Technical sciences

УДК 004.43

**SUTULA ALEXANDER**

student

National Technical University of Ukraine “Kiev Polytechnic Institute”

# FUNCTIONAL REACTIVE PARADIGM ADVANTAGES FOR ANDROID DEVELOPMENT

**Summary:** This article describes conceptual difference between imperative, reactive paradigms and functional reactive style advantages in Android development. Solutions of imperative paradigm main problems are described.

**Key words:** Android, functional reactive paradigm

Today mobile applications development involves using the Java as primary programming language which is based on imperative paradigm.

Imperative programming is programming paradigm, where algorithm is described as a sequence of instructions that change program state. Assigning, which is intensively used in imperative paradigm, increases the difficulty of computing models and assumes in vulnerability to errors accompanied by objects state changes that are extremely difficult to trace.[1] Also, because of the condtinuous changing states, with multithreading processing flow, locks, etc. must be used. In addition, due to probable external states changes in a function, this has to be checked, along with the usual test of the output values of the function.

Android application should provide fault tolerance at work in terms of volatility internet connection, limited device memory, etc. So, in the foreground comes errors processing, such as lost connections, server unavailability problem with converting JSON in pojo, incorrect caching. Imperative paradigm forces

creation of additional objects and error handlers. That leads to code size increase and thus creates the threat of mistakes. [2]

When requests based on the existing are created, arise problems associated with the interaction of these two requests and application structure complexity raising by adding new classes and abstractions that handle bugs.

Alternatively, it is possible to use a combination of reactive and functional programming paradigms.

Functional programming - paradigm, which considers the program as calculating mathematical functions and avoids states and variable data. Functional programming is focused on the using functions, as opposed to changes in the condition and performance of sequences of commands.

This approach offers several advantages, including:

- Code reliability improvement. The advantage computing without conditions - code safety increase by clear structuring and makes side effects tracking unnecessary. Any function works only with local data and performs same way, regardless of where it is called. Data mutation exclusion eliminates some errors that are hard to detect (such as random assignment incorrect value in the global variable in imperative program);

- Parallelism capabilities. Functional programs provide the greatest possibilities for computations automatic parallelization. No side effects are guaranteed, so parallel function call for different parameters calculation is always permissible- calculation order cannot influence call output;

- Simplicity of unit testing construction. Because the function in functional programming does not produce side effects, objects won’t be modified both inside and outside scope (as opposed to imperative programs where one function can set any external variable that will be read by another function). The only effect from calculation of the function is returned result, which can be affected only by the outcome - the arguments value. Therefore, testing every feature can be lead to calculating it for different arguments sets and

there cannot be affected by functions call order or external conditions. Function passing unit test guarantee the quality of the program. [3]

Reactive programming - programming paradigm based on the concepts of functional programming and oriented data streams and the propagation of change.

Reactive programming has foremost been proposed as a way to simplify interactive user interfaces, animations, in real time systems development. It allows entering and managing parallelism for program optimization. The paradigm is based on four principles: fault tolerance, changes sensitivity, flexibility and scalability.

Most Android application are based on permanent interaction with the user. Implementing reactive approach and functional programming in the development process increases application transparency, code readability and helps avoid errors associated with the program states and division into streams. [4]

The ability to use functional paradigm partially appeared in Java version 8, which currently is not supported by Dalvik JVM, and therefore can not be used to develop Android application. [4]

RXJava framework provides functional reactive paradigm for application development. It enables creating asynchronous systems based on processing events. Written in Scala and uses its tools. The framework provides ability develop distributed application services without the locks, synchronization or multithreading security concepts, applying reactive functional declarative style. And, key framework feature becomes classes and wrappers set for convenient Android application development, such as:

1. Sheduler, which plans and oversees the launch and implementation processes on different flows.

2. Operators to facilitate the Activity and Fragments, multiple callbacks for tracking their life cycle.

3. Wrappers for all messages and notifications, allowing them to combine all calls.

Unlike classical (imperative) developing, a Java application framework RXJava provides compliance with the four reactive programming basic principles:

1. The fault tolerance principle: all transactions results are always predictable. Potential problem areas and possible errors are known, what facilitate processing them.

2. The sensitivity principle: the database or server connection is protected from failure due to timeout, in case of problem request will be repeated several times and then will return result of caching.

3. The event orientation principle: during the request execution and processing results, the program always responds to events: inquiry successful / unsuccessful completion, handling outcome end. In addition, there is the option to subscribe several methods on one Observable and keep a consistent state of the whole system.

4. The scalability principle: adding new functionality does not require previous code changes. Convenient login errors, saving stacktrace, filter results, and so on is provided.

RXJava framework used for projects written in Java, with Scala abstraction and classes, extends Java, by adding additional features of functional languages, but also leads to code size and complexity increasing. Combining RXJava framework and Java language does not provide full functional paradigm support. For aforementioned problems avoidance usage Scala language with this framework is suggested. [4]

Scala - multiparadigm programming language combining the properties of object-oriented and functional paradigms. Scala provides all Java features and adds modern abstraction such as trait, implicit, type-checked null, block, pattern matching, monads. Since version 2.11 Scala needs at least Java 6 version. It`s

suitable for conversion into acceptable Dalvik bytecode. Therefore, pure Scala cannot be used in Android projects.

Scala provides ability to build strictly typed model (Domain Speccific Language - DSL) to adapt to each subject area and express it into language structures. Macroid is Scala DSL for developing Android applications.

Macroid DSL solved number of structural Android classical architecture interfaces problems:

- Excess files in the project – arising while each layout being stored in a separate file;
- Lack namespace arising because of layouts storage features in a single directory;
- Models duplication for different screen sizes adapting.

Macroid solves XML markup problem by describing the layout with Scala code that eliminates modularity and division into separate files. Media queries solve breakdown interfaces problem for different screen sizes, devices configurations.

Macroid also adds new abstractions such as: Tweak, media queries, Actions, Snails, which provides:

- High-level abstraction. Extraction common parts due to Tweak and standard widgets;
- Safe Context use. Owing to the AppContext and ActivityContext separation, which are received and shipped separately. AppContext storage in Macroid remains unchanged, and ActivityContext will be stored in WeakReference. This solves the problem of memory leaks.
- Threadsafety. Using Macroid UI action wrappers to work with the interface. UI action declares actions combination and method calls sequence for each stream separately.
- Creating complex animations. Animation management using abstraction Snails.

Conclusion: the basic advantages of functional reactive paradigm for Android projects, in comparison to the imperative paradigm, were reviewed. Solutions imperative paradigm main problems have been considered. Main tools for implementation functional paradigm namely: RXJava framework, Scala were investigated. Framework RXJava in Android projects can be used with Java, and with Scala. Combining Scala and RXJava solves the problem of excess abstractions and classes, provides the ability to use previously unavailable tools, such as DSL, which significantly increases the effectiveness of the program, as well as the reliability of code, eliminates some fundamental problems of Android development. This implementation allows succinctly process exceptional situations arising in unstable conditions device usage and build a scalable structure of application-oriented events. Functional reactive paradigm enable developing flexible and failsafe user interfaces for Android applications.

## References:

1. An introduction to functional programming through lambda calculus / Greg Michaelson - New York : Dover Publications, Inc., 2015. - 309 p.

2. Android Programming: The Big Nerd Ranch Guide (2nd Edition) / Bill Phillips, Chris Stewart, Brian Hardy, Kristin Marsicano - Atlanta : Big Nerd Ranch, 2015. - 600 p.

3. Couchbase Server 3.0/3.1 Developer Guide & Older SDKs [Electronic resource]. – http://docs.couchbase.com/developer/java-2.0/observables.html

4. Macroid Documentation [Electronic resource]. – http://macroid.github.io