УДК 665.9
DOI: 10.15587/2313-8416.2015.56328

ЭКСПЕРИМЕНТАЛЬНАЯ ОЦЕНКА ЭФФЕКТИВНОСТИ РАЗРАБОТКИ БИЗНЕС-ПРИЛОЖЕНИЙ ПОД МОБИЛЬНУЮ ОПЕРАЦИОННУЮ СИСТЕМУ ANDROID

© А. В. Дмитренко

Целью исследования было на основе экспериментальной оценки метрик внутреннего качества ПО экспериментально обосновать целесообразность использования модели реализации среды облачных вычислений BaaS в разработке мобильных приложений. В ходе исследований были получено дальнейшее развитие методов выбора и обоснования архитектурных решений бизнес-приложений операционной системы Android, что позволяет снизить трудозатраты на разработку и сопровождение ПО
Ключевые слова: *android, мобильное приложение, java, gradle, сервис, модель, программное обеспечение, baas*

The aim of the study was experimentally justify the usefulness of the model implementation of cloud computing environment BaaS in the development of mobile applications on the basis of experimental evaluation of the internal quality metrics of software. Studies have received the further development of methods for the selection and validation of architectural solutions for business applications of the operating system Android, which allows to reduce labor costs for the development and maintenance of software
Keywords: *android, mobile app, Java, Gradle, service, model, software, Baas*

1. Введение

С началом информационной эпохи стала возрастать потребность общества обрабатывать всё большие объёмы информации, а также предоставлять доступ к данным в произвольный момент времени. В связи с этим в ИТ-индустрии интенсивно развивается отрасль разработки приложений для мобильных устройств. Возможность доступа к любому контенту, на любом устройстве в любой сети независимо от географического местоположения, становится реальностью. По данным исследований, в 2013 и 2014, бюджеты, выделенные на разработку мобильных приложений возросли более, чем на 40 % больше, чем в 2010 году В организациях рассматривают вопросы о расширении числа работников, использующих в бизнес-процессах смартфоны (более 70 % к 2013 году) [1]. Основными проблемами, с которыми сталкиваются разработчики приложений для мобильных устройств, являются: большое количество различных типов мобильных устройств и платформ, огромные объемы и разнообразные типы хранящихся данных, а также возрастающая угроза безопасности [2].

Для устранения этих проблем в настоящее время разработчиками ПО используется подход, обеспечивающий хранение информации с использованием сред облачных вычислений (ОВ). Идея среды ОВ была впервые выдвинута в 1961 году Джоном Маккарти, однако в связи с относительно низкой пропускной способностью каналов передачи данных на тот момент, не могла быть реализована.

Вопросами теории и практики разработки приложений для мобильных устройств исследователи занимаются в следующих направлениях: разработка приложений для мобильных устройств занимаются. Наиболее известными работами в этой области являются следующие: разработка мобильных приложений, облачные вычисления: практические подходы, практическое руководство в облачные вычисления, применение облачных вычислений в бизнес-процессах предприятия, архитектура облачных вычислений, облачные вычисления: управление и безопасность и др. и др.

Однако в настоящее время в Украине не достаточно уделяется внимания решению вопросов, связанных с внедрением моделей облачных вычислений в практические решения разработчиков бизнес-приложений для мобильных устройств, что и определяет актуальность темы исследования.

2. Анализ литературных данных и постановка проблемы

Анализ литературных данных использования облачных вычислений при разработке мобильных приложений показал следующее:

1. В отрасли информационных технологий широко используются облачные вычисления, представляющие собой комплексное решение, предоставляющее ИТ-ресурсы в виде сервиса [3].

2. К основным преимуществам облачных вычислений относятся масштабируемость, распределение ресурсов по требованию, высокий уровень надежности.

3. Концепция ОВ основана на уровнях, каждый из которых предоставляет определенную функциональность: инфраструктура услуги, система хранения, платформа, приложение, сервисы, клиенты. Поставщики этих уровней предлагают очень разные сервисы и рабочие режимы.

4. Выделяют четыре основных модели реализации сред облачных вычислений: IaaS (инфраструктура как услуга), PaaS (платформа как услуга), SaaS (программное обеспечение как услуга), mBaaS/BaaS (серверная платформа для мобильных приложений как сервис).

5. На рынке сегодня существует множество платформ для организации облачных вычислений: OpenStack, CloudFoundry, AmazonWebServices, Rackspace, WindowsAzure, Google App Engine, Force.com, VMWarevCloud, IBMCloud, QuickBlox.

6. Основные проблемы разработки мобильных приложений связаны с большим количеством раз-

личных типов мобильных устройств и платформ, большим количеством объемом и типов хранящихся данных, а также возрастающей угрозой безопасности.

7. Перед разработчиками мобильных приложений стоит проблема выбора модели реализации сред облачных вычислений для реализации приложений. Наиболее подходящим решением для разработки мобильных приложений является модель mBaaS на платформе QuickBlox.

Постановка проблемы – на основе экспериментальной оценки метрик внутреннего качества ПО экспериментально обосновать целесообразность использования модели реализации среды облачных вычислений BaaS в разработке мобильных приложений.

3. Объект, цель и задачи исследования

Объект исследования – процессы разработки бизнес-приложений мобильной операционной системы Android [4].

Цель эксперимента: обосновать целесообразность использования модели реализации среды облачных вычислений BaaS в разработке мобильных приложений с помощью метрик кода.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ проблем, связанных с использованием моделей сред реализации облачных вычислений в бизнес-приложениях для мобильных устройств.

2. Обосновать выбор модели реализации среды облачных вычислений в бизнес-приложениях для мобильных устройств.

3. Спланировать эксперимент для проверки эффективности использования модели реализации среды облачных вычислений BaaS: определить требования к прототипу, обосновать выбор инструментального средства, разработать методику проверки эффективности .

4. Разработать алгоритмы для функциональных модулей прототипов.

5. Разработать прототип бизнес-приложения для мобильного устройства ОС Android для проведения эксперимента.

6. Сделать анализ полученных результатов эксперимента и выявить негативные и позитивные стороны использования облачных вычислений в разработке бизнес-приложениях для мобильных устройств на базе ОС Android.

4. Материалы и методы и метрики исследования

Методы исследования включают в себя методы теории языков программирования, методы проектирования информационных систем и построения человеко-машинных интерфейсов для мобильных устройств, методы планирования эксперимента, методы теории вероятности и математической статистики. В ходе эксперимента были использованы такие метрики как: CBO, глубина дерева наследования, LCOM, RFC [6], NOC, WMC.

CBO позволяет определить количество классов, с которыми связан данный класс. Это означает, что один класс использует методы или экземпляры другого класса. Глубина класса в иерархии наследования есть максимальная длина от узла класса до корня дерева, измеряемая в предках класса. Чем глубже наследование класса в иерархии, тем большее количество методов он, возможно, наследует. И тем более непредсказуемым является поведение класса.

Метрика LCOM показывает, насколько методы не связаны друг с другом через свойства (переменные). Если все методы обращаются к одинаковым свойствам, то LCOM=0.

RFC позволяет определить количество методов, которое может быть выполнено в ответ на получение сообщения данным классом. В этой метрике учитываются не только выполняемые методы данного класса, но и методы других классов.

NOC позволяет определить количество непосредственных потомков класса. Значение метрики NOC идентичные для двух прототипов, только один класс имеет три потомка, чем больше потомков у класса, тем большее влияние он оказывает на систему в целом, в тестируемых прототипах количество потомков не превышается.

С помощью метрики WMC можно оценить сложность класса. Большое количество методов базового класса потенциально распространяет свое влияние и на потомков этого класса. Поскольку все потомки наследуют все методы этого класса, то усложнение методов базового класса может отразиться на потомках класса.

5. Анализ результатов экспериментальных исследований прототипов

5.1 Количество строк кода

Первая метрика, расчет которой был произведен – количество строк кода в классе.

Результаты вычисления метрик можно показаны в виде графика (рис. 1).

На графике видно, что подавляющее большинство классов в проекте являются небольшими – не более 100 строк кода, и только несколько классов содержат более 100 строк. Это позволяет сказать, что большинство классов разработаны для выполнения небольших, строго определённых задач. Крупные же классы являются ядром проекта, и содержат общую логику, согласующую работу всех его элементов.

Также видно, что количество классов в прототипе 1 больше, чем в прототипе 2 и количество срок кода тем самым больше, что дает нам уверенность сказать, что при использовании прототипа 2 код меньше и большая его часть сохраняется в библиотеке по взаимодействии приложения сё облачным сервисов BaaS.

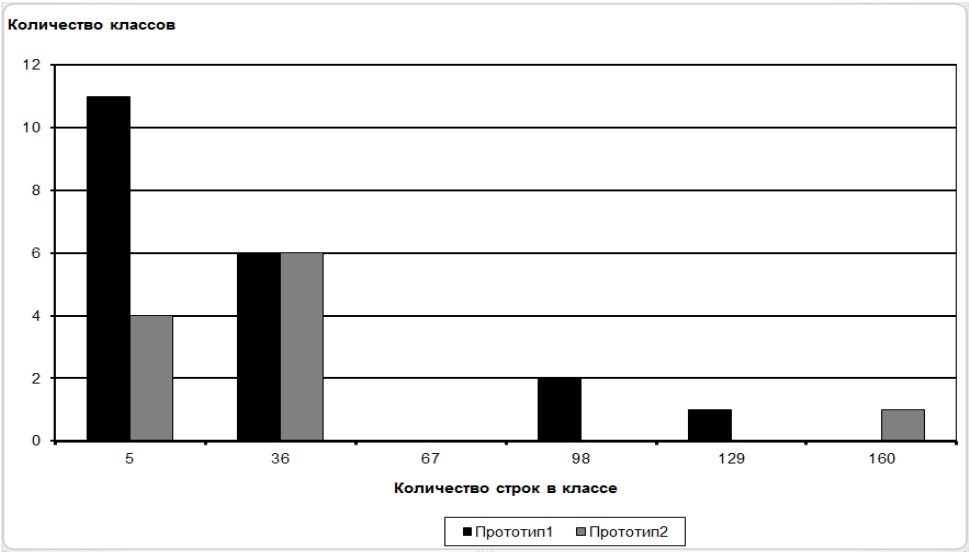

Рис. 1. График расчета количества строк кода

5. 2. Метрики С. Чидамбера и К. Кемерера

Набор метрик С. Чидамбера и К. Кемерера является основой оценки проекта реализованного на языке объектно-ориентированного программирования.

Использование наборов метрик С. Чидамбера и К. Кемерера позволяет не только определять сложность проекта, но модифицировав определенные характеристики, учесть особенности разрабатываемого проекта, провести эффективный анализ информационной системы, выявить уязвимые места.

Анализ кодов двух прототипов по набору метрик С. Чидамбера и К. Кемерера приведен в табл. 1, 2.

Таблица 1

Расчет метрик кода прототипа 1

№ п/п	Название класса	CBO	DIT	LCOM	RFC	NOC	WMC
1	com.quickblox.sample.customobjects.utils.DialogUtils	4	1	2	8	0	9
2	com.quickblox.sample.customobjects.model.Note	5	1	5	47	0	14
3	com.quickblox.sample.customobjects.helper.DataHolder	6	1	1	129	0	12
4	com.quickblox.sample.customobjects.db.tables.NoteTable.Cols	2	1	0	0	0	0
5	com.quickblox.sample.customobjects.db.tables.NoteTable	4	1	0	0	0	0
6	com.quickblox.sample.customobjects.db.tables.CommentTable.Cols	2	1	0	0	0	0
7	com.quickblox.sample.customobjects.db.tables.CommentTable	4	1	0	0	0	0
8	com.quickblox.sample.customobjects.db.DatabaseProvider	4	2	2	29	0	23
9	com.quickblox.sample.customobjects.db.DatabaseManager	6	1	1	134	0	26
10	com.quickblox.sample.customobjects.db.DatabaseHelper	5	2	1	11	0	7
11	com.quickblox.sample.customobjects.db.Co	3	1	1	3	0	1
12	com.quickblox.sample.customobjects.adapter.NoteListAdapter.ViewHolder	1	1	0	0	0	0

Таблица 2

Количество строк кода прототипа 2

№ п/п	Название класса	CBO	DIT	LCOM	RFC	NOC	WMC
1	com.quickblox.sample.customobjects.utils.DialogUtils	5	1	2	8	0	9
2	com.quickblox.sample.customobjects.model.Note	3	1	6	3407	0	12
3	com.quickblox.sample.customobjects.helper.DataHolder	5	1	3	258	0	20
4	com.quickblox.sample.customobjects.adapter.NoteListAdapter.ViewHolder	1	1	0	0	0	0
5	com.quickblox.sample.customobjects.adapter.NoteListAdapter	6	2	4	0	0	9
6	com.quickblox.sample.customobjects.activities.SplashActivity	8	5	1	471	0	11
7	com.quickblox.sample.customobjects.activities.ShowNoteActivity	17	8	1	233	0	25
8	com.quickblox.sample.customobjects.activities.DisplayNoteListActivity	8	8	3	33	0	6
	Сумма	67	46	24	7846	3	105

Разберем каждый параметр в обоих прототипах по отдельности и проанализируем результаты [6].

С гистограммы (рис. 2) видно, что в прототипе 2 больше связность классов, чем в прототипе 1. Поэтому можем сказать, что большинство классов прототипа 2 требуют намного большей связности кода для выполнения операции с базой данных, анализ ответов от сервера также делает более связанными классы и мало переносимыми – что не очень хорошо для ООП-программ. прототип 1 показал хорошие результаты связности классов.

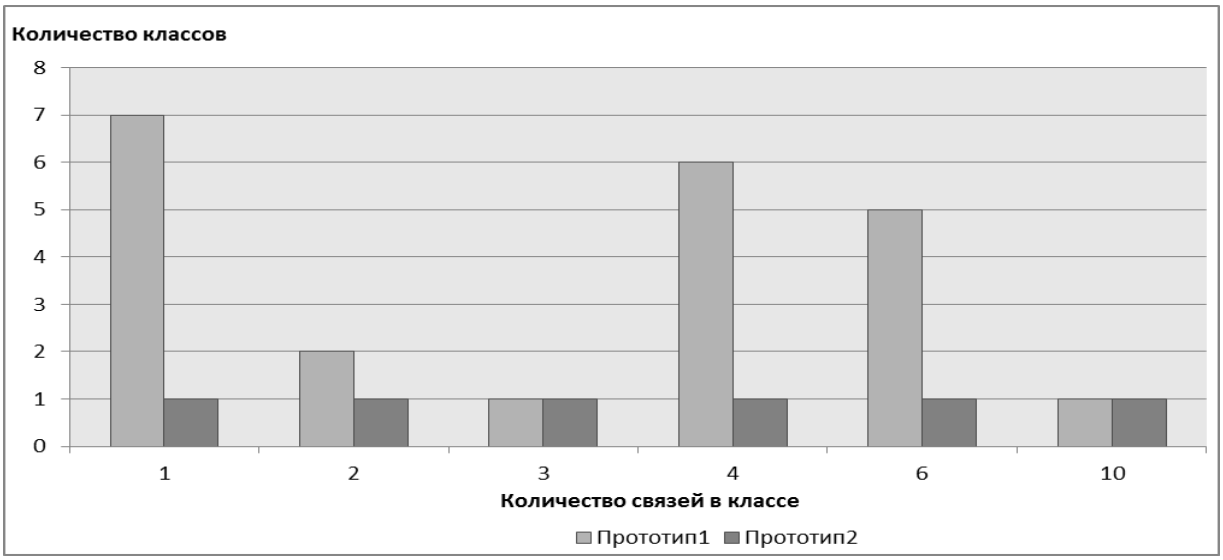

Рис. 2. Графическое представление расчетов метрики CBO

Глубина класса в иерархии наследования есть максимальная длина от узла класса до корня дерева, измеряемая в предках класса. Чем глубже наследование класса в иерархии, тем большее количество методов он, возможно, наследует. И тем более непредсказуемым является поведение класса. На рис. 3 показано графическое представление расчетов метрики DIT.

Анализируя график можно сказать, что глубина класса в иерархии наследования в двух прототипах одинакова, за исключением прототипа 1 так как у него больше количество классов.

Метрика LCOM показывает, насколько методы не связаны друг с другом через свойства (переменные). Если все методы обращаются к одинаковым свойствам, то LCOM=0. С графика (рис. 4) видно, что в прототипе 2 чуть больше связность классов, чем в прототипе 1, но совсем не значительная разница.

На рис. 5 показан результат расчетов метрики, можем сказать, что в прототипе 1 больше число методов, которые могут быть выполнены в ответ на получение сообщения классами. Поэтому чем большее число методов может быть вызвано в ответ на пришедшее сообщение, тем выше сложность класса и тем более трудоемки тестирование и отладка класса [7].

Анализируя график результатов метрики WMC (рис. 6), классы с большим числом методов с большой долей вероятности специфичны для приложения. В прототипе 1 больше методов, так как метрика связана с сопровождением программ, поскольку, чем больше число методов и выше их сложность, тем выше затраты на проектирование, кодирование, испытания и сопровождение класса.

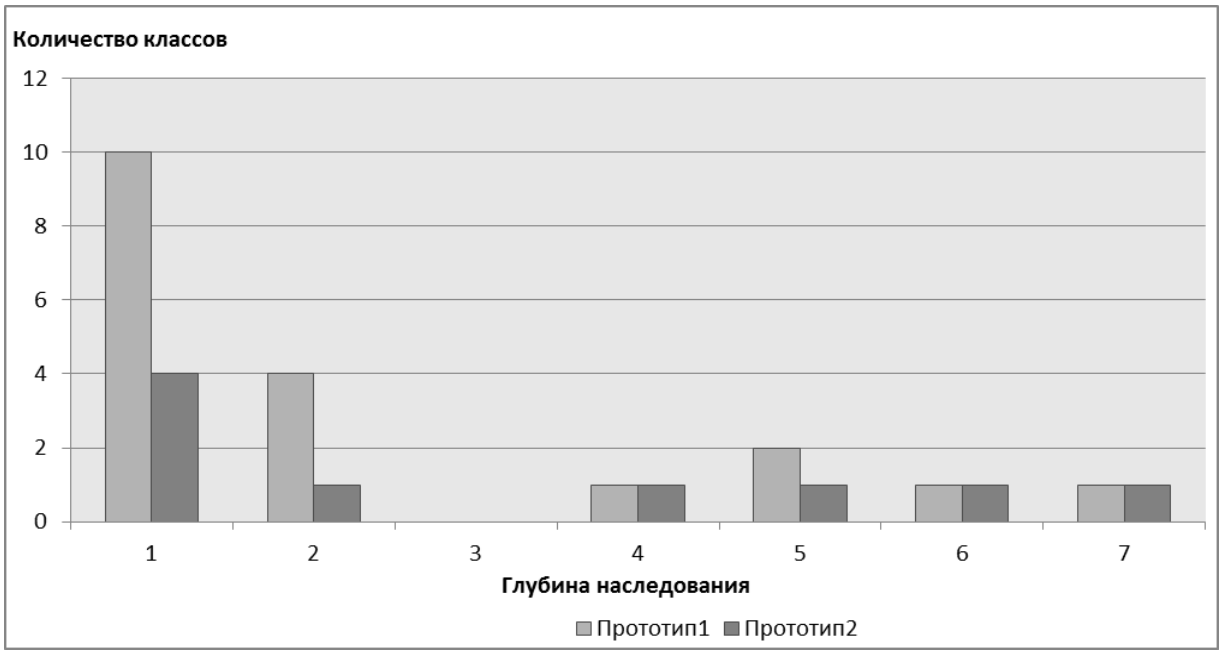

Рис. 3. Графическое представление расчетов метрики DIT

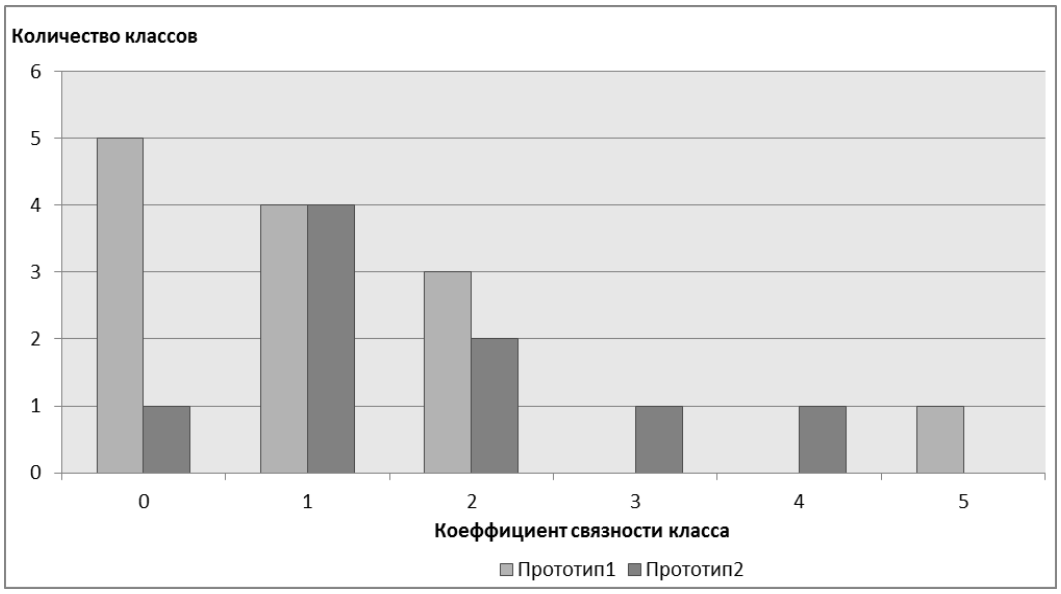

Рис. 4. Графическое представление расчетов метрики LCOM

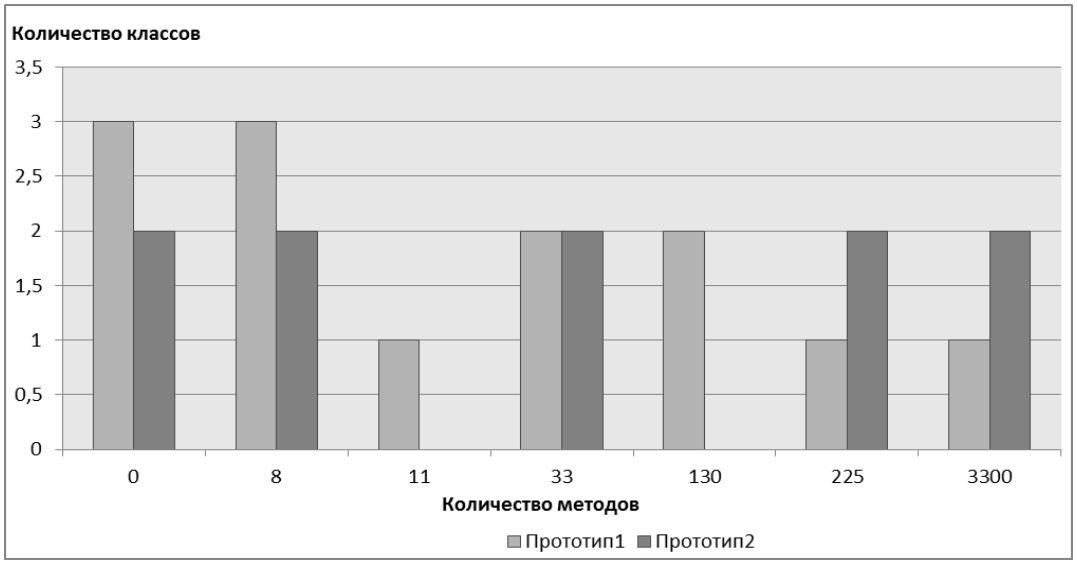

Рис. 5. Графическое представление расчетов метрики RFC

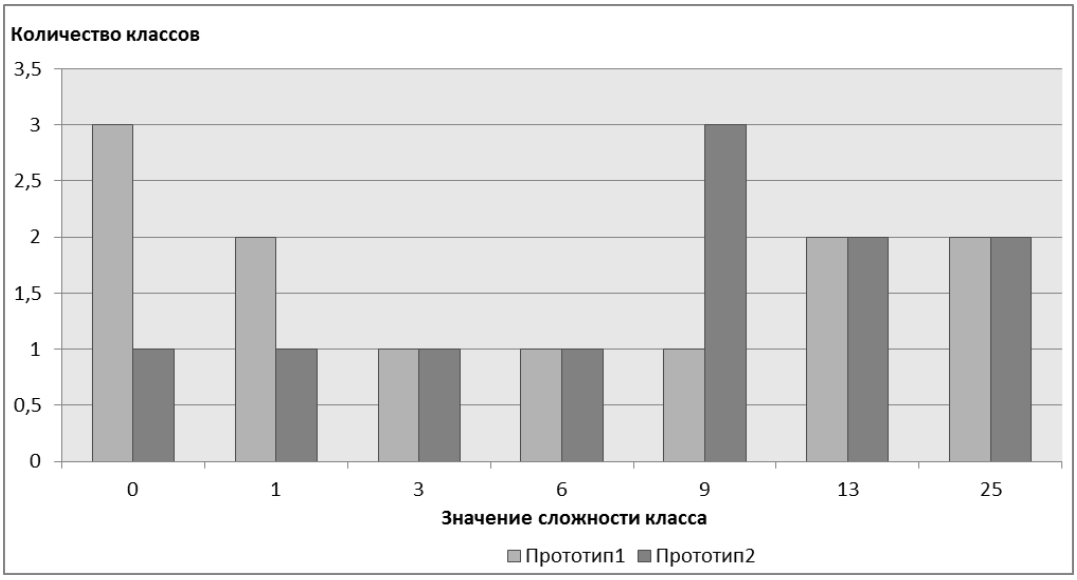

Рис. 6. Графическое представление расчетов метрики WMC

5. 3. Проверка статистических гипотез по критерию Фишера

Проанализировав исходные коды двух прототипов за метриками С. Чидамбера и К. Кемерера, просуммируем данные для каждого прототипа по каждой метрике (табл. 3).

Таблица 3

Суммарное количество значений метрик для каждого прототипа

№ п/п	Название метрики	ПРОТОТИП 1	ПРОТОТИП 2
1	CBO	95	67
2	DIT	58	46
3	LCOM	25	24
4	RFC	4056	7846
5	NOC	3	3
6	WMC	140	105

Для статистического анализа двух выборок – воспользуемся критерием Фишера, в первую очередь построим гипотезы.

H_o: Между значениями параметров качества в первой выборки [95, 58, 25, 4056, 3, 140] и второй выборки [67, 46, 24, 7846, 3, 105] нет существенных различий.

H_1: Между значениями параметров качества в первой выборки и второй выборке есть существенные различия.

Для подтверждения гипотезы вычислим эмпирическое значение критерия Фишера, в соответствии с методикой, приведенной в главе 2.

Получаем, что эмпирическое значение $F_{эмп}$=0, 00688. Определяем по таблице критическое значение. Для уровня значимости 0,05 $F_{кр}$=5, 109 , а для уровня значимости 0, 01 $F_{кр}$=9,155 [50].

Интерпретация результата: $F_{эмп}<F_{кр}$ для двух уровней значимости. Значит, H_o гипотеза отвергается и принимается H_1 – между значениями параметров качества в первой выборки и второй выборке есть существенные различия.

6. Обоснование результатов исследования

Результаты анализа позволяют сделать следующие общие выводы об сравнении двух разных архитектур разработки приложений для ОС Android. Проанализировав внутреннее качество прототипов, стало ясно, что прототип 1 имел сложную структуру, по сравнению с прототипом 2, хоть и использовал только встроенные средства разработки приложений. Но то, что он имеет возможность работать автономно – без привязки к интернету, дает ему выигрыш над прототипом 2. Прототип 2 обладает возможностью авторизации пользователей, что дает преимущества безопасности использования данных и их хранения.

В случаи удаления приложения с устройства, прототип 1 потеряет все данные, а у прототипа 2 данные будут синхронизированные с сервисом BaaS. Также нужно отметить то, что код прототипа 2 гораздо проще сопровождать, так как его объемы меньше чем прототипа 1. Это связано с тем, что большую часть тяжелой работы берет на себя сервис BaaS, в свою очередь не нагружая клиента.

В целом же следует отметить высокое качество двух прототипов. Большинство классов имеют очень небольшой размер и сложность, а также характеризуются невысокой внешней связностью. Оба прототипы хорошо показали себя на своих функциональных возможностях.

7. Выводы

В эпоху быстро развивающихся информационных технологий, растущего числа мобильных устройств и количества ПО для них, особенно остро стоит проблема выбора архитектуры будущего приложения. Данная проблема появилась не только из-за множества функционала, который предстоит реализовать в приложении, но еще из-за необходимости дальнейшего сопровождения кода, который реализует ту или иную архитектуру приложения. Существует архитектура приложений с использованием облачных вычислений, она представляет собой комплексное решение, предоставляющее ИТ-ресурсы в виде сервиса. К основным преимуществам облачных вычислений относятся масштабируемость, распределение ресурсов по требованию, высокий уровень надежности.

Приведены алгоритмы, в соответствии с которыми выполняется расчет формальных параметров исходного кода. Они направлены на оценку размера, сложности и структуры кода, что послужит основой для исследования его качества. А также были показаны алгоритмы работы прототипов с базами данных. Алгоритмы хорошо выделил различия двух различных архитектур приложений. Выяснено, что алгоритм работы с базой данных, который использует облачный сервис BaaS и имеет более сложную структуру, так как перед работой с базой данных дополнительно нужно выполнить авторизацию на сервере, но это не является избытком, в свою очередь увеличивает уровень безопасности работы с данными базы данных.

Литература

1. Marks, E. A practical guide to cloud computing [Text] / E. Marks, B. Lozano. – John Wiley & Sons, 2010.

2. Roebuck, K. Mobile Application Development [Text] / K. Roebuck. – Lightning Source Inc., 2011.

3. Rhoton, J. Application of cloud computing in the business processes of the enterprise [Text] / J. Rhoton. – Recursive Press, 2010.

4. Herbert, L. The Forrester Wave: Enterprise Mobility Services, Q1 2013 [Text] / L. Herbert // Forrester. – 2013. – Available at: http://www.forrester.com/pimages/rws /reprints/ document/87581/oid/1LT

5. Rittinghouse, J. Cloud computing architecture [Text] / J. Rittinghouse, J. Ransome. – CRC Press, 2010.

6. Velte, A. Cloud Computing: A Practical Approach [Text] / A. Velte, T. Velte, R. Elsenpeter. – McGraw Hill, 2009. – 334 p. – Available at: http://ftp.sustech.edu/cloud/ Toby_Velte,_Anthony_Velte,_Robert_Elsenpeter_Cloud_Computing,_A_Practical_Approach__2009.pdf

7. Rittinghouse, J. Cloud computing: management and safety, etc. [Text] / J. Rittinghouse, J. Ransome. – CRC Press, 2010.

References

1. Marks, E., Lozano, B. (2010). A practical guide to cloud computing. John Wiley & Sons.

2. Roebuck, K. (2011). Mobile Application Development. Lightning Source Inc.

3. Rhoton, J. (2010). Application of cloud computing in the business processes of the enterprise.Recursive Press.

4. Herbert, L. (2013). The Forrester Wave: Enterprise Mobility Services, Q1 2013. Forrester. Available at: http://www.forrester.com/pimages/rws/reprints/document/87581/oid/1LT

5. Rittinghouse, J., Ransome, J. (2010). Cloud computing architecture. CRC Press.

6. Velte, A., Velte, T., Elsenpeter, R. (2009). Cloud Computing: A Practical Approach. McGraw Hill, 334. Available at: http://ftp.sustech.edu/cloud/Toby_Velte,_Anthony_ Velte,_Robert_Elsenpeter_Cloud_Computing,_A_Practical_ Approach__2009.pdf

7. Rittinghouse, J., Ransome, J. (2010). Cloud computing: management and safety, etc. CRC Press.

Рекомендовано до публікації д-р техн. наук, професор Шабанов С. Ю.
Дата надходження рукопису 17.11.2015

Дмитренко Андрей Викторович, кафедра программной инженерии, Харьковский национальный университет радиоэлектроники, пр. Ленина, 14, г. Харьков, Украина, 61166
E-mail: grenka487@gmail.com

УДК 519.87:651.4/.9
DOI: 10.15587/2313-8416.2015.56345

МАТЕМАТИЧНА МОДЕЛЬ ПОШУКОВОГО ОБРАЗУ ДОКУМЕНТУ В СИСТЕМІ УПРАВЛІНСЬКОГО ДОКУМЕНТООБІГУ

© В. І. Кунченко-Харченко

Було проаналізовано особливості формування інформаційного потоку, логічна структура мовного представлення розширеної семантичної мережі, побудовано акцептор кінцевого автомату для системи документообігу, що сприймає нескінченний алфавіт. Модель інформаційних потоків в системі прийнята за аналогією до моделі Бартона-Кеблера та врахувує статичну і динамічну складові від загальних обсягів повідомлень в системі. В моделі враховано, що для забезпечення релевантності пошуку індексаторам рекомендується брати не більше трьох словоформ вхідного алфавіту
Ключові слова: *інформаційний потік, акцептор, пошуковий образ документа, база знань, індексування документів*

It was analyzed the spatiality of forming the information flow, logical structures of layout of the extended semantic network, built the acceptor of the terminal automat for document management system. Ones accept the infinity alphabet. The model of information flow in the system was accepted similarly to the Burton-Kebler model. Ones consists of static and dynamic parts of general massages' volume of the system. The model takes into account that to ensure the relevance of the search isn't recommended to take more than three word forms of the input alphabet for indexers
Keywords: *information flow, accepter, search document image, knowledge base, document indexing*

1. Вступ

Відомо, що основні принципи пошуку були сформульовані ще в першій половині XX століття. Як правило пошуку документів Процес створення вказівників на документи називається індексуванням, а терміни, що використовуються для індексування, називаються термінами індексування. Масив вказівників, отриманий після індексації інформаційних ресурсів називається індексом (Index database). Пошук стає більш ефективним, якщо відбувається за словником шуканих термів. Інформаційно-пошукові мови – це основна частина інформаційно-пошукової системи. Тому, ІПМ-основна частина і від неї залежить якість всієї системи. До складу ІПМ входить:

1) словник індексованих термінів є множиною термінів індексування;

2) кодовий словник – множина кодових термінів;

3) словник входів – множина вхідних термінів;

4) допоміжні засоби мови індексування – засоби, що використовуються разом з індексацій ними термінами для розширення або звуження визначених понять;

5) правила використання мови індексування.

2. Огляд літературних джерел по темі статті

У. Е. Батеном була розроблена система для пошуку патентів [1]. Запропонована ним система класифікувала документи у відповідності з поняттями, до яких він мав відношення. В даній моделі для пошуку визначеного документу, якщо в ньому розглядалося декілька понять (аналог сучасної бази знань [2]) необхідно було сумістити карти, що відповідають даним поняттям. Номер необхідного патенту визначається з позиції проміжку. З того часу основні принципи інформаційного пошуку не змінилися [3–5], пошук відбувається не по текс-