Topic 2. Software Development Life Cycle and Methodologies

Содержание:

1. Разработка ПО.
2. Верификация и валидация.
3. V- model.
4. Spiral model.
5. Waterfall model.
6. SCRUM.
7. CANBAN.
8. MSF.
9. RUP.
10. Жизненный цикл продукта.

Разработка ПО — это род деятельности и процесс, направленный на создание и поддержание работоспособности, качества и надежности программного обеспечения, используя технологии, методологию и практики из информатики, управления проектами, математики, инженерии и других областей знаний.

Методология — это система принципов, а также совокупность идей, понятий, методов, способов и средств, определяющих стиль разработки программного обеспечения.

Прогнозируемые методологии фокусируются на детальном планировании будущего.

Адаптивные методологии нацелены на преодоление ожидаемой неполноты требований и их постоянного изменения.

Верификация — это обычно внутренний процесс управления качеством, обеспечивающий **согласие с правилами, стандартами или спецификацией**.
Валидация — подтверждение на основе представления объективных свидетельств того, что требования, предназначенные для **конкретного** использования или применения, выполнены.

Простой способ запомнить разницу между валидацией и верификацией заключается в том, что валидация подтверждает, что «вы создали правильный продукт», а верификация подтверждает, что «вы создали продукт таким, каким и намеревались его сделать».

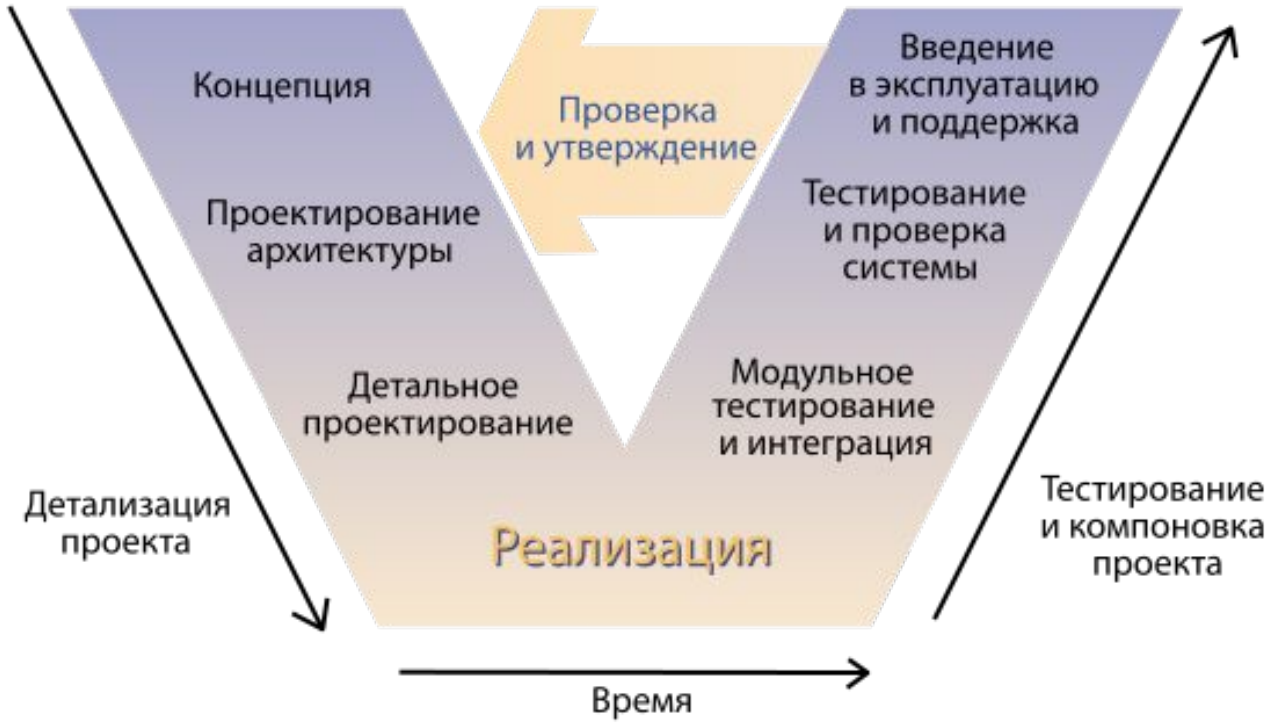
V-Model

Основной принцип V-образной модели заключается в том, что детализация проекта возрастает при движении слева направо, одновременно с течением времени, и ни то, ни другое не может повернуть вспять. Итерации в проекте производятся по горизонтали, между левой и правой сторонами буквы.

V-Model


Планирование проектов и требований
Производство, эксплуатация и сопровождение
Анализ требований продукта и спецификаций
Системное и приемочное тестирование
Разработка архитектурного проекта на высшем уровне
Интеграция и тестирование
Детализированная разработка проекта
Модульное тестирование
Кодирование

V-Model


Концепция
Проектирование архитектуры
Детальное проектирование
Проверка и утверждение
Введение в эксплуатацию и поддержка
Тестирование и проверка системы
Модульное тестирование и интеграция
Реализация
Детализация проекта
Тестирование и компоновка проекта
Время

V-Model

V-модель обеспечивает поддержку в планировании и реализации проекта. В ходе проекта ставятся следующие задачи:

Минимизация рисков: V-образная модель делает проект более прозрачным и повышает качество контроля проекта путём стандартизации промежуточных целей и описания соответствующих им результатов и ответственных лиц.

Повышение и гарантии качества: V-Model — стандартизованная модель разработки, что позволяет добиться от проекта результатов желаемого качества. Промежуточные результаты могут быть проверены на ранних стадиях.

Уменьшение общей стоимости проекта: Ресурсы на разработку, производство, управление и поддержку могут быть заранее просчитаны и проконтролированы. Получаемые результаты также универсальны и легко прогнозируются.

Повышение качества коммуникации между участниками проекта: Универсальное описание всех элементов и условий облегчает взаимопонимание всех участников проекта.

V-Model

Достоинства:

• Пользователи V-Model участвуют в разработке и поддержке V-модели.

• На старте любого проекта V-образная модель может быть адаптирована под этот проект, так как эта модель не зависит от типов организаций и проектов.

• V-model позволяет разбить деятельность на отдельные шаги, каждый из которых будет включать в себя необходимые для него действия, инструкции к ним, рекомендации и подробное объяснение деятельности.

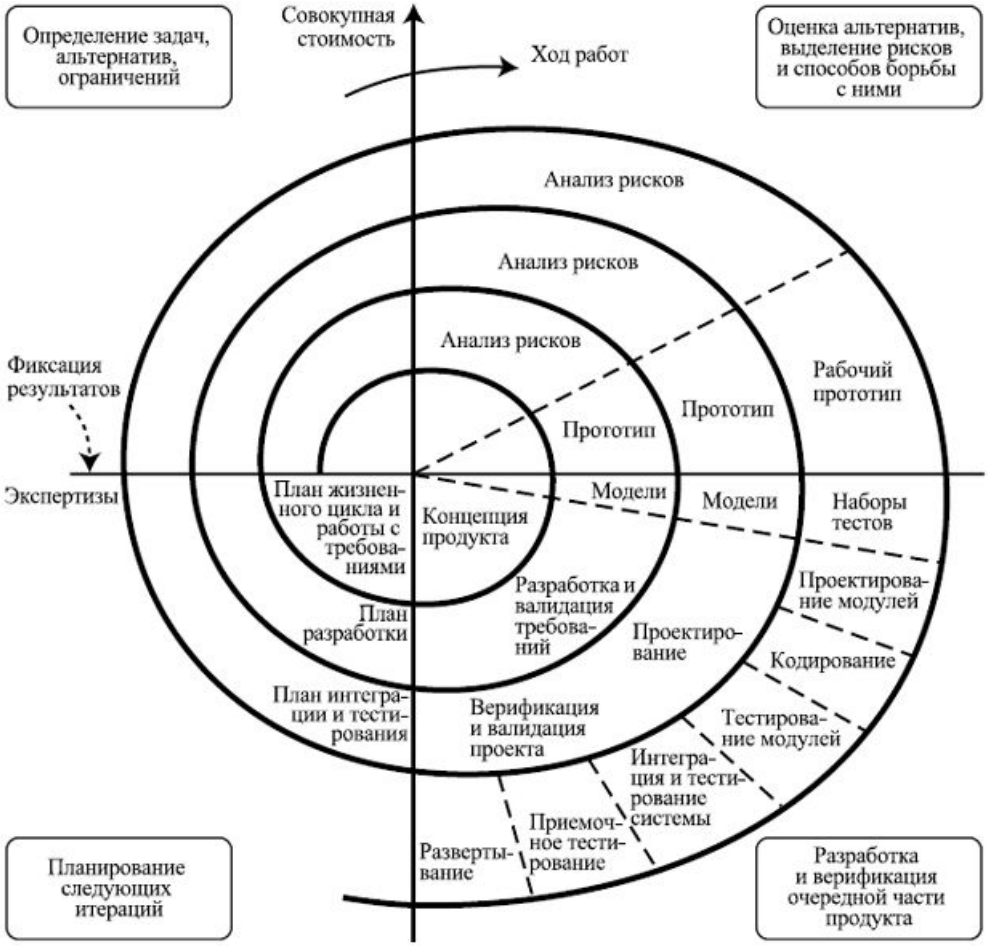
Ограничения:

• Не регулируется размещение контрактов на обслуживание.

• Организация и выполнение управления, обслуживания, ремонта и утилизации системы не учитываются в V-модели.

• V-образная модель больше касается разработки программного

Spiral Model



Spiral Model

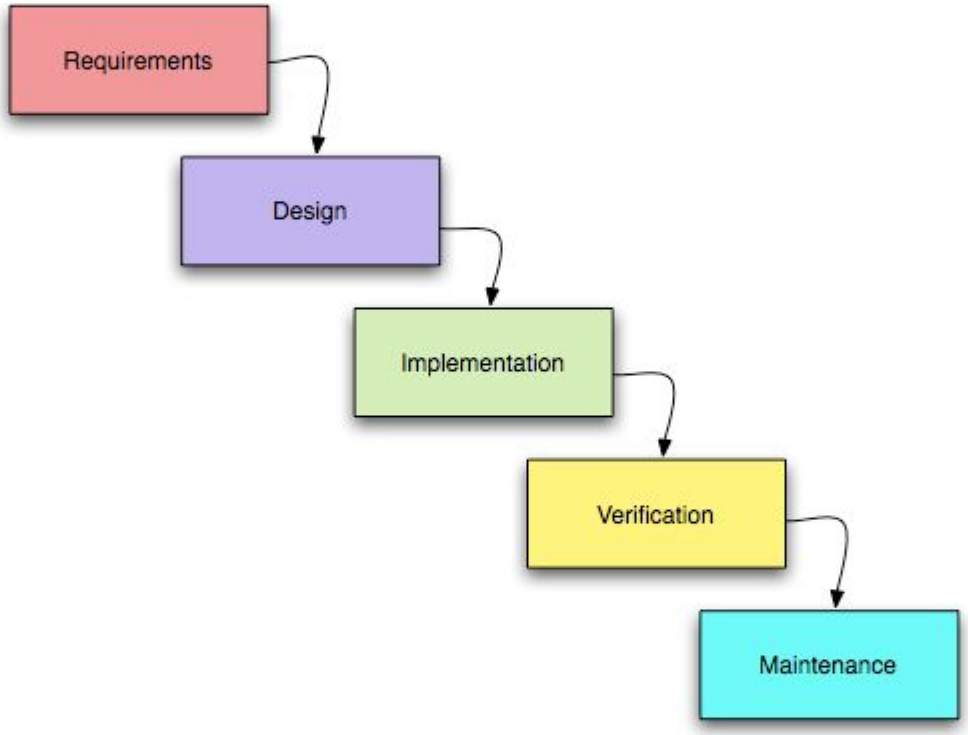
Представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции, делающая упор на начальные этапы жизненного цикла: анализ и проектирование.

Spiral Model

Каждый виток спирали соответствует созданию фрагмента или версии программного обеспечения, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации. Каждый виток разбит на 4 сектора:

- оценка и разрешение рисков,
- определение целей,
- разработка и тестирование,
- планирование.

Waterfall Model



Waterfall Model

Каскадная 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.

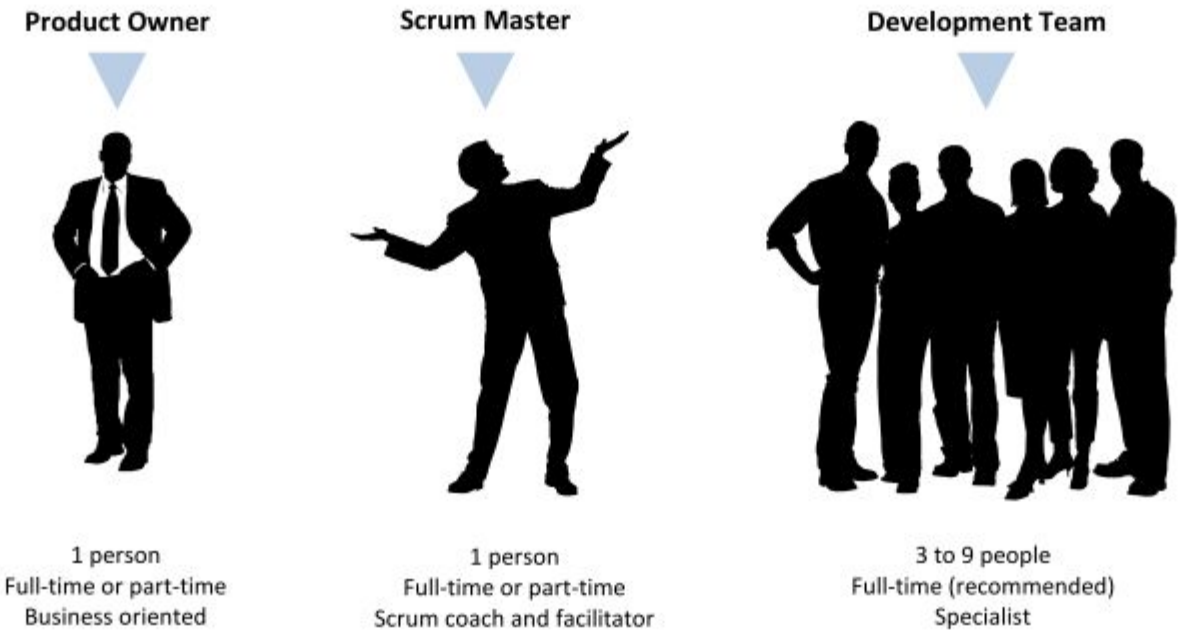
Waterfall Model

Методику «Каскадная модель» довольно часто критикуют за недостаточную гибкость и объявление самоцелью формальное управление проектом в ущерб срокам, стоимости и качеству. Тем не менее, при управлении большими проектами формализация часто являлась очень большой ценностью, так как могла кардинально снизить многие риски проекта и сделать его более прозрачным. Поэтому формально была закреплена только методика «каскадной модели» и не были предложены альтернативные варианты ведение проектов.

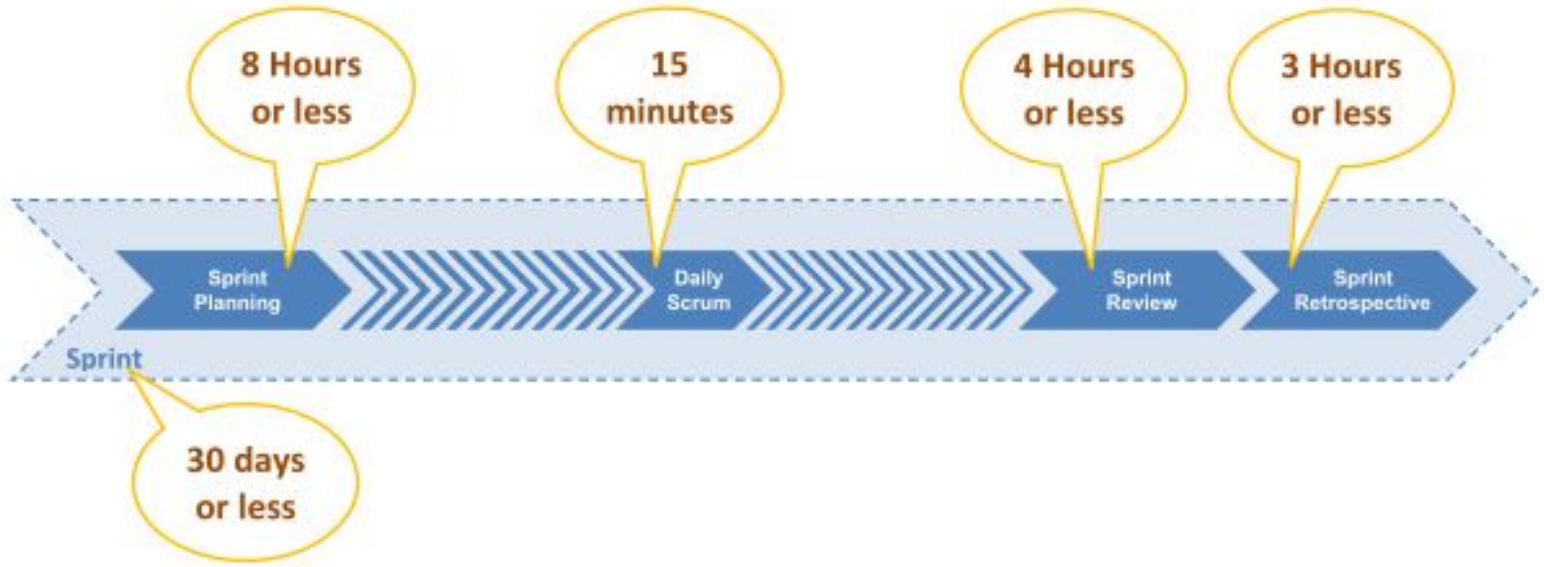
SCRUM

SCRUM — методология, предназначенная для **небольших команд** (3-9 человек). Весь проект делится на итерации (спринты) продолжительностью 30 дней или меньше. Выбирается список функций системы, которые планируется реализовать в течение следующего спринта. Самые важные условия — неизменность выбранных функций во время выполнения одной итерации и строгое соблюдение сроков выпуска очередного релиза, даже если к его выпуску не удастся реализовать весь запланированный функционал. Скрам Мастер проводит ежедневные 15 минутные совещания, которые так и называют — daily scrum, результатом которых является определение функции системы, реализованных за предыдущий день, возникшие сложности и план на следующий день. Такие совещания позволяют постоянно отслеживать ход проекта, быстро выявлять возникшие проблемы и оперативно на них реагировать.

SCRUM Team


Product Owner
Scrum Master
Development Team
1 person
Full-time or part-time
Business oriented
1 person
Full-time or part-time
Scrum coach and facilitator
3 to 9 people
Full-time (recommended)
Specialist

SCRUM Events



KANBAN

KANBAN – гибкая методология разработки программного обеспечения, ориентированная на задачи.

Основные правила:

- **визуализация разработки**:
- **разделение работы на задачи**;
- **использование отметок** о положение задачи в разработке;
- **ограничение работ**, выполняющихся одновременно, на каждом этапе разработки;
- **измерение времени цикла** (среднее время на выполнение одной задачи) и оптимизация процесса.

Преимущества KANBAN:

- **уменьшение числа параллельно выполняемых задач** значительно уменьшает время выполнения каждой отдельной задачи;
- **быстрое выявление проблемных задач**;
- **вычисление времени** на выполнение усредненной задачи.


EXPEDITE
GOALS
STORY QUEUE
ELABORATION & ACCEPTANCE
DEVELOPMENT
TEST
DEPLOYMENT
DONE!
YOUR WAIT TIME FROM HERE IS ABOUT 14 DAYS
YOUR WAIT TIME FROM HERE IS ABOUT 18 DAYS
5
3
5
3
1

Во-первых, нужно сразу понять, что Канбан — это не конкретный процесс, а система ценностей. Как, впрочем, и SCRUM с XP. Это значит, что никто вам не скажет что и как делать по шагам.
Во-вторых, весь Канбан можно описать одной простой фразой — **«Уменьшение выполняющейся в данный момент работы (work in progress)»**.
В-третьих, Канбан — это даже еще более «гибкая» методология, чем SCRUM и XP. Это значит, что она не подойдет всем командам и для всех проектов. И это также значит, что команда должна быть еще более готовой к гибкой работе, чем даже команды, использующие SCRUM и XP.

Разница между Канбан и SCRUM:
— В Канбан нет таймбоксов ни на что (ни на задачи, ни на спринты)
— В Канбан задачи больше и их меньше
— В Канбан оценки сроков на задачу опциональные или вообще их нет
— В Канбан «скорость работы команды» отсутствует и считается только среднее время на полную реализацию задачи

Канбан разработка отличается от SCRUM в первую очередь ориентацией на задачи. Если в SCRUM основная ориентация команды — это успешное выполнение спринтов (надо признать, что это так), то в Канбан на первом месте задачи.
Спринтов никаких нет, команда работает над задачей с самого начала и до завершения. Деплоймент задачи делается тогда, когда она готова. Презентация выполненной работы — тоже. Команда не должна оценивать время на выполнение задачи, ибо это имеет мало смысла и почти всегда ошибочно вначале.
Цели проекта:
Необязательный, но полезный столбец. Сюда можно поместить высокоуровневые цели проекта, чтобы команда их видела и все про них знали. Например, «Увеличить скорость работы на 20%» или «Добавить поддержку Windows 7».
Очередь задач:
Тут хранятся задачи, которые готовы к тому, чтобы начать их выполнять. Всегда для выполнения берется верхняя, самая приоритетная задача и ее карточка перемещается в следующий столбец.

Проработка дизайна:
этот и остальные столбцы до «Закончено» могут меняться, т.к. именно команда решает, какие шаги проходит задача до состояния «Закончено».
Например, в этом столбце могут находиться задачи, для которых дизайн кода или интерфейса еще не ясен и обсуждается. Когда обсуждения закончены, задача передвигается в следующий столбец.
Разработка:
Тут задача висит до тех пор, пока разработка фичи не завершена. После завершения она передвигается в следующий столбец.
Или, если архитектура не верна или не точна — задачу можно вернуть в предыдущий столбец.
Тестирование:
В этом столбце задача находится, пока она тестируется. Если найдены ошибки — возвращается в Разработку. Если нет — передвигается дальше.
Деплоймент:
У всех проектов свой деплоймент. У кого-то это значит выложить новую версию продукта на сервер, а у кого-то — просто закомитить код в репозиторий.
Закончено:
Сюда стикер попадает только тогда, когда все работы по задаче закончены полностью.

Весь Канбан можно описать всего тремя основными правилами:
1. **Визуализируйте производство**
— Разделите работу на задачи, каждую задачу напишите на карточке и поместите на стену или доску.
— Используйте названные столбцы, чтобы показать положение задачи в производстве.
2. **Ограничивайте WIP** (work in progress или работу, выполняемую одновременно) **на каждом** этапе производства.
3. **Измеряйте время цикла** (среднее время на выполнение одной задачи) и **оптимизируйте постоянно процесс**, чтобы уменьшить это время.

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.


Deployment Complete
Deployment Stabilized
Site Deployments Complete
Core Technology Deployed
Deploying
Envisioning
Core Team Organized
Vision/Scope Baselined
Release Readiness Approved
Vision/Scope Approved
Pilot Complete
Pre-Production Testing Complete
Release Candidates
User Acceptance Testing Complete
Zero Bug Bounce
Bug Convergence
Stabilizing
MSF
Planning
Technology Validation Complete
Functional Specifications Baselined
Master Project Plan Baselined
Master Project Schedule Baselined
Development/Test Environment Set Up
Developing
Scope Complete
Project Plans Approved
Proof of Concept Complete
Internal Release 1
Internal Release 2
Internal Release n

RATIONAL UNIFIED PROCESS

RUP — методология разработки программного обеспечения, созданная компанией Rational Software.

В основе методологии лежат 6 основных принципов:

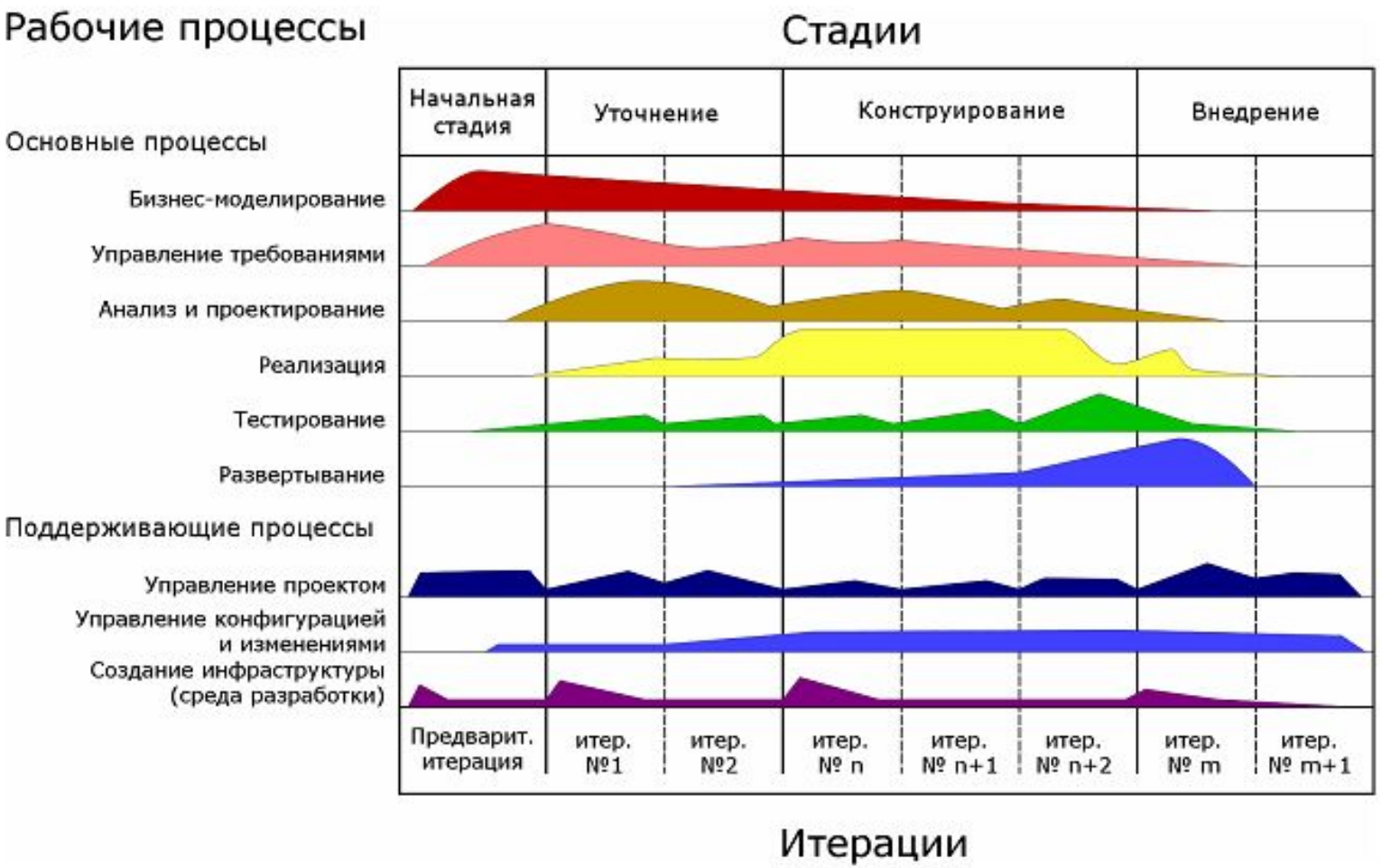
- **компонентная архитектура**, реализуемая и тестируемая на ранних стадиях проекта;
- **работа над проектом в сплочённой команде**, ключевая роль в которой принадлежит архитекторам;
- **ранняя идентификация** и непрерывное устранение возможных рисков;
- **концентрация на выполнении требований** заказчиков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

RATIONAL UNIFIED PROCESS

Использование методологии RUP направлено на итеративную модель разработки. Особенность методологии состоит в том, что степень формализации может меняться в зависимости от потребностей проекта. 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.

Фазы в RUP:

- **Начальная стадия (Inception)**
- **Уточнение (Elaboration)**
- **Построение (Construction)**
- **Внедрение (Transition)**


Рабочие процессы
Стадии
Начальная стадия
Уточнение
Конструирование
Внедрение
Основные процессы
Бизнес-моделирование
Управление требованиями
Анализ и проектирование
Реализация
Тестирование
Развертывание
Поддерживающие процессы
Управление проектом
Управление конфигурацией и изменениями
Создание инфраструктуры (среда разработки)
Предварит. итерация
итер. №1
итер. №2
итер. № n
итер. № n+1
итер. № n+2
итер. № m
итер. № m+1
Итерации

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

В любой модели ЖЦ ПО имеется несколько характеристик качественного тестирования:

- Каждому процессу разработки соответствует свой процесс тестирования
- Каждый уровень тестирования имеет свои цели тестирования
- Анализ и дизайн тестов для какого-либо уровня тестирования должны начинаться одновременно с соответствующей деятельностью разработчиков
- Тестировщики должны быть вовлечены в процесс просмотра и рецензирования документов, как только становятся доступными их предварительные версии.
- Тестировщик должен быть вовлечен в процесс разработки как можно раньше.

Цикл разработки ПО

Идея – точка отсчета или то, с чего начинается любой проект.

Дизайн (product design) – разработка и документация того, как необходимо воплотить идею в жизнь, др. словами как готовый продукт должен выглядеть/работать.

Спецификация (specification) – документ описывающий требования к продукту.

1. **Акцент на деталях и их четкое определение.**
2. **Забота о недопущении неверного толкования.**
3. **Непротиворечивость с другими документами.**
4. **Логическая взаимосвязь компонентов.**
5. **Полнота охвата предмета.**
6. **Соответствие нормативным актам.**
7. **Соответствие деловой практике.**

Статусы спецификации:

- **Черновик** (Draft)
- **Ожидание утверждения** (Approval Pending)
- **Утверждено** (Approved)

CVS – система контроля версий.

Макеты (mock-up), **блоксхемы** (flow chart) и **примеры** (example).

Кодирование.

Работа программиста заключается в том, чтобы перевести вещи, отраженные в спецификации, на язык программирования.

Основные занятия программиста:

- **Написание кода**
- **Интеграция кода**
- **Ремонт багов**

Как правило, существуют минимум две версии ПО: **тестовая** и **продакшин**.

Причины возникновения багов в коде:

- Некачественные и/или изменяющиеся спецификации
- Личностные качества программиста
- Отсутствие опыта
- Пренебрежение стандартами кодирования
- Сложность системы
- Баги в ПО третьих лиц
- Отсутствие юнит-тестирования
- Нереально короткие сроки для разработки

Тестирование и ремонт багов

Тест приемки (smoke test).

Тестирование новых компонентов (new feature testing).

Регрессивное тестирование (regression testing).

Система трэкинга багов (Bug Tracking System).

Build — это версия версии ПО.

Перед тестированием **обязательно** нужно проверить версию ПО.

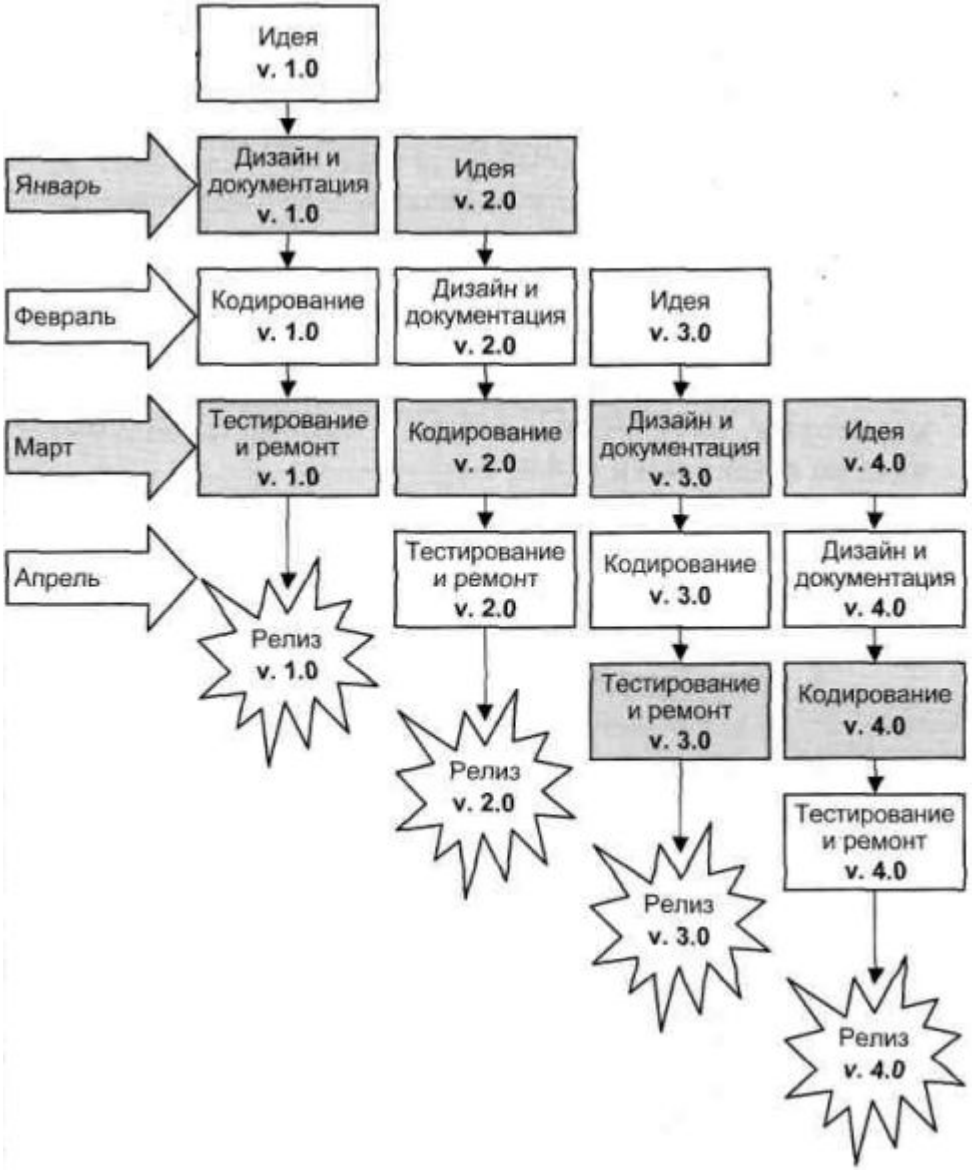
Релиз

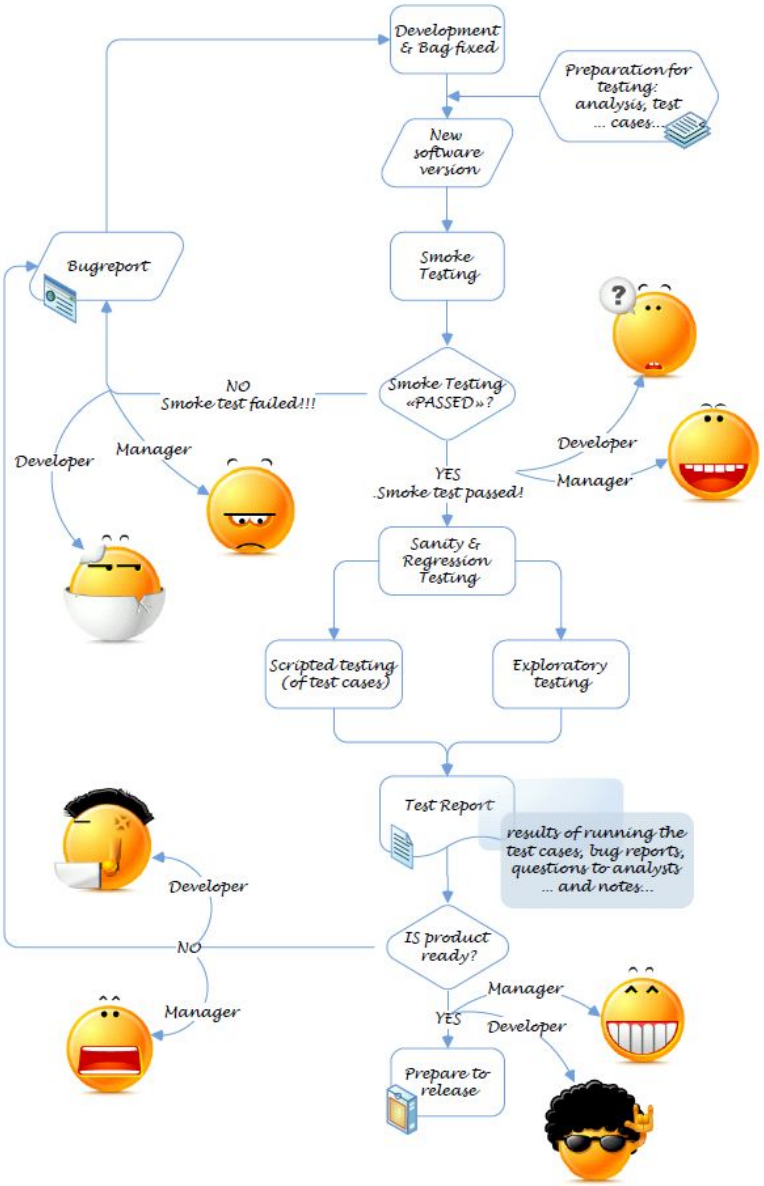
Release (англ.) — выпуск.

Виды релизов:

- **major** release – 1.0
- **minor** release – 1.1
- **patch** release – 1.11

Релиз, как правило, происходит ночью, когда с продуктом работает минимальное количество пользователей.


Идея
v. 1.0
Январь
Дизайн и документация
v. 1.0
Идея
v. 2.0
Февраль
Кодирование
v. 1.0
Дизайн и документация
v. 2.0
Идея
v. 3.0
Март
Тестирование и ремонт
v. 1.0
Кодирование
v. 2.0
Дизайн и документация
v. 3.0
Идея
v. 4.0
Апрель
Релиз
v. 1.0
Тестирование и ремонт
v. 2.0
Кодирование
v. 3.0
Дизайн и документация
v. 4.0
Релиз
v. 2.0
Тестирование и ремонт
v. 3.0
Кодирование
v. 4.0
Релиз
v. 3.0
Тестирование и ремонт
v. 4.0
Релиз
v. 4.0


Development & Bag fixed
Preparation for testing: analysis, test ... cases...
New software version
Smoke Testing
Bugreport
Smoke Testing «PASSED»?
NO
Smoke test failed!!!
Developer
Manager
YES
Smoke test passed!
Developer
Manager
Sanity & Regression Testing
Scripted testing (of test cases)
Exploratory testing
Test Report
results of running the test cases, bug reports, questions to analysts ... and notes...
Developer
NO
IS product ready?
Manager
YES
Developer
Manager
Prepare to release

Освежим:

1. Перечислите стадии цикла разработки ПО.
2. Перечислите болезни спецификаций.
3. Для чего нужно утверждение спецификаций?
4. Для чего нужно замораживание спецификаций?
5. Перечислите причины появления багов кода.
6. Для чего нужно замораживание кода?
7. Что такое тест приемки?
8. Что случается, если тест приемки не пройден?
9. Виды релизов.
10. Какие методологии разработки ПО вам известны?
11. Когда целесообразно начинать тестирование?

Литература:

1. http://www.protesting.ru/
2. http://ru.qahelp.net/
3. http://habrahabr.ru/
4. The Scrum Master Training Manual, v. 1.2., By Nader K. Rad, Frank Turley, Copyright © 2013 Management Plaza.
5. «Тестирование Дот Ком или пособие по жесткому обращению с багами в интернет-стартапах» Р. Савин.
6. Канбан в IT (Kanban Development)., http://habrahabr.ru
7. Rational Unified Process., https://ru.wikipedia.org