Юсина К.М
студентка,
Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

ДОСЛІДЖЕННЯ ПРОГРАМНИХ АРХІТЕКТУР ДЛЯ ПОБУДОВИ НАВІГАЦІЙНИХ ДОДАТКІВ З ВИКОРИСТАННЯМ GPS

Android, як операційна система з відкритим кодом, досягла великих успіхів. За даними статистики дослідницької компанії Gartner, операційна система Android зайняла 80,7% частки ринку в четвертому кварталі 2015 року [1]. Наразі у Google Play доступні понад 2 мільйони додатків [2]. Більше того, в минулому році Google Play отримав більше 65 мільйонів оновлень.

Ці додатки в Google Play – це оригінальні програми, для яких потрібен пакет APK для встановлення. Окрім власних додатків, існують також веб-додатки та гібридні додатки, які базуються на веб-технологіях, наприклад. HTML5, CSS та JavaScript. Як правило, такі програми засновані на веб-браузері. Таким чином, нативний додаток (native app) є більш потужним, ніж веб-додаток, оскільки вона безпосередньо взаємодіє з апаратним забезпеченням мобільного пристрою. Дисертація зосереджена на нативних додатках.

Швидка ітерація в додатку для Android – це великий виклик для ефективності та якості розвитку, на які впливає архітектура. Стверджується, що Model-View-Presenter (MVP) і Model-ViewViewModel (MVVM) краще, ніж архітектура Model-ViewController (MVC), яка використовується за замовчуванням для Android. Але немає ніяких емпіричних даних для підтвердження цієї точки зору. Крім того, рефакторинг проекту для прийняття нової архітектури вимагає великих зусиль від розробки до тестування. Таким чином, деякі організації, які створюють додатки скептично ставляться до використання MVP і MVVM в Android.

«Архітектура програмного забезпечення або комп’ютерної системи є структурою або структурами системи, яка складається з елементів програмного забезпечення, зовнішніх видимостей цих елементів та зв’язків між ними [8]».

Хороша архітектура програмного забезпечення є одним з ключових факторів, що сприяють успіху програмного забезпечення [5]. У життєвому циклі розробки програмного забезпечення, чим раніше ми знаходимо проблему, тим менше коштує це виправити. Дизайн архітектури є ідеальною фазою для пошуку потенційних проблем. Це на ранньому етапі, де після збору вимог, але до зобов’язань щодо ресурсів [8]. Крім того, архітектура програмних систем має значний вплив на якість програмного забезпечення. Тому оцінка архітектури є критичною для програмного забезпечення.

Оцінка архітектури спрямована на те, щоб архітектура системи відповіла діловим цілям та вимогам щодо якості програмного забезпечення [8; 9]. Дослідники розробили ряд методів оцінки, які можна класифікувати за чотирма типами: сценарієм, моделюванням, досвідом та математичним моделюванням

[6; 7]. Серед цих типів метод оцінки на основі сценарію є більш точним [9], включаючи ATAM (Метод аналізу архітектури відстеження), SAAM (Метод аналізу архітектури програмного забезпечення) тощо.

На сьогоднішній день в декількох дослідженнях були розглянуті різні методи оцінки [8; 9; 10]. Patidar та Suman [9] порівнюють вісім різних сценарійських методів з різних аспектів: ціль методу, атрибути якості, діяльність, підтримка інструментів тощо. Clements et al. [8] пояснюють три методи оцінки з чотирьох перспективних причин, часу, підходів, людей та результатів.

Ці тези спрямована на те, щоб провести ретельний аналіз, щоб з'ясувати, чи архітектура MVP та MVVM є кращою, ніж MVC, з точки зору якості.

Щоб відповісти на це питання, обраний метод компромісного аналізу архітектури. Потім ми встановили три критерії: тестованість, модифікованість та продуктивність. Для кожного атрибута якості визначаються ключові фактори. Порівняння спирається на ці обрані критерії.

Мобільні додатки мають високу частоту оновлення. Дослідження, проведені на роботах сотні mobile розробників, показують, що для розробки першої версії нативного додатка потрібно близько 18 тижнів [3]. Після цього найуспішніша програма оновиться 1-4 рази на місяць, щоб йти в ногу з ринковими темпами [4]. Це великий виклик для mobile розробників оскільки вони повинні завершити розробку за короткий час і, тим самим, забезпечити якість. Ефективність та якість роботи пов'язані з особистими навичками програмістів. Проте існують деякі зовнішні методи для його вдосконалення.

Наприклад, в 2013 році компанія Google випустила розробку IDE Android Studio для заміни Eclipse. У Google I / O 2016 компанія Google анонсувала моментальний запуск нової функції, щоб прискорити процес компіляції. Сторонні бібліотеки із відкритим кодом не дозволяють розробникам повторно винаходити колесо та заощадити їм великих зусиль. Котлінська мова випустила першу версію, яка є більш лаконічною та малою. Пізніше компанія Google випустила найновіший SDK для Android N, який почав підтримувати Java 8, щоб зробити програму більш ефективною. Нова нативна архітектура додатків замінює архітектуру за замовчуванням на MVC.

Наразі офіційна архітектура Android для додатків Android – Model-View-Controller (MVC). Це класичний патерн, який успішно впроваджувався в веб-розробці. Проте в Android-розробці він зіштовхнувся з проблемами. View і Controller тісно пов'язані, що робить підтримку та розвиток важчими.

Нещодавно компанія Google випустила проект – Android Architecture Blue Print на GitHub. У цьому проекті було розроблено два основних приклади в моделі Model View Viewer (MVP) та Model-View-ViewModel (MVVM). На базі MVP, MVVM представляє нову бібліотеку для зменшення коду. До цих пір цей репозиторій отримав близько 8000 запусків.

Стверджується, що MVP / MVVM краще, ніж архітектура MVC. Але, по-перше, є невелика стаття, що пояснює архітектуру MVP та MVVM на мобільній платформі. По-друге, немає емпіричних даних та ретельного аналізу, щоб підтримати цю точку зору. Таким чином, розробники все ще сумніваються, чи варто переходити від MVC до нових архітектур. Оскільки такі рішення

потребують рефакторингу всього проекту, який вимагає великих зусиль від розробки до тестування. Крім того, для тих, хто використовував MVP, зменшуючи кодування роботи з MVVM здається привабливим. Тим не менш, вони також стурбовані тим, що це спричинить нові проблеми.

Дисертація спрямована дослідження архітектур MVP / MVVM на платформі Android. Таким чином, дослідження в даній роботі полягає в наступному:

Чи є MVP та MVVM краще, ніж MVC з точки зору якості?

Після написання дипломної роботи маємо 1) зрозуміти архітектуру MVP і MVVM на платформі Android; 2) знати про переваги та недоліки для MVP та MVVM; 3) вибрати їх переважну архітектуру на основі їх вимог до якості.

Аналіз та експерименти показують, що MVP та MVVM мають кращу можливість тестування, модифікованість (низький рівень зв'язку) та продуктивність (споживає менше пам'яті). Тобто MVP і MVVM краще MVC за вибраними трьома критеріями. Але для MVP та MVVM немає жодних доказів того, що один перевершує інший. Ці дві архітектури мають аналогічну продуктивність, тоді як MVP забезпечує кращу модифікованість і MVVM забезпечує кращу можливість тестування.

Список використаних джерел:

1. Woods V. and R. v. d. Meulen, «Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015,» Gartner, 18 February 2016. [Online]. Available: http://www.gartner.com/newsroom/id/3215217. [Accessed 28 May 2016].

2. AppBrain, «Number of Android applications,» AppBrain, 28 5 2016. [Online]. Available: http://www.appbrain.com/stats/number-of-android-apps. [Accessed 28 May 2016].

3. CrispyCodes, «How Long Does It Take To Build An IOS or Android App?,» Cripsy codes, 18 Feb 2014. [Online]. Available: http://visual.ly/how-long-doesit-take-build-ios-or-android-app. [Accessed 29 5 2016].

4. K. YARMOSH, «How Often Should You Update Your App?,» savvyapps, 12 January 2016. [Online]. Available: http://savvyapps.com/blog/how-oftenshould-you-update-your-app. [Accessed 29 May 2016].

5. Medvidovic N. and R. N. Taylor, «Software architecture: foundations, theory, and practice», in Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering, Cape Town, South Africa, ACM, 2010, pp. 471-472.

6. Chen L., M. A. Babar and B. Nuseibeh, «Characterizing Architecturally Significant Requirements,» IEEE Software, vol. 30, no. 2, pp. 38-45, 2013.

7. Bass L., Software Architecture in Practice, Pearson Education India,, 2007.

8. Clements P., R. Kazman and M. Klein, «Evaluating a Software Architecture», in Evaluating Software Architectures: Methods and Case Studies, AddisonWesley Professional, 2001, pp. 19-42.

9. A. Patidar and U. Suman, «A survey on software architecture evaluation methods,» in Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on, Indore, 2015.

10. Mattsson M., H. Grahn and F. Mårtensson, «Software architecture evaluation methods for performance, maintainability, testability, and portability,» in Second International Conference on the Quality of Software Architectures, 2006.