Framework Library With Guidelines For Effective TV Application Development

Nemanja Kovačev, Veljko Ilkić, Dejan Nađ, Nikola Vranić

***Abstract* – TV application development is becoming an important field of software industry. As with the software development in general, the aim is to build a high-quality and cost-effective application which is as fast to build and as easy to maintain as possible. We noticed several issues with the existing approaches to TV application development. This paper presents a framework with guidelines for effective TV application development along with benefits of using this kind of framework in solving those issues.**

***Index Terms* – TV application; Android; Java; MVP; framework; core; SDK.**

I. Introduction

During the last decade, Android operating system [1] and applications have become a significant part of software industry. Android as an operating system has become omnipresent in our surroundings – in phones, cars, wearables, etc. One of such fields is TV application development. Along with the increasing number of developers working on those applications, a number of different approaches emerged. During the development of TV applications we noticed that most of the applications had the same code structure and a lot of common features. We also observed that the existing frameworks and methods for TV application development were not efficient enough for the pace we needed to have. Regarding that fact, and in order to speed up development process, reduce code duplication and improve robustness of a product, we have decided to construct a framework library to make a project kick-off phase easier. We designed the application core to be the starting point for the development of TV applications. The core is written in Java and is platform independent as shown in Fig. 1.

Nemanja Kovačev, RT-RK Institute for Computer Based Systems, Narodnog fronta 23a, Novi Sad, Serbia (e-mail: nemanja.kovacev@rt-rk.com).

Veljko Ilkić, RT-RK Institute for Computer Based Systems, Narodnog fronta 23a, Novi Sad, Serbia (e-mail: veljko.ilkic@rt-rk.com).

Dejan Nađ, RT-RK Institute for Computer Based Systems, Narodnog fronta 23a, Novi Sad, Serbia (e-mail: dejan.nadj@rt-rk.com).

Nikola Vranić, RT-RK Institute for Computer Based Systems, Narodnog fronta 23a, Novi Sad, Serbia (e-mail: nikola.vranic@rt-rk.com).

The core connects the UI (User Interface) on one side with the backend and database APIs (Application Programming Interfaces) on the other side.

The way how the core is connected to other components as well as the data flow between them is shown in Fig. 2. The diagram shows how it is possible to make a custom component by API implementation, i.e. a backend handler which communicates with a different backend or a player handler which uses a different player. The diagram also shows how we introduced entities which are common for all applications.

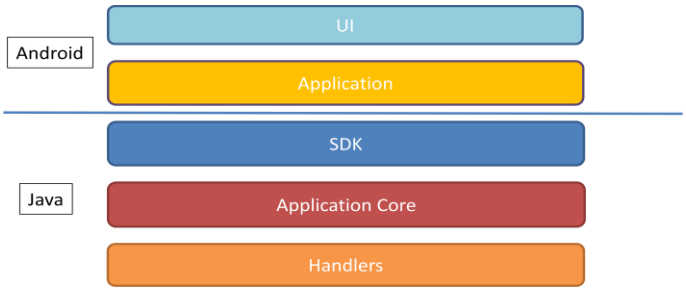

Fig. 1. Architecture overview

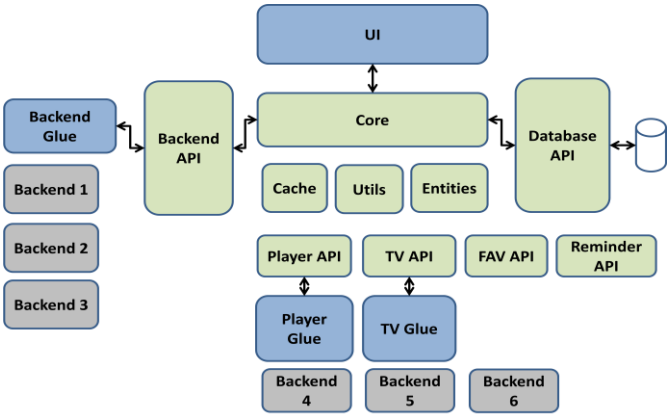

Fig. 2. Top view of the framework

II. THE DESIGN PATTERN

In this chapter we will present the design pattern we are using for the development of TV applications. During the application development we came to a conclusion that the MVP (Model-View-Presenter) pattern [2] is the best design pattern for TV application development. The application core presents the model, Android application itself is the view and SDK is the presenter. The model is an interface which defines the data to be displayed, the view is a passive interface that displays data and the presenter is the "middle-man" between the model and the view. User events are passed from the view to the presenter which updates the model. The model sends events to the presenter which updates the view as shown in Fig. 3. In this framework the view is named the Scene and the presenter is named the Scene Manager. The Scene sends information to the Scene Manager via a scene listener interface and the Scene Manager propagates that information to the model. The model sends updated data to the Scene Manager which updates the Scene via the refresh method.

III. COMPONENTS OF THE FRAMEWORK

Here we will list and describe the crucial components of our framework: Activities, Fragments, Scenes, Scene Managers, SDK class and Information Bus.

A. Activities and Fragments

The Main Activity [3], [4] inside Android TV application can be considered as an entry point for the framework, and at the same time as the View entity of whole system. It is in charge of drawing Android elements on the screen on request from the core module by inflating fragment views [3] defined in the scenes.

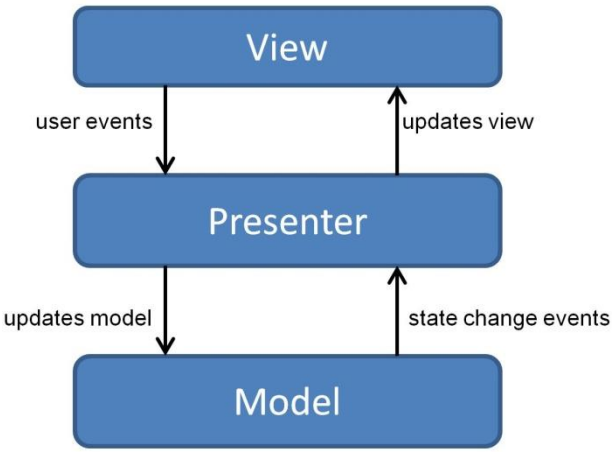

Fig. 3. The Model-View-Presenter design pattern

It is registered on Information Bus, which will be explained later in the text, as the listener of events from the core - it waits for show, hide and destroy events submitted by the *AppHandler* and does the appropriate actions on the scene views.

B. Scenes and Scene Managers

Scenes are just passive Views [2] which contain fields and methods. There are 4 graphical layers defined inside the framework as shown in Fig. 4. This enables us to cover all standard use cases for different TV applications and designs. A Scene can be shown and managed in different graphical layers which is more convenient for proper user experience. So a scene can be placed inside the default layer, in overlay, as notification and in the global layer (i.e. a global widget – such as the volume bar – if it is visible, no other entity will accept key events).

Scene managers are responsible for scene control – Scene managers communicate via *triggerAction* method which requires action type. Based on the type of action a manager decides how a scene should be managed. The scene can be shown in specified graphical layer, hidden or destroyed.

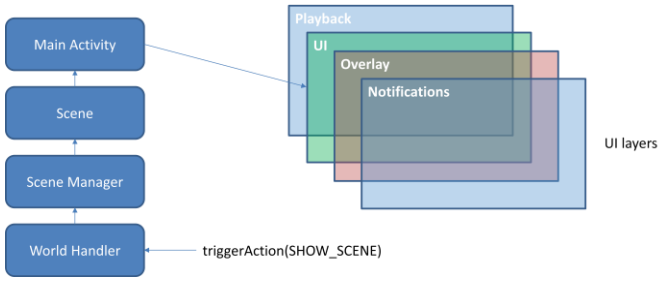

Fig. 4. Event propagation

C. SDK Class

Software Development Kit (SDK) is a Singleton class [2] which contains instances of all handlers. Handler methods are asynchronous and data is retrieved through the *AsyncDataReceive* callback. In order to obtain the needed data, a Scene will call its Scene Manager which will then directly ask SDK for data via an appropriate handler and the data will be received by the *AsyncDataReceive* callback.

The *WorldHandler* class contains instances of registered scene managers. That class enables accessing a scene manager that belongs to some other scene.

The *AppHandler* class controls the scenes switching mechanism – this class takes care of which scene is active (displayed). When Scene Manager's action is triggered, the *AppHandler* will delegate an event to the corresponding scene

(depending on the triggered action type) and then the Scene will notify the AppHandler which will ensure scene switching.

The list of all handlers contained in the SDK is shown in Table I.

TABLE I
THE LIST OF ALL HANDLERS IN THE SDK CLASS

WorldHandler	AppHandler
DatabaseHandler	AccountHandler
BluetoothHandler	ChannelsHandler
DisplayHandler	EpgHandler
FavoritesHandler	NotificationHandler
PlayerHandler	PrefsHandler
ProfilesHandler	ReminderHandler
SearchHandler	TimeHandler
TvHandler	VodHandler
VolumeHandler	UpdateHandler
ParentalControlHandler	GuideHandler
RegionHandler	ConfigurationHandler
PaymentHandler	LanguageHandler
PackagesHandler	DeviceHandler
CategoryHandler	TrialHandler
ItemInfoHandler	

D. *Information Bus*

Information Bus provides support for sending and receiving events between different application modules. It's a part of the Observer design pattern [5]. Every event has an ID and a data object and can be submitted and received by using the Information Bus.

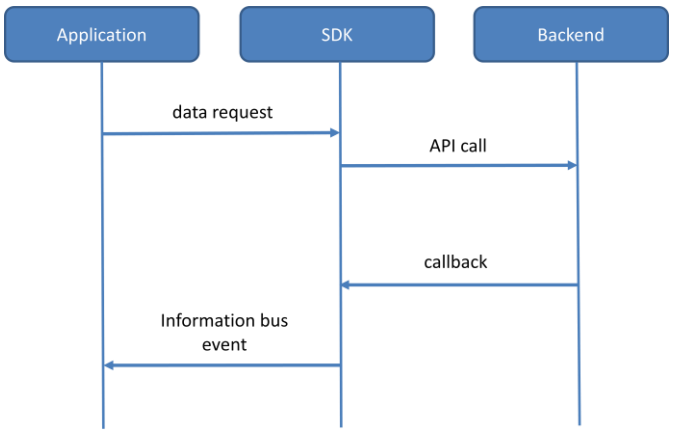

Fig. 4. Example of communication between application and backend

IV. THE USUAL SCENARIO

Here we will describe the usual scenario of scene creation and scene destruction in our TV applications.

A. *Scene creation*

The first step is to create a Scene class which extends the generic UI core scene. The layout can be created in two ways – one is to create a layout xml file [3] and the other way is to dynamically create a layout inside a scene class. The second step is to create a scene listener interface with methods needed by the scene. This listener is used for sending request to the Scene Manager. The third step is to create a Scene Manager class which extends the generic UI core scene manager followed by an override of the *createScene* method and creation the scene object in the Scene Manager. And the final step includes the call of the *setScene* method at the end of *createScene* method in order to save the Scene instance.

B. *Scene destruction*

The destruction of a visible scene is performed by the scene's destroy method. Scene Manager triggers the destroy action and calls the scene's destroy method. Scene notifies the *AppHandler* that it is destroyed. The *AppHandler* submits the scene destroy event which is received by the Main Activity which then destroys the scene's view.

V. MEASUREMENT

In this chapter we will present the measurement we conducted.

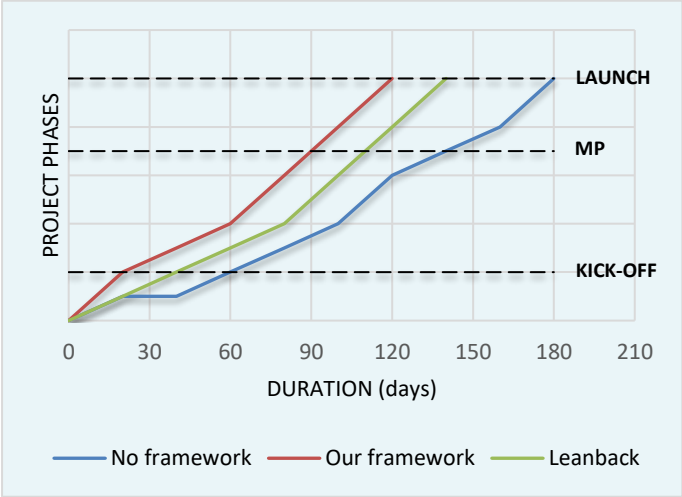

Fig. 5. Measured project timelines

Since our team is among the first teams to develop TV applications there are still not many developers dealing with the issues mentioned in our paper so the data in available literature is scarce at the moment. The measurement was based on our considerable experience in constructing TV

applications. Fig. 5 shows three projects with the same set of features developed by our team. Each project was constructed by different approach – using no framework whatsoever, using our framework and using Android's Leanback library. The approach with no framework is the hardest and slowest because there is no starting point so the development has to be done from scratch. The approach with Leanback library has its advantages in terms of templates and predefined features but it is hard to customize. The approach with our framework is the fastest since we have a good starting point – a basic TV application with basic features (Channel list, Channel zapper, TV guide, VoD) which can easily be customized.

VI. CONCLUSION

In this paper we have presented our approach to TV application development. Our experience shows that the usage of our framework library considerably reduces the amount of code in the application, makes the code more readable and easier to maintain and also shortens the time needed for coding thus making TV application development significantly more effective. By utilizing this framework we managed to reduce time and effort for TV application development. The possible downside of this approach is that it can be complicated during the first few weeks of development for programmers who haven't had previous experience with TV application development but that gets compensated in the later phases of the development.

Although our method is more efficient in comparison to other methods of TV application development [6], [7], [8], [9], [10] there is still room for further improvements such as the introduction of more common features and sub-library modules for TV centric applications. Other improvements include data and performance optimizations as well as support for different Android versions and screen resolutions within the framework by utilizing dynamic UI support.

ACKNOWLEDGMENT

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, under grant number: III_044009_2.

REFERENCES

[1] Official Android website – https://www.android.com
[2] K. Mew, Android Design Patterns and Best Practice, Birmingham, United Kingdom: Packt Publishing Ltd, 2016.
[3] J. Horton, Android Programming for Beginners, Birmingham, United Kingdom: Packt Publishing Ltd, 2015.
[4] B. Phillips, C. Stewart, B. Hardy, K. Marsicano, Android Programing: The Big Nerd Ranch Guide, 2nd edition, Atlanta, USA: Big Nerd Ranch Guides, 2015.
[5] H. Schildt, Java: The Complete Reference, 9th edition, USA: McGraw-Hill Education, 2014.
[6] E. G. Lima and R. de Andrade Lira Rabêlo, "An architectural model for communication between the iDTV and mobile devices," *2015 International Conference on Computing, Networking and Communications (ICNC)*, Garden Grove, CA, pp. 1102-1105, 2015.
[7] Y. Wahyu, F. Oktafiani and Y. P. Saputera, "Development of Set Top Box (STB) for DVB-T2 standard television based on android," *2014 8th International Conference on Telecommunication Systems Services and Applications (TSSA)*, Kuta, pp. 1-4, 2014.
[8] Zhonghong Xu, Lingjun Yang and Sanxing Cao, "Design and implementation of mobile lightweight TV media system based on Android," *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, pp. 730-733, 2016.
[9] M. Milanović, B. Pavlović, I. Petrović and T. Maruna, "One implementation of UI TV application on adnroid STB," *2013 21st Telecommunications Forum Telfor (TELFOR)*, Belgrade, pp. 724-726, 2013.
[10] S. Pravin and R. BalaKrishnan, "Set top box system with android support using Embedded Linux operating systempaper," *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, Nagapattinam, Tamil Nadu, pp. 474-478, 2012.