A Framework for iOS Application Development

Youcong Ni  
State Key Lab of Software Engineering, Wuhan University, Wuhan, China  
Faculty of Software Fujian Normal University, Fujian, China  
Email: youcongni@foxmail.com

Bei Chen  
Faculty of Software Fujian Normal University, Fujian, China  
Email: cbfjnu@gmail.com

Peng Ye  
College of Mathematics and Computer, Wuhan Textile University, Hubei, China  
Email: whuyyp@126.com

Chunyan Wang  
Faculty of Software Fujian Normal University, Fujian, China  
Email: wcychunyanwcy@gmail.com

Abstract—In order to improve efficiency and quality of mobile application development on the iOS platform, this paper proposes a framework named AF4iOS, which is designed based on class libraries and existing software frameworks on iOS platform. The AF4iOS framework is divided into three independent layers: user interface (UI) layer, domain layer and resource layer. The AF4iOS framework encompasses a variety of components, which encapsulate the usage and accessing of various resources, such as UI, data, web service and communication. These extensible and reusable components can accelerate the progress of development and enhance the quality of product. Finally, the availability and effectiveness of the AF4iOS framework is demonstrated through a case study.

Index Terms—Application Framework, iOS Application Development

I. INTRODUCTION

At the 2011 World Wide Developer’s Conference, Steve Jobs announced that Apple had sold over 200 million iOS devices, with over 225 million registered customers. These customers have downloaded over 14 billion apps so far, resulting in over $2.5 billion paid to iOS developers over the last three years [1]. In 2013, there are 7.1 billion people in the world and number of mobile subscriptions is almost 6.8 million [2]. At present, the number of iOS apps in the Apple’s App Store has already been more than 700,000 and is still growing. In such competitive environment, iOS apps should be quickly developed and deployed in order to obtain and maintain competitive advantage. Meanwhile, these apps must meet the increasingly strict quality requirements of App Store. As a result, iOS apps must be developed efficiently and keep high quality.

Software framework plays important role in improving efficiency and quality of apps development [3]. Many frameworks or middlewares [4, 5] for specific mobile computation domains has been proposed. Based on class libraries of iOS, there are a few of open source software frameworks [6], which encapsulate the usage and accessing of specific resources in order to ease the iOS apps development difficulty level. For example, MagicalRecord framework can provide capability to simply access database resource through encapsulating CoreData [7] class library. Web resources can be conveniently accessed by means of AFNetworking framework worked on CFNetwork [8] class library. However, how to integrate existing software frameworks to form unified iOS apps development framework have not been well solved.

Aim at this problem, the framework, named AF4iOS, is proposed in this paper. The AF4iOS framework, designed on the basis of iOS class libraries [9] and related open source software frameworks [10], is divided into three independent layers: user interface (UI) layer, domain layer and resource layer. In the AF4iOS framework, a variety of components that encapsulate the usage and accessing of various resources [11], such as UI, data, web service, and communication are encompassed. And these extensible and reusable components can accelerate the progress of development and enhance the quality of product [12].

The remainder of this paper is organized as follows. In the next section, the AF4iOS framework is illustrated in detail; Section 3 demonstrates applicability of AF4iOS through a case study; Finally, Section 4 concludes the paper and future works.

II. AF4IOS FRAMEWORK

The AF4iOS framework, shown in Fig. 1, is split into three layers by the layered architecture pattern [13]: user interface (UI) layer, domain layer and resource layer. UI
layer is used to implement the GUI and provide the capability of interaction between users and iOS apps. Business logics and business entities are encapsulated by domain layer that provide UI layer with the interfaces of business functions. Resource layer encapsulates the usage and access of various resources including data, web service and communication, providing the operation interfaces of resources to domain layer. The functions of iOS apps can be implemented by means of interaction and cooperation among the three layers. AF4iOS will be illustrated in detail as follows.

A. Resource Layer

As shown in Fig. 2, Resource layer includes data component, web service component and communication component that encapsulate the access and use of data, web services and communication resources, respectively. These components provide a few of operation functions for corresponding resources.

- **DBData** and **FileData** in the data component can operate database and files respectively. And **SoapWS** and **RestWS** can access soap and restful style web service resources in the web service component separately. In communication component, **BLE** can provide the capability of operating Bluetooth low energy (BLE) while **SMS** can be used to send and receive the short messages. Due to the limitation of paper length, one type of resource operation is demonstrated in data component, web service component and communication component.

- **DBData Component**

  As shown in Fig. 3, **DBData** component provides two kinds of operations by **DBDataContext** class. One is called CRUD operations including create, read, update, and delete objects. And other is transaction operations such as begin, rollback and commit transaction. **DBDataContext** class implements responsibilities on the support of its attribute **managedObjectContext**, which can be obtained by invoking **getManagedObjectContext** method in **DataHelper** class. The method **getManagedObjectContext** can complete three tasks: 1) create the instance of **NSPersistentStoreCoordinator** based on the position of database file; 2) create the instance of **NSManagedObjectModel** in terms of managed object model file; 3) create the instance of **NSManagedObjectContext** and associate this instance with the instances created in task 1) and 2).

- **SoapWS Component**

  **SoapWSCContext** class and **SoapWSDelegate** delegate in **SoapWS** component can provide a universal method to asynchronously request web service and process its response, as shown in Fig. 4. The specified operation of a web service can be called by designating the name of the web service, operation and arguments in **callSoapWS** method of **SoapWSCContext** class. And different callback method in **SoapWSDelegate** delegate will be activated. When the web service successfully executed, **handlerResult** method will be called, otherwise **handlerFault** method will be called.

To fulfill its responsibilities, **callSoapWS** method needs the supports of SudzCService component, **SoapWSHelper** class and the web service mapping file. **SudzCService** component can be automatically generated after **SudzC** tool receives and processes the WSDL file. **SudzCService** implements marshalling and unmarshalling soap packages. Meanwhile, **SudzCService** can apply Http protocol to invoke a web service in aid of **CFNetwork** class library of iOS platform. Client can address the web service by the local service class in **SudzCService**. The web service mapping file defines associations between the name of a web service and the name of local service class. **SoapWSHelper** class can find the local service class by matching in the web service mapping file according to the inputted the name of web service. Furthermore it
creates instance of local service class by reflection mechanism [14] and return this instance to SoapWSContext. Finally SoapWSContext process response on the support of the instance sent by SoapWSHelper.

**BLE Component**

As shown in Fig. 5, BLEContext class, which supplies the functions of access BLE peripheral data by calling its readValue and writeValue method need three UUID (Universally Unique Identifier) parameters of peripheral, service and characteristic. Specifically, the following steps describe process of reading BLE peripheral data in readValue method.

1) Call scanForPeripherals method of CBCentralManager and retrieve the list of peripherals named prelist;

![Figure 4. SoapWS component](image)

2) Find the peripheral of designated UUID, named perph, in prelist. If perph is not found, handlerPeripheralNotFound method in BLEContextDelegate will be activated and the readValue method will quit;

3) Call the connectPeripheral method of CBCentralManager to try to connect with the perph. If the attempt ends in failure, handlerConnectFail method will be activated and then the readValue method will quit;

4) Invoke discoverServices method of perph and retrieve the list of services named sevclist;

5) Find the service of designated UUID, named sevc, in sevclist. If sevc is not found, handlerServiceNotFound method in BLEContextDelegate will be activated and the readValue method will quit;

6) Invoke discoverCharacteristics method of perph and retrieve the list of characteristic named chrtlist;

7) Find the characteristic of designated UUID, named chrt, in chrtlist. If chrt is not found, handlerCharactNotFound method in BLEContextDelegate will be activated and the readValue method will quit;

8) Reading data from chrt by calling readValueForCharacteristic method of perph. And then handlerReadResult method in BLEContextDelegate will be activated.

It is a similar process for writeValue method in BLEContext.

**B. Domain Layer**

In domain layer, as shown in Fig. 6, BusinessService and BusinessEntity component are contained. The former encapsulates business logics and provides business services on the basis of data, web service and communication foundational services. And the latter represents business entities and their relationships.

There are mainly three kinds of foundational services by defined DataService, WSService and CommService component, respectively. The DataService component includes DBDataService and IconDataService class. DBDataService class encompasses dbDataContext static attribute whose type is DBDataContext class from DBData Component in resource layer, and sharedInstance static method. By calling sharedInstance, any business service class, which needs to access SQLite database, can obtain the single instance of DBDataContext based on singleton pattern. WSService and CommService component are built on the basis of WebService and Communication in resource layer, respectively. And their definition are similar to DBDataService.

![Figure 5. BLE component](image)

**C. UI Layer**

As shown in Fig. 7, UI Layer adopts MVC patterns [15], and provides abstract view controller and view model: ViewController and ViewModel.

ViewController class encapsulates the model data for rendering the UI. There are verifyModel and fireModelChanged method in it. The verifyModel method is abstract method for finding errors in ViewModel and storing error attributes, corresponding error messages defined in ErrorInfo class to error list. When ViewModel is changed, the fireModelChanged
method can be called. The execution process of `fireModelchanged` method follows the steps: 1) check for errors in the `ViewModel` by calling `verifyModel` method; 2) If any error is found, `modelInError` method in the `ViewModelDelegate` will be invoke, otherwise `modelChanged` method in the `ViewModelDelegate` will be called.

`ViewController` inherits the `UIViewController` of UI Kit class library on iOS platform and the `initModel` and `updateView` method are provided. The `initModel` is abstract method whose responsibility is to obtain business entities by invoking the business service encapsulated in domain layer, and transform these entities into data of `ViewModel`. The `updateView` method can used to render view on the basis of data in `ViewModel`. The two methods of `initModel` and `updateView` are called in turn by the `viewDidLoad` method of `ViewController`. Meanwhile, the `ViewController` implements `modelChanged` and `modelInError` method in `ViewModelDelegate` delegate. The `modelChanged` will call the `updateView` to bring view into correspondence with `ViewModel`. And the `modelInError` method shows error message dialog.

### III. Case Study

A iOS mobile application, called iStep, is developed based on the AF4iOS framework. Three core functions including data display, data sharing and data synchronization in iStep are taken as case study to show the validity of AF4iOS. A simple description for these functions is as follows:

1) The user can look up her/his sport data in a certain day such as steps, distance, calorie and percentage of goal by using the data display function;
2) The data sharing function can used to upload the sport data in selected date to social network.
3) The data on peripheral pedometer can synchronize with iStep based on BLE communication by means of the data synchronization function.

The UI of three functions is shown in Fig. 8. Sport data of the last day and the next day can show through clicking “left” and “right” button. The “share” and “sync” button are used to perform the data sharing and data synchronization, respectively.

For implementing above mentioned three functions, the design at the component level is given based on the AF4iOS framework. Specifically, the design of components in UI layer and domain layer are presented as follows.

#### A. Design Components in Domain Layer

In domain layer, business entities and business service relevant to the three functions in the case need to be defined. As shown in Fig. 9, the `User`, `StepData`, `SportPlan` three classes and their association are designed to represent business entities and their relationships.

Meanwhile, the four methods of `getStepData`, `getSportPlan`, `shareStepData` and `synchroStepData` are used to define business logics. They are encapsulated in the `StepDataService` class to describe business service.

The `StepDataService` needs to access resources of database, web service and BLE in order to perform corresponding business functions of retrieving, sharing and synchronizing sport data. To simplify design, `StepDataService` can reuse `DBDataService`, `SoapWS-Service` and `BLESerivice` foundation classes that are predefined in domain layer of AF4iOS.

#### B. Design components in UI Layer

For this case, the two classes of `DisplayViewModel` and `DisplayViewController` in UI Layer are designed by reusing the `ViewModel` and `ViewController` class of AF4iOS. As shown in Fig. 10, the `DisplayViewModel` inherits `ViewModel` and defines the attributes such as date, step, distance, calorie and percentage for rendering view in Fig. 8. The `verifyModel` is an overloading method which can check correctness of value of these attributes. The `DisplayViewController` implements two abstract methods of `initModel` and `updateView` derived from `ViewController`. The `initModel` calls `getStepData` and `getSportPlan` method in `StepDataService` to retrieve `StepData` and `SportPlan` entity, respectively.
These entities are used to set attributes of DisplayViewModel. The updateView method renders view by using data in DisplayViewModel.

In addition, the four methods of shareButtonClick, syncButtonClick, leftArrowButtonClick and rightArrowButtonClick can deal with corresponding events arising from UI. The shareButtonClick and syncButtonClick call shareStepData and SyncStepData method in StepDataService to perform corresponding business tasks. The executing process of leftArrowButtonClick includes two steps. The first step, similar to initModel, is to update DisplayViewModel by the last day.

Figure 8. UI of core functions in iStep

Figure 9. Domain design in case study

The second step is to call fireModelChanged method to update view. Due to similarity between leftArrowButtonClick and rightButtonClick, the latter is omitted in this paper.

Figure 10. UI design in case study

IV. CONCLUSION

A mobile application development framework AF4iOS is presented in this paper. Following the features of layered model, AF4iOS is split into three layers of UI layer, domain layer and resource layer which respectively handle view rendering, business logic and resource access. Furthermore, a variety of classes, which encapsulate different foundational functions, are defined based on class libraries and existing software frameworks on iOS platform in each layer of AF4iOS. These predefined classes will be reused to design user-defined classes according to requirement of application. As a result, AF4iOS is extensible and reusable framework that can help developer to speed up the development of the application and promote the quality of product. Finally, a case study of mobile application named iStep show the validity of the proposed framework.

In the future, based on AF4iOS framework we will define the iOS apps design steps that is seamlessly integrated with the agile processes so as to form a kind of agile iOS apps design method.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China under Grant (No. 61305079), the open fund of State Key Laboratory of Software Engineering(No.SKLSE2012-09-28), the Natural Science Foundation of Fujian Province of China under Grant(No.2012J01250), the project of preeminent youth fund of Fujian province(No.JA12471), the project of Fujian education department(No.JA12077, No.JA12080), outstanding young teacher training fund of Fujian Normal University (No.fjsdjk2012083), Science and Technology Program Key Project of Fujian Province of China (No.2011H6006), Science Research Foundation of Hubei Provincal Department of Education under Grant(No.B20111607).

REFERENCES


**Youcong Ni** is a Ph.D. and lecturer in Faculty of Software Fujian Normal University, Fujian, China. His current research interest includes software architecture, mobile cloud computing and search-based software design.

**Bei Chen** is a master student in Faculty of Software Fujian Normal University, Fujian, China. His current research interest includes software architecture and mobile cloud computing.

**Peng Ye** is a Ph.D. and lecturer in College of Mathematics and Computer, Wuhan Textile University, Hubei, China. His current research interest includes software architecture, mobile cloud computing and ontology-based software design.

**Chunyan Wang** is a master student in Faculty of Software Fujian Normal University, Fujian, China. Her current research interest includes software architecture and search-based software design.