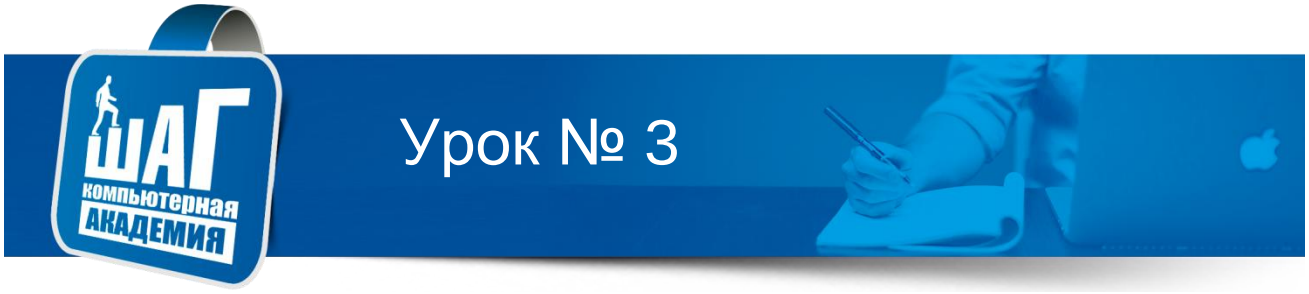

Курс: «Мануальное тестирование ПО»

Тема: Методологии процесса разработки программного обеспечения: Водопадная модель, спиральная модель, итеративная модель (agile, scrum, xp), RUP, MSF.

План

1. Введение.
2. Водопадная модель.
3. V- образная модель.
4. Спиральная модель.
5. Итеративная модель (agile, scrum, xp)
6. RUP, MSF.

1. Введение.

Методология — это система принципов, а также совокупность идей, понятий, методов, способов и средств, определяющих стиль разработки программного обеспечения.
Модель жизненного цикла программного обеспечения — это структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.
Все модели можно разделить на 3 основных группы:

1. Инженерный подход
2. С учетом специфики задачи
3. Современные технологии быстрой разработки.

2. Водопадная модель.

Водопадная модель (WATERFALL MODEL)— модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.

В водопадной модели , фазы разработки ПО идут в таком порядке:

1. Определение требований
2. Анализ
3. Проектирование
4. Кодирование
5. Тестирование и отладка
6. Эксплуатация
7. Поддержка

Рис. 1 Водопадная модель.

Следуя каскадной модели, разработчик переходит от одной стадии к другой строго последовательно. Сначала полностью завершается этап «определение требований», в результате чего получается список требований к ПО. После того как требования полностью определены, происходит переход к проектированию, в ходе которого создаются документы, подробно описывающие для программистов способ и план реализации указанных требований. После того как проектирование полностью выполнено, программистами выполняется реализация полученного проекта. На следующей стадии процесса происходит интеграция отдельных компонентов, разрабатываемых различными командами программистов. После того как реализация и интеграция завершены, производится тестирование и отладка продукта; на этой стадии устраняются все недочёты, появившиеся на предыдущих стадиях разработки. После этого программный продукт внедряется и обеспечивается его поддержка — внесение новой функциональности и устранение ошибок.

Переход от одной фазы разработки к другой происходит только после полного и успешного завершения предыдущей фазы, и что переходов назад либо вперёд или перекрытия фаз — не происходит.

Водопадная модель- это тот процесс, который можно сертифицировать. Используют эту модель для разработки ПО, которое используется в работе медицинского оборудования, космических кораблей и т.д.

Одном из главных недостатков водопадной модели является «предрасположенность» к возможным несоответствиям полученного в результате продукта и требований, которые к нему предъявлялись. Основная причина этого

заключается в том, что полностью сформированный продукт появляется лишь на поздних этапах разработки, но так как работу на разных этапах обычно выполняли различные специалисты и проект передавался от одной группы к другой, то по принципу испорченного телефона оказывалось так, что на выходе получалось не совсем то, что предполагалось вначале.

3. V- образная модель.

V-образная модель. Была предложена для того, чтобы устранить недостатки каскадной модели, а название – V-образная, или шарнирная – появилось из-за ее специфического графического представления (рис. 2).

Рис.2 V-образная модель.

V-образная модель дала возможность значительно повысить качество ПО за счет своей ориентации на тестирование. Эта модель во многом разрешила проблему соответствия созданного продукта выдвигаемым требованиям благодаря процедурам верификации и аттестации на ранних стадиях разработки (пунктирные линии на рисунке указывают на зависимость этапов планирования/постановки задачи и тестирования/приемки).

Однако в целом V-образная модель является всего лишь модификацией каскадной и обладает многими ее недостатками. В частности, и та и другая слабо приспособлены к возможным изменениям требований заказчика. Если процесс разработки занимает продолжительное время (иногда до нескольких лет), то полученный в результате продукт может оказаться фактически ненужным заказчику, поскольку его потребности существенно изменились.

В той же мере актуален и вопрос влияния научно-технического прогресса: требования к ПО выдвигаются с учетом текущего состояния научных и практических достижений в области аппаратно-программного обеспечения, однако IT-сфера развивается очень быстро, и затянувшийся процесс разработки способен привести к созданию продукта, который базируется на устаревших технологиях и оказывается неконкурентоспособным еще до своего появления.

Важен также вопрос планирования показателей ожидаемой функциональности, поскольку в этих моделях он является не более чем допущением: в частности, определить, какую скорость обработки данных обеспечит создаваемый продукт либо сколько он будет занимать памяти, на этапе постановки задачи практически невозможно. Если подобные требования четко зафиксированы в условиях договора между заказчиком и исполнителем, то вполне вероятно, что полученное решение не будет им удовлетворять, хотя известно это станет только на завершающих этапах разработки, когда основные ресурсы уже израсходованы.

В итоге заказчик будет вынужден либо мириться с ограничениями созданного на основе рассмотренных моделей решения, либо дополнительно инвестировать средства, чтобы получить действительно то, что необходимо.

4. Спиральная модель

Спиральная модель представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции.

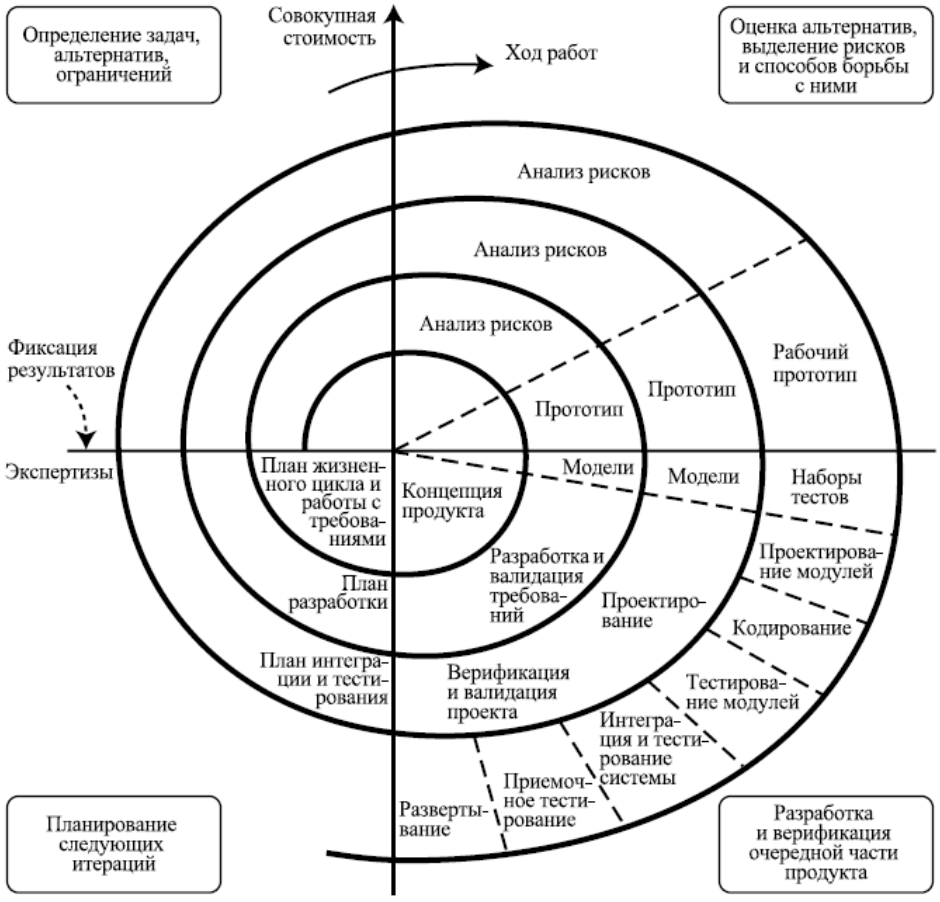

Рис. 3 Изображение хода работ по спиральной модели.

В квадранте 1 – анализ требований, альтернативных вариантов и ограничений – определяются рабочие характеристики, выполняемые функции, стабильность (возможность внесения изменений), аппаратно/программный интерфейс.

Определяются альтернативные способы реализации системы (разработка, повторное использование компонент, покупка, договор подряда и т.п.). Определяются ограничения, налагаемые на применение альтернативных вариантов (затраты, график выполнения, интерфейс, ограничения среды и др.).

Определяются риски, связанные с недостатком опыта в данной предметной области, применением новой технологии, жесткими графиками, недостаточно хорошо организованными процессами.

В квадранте 2 – оценка альтернативных вариантов, идентификация и разрешение рисков – выполняется оценка альтернативных вариантов, рассмотренных в предыдущем квадранте; оценка возможных вариантов разрешения рисков. Выполняется прототипирование как основа для работ следующего квадранта.

В квадрант 3 – разработка продукта текущего уровня – включаются действия по непосредственной разработке системы или программного продукта: проектирование системы и ее программных компонентов, разработка и тестирование исходных текстов программ, интеграция тестирование и квалификационные испытания продукта или системы и т.п.

В квадранте 4 – планирование следующей фазы – выполняются действия, связанные с разработкой планов проекта, управления конфигурацией, тестирования, установки системы.

Наиболее важным отличием этой модели от других является учет рисков. Риск – это вероятность того, что что-то может пойти не так. Из-за материализации рисков превышаются сроки, и происходит перерасход средств, поэтому необходимо учитывать риски и принимать меры по их смягчению.

Спиральная модель ориентирована на большие, дорогостоящие и сложные проекты. В условиях, когда бизнес цели таких проектов могут измениться, но требуется разработка стабильной архитектуры, удовлетворяющей высоким требованиям по нагрузке и устойчивости, имеет смысл применение Spiral Architecture Driven Development. Данная методология, включающая в себя лучшие идеи спиральной модели и позволяет существенно снизить архитектурные риски, что является немаловажным фактором успеха при разработке крупных систем.

Итерация - это законченный цикл разработки, приводящий к выпуску внутренней или внешней версии изделия (или подмножества конечного продукта), которое совершенствуется от итерации к итерации, чтобы стать законченной системой.

Каждый виток спирали соответствует созданию фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы на следующем витке спирали. На каждой итерации углубляются и последовательно конкретизируются детали проекта, в результате чего выбирается обоснованный вариант, который доводится до окончательной реализации.

Использование спиральной модели позволяет осуществлять переход на следующий этап выполнения проекта, не дожидаясь полного завершения текущего – недоделанную работу можно будет выполнить на следующей итерации. Главная задача каждой итерации – как можно быстрее создать работоспособный продукт, который можно показать пользователям системы. Таким образом существенно упрощается процесс внесения уточнений и дополнений в проект.

5. Итеративная модель (agile, scrum, xp)

Итеративная модель (iteration — повторение) — выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы. Проект при этом подходе в каждой фазе развития проходит повторяющийся цикл: Планирование — Реализация — Проверка — Оценка.

Преимущества итеративного подхода:

- снижение воздействия серьезных рисков на ранних стадиях проекта, что ведет к минимизации затрат на их устранение;
- организация эффективной обратной связи проектной команды с потребителем и создание продукта, реально отвечающего его потребностям;
- акцент усилий на наиболее важные и критичные направления проекта;
- непрерывное итеративное тестирование, позволяющее оценить успешность всего проекта в целом;
- раннее обнаружение конфликтов между требованиями, моделями и реализацией проекта;
- более равномерная загрузка участников проекта;
- эффективное использование накопленного опыта;
- реальная оценка текущего состояния проекта и, как следствие, большая уверенность заказчиков и непосредственных участников в его успешном завершении.

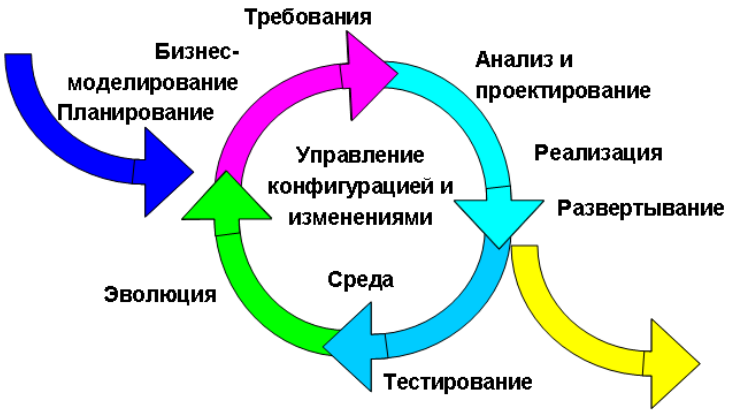

Рис. 4 Итеративная модель.

Agile-методы.

Гибкая методология разработки (*Agile software development*) — серия подходов к разработке программного обеспечения, ориентированных на использование интерактивной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует несколько методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование(XP), Scrum.

XP - одна из гибких методологий разработки программного обеспечения.
Основные группы экстремального программирования:
- короткий цикл обратной связи;
- непрерывный процесс;
- понимание, разделяемое всеми;
- социальная защищенность программиста.

XP предполагает написание автоматических тестов. Особое внимание уделяется двум разновидностям тестирования:
- юнит-тестирование модулей;
- функциональное тестирование.

Одним из приёмов XP является парное программирование. Парное программирование предполагает, что весь код создается парами программистов, работающих за одним компьютером.

Скрам (Scrum) — это ещё одна из гибких методологий разработки ПО. Характеризуется набором принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО, с новыми возможностями, для которых определён наибольший приоритет.
Основные роли в методологии Scrum**:**
Скрам мастер- проводит совещания, следит за соблюдением всех принципов скрам, разрешает противоречия и защищает команду от отвлекающих факторов.
Владелец продукта- представляет интересы конечных пользователей и других заинтересованных в продукте сторон.
Скрам команда- команда разработчиков проекта, состоящая из специалистов разных профилей: тестировщиков, архитекторов, аналитиков, программистов и т.д. Размер команды в идеале составляет 7-9 человек.

Спринт — итерация в скраме, в ходе которой создаётся функциональный рост программного обеспечения. Жёстко фиксирован по времени. Длительность одного спринта от 2 до 4 недель.
Бэклог проекта — это список требований к функциональности, упорядоченный по их степени важности, подлежащих реализации. Элементы этого списка называются «пожеланиями пользователя» (user story) или элементами бэклога .

Диаграмма сгорания задач (Burndown chart)- это диаграмма, показывающая количество сделанной и оставшейся работы. Обновляется ежедневно с тем, чтобы в простой форме показать подвижки в работе над спринтом.

Также примером реализации итеративного подхода служит методология RUP.

6. RUP, MSF.

RATIONAL UNIFIED PROCESS — методология разработки программного обеспечения, созданная компанией Rational Software.
В основе методологии лежат 6 основных принципов:

- компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта;
- работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам;
- ранняя идентификация и непрерывное устранение возможных рисков;
- концентрация на выполнении требований заказчиков к исполняемой программе;
- ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки;
- постоянное обеспечение качества на всех этапах разработки проекта.

Использование методологии RUP направлено на итеративную модель разработки. Особенность методологии состоит в том, что степень формализации может меняться в зависимости от потребностей проекта. Можно по окончании каждого этапа и каждой итерации создавать все требуемые документы и достигнуть максимального уровня формализации, а можно создавать только необходимые для работы документы.
За счет такого подхода к формализации процессов методология является достаточно гибкой и широко популярной.
Данная методология применима как в небольших и быстрых проектах, где за счет

отсутствия формализации требуется сократить время выполнения проекта и расходы, так и в больших и сложных проектах, где требуется высокий уровень формализма, например, с целью дальнейшей сертификации продукта. Это преимущество дает возможность использовать одну и ту же команду разработчиков для реализации различных по объему и требованиям.

MICROSOFT SOLUTIONS FRAMEWORK — методология разработки программного обеспечения, предложенная корпорацией Microsoft. MSF опирается на практический опыт Microsoft и описывает управление людьми и рабочими процессами в процессе разработки решения.
Базовые концепции и принципы модели процессов MSF:

- единое видение проекта — все заинтересованные лица и просто участники проекта должны чётко представлять конечный результат, всем должна быть понятна цель проекта;
- управление компромиссами — поиск компромиссов между ресурсами проекта, календарным графиком и реализуемыми возможностями;
- гибкость – готовность к изменяющимся проектным условиям;
- концентрация на бизнес-приоритетах — сосредоточенность на той отдаче и выгоде, которую ожидает получить потребитель решения;
- поощрение свободного общения внутри проекта;
- создание базовых версии — фиксация состояния любого проектного артефакта, в том числе программного кода, плана проекта, руководства пользователя, настройки серверов и последующее эффективное управление изменениями, аналитика проекта.

MSF предлагает проверенные методики для планирования, проектирования, разработки и внедрения успешных IT-решений. Благодаря своей гибкости, масштабируемости и отсутствию жестких инструкций MSF способен удовлетворить нужды организации или проектной группы любого размера. Методология MSF состоит из принципов, моделей и дисциплин по управлению персоналом, процессами, технологическими элементами и связанными со всеми этими факторами вопросами, характерными для большинства проектов.

Существует множество различных методологий разработки программного обеспечения, они не универсальны и описываются различными принципами. Выбор методологии разработки для конкретного проекта зависит от предъявляемых требований.