# Release

**Связь между сборкой и выпуском в тестировании программного обеспечения**

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

**В чем разница между сборкой и выпуском в тестировании программного обеспечения**

Главное отличие между сборкой и выпуском в тестировании программного обеспечения является то, что Build — это версия программного обеспечения, которую команда разработчиков передает команде тестирования для целей тестирования, а Release — это программное обеспечение, которое команда тестирования передает клиенту.

**Что такое релиз в тестировании программного обеспечения?**

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.

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. Команда разработчиков передает сборку команде тестирования, в то время как команда тестирования предлагает релиз своим клиентам.

**Релизный цикл**

Планирование выпуска — это процесс, в котором менеджеры по выпуску координируют свои действия с клиентами, бизнес-пользователями, разработчиками и другими экспертами, чтобы понять жизненный цикл разработки программного модуля, препятствия, требования к выпуску и сроки, когда программное обеспечение будет развернуто в различных средах, таких как производство.

Выпуски можно разделить на выпуски корпоративного уровня и командного уровня. Когда первая команда заканчивает с разработкой и тестированием своей части фичи, она передается второй команде. Вторая команда проводит интеграцию/разработку/ тестирование и передает следующей команде. И так до тех пор, пока все компонентные команды, которые участвуют в разработке фичи, не пройдут по этому флоу. Feature Owner валидирует фичу, и она отправляется в релиз.

**Стадии разработки ПО**

В разработке программного обеспечения 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.

Стадии Beta и Alpha не являются показателями нестабильности, так как присваиваются программе один раз или один раз за серию (серией, в данном случае, считается число до первой точки), в зависимости от системы разработки. Они могут присваиваться нескольким выпускаемым версиям подряд.

Изначально эти 7 этапов использовались на сайте SourceForge. Впоследствии эту нумерацию подхватил PyPI, хостинг пакетов для языка Python.

1. **Планирование** (*planning*). Автор зарезервировал название за проектом и начал очерчивание функциональности. Версии, как правило, не имеет.
2. **Пре-альфа** (*pre-alpha*). Уже есть какая-то программа, дающая представление о том, что она будет делать. Идёт разработка, добавление новой функциональности, рефакторинг. Архитектура программы в любой момент может полностью измениться. В этот момент программа уже может получить версию, обычно 0.x.y.
3. **Альфа** (*alpha*). Архитектура программы очевидна. Близкие к разработчику люди уже могут пользоваться программой. Идёт тестирование и доведение до продукта.
4. **Бета**(*beta*). Программа полнофункциональна. Идёт тестирование, исправление ошибок и проблем с производительностью, совершенствование эргономики.
5. **Готовая/стабильная** (*production/stable*). Нет критичных ошибок, оттестированы все основные сценарии использования. Идёт исправление ошибок, добавление новой функциональности. В этот момент программе можно дать версию 1.0.
6. **Зрелая** (*mature*). Больше года в состоянии «готовая/стабильная», не просят крупной функциональности, нет крупных и критичных ошибок. Идёт исправление мелких ошибок.
7. **Брошенная** (*inactive*). Разработка давно не ведётся. Найденные проблемы, скорее всего, не будут исправлены.

**Что такое Milestone?**

**Milestone (майлстоун)** — термин, использующийся в управлении проектами, который означает важную веху, ключевой этап, либо ключевого мероприятия, подписание важных документов или любые другие значительные действия, предусмотренные. В дополнение к сигнализации о завершении некого ключевого этапа. Многие методы и инструменты управления проектами определяют веху как задачу нулевой продолжительности. Это техническое определение носит упрощенный характер, поскольку вехи занимают отдельное место в управлении проектом, и можно управлять проектом или портфелем по вехам. При пилотировании по этапам обзор проекта состоит из проверки того, пройден ли этап или нет, и в определении пересчитанной даты перехода. Мониторинг прохождения основных этапов предоставляет несубъективную, контролируемую и

разделяемую информацию о ходе выполнения проекта. Некоторые индикаторы могут быть рассчитаны с использованием
контрольных точек для измерения прогресса или задержки проекта. Можно указать процент (%) успешно пройденных вех по сравнению с целевым% или последующие меры по достижению вехи, которая дрейфует больше всего. Ключевые события проекта, показывающие определенный прогресс проекта. Это может быть окончание работы, задачи, это также важное событие. В начальной и конечной вехах проекта есть промежуточные этапы, которые фиксируют промежуточные этапы в развитии проекта. В сочетании с методологией планирования, такой как метод PERT или метод критического пути (CPM), контрольные точки позволяют руководителям проектов более точно определять, просрочены ли результаты проекта. Ограничивая даты, связанные с этапами, можно определить критический путь для основных интервалов планирования в дополнение ко всему проекту.

## Альфа, бета-релиз и релиз-кандидат

**Pre-Alpha — начальная разработка**

**Начальная стадия разработки** — 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.

**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.

**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.

**Вечная бета**

Тим О'Райли, открытого ПО, выводит особый вид программ — «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.
- На производителе лежит особая ответственность за пользовательские данные, и многие уходят от неё, предоставляя пользователям вечную бету.

**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.