УДК 004

Козлов Илья Николаевич, магистрант,
Новосибирский государственный технический университет,
г. Новосибирск

Томилов Иван Николаевич, кандидат технических наук, доцент,
Новосибирский государственный технический университет,
г. Новосибирск

ОБЗОР ИНСТРУМЕНТОВ НЕПРЕРЫВНОГО ИНТЕГРИРОВАНИЯ И РАЗВЁРТЫВАНИЯ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аннотация. В статье рассматривается концепция непрерывной интеграции и непрерывной доставки (CI/CD) в разработке программного обеспечения (ПО). Эта концепция основана на использовании непрерывного тестирования и доставки кода для повышения скорости и качества разработки. CI/CD представляет собой методологию, которая объединяет разработку и эксплуатацию ПО, соответствуя принципам Agile и воплощает стратегии и компоненты DevOps.

Ключевые слова: GitHub, GitLab, CI, CD, инструменты интегрирования, инструменты развёртывания, сборка приложений, Jenkins.

За последнее десятилетие IT-рынок свидетельствовал о появлении и применении множества методик разработки программного обеспечения. Современные приложения создаются с использованием различных платформ и инструментов, что создает потребность в механизмах интеграции и тестирования внесенных изменений. Исходным пунктом для этого процесса являлись каскадные модели, также известные как waterfallmodels. Однако со временем каскадная модель, предназначенная для сложных и многоэтапных процессов, быстро утратила свою актуальность на рынке. В рамках этой модели

каждый разработчик отвечает только за свою часть кода, а интеграция происходит только на финальной стадии разработки. Главным недостатком данной модели является то, что обнаружение ошибок приводит к задержкам в завершении всего процесса разработки.

Если над крупным проектом работают команды программистов и тестировщиков, то вносимые изменения в код происходят не один или два раза в день, и каскадная модель уже не подходит для эффективного функционирования. Например, в сфере веб-разработки изменения в коде должны быть внесены часто и быстро, и поэтому данная модель оказывается крайне неэффективной и не пригодной для таких случаев.

Вместо каскадной модели появились более гибкие методологии, такие как Agile, которая упрощает работу менеджеров и помогает ускорить выпуск будущих проектов. Для обеспечения непрерывной поставки приложений конечным пользователям были разработаны инструменты, такие как Jenkins, TeamCity и другие. Они обеспечивают возможность непрерывной доставки кода и непрерывного тестирования, что приводит к повышению скорости и качества разрабатываемых продуктов [3].

В 1991 году американский программист Гради Буч предложил набор инструментов для разработчиков под названием «Непрерывная интеграция/непрерывное развертывание» или сокращенно CI/CD. Это нововведение помогает крупным командам разработчиков выпускать изменения несколько раз в день и не ограничивает количество необходимых версий, которые могут быть выпущены.

Каскадная модель сборки приложений представляет собой процесс разработки программного обеспечения, который характеризуется последовательным выполнением задач, включающих определение требований, проектирование, конструирование, воплощение, тестирование и отладку, инсталляцию и поддержку. Эта модель была предложена Уинстоном Уокером Ройсом в 1970 году.

В каскадной модели преобладает линейный порядок выполнения задач, где каждая фаза разработки последовательно переходит к следующей. Однако основным недостатком данной модели является необходимость успешного завершения предыдущей фазы для перехода к следующей. Это ограничивает гибкость модели и делает ее критикуемой, поскольку управление проектом становится приоритетом, в ущерб другим аспектам, таким как стоимость, качество и сроки выполнения. Однако в случае крупных проектов это может также считаться преимуществом.

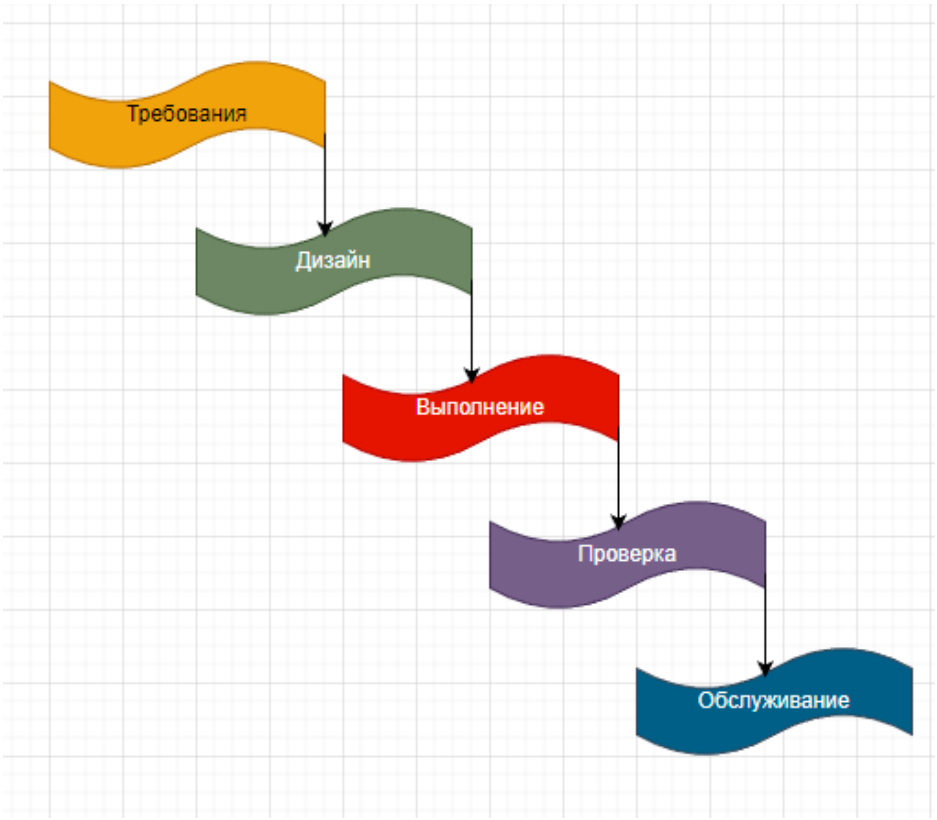

Рисунок 1. Пример последовательного перехода
от одной фазы к другой в каскадной модели.

Таким образом, каскадная модель сборки приложений представляет собой структурированный подход к разработке программного обеспечения, который хотя и имеет некоторые ограничения, может быть эффективно использован в крупных проектах.

Гибкая методология разработки, известная также как Agile software development, представляет собой совокупность подходов и практик, основанных на "Манифесте гибкой разработки программного обеспечения". Agile является эффективной практикой для организации работы творческих групп, работающих над одним большим проектом. [6].

Agile – это семейство процессов разработки ПО, которое определяет основные ценности и принципы, руководствуясь которыми команды разрабатывают программное обеспечение. Основные принципы, изложенные в «Манифесте гибкой разработки программного обеспечения», включают удовлетворение потребностей заказчика через постоянные поставки ПО, гибкость в изменении требований для повышения конкурентоспособности, частые обновления ПО, постоянное взаимодействие между заказчиками и разработчиками, ориентацию на интересы заинтересованных сторон и т.д. [6].

В Agile разработке продукт создается постепенно, серийно, каждая последующая итерация и версия продукта имеет больше функциональных возможностей, чем предыдущая.

Agile обладает преимуществами быстроты, фокуса и адаптивности. Периодические обновления и постепенное улучшение функциональности сокращают время разработки и достижение конечного результата.

Однако Agile также имеет недостатки. Снижение значимости документации, краткосрочное планирование и постоянные изменения требований могут приводить к ошибкам в создании архитектуры проекта и накоплению дефектов, что в конечном итоге снижает качество продукта.

Рисунок 2. Пример использования гибкой разработки программного обеспечения

Основная идея CI/CD заключается в использовании инструментов, которые позволяют автоматизировать процессы разработки, тестирования, релиза и мониторинга ПО. При разработке приложений в рамках CI/CD применяются различные типы тестирования, такие как интеграционное тестирование, статический анализ кода, юнит-тестирование, ручное и автоматическое тестирование [3][4].

CI/CD обеспечивает автоматизированную сборку и поставку приложений, что позволяет разработчикам сосредоточиться на реализации бизнес-требований, устранении уязвимостей и написании качественного кода. Непрерывная интеграция в рамках CI/CD обеспечивает последовательный и автоматизированный процесс сборки и тестирования приложений после каждого изменения кода. Непрерывная доставка автоматизирует развертывание приложений и поставку артефактов в различные окружения.

Инструменты CI/CD также позволяют настраивать специфические окружения, обращаться к различным серверам и базам данных для выполнения дополнительных действий при интеграции приложения. В целом, использование CI/CD способствует более быстрой и надежной разработке программного обеспечения, улучшению сотрудничества между разработчиками и эксплуатацией, а также повышению качества и удовлетворенности пользователей [4].

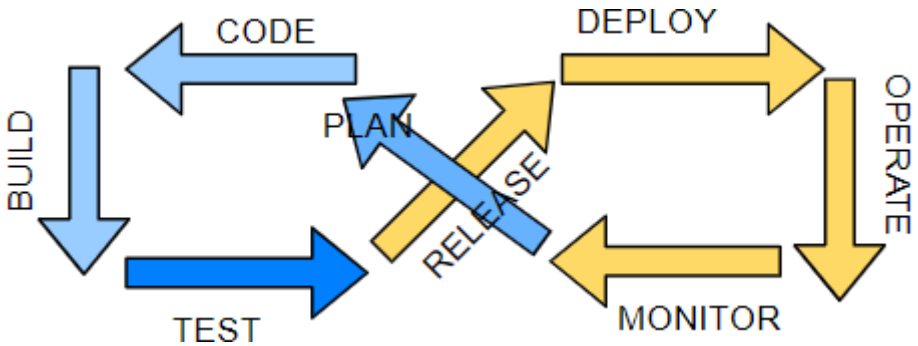

Рисунок 3. Концепция непрерывной интеграции и доставки

Первый этап – планирование. На данной стадии строится архитектура будущего приложения.

Второй этап – написания кода приложения.

Третий этап – сборка приложения из написанного кода.

Четвертый этап – модульное или ручное тестирование. Если этот этап не завершается успехом, то весь цикл прерывается.

Пятый этап – релиз приложения. На данном этапе подготавливается стабильная версия приложения, которая прошла этап сборки и тестирования успешно.

Шестой этап – поставка нашего приложения на сервер.

Седьмой этап – рабочее приложение. Данный этап подразумевает под собой полностью функционирующее приложение.

Восьмой этап – это мониторинг нашего приложения. На данном этапе собирается статистика пользователей и функциональности системы.

Концепция непрерывной интеграции и непрерывной доставки (CI/CD) применяется в качестве конвейера для облегчения непрерывной сборки и выпуска артефактов. Она позволяет проводить различные тесты на каждом этапе разработки и выполнять их запуск с развертыванием кода в тестовое и релизное окружения, которые будут использованы конечными пользователями [3].

Эта методология успешно применяется в различных типах проектов, особенно в новых разработках программного обеспечения, основанных на микросервисной архитектуре.

Однако переход на новую методологию требует адаптации. Необходимо установить новые процессы и роли сотрудников, а также найти точки интеграции существующих и будущих процессов. Также стоит отметить, что ответственность разработчика за продукт возрастает. Теперь разработчик не только пишет код по заданной технической спецификации, но и проходит этап предварительной сборки, что повышает требования к его навыкам.

Одним из преимуществ данной методологии является большой выбор открытого программного обеспечения и инструментов. Возможность выбора инструментов, соответствующих требованиям клиентов и особенностям

проекта, представляет собой несложную задачу. На рынке доступно множество инструментов для непрерывной интеграции, каждый из которых имеет свои особенности и функции.

Ниже будет представлена таблица с обзором инструментов непрерывной интеграции и развёртывания.

Таблица 1

Обзор инструментов непрерывной интеграции и развёртывания

Критерий	Jenkins	GitLab CI	BitBucket Pipelines	TeamCity	Circle CI
Тип инструмента	CI/CD сервер	CI/CD сервер	CI/CD сервер	CI/CD сервер	CI/CD сервер
Лицензия	Бесплатная	Бесплатная	Платная	Платная	Бесплатная/платная
Поддержка языков	Широкий спектр	Широкий спектр	Ограниченный выбор	Широкий спектр	Широкий спектр
Интеграция	Многочисленные интеграции доступны	Интегрирован с GitLab	Интегрирован с BitBucket, Jira	Многочисленные интеграции доступны	Многочисленные интеграции доступны
Удобство	Имеет множество настроек и гибких возможностей	Интуитивный интерфейс и легкая установка	Простая конфигурация и управление	Интуитивный интерфейс и легкая установка	Интуитивный интерфейс и легкая установка
Комьюнити	Большое	Большое	Ограниченное	Большое	Большое
Дополнительные возможности	Множество плагинов и расширений	Встроенный контейнерный реестр	Ограниченное количество возможностей	Широкий выбор плагинов и расширений	Множество плагинов и расширений
Поддержка Docker	Да	Да	Да	Да	Да

Распределенная сборка	Да	Да	Да	Да	Да
Настраиваемый пайплайн	Да	Да	Да	Да	Да
Веб-интерфейс	Да	Да	Да	Да	Да
Отчеты и уведомления	Да	Да	Да	Да	Да
Интеграция с инструментами	Доступны многочисленный интеграции	Интегрирован с GitLab, Jira, Docker, Kubernetes и другие	Интегрирован с BitBucket, Jira, Docker и другие	Доступны многочисленный интеграции	Интегрирован с GitHub, Jira, Docker, Kubernetes, Slack и другие

Для маленькой команды разработчиков (до 10 человек) оптимальный вариант набора инструментов с учетом трекера задач и хранилищем кода будет следующий:

1) Jenkins – открытое программное обеспечение, базирующееся на JavaVirtualMachine имеет огромное количество плагинов, которые помогают в сборке, автоматизации и развертывании любых проектов, дружественный интерфейс, легко устанавливается в любую операционную систему, позволяет писать скрипты на groovy с использованием java зависимостей [2].

2) Bitbucket – надежное хранилище для кода, интегрирован с Jira. Он обладает простой конфигурацией и управлением проектами [6].

3) Slack – популярный мессенжер для командного общения. Имеется возможности интегрирования с Jira, Butbucket, Jenkins.

4) Jira – мощный инструмент для управления задачами и проектами. Он интегрируется с Bitbucket и имеет расширенные возможности отслеживания задач и управления процессами разработки [6].

В данной статье были рассмотрены модели программной разработки и рассмотрены инструменты непрерывной интеграции и развёртывания при разработке ПО. Также был выбран стек технологий, который подходит для небольшой команды разработчиков для увеличения скорости и качества разрабатываемых продуктов.

Списоклитературы:

1. Kitchenham, B.A. Software Engineering for Large Software Systems / B.A. Kitchenham. – Netherlands: B.A. Kitchenham, 1990. – 374 с.

2. Jenkins Documentation [Электронныйресурс] // Jenkins. – Режимдоступа: https://jenkins.io/doc/

3. Выстраиваем процесс разработки и CI pipeline, или как разработчику стать DevOps для QA [Электронный ресурс] // Хабрахабр. – Режим доступа: https://habr.com/post/330366/

4. What is CI/CD? [Электронный ресурс] // Mabl. – Режим доступа: https://www.mabl.com/blog/what–is–cicd – (Дата обращения: 20.05.2021).

5. TeamCity Documentation [Электронный ресурс] // TeamCity. – Режим доступа: https://www.jetbrains.com/teamcity/features/

6. Atlassian Documentation [Электронный ресурс] // Atlassian – Режим доступа:https://confluence.atlassian.com/alldoc/atlassian-documentation-32243719.html