Технологии разработки программного обеспечения

Практическое занятие 1

Жизненный цикл разработки ПО (англ. SDLC – Software development lifecycle) – это серия из шести фаз, через которые проходит любая программная система.

Абсолютно любое ПО проходит через 6 основных шагов, начиная от простой идеи и заканчивая использованием её конечным пользователем.

Типичный жизненный цикл разработки состоит из следующих фаз:

1. Сбор и анализ требований (Planning and Requirement Analysis)

Это, пожалуй, самый ответственный и важный из всех шагов к созданию успешной программной системы. Вся собранная информация используется для планирования базового проектного подхода.

Дополнительно идет планирование требований по обеспечению качества и выявления различных рисков, связанных с проектом. Результатом анализа является определение различных технических подходов, которые можно использовать для успешной реализации проекта с минимальными рисками. Планируйте то, что вы можете контролировать, и помните о вещах,

планировать которые вы не сможете. Это поможет вам получить прочную основу для перехода ко второму этапу.

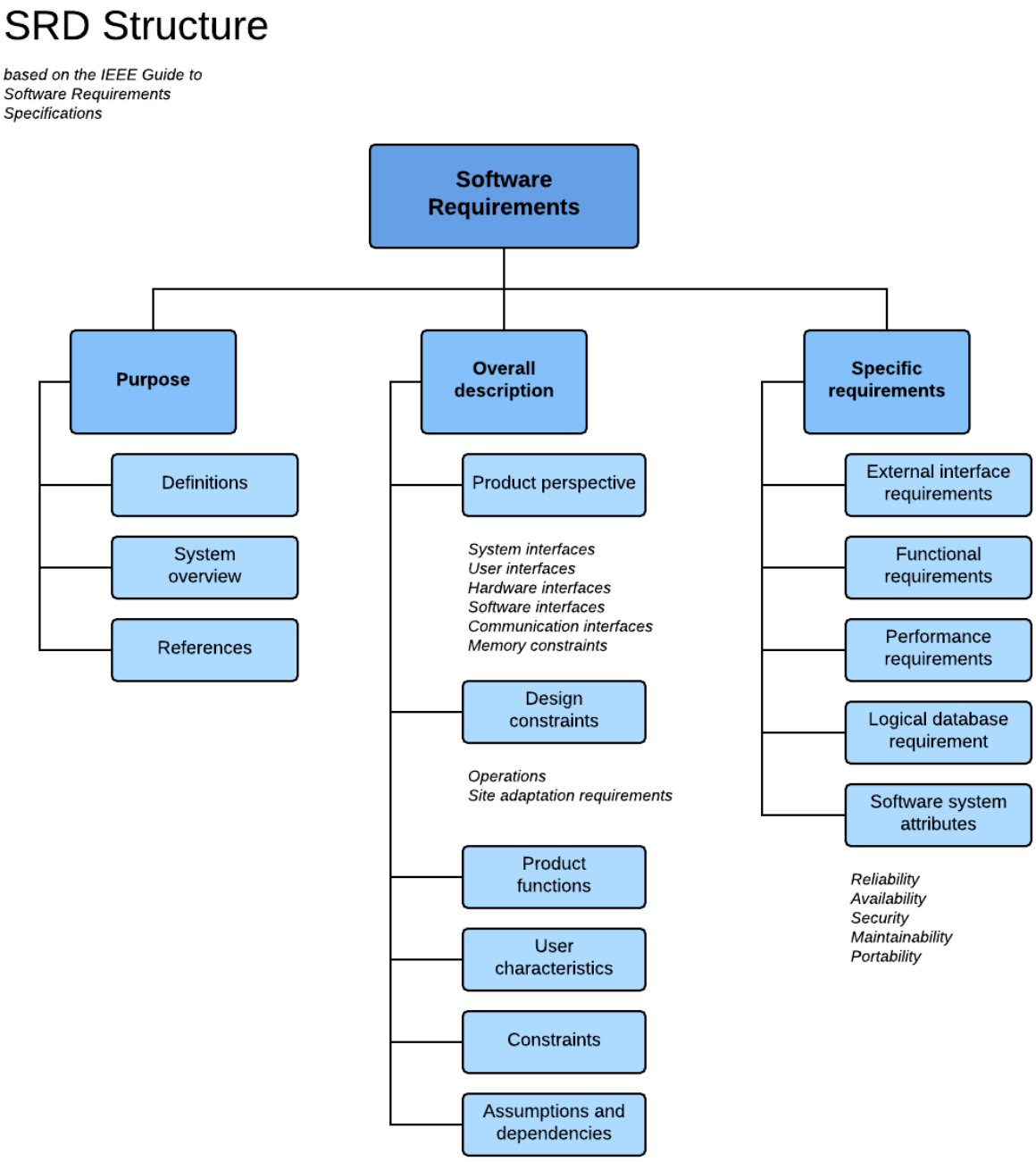
Проблема: не соответствующие ожидания и часто изменяющиеся требования: заказчик и команда не понимают, какую реально пользу принесёт продукт.

Решение: определить скоп работ, согласовать четкий, краткий документ с требованиями, создать прототипы (макеты) для подтверждения и уточнения окончательных требований.

2. Документирование требований (Defining Requirements)

Как только базовый анализ требований будет выполнен, следующим шагом будет четкое определение и документирование требований к продукту, утверждение со стороны клиента. Если одной из целей первого этапа является понимание и анализ требований, то на этом этапе все цели должны быть прописаны, это защита обеих сторон.

Важно четко определить и прописать, что требуется выполнить, это делается с помощью SRS (Software Requirement Specification). Документ содержит все требования к продукту, которые должны быть спроектированы и разработаны в течение жизненного цикла проекта.



Проблема: большой многостраничный список требований

Решение: выяснить высокоуровневые требования и, в ходе разработки и коммуникации с заказчиком, дописывать ТЗ. То есть разработка идет параллельно с Техническим заданием, а в процессе корректируется план.

3. Дизайн (Design the Product Architecture)

SRS это ориентир для разработчиков, чтобы предложить лучшую архитектуру для продукта. Обычно предлагается несколько подходов к проектированию архитектуры продукта. Все предложенные подходы документируются в спецификации DDS (Design Document Specification) и выбирается наилучший подход к проектированию. Данный подход очень четко

определяет все архитектурные модули продукта, а также его связь с внешними и сторонними модулями.

Проблема: выбрали неправильную архитектуру.

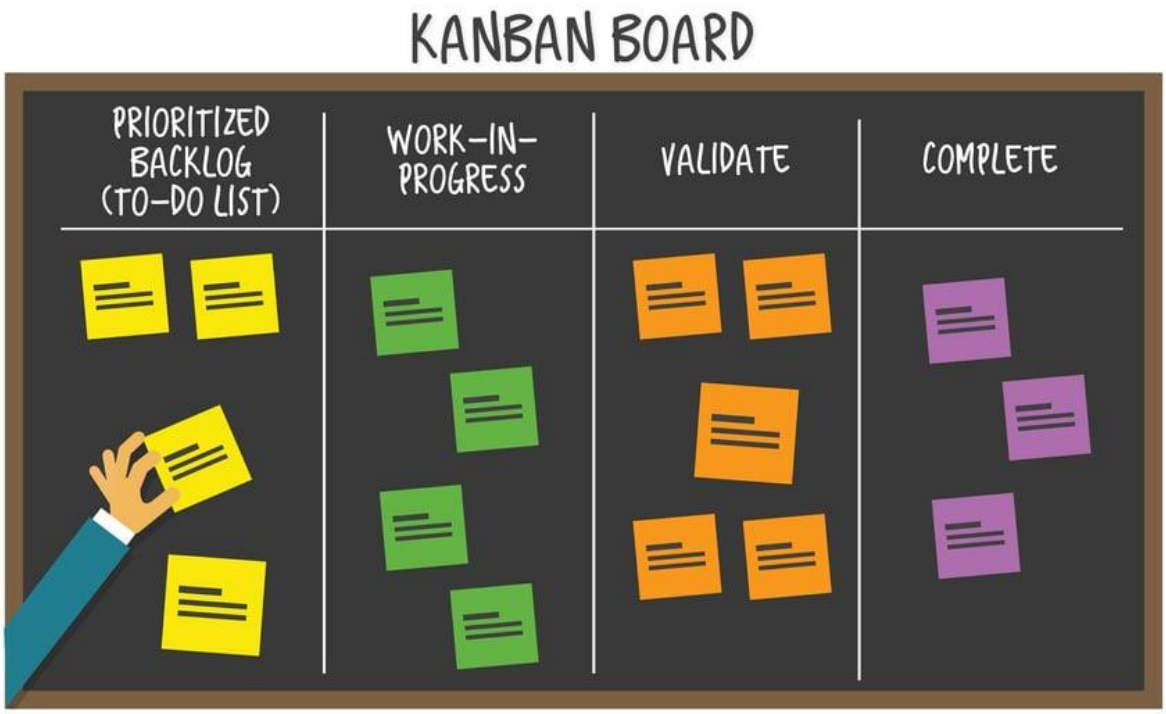
Решение: наличие в команде разработчиков опытных тимлидов, которые смогли бы предложить подходящую архитектуру, на основе своего успешного опыта в схожих проектах.

4. Разработка ПО (Building or Developing the Product)

Здесь начинается разработка и сборка продукта. Весь программный код, новые модули и фичи разрабатываются на основании DDS. Чем лучше написана эта документация, тем быстрее будет идти имплементация. На этом этапе подключается команда разработчиков. Написанный код должен покрываться Unit-тестами, а взаимодействие новых фич с другими модулями тестироваться с помощью интеграционных тестов. Эти активности выполняются именно командой разработчиков, а не QA специалистами.

Проблема №1: Слабая коммуникация между командой. Разработчик не интересуется прогрессом проекта, проблемами коллег.

Решение: Daily meetings, 100% вовлеченность, Скрам доска (интерактивность).



Проблема №2: Невыполнимые сроки

Заказчик хочет, чтобы его продукт был готов в ближайшее время. Менеджер проекта пытается объяснить клиенту к чему приведет такая спешка,

но этого недостаточно. Клиент ставит невыполнимые дедлайны и не слушает возражения менеджера проекта. В результате команда разработчиков сдается и пробует закрыть задачи в слишком короткие сроки. Как следствие – критические баги из-за спешки: команда не успевает, качество продукта снижается, клиент не доволен и решает, что виновата команда.

Решение: Менеджер проекта должен стоять на своём и аргументировать дедлайны. Клиент должен понять, что если команда разработчиков будет торопиться, то не реализует часть функционала. Если всё же такой срок реализации критичен для клиента, мы стараемся выявить какие задачи находятся у заказчика в приоритете.

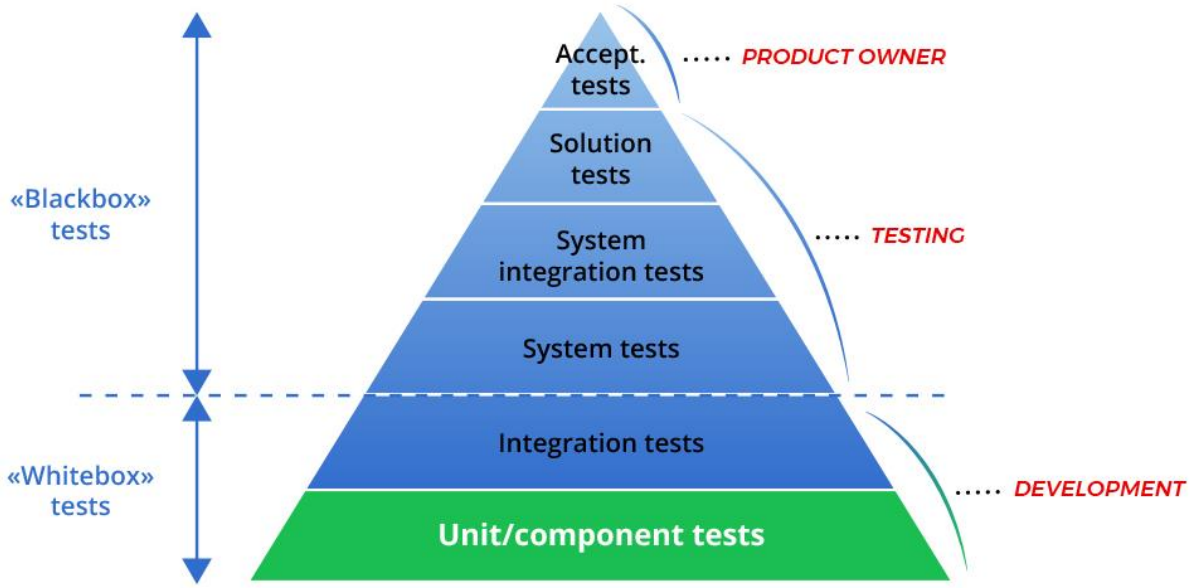
Проблема №3: Добавление не оговоренных фич

99% заказчиков ошибаются именно в этом месте. В ходе разработки клиент отклоняется от оговоренного ТЗ и хочет добавить ещё фич в продукт. В результате вместе с ростом скопа фич, увеличиваются сроки и бюджет на разработку, деньги заканчиваются, а готово только 50% продукта.

Решение: Менеджер проекта должен объяснить клиенту, к чему приведет добавление новых фич в проект, отстаивать свою позицию и держаться SRS. Поэтому так важна вторая фаза Жизненного цикла разработки ПО.

5. Тестирование (Testing the Product)

Именно тестирование, в основном, затрагивает все этапы жизненного цикла. Дефекты продукта регистрируются, отслеживаются, исправляются и повторно тестируются. Это происходит до тех пор, пока продукт не достигнет стандартов качества, которые прописаны в SRS. На данном этапе в процесс разработки подключается команда мануальных тестировщиков или автоматизаторов.



Проблема: тратится слишком много времени на поиск причин багов. Попытки найти и исправить дефекты после завершения разработки – сложный процесс, который может привести к большому количеству исправлений.

Решение: проводить тестирование параллельно задачам, сразу же по их завершению.

6. Внедрение и поддержка продукта (Deployment in the Market and Maintenance)

Как только продукт протестирован, он выходит в релиз. Иногда внедрение происходит поэтапно, в соответствии с бизнес-стратегией. Продукт сначала может быть выпущен в ограниченном сегменте и протестирован в реальной бизнес-среде, это UAT-тестирование (User Acceptance Testing). Затем, основываясь на отзывах, продукт может быть выпущен как есть, или с предлагаемыми улучшениями. После того, как продукт выпущен на рынок его обслуживание выполняется для существующей клиентской базы, и на этом этапе подключаются Support-команды.

Проблема №1: отсутствие обратной связи, реальных отзывов потенциальных пользователей продукта.

Решение: не ждать конца разработки, а как можно скорее запускать продукт, чтобы получить отзывы от реальных пользователей и на основе их предпочтений приоритезировать дальнейший функционал.

Проблема №2: слабая инфраструктура проекта на стороне клиента.

Часть заказчиков предпочитают размещать сервера приложений не на Azure, AWS, Google и т.д, а в своей внутренней сети. Команда не может

гарантировать стабильную работу, из-за сбоев, которые происходят на стороне клиента.

Решение: предупредить клиента, о возможных проблемах, предложить решения для их устранения.

Это шесть основных стадий жизненного цикла разработки системы, и это повторяющийся процесс для каждого проекта. Важно отметить, что должен поддерживаться отличный уровень коммуникации с заказчиком. Для реализации требований очень важны и полезны прототипы. Строя систему короткими итерациями, можно гарантировать соответствие требованиям потребителя до того, как построить целую систему.