ВВЕДЕНИЕ В ПРОФЕССИЮ

Разработка программного обеспечения — сфера, которая будет в ближайшее время только расти, несмотря ни на эпидемию коронавируса, ни на экономический кризис. Соответственно, будет увеличиваться дефицит технических специальностей, связанных с информационными технологиями.

Одна из них — инженер по качеству ПО. В тестировщиках нуждаются практически все компании, которые занимаются разработкой программного обеспечения, сервисов, мобильных приложений и игр.

Почему бывает так, что программы работают неправильно? Программы разрабатываются и создаются людьми, которые допускают ошибки. Эти ошибки называются дефектами или багами.

Тестирование – не изолированный процесс. Это часть модели жизненного цикла программного обеспечения.

ОСНОВНЫЕ ПОНЯТИЯ

Тестирование программного обеспечения — проверка соответствия между реальным и ожидаемым поведением программы.

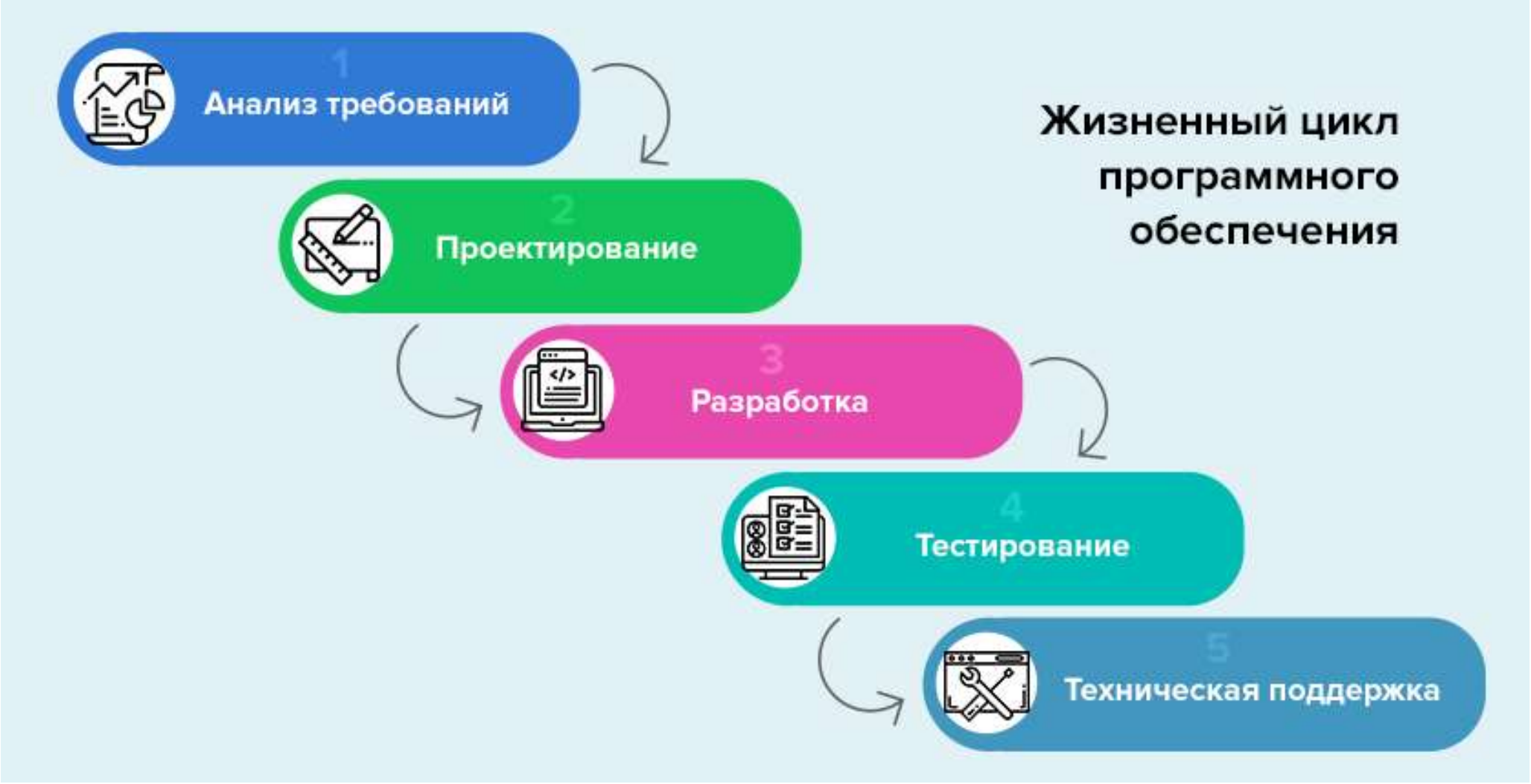
Качество ПО – комплекс характеристик программного продукта, определяющих способность выполнять возложенные на него функции.

Баг (Дефект) – это несоответствие фактического результата выполнения программы ожидаемому результату.

Баг-репорт – это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования.

Требования — это описание того, что должно быть реализовано. Требования описывают то, что необходимо реализовать, без детализации технической стороны решения.

ЖИЗНЕННЫЙ ЦИКЛ ПО



1. **Анализ требований.**

   Жизненный цикл разработки ПО начинается со стадии анализа. **Цель этой стадии – определение детальных требований к системе**.

   На данной стадии бизнес-аналитики формируют документацию, в которой написано как должно выглядеть и работать приложение. Также на этой стадии дизайнеры интерфейсов создают макеты будущего приложения. Данной документацией в дальнейшем будут пользоваться программисты и тестировщики. Программисты будут реализовывать требования, а тестировщики проверять, что разработчики сделали именно, что указывалось в документации.

2. **Проектирование.**

   На стадии проектирования программисты, руководствуясь требованиями, **разрабатывают дизайн системы,** т.е. планируют как будут реализовывать требования. Определяются технологии, инструменты, языки программирования, которые будут использоваться в проекте.

3. **Разработка.**

После того как требования и дизайн продукта утверждены, происходит переход к следующей стадии жизненного цикла – непосредственно разработке. Здесь начинается **написание программистами кода программы** в соответствии с ранее определенными требованиями. **Frontend** программисты разрабатывают пользовательский интерфейс, для этого они берут макеты, которые сделали дизайнеры и переводят их в компьютерный код, а **backend** разработчики создают логику работы приложения, а также занимаются базами данных, которые хранят всю информацию приложения.

4. **Тестирование**

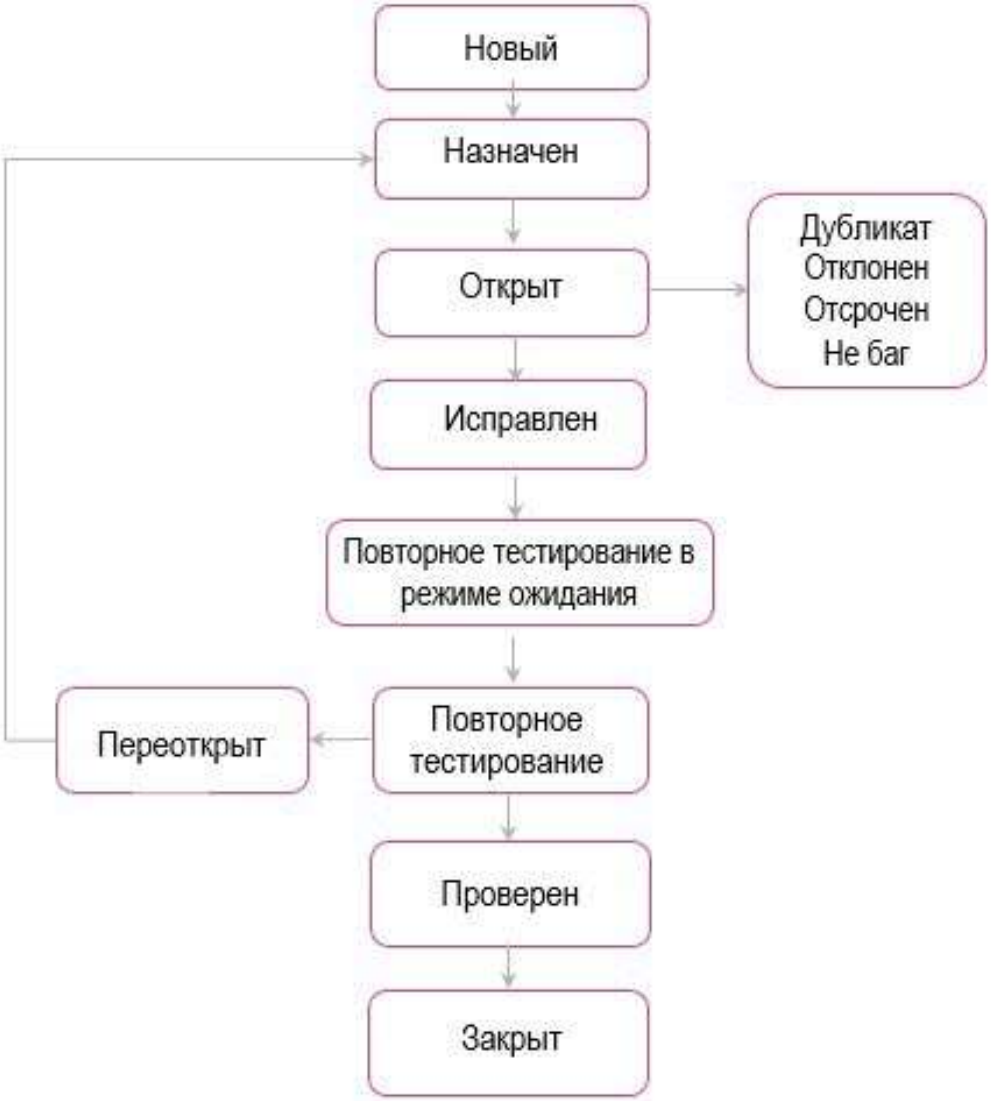
Тестировщики занимаются поиском дефектов в программном обеспечении и сравнивают описанное в требованиях поведение системы с реальным. В фазе тестирования обнаруживаются пропущенные при разработке баги. При обнаружении дефекта, тестировщик составляет отчет об ошибке, который передается разработчикам. Последние его исправляют, после чего тестирование повторяется – но на этот раз для того, чтобы убедиться, что проблема была

исправлена, и само исправление не стало причиной появления новых дефектов в продукте. Тестирование повторяется до тех пор, пока программа не будет работать согласно заявленным требованиям. Виды, методы и техники тестирования мы подробно рассмотрим дальнейших уроках.

5. Техническая поддержка.

Когда программа протестирована и в ней больше не осталось серьезных дефектов, приходит время релиза и передачи ее конечным пользователям. После выпуска новой версии программы в работу включается отдел технической поддержки. Его сотрудники обеспечивают обратную связь с пользователями, их консультирование и поддержку. В случае обнаружения пользователями тех или иных пост-релизных багов, информация о них передается в виде отчетов об ошибках команде разработки, которая, в зависимости от серьезности проблемы, либо немедленно выпускает исправление (т.н. hot-fix), либо откладывает его до следующей версии программы.

ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА



После того как мы нашли и завели баг в баг-трекинговой системе начинается жизненный цикл бага. Через какие стадии проходит баг:

Новый – отчет о дефекте заводится в баг-трекинговую систему в первый раз.
Назначен – отчет о дефекте назначается на соответствующего разработчика.
Открыт – разработчик берет отчет о дефекте в работу для анализа и исправления.
Исправлен – разработчик сделал необходимые изменения в коде и проверил эти изменения сам. Отчет о дефекте с этим статусом возвращается обратно тестировщику.
Повторное тестирование в режиме ожидания – после исправления дефекта разработчик вернул задачу тестировщику для повторного тестирования. Тестирование находится на рассмотрении у тестировщика.
Повторное тестирование – на этой стадии тестировщик выполняет повторное тестирование измененного кода, который был предоставлен разработчиком, для проверки, исправлен ли дефект или нет.
Проверен – если дефект не воспроизводится, тестировщик подтверждает, что этот дефект исправлен.
Переоткрыт– если дефект все же воспроизводится, даже после его исправления разработчиком, тестировщик переоткрывает его и назначает на разработчика. Этот дефект проходит через жизненный цикл дефекта еще раз.

Закрыт – если тестировщик уверен, что дефект больше не воспроизводится, то он его закрывает. Этот статус означает, что дефект исправлен, протестирован и одобрен.
Дубликат – если дефект повторяется дважды или есть два бага, которые являются следствием одной причины, то одному из них присваивается данный статус.
Отклонен – если разработчик считает, что этот дефект не является обоснованным или веским, и дефект не будет рассматриваться для исправления или реализации, он его отклоняет.
Отсрочен – ожидается, что дефект, которому присвоили такой статус, будет исправлен в следующих версиях. Причин для присвоения этого статуса может быть несколько: приоритет дефекта низкий, нехватка времени, данный дефект не повлечет больших сбоев в программном продукте.
Не баг – этот статус присваивается, если в функционал приложения не будет внесено никаких изменений. Например, если заказчик просит изменить цвет или размер кнопок, или текста – это не дефект, а просто изменения в дизайне приложения.

ПРИНЦИПЫ ТЕСТИРОВАНИЯ

Принцип 1 — Тестирование показывает наличие дефектов.

Тестирование может показать наличие дефектов в программе, но не доказать их отсутствие. В то же время, даже если дефекты не были найдены в процессе тестирования, нельзя утверждать, что их нет.

Принцип 2 — Исчерпывающее тестирование невозможно.

Невозможно провести исчерпывающее тестирование, которое бы покрывало все комбинации пользовательского ввода и состояний системы, за исключениям совсем уж примитивных случаев.

Принцип 3 — Раннее тестирование.

Следует начинать тестирование на ранних стадиях жизненного цикла разработки ПО, чтобы найти дефекты как можно раньше.

Принцип 4 — Скопление дефектов.

Разные модули системы могут содержать разное количество дефектов – то есть, плотность скопления дефектов в разных элементах программы может отличаться. Большая часть дефектов находится в ограниченном количестве модулей.

Принцип 5 — Парадокс пестицида.

Прогоняя одни и те же тесты вновь и вновь, Вы столкнетесь с тем, что они находят все меньше новых ошибок. Поскольку система эволюционирует, многие из ранее найденных дефектов исправляют и старые тесты больше не срабатывают. Чтобы преодолеть этот парадокс, необходимо периодически вносить изменения в используемые наборы тестов, корректировать их с тем, чтобы они отвечали новому состоянию системы и позволяли находить как можно большее количество дефектов.

Принцип 6 — Тестирование зависит от контекста.

Выбор методологии, техники и типа тестирования будет напрямую зависеть от природы самой программы. Тестирование проводится по-разному в зависимости от контекста. Например, сайт с большой посещаемостью должен пройти через серьезное тестирование производительности, чтобы показать возможность работы в условиях высокой нагрузки. Из тех же соображений для банковского ПО критически важна безопасность.

Принцип 7 — Заблуждение об отсутствии ошибок.

Тот факт, что тестирование не обнаружило дефектов, еще не значит, что программа готова к релизу.