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

УДК 004.414.2

Дерех
Александр Сергеевич

Программное средство для автоматизации continuous integration

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

Научный руководитель
Самаль Дмитрий Иванович
кандидат технических наук, доцент

МИНСК 2019

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Целями диссертационной работы являются:

- исследование существующих подходов к построению процессов в IT компаниях;
- исследование существующих систем управления проектами в IT компаниях;
- исследование существующих систем continuous integration и continuous delivery;
- разработка программного средства, позволяющего вывести понятие continuous integration на новый уровень путем автоматизации большого количества процессов, используемых в современной разработке программного обеспечения.

Для достижения поставленных целей необходимо решить следующие задачи:

- проанализировать существующие подходы к построению процессов в IT компаниях;
- проанализировать существующие системы управления проектами в IT компаниях;
- проанализировать существующие системы continuous integration и continuous delivery;
- разработать программное средство, позволяющее вывести понятие continuous integration на новый уровень.

Объектом исследования являются подходы к построению continuous integration и continuous delivery процессов в тесной интеграции с процессами управления проектами в IT компаниях.

Предметом исследования являются процессы управления проектами в IT компаниях и процессы continuous integration и continuous delivery.

Практическая значимость работы заключается в том, что разработанная система позволяет разработчикам, тестировщикам и менеджерам быстрее и эффективнее выполнять свои обязанности за счет автоматизации большинства процессов. Использование автоматизации процессов приведет к сокращению времени доставки нового функционала.

ВВЕДЕНИЕ

В большинстве своем, все процессы разработки служат, по сути, одной простой цели – сократить время между рождением идеи и ее доставки в "боевое" окружение при какой-то заданной, в общем случае константной, величине качества кода. А плохая это идея или хорошая совершенно не важно. Плохие идеи обычно ускоряют, дабы проверить и затем отказаться и, возможно, по-быстрому дезинтегрировать. Стоит отметить, что процесс отката на предыдущую версию тоже ложится на плечи робота, которым автоматизированы ваши процессы. Непрерывная интеграция и доставка обычно кажутся спасательным кругом в мире разработки. Казалось бы, ничего сложного: есть идея, есть код – все очень просто. Все бы хорошо, если бы не одно большое "НО". Процесс интеграции и доставки довольно сложно формализовать в отрыве от используемых технологий и бизнес-процессов, протекающих в компании.

Одними из наиболее важных компонентов в разработке программного обеспечения являются сборка и развертывание продукта. Сборка – процесс преобразования исходного кода проекта в упакованный запускаемый файл. Примером сборки может служить упаковывание Java-приложения в jar-файл для запуска на Java Virtual Machine (JVM). Развертыванием продукта является процесс переноса изменений в приложении с локальной среды разработчиков в среду эксплуатации (по-другому данная среда описывается термином production). Обычно в процессе развертывание происходит доставка собранного проекта в специальный репозиторий и дальнейшее скачивание проекта вместе с его установкой и конфигурацией.

На данный момент в разработке приложений существует четкая тенденция к ускорению цикла разработки: кодирование, тестирование, сборка, развертывание. Конечным результатом цикла является новый релиз программного продукта. Очень часто команде разработчиков приходится доставлять новые релизы по несколько раз день. В свою очередь, все процессы, описанные в цикле, занимают достаточно много времени и являются рутинной работой. Дополнительно можно упомянуть человеческий фактор, когда небольшое отклонение от процедуры может привести к ошибкам в новом релизе, которые впоследствии очень сложно отследить.

Методология "Continuous Integration" имеет конечную цель в виде построения максимально быстрого, повторяемого и надежного процесса внедрения изменений в программный продукт. В рамках настоящей диссертационной работы вместо англоязычного термина "Continuous Integration" будет также использоваться русскоязычный синоним «непрерывная интеграция», а вместо англоязычного термина "Continuous Delivery" –

русскоязычный термин «непрерывная доставка». В качестве рабочих микропроцессов рассматриваются изменение кода, тестирование, сборка, регистрация пакета и развертывание. Изменение программного кода регистрируется системой контроля версий и отправляется для проверки в интеграционное тестирование. Тестированием является процесс контроля качества продукта и его целостности. После процесса сборки, описанного в начале статьи, продукт проходит процедуру регистрации в виде отдельного атомарного релиза. Процесс непрерывной интеграции продукта завершается развертыванием на эксплуатируемой платформе (обычно это тестовый сервер).

Внедрение методологии "Continuous Integration" несет за собой несколько преимуществ:

1. Ускорение процесса разработки.
2. Надежность успешности внесения изменений.
3. Сокращение рисков развертывания кода с проблемами.
4. Ускорение всех процессов доставки нового функционала путем автоматизации.
5. Возможность быстро и четко определить возникшие ошибки и искоренить источники этих ошибок.
6. Сокращение стоимости исправления дефекта, за счет раннего его выявления.
7. Постоянное наличие текущей стабильной версии вместе с продуктами сборок – для тестирования, демонстрации, и т. п.
8. Немедленный эффект от неполного или неработающего кода приучает разработчиков к работе в итеративном режиме с более коротким циклом.

Было принято решение разработать и внедрить на предприятии систему для автоматизации связей между всеми процессами доставки нового функционала приложения от идеи до производственной среды (production) со следующими свойствами и возможностями:

- автоматическое изменение статусов задач в системе управления проектом;
- автоматический подбор разработчиков во время этапа code review;
- автоматический запуск тестов и code coverage;
- автоматическая сборка приложения нужной версии, включающей изменения, внесенные разработчиком;
- масштабирование системы и высокая пропускная способность;
- модульная архитектура и микросервисный подход в архитектуре;
- интеграция с мессенджером Slack и автоматическое оповещение заинтересованных лиц;
- автоматическая доставка внесенных изменений при соблюдении всех необходимых правил.

СОДЕРЖАНИЕ РАБОТЫ

Диссертация состоит из введения, общей характеристики работы, четырех глав, заключения, списка использованных источников.

В первой и второй главах представлен анализ предметной области – обзор существующих систем непрерывной интеграции и непрерывной доставки, а также систем управления проектами. Рассмотрены положительные и отрицательные стороны существующих систем. Выявлены основные существующие проблемы в рамках тематики исследования, показаны направления их решения.

Третья глава посвящена обзору существующих процессов и моделей разработки программного обеспечения. Целью данной главы было определить пути и возможности для автоматизации данных процессов.

Четвертая глава посвящена разработке и внедрению собственной системы ускорения процессов разработки программного обеспечения. Подробно рассмотрены архитектура системы и аспекты выбора микросервисного подхода.

Общий объем работы составляет 67 страниц, из которых основного текста – 46 страниц, 13 рисунков на 13 страницах, 4 таблицы на 5 страницах и список использованных источников из 24 наименований на 2 страницах.

ЗАКЛЮЧЕНИЕ

В нынешних реалиях существует много различных систем для разных задач: системы управления проектом, системы хранения кода, обеспечивающие проведение code review, системы continuous integration, системы общения всей команды проекта. Однако, не существует системы, которая бы обеспечивала полный цикл разработки от идеи, постановки задачи, до появления кода в production. Другими словами, необходима система аггрегатор всех разнородных систем, без которых нельзя обойтись при разработке программного обеспечения.

В ходе исследования были изучены основные системы continuous integration, системы управления проектами и подходы к построению процессов разработки программного обеспечения. Исследование показало, что все существующие системы обладают огромными возможностями, но каждая в своей конкретной сфере даже несмотря на то, что в этих системах существует большое количество интеграций с другими. Поэтому, необходимо дополнительное время на ручное управление процессами разработки, например: назначение подходящего разработчика для проведения code review в системе хранения кода, перевод задач из статуса Code Review в статус To Test в системе управления проектом, запуск необходимой сборки для тестирования руками в системе continuous integration. Все процессы, которые не автоматизированы приходится необходимо делать человеку, а это очень существенно замедляет весь процесс разработки и тормозит каждого участника процесса, например разработчик не может приступить к следующей задаче сразу, а вынужден руками запустить сборку тестов, подобрать нужных кандидатов для процесса code review и т.п. Такие вещи зачастую отнимают больше времени, чем может казаться.

В рамках магистерской работы была реализована система автоматизации continuous integration. С одной стороны, понятие continuous integration предполагает некую автоматизацию и сказать «автоматизация continuous integration» будет несколько неправильно, однако это можно сделать, учитывая какие результаты по автоматизации были достигнуты. Данная система позволяет программисту написать код для необходимой задачи и, если он всё сделал правильно, ему не нужно ни о чем заботиться и приступать к следующей задаче, а его код автоматически попадет в центральную ветку и, что самое главное, задача будет проведена по все процессам «самостоятельно».

СПИСОК ПУБЛИКАЦИЙ СОИСКАТЕЛЯ

1 – А. Дерех А.С. Почему необходимо использовать continuous integration в процессе разработки программного обеспечения // Студенческий: электронный научный журнал 2018. № 23(43). URL: https://sibac.info/journal/student/45/128480.