Tailoring Software Architecture Concepts and Process for Mobile Application Development

Felix Javier Acero Salazar, Marco Brambilla
Politecnico di Milano. Dipartimento di Elettronica, Informazione e Bioingegneria.
Piazza Leonardo da Vinci, 32. 20133 Milan, Italy
felixjavier.acero@mail.polimi.it, marco.brambilla@polimi.it

ABSTRACT

Enabled by the continuous improvement of the hardware and software in mobile devices, mobile applications have evolved into very complex pieces of software. Yet, such increase in complexity has not been paired by an increased awareness, among developers, of the important role that some software engineering processes play in managing such complexity.

In this paper we focus on the architectural design of mobile applications: we show how this aspect is still overlooked by mobile app developers; we present a high level process and several concepts that aim to guide developers in the creation of suitable architectures for their apps; and we describe the advantages of integrating architectural thinking within the mobile app development process.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; D.2.29 [**Software Engineering**]: Management

Keywords

Software architecture, mobile applications, architectural design, mobile software architecture.

1. INTRODUCTION

Mobile applications today are not any more simple content fruition interfaces that present data on the screen and capture user interactions. Instead, they also manage complicated business logic, enable mechanisms for accessing and synchronizing data stored in local and remote data sources, gather contextual information, enable communication paths to access external service providers and orchestrate the operation of a growing set of wearable devices and sensors to which they offer processing power and communication capabilities with the cloud and other networked systems.

Two important consequences of such increase in complexity are: *1)* mobile applications now need to be developed by larger developer teams; *2)* the processes used to develop and maintain these applications have become more relevant.

Given the rather narrow advice offered by platform developers like Apple and Google in this regard, a particular process in which we want to concentrate in this paper is the architectural design; for it is through it that mobile applications can achieve important quality attributes like modifiability, testability and reusability.

Following this line of thought, this paper aims to reinforce architectural thinking among mobile developers by mapping some architectural concepts on to the mobile world, and presenting a high level process that developers can follow to design suitable architectures for their apps.

2. ARCHITECTURE: THE MISSING CONCERN OF MOBILE DEVELOPMENT

Over the past decades, software architecture has consolidated as an important sub-field of software engineering [7]. Its importance and value has resonated across practitioners to the point where it is hard to think of any reasonably complex software project that does not give special attention to the architecture of the system. In the mobile world, however, it seems that much remains to be done.

Mobile application developers typically obtain a good part of the guidance for how to implement their applications directly from platform developers. Apple and Google, for example, continuously publish design and programming guidelines that are meant to instruct developers in the creation of high quality apps. But when it comes to software architecture, the advice offered by some platform developers is rather narrow.

In the case of iOS, for example, Apple advises a particular flavor of the MVC pattern as the best way for programming iOS apps [10]. Following this vision, iOS applications are typically organized in Model-View-ViewController triplets that collaborate using the communication mechanisms that are endemic to the platform — i.e Delegates, Target/Actions, Outlets, Key-Value Observers and Segues. Using the MVC architectural pattern to structure an entire iOS application comes handy when developing simple applications; for it only requires a handful of classes to support all the application use cases. In contrast, following the same path when creating non-trivial applications may lead to known problems like the *Massive View Controllers* [1] issue.

While the *Massive View Controllers* issue can be considered as a phenomenon endemic to the iOS platform, it does serve to illustrate one of the challenges that software archi-

[1] http://www.objc.io/issue-1/lighter-view-controllers.html

DeMobile'15, August 31, 2015, Bergamo, Italy
© 2015 ACM. 978-1-4503-3815-8/15/08...$15.00
http://dx.doi.org/10.1145/2804345.2804350

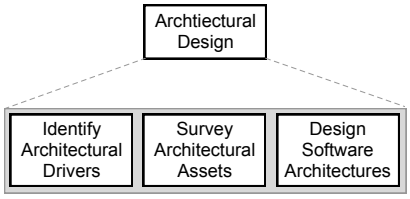

Figure 1: Mobile Architecture Process

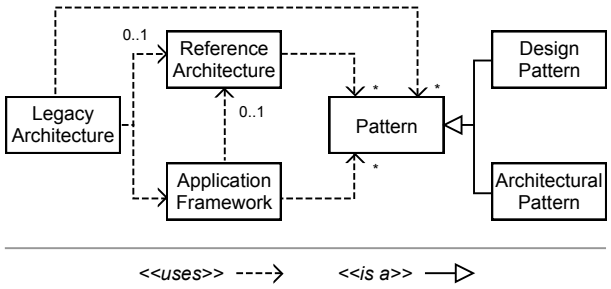

Figure 2: Architectural Assets Metamodel adapted from [3]

tecture faces in the mobile world: the **false intuition that the software architecture of mobile applications is a solved issue** and that following the guidelines of platform developers is enough to guarantee that a mobile application would have a good architectural design.

3. MOBILE SOFTWARE ARCHITECTURE PROCESS

To address this issue, our paper highlights the important role that the architectural design plays in the development of complex mobile applications. Our aim is not to propose a new architectural methodology; instead, we present **a high-level process that can guide application developers in the task of designing a suitable software architecture** for their mobile apps.

Several publications discuss the architectural design process [1]. We extract the most relevant stages of the architecture process for mobile applications, as shown in Figure 1: *Identify Architectural Drivers*, *Survey Architectural Assets* and *Design Software Architecture*. We will describe each of them in the following sections. Moreover, to provide a better idea of how each of the activities in the process maps onto a real life project, we will reflect over the experience that the software engineering team of Coursera[2] had while re-designing the architecture of their Android and iOS apps, based on two presentations given to the developer community [5] [14].

3.1 Identify Architectural Drivers

The first stage of the process is the identification of the main *architectural drivers* of the application. According to [1], the *architectural drivers* of an application are the collection of functional, quality and business requirements that shape the architecture of the application.

Though different mobile applications will likely have different *architectural drivers*, two quality attributes that may define the shape of the architecture of a large subset of mobile applications are: modifiability and testability.

Mobile applications today need to be designed for change. On the one hand, they need to adapt to the shifting business requirements that frequently imply the addition, update and deletion of some features of the application. On the other hand, since mobile applications frequently rely on third party libraries and Web APIs, and these are bound to change [4], having an architecture that allows for the easy replacement, deletion and addition of functionality, while managing the cost and scope of change, is an important goal.

Update cycles of mobile applications, on the other hand, are lengthier than those of web applications. Patching an application bug implies the release of an update for the application which needs to go through the revision and approval process of platform developers. Identifying and catching errors before an application is released is, therefore, an important concern of mobile software engineers. In consequence, having an application structure that enables easy testing for as many components as possible, is an increasingly important *architectural driver*.

In the case of Coursera, for example, the main quality attributes that shaped the structure of their architecture were: modifiability, reusability and testability. On the one hand, they needed to allow developer teams to work independently on different features of the application, which required a highly modular structure. On the other hand, they wanted to allow external developers to create their own features while leveraging the capabilities of the existing infrastructure, which in turn required that some of the core functionalities of the application had to be reusable and easily shared across developer teams. Finally, the software engineering team also wanted to enable more testing across the different components of the application to reduce the probability of shipping faulty applications.

3.2 Survey Architectural Assets

Once application developers have identified the *architectural drivers* that may have an important influence on the final structure of the application, they should survey the field looking for inspiration that can help them solve the challenges imposed by these drivers. Since the final goal is to create a suitable software architecture for a mobile application, this survey should focus on *architectural assets*.

Architectural assets are defined by [3] as a set of reusable resources that can significantly help architects in their work since it reduces the number of things that the architect needs to be concerned with. *Architectural assets* are, therefore a good source of inspiration for mobile software engineers.

Although [3] provides an exhaustive list of the *architectural assets* that may be used by a software architect, in the following sections we present a subset that have special value in the mobile context. Figure 2 shows an adaptation of the metamodel presented by [3] that highlights the relationships that exist between some of these assets.

3.2.1 Architectural Patterns

An architectural pattern[3] is a coarse grained pattern that provides an abstract framework for a family of systems [13].

[2]https://www.coursera.org/

[3]also referred to as an architectural style in [7]

They help define the fundamental structure of the application [2] and specify a design vocabulary, constraints on how that vocabulary is used, and semantic assumptions about that vocabulary [7]. Examples of architectural patterns are: Layers, Pipe and Filter, Client/Server, N-Tier, Service Oriented and Message Bus [13, 7].

An architectural pattern of special relevance for rich mobile applications is the *Layers pattern*. The *Layers pattern* is described by [2], as an strategy to effectively divide the responsibilities of an application across the objects that compose it. Figure 3 (adapted from [13]), for example, shows a *reference model* of a rich mobile application organized according to the *Layers pattern*.

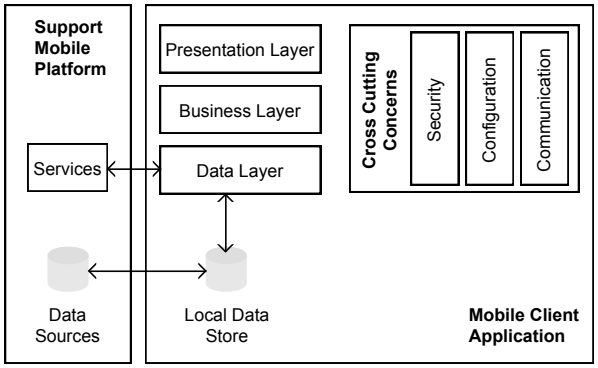

Figure 3: Layers Pattern adapted from [13]

Another pattern of notable importance for mobile applications is the MVC architectural pattern. However, as we discussed in Section 2, this pattern needs to be applied with caution as an architectural style, for it seems to be better suited for structuring thin or simple mobile apps, rather than rich or complex ones.

3.2.2 Reference Architectures

A reference architecture captures the basic building blocks and architectural design decisions of a system in certain application or technology domain [6]. They also represent reusable architecture knowledge in the form of generic artifacts, standards, design guidelines, architectural styles or domain vocabulary [6].

A particular reference architecture that has been widely discussed within the iOS developer community is VIPER [9]. VIPER is a reference software architecture, introduced by Mutual Mobile[4], with the aim of facilitating the design of software architectures for rich iOS apps. The main components of VIPER are: Views, Presenters, Interactors, Routers, Entities and Data Stores, and the relationships between them are shown in the Figure 4.

Although VIPER was introduced within the context of iOS development, nothing in its structure ties the architecture to a particular technology or mobile platform. In fact, when designing the architecture of their mobile apps, the software engineering team of Coursera used VIPER as the initial point from where the specific architectures for their iOS and Android applications were then extracted.

3.2.3 Application Frameworks

[4]http://www.mutualmobile.com/

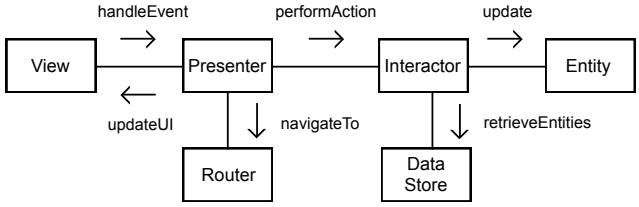

Figure 4: VIPER Architecture adapted from [9]

An application framework, as defined by [3], represents the partial implementation of a specific area of an application. In the mobile context, application frameworks play a fundamental role since they allow developers create sophisticated applications providing only the code that implements the differential aspects of their apps.

An application framework serves as an extensible skeleton that is customized by the methods defined by the user [11]. In consequence, when developing a mobile application, developers often find themselves writing code that is called by the methods within the frameworks. In iOS, for example, this will be the case of the code that developers write in UIViewController methods like viewDidLoad, viewWillAppear, viewDidAppear and similar. In Android, instead, the code written in Activity methods like onCreate, onResume or onPause would be equivalent examples.

In conclusion, since in the mobile context application frameworks are almost unavoidable, they are an important *architectural asset* and one that application developers need to get to know very well.

3.2.4 Legacy Architectures

Legacy architectures are an asset that can be defined as any kind of existing architecture. In the mobile world, this is an asset that is frequently available as software organizations typically publish different versions of their apps, and even have more than one application on the stores. The architectures used to develop all of these applications represent an important source of inspiration for mobile software architects, because all of them provide a privileged window to the structure of the system, help identify which strategies were effective, and hihglight some of the constraints imposed by the organization for granting coherency among its products.

3.3 Design the Software Architecture

The final step of the process, consists in the creation of a software architecture that is suitable for the mobile application. Inspiration for this task may come from different places, but according to [12], *theft, method and intuition* are among the most relevant. *Theft* refers to the inspiration that software architects may take from existing architectures, architectural patterns, reference models, application frameworks, and other sources (as we did in previous section); *Intuition* refers to the experience and skills of the software architect; and *method* refers to the systematic process by which the final architecture of an application is derived from its requirements.

Though several methods have been proposed in the research community, a particular method that aligns with the concepts that we have introduced so far is the *Attribute Driven Design* method, which decomposes a system or sys-

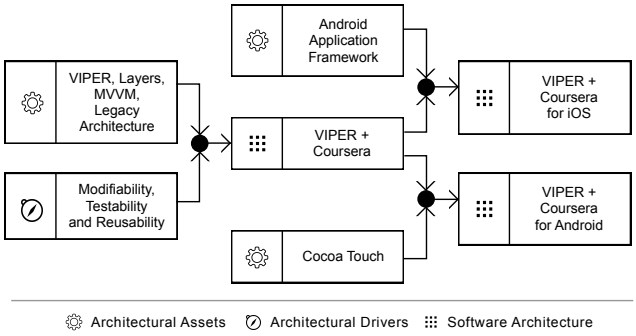

Figure 5: Coursera Mobile Software Architecture Process

tem element by applying architectural tactics and patterns that satisfy its driving requirements [1].

Applying this, or other method, in the mobile context however, should be done bearing in mind that the application frameworks used by the different mobile platforms may have an important influence over the shape of the final architecture. In consequence, a mobile software engineering team should first attempt to create a general architecture that satisfies the main *architectural drivers* at a *platform independent* level, and then apply some refinements over this general architecture to obtain the *platform specific* architecture for each of the target platforms. This is, in fact, the strategy used by the engineering team of Coursera, as discussed next.

4. THE COURSERA CASE

To see more clearly how the different steps of the process can provide a clearer view of the architectural design, we present the main drivers and assets that lead to the final architectural design of the Coursera application for Android and iOS.

In Figure 5 we can see how the VIPER *reference architecture* was a core asset that informed the final architectural design of the Coursera application. Other assets like *Layers pattern*, the *MVVM* architectural pattern [8], and the experience had with the previous architectural design of the application also help shape the final structure. On the other hand, requirements like allowing external developers to create features for the application, as well as allowing internal developer teams to work on different features independently were some of the architectural drivers that helped brew the final solution. Another aspect that is evident in Figure 5 is that the application frameworks used in Android and iOS also affected the architectural design process and led to the creation of platform specific architectures for each of the target platforms.

5. DISCUSSION AND CONCLUSIONS

In this paper we aimed to broaden the perspective of application developers regarding mobile software architectures, which become more relevant as mobile applications increase their complexity. Besides highlighting the importance of the architectural design, we presented a high-level process composed of three stages, aiming to guide developers in the creation of suitable architectures for their mobile apps.

The proposed process has three benefits: *1)* it is based on well known architectural processes [1, 3]; *2)* it maps the main concepts and knowledge of software architecture on to the mobile world; *3)* it advocates for a different view of mobile software architectures in which process and methodology are favored over predefined solutions. Finally, though the process represents a step towards the definition of architectural design practices for the mobile world, we think it is still too general when it comes to the actual design of the software architecture, and thus this work paves the way for further research to tailor some of the existing architectural methodologies for the mobile context.

6. REFERENCES

[1] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison Wesley, second edition, 2003.

[2] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern Oriented Software Architecture - A System of Patterns*. John Wiley & Sons, Inc, 1996.

[3] Peter Eeles. Understanding architectural assets. *7th IEEE/IFIP Working Conference on Software Architecture, WICSA 2008*, pages 267–270, 2008.

[4] Tiago Espinha, Andy Zaidman, and Hans-Gerhard Gross. Web API growing pains: Loosely coupled yet strongly tied. *Journal of Systems and Software*, 100:27–43, 2015.

[5] Mustafa Furniturewala. Building Modular iOS Apps. Using Swift, iOS 8 and MVVM to maintain a complex mobile app with a large team. https://realm.io/news/modular-ios-apps/, 2015.

[6] Matthias Galster. Software Reference Architectures : Related Architectural Concepts and Challenges. pages 5–8, 2015.

[7] David Garlan. Software architecture: a travelogue. *Proceedings of the on Future of Software Engineering*, pages 29–39, 2014.

[8] Raffaele Garofalo. *Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern*. Microsoft Press, Mar 2011.

[9] Jeff Gilbert and Conrad Stoll. Architecting iOS Apps with VIPER. http://www.objc.io/issue-13/viper.html, Jun 2014.

[10] Apple Inc. Cocoa Core Competencies. https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html, Sep 2013.

[11] Ralph E. Johnson and Brian Foote. Designing Reusable Classes. *Journal of Object-Oriented Programming*, 1(2):22–35, 1988.

[12] Philippe Kruchen. Mommy, Where Do Software Architectures Come From? *1st International Workshop on Architectures for Software Systems, Seattle*, 1995.

[13] J.D. Meier, Alex Homer, David Hill, Jason Taylor, Lonnie Wall, Prashant Bansode, Akshay Bogawat, and Rob Boucher Jr. *Mobile Application Architecture Guide*. Microsoft, 2008.

[14] Yixin Zhu. Mobile Architecture and Android Design. https://youtu.be/iWf11tRRBB4, 2014.