Model-Driven Cross-Platform Application Framework: Addressing Common Errors

Uyen Bui

California State University, Long Beach

**Feature at a Glance**

There are approximately 5 billion unique mobile users in the world as of 2019, and of those users, 3 billion are specifically smartphone users. With the majority of smartphone users’ time being spent on mobile apps, app developers must work harder than ever to ensure the best user experience. However, this can be difficult due to the duopoly of iOS and Android in the smartphone operating system (OS) market, where users are split across two different platforms.  Therefore, developers have turned towards the cross-platform development approach, where apps can be developed for the two different platforms using one source code. In this paper, I present design recommendations for a cross-platform framework to ensure development goes as smoothly as possible, as well as future areas to explore with regards to cross-platform development.

**Keywords**: mobile apps, user interface, platform-independent model, platform-specific model, energy consumption

As of January 2019, there are approximately 5.112 billion unique mobile users in the world (Kemp, 2019). Out of those 5 billion users, 3.2 billion are specifically smartphone users (Gu, 2019). When using smartphones, users spend 90% of the time on mobile applications (apps) (Wurmser, 2019). By extension, 3.2 billion smartphone users are also smartphone app users, indicating that great care must go into the development of those applications. Smartphone app development would be simpler if everyone used the same device and operating system, but this is not the case. In the United States, there is a near even split of iOS and Android users, with iOS owning 55.65% of the market and Android owning 44.15% ("Mobile operating system market share United States of America," n.d.). With these two platforms owning the majority of the mobile share market, deciding which platform to develop for is a dilemma many app developers may face.

**Types of Mobile Applications**

There are different types of mobile apps, which can be summarized as native, cross-platform, and web (Latif, Lakhrissi, Nfaoui, & Es-Sbai, 2016). Some may categorize web apps as a type of cross-platform app, but I have them as separate because they are accessed via a web browser on a smartphone rather than downloaded and installed from an app store (Biørn-Hansen, Grønli, Ghinea, & Alouneh, 2019). Because there is limited access to device features (Xanthopoulos & Xinogalos, 2013) the user experience suffers. Native apps, on the other hand, are specifically designed specifically for one platform (Heitkötter, Hanschke, & Majchrzak, 2013). As a result, the user interface looks and acts in a manner that is consistent with the platform and the application has access to all device features (e.g. camera, GPS) (Heitkötter et al., 2013). However, the downside of native apps is that they are both time-consuming and costly to develop, since the code cannot be reused for other platforms (Heitkötter et al., 2013). This means if a developer wanted to develop native apps for two platforms, two separate codes need to be written.

With there being large user bases for both iOS and Android, I believe greater focus should be placed on cross-platform development. Code is written once and can be reused for several platforms. There are several approaches to cross-platform app development, which are hybrid, interpreted, cross-compiled, and model-driven (Biørn-Hansen, Grønli, & Ghinea, 2018). Hybrid apps are developed by using a WebView component to display content on an embedded web browser within a native app (Biørn-Hansen et al., 2018). Although they have the appearance of a native app, access to device and platform features are limited unless the specific plugins are provided by the framework. The interpreted approach has code written in one language (e.g. Javascript) which is then interpreted to the native code on different platforms via a Javascript interpreter (Biørn-Hansen et al., 2018). The cross-compiled approach writes a code in a language understood by the desired platforms (e.g. C#). The language is then compiled into the native byte code, which can then be executed on the native platform (Biørn-Hansen et al., 2018). For the model-driven approach, a model is built based on logic and user interfaces using a domain-specific language (DSL) provided by the framework. The framework then generates the native source code without using interpreters or WebView (Biørn-Hansen et al., 2018).

**Common Issues of Cross-Platform Applications**

Cross-platform apps seem ideal due to needing to only write one base code, but they are not without their issues. According to a survey done in 2019, the issues commonly reported regarding cross-platform application development include a decline in performance in comparison to native applications, less than optimal user experience and user interface, lack of a developed user community and design framework, difficulty with both integrating the application into the device application programming interface (API) and debugging and/or testing the application, and security issues (Biørn-Hansen et al., 2019). Regarding the decreased performance, simply using a cross-platform framework to develop an app, even if the app is native, causes increased energy consumption (Ciman & Gaggi, 2017). The task that specifically requires the most energy consumption is the updating of the user interface. Other factors that affect energy consumption include how often the data from sensors (GPS, accelerometer, light sensor) is updated and whether unnecessary code is present in the app (Ciman & Gaggi, 2017).

**Design Recommendations for a Cross-Platform Framework**

Creating a solid framework that addresses the existing common issues in cross-platform development allows developers to be aware of what to focus on in particular when creating their base code. If the framework is inappropriate or inadequate for app development, the resulting app will be of low quality and will require many fixes before it provides satisfactory user experience. A “definitive” framework proposed by Rieger & Majchrzak (2019) identifies certain criteria in order to assist developers in evaluating cross-platform frameworks and consequently choosing which is most appropriate for their task. These criteria are referred to as infrastructure, development, app, and usage (Rieger & Majchrzak, 2019). From these criteria, I will extract what I consider to be the most important or relevant features to include in a framework and, if applicable, provide design recommendations. Prior to evaluating the actual app development framework, however, I recommend using the model-driven development approach in order to first develop a solid framework specifically for the app user interface.

From a human factors perspective, I believe the model-driven development approach is a solid foundation for a framework because it takes into account the user first, as models of the user interface (UI) are created to guide the building of the base code. The UI is already known to be of great importance when it comes to smartphone apps in general (Rieger & Majchrzak, 2019), and decreased quality of the UI is a commonly reported concern regarding cross-platform development (Biørn-Hansen et al., 2019). When using the model-driven development approach, great care must be taken in developing a solid platform-independent model (Usman, Iqbal, & Khan, 2017) of the UI. In conceiving the platform-independent model (PIM) UI, developers can refer back to design principles set by Nielsen (1994) and Shneiderman (2004). However, it is important that the developer keeps in mind that the UI is to be initially designed as platform-independent, meaning that the specific design and interaction guidelines of each platform are not to be implemented yet. Additionally, in regards to the development of the platform-specific models (PSM), some design principles may not need to be entirely followed (e.g. consistency) in order to incorporate the platform-specific design and interaction guidelines. When those guidelines are incorporated, the app is ultimately able achieve a native look and feel (Angulo & Ferre, 2014), along with having a great user interface due to adhering to classic design principles.

Heading into the infrastructure criteria of the “definitive” framework, a developer should see if the platforms they wish to develop an app for are supported by the framework (Rieger & Majchrzak, 2019). At a bare minimum, I recommend that the framework includes support for iOS and Android on smartphone devices but only for a certain version of the operating system and higher, as the author mentions that there may be large differences even within the same operating system (Rieger & Majchrzak, 2019). By defining a specific range of versions for the operating system to be supported by the framework, developers can further narrow down what to include and what not to include in developing their app, ensuring it is more optimized. Additionally, there should be support for the main app store of major platforms iOS and Android in order to allow the developer to distribute their app (Rieger & Majchrzak, 2019). For the development criteria, the framework should support the developer and allow for ease and speed of development. This includes providing tutorials, built-in auto completion and debuggers when coding, and tools for app testing (Rieger & Majchrzak, 2019).

It is stated in the “definitive” framework that the speed at which the app is developed can depend on the number and type of programming languages supported by the framework (Rieger & Majchrzak, 2019). Often times in a model-driven development approach, developers must learn the single domain specific language (DSL) of the framework rather than a common programming language (Biørn-Hansen et al., 2018). Additionally, the DSL varies from framework to framework (Biørn-Hansen et al., 2018). Rather than only including one commonly used programming language or one DSL, a recommendation to facilitate quicker development would either be for the framework to support a variety of commonly used programming languages that will be used after a model is conceived (C#, Javascript) or to work on developing a universal modeling language (UML) for app development, which is already present in software development (Osis & Donins, 2017).

For app criteria, the framework should allow for access to all device features and platform features, take into account input and output heterogeneity for the device, and provide support for security features (Rieger & Majchrzak, 2019). Because the model-driven approach is used, platform features should be able to be accessed directly, as opposed to relying on plugins (Biørn-Hansen et al., 2018). Regarding security features, it is recommended that there is basic or advanced support so inexperienced developers do not have to stress about manually implementing them (Rieger & Majchrzak, 2019). Attributes of security to consider providing support for are confidentiality, integrity, and control (Parker, 1988). For example, regarding confidentiality, the app should be designed to only ask for permission to access a device feature if it is needed. Essentially, a solid security framework should be developed in order to implement it within a cross-platform development framework, as app security remains a major concern (Biørn-Hansen et al., 2019).

The performance criteria includes energy consumption, which is covered in Ciman and Gaggi’s energy consumption study (2017). Based on their study, developers should already be aware that energy consumption may increase due to the usage of a cross-platform framework (Ciman & Gaggi, 2017), so the focus should be placed on optimizing the code. Knowing that the tasks that have the highest impact on energy consumption are updating the user interface (UI) and how often data from sensors (e.g. GPS) are updated (Ciman & Gaggi, 2017), as well as knowing unnecessary code negatively impacting the battery life specific design (Thiagarajan, Aggarwal, Nicoara, Boneh, & Singh, 2012) recommendations can be made. Within the framework, there can be suggestions to the developer during coding in order to eliminate code deemed unnecessary, and perhaps options to limit the frequency at which sensors and the UI are updated rather than letting them update as they want. Additionally, the developer should take care to code the UI in the most efficient way instead of only focusing on achieving a certain appearance. Other recommendations include simulations within the framework to estimate the amount of energy consumed based on a certain battery capacity of a device, or the company behind the framework can provide to the developer an external tool similar to Monsoon PowerMonitor to directly measure the amount of energy required from the battery for specific tasks (Ciman & Gaggi, 2017). However, it is important to find a balance when working on optimizing performance since other areas may suffer if performance is highly prioritized (e.g. user interface) and performance itself will deteriorate if neglected (e.g. app becomes bloated due to many other features being implemented) (Rieger & Majchrzak, 2019).

**Applications in Summary**

Using a model-driven development approach allows developers to essentially create a framework focusing first and foremost on the user interface (UI), which is considered to be an important aspect of smartphone apps. By developing a solid framework for the platform-independent model (Usman et al., 2017) (PIM) UI using classic design principles from Nielsen (1994) and Shneiderman (2004), the consequent platform-specific model (PSM) UI is both consistent with the design and interaction guidelines of the target platform and generally has a good UI. In other words, the UI looks and feels like a native UI while also following the golden design principles.

 In the actual framework, defining both specific platforms and versions of those platforms allows developers to focus on delivering a more optimized app due to having a clearer picture of what can and cannot be included based on operating system capabilities. This will ensure a more consistent performance of the app as well. Additionally, supporting one or more commonly supported programming or defining a universal modeling language (UML) for the model-driven approach can decrease the development time due to the likelihood of the developer already being familiar with the language and therefore not having a large learning curve when beginning to program the app. At a minimum, the framework should also provide support to the developer in the form of tutorials, auto-completion of code, debugging tools, and so on. All of these serve to speed up and increase the ease of the development progress, as well as improve the quality of the app.

 Having the framework include security addresses the ongoing issue of app security as well as partially relieving the developer of having to manually implement their own security features. The result is a secure app that respects confidentiality, for example, and does not receive access to a device feature unless given permission by the user. This also ensures that the app is accessing appropriate features as it is relevant to the app domain. Regarding energy consumption, incorporating a way to eliminate code deemed unnecessary as well as being able to control the refresh rate of sensors and the UI will reduce the energy consumption and by extension the performance of the app. Being able to externally examine the power consumption of certain tasks can help to determine what could be the cause of battery drain, which developers can then go back and fix.

**Limitations and Future Areas of Consideration**

 Biørn-Hansen et al., (2018) mentions the lack of accessibility in the literature of cross-platform development. For future frameworks, modeling specific accessibility features is something that can be included when conceiving the platform-independent model of the UI. This way, accessibility features can be incorporated into the app for users who are in need of them (Krainz, Feiner, & Fruhmann, 2016). Additionally, security is only briefly mentioned this proposed platform, though it is an ongoing issue as it is not commonly mentioned in the context of cross-platform applications (Biørn-Hansen et al., 2019). The development of a universal modeling language (UML) is also something to be further researched, as it already established within the context of software development (Osis & Donins, 2017) but nothing appears to be solidified for cross-platform development. It would be useful in order to create a model-driven approach that uses a common domain specific language (DSL).

The role of the device is also not mentioned regarding the performance of apps and the perceived performance of apps. Although the performance of a cross-platform app is inherently worse than a native app, the performance is still good overall if on a higher-end device (Willocx, Vossaert, & Naessens, 2015). However, the device and its attributes can potentially negatively influence the perception of the app performance (Noei, Syer, Zou, Hassan, & Keivanloo, 2017). This can be something to take into consideration for future iterations of cross-platform frameworks.

Because the proposed recommendations are for a framework specifically for iOS and Android smartphones, future areas to consider are other current mobile platforms. This includes the development of a cross-platform applications for wearables (e.g. Apple Watch), tablets, and Smart TVs. Due to the varying different user interfaces, it may be beneficial to keep the model-driven approach and once again build a platform-independent model that can apply to all of the desired platforms. In fact, there is already a study examining how to simultaneously develop for the plethora of mobile devices (Rieger & Kuchen, 2019) by using the model-driven approach to develop a platform-independent model. This indicates that in order to tackle developing simultaneously for several devices, it may be most beneficial to further research and improve the model-driven approach. Additionally, the common concerns with cross-platform development with regards to smartphone applications may also carry over to other mobile devices, particularly issues with security and energy consumption. Other unique issues, such as the synchronization of apps from wearable devices to a smartphone (Rieger & Kuchen, 2019), may be identified and accounted for in an updated cross-platform framework.

References

Angulo, E., & Ferre, X. (2014). A case study on cross-platform development frameworks for

mobile applications and UX. *Proceedings of the XV International Conference on Human*

*Computer Interaction - Interacción 14*. doi: 10.1145/2662253.2662280

Biørn-Hansen, A., Grønli, T.-M., & Ghinea, G. (2018). A survey and taxonomy of core

concepts and research challenges in cross-platform mobile development. *ACM*

*Computing Surveys*, *51*(5), 1–34. doi: 10.1145/3241739

Biørn-Hansen, A., Grønli, T.-M., Ghinea, G., & Alouneh, S. (2019). An empirical study of

cross-platform mobile development in industry. *Wireless Communications and Mobile*

*Computing*, *2019*, 1–12. doi: 10.1155/2019/5743892

Ciman, M., & Gaggi, O. (2017). An empirical analysis of energy consumption of cross-platform

frameworks for mobile development. *Pervasive and Mobile Computing*, *39*, 214–230.

doi: 10.1016/j.pmcj.2016.10.004

Heitkötter, H., Hanschke, S., & Majchrzak, T. A. (2013). Evaluating Cross-Platform

Development Approaches for Mobile Applications. *Lecture Notes in Business Information Processing Web Information Systems and Technologies*, 120–138. doi: 10.1007/978-3-642-36608-6\_8

Gu, T. (2019, September 17). Newzoo's global mobile market report: Insights into the world's

3.2 billion smartphone users, the devices they use & the mobile games they play.

Retrieved October 30, 2019, from <https://newzoo.com/insights/articles/newzoos-global->

mobile-market-report-insights-into-the-worlds-3-2-billion-smartphone-users-the-devices-

they-use-the-mobile-games-they-play/.

Kemp, S. (2019, January 30). Digital 2019: global digital overview - datareportal – global

digital insights. Retrieved from <https://datareportal.com/reports/digital-2019-global->

digital-overview.

Krainz, E., Feiner, J., & Fruhmann, M. (2016). Accelerated development for accessible apps –

model driven development of transportation apps for visually impaired

people. *Lecture Notes in Computer Science Human-Centered and Error-Resilient*

*Systems Development*, 374–381. doi: 10.1007/978-3-319-44902-9\_25

Latif, M., Lakhrissi, Y., Nfaoui, E. H., & Es-Sbai, N. (2016). Cross platform approach for mobile

application development: A survey. *2016 International Conference on Information*

*Technology for Organizations Development (IT4OD)*. doi: 10.1109/it4od.2016.7479278

Mobile operating system market share United States of America. (n.d.). Retrieved October 30,

2019, from <https://gs.statcounter.com/os-market-share/mobile/united-states-of-america>.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. *Proceedings of the*

*SIGCHI Conference on Human Factors in Computing Systems Celebrating*

*Interdependence - CHI 94*. doi: 10.1145/191666.191729

Noei, E., Syer, M. D., Zou, Y., Hassan, A. E., & Keivanloo, I. (2017). A study of the relation of

mobile device attributes with the user-perceived quality of Android apps. *Empirical*

*Software Engineering*, *22*(6), 3088–3116. doi: 10.1007/s10664-017-9507-3

Osis, J., & Donins, U. (2017). *Topological UML modeling: An improved approach for domain*

*modeling and software development*. Amsterdam, Netherlands: Elsevier.

Parker, D. B. (1998). *Fighting computer crime: A new framework for protecting information*.

New York: John Wiley & Sons, Inc.

Rieger, C., & Kuchen, H. (2019). A model-driven cross-platform app development process for

heterogeneous device classes. *Proceedings of the 52nd Hawaii International*

*Conference on System Sciences*. doi: 10.24251/hicss.2019.894

Rieger, C., & Majchrzak, T. A. (2019). Towards the definitive evaluation framework for cross-

platform app development approaches. *Journal of Systems and Software*, *153*, 175–199.

doi: 10.1016/j.jss.2019.04.001

Shneiderman, B., & Plaisant, C. (2004). *Designing the user interface: Strategies for effective*

*human-computer interaction*. Boston: Pearson/Addison Wesley.

Thiagarajan, N., Aggarwal, G., Nicoara, A., Boneh, D., & Singh, J. P. (2012). Who killed my

battery?: Analyzing mobile browser energy consumption. *Proceedings of the 21st*

*International Conference on World Wide Web - WWW 12*. doi:

10.1145/2187836.2187843

Usman, M., Iqbal, M. Z., & Khan, M. U. (2017). A product-line model-driven engineering

approach for generating feature-based mobile applications. *Journal of Systems and*

*Software*, *123*, 1–32. doi: 10.1016/j.jss.2016.09.049

Willocx, M., Vossaert, J., & Naessens, V. (2015). A quantitative assessment of performance in

mobile app development tools. *2015 IEEE International Conference on Mobile*

*Services*. doi: 10.1109/mobserv.2015.68

Wurmser, Y. (2019, May 30). Time spent with mobile 2019: US mobile. Retrieved October 30,

2019, from <https://www.emarketer.com/content/us-time-spent-with-mobile-2019>.

Xanthopoulos, S., & Xinogalos, S. (2013). A comparative analysis of cross-platform

development approaches for mobile applications. *Proceedings of the 6th Balkan*

*Conference in Informatics on - BCI 13*. doi: 10.1145/2490257.2490292

**Uyen Bui** is a student in the Master of Science in Human Factors program at California State University, Long Beach (CSULB). She received her Bachelor of Arts in Psychology at CSULB. Her research interests currently lie in the areas of human-computer interaction, usability, user experience and user interface. Uyen can be reached at uyen.bui@student.csulb.edu.