МЕТОДИКА ВНЕДРЕНИЯ ТЕХНОЛОГИЙ НЕПРЕРЫВНОЙ ИНТЕГРАЦИИ И НЕПРЕРЫВНОЙ ДОСТАВКИ В ПРОЦЕСС РАЗРАБОТКИ ВЕБ-ОРИЕНТИРОВАННЫХ ПРОГРАММНЫХ СРЕДСТВ

Турчин Артур Чеславович

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

Процесс разработки и жизненный цикл задач

Автоматизацию процессов разработки стоит начинать после анализа собственного рабочего процесса, многие проблемы можно исправить за счет изменения административных частей, затем анализировать ситуацию и автоматизировать то, на что уходит много времени.

Рассмотрим подход к организации процесса разработки через планирование и сборки релизов. Данный подход характеризуется более простой организацией и предсказуемостью процесса. Критерии, когда его можно использовать в компаниях занимающихся разработкой собственного программного продукта:

- заказчик недостаточно доверяет команде разработки;

- команда занимается планированием всех задач;

- не требуется максимально возможная гибкость процесса разработки;

- руководство очень редко меняет планы в течении периода разработки;

- сборка проекта для тестирования занимает более 30 минут;

- использование специалистов по тестированию на аутсорсинге, когда в их задачи входит только проверка сборки на соответствие поставленным требованиям;

- нехватка квалификации и степени погружения в проект специалистов по тестированию, для правильного запуска необходимых частей проекта;

- в случае резкого изменения плана, все участники понимают и принимают тот факт, что не все задачи пойдут в релиз.

Перейдем к подходу в режиме потока. Данный подход накладывает собственные требования к команде разработчиков, вынуждает постоянно следить за равномерностью самого потока, чтобы задачи не зависали надолго в одном статусе. Повышает порог вхождения для всей команды, разработчикам требуется уделять большое внимание на дробление больших задач от бизнес-аналитиков на итерационные части, чтобы была возможность их реализовать поэтапно, иначе задача будет долго находиться в стадии реализации и тестирования, что повлечет нарушение и блокирование потока [2, с. 238]. Требуется обучить работе с проектом тестировщиков, чтобы они могли самостоятельно собрать для тестирования только функционал, в котором были произведены изменения, иначе много времени будет расходоваться постоянно на ожидание сборки целого продукта и, как следствие, весь поток начнет замедляться.

К режиму потока стоит прибегнуть если:

- требуется очень высокая гибкость процесса разработки;

- планы компании по развитию продукта часто меняются;

- требуется обеспечить постоянную и высокую скорость разработки;

- все члены команды обладают высокой квалификацией;

- есть возможность уделять большое внимание процессу оценки задач разработчиками;

- команда вовлечена в процесс разработки задач на всех его стадиях.

Компания ARTOX уже год придерживается режима потока, это обеспечило более высокую скорость работы за счет ежедневных обновлений продукта, минимизированы издержки в случае координального изменения плана разработки, перестали тратить время на согласование релиза.

Взаимодействие непрерывной интеграции и доставки

Давайте разберемся, что стоит автоматизировать в первую очередь, если в проекте нет непрерывной интеграции и доставки, чтобы эффект от внедрения был максимально быстрым и ощутимым.

Так или иначе, но непрерывная доставка требует наличия непрерывной интеграции, хотя бы в самом простом варианте [1]. Поэтому в веб-проекте начинать с интеграции стоит по следующим причинам:

- для работы программного средства требуется сложная сборка CSS-стилей и JS-скриптов;

- программное средство использует множество подключаемых пакетов от сторонних разработчиков;

- в разрабатываемом программном средстве вместе с разработкой новой функциональности идет поддержка и разработка новых модульных тестов;

- команда работает по процессу с релизами, когда нужно часто интегрировать изменения отдельных задач в релиз.

Стоит помнить о том, что экономия временных затрат будет происходить в команде, где над одним проектом трудится три и более разработчиков, которые часто производят изменения в самых разных местах программного средства.

Непрерывную доставку, без организации непрерывной интеграции, можно начинать когда веб-приложение обладает следующими качествами [3, с. 18]:

- при обновлении программного средства происходит недоступность приложения продолжительностью более 10 секунд при каждом обновлении;

- программное средство требуется доставить на три и более серверов при каждом обновлении;

- программное средство не требует артефактов или их сборка очень проста.

Если приложение обладает атрибутами из обоих списков, то наиболее сбалансированный процесс внедрения непрерывной интеграции и доставки выглядит следующим образом:

1. В случае отсутствия системы мониторинга над веб-проектом, необходимо обеспечить ее в первую очередь, она поможет быстрее уведомлять команду о сбоях в работе, а в случае частых релизов продукта, быстрее сообщать о том, что обновление привело к критическому сбою.

2. Найти решение проблемы недоступности приложения во время обновления или свести его

к минимуму.

3. Настроить сборку всех артефактов приложения: CSS-стили, JS-скрипты, сторонние пакеты.

4. Автоматизировать доставку и развертывание на серверах.

5. Автоматизировать доставку и развертывание на серверах.

6. Реализовать возможность быстрого отката изменений до предыдущего состояния.

Таким образом, мы получили оптимальную методику обеспечения непрерывного процесса разработки для использования в компаниях, занимающихся развитием собственного программного средства длительностью в несколько лет, выявили проблемы, которые ранее не проявлялись.

Выводы:

1. Прежде всего, команде стоит выбрать наиболее оптимальный и сбалансированный процесс гибкой разработки для решения своих задач, с планированием или в режиме потока, оба подхода в разных условиях хорошо себя зарекомендовали.

2. Любые изменения в устоявшемся процессе занимают достаточно много времени, поэтому всегда нужно начинать с оценки целесообразности изменений.

3. Иногда, для достижения хороших результатов требуется, выполнить предварительные работы, не связанные с непрерывной интеграцией или доставкой напрямую, например наладить систему мониторинга проекта.

Список литературы:

1. Блог Мартина Фаулера [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.martinfowler.com/articles/continuousIntegration.html (дата обращения: 15.07.2014)

2. Джефф Паттон. Пользовательские истории. Искусство гибкой разработки ПО, 2017. – 238 с.

3. Eberhard Wolff. A Practical Guide to Continuous Delivery, 2018. – 18 с.