Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.414.2

Сорокина
Анна Геннадьевна

Методики выбора и оценки программной архитектуры

АВТОРЕФЕРАТ
на соискание степени магистра информатики и вычислительной техники
по специальности 1-40 81 01 – «Информатика и технологии разработки программного обеспечения»

_____

Научный руководитель
Волорова Наталья Алексеевна
Доцент, кандидат технических наук

_____

Минск 2019

ВВЕДЕНИЕ

Требования к разрабатываемому ПО увеличиваются вслед за ростом ожиданий и потребностей пользователей. Только приложения с хорошо продуманной архитектурой могут быть разработаны за приемлемое время и в рамках бюджета. К тому же каждое из разрабатываемых приложений должно отвечать определенным параметрам безопасности, надежности, производительности и поддерживаемости.

Очевидно, что невозможно реализовать все требования к надежности, производительности, безопасности и поддерживаемости программных продуктов одновременно. Поэтому каждый архитектор ПО сталкивается с проблемами выбора архитектурных решений для решения определенной технической проблемы с минимальным ущербом качеству и сложности разрабатываемого ПО.

Но каждый архитектор имеет собственный подход к сбору и анализу требований, а также к определению архитектуры. Поэтому даже в команде, состоящей из нескольких опытных архитекторов, могут возникнуть трудности при выборе окончательного архитектурного решения.

С развитием же гибких методологий разработки программного обеспечения частой становится ситуация, когда в команде нет постоянного архитектора, а иногда архитектора, а как следствие и архитектуры нет даже на начальных этапах разработки. В таких командах вся сложность принятия архитектурных решений ложится на разработчиков.

Поскольку закладывание архитектуры программного обеспечения выполняется на ранней стадии разработки, а решения в области проектирования сильно влияют на уровень качества, который может быть достигнут, то важно убедиться, что архитектура программного обеспечения поддерживает требуемые уровни качества даже если бремя разработки архитектуры легло на плечи разработчиков или начинающих архитекторов.

Поэтому для разработки качественной архитектуры необходимы методики выбора архитектурных решений и методики оценки получившегося результата. Специально разработанные методы оценки архитектуры помогают выявить неадекватные дизайн и, таким образом, снижают риск внедрения систем с недостаточным качеством. Так же такие методы дадут возможность нескольким архитекторам прийти к одинаковым и эффективным решениям при схожих технических условиях. А в дальнейшем помогут оценить соответствие разработанной архитектуры выданному техническому заданию.

Цель данной работы в разработке инструментария, способного облегчить принятие высокоуровневых и детальных архитектурных решений разработчикам и архитекторам, в особенности начинающим.

Для того чтобы достичь поставленной цели, были проанализированы существующие методы выбора и оценки программной архитектуры, а некоторые используемые в них методики были адаптированы для использования разработчиками. На основании этого была разработана программная система для помощи разработчикам в разработке архитектуры для их систем, модулей и компонентов.

Важно помнить, что стоимость, качество и время разработки - это три основные проблемы в проектах разработки программного обеспечения. Как и в большинстве инженерных дисциплин в программировании существуют средства и методы, как для прогнозирования качественных характеристик разрабатываемой системы, так и для оценки таких характеристик в уже завершенных системах. Но оценка качества продукта после его завершения представляет большой риск того, что усилия были потрачены на программный продукт с неадекватными параметрами качества. Поэтому очень важно начинать прогнозировать качество системы на самых ранних этапах ее разработки, а именно на этапе разработки архитектуры.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Цель, задачи и результаты исследования

Целью магистерской диссертации является разработка инструментария, способного облегчить принятие высокоуровневых и детальных архитектурных решений разработчикам и архитекторам.

При разработке веб-приложения использовался язык программирования Javascript и библиотеки Angular и Angular Material.

В результате разработки было создано веб-приложение, предоставляющее простой в использовании инстументарий для принятия и анализа архитектурных решений. Приложение включает в себя средства для документирования требований, визуализации архитектурных решений, а так же функциональность, помогающую в принятии решений как в условиях неопределенности (таких как выбор технологий), так и в решении сложных проблем с помощью их декомпозиции.

Во время разработки приложения так же было уделено внимание его расширяемости и поддерживаемости, чтобы иметь возможность продолжать добавления новой функциональности и расширение существующей функциональности новыми данными и методами.

Структура и объем диссертации

Полный объем диссертации в страницах – 71.
Количество иллюстраций – 23, таблиц – 1.
Количество использованных библиографических источников – 11.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Создание архитектуры программного обеспечения - это процесс разработки и спецификации правил, по которым будет создаваться программное обеспечение, и по которым компоненты системы будет вести себя и взаимодействовать. Архитектура также включает в себя создание правил для команды разработчиков. Качество разрабатываемой системы очень сильно зависит именно от этапа разработки архитектуры. От того как тщательно были собраны и проработаны функциональные и нефункциональные требования (ограничения и атрибуты качества).

К сожалению, с приходом в разработку гибких методологий количество проблем с архитектурой приложений только увеличивается. Основная проблема архитектуры для команд, работающих по гибким методологиям в том, что часто все архитектурные решения ложатся на плечи разработчиков. Но, во-первых, опыта разработчиков недостаточно для принятия правильных архитектурных решений, а во-вторых, многие разработчики опираются на принцип гибкой методологии, который гласит, что «простота — искусство минимизации лишней работы — крайне необходима», при этом забывая о другом основном принципе: «постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта». Такой подход к простоте дает плохой старт для дальнейшего развития системы.

Конечно, гибкой команде не подойдет стандартный подход к разработке архитектуры всего проекта заранее по многим причинам. Но введение в такую команду «гибкого архитектора», а так же тесное сотрудничество разработчиков и архитекторов должно привести к архитектуре, которая соответствует потребностям команды, а также требованиям заказчика. При таком подходе архитектура будет направлять команду в направлении, которое даст ей наилучшие шансы на успех, не указав слишком много, чтобы не лишить проект и команду гибкости для работы в гибкой манере. Важно найти компромисс между тем, чтобы позволять команде выбирать, что имеет смысл для их проекта, при этом, не позволяя им выбирать технологии, которые несовместимы с остальной частью приложения.

Чтобы упростить взаимодействие в таких командах, а так же чтобы помочь командам, которые по определенным причинам совсем не имеют архитектора в своем составе в данной магистерской работе было разработано приложение Architecture Analyzer.

Так как при разработке программного обеспечения самым важным критерием является его востребованность. То в рамках магистерской диссертации на этапе сбора требований дополнительно был проведен опрос

среди разработчиков. Согласно нему 40% опрошенных в дополнение к разработке занимаются еще и проработкой архитектуры для разрабатываемых ими систем. В то же время очень мало людей в достаточной мере владеет формальными языками для проектирования и моделирования архитектуры, которые широко используются профессиональными архитекторами. И только 25% опрошенных хотя бы слышали о существовании методов анализа архитектуры. Что доказывает, что разработчикам нужны менее специализированые инструменты, чем те, которые на постоянной основе используются полноценными архитекторами.

В ходе опроса так же выяснилось, что, как и предполагалось, отсутствие архитектуры является проблемой в довольно большом количестве команд особенно в компаниях меньшего размера, а так же в части библиотек и фреймворков. Но при этом многие разработчики утверждают, что они прилагают довольно много усилий, пытаясь исправить эту ситуацию. В частности они используют такие доступные им средства, как бумагу, маркерные доски и прототипирование на одном из языков программирования для проработки необходимой им архитектуры.

После обработки результатов опроса было принято решение о разработке системы, имеющей название Architecture Analyzer (Анализатор Архитектуры). Для разработки в силу их преимуществ использовались веб-технологии, а именно языки html, css и javascript и уже завоевавший популярность фреймворк Angular, а так же библиотека Angular Material. В качестве языка интерфейса приложения был выбран английский язык. Это обусловлено тем, что английский является основным языком общения между странами, а в особенности между разработчиками.

В ходе разработки требований к системе были изучены несколько видов методов, которые могут быть полезны разработчикам и архитекторам. Одним из типов методов, которые можно применить при разработке архитектуры являются методы анализа компромиссов и принятия решений. Использование методов анализа компромиссов позволяет выявлять и классифицировать возникающие проблемы, тщательно анализировать всю поступающую информацию и принимать решения, основываясь только на объективных фактах.

Один из методов анализа компромиссов, а именно метод анализа иерархий был взят за основу функциональности под названием Choice Helpers (помощники в выборе). Choice Helper представляет из себя небольшой опросник, который сравнивает предоставленные ему объекты по определенным атрибутам качества. Были детально проработаны 2 помощника в выборе, а именно помощник в выборе библиотеки и помощник в выборе технологий. Но

в силу простой расширяемости данной функциональности, количество помощников может быть с легкостью увеличено по мере необходимости.

Так же в ходе магистерской работы были рассмотрены различные методы проектирования архитектуры. Были рассмотрены методы начиная от широко известного метода дизайна на основе шаблонов, т.е. дизайна основываясь на архитектурных стилях и шаблонах проектирования, т.е. на хорошо известных и широко применимых методах решения архитектурных проблем. Заканчивая менее известными среди разработчиков методами основанными на модели множественных представлений, на оценке и трансформации (Evaluation and Transformation Based Design) и методом на основе атрибутов (Attribute-driven design method). Не смотря на их меньшую известность стоит заметить, что они в той или иной мере применяются разработчиками просто на интуитивном и зачастую бессистемном уровне.

На основе знаний, полученных из рассмотрения методов проектирования была разработана функциональность по сбору требований (Requirements Collector) в том числе и вероятных требований (Likely Changes), так как во многом именно вероятные изменения оказывают значительное влияние на итоговую архитектуру. К тому же ссновываясь на способах итеративного и рекурсивного решения проблем, такой способ, например, используется в методе проектирования, основанного на атрибутах, была предложена и реализована уникальная функциональность под названием Problem Decomposer (Декомпозитор Проблемы). Основная идея этого метода заключается именно в рекурсивном разбиении проблемы на более мелкие и итеративном их решении. В результате применения этого метода получается дерево независимых подпроблем с их решениями.

В пятой главе были рассмотрены методы анализа (оценки) архитектуры. В качестве примеров были выбраны самый первый задокументированный метод, который так и называется «Метод анализа программной архитектуры» (Software architecture analysis method – SAAM). Так же был рассмотрен метод «Семинары об атрибутах качества» (Quality Attribute Workshops - QAW), который в отличии от SAAM может быть применен на более ранних этапах проектирования, т.е. еще на этапе концептуальной архитектуры.

Для того, чтобы разработчики могли максимально опираться на объективные, а не субъективные суждения при разработке архитектуры была предложена функциональность под названием Architecture Checklists (Архитектурные Списки). Так же как и рассмотренные методы анализа архитектуры, она дает не только способ доказательства того, что архитектура удовлетворяет всем поставленным условиям, но и являются способом проверки результата для самого архитектора. Было реализовано 2 вида архитектурных списков – это проверочный список для разбиения на модули и контрольный

список для дизайна на основе требований. Но так же как и с помощниками в выборе эта функциональность легко дополняется новыми списками.

Как было замечено из проведенного опроса, разработчики часто недооценивают или используют неправильно средства для визуализации программного обеспечения, которое они в данный момент разрабатывают. Это особенно тесно связано с тем фактом, что практически все разработчики не знают UML на достаточном уровне, чтобы начать использовать его без дополнительного обучения. Что в свою очередь обуславливает то, что для коммуникации между собой они используют архитектурные диаграммы, представляющие из себя неформальные наброски состоящие из различных геометрических фигур и условных обозначений. Это содержит в себе ряд недостатков как минимум с той точки зрения, что зачастую делается с неправильным уровнем детализации и без использования специализированных инструментов, т.е. просто на листе бумаги или доске.

В шестой главе было уделено особое внимание именно вопросу визуализации и построение диаграм с различным уровнем детализации. А так же была разработана функциональность «Sketches» (Наброски).

ЗАКЛЮЧЕНИЕ

Целью данной работы была разработка инструментария для того, чтобы упростить принятие архитектурных решений для профессиональных разработчиков ПО и начинающих архитекторов.

На первом этапе работы были выявлены основные требования к разрабатываемой системе. Для выявления требований использовались такие средства как:

- анализ существующих программных систем;
- анализ литературы по разработке архитектуры как для архитекторов, так и для программистов;
- проведение опроса среди разработчиков;
- анализ математических подходов, используемых для принятия решений в ситуациях неопределенности;
- использование собственного опыта в разработке и проектировании систем;
- анализ основных методов, используемых архитекторами для выбора и оценки архитектуры.

Благодаря приведенным выше действиям были подготовлены требования к функциональности, в том числе и уникальной для данной системы. На основании этих требований была разработана программная система, содержащая в себе инструментарий для помощи разработчикам и архитекторам в принятии архитектурных и около архитектурных решений для разрабатываемых ими проектов. Разработанная программа Architecture Analyzer включает в себя средства для документирования требований, визуализации архитектурных решений, а так же функциональность, помогающую в принятии решений как в условиях неопределенности (таких как выбор технологий), так и в решении сложных проблем с помощью их декомпозиции.

Из чего можно судить, что поставленная в начале работы цель была успешно достигнута.

Главным преимуществом разработанного проекта является его простота использования, в том числе даже для неподготовленных пользователей, которые никогда до этого не сталкивались с необходимостью принятия архитектурных решений и/или со специализированными методами выбора и анализа архитектуры.

Во время разработки приложения так же было уделено внимание его расширяемости и поддерживаемости, так как очевидно, что разработанная система будет продолжать расширяться как в сторону добавления новой функциональности, так и в сторону расширения существующей

функциональности новыми данными и методами. Например, с большой вероятностью будут расширяться списки помощников в выборе (Choice Helpers) и архитектурные списки (Architecture Checklists).

СПИСОК ПУБЛИКАЦИЙ СОИСКАТЕЛЯ

1 – А. Сорокина А.Г. Проблемы архитектуры в гибких методологиях разработки ПО // Студенческий: электронный научный журнал 2018. № 23(43). URL: https://sibac.info/journal/student/43/123934.

2 – А. Сорокина А.Г. Улучшение качества оценки архитектуры ПО с помощью использования SAAM методологии // 54-я научная конференция аспирантов, магистрантов и студентов БГУИР, 2018 г – с. 204 – 205.