is almost 9% in the latest report (Canalys research, 2010).

Architecture:

Operating system of Windows Mobile is built based on Windows CE which is an operating system developed by Microsoft handheld computers and embedded systems. So the architecture of Windows Mobile is the same as Windows CE which according to the official website of the platform is a layered architecture includes application layer, operating system layer and OEM layer (Windows CE Architecture, 2010). The result of looking at the architecture of the platform by the architectural openness model is shown in Figure 9. In the native applications layers there are default applications set by Microsoft such as internet client services and user interfaces. Middleware layer includes core DLL, object store, multimedia technologies, etc. Kernel layer includes OEM adaptation layer (OAL), device drivers, boot loader, etc (*ibid*).

Openness Factors:

There same limitation in documentation of Blackberry is observed in Windows Mobile platform and there is not explicit statement to discuss the openness degree of different layers of the platform. However, there is some explanation in the documents of the Windows CE platform shows to some extend the platform is open for developers and device manufacturers to customize the operating system. For instance, platform builder is a tool introduced in the Microsoft development guide for customizing the Microsoft Windows CE (Platform Development, 2010). It seems that device manufactures and developers can modify, extend or completely replace various elements of Windows CE (*ibid)* but it is not determined whether the same openness is considered for Windows Mobile or not. So the openness degree of each layer of Windows Mobile platform is validated by a qualitative interview. Figure 9 is created based on the results of the interview.

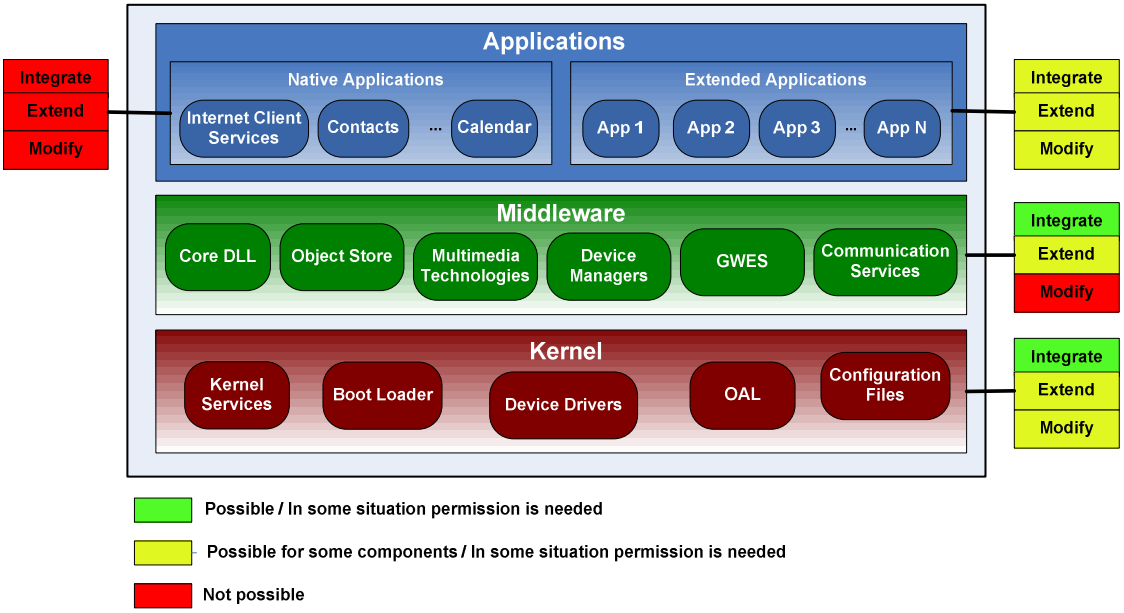

Figure 9. Architectural Openness Model in Windows Mobile

4.5. Qualitative Interviews Results

The aim of *architectural openness model* and *factors* is identification of accessibility in different architectural layers of a mobile platform from the highest level (application layer) to the lowest level (kernel layer) and also finding out the licensing aspects of platforms. The goal of interviews with application developers of main mobile platforms is confirming this identification based on developers' experiences. The following results are gained from the interviews.

4.5.1. Relationship of Openness in a Mobile Platform with Architectural and Licensing Aspects of the Platform

All of the interviewees believe that architectural and licensing aspects of a mobile platform affects its openness degree. One of them counts the programming language, the way libraries of a platform work, if the frameworks are object oriented or not and the SDKs provided to extend the platform as the architectural aspects of the platform and believes that "a lot of openness depends on the architecture". Another interviewee thinks that if the architecture of a platform is messy it will be hard to make it open and on the other hand, if the architecture is modular and has layers and subsystems then it will be easy to open it up. In their point of view, the licensing aspects also affect the openness of

a platform since to access the different layers of the architecture of a layer or to modify the components of a platform you need to have a license.
But in their opinion, the architectural and licensing aspects are not the only parameters that affect the openness degree of a platform. “You can do a lot with architecture to open a platform but it is not the only aspect”, one of the developers believes. He explains that the available samples and examples for developers, the provided community and the platform documentation are some other aspects that affect the openness of a platform.

4.5.2. Importance of Openness in a Mobile Platform for Developers

Overall, for most of interviewees the openness degree of a mobile platform is not such important and they care less about the openness of a platform. Since they are commercial developers, for choosing a platform the financial aspects of the platform is more important and determinant for them. One of them believes that managers should more care about openness and they should be aware what is allowed to do and what is not allowed. Another explains that if a platform is open enough to make a lot of money for him, then the platform is interesting for him.
Nevertheless, openness in higher levels of a mobile platform is a considerable aspect for interviewees. For example, although Apple has supported some APIs in the kernel layer of the iPhone and RIM has not supported any API in the kernel layer of the Blackberry, one of the interviewees considers the Blackberry more open than the iPhone, because native applications in the Blackberry are allowed to integrate but in the iPhone are not. All of the interviewees are more confident about the openness degree of higher layers in their favorite platform than the lower layers. They are not sure about kernel layer and most of them about middleware layer because they have not tried to integrate, extend or modify components of these layers. One of them explains that “the middleware you want to integrate is huge, so … you as a business developer do not care about extending or modifying it”. Most of them believe that even if a platform opens up its middleware or kernel layer, the people who benefit from the openness of the platform and are interested to access or modify the components of the lower layers are device manufacturers not application developers. All of the interviewees are almost satisfied by current openness degree of their favorite platforms and just one of them would like to see a small part of his favorite platform to see opened up which is Wi-Fi library in Blackberry.

4.5.3. Openness in Main Mobile Platforms

Openness in Android: For the Android, as mentioned before, an Android developer who could be contacted via a qualitative interview was not available. So the results are gained from online discussion and also some interviewees who have some experiences about Android.

Although it is difficult to commit modifying of lower layers of the platform as Google controls it closely, but it is possible for a developer to code in any of the layers, one of the Android developers explains online. He clarifies more his opinion about openness in Android by separating the situation in theory and in reality based on his experience: “In theory, Android is open source. If a developer wanted to make changes to something in the Kernel, they could submit it to Google who manages the project, and Google would approve the change, and allow it into the framework. In practice, I think it is difficult for anyone outside of Google to get changes into the core of the framework”. He explains that for instance replacing the Dalvik Virtual Machine would be difficult, as it is the core of the middleware framework. But another interviewee who is mainly a Blackberry developer but has some experiences in Android thinks that developers can replace Dalvik Virtual Machine or SQLite library by their own libraries. All of them believe that even if in practice Android is totally open and everyone can modify the source, but this openness makes sense for device manufacturers not commercial developers.

Figure 5 shows the architectural openness in Android which is acquired from literature and documentation of the platform and online discussion.

Openness in iPhone: In general, Apple has published some public APIs for the iPhone that are the access points of the platform and developers can integrate them in their applications, as the iPhone application developer explains. There are also some private APIs which, if developers use them in their applications, the applications will be rejected when developers want to publish them on AppStore.

To discuss specifically about accessibility of each of architectural layers of the platform, the interviewee explains each layer according to his experiences. “So for really small and specific parts of native application you can integrate them and I would not say you can integrate native applications” (or you can integrate just 5 or 10 percent of them such as AppStore application). “You cannot extend the native applications and add functionality to them”. About extended applications, depends on how they operate, developers can integrate them into their applications. “For example you can open a twitter application on

the phone and send a message to twitter from your application". They might be some open source applications on AppStore which developers can get them, integrate them in their applications, extend or modify them and submit them again into AppStore. One example is a framework which Facebook application uses it and is an open source framework. In the middleware "often you would be able to extend the components of this layer". Although you do not have the source of the APIs but you can add category of methods to the classes and subclass everything you want to. "Then you can recompile it, use it in your application and submit it to AppStore without any problem … There are some exceptions that you should not do this or better not to do this". The components of this layer are not modifiable and developers cannot change them. And finally in the kernel, developers can integrate the components of the layer, but it is less important for developers and they less often use this layer. It is hard to extend this layer because it is written in C language. But still there are some components although it is hard to extend, but developers can add new functions to them. For instance the core foundation network library which is used to get data from URLs, developers can extend it and put a layer on top. The same as middleware layer the components of this layer are not possible to be modified. As a conclusion, the interviewee believes that developers and users are not allowed to extend or modify the native applications, or modify the middleware and kernel layer. They are allowed to integrate and extend all of the components of the middleware and integrate all of the components of the kernel layer and extend most of the components of this layer. And about extended applications everything depends on the application.

To discuss more about the openness degree of the iPhone, the interviewee also describes the licensing status of the platform. He explains that "when you want to publicize an application, Apple will do a quality testing, to make sure it does not crash, does not use internet connection gracefully, you do not use anything there is copyright on, and on more technical phase, to check if you have used APIs that have make public and documented. ... If you use an API which is not documented your app will be rejected. So you know it and you do not want to get permission".

Figure 6 shows the architectural openness in the iPhone which is gained from literature and confirmed by the interview.

Openness in Symbian: The interviewee of Symbian part which is a Symbian application developer explains about openness degree of the platform by describing the accessibility

of each layer. Extended applications seem to be like iPhone and access to them depends on the way the applications support. “If other applications support an open architecture, you can use them”. He believes that extend or modify them is almost impossible. In native applications, those applications that have API, developers can integrate them. Contacts application is one of them but all of the applications of this layer do not have API and it is confined to some of them. He thinks that although Symbian Foundation has released the whole source code of the platform, but “if you get the source code of the applications from Symbian foundation, modify them and submit them to the platform, they will be rejected”. So in his point of view native applications are only possible to be integrated, not extended or modified. In the middleware layer, “everything which is documented and has public API you can integrate”. The interviewee has not used all of the components of this layer but he thinks developers can extend and subclass most of the components and modifying is almost not allowed. About kernel layer, he also thinks although it is open but only manufacturers benefit from this openness and “integrate, extend or modify this layer is pointless for developers”. The reason is that “if you as a commercial developer change a component of the kernel and use the new component in your application, then all of the people who want to use your application should have your new version of the kernel on their phone and it is not possible”. He also mentions another reason for willingness of developers to extend or modify the middleware or kernel of the platform. He explains that with use of engines, developers can have their own for example middleware and integrate it to their application, but since middleware of Symbian works better so developers are not interested to change the middleware or kernel of the platform. He concludes that regarding to releasing the source code of the platform by Symbian Foundation, “you as a person who is interested, can change most of the components of the phone for yourself, but it does not have sense for commercial developers. It is more interesting for phone manufactures”.

The interviewee also explains the licensing status of the platform. “If you want to publish your application you need to submit it to the platform and sign your application. The sign process is easy. You can do it on Symbian website. Symbian will check technical aspects to see if you have used API properly”. If a device user install an application which is not signed, every time the application wants to use an API of the platform the user should confirm that it is true and application is allowed to use the API. Besides this reason, other

developers can steal an application which is not signed. So developers sign their applications in Symbian.
As a summary, Figure 7 shows the architectural openness in Symbian which is figured out from literature and confirmed by the interview.

Openness in Blackberry: The Blackberry application developer who is the interviewee for Blackberry part initializes the discussion by a general statement: "About blackberry, if my boss comes to my desk and says would it be nice if we have stuff like x, y or z in the application, I will probably say sure no problem then I will find APIs … and then I can just develop stuffs that I would want to do". He adds that there is one exception, "Wi-Fi access point is not available in library".
Then he goes further through different layers of the application from the lowest one to the highest. "In Blackberry you cannot do anything with Kernel layer and everything which is provided in kernel is abstracted for used by Middleware layer". He believes that as a business developer nobody cares about kernel because every service a developer needs is supported by middleware. The middleware which is possible to be integrated is huge, and again he believes that nobody care about extending or modifying it. In native applications, there are pretty much applications which have API and developers can integrate them such as map application. Developers cannot extend or modify native applications since there is no source code of them, but it is possible to extend or modify their appearance via the user interface. About extended application the story is the same as other platforms. If the application is open source or supports an API, developers can integrate or modify it. He believes that there is some open source stuff in application store but not a lot of them.
Finally he explains about licensing situation of the platform. A user can install an unsigned application on Blackberry. "But if the application uses protected features such as Location or Radio Access it needs to be signed". Since it is hard for a developer to just use unrestricted APIs, usually they need to sign their applications. Figure 8 demonstrates the architectural openness in Blackberry platform, which is gained from literature and confirmed by the interview.

Openness in Windows Mobile: In case of Windows Mobile also, the interviewee who is a Windows Mobile application developer explains about the openness degree of the platform by describing the accessibility of different layers of the platform. He starts to

describe the layers from the lowest one to the highest: “You can create your own driver. I do not have experience about other components of the kernel but I think you are not allowed to modify power management and other components, but they have API and you can only use them. You can extend some components of the middleware like GPS library, but you cannot modify them. You cannot integrate or extend or modify the native applications in Windows Mobile. For extended applications, if developers support the source code or the API you can extend, integrate or modify the application but usually it does not happen”. He also mentions the licensing situation of the platform. “For some mobile devices you need to sign the applications and for some you do not need”. If a developer wants his application to be signed or certified by Microsoft, he would have to explain why he is change or extending some. If integrating, extending or modifying the components is not done in a proper way, the application will be rejected. Figure 9 shows the architectural openness in Windows Mobile, which is gained from documents and literature and confirmed by the interview.

5. Analysis

The results of the research process show the different openness strategies in the main platforms. According to literature and documents, the Android and Symbian platforms are almost completely open since Google and Symbian Foundation have released the whole source code of the platforms and device manufacturers, developers and users can download the source code and do everything they want with the components of each layer of the platforms. In this case even Symbian is more open since there is no limitation set by Symbian Foundation about integrating, extending or modifying of any components of the platform, where in the kernel layer of the Android some components, like power management which users are only allowed to integrate it and not extend or modify. But the situation in practice is different and the experiences of developers show that these two platforms are not as open as they seem, since there are some controls governed by Google and Symbian Foundation which do not allow commercial developers to make every desired change to the platform and the target people of releasing the source code are device manufacturers like HTC. In this case Android is more open since there is less restriction set by Google for submitting an extension or modification of the platform. However, even in practice Android and Symbian are the most open platforms among current main mobile platforms, although they are not completely open. After these two

platforms, Windows Mobile and Blackberry are situated near together. There are not enough materials on the documentation of these platforms to discuss about accessibility to different layers of the platforms, but according to the experiences of the developers, Windows Mobile is more open than Blackberry since kernel layer in Blackberry is almost close and users cannot access to it directly but in Windows Mobile users even can write their own drivers for the device. But on the other hand native applications users can do more with Blackberry than Windows Mobile. In Windows Mobile, even integrate the native applications is not possible, but in Blackberry some native applications that have APIs can be used by users. The licensing situation for both platforms is almost the same and in both cases there is some situation that people can use unsigned applications. The big difference here is that Windows Mobile can be installed on different devices, but Blackberry is a proprietary software platform that is installed only on Blackberry devices. So totally the conclusion is that Windows Mobile is more open than Blackberry. And finally in the spectrum of openness, after these two platforms the iPhone is situated which the least open platform among the main mobile platforms. Although developers can integrate the kernel layer of the platform in their applications and in this sense it is more open than Blackberry, but on the other hand developers can integrate several native applications in the Blackberry which is almost not possible for any native applications of the iPhone. So in the case of accessibility of architectural layers, Blackberry and iPhone are situated nearly in the spectrum but the main thing that distinguish the iPhone from Blackberry and brings it to the end of the spectrum is licensing situation which is controlled for the iPhone and restriction set by Apple is much more than other platforms since every application before submitting to the application store should be signed by Apple and the process of quality testing of the application is strongly controlled. Table 3 summarizes the comparison of openness strategy in the main mobile platforms based on the architectural aspects and licensing situation of the platforms.

Table 3. Comparison of Openness Strategy in the Main Mobile Platforms

Platform	Factor	Possibility statuses	If possible→Licensing statuses
Android	Integrate extended applications	Possible for some components	Permission is not needed
	Extend extended applications	Possible for some components	Permission is not needed
	Modify extended applications	Possible for some components	Permission is not needed
	Integrate native applications	Possible	Permission is not needed
	Extend native applications	Possible	Permission is not needed
	Modify native applications	Possible	In some situation permission is needed
	Integrate middleware	Possible	Permission is not needed
	Extend middleware	Possible	Permission is not needed
	Modify middleware	Possible	In some situation permission is needed
	Integrate kernel	Possible	Permission is not needed
	Extend kernel	Possible for some components	Permission is not needed
	Modify kernel	Possible for some components	In some situation permission is needed
Symbian	Integrate extended applications	Possible for some components	In some situation permission is needed
	Extend extended applications	Possible for some components	In some situation permission is needed
	Modify extended applications	Possible for some components	In some situation permission is needed
	Integrate native applications	Possible for some components	In some situation permission is needed
	Extend native applications	Possible	In some situation permission is needed
	Modify native applications	Possible	In some situation permission is needed
	Integrate middleware	Possible	In some situation permission is needed
	Extend middleware	Possible	In some situation permission is needed
	Modify middleware	Possible	In some situation permission is needed
	Integrate kernel	Possible	In some situation permission is needed
	Extend kernel	Possible	In some situation permission is needed
	Modify kernel	Possible	In some situation permission is needed
Windows Mobile	Integrate extended applications	Possible for some components	In some situation permission is needed
	Extend extended applications	Possible for some components	In some situation permission is needed
	Modify extended applications	Possible for some components	In some situation permission is needed
	Integrate native applications	Not possible	
	Extend native applications	Not possible	
	Modify native applications	Not possible	
	Integrate middleware	Possible	In some situation permission is needed
	Extend middleware	Possible for some components	In some situation permission is needed
	Modify middleware	Not possible	
	Integrate kernel	Possible	In some situation permission is needed
	Extend kernel	Possible for some components	In some situation permission is needed
	Modify kernel	Possible for some components	In some situation permission is needed
Blackberry	Integrate extended applications	Possible for some components	In some situation permission is needed
	Extend extended applications	Possible for some components	In some situation permission is needed
	Modify extended applications	Possible for some components	In some situation permission is needed
	Integrate native applications	Possible for some components	In some situation permission is needed
	Extend native applications	Not possible	
	Modify native applications	Not possible	
	Integrate middleware	Possible	In some situation permission is needed
	Extend middleware	Not possible	
	Modify middleware	Not possible	
	Integrate kernel	Not possible	
	Extend kernel	Not possible	
	Modify kernel	Not possible	
iPhone	Integrate extended applications	Possible for some components	Permission is always needed
	Extend extended applications	Possible for some components	Permission is always needed
	Modify extended applications	Possible for some components	Permission is always needed
	Integrate native applications	Possible for some components	Permission is always needed
	Extend native applications	Not possible	
	Modify native applications	Not possible	
	Integrate middleware	Possible	Permission is always needed
	Extend middleware	Possible	Permission is always needed
	Modify middleware	Not possible	
	Integrate kernel	Possible	Permission is always needed
	Extend kernel	Possible for some components	Permission is always needed
	Modify kernel	Not possible	

This study has also some implicit results which are achieved from the interviews with developers. The interviews show that most of developers, which are typically commercial developers, do not care about architectural openness of a platform. For developers the tools and languages are supported to develop applications, guides and documentation for developing and financial aspects of the platforms are more considerable. When they want to consider the architectural openness of a platform they more care about higher layers of the platform. The reason is that first of all they think even if the platform is mostly open, the lower layers provided by the platform supplier work well and they do not need to spend time to extend or modify it. The second reason is that if they modify lower layers such as components of the middleware or kernel, the application users need also to change the middleware or kernel of their devices which is not a practical work. So in their point of view increasing the openness of a platform and releasing the source code of lower layers make sense for device manufacturers who want to customize the platform for their own devices. This result brings some limitation for this study which is argued in the discussion section.

6. Discussion

As previous sections show, the main result of this thesis is the identification of openness degree in the main mobile platforms based on the architectural aspects and licensing situation of the platform. This results are gathered by looking through the lens of architectural openness model at the architecture of the platforms, which is typically a layered architecture, and accessibility of each layer in the architecture. Licensing situation is another important factor to determine the openness strategy of a platform that is studied in the literature and added to the architectural aspects of the platform. All of these results are confirmed by some interviews with developers of the platforms.

The first limitation of this research is realized in the literature review stage. As discussed in related work part, although there is some literature compares most of studied mobile platforms of this research together, even regarding the openness of the platforms, but none of them discuss the openness strategy of the platforms based on technical aspects such as architecture of the platforms or software access points of the platform. Even literature about general software platforms do not discuss the openness of platforms based on the architectural aspects. The architectural openness model developed in this study is a tool to aim identification of the openness strategy in mobile platforms based on

their architectural aspects. Despite this model is based on typical layered architecture of mobile platforms and is applicable to all main mobile platforms, but the efficiency of the model to identify the openness strategy of the mobile platforms would be confirmed by an empirical study which is not done here due to restricted time of the research. The recommended way of confirming the model is conducting some qualitative interviews with some experts in software architecture and software openness areas and applies their feedback to enrich the model and increase its efficiency. Another boundary in literature review part is incompleteness of documents of the mobile platforms to finding out the openness strategy and access points of the platforms. It is not easy to contact technical engineers and architects of the companies like Google, Microsoft, RIM, etc. but if they accepted to be interviewed, the results of the study would be more reliable.

The second limitation of this study as shown in results of the interviews is that architectural openness of the studied platform is not important for interviewees which are commercial developers. Since they do not care about the openness especially in lower layers of the platforms, they have not tried to extend or modify the components of the lower layers. As a result they are not certain about accessibility of all layers of their favorite platform and it could affect the reliability of the results. One possible solution for this issue is to interview with more application developers for each platform. But as the current interviewees claim, generally commercial developers care less about the openness in a mobile platform and if they want to consider the openness of a platform, they will focus on the higher layers of the platform. On the other hand, as interviews show, device manufactures would benefit more from openness in mobile platforms since they can customize even lower layers of open platforms like the Android and Symbian and to some extend Windows Mobile for using the modified platform in their devices. So the better improvement of performing the research method would be conducting interviews with some technical engineers in device manufacturers if it is not difficult to contact them.

7. Conclusions

The overall aim of this research was identification of openness strategy in mobile platforms based on the software architecture of the platforms. This section revisits the specific objectives of this research, summarizes the findings of the research, offers

conclusions based on the findings, and finally suggest some recommendations for future works based on the limitations of the research work.

7.1 Research Objectives: Summary of Findings and Conclusions

Research Objective 1: Building a model to describe the architectural openness of mobile platforms

Architectural openness model is built based on a typical layered architecture which is applicable to all main mobile software platforms and compromises applications layer, middleware layer and kernel layer. Applications layer itself includes two parts, extended applications and native applications. Besides the layered architecture, the model shows three way of accessibility to the platform for each layer. These ways are integrating, extending or modifying a layer.

Research Objective 2: Defining architectural openness factors by considering the licensing aspects of mobile platforms in the model

Although the model shows the platform access and extension methods in different levels, but to demonstrate the openness strategy of a platform, licensing aspects of the platform should also be considered. By considering licensing situation of mobile platforms, some factors are defined to identify openness strategy of platforms and presented in table 2. For example for kernel layer of a platform, the result of applying the openness factors can be: Modifying the kernel layer is possible for some components of the layer but it always needs permission from the platform supplier.

Research Objective 3: Looking at main mobile platforms by the lens of developed model and factors to determine how open the platforms are

The openness strategy of main mobile platforms include Android, Symbian, iPhone, Windows Mobile and Blackberry is discussed by applying the architectural openness model and factors in the architecture and licensing aspect of the platforms which are gained from the literature. The results are demonstrated by a figure for each platform. Due to insufficiency of literature and documents of the platforms, some factors needs to be confirmed by interviews.

Research Objective 4: Conducting some qualitative interviews with application developers of the platforms to confirm the results of previous step

For each platform except the Android, one qualitative interview with an application developer of the platform is conducted. The developers are asked to explain about the

openness of their favorite platform based on the accessibility of each architectural layer of the platform and also licensing situation of the platform. The results show that for some layers especially lower ones, commercial developers are not sure about openness degree of the platform because they do not care about the openness in the lower layers and have not tried to extend or modify them. But about the higher layers and licensing situation of the platform the results of the interviews are valuable. In some cases the results confirmed the previous results gained from documents of the platforms, and in some cases the experience of developers are different from results of literature. So they believe that even for platforms like Android and Symbian, the openness degree in the reality is fewer than what is claimed in the theory.

Finally a comparison of openness strategy in five main mobile platforms is presented in this thesis which is based on the results of looking at the platforms by lens of architectural openness model and factors and evaluation of results by conducting interviews.

7.2 Recommendations for Future Works

As argued in the discussion section insufficiency of literature and documents, lack of time to do more interviews with other application developers and interview with some engineers from devices manufacturers, restrictions on access to technical engineers in platform suppliers such as Google and Microsoft are the main limitations of this research and affect the reliability of the results. The architectural openness model and factors are valuable results of this research. It is recommended, for increasing the reliability of the model and factors, that some expert people from software architecture and software openness areas also be interviewed. For improving the validation of openness strategy of mobile platforms, interview with technical engineers of mobile device manufactures is also recommended.

Acknowledgements. The author would like to thank Slinger Jansen for his helpful advices. His support has directed the research and positively influenced the quality of the results of this thesis and was impossible without that contribution. Furthermore, the author would like to thank Alan Carlson for his valuable comments during the research. Finally, the author would like to thank the mobile application developers for their contribution in the qualitative interviews.

8. References

Alexandrov, A.D., Ibel, M., Schauser, K.E. and Scheiman, C.J. (1997). *Extending the operating system at the user level: the ufo global file system*, In 1997 Annual Technical Conference On Unix and Advanced Computing Systems (USENIX' 97).

Android Market, (2008). [Online] Available at: http://market.android.com/publish, Last retrieved: 2010-04-13.

Android Power Management, (2010). In Android Platform Developer's Guide. [Online] Available at: http://pdk.android.com/online-pdk/guide/power_management.html, Last retrieved: 2010-04-13.

Arief, B. Gacek, C. and Lawrie, T. (2001). *Software Architectures and Open Source Software – Where can Research Leverage the Most?*, In 1st Workshop on Open Source Software Engineering:Making Sense of the Bazaar (part of the 23rd ICSE) : 3–5.

Alspaugh, T.A., Asunction, H.U. and Scacchi, W. (2009). *Analyzing Software Licenses in Open Architecture Software Systems,* In FLOSS '09: Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development: 54–57.

Bass, L., Clements, P. and Kazman, R. (2003). *Software Architecture in Practice,* Second Edition. Addison-Wesley, Boston: p 21.

Biggam, J. (2008). *Succeeding with Your Master's Dissertation: A Step-by-Step Handbook,* Open University Press, Berkshire, England: 138-143.

Bring Up, (2010). In Android Platform Developer's Guide. [Online] Available at: http://pdk.android.com/online-pdk/guide/bring_up.html, Last retrieved: 2010-04-13.

Britten, N. (1995). *Qualitative interviews in medical research*. BMJ 311:251–3.

Brown, A.W. and Booch, G. (2002). *Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors,* In Proceedings of 7th International Conference on Software Reuse, LNCS, Vol. 2319. Springer: 123-136.

Bryman, A. and Bell, E. (2007). *Business Research Methods,* Oxford: Oxford University Press: p 104.

Burnard, P. (1991), *A Method of Analysing Interview Transcripts in Qualitative*

Research, Nurse Education Today 11: 461-466.

Buschmann F., Meunier R., Rohnert H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley & Sons: p 31.

Canalys research release 2010, (2010). [Online] Available at: http://www.canalys.com/pr/2010/r2010021.html, Last retrieved: 2010-04-21.

Chen, J. (2008). *An Introduction to Android,* [Online] Available at: http://sites.google.com/site/io/an-introduction-to-android, Last retrieved: 2010-04-09.

Childers, B. (2009). *Android Everywhere!,* Linux Journal, 2009(186).

Cho, Y.C. and Jeon, J.W. (2007). *Current Software Platforms on Mobile Phone*, International Conference on Control, Automation and Systems. Seoul: 1862-1867.

Constantinou, A. (2008). *Mapping open source into mobile: who, where and how,* VisionMobile Ltd., available via: http://www.visionmobile.com/blog/2008/12/mapping-open-source-into-mobile-who-where-and-how/, Last retrieved: 2010-02-19.

Contribution Process, (2010). In Symbian Developer Community, [Online] Available at: http://developer.symbian.org/wiki/index.php/Contribution_Process, Last retrieved: 2010-04-23.

Dalvik, (2010). In Android Platform Developer's Guide. [Online] Available at: http://pdk.android.com/online-pdk/guide/dalvik.html, Last retrieved: 2010-04-13.

Fleming, R. (2010). *Google Android Powered TV Coming in Fall*, [Online] Available at: http://www.digitaltrends.com/home-theater/google-android-powered-tv-coming-in-fall/?news=123, Last retrieved: 2010-04-09.

Foundation Builds, (2010). In Symbian Developer Website, [Online] Available at: http://developer.symbian.org/wiki/index.php/Developer_Guidelines:_Foundation_Builds, Last retrieved: 2010-04-23.

Hall, S. P. Anderson, E. (2009). *Operating Systems for Mobile Computing*. Consortium for Computing Sciences in Colleges, USA.

Hashimi, S.Y. and Komatineni, S. (2009). *Introducing the Android Computing Platform*, In: Pro Android, Apress: 1-19.

iPhone Developer Program License Agreement, (2009). [Online] Available at: http://www.eff.org/files/20100302_iphone_dev_agr.pdf, Last retrieved: 2010-04-20.

iPhone OS Overview, (2009). In iPhone OS Reference Library, [Online] Available at: http://developer.apple.com/iphone/library/referencelibrary/GettingStarted/URL_iPhone_OS_Overview/index.html, Last retrieved: 2010-04-13.

iPhone OS Technology Overview, (2009). In iPhone OS Reference Library, [Online] Available at: http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html, Last retrieved: 2010-04-15.

Jansen, S., Brinkkemper, S., Hunink, I., Demir, C. (2008). *Pragmatic and Opportunistic Reuse in Innovative Start-Up Companies*, IEEE Software, 25(6): 42-49.

Jansen, S., Finkelstein, A., and Brinkkemper, S. (2009). *Business network management as a survival strategy: A tale of two software ecosystems.* In Proceedings of the First Workshop on Software Ecosystems. CEUR–WS, vol. 505.

Klatt, B. (2008). *Software Extension Mechanisms*, Fakultt fr Informatik,. Karlsruhe, Germany, Interner Bericht 2008-08. [Online]. Available: http://www.bar54.de/benjamin-klatt-software-extension-mechanism.pdf, Last retrieved: 2010-04-23.

Koh, D. (2010), *Microsoft integrates Xbox Live and Zune with Windows Phone 7 Series,* [Online] Available at: http://asia.cnet.com/reviews/mobilephones/0,39050603,62061210,00.htm, Last retrieved: 2010-04-26.

Kvale, S. (1996). InterViews: *An Introduction to Qualitative Research Interviewing*, Sage Publications: p 2.

Lamothe, A. (2006). *Degrees of Openness,* [Online] Available at: http://linuxdevcenter.com/pub/a/linux/2006/11/09/degrees-of-openness.html, Last retrieved: 2010-05-20.

Lin, F. and Ye, W. (2009). *Operating System Battle in the Ecosystem of Smartphone Industry,* In Proc. of 2009 International Symposium on Information Engineering and Electronic Commerce: 617-621.

Live from Google I/O – Android: Integrate, Replace and Extend, [Online] Available at: http://mobilementalism.com/2008/05/28/live-from-google-io-android-integrate-replace-and-extend/, Last retrieved: 2010-04-09.

Maxwell, E. (2006). *Open Standards, Open Source,and Open Innovation: Harnessing the*

Benefits of Openness, Innovations: Technology, Governance, Globalization 1(3): 119-176.

Medford, C. (2008). *Apple, Nokia Battle for Mobile Startups,* [Online] Available at: http://www.redherring.com/Home/24459, Last retrieved: 2010-03-13.

Nakagawa, E. Y., Souza, E. P. M., Murata, K. B., Andery, G. F., Morelli, L. B., Maldonado, J. C. (2008). *Software Architecture Relevance in Open Source Software Evolution: A Case Study,* In Proc. of Annual IEEE International Computer Software and Applications Conference: 1234-1239.

Open Handset Alliance – Overview, (2007). [Online] Available at: http://www.openhandsetalliance.com/oha_overview.html, Last retrieved: 2010-04-09.

Oxford English dictionary – *openness definition*, (2010), [Online] Available at: http://dictionary.oed.com/cgi/entry/00332458?single=1&query_type=word&queryword=openness&first=1&max_to_show=10, Last retrieved: 2010-04-07.

Patel, N. (2010). *Confirmed: Marketplace will be the only way to get apps on Windows Phone 7 Series*, [Online] Available at: http://www.engadget.com/2010/03/15/confirmed-marketplace-will-be-the-only-way-to-get-apps-on-windo/, Last retrieved: 2010-04-26.

Platform Development, (2010). [Online] Available at: http://msdn.microsoft.com/en-us/library/ms902061.aspx, Last retrieved: 2010-05-22.

Platform Opening, Get Started, (2010). In Symbian Developer Community, [Online] Available at: http://developer.symbian.org/wiki/index.php/Platform_Opening/Get_Started, Last retrieved: 2010-04-21.

Prehn, S. (2007). *Open Source Software Development Process,* Term Paper in AG Software Engineering Seminar SS07.

Raymond, E.S. (2001). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary,* Revised Edition, O'Reilly and Associates, Inc.

Sadun, E. (2008). *Unpublished iPhone API Crackdown in Progress,* [Online] Available at: http://arstechnica.com/apple/news/2008/12/private-iphone-api-crackdown-in-progress.ars, Last retrieved: 2010-04-20.

Salipante, P., Notz, W. and Bigelow, J., (1982). *A matrix approach to literature reviews,* In: *Re*search *in Organizational Behavior,* Vol. 4. Jai Press, CT: 321-348.

Scacchi, W. (2004). *Free and Open Source Development Practices in the Game Community,* IEEE Software, 21(1): 59-67

Sim, N., Turnbull, R. and Walker, M.D. (2006). *Open devices — their role in supporting converged services,* BT Technology Journal, 24(2): 200-204.

Signing Your Application, (2010). In Android Developer's Guide. [Online] Available at: http://developer.android.com/guide/publishing/app-signing.html, Last retrieved: 2010-04-13.

Smith, C. and Evans, B. (2010). *Apple Launches iPad*, [Online] Available at: http://www.apple.com/pr/library/2010/01/27ipad.html, Last retrieved: 2010-04-13.

Sonera MediaLab, (2003). *Symbian Application Development Wall Paper*, Available at: http://www.medialab.sonera.fi/workspace/SymbianAppDevelopmentWhitePa.pdf, Last retrieved: 2010-04-21.

Sukanen, J. (2004). *Extension framework for Symbian OS applications*, Helsinki University of Technology.

Symbian Foundation, (2010). *Symbian Completes Biggest Open Source Migration Project Ever*, [Online] Available at: http://www.symbian.org/news-and-media/2010/02/04/symbian-completes-biggest-open-source-migration-project-ever, Last retrieved: 2010-04-20.

Symbian Foundation System Model, (2009). In Symbian Developer Community, [Online] Available at: http://developer.symbian.org/downloads/system_models/foundationpkg_22-05-09.svg, Last retrieved: 2010-04-21.

Symbian is Open Source, (2010). [Online] Available at: http://www.symbian.org/symbian-feature-set/symbian-is-open-source, Last retrieved: 2010-04-20.

Symbian Power Management, (2010). In Symbian Developer Community, [Online] Available at: http://developer.symbian.org/main/documentation/reference/s^3/doc_source/guide/KernelandHardwareServices/kernelarch/Concepts/PowerManagementFramework.html#powermgt%2emodel, Last retrieved: 2010-04-23.

Symbian System Model, (2010). In Symbian Developer Community, [Online] Available at: http://developer.symbian.org/wiki/index.php/Symbian_System_Model, Last retrieved: 2010-04-21.

Tranfield, D., Denyer, D. and Smart, P. (2003) *Towards a methodology for developing evidence-informed management knowledge by means of systematic review, British Journal of Management,* Vol. 14, No.3, pp.207-222.

Verkasalo, H. (2009). *Open Mobile Platforms, Modeling the Long-Tail of Application Usage,* In Proc. of Fourth International Conference on Internet and Web Applications and Services: 112-118.

Webster, J. and Watson, R.T. (2002). *Analyzing the Past to Prepare for the Future: Writing a Literature Review,* MIS Quarterly, 26(2): xiii-xxiii.

West, J. (2003). *How open is open enough? Melding proprietary and open source platform strategies,* Research Policy 32: 1259-1285.

What is Android? (2010), In Android Developer's Guide. [Online] Available at: http://developer.android.com/guide/basics/what-is-android.html, Last retrieved: 2010-04-12.

Windows CE Architecture, (2010). [Online] Available at: http://msdn.microsoft.com/en-us/library/ms905093.aspx, Last retrieved: 2010-05-22.

Yamakami, T. (2009). *Foundation-based Mobile Platform Software Engineering: Implications to Convergence to Open Source Software,* ACM International Conference Proceeding Series; Vol. 403, In Proc. of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human: 206-211.

Appendix

A. Questionnaire for Qualitative Interviews

Aim of the interview: Validating the openness strategy of mobile platforms by finding out the platform developers' experiences.

A. **General questions about openness in mobile platforms:**

1. In your point of view, how much the openness degree of a platform depends on its architectural aspects and licensing aspects?
2. What would make you consider a "platform architecture" fully open?
3. How closed must a platform be before you leave it?

B. **Questions about your favorite platform:**

4. Which parts of the architecture of your favorite platform are accessible? (Which parts you can integrate, extend or modify?)
5. For extend or integrate or modify of which parts of your favorite platform do you need permission from the platform supplier?
6. Which parts of the architecture of your favorite platform would you like to see opened up?