работать на любой другой машине Linux, независимо от каких-либо пользовательских настроек, которые могут быть у машины, которые могут отличаться от машины, используемой для написания и тестирования кода.

Docker помогает в решение унификации использования программного обеспечения, но остается необходимость мониторинга и поддержки самих контейнеров Docker, в чем помогает Kubernetes.

Kubernetes, или k8s, или ”kube"− открытое программное обеспечение для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями. Оригинальная версия разработана компанией Google. Впоследствии Kubernetes был передан под управление Cloud Native Computing Foundation. Предназначение Kubernetes − предоставить «платформу для автоматического развёртывания, масштабирования, управления приложениями на кластерах или отдельных хостах». Kubernetes группирует контейнеры, составляющие приложение, в логические блоки для упрощения управления и обнаружения. Kubernetes устраняет многие ручные процессы, связанные с развертыванием и масштабированием контейнерных приложений. Другими словами, Kubernetes позволяет объединять группы хостов, работающих под управлением контейнеров Linux, и легко и эффективно управлять этими кластерами. Такие кластеры могут охватывать узлы в общедоступных, частных или гибридных облаках.

Kubernetes поддерживает различные технологии контейнеризации, включая Docker, VMWare и ряд других. Kubernetes является наиболее популярным решением по сравнению с Docker Swarm и Nomad ввиду своего функционала и расширяемости, несмотря на сложности и нужный уровень технической подготовки для работы с ним. Целью исследования является построение масштабируемого отказоустойчивого кластера контейнеров Docker, поэтому использоваться будет Kubernetes.

В работе Apache Solr и Apache Hadoop (в частности, используемая его часть HDFS) помещены в Docker контейнеры для унификации и упрощения работы с ними. Docker контейнеры сгруппированы в Kubernetes кластер для упрощения управления и мониторинга. Таким образом, решение представляет собой конфигурируемый кластер полнотекстового поиска, удобный для обучения и разработки.

Список использованных источников:

1. Хайлоад. Полнотекстовый Поиск. [Электронный ресурс]. – Режим доступа:https://ruhighload.com/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9+%D0%BF%D0%BE%D0%B8%D1%81%D0%BA - Дата доступа: 24.03.2018.

2. Apache Solr. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Apache_Solr. – Дата доступа: 25.03.2017.

3. Docker Cookbook – O'Reilly Media, 2015. [Электронный ресурс]. – Режим доступа: http://shop.oreilly.com/product/0636920036791.do – Дата доступа: 25.03.2017.

4. Scaling Big Data with Hadoop and Solr - Second Edition – Packt, 2015. [Электронный ресурс]. – Режим доступа: http://shop.oreilly.com/product/0636920036791.do – Дата доступа: 25.03.2017.

УЛУЧШЕНИЕ КАЧЕСТВА ОЦЕНКИ АРХИТЕКТУРЫ ПО С ПОМОЩЬЮ ИСПОЛЬЗОВАНИЯ SAAM МЕТОДОЛОГИИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Сорокина А.Г.

Волорова Н.А. – к.т.н., доцент

В настоящее время архитектура программного обеспечения является объектом повышенного внимания, как в академических кругах, так и в промышленной разработке. Но, несмотря на популярность этой темы, достаточно мало внимания уделяется методам анализа разработанной архитектуры, которые могут показать, что она удовлетворяет определенным свойствам.

На большинстве крупных проектов особенно из сферы финансов, обороны и медицины проводят рецензирование разработанного дизайна, которое является важным компонентом процесса создания программного обеспечения. Тем не менее, такие проверки часто проводятся не постоянно и многие участники процесса считают их напрасной тратой времени, так как они не выявляют всех проблем и дефектов, которые впоследствии выявятся только в процессе разработки, что уже слишком поздно.

Эта проблема появляется вследствие того, что в большинстве случаев сложно объективно оценить заявления разработчика о качествах, присущих созданной программной архитектуре. Примерами таких заявлений являются фразы на подобие [1]:

- Мы разработали компоненты, которые можно перенастроить с минимальными усилиями.
- Этот метод поощряет такое разделение приложения на части, которое уменьшит стоимостьпоследующих модификаций системы.
- Этот подход лучше, чем традиционные подходы с точки зрения модульности и повторного использования кода.

Трудность в оценки этих требований возникает по двум причинам. Во-первых, различные архитектурные решения не используют общий словарь. А когда разрабатываются новые архитектуры, то они обычно разрабатывают новые термины для описания или используют старые термины по-новому. Во-вторых,

часто сложно связать архитектурные абстракции с проблемами развития системы. Разработчики, как правило, концентрируются на функциональных особенностях своих архитектур и редко решают, как их архитектуры решают проблемы в рамках развития определенной системы.

Как одно из решений описанных выше проблем в рамках этой работы рассмотрим процесс рецензирования программного обеспечения, основанный на методе анализа архитектуры ПО (Softwarearchitectureanalysismethod - SAAM). SAAM является первым задокументированным методом анализа архитектуры приложений и служит основой для многих более современных методов анализа архитектуры. Например, он является предшественником для метода анализа альтернатив (Architecturetradeoffanalysismethod - ATAM).

Процесс анализа архитектуры SAAM представляет собой метод, который призван помочь в оценке и понимании архитектуры программного обеспечения и в решении проблем качества, таких как переносимость, сопровождаемость, расширяемость, модульность, повторное использование и т.д. Согласно результатам исследований, использование специализированных методов анализа программной архитектуры, а в частности SAAM, дает потенциальные преимущества по сравнению с традиционным процессом рассмотрения проектов при выявлении и уточнении требований.

Основной целью методологии SAAM является выяснение того как были достигнуты конкретные атрибуты качества приложения и как возможные изменения в будущем повлияют на эти атрибуты. Поэтому главным этапом SAAM процесса является идентификация сценариев, которые показывают то, как система может использоваться или модифицироваться. Для оценки и сравнения сценариев рассчитывается их стоимость путем подсчета количества необходимых изменений. К тому же в процессе оценке возникает понимание того на какие компоненты влияет каждый сценарий, что позволяет выявить компоненты, которые учувствуют в слишком большом количестве сценариев и как следствие являются показателем плохого разделения ответственностей в архитектуре.

Процесс анализа состоит из 6 активностей [2]:

1) Разработка сценариев;
2) Описание архитектуры программного обеспечения;
3) Классификация сценариев и задание им приоритетов;
4) Оценкаиндивидуальныхсценариев;
5) Взаимодействие сценариев;
6) Общая оценка;

Часто дополнительно выделяют этапы подготовки и беглого обзора архитектуры, на которых подробно изучается вся предоставленная документация. При наличии нескольких вариантов архитектур добавляется этап сравнения этих архитектур, используя выбранные сценарии.

Как можно заметить из вышеперечисленного, SAAM предоставляет нам много преимуществ. А именно:

- Дает возможность обнаружения проблем на ранних стадиях;
- Помогает улучшить документацию и наше понимание системы и путей ее развития;
- Привлекает к процессу оценки архитекторы дополнительных заинтересованных людей, таких как акционеры, архитекторы, команды сопровождения и разработки;
- Предоставляет методы и рекомендации по повышению качества и классификации различных сценариев;
- Позволяет проводить объективное сравнение нескольких архитектур благодаря унифицированному словарю и системе оценок и с учетом потребностей конкретной системы.

Список использованных источников:

1. Rick Kazman, Len Bass, Gregory Abowd, Mike Webb, SAAM: A Method for Analyzing the Properties of Software Architectures

2. Ali Athar, A Comparative Analysis of Software Architecture Evaluation Methods

ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ ДЕЙСТВИТЕЛЬНЫХ И КОМПЛЕКСНЫХ КОРНЕЙ ТРИГОНОМЕТРИЧЕСКИХ И ГИПЕРБОЛИЧЕСКИХ УРАВНЕНИЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Таланец А.В.
Стройникова Е. Д. – ассистент кафедры информатики

Многие процессы окружающей среды описываются с помощью уравнений. Уравнения широко применяются в различных областях науки и техники. Решение уравнений – составная часть описания процессов, позволяющая изучить их. Одним из способов решения уравнений является графический способ. Графическое представление корней уравнения даёт полное представление о количестве корней уравнения, позволяет лучше понять взаимосвязь коэффициентов уравнения и расположения корней, даёт глубокое понимание связи между уравнениями и функциями. В данном докладе приведён пример графического представления корней тригонометрических и гиперболических функций.