A Study of Software Reuse in Cross-Platform Mobile Application Development

Tajammul Shah and Zaheer Ahmad

***Abstract*— The rapid growth of the mobile application market and the increasing popularity of the three major mobile platforms (Android, iOS and Windows Phone) has forced the mobile application developers to move to cross-platform mobile application development. However, this shift has also increased the resources and effort required to develop a mobile application. To minimize this increase, different approaches are adopted by the developers for cross-platform application development. These approaches enable the developers to reduce the required cost and effort by introducing wide extent of software reuse. This paper focuses on the study of software reuse in the different approaches adopted in cross-platform application development and the limitations of those approaches.**

I. Introduction

Over the past few years, the highly increasing popularity of smartphones has attracted the attention of the software community [1]. Now most of the businesses and individual developers consider the mobile applications market as their primary target. However, this market is not relying on a single mobile platform. In fact, there are a number of different mobile platforms available three most popular are Android, iOS and Windows phone. There are currently about 1,600,000 applications on the Google Play Store (Android market), 1,500,000 applications on the Apple App Store and 340,000 applications on Windows Phone Store [2]. Because of this popularity, the mobile applications development community mostly targets all of these major mobile platforms for the success of their applications. The approach to the development of mobile applications that can be used on multiple mobile platforms is called cross-platform mobile application development and has currently become very popular [3]. However, due to the differences in the mobile platforms, the developers need to build a single application multiple times in order to target multiple mobile platforms. This approach results a huge increase in the development and maintenance cost [6], [13], [14] and also increases the software engineering effort needed to develop the application [12].

To address this issue, different approaches are used for cross-platform applications development e-g web-based applications development and hybrid applications development. These approaches provide a singular development environment and the resultant application can be run on multiple platforms, thus introduce a wide extent of software reuse and therefore, greatly reduce the cost and effort required for the development and maintenance [4], [16]. However, apart from these advantages, these approaches also have some limitations that make the use of these approaches limited to specific range of applications. These limitations are discussed in detail, later in this paper.

Our paper focuses on the study of software reuse in the different approaches adopted for cross-platform mobile applications development. Our study is based on the three different approaches; Native Application Development (NAD), Web-based Application Development (WAD) and Hybrid Application Development (HAD). We also discuss the strengths and weaknesses of each of these three approaches.

The authors are with the Department of Computer Software Engineering, Military College of Signals, National University of Sciences & Technology Islamabad, Pakistan.
E-mail: {tajammulshah, zaheerahmad}.mscs22@students.mcs.edu.pk

II. Related Work

Much research work can be found related to cross-platform mobile application development. For example, three different approaches used for cross-platform application development are identified that are Native Application development, Web-based Application Development and Hybrid Application Development [13], [5], [6], [10], [11] and a detailed comparison of these approaches is also presented. For the hybrid applications development, different tools and frameworks are used e-g Phonegap, Xamarine, Appcelerator Titanium and Corona etc. A number of articles can be identified that discuss, compare and evaluate these tools and point out the advantages and disadvantages of each of these tools and frameworks [4], [7], [8], [13], [22]. As in cross-platform application development, different platforms have different requirements for development, for example, different programming languages, software development kits (SDKs) and different integrated development Environments (IDEs), code reuse is a big challenge in cross-platform application development [10], [11]. Although in single-platform application development (e-g Android), software reuse can be achieved to a wide extent [9].

III. Cross-Platform Mobile Application Development

Cross-platform mobile application development is the approach to the development of mobile applications that can be used on multiple mobile platforms. Cross-platform application development is becoming very popular because the mobile application market has split among several mobile platforms with the majority shares among Android, iOS and Windows Phone [2]. Therefore, the developers do not rely on a single mobile platform and rather target multiple mobile platforms for the success of their applications. However, this also increases the resources and effort required in the development, testing and maintenance of the applications [10]. For cross-platform mobile application development, three different approaches are adopted: Native Application Development (NAD), Web-based Application Development (WAD) and Hybrid Application Development (HAD) [10], [13], [15]. Each of these approaches is discussed as follows.

A. Native Application Development

Native application development involves developing the application on a native mobile platform, which can be Android, iOS or Windows Phone. Thus to target multiple platforms, a separate application is developed for each platform. Developing a native application requires specific programming language for example, Java is used for Android, Objective C or Swift is used for iOS and C# is used for Windows Phone applications development. It also requires platform specific IDEs for example, Android Studio is the officially used IDE for Android applications development [19], Xcode is used for iOS [20] and Microsoft Visual Studio is used for developing applications for Windows Phone [21].
The main advantage of native applications is the high performance [17]. Native applications are compiled into machine code, which results in the best possible performance. Therefore, native applications offer the best graphics and animations and are suitable for highly graphical applications such as games [18]. Another advantage is that native applications provide full access to the hardware e-g camera, accelerometer, microphone, GPS and gyroscope sensors of the device. Moreover native applications also provide the richest user experience [6].
Disadvantages of the NAD approach include increased cost and time for development and maintenance [18], [17], [13], [6], [14]. The reason is that for each platform the developer has to learn specific programming language, SDK and IDE as well as maintain a separate code base for each platform. Another disadvantage is the lack of portability in native applications because each application is developed for a specific platform and cannot be used on any other platform [18]. Besides these disadvantages, the NAD approach is the most common approach adopted by the developers [10].

B. Web-based Application Development

Web-based applications or Mobile-web applications are browser-based applications that are developed using the HTML and JavaScript technologies [6]. These applications are downloaded from the web and require a web browser to run. No application related components are installed on the device and the application's data is server driven.
The main advantage with web-based applications is that they can be accessed from all devices via the platform's browser [5] and therefore, are not confined to a single platform only. This greatly reduces the cost and effort for development and maintenance as only a single application can be used on all platforms. Another advantage is that these applications do not require app store submission [15], so the developer does not have to wait for long approval process (mainly an issue with Apple app store submission). Also in web-based applications, the data and the application is hosted on a server so maintenance can be done remotely on the server.
The disadvantages of the web-based applications include limited or no access to the APIs that can access device's data, storage, camera and other sensors [6], [15]. However, the recent adoption of HTML5 standard provides access to the device's hardware and software components through some APIs. Another disadvantage is the extra rendering time that can affect the user experience. Also there are some situations where the mobile-web applications are inaccessible for the end user e-g when the device is in Airplane mode [6].

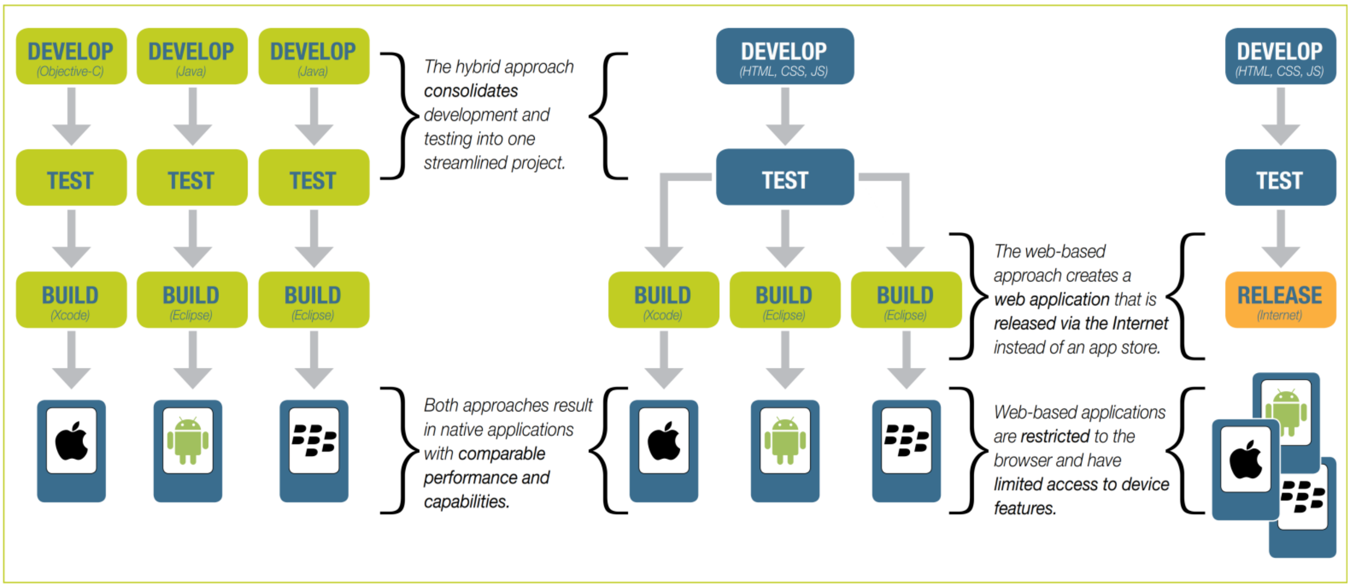

DEVELOP (Objective-C)
DEVELOP (Java)
DEVELOP (Java)
TEST
TEST
TEST
BUILD (Xcode)
BUILD (Eclipse)
BUILD (Eclipse)
The hybrid approach consolidates development and testing into one streamlined project.
Both approaches result in native applications with comparable performance and capabilities.
DEVELOP (HTML, CSS, JS)
TEST
BUILD (Xcode)
BUILD (Eclipse)
BUILD (Eclipse)
The web-based approach creates a web application that is released via the Internet instead of an app store.
Web-based applications are restricted to the browser and have limited access to device features.
DEVELOP (HTML, CSS, JS)
TEST
RELEASE (Internet)

Fig. 1: Hybrid applications are closing the gap between native and web-based application development.

C. *Hybrid Application Development*

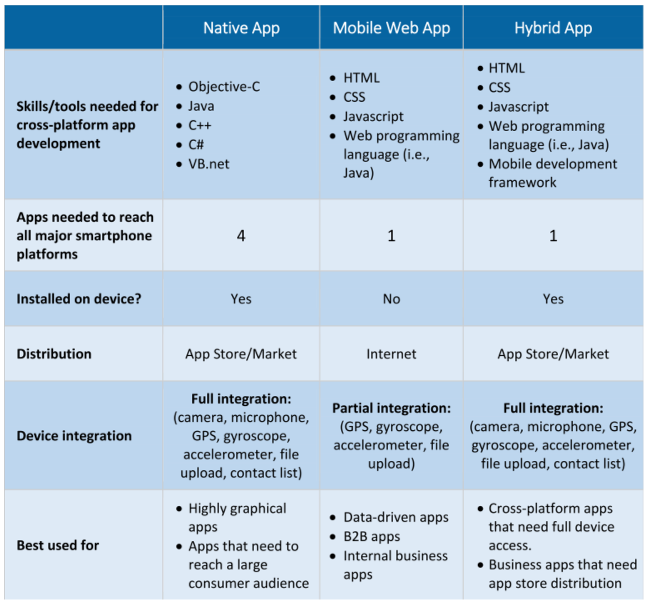
The hybrid application development approach is a cross between the NAD and the WAD approaches [18]. In this approach, HTML5 and JavaScript are used as development technologies and no detailed knowledge of the target platform is required. A hybrid application is generally a mobile-web application embedded inside a native container (like UIWebview in iOS and Webview in Android) [6]. Like native applications, the hybrid applications must be installed on the device and therefore are also subjected to app store submission [18]. Different tools and frameworks are now available that support hybrid applications development e-g PhoneGap, Appcelerator Titanium, Sencha, Xamarin and Corona etc. A detailed analysis of these tools is available in a number of articles e-g [8], [13], [22], [5]. Fig. 1 shows how hybrid applications close the gap between native and web-based applications. In the NAD approach, three different applications are developed from scratch to support the three different mobile platforms. Whereas in the WAD approach, a single application developed is run on all the three platforms. In case of HAD, a single application is developed and tested, like the WAD approach, but built separately for each platform like the NAD approach.

Hybrid applications combine the advantages of both native and web-based applications. Like native applications they provide full access to the device's hardware and data [6], [15], [18]. Like web-based applications, a single application is developed using web technologies and can be deployed on multiple platforms by just wrapping it inside a platform specific shell. This reduces the cost and effort.

The disadvantages of hybrid applications include greater processing time as compared to native applications and also the lack of native look and feel. Moreover these applications can also suffer from the platform specific behavior of JavaScript and threading model incompatibilities with JavaScript. Fig. 2 shows a side by side comparison of native, web-based and hybrid applications.

IV. Software Reuse in Cross-Platform Mobile Application Development

Software reuse is the process of using existing software components to create new software systems and it facilitates the increase of productivity, quality, reliability and the decrease of cost and implementation time. While code is the most common reusable component, other components and assets produced during the software development process can also be reused e-g software designs, specifications, prototypes, tests cases, data, frameworks, and documentation etc.
In mobile applications development, software reuse can be achieved to a wide extent in the case of single-platform application development. In an empirical study conducted for analyzing software reuse in Android applications, three different types of software reuse were identified as, inheritance reuse, code reuse and framework reuse of whole apps



	Native App	Mobile Web App	Hybrid App
Skills/tools needed for cross-platform app development	• Objective-C • Java • C++ • C# • VB.net	• HTML • CSS • Javascript • Web programming language (i.e., Java)	• HTML • CSS • Javascript • Web programming language (i.e., Java) • Mobile development framework
Apps needed to reach all major smartphone platforms	4	1	1
Installed on device?	Yes	No	Yes
Distribution	App Store/Market	Internet	App Store/Market
Device integration	**Full integration:** (camera, microphone, GPS, gyroscope, accelerometer, file upload, contact list)	**Partial integration:** (GPS, gyroscope, accelerometer, file upload)	**Full integration:** (camera, microphone, GPS, gyroscope, accelerometer, file upload, contact list)
Best used for	• Highly graphical apps • Apps that need to reach a large consumer audience	• Data-driven apps • B2B apps • Internal business apps	• Cross-platform apps that need full device access. • Business apps that need app store distribution

Fig. 2: Side by side comparison of native, web-based and hybrid applications [18]

[9]. The study was carried out on hundreds of thousands of Android applications across thirty different categories. The results show that about 18.75% of Android application classes inherit from a base class in the Android API and 35.78% of the classes inherit from a domain-specic base class; 84.23% of classes across all categories of applications occur in two or more applications and 17,109 applications were a direct copy of another application. This means that in the case of single platform application development, the application developers perform significant software reuse, despite the lack of formal software engineering training.
However, the problem arises in the case of cross-platform application development, where the application is built for different platforms using different programming languages, SDKs and IDEs. This difference in the development requirements for different mobile platforms makes the software reuse a big challenge in cross-platform applications development. This section focuses on the study of software reuse in cross-platform application development. As three different approaches are adopted for cross-platform applications development (i-e NAD, WAD and HAD), we must first discuss how much software reuse is possible in each of these approaches and then their limitations and alternatives.

A. *Software Reuse in NAD*

In native applications development approach, a single application is developed multiple times to support multiple mobile platforms. According to a survey, most of the developers see the multiple platform application development as a big challenge in mobile applications development [10]. The survey says that most of the developers write the same application from scratch for each platform and often find it impossible or challenging to port functionality across different mobile platforms. The survey shows that it is almost

impossible to achieve code reuse in cross-platform application development because each mobile platform has different requirements for development e-g programming language, development environment etc. Even if the programming languages may be the same (e-g Android and Blackberry both use Java for application development), different mobile platforms work in different ways and the code is often written from scratch. If any portion of the code is reused across different platforms, it does not generate quality results.
Besides the code reuse, the user interface (UI) design is also a big challenge in cross-platform application development. This is due to the difference in the HCI guidelines across different mobile platforms. Thus the developers have to carry out a screen by screen design for each platform separately to preserve the behavioral consistency across multiple platforms. Similar is the case with the testing phase also, where separate testing is performed for each platform to test the flow and the behavior of the application on each platform. Although the test cases may be the same for each platform but they must be implemented uniquely on each platform. In the maintenance phase also, a lot of cost and effort is required to maintain a separate application for each platform. So in the NAD approach, there is no or very little software reuse, which greatly increases the time, effort and budget required for the development and maintenance of an application.

B. Software Reuse in WAD

In web-based applications, a single application is developed in web technologies like HTML and JavaScript and used across multiple mobile platforms. This approach may be considered as a solution to the reuse problem in the native applications. The web-based applications can provide complete software reuse across different mobile platforms as they are platform independent and once developed, they can be used on multiple platforms via the platform browser. This approach is based on write once and run everywhere. In the WAD approach, only a single application is designed, developed, tested and released, that can, then, be used across multiple platforms without any changes required in the design and implementation. This also results in an easy maintenance of the application.
Though in terms of software reuse, the WAD approach is far more better than the NAD approach but still it carries some technical disadvantages (as discussed in Section III.B.) and is not a choice of most of the developers [10].

C. Software Reuse in HAD

The hybrid application development approach is considered as a bridge to fill the gap between the NAD and the WAD approaches. This approach provides full software reuse (which is a problem in native applications) as well as tries to overcome the disadvantages of web-based applications (e-g lack of hardware access and lower performance).
As a hybrid application is generally a web-based application wrapped in a native shell, it combines the advantages of both native and web-based applications. Like web-based applications, a single application is developed and its functionality is tested. After successful testing the application is then wrapped in a separate platform-specific native shell and built for each platform. Thus all the assets of an application are reused across all the platforms. Also wrapping the application in a native shell gives it somewhat native look and feel and better performance as compared to web-based applications. Therefore, the HAD approach can be considered as the best solution both in terms of software reuse and technical capabilities (such as full hardware access and better performance). This approach is also cost and time efficient as compared to the NAD approach and also easy to maintain. This is the reason that the popularity of the HAD approach is rapidly increasing among the developers as compared to native and web-based approaches [23].

D. Limitations and Alternatives

The above discussed approaches (i-e WAD and HAD) have some limitations (such as performance, user experience and hardware access) and are technological solutions to the software reuse problem in the NAD, rather than the desired software engineering approach to reuse early software engineering assets [12]. Therefore the researchers are trying to propose a software engineering solution that can achieve both the advantages of native applications as well as software reuse across multiple platforms in cross-platform applications development. One such proposed solution is the Model-Driven Mobile Development [24], [25], [26].
Model-driven development fits as a nice solution to the problem of cross-platform mobile applications development. In model driven development, the development of substantial parts of a program is based on models. Source code is generated from automatic transformations of developer-defined models. There can also be several transformations that generate code for different target platforms. Application developers describe their application on a relatively high level and this description is translated into code for different mobile platforms. In model-driven application development there is not only a single, cross-platform code base (i-e the model) but also an increased abstraction level and fast development cycles. Generated applications are truly native and do not suffer from any of the issues that the Web-based approaches do with respect to native look & feel.

V. Conclusion

With the increasing demands of the mobile application market and the distribution of the market shares among different platforms, developers are switching towards cross-platform mobile applications development. As the majority of market shares belong to three different mobile platforms (Android, iOS and Windows Phone), developers mainly target these three platforms for the success of their mobile applications. However due to the differences in the programming languages, SDKs and IDEs, cross-platform application development is a big challenge in the mobile applications development.
The aim of this paper was to study the different approaches

used in the cross-platform application development and to discuss software reuse in each of these approaches along with their limitations. Our study concluded with the result that though native application development approach is the best in terms of end user experience and technical capabilities (such as full hardware access and best performance), it is not a good choice in terms of software reuse. The NAD approach can achieve software reuse to a small extent, almost no software reuse. The alternative solution to the NAD approach is the web-based application development approach. Although this approach provides full software reuse across multiple mobile platforms, it is not considered as a good solution because of the technical weaknesses such as lack of hardware access and lower performance. The currently best approach for cross-platform application development may be the HAD approach which combines the advantages of both native and web applications. Like web-based applications, this approach enables full software reuse across different mobile platforms and provides better performance and full hardware access like native applications. However, these solutions are the technological solutions rather than a software engineering solution. One software engineering solution for the problem of cross-platform application development is the Model-Driven Mobile Development. This approach bases the development on models and these models are then translated into source code. The model-driven approach enables software reuse as there is a single code base (i-e the model) that can be reused across multiple mobile platforms. Moreover the applications developed using this approach are truly native and are far superior to the web-based approaches.

References

[1] D. Bosomworth, "Mobile marketing statistics 2015", Smart Insights, 2015. [Online]. Available: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics. [Accessed: 08- Nov- 2015].

[2] Statista, "Number of apps available in leading app stores 2015 — Statistic", 2015. [Online]. Available: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores. [Accessed: 08- Nov- 2015].

[3] K. Shah, "The Rise of the Cross Platform App Development Market", Enterprisecioforum.com, 2015. [Online]. Available: http://www.enterprisecioforum.com/rise-cross-platform-app-development-mark. [Accessed: 08- Nov- 2015].

[4] Hartmann, Gustavo, Geoff Stead, and Asi DeGani. "Cross-platform mobile development." Mobile Learning Environment, Cambridge (2011).

[5] Heitktter, Henning, Sebastian Hanschke, and Tim A. Majchrzak. "Evaluating cross-platform development approaches for mobile applications." In Web information systems and technologies, pp. 120-138. Springer Berlin Heidelberg, 2013.

[6] Xanthopoulos, Spyros, and Stelios Xinogalos. "A comparative analysis of cross-platform development approaches for mobile applications." InProceedings of the 6th Balkan Conference in Informatics, pp. 213-220. ACM, 2013.

[7] Corral, Luis, Andrea Janes, and Tadas Remencius. "Potential advantages and disadvantages of multiplatform development frameworksa vision on mobile environments." Procedia Computer Science 10 (2012): 1202-1207.

[8] Dalmasso, Isabelle, Soumya Kanti Datta, Christian Bonnet, and Navid Nikaein. "Survey, comparison and evaluation of cross platform mobile application development tools." In Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International, pp. 323-328. IEEE, 2013.

[9] Mojica, Israel J., Bram Adams, Meiyappan Nagappan, Steffen Dienst, Theodore Berger, and Ahmed E. Hassan. "A large-scale empirical study on software reuse in mobile apps." Software, IEEE 31, no. 2 (2014): 78-86.

[10] Joorabchi, Mona Erfani, Ali Mesbah, and Philippe Kruchten. "Real challenges in mobile app development." In Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on, pp. 15-24. IEEE, 2013.

[11] Wasserman, Anthony I. "Software engineering issues for mobile application development." In Proceedings of the FSE/SDP workshop on Future of software engineering research, pp. 397-400. ACM, 2010.

[12] Dehlinger, Josh, and Jeremy Dixon. "Mobile application software engineering: Challenges and research directions." In Workshop on Mobile Software Engineering, vol. 2, pp. 2-2. 2011.

[13] Adam M. Christ, "Bridging the Mobile App Gap," Inside the Digital Ecosystem, 11, no. 1 (2011): 28.

[14] Kony, "RES-Glossary-Cross-Platform-Mobile-Development", 2013. [Online]. Available: http://www.kony.com/resources/glossary/cross-platform-mobile-development. [Accessed: 01- Dec- 2015].

[15] Amatya, Suyesh. "Cross-Platform Mobile Development: An Alternative to Native Mobile Development." (2013).

[16] S. Jones, "Cross-Platform Developer Tools 2012 - VisionMobile", VisionMobile, 2012. [Online]. Available: http://www.visionmobile.com/blog/2012/02/crossplatformtools/. [Accessed: 30- Nov- 2015].

[17] A. Ziflaj, "Native vs Hybrid App Development", SitePoint, 2014. [Online]. Available: http://www.sitepoint.com/native-vs-hybrid-app-development/. [Accessed: 30- Nov- 2015].

[18] Michaels, ross & cole, ltd, Native mobile apps: The wrong choice for business? A white paper from michaels, ross & cole, ltd. (mrc).

[19] A. Overview, "Android Studio Overview — Android Developers", Developer.android.com, 2015. [Online]. Available: http://developer.android.com/tools/studio/index.html. [Accessed: 15- Dec- 2015].

[20] Developer.apple.com, "Xcode - IDE - Apple Developer", 2015. [Online]. Available: https://developer.apple.com/xcode/ide/. [Accessed: 15- Dec- 2015].

[21] Msdn.microsoft.com, "Getting started developing apps for Windows Phone 8 and Windows 8", 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/apps/jj714071(v=vs.105).aspx. [Accessed: 15- Dec- 2015].

[22] Ribeiro, Alejandro, and Airton R. da Silva. "Survey on Cross-Platforms and Languages for Mobile Apps." In Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the, pp. 255-260. IEEE, 2012.

[23] D. Ramel, "Mobile Report: Native Development Trails Web, Hybrid – ADTmag", Adtmag.com, 2015. [Online]. Available: https://adtmag.com/articles/2015/04/27/mobility-survey.aspx. [Accessed: 01- Dec- 2015].

[24] Vaupel, Steffen, Gabriele Taentzer, Jan Peer Harries, Raphael Stroh, Ren Gerlach, and Michael Guckert. "Model-driven development of mobile applications allowing role-driven variants." In Model-Driven Engineering Languages and Systems, pp. 1-17. Springer International Publishing, 2014.

[25] Balagtas-Fernandez, Florence T., and Heinrich Hussmann. "Model-driven development of mobile applications." In Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on, pp. 509-512. IEEE, 2008.

[26] Heitktter, Henning, Tim A. Majchrzak, and Herbert Kuchen. "Cross-platform model-driven development of mobile applications with md 2." In Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 526-533. ACM, 2013.