Связь между сборкой и выпуском в тестировании программного обеспечения

Разработка программного обеспечения - сложный процесс. Это сложнее, чем написание обычной компьютерной программы, так как заказчик работает с конечным рабочим продуктом. Поэтому важно протестировать программное обеспечение. **Тестирование** - это процесс проверки и подтверждения правильности работы программного обеспечения в соответствии с требованиями. Это помогает снизить затраты и доставить работающему, полезному продукту клиенту. **Build и Release** - это два термина в разработке и тестировании.

После разработки программного модуля разработчики преобразуют исходные коды в автономную форму или исполняемый код. Затем команда разработчиков передает сборку команде тестирования для выполнения тестирования. Сборка находится в стадии тестирования; возможно, он уже прошел тестирование или нет. Команда тестирования программного обеспечения проверяет эту сборку. Если он состоит из нескольких ошибок и не соответствует требованиям, команда тестирования программного обеспечения отклоняет эту сборку. Сборки происходят до релиза, и они генерируются чаще.

Релиз является окончательным приложением после завершения разработки и тестирования. После тестирования сборки группа тестирования сертифицирует это программное обеспечение и доставляет его заказчику. Для одного выпуска возможно иметь несколько сборок. Таким образом, это программное обеспечение доставляется заказчику после завершения этапов разработки и тестирования. Более того, релиз основан на сборках и может иметь несколько сборок.

Один выпуск может иметь несколько сборок, в то время как выпуск основан на сборках.

Разница между сборкой и выпуском в тестировании программного обеспечения

Под сборкой понимается автономный программный артефакт, сгенерированный после преобразования исходного кода в исполняемый код, который можно запустить на компьютере. Релиз, с другой стороны, является распространением окончательной версии приложения.

Другими словами, Build - это версия программного обеспечения, которую команда разработчиков передает команде тестирования для целей тестирования, а Release - это программное обеспечение, которое команда тестирования передает клиенту.

В итоге, эти определения объясняют принципиальную разницу между сборкой и выпуском.

Таким образом, тестирование является основным отличием между сборкой и выпуском. Сборка все еще находится в фазе тестирования (может быть уже протестирована или еще не проверена), но релиз больше не требует тестирования.

«Релизный цикл» отдельно взятой команды в рамках одного спринта

Релизами называют запланированные изменения, которые вносят в программное или аппаратное обеспечение.

Релизный цикл − это комплексный процесс, объединяющий планирование, реализацию, тестирование и внедрение изменений в ИТ-продукт, а также контроль их качества при дальнейшем использовании.

1. Планирование

На этом этапе собираются требования к продукту от клиентов и внутренних заказчиков, анализируется актуальность предложенных изменений с учетом ресурсов, компетенций, времени на реализацию и совместимости с уже существующими функциями ПО.

Приоритетные запросы группируются и разделяются на блоки, задачи и подзадачи. Составляется подробный план релиза со сроками и контрольными точками. Указываются специалисты, которых необходимо привлечь к участию в проекте.

2. Сборка релиза

Организуется процесс разработки и стартует реализация намеченных функциональных изменений. Объемный по содержанию релиз разбивается на спринты и итерации.

3. Тестирование

Проводится тестирование доработок в собственной ИТ-среде. При необходимости формируются группы пользователей, которым предоставляется доступ к бета-версии для выявления критичных багов и первичной «обкатки» новых возможностей. По итогам тестирований производится финальная подготовка релиза с исправлением всех ошибок.

4. Развертывание

Осуществляется выпуск изменений, передача в эксплуатацию, документирование изменений. Выходит новая версия или сам конечный продукт. Все пользователи получают доступ к функциям, которые разрабатывались в релизе.

5. Контроль качества (внедрение изменений)

После развертывания участвующая в релиз менеджменте команда подводит итоги, анализирует выполненное. Если при внедрении новых функций возникает потребность в сопутствующих доработках, специалисты фиксируют их и планируют дальнейшее усовершенствование продукта в рамках нового релиза.

«Релизный цикл» в рамках отдельно взятой версии ПО, разрабатываемой несколькими командами

Управление релизами охватывает все этапы выпуска программного обеспечения, от разработки и тестирования до развертывания. Управление релизом требуется каждый раз, когда запрашивается новый продукт, или даже изменения в продукт существующий. Есть пять основных шагов релизного цикла:

1. Планирование релиза

Этап планирования в большинстве случаев интенсивен, так как именно на этом этапе весь релиз организуется от начала до конца. Надёжный план релиза помогает придерживаться верного пути и обеспечивать надлежащее соблюдение стандартов и требований.

При планировании релизов считается, что разработанные несколькими командами приложения нуждаются в согласованном подходе, который должен быть выпущен заранее. Именно здесь в игру вступает концепция «трейн-релиза». Следуя подходу трейн-релиза, команды могут планировать изменения на основе релизов и отправлять их в Play Store.

Самым первым шагом, является определение временных интервалов каждого этапа. В нашем случае этап разработки — это две недели. Затем нужно определить, сколько времени вы хотите потратить на интеграционное тестирование и этапы развёртывания. Ниже пример интервалов:

- Отсечение: начинается 1-й день.
- Внутреннее тестирование версий альфа/UAT: Три дня.
- Поэтапное развертывание [1%] Представление на ревью.

- Продолжительность ревью: 3 дня.
- Один день при 20% пользователей.
- Один день при 50% пользователей, в тот же день рост до 100% пользователей.

В соответствии с приведенным выше примером в общей сложности потребуется 10 дней, пока новая версия приложения не будет общедоступна.

Следующий шаг на пути к релизам — создание рабочего процесса, к которому на протяжении всего релиза могут обращаться как наши команды, так и ключевые заинтересованные стороны. Рабочий процесс сразу объясняет, как устроен весь релиз и какую роль играет каждый член команды.

Важные аспекты планирования релизов:

- релиз-менеджер управляет релизом и координирует его;
- принимается только код с проведенным ревью и протестированный код;
- в процессе есть этап внедрения;
- мониторинг выпущенной версии приложения.

2. Построение релиза

После того как план выпуска готов, можно приступать к тестированию продукта для релиза. Это будет фактическое «тестирование уровня пользователя» продукта на основе изложенных в плане выпуска требований.

В определённый день и время (например, в 15:00) происходит замораживание/отсечение кода. До этого момента команды успевают посмотреть, протестировать и смержить фичи в ветку разработки, которая должна быть частью трейн-релиза. В 15:00 релиз-менеджер создаст из ветки разработки ветку релиза.

Автоматизируя переход ветки, проверяются все пороговые значения производительности, бенчмарк и автоматизации из предыдущих релизов в маркет устанавливаются на текущий как основание сравнения, а трейн-релиз блокируется от дальнейших изменений.

Как только код замораживается, начинается новый цикл разработки, и все участвующие команды начинают новый спринт и продолжают разработку. Самое замечательное в трейн-релизе: все знают о следующем, запланированном релизе, и это помогает людям соответствующим образом планировать свою работу.

3. Релиз веток и контроль версий

Разработка продукта обычно не останавливается, когда заканчивается разработка для релиза, поэтому первое, о чем мы думаем, это как заморозить тестируемую сборку и в то же время поработать над новыми возможностями для следующего релиза.

Именно здесь на помощь приходит хитрая стратегия ветвления, в которой есть концепция разветвления. Ветвление кода означает, что точно так же, как

у веток дерева, код веток до определенной точки совпадает, а затем разбивается на две копии.

Каждая ветвь может развиваться независимо от другой. В этом случае одна копия — ветка релиза — остаётся замороженной на том месте, где вы завершили разработку (отсечённая ветка). Другая ветка (ветка разработки) может быть изменена новым кодом и исправлениями ошибок, не затрагивая ветку релиза вообще. Если ошибка найдена в релиз-кандидате, исправление может быть разработано и добавлено в ветку релиза.

Таким образом, следующая сборка, которую снова соберут из ветки выпуска, может быть идентична первой, за исключением исправления одной ошибки. Таким образом, можно минимизировать риск появления новых ошибок в выпуске и изолировать баги от нового кода. Исправление также применяется к ветке разработки, чтобы гарантировать, что та же самая ошибка не попадёт в следующий релиз. Другое преимущество релиз-ветвления состоит в том, что как только действительно публикуется код, имеется «замороженная» ветка, которая представляет собой точную копию опубликованной кодовой базы. Можно вернуться к этому коду в любое время для справки.

4. Пользовательское приемочное тестирование

Пользовательское приемочное тестирование является наиболее важным шагом для управления релизом из-за объема собранных данных и исправлений, необходимых для того, чтобы сделать сборку именно такой, какой она должна быть для официального запуска.

Пользователи — это именно те люди, которые будут пользоваться приложением. Поэтому крайне важно сделать пользователей частью всей стратегии обеспечения качества в процессе разработки программного обеспечения. Вот где пригодится UAT (User Acceptance Testing). Этот тип тестирования, как никакой другой, ставит потребности пользователей в центр работы над продуктом. Вот некоторые из вопросов, на которые такое тестирование пытается ответить:

- могут ли пользователи работать с приложением?
- соответствует ли поведение приложения ожиданиям?
- решает ли приложение проблему пользователей?

Без эффективного UAT шансы на успех разрабатываемого проекта значительно снижаются. Именно поэтому пользовательское является такой важной частью процесса поставки. Как отмечалось ранее, UAT — часть итеративного процесса. По мере выявления ошибок команда возвращается, чтобы исправить их. Ошибка исправляется в ветке релиза, а затем мержится обратно в ветку разработки. Сборка должна пройти стадию UAT, чтобы её можно было изучить на предмет окончательного внедрения и выпуска.

5. Подготовка и релиз

Подготовка релиза также включает в себя заключительную проверку качества командой по контролю качества. В ходе проверки группа по контролю качества проводит окончательные проверки, чтобы убедиться, что сборка соответствует минимальным стандартам приемки и бизнес-требованиям из плана релиза.

После завершения ревью релиз-группа завершит релиз для начала развертывания. Перед тем, как сборка может быть развернута в живой среде, она должна быть утверждена соответствующими ответственными командами, такими как команда дизайна. UAT гарантирует, что результат утверждается до передачи на следующий этап.

6. Анализ после релиза

Работа по управлению релизом не заканчивается, когда публикуется код, продолжаясь до тех пор, пока вы не будете готовы выпустить релиз снова. Дл того, чтобы приложение было успешным, ему нужно хорошее ревью, а также нужно следить за релизом в производственной среде, чтобы исправлять ошибки, внедрять функции, которые нужны людям, и решать проблемы пользователей.

Кроме того, ревью приложения дают представление о продукте, которое гораздо труднее получить при других подходах. И Google Play, и App Store предоставляют разработчикам приложений возможность отвечать на отзывы, что может быть невероятно полезным инструментом для получения дополнительной информации о проблемах с приложением от пользователей. Отзывы могут выявить проблемы, с которыми сталкиваются пользователи при работе с вашим приложением, и проинформировать о будущих изменениях.

Стадии разработки ПО

Жизненный цикл успешной компьютерной программы может быть очень долгим; изменения в программе бывают разными — от исправления ошибки до полного переписывания. В большинстве случаев название программы остаётся тем же, изменяется подназвание — так называемая **версия ПО.**

Версия программы может быть целым числом (Corel Draw 11), последовательностью чисел (JDK 1.0.3), годом (Windows 2000) или текстом (Embarcadero Delphi XE).

Milestone (веха) – контрольная точка, значимый, ключевой момент (например, переход на новую стадию, новый этап в ходе выполнения проекта). Как правило, с этим моментом связано завершение какого-либо ключевого мероприятия, подписание важных документов или любые другие значительные действия, предусмотренные планом проекта. Сдвиг вехи приводит к сдвигу всего проекта.

В дополнение к сигналу о завершении некоего ключевого этапа, веха употребляется в значении принятие важного, ключевого решения, способного изменить весь ход проекта. В этом смысле вехи отмечают не только контрольные точки процесса, но и указывают направление движения.

В каждый этап разработки добавляется изменение, оказывающее влияние на функционал ПО в целом. Это происходит в пределах каждого Milestone. Каждой вехе присваивается порядковый номер.

Стадии разработки программного продукта – это этапы, которые проходят команды разработчиков ПО, прежде чем программа станет доступной для широко круга пользователей. Стадии разработки используются для описания степени готовности программного продукта. Также стадия разработки может отражать количество реализованных функций, запланированных для определённой версии программы. Стадии либо могут быть официально объявлены и регламентируются разработчиками, либо иногда этот термин используется неофициально для описания состояния продукта.

Каждой стадии разработки ПО присваивается определенный порядковый номер. Также каждый этап имеет свое собственное название, которое характеризует готовность продукта на этой стадии.

1. Pre-Alpha — начальная разработка

— это период времени со старта разработки до выхода стадии альфа. Также так называются программы, не вышедшие ещё в стадию альфа или бета, но прошедшие стадию разработки, для первичной оценки функциональных возможностей в действии. В отличие от альфа- и бета-версий, начальный этап может включать в себя не весь спектр функциональных возможностей программы. В этом случае подразумеваются все действия, выполняемые во время проектирования и разработки программы вплоть до тестирования. К таким действиям относятся: разработка дизайна; анализ требований; собственно разработка приложения; отладка отдельных модулей.

2. Alpha — внутренняя разработка

Стадия начала тестирования программы в целом специалистами-тестировщиками, обычно не разработчиками программного продукта, но, как правило, внутри организации или сообществе разрабатывающих продукт. Также это может быть стадия добавления новых функциональных возможностей. Программы на данной стадии могут применяться только для ознакомления с будущими возможностями.

Как правило, альфа-тестирование заканчивается заморозкой функциональности и переходит в бета-тестирование.

3. Beta — общественная разработка

Стадия активного бета-тестирования и отладки программы, прошедшей альфа-тестирование (если таковое было). Программы этого уровня могут быть использованы другими разработчиками программного обеспечения для испытания совместимости. Тем не менее программы этого этапа могут содержать достаточно большое количество ошибок.

Поскольку бета-продукт не является финальной версией и публичное тестирование производится на страх и риск пользователя, производитель не

несёт никакой ответственности за ущерб, причинённый в результате использования бета-версии.

4. Вечная бета

Когда программа находится в бета-стадии неопределённый период времени. Такой механизм уместен в интернете, где ПО обладает такими свойствами:

- вместо инсталляторов программ — интернет-службы с дешёвой масштабируемостью;
- необычные и уникальные подборки данных, которые становятся богаче, когда расширяется пользовательская публика;
- конечные пользователи привлекаются в разработку. Их коллективный разум используется для техподдержки «длинного хвоста» с необычными запросами.
- программа выходит за рамки одного устройства;
- упрощённые пользовательские интерфейсы, принципы разработки и бизнес-модели;
- а производителе лежит особая ответственность за пользовательские данные, и многие уходят от неё, предоставляя пользователям вечную бету.

5. Release candidate — предварительная версия

Стадия-кандидат на то, чтобы стать стабильной. Программы этой стадии прошли комплексное тестирование, благодаря чему были исправлены все найденные критические ошибки. Но в то же время существует вероятность выявления ещё некоторого числа ошибок, не замеченных при тестировании. Если в течение установленного времени не будет найдено крупных недоработок — становится RTM-версией. Пример: Windows 7 RC 7100.

6. Выпуск

После выпуска программное обеспечение обычно называется «стабильным выпуском» (stable release).

Формальный термин часто зависит от способа выпуска: физический носитель, онлайн-выпуск или веб-приложение.

7. Release to manufacturing / выпуск в производство

Обозначение готовности программного продукта к тиражированию. Это стабильная версия программы, прошедшая все предыдущие стадии, в которых исправлены основные ошибки. RTM предшествует общей доступности (GA), когда продукт выпущен для общественности.

Данный термин обычно используется в определённых розничных условиях массового производства программного обеспечения, чтобы показать, что программное обеспечение соответствует определённому уровню качества и готово к массовому розничному распространению. RTM может также означать в других контекстах, что программное обеспечение было поставлено или выпущено клиенту или заказчику для установки или распространения на соответствующие компьютеры или компьютеры конечных пользователей оборудования.

Этот термин не определяет механизм или объём поставки; он лишь указывает, что качество является достаточным для массового тиражирования.

8. General availability / общедоступность

— стадия маркетинга, на которой завершены все необходимые мероприятия по коммерциализации, и программный продукт доступен для покупки, в зависимости, однако, от языка, региона, электронной или медийной доступности. Деятельность по коммерциализации может включать проверку безопасности и соответствия требованиям, а также локализацию и продвижение по всему миру. Время между выпуском в производство и общедоступностью может составлять от недели до нескольких месяцев. Это время необходимо для завершения всех мероприятий по коммерциализации, требуемых GA. На данном этапе программное обеспечение «вышло в жизнь» (gone live).

9. Release to web / веб-релиз

Выпуск в интернет (RTW) или веб-релиз является средством доставки программного обеспечения, которое использует интернет для его распространения. При этом изготовитель не задействует никакие физические носители. Веб-релизы становятся все более распространёнными по мере роста использования интернета.

10. Поддержка

В течение поддерживаемого срока службы программного обеспечения к нему выпускаются сервисные выпуски (service releases), патчи или пакеты обновления, иногда также называемые «промежуточные выпуски» (interim releases).

Например, в операционных системах Windows основная фаза поддержки длится 5-6 лет с момента общедоступности. В ОС типа Ubuntu существуют специальные версии LTS (Long Time Support), срок поддержки которых составляет 5 лет против 9 месяцев у обычных.

11. Прекращение поддержки

На этом этапе производитель объявляет об устаревании продукта и отказа от дальнейшей поддержки.