Жизненный цикл программного обеспечения (Software Life Cycle Model) — это период времени, который начинается с момента принятия решения о создании программного продукта (средства) и заканчивается в момент его полного изъятия из эксплуатации.

«Программный продукт (software product) — совокупность компьютерных программ, процедур и, возможно, связанных с ними документации и данных».

В источниках могут встречаться разные наименования стадий, но, как правило, они сводятся к следующим:

Идея

Анализ требований

Проектирование

Разработка

Тестирование

Стабилизация и релиз

Поддержка и сопровождение

Каскадная (водопад) — модель, представляющая собой последовательное выполнение каждого из этапов жизненного цикла.

V-модель — разработка через тестирование, тесты внедряются в каждый из этапов производства.

Спиральная модель — предполагает цикличное повторение каждой стадии производства из раза в раз, вплоть до выпуска продукта.

Agile — гибкий подход к процессам разработки и производства, в основе которого лежат быстрый результат и взаимодействие с людьми.

Scrum — гибкая методика управлениями всеми стадиями жизненного цикла.

SRS - Software Requirement Specification спецификация требований к программному обеспечению

Тест-кейсы — профессиональная документация тестировщика, последовательность действий, направленная на проверку какого-либо функционала, описывающая, как прийти к фактическому результату.

«Низкая гибкость в управлении проектом, непригодность промежуточного продукта для использования, позднее обнаружение проблем в функционировании готового продукта — все это существенные недостатки каскадной модели.»

В связи с использованием данной модели процессы жизненного цикла довольно плохо оптимизируются, так как она не обеспечивает загрузку всей команды одновременно, а результат сильно растянут во времени. Тем не менее она пользуется спросом в отраслях с повышенными требованиями к качеству, например, там, где от каждой ошибки может зависеть жизнь человека.

Работа тестировщика в рамках каскадной модели будет, скорее всего, более монотонной и скрупулезной, но не менее интересной, чем в других моделях. Непосредственно перед этапом тестирования необходимо выделить достаточно времени, чтобы изучить всю документацию и проанализировать текущий набор задач (лучше всего сделать это сразу после того, как их передадут в разработку). Как вы понимаете, цена ошибки на конечной стадии очень велика и большую часть рутинной работы будет составлять перепроверка.

**Спиральная модель жизненного цикла**

Суть спиральной модели в том, что весь процесс создания конечного продукта представлен в виде условной плоскости, разбитой на четыре сектора:

определение целей;

оценка рисков;

разработка и тестирование;

планирование новой итерации.

При прохождении каждого цикла спирали на выходе получаем прототип. Так функционал продукта наращивается с каждым витком, и в итоге доводится до нужного (запланированного) состояния, вплоть до релиза (конец спирали).

Главная особенность спиральной модели — концентрация на возможных рисках. Для их оценки даже выделена соответствующая стадия.

Основные типы рисков, которые могут возникнуть в процессе разработки ПО:

Нереалистичный бюджет и сроки;

Дефицит специалистов;

Частые изменения требований;

Чрезмерная оптимизация;

Низкая производительность системы;

Несоответствие уровней квалификации специалистов разных отделов.

Использование спиральной модели будет уместно:

при внедрении крупных долгосрочных проектов с отсутствием чётких требований или вероятностью их динамического изменения;

когда важен высокопрофессиональный анализ рисков и затрат;

при разработке новой линейки продуктов.

Манифест гибкой разработки программного обеспечения (**Agile Manifesto**) — основной документ, содержащий описание ценностей и принципов гибкой разработки программного обеспечения.

Текст манифеста доступен более чем на 50 языках (и на русском), включает в себя четыре ценности, двенадцать принципов. С манифестом можно ознакомиться по ссылке.

Сейчас можно смело сказать, что Agile (Agile software development) — обобщающий термин для целого ряда подходов и практик, основанных на ценностях «Манифеста гибкой разработки программного обеспечения» и принципах, лежащих в его основе.

В процессе работы и усовершенствования подходов методологии были сформированы и зафиксированы 12 принципов AGILE:

Наивысшим приоритетом является удовлетворение потребностей клиента, благодаря регулярной и ранней поставке ценного программного обеспечения.

Изменение требований приветствуется даже на поздних стадиях разработки.

Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.

На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.

Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.

Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

Работающий продукт — основной показатель прогресса.

Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно.

Постоянное внимание к техническому совершенству и к качеству проектирования повышает гибкость проекта.

Простота — искусство минимизации лишней работы — крайне необходима.

Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.

Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Задачи в Sсrum-методологии выступают в виде артефактов. Это тот пул работы, который нужно выполнить, чтобы закрыть спринт. Информация о проекте в таком случае прозрачна для всех участников.

Есть три обязательных артефакта:

Product Backlog — это список всех элементов продукта, требований к ним и любой связанной с продуктом информации. Бэклог продукта формируется на всем протяжении проекта.

Sprint Backlog — набор элементов бэклога продукта, который команда принимает в разработку на ближайший спринт.

Инкремент — сумма завершенных во время спринта элементов бэклога продукта и всех инкрементов предыдущих спринтов. То есть это текущее состояние разрабатываемого продукта, включая доработки последнего спринта.

Есть также список необязательных артефактов. Они помогут вашей команде облегчить жизнь. С полным перечнем артефактов и подробным руководством от создателей методик и практик внедрения можно познакомиться по ссылке.

На рынке огромное количество таск-трекеров они же таск-менеджеры (англ. task tracker, «отслеживатель задач»), специализированные сайты или программы на компьютере, которые можно использовать как сервисы для совместной работы. И в них же команды заводят задачи, смотрят статусы, отвечают на вопросы (заказчиков, например). Существует великое множество разных таск-трекеров, которые могут размещаться как на сервере компании, так и в облаке. Сейчас сложно сказать, какой из сервисов наиболее популярный, востребованный, удобный. Кто-то выбирает JIRA, Asana, Яндекс.Трекер (о них можно почитать здесь), а для других команд подходит интерактивная доска Trello.

Вот на ней мы и остановимся, так как она доступна, располагает к себе приятным и понятным интерфейсом, достаточным количеством функций в бесплатном режиме. Итак, регистрируемся в Trello.

https://habr.com/ru/company/hygger/blog/460985/