

Подготовительный этап в Проектировании информационных систем

Стандарты жизненного цикла ПО

- * ГОСТ 34.601-90
- * ISO/IEC 12207:1995 (российский аналог — ГОСТ Р ИСО/МЭК 12207-99)
- * ГОСТ 19.xxx Единая система программной документации (ЕСПД)

<http://rugost.com/>

ISO/IEC 12207

- * ISO/IEC 12207:1995 «Information technology–Software life cycle processes» с дополнениями и изменениями ISO/IEC 12207:1995/AMD 1:2002 и ISO/IEC 12207:2002/AMD 2:2004 (принят в новой редакции в 2008 году)
- * ISO/IEC 12207:2008 «Systems and software engineering–Software life cycle processes»
- * ISO/IEC TR 15271:1998 Information technology – Guide for ISO/IEC 12207 (Software Life Cycle Processes)
- * ISO/IEC TR 16326:1999 Software engineering – Guide for the application of ISO/IEC 12207 to project management
- * Спецификации ISO/IEC 12207:1995, ISO/IEC TR 15271:1998 и ISO/IEC TR 16326:1999 введены в качестве национальных стандартов РФ

Разработка и проектирование

- * Подготовка программного средства.
- * Анализ требований технического задания.
- * Проектирование архитектуры программного средства.
- * Детальное проектирование программного средства.
- * Конструирование программного средства.
- * Комплексование программного средства.
- * Тестирование.

Подготовка ПС

- * Выбор модели жизненного цикла программного средства, соответствующей области реализации, величине и сложности проекта (если это не указано заказчиком в договоре)
- * Оформление выходных результатов в соответствии с процессом документирования.
- * Оформление возникающих проблем и устранение несоответствий, обнаруженных в программных продуктах и задачах, если таковые применяются при разработке
- * Выбор и адаптация стандартов, методов, инструментарий, языков программирования (если они не установлены в договоре), которые будут использоваться для выполнения работ в процессе разработки и во вспомогательных процессах.
- * Разработка плана проведения работ процесса разработки. Планы должны охватывать конкретные стандарты, методы, инструментарий, действия и обязанности, связанные с разработкой и квалификацией всех требований, включая безопасность и защиту.

Анализ ТЗ

- * Анализ области применения разрабатываемой системы с точки зрения определения требований к ней.
- * Оформление требований к программному средству, которые должны описывать:
 - * функциональные и технические требования, включая производительность, физические характеристики и окружающие условия, под которые должен быть создан программный объект архитектуры (далее - программный объект);
 - * требования к внешним интерфейсам программного объекта архитектуры;
 - * квалификационные требования;
 - * требования безопасности, включая требования, относящиеся к методам эксплуатации и сопровождения, воздействию окружающей среды и травмобезопасности персонала;
 - * требования защиты, включая требования, относящиеся к допустимой точности информации;
 - * эргономические требования, включая требования, относящиеся к ручным операциям, взаимодействию "человек-машина", персоналу и областям, требующим концентрации внимания человека, связанным с чувствительностью объекта к ошибкам человека и обученности персонала;
 - * требования к определению данных и базе данных;
 - * требования по вводу в действие и приемке поставляемого программного продукта на объекте(ах) эксплуатации и сопровождения;
 - * требования к документации пользователя;
 - * требования к эксплуатации объекта пользователем;
 - * требования к обслуживанию пользователя.

- * Оценка ТЗ с учётом следующих критериев (при этом результаты оценок должны быть документально оформлены): учёт потребностей заказчика;
- * соответствие потребностям заказчика;
- * тестируемость;
- * выполнимость проектирования системной архитектуры;
- * возможность эксплуатации и сопровождения.

Проектирование архитектуры программного средства

- * Определение общей архитектуры системы (архитектура верхнего уровня). В архитектуре должны быть указаны объекты технических и программных средств и ручных операций. Должно быть обеспечено распределение всех требований к системе между объектами архитектуры. Затем должны быть определены объекты конфигурации технических и программных средств и ручных операций на основе объектов архитектуры. Должна быть документально оформлена привязка системной архитектуры и требований к системе относительно установленных объектов.
- * Оценка системной архитектуры и требований к объектам архитектуры с учётом следующих критериев (при этом результаты оценок должны быть документально оформлены):
 - * учёт требований к системе;
 - * соответствие требованиям к системе;
 - * соответствие используемых стандартов и методов проектирования;
 - * возможность программных объектов архитектуры выполнять установленные для них требования;
 - * возможности эксплуатации и сопровождения.

Детальное проектирование программного средства

- * Трансформирование требований к программному объекту в архитектуру, которая описывает общую структуру объекта и определяет компоненты программного объекта.
- * Разработка и оформление общего (эскизного) проекта внешних интерфейсов программного объекта и интерфейсов между компонентами объекта.
- * Разработка и оформление общего проекта базы данных.
- * Разработка и оформление предварительной версии документации пользователя.
- * Разработка и оформление предварительных общих требований к тестированию программного объекта и графику сборки программного продукта.
- * Оценка архитектуры программного объекта и эскизные проекты интерфейсов и базы данных по следующим критериям:
 - * учёт требований к программному объекту;
 - * внешняя согласованность с требованиями к программному объекту;
 - * внутренняя согласованность между компонентами программного объекта;
 - * соответствие методов проектирования и используемых стандартов;
 - * возможность технического проектирования;
 - * возможность эксплуатации и сопровождения.

Конструирование программного средства

- * Разработка технического проекта для каждого компонента программного объекта.
- * Разработка технического проекта внешних интерфейсов программного объекта, интерфейсов между компонентами программного объекта и между программными модулями.
- * Разработка технического проекта базы данных.
- * Определение требований к испытаниям и программе испытаний программных модулей.
- * Оценка технического проекта тестирования по следующим критериям:
 - * учёт требований к программному объекту;
 - * внешнее соответствие спроектированной архитектуре;
 - * внутренняя согласованность между компонентами программного объекта и программными модулями;
 - * соответствие методов проектирования и используемых стандартов;
 - * возможность тестирования;
 - * возможность эксплуатации и сопровождения.

Комплекси́рование программного средства

- * Разработка и документальное оформление следующие продукты:
 - * каждый программный модуль и базу данных;
 - * процедуры испытаний (тестирования) и данные для тестирования каждого программного модуля и базы данных.
- * Разработка плана сборки для объединения программных модулей и компонентов в программный объект. План должен включать требования к испытаниям (тестированию), процедуры тестирования, контрольные данные, обязанности исполнителя и программу испытаний. План должен быть документально оформлен.
- * Сбор программных модулей и компонентов.
- * Сбор объектов программной в единую систему вместе с объектами технической конфигурации, ручными операциями и, при необходимости, с другими системами.

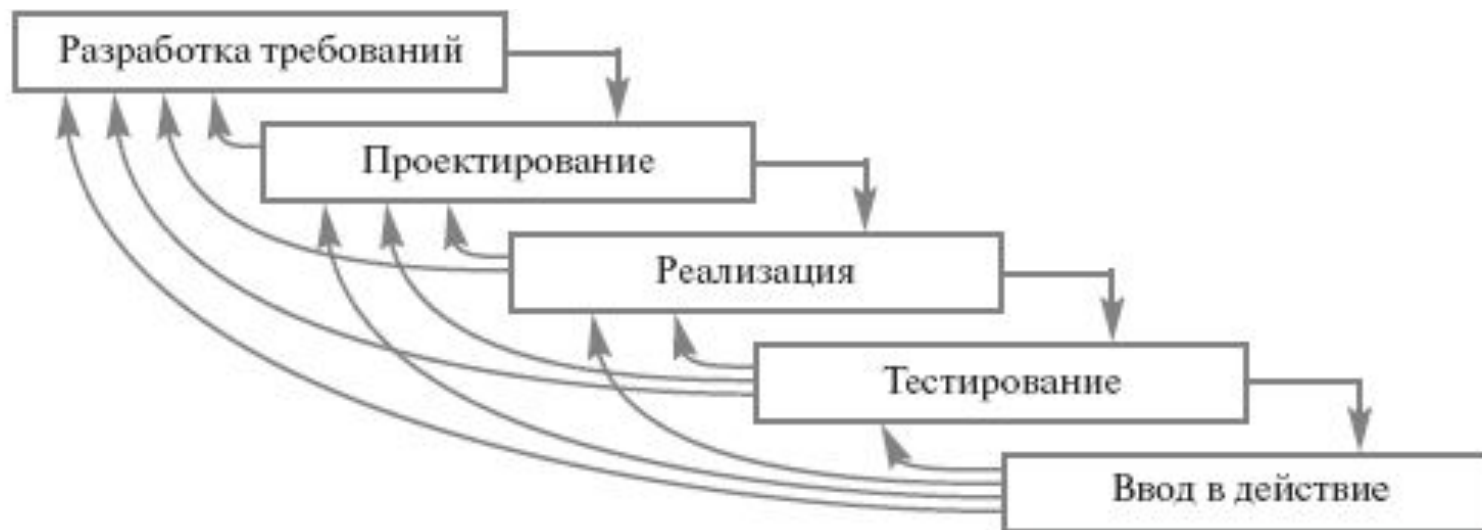
Тестирование

- * Тестирование в соответствии квалификационным требованиям к программному объекту.
- * Оценка проекта, запрограммированного программного объекта, тестирование по следующим критериям:
 - * тестовое покрытие требований к программному объекту;
 - * соответствие ожидаемым результатам;
 - * возможность сборки и тестирования системы (при их проведении);
 - * возможность эксплуатации и сопровождения.
- * Тестирование системы и оценена по следующим критериям:
 - * тестовое покрытие требований к системе;
 - * соответствие ожидаемым результатам;
 - * возможность эксплуатации и сопровождения.
- * Проведение аудиторской проверки и доработка

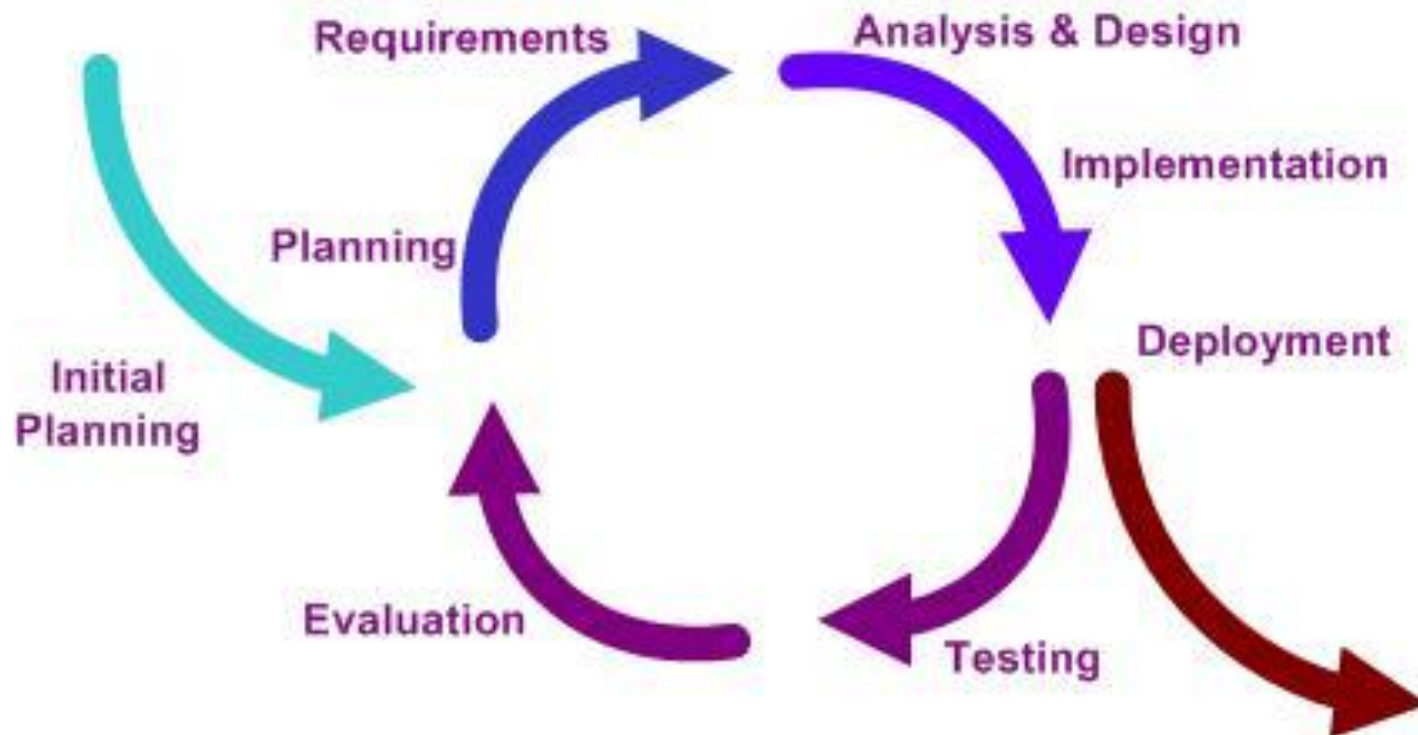
Водопадная (каскадная, последовательная) модель



Поэтапная модель с промежуточным контролем



Итерационная модель

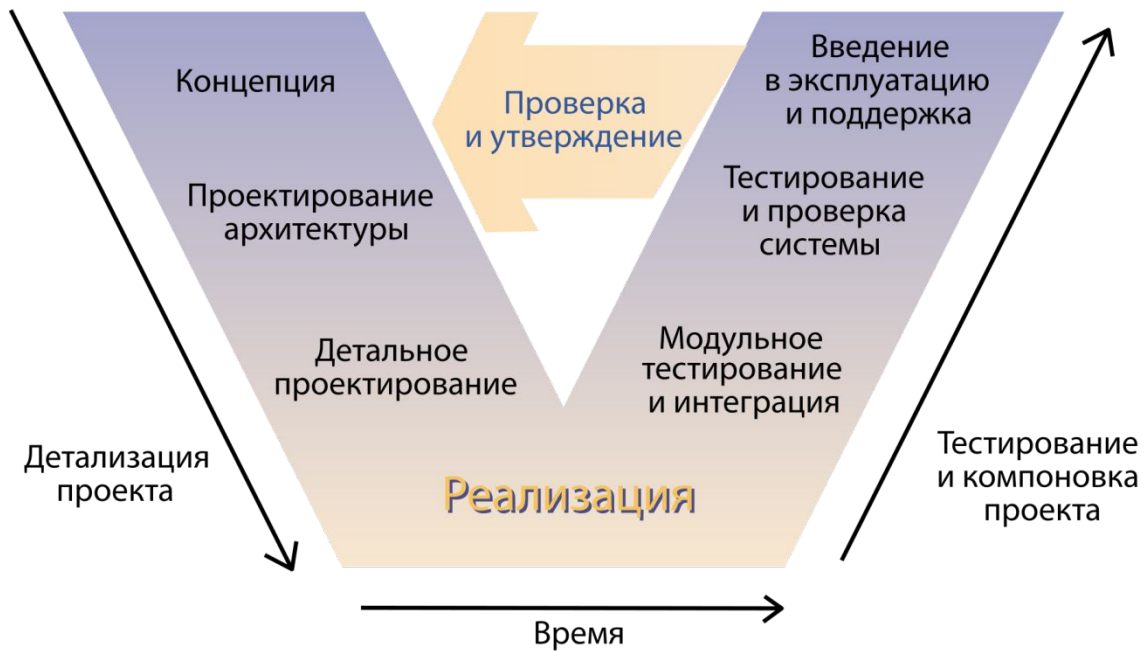


Спиральная модель



- * Concept of Operations (COO) — концепция (использования) системы;
- * Life Cycle Objectives (LCO) — цели и содержание жизненного цикла;
- * Life Cycle Architecture (LCA) — архитектура жизненного цикла; здесь же возможно говорить о готовности концептуальной архитектуры целевой программной системы;
- * Initial Operational Capability (IOC) — первая версия создаваемого продукта, пригодная для опытной эксплуатации;
- * Final Operational Capability (FOC) — готовый продукт, развернутый (установленный и настроенный) для реальной эксплуатации.

V-Model



Гибкая методология разработки

- * Основные идеи:
- * Личности и их взаимодействия важнее, чем процессы и инструменты;
- * Работающее программное обеспечение важнее, чем полная документация;
- * Сотрудничество с заказчиком важнее, чем контрактные обязательства;
- * Реакция на изменения важнее, чем следование плану.

* Принципы:

- * удовлетворение клиента за счёт ранней и бесперебойной поставки ценного программного обеспечения;
- * приветствие изменений требований даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
- * частая поставка рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
- * тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
- * проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
- * рекомендуемый метод передачи информации — личный разговор (лицом к лицу);
- * работающее программное обеспечение — лучший измеритель прогресса;
- * спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
- * постоянное внимание улучшению технического мастерства и удобному дизайну;
- * простота — искусство не делать лишней работы;
- * лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
- * постоянная адаптация к изменяющимся обстоятельствам.

Методологии

- * Agile Modeling
- * Agile Unified Process
- * Agile Data Method
- * DSDM
- * Essential Unified Process
- * Экстремальное программирование
- * Feature driven development (FDD)
- * Getting Real
- * OpenUP
- * Scrum
- * Бережливая разработка программного обеспечения

Экстремальное программирование

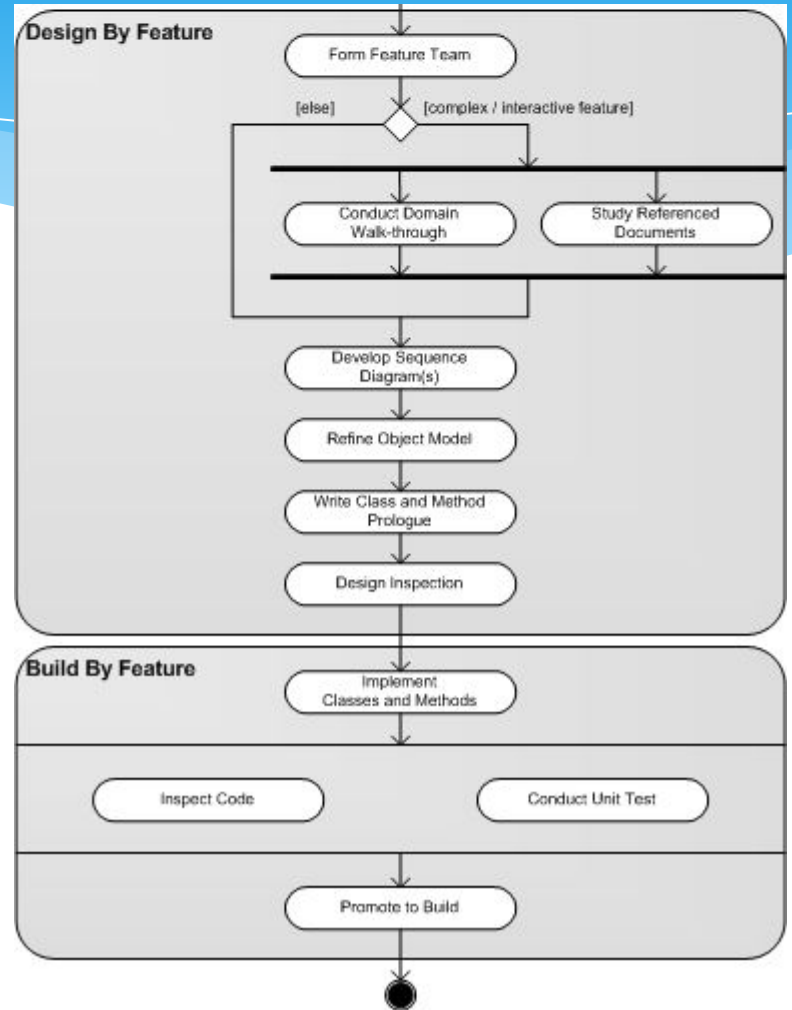
- * Короткий цикл обратной связи (Fine scale feedback)
 - * Разработка через тестирование (Test driven development)
 - * Игра в планирование (Planning game)
 - * Заказчик всегда рядом (Whole team, Onsite customer)
 - * Парное программирование (Pair programming)
- * Непрерывный, а не пакетный процесс
 - * Непрерывная интеграция (Continuous Integration)
 - * Рефакторинг (Design Improvement, Refactor)
 - * Частые небольшие релизы (Small Releases)
- * Понимание, разделяемое всеми
 - * Простота (Simple design)
 - * Метафора системы (System metaphor)
 - * Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
 - * Стандарт кодирования (Coding standard or Coding conventions)
- * Социальная защищенность программиста (Programmer welfare):
 - * 40-часовая рабочая неделя (Sustainable pace, Forty hour week)

Бережливая разработка программного обеспечения

- * Исключение затрат. Затратами считается всё, что не добавляет ценности для потребителя. В частности: излишняя функциональность; ожидание (паузы) в процессе разработки; нечёткие требования; бюрократизация; медленное внутреннее сообщение.
- * Акцент на обучении. Короткие циклы разработки, раннее тестирование, частая обратная связь с заказчиком.
- * Предельно отсроченное принятие решений. Решение следует принимать не на основе предположений и прогнозов, а после открытия существенных фактов.
- * Предельно быстрая доставка заказчику. Короткие итерации.
- * Мотивация команды. Нельзя рассматривать людей исключительно как ресурс. Людям нужно нечто большее, чем просто список заданий.
- * Интегрирование. Передать целостную информацию заказчику. Стремиться к целостной архитектуре. Рефакторинг.
- * Целостное видение. Стандартизация, установление отношений между разработчиками. Разделение разработчиками принципов бережливости. «Мыслить широко, действовать мало, промахиваться быстро; учиться стремительно».

Feature driven development

- * разработка общей модели;
- * составление списка необходимых функций системы;
- * планирование работы над каждой функцией;
- * проектирование функции;
- * реализация функции.



Cleanroom Software Engineering

- * Разработка программного обеспечения основывается на формальных методах.
- * Инкрементальная реализации в рамках статистического контроля качества
- * Статистическое тестирование
- * Формальная верификация

OpenUP

- * Совместная работа с целью согласования интересов и достижения общего понимания;
- * Развитие с целью непрерывного обеспечения обратной связи и совершенствования проекта;
- * Концентрация на архитектурных вопросах на ранних стадиях для минимизации рисков и организации разработки;
- * Выравнивание конкурентных преимуществ для максимизации потребительской ценности для заинтересованных лиц.

RAD

- * Инструментарий должен быть нацелен на минимизацию времени разработки.
- * Создание прототипа для уточнения требований заказчика.
- * Цикличность разработки: каждая новая версия продукта основывается на оценке результата работы предыдущей версии заказчиком.
- * Минимизация времени разработки версии, за счёт переноса уже готовых модулей и добавления функциональности в новую версию.
- * Команда разработчиков должна тесно сотрудничать, каждый участник должен быть готов выполнять несколько обязанностей.
- * Управление проектом должно минимизировать длительность цикла разработки.

Фазы разработки

- * Планирование - совокупность требований, полученных при системном планировании и анализе процедуры разработки жизненного цикла (SDLC). На этом этапе пользователи, менеджеры и IT-специалисты обсуждают задачи проекта, его объём, системные требования, а также сложности, которые могут возникнуть при разработке. Фаза завершается согласованием ключевых моментов с RAD-группой и получением от руководителей проекта разрешения на продолжение.
- * Модель быстрой разработки приложений (RAD)
- * Пользовательское проектирование - на протяжении данного этапа пользователи, взаимодействуя с системными аналитиками, разрабатывают модели и прототипы, которые включают в себя все необходимые системные функции. Для перевода пользовательских прототипов в рабочие модели RAD-группа обычно использует технику объединенной разработки приложений (JAD) и CASE-инструменты. Пользовательское проектирование оказывается длительным интерактивным процессом, который позволяет пользователям понять, изменить и в конечном счете выбрать рабочую модель, отвечающую их требованиям.
- * Конструирование - этап, в котором основная задача заключается в разработке программ и приложений. Аналогична стадии "реализация" в SDLC. В RAD, однако, пользователи продолжают принимать участие и по-прежнему могут предлагать изменения или улучшения в виде разработанных ими докладов. В их задачи входит программирование и разработка приложений, написание кода, интеграция модулей и системное тестирование.
- * Переключение - включает в себя операции по конверсии данных, тестирование, переход на новую систему и тренировку пользователей. По своим задачам напоминает финальную стадию SDLC. Сравнивая с традиционными методами разработки ПО, весь процесс оказывается сжатым по времени. Как результат, новая система оказывается быстрее построенной, доставленной до заказчика и установленной на рабочих местах.

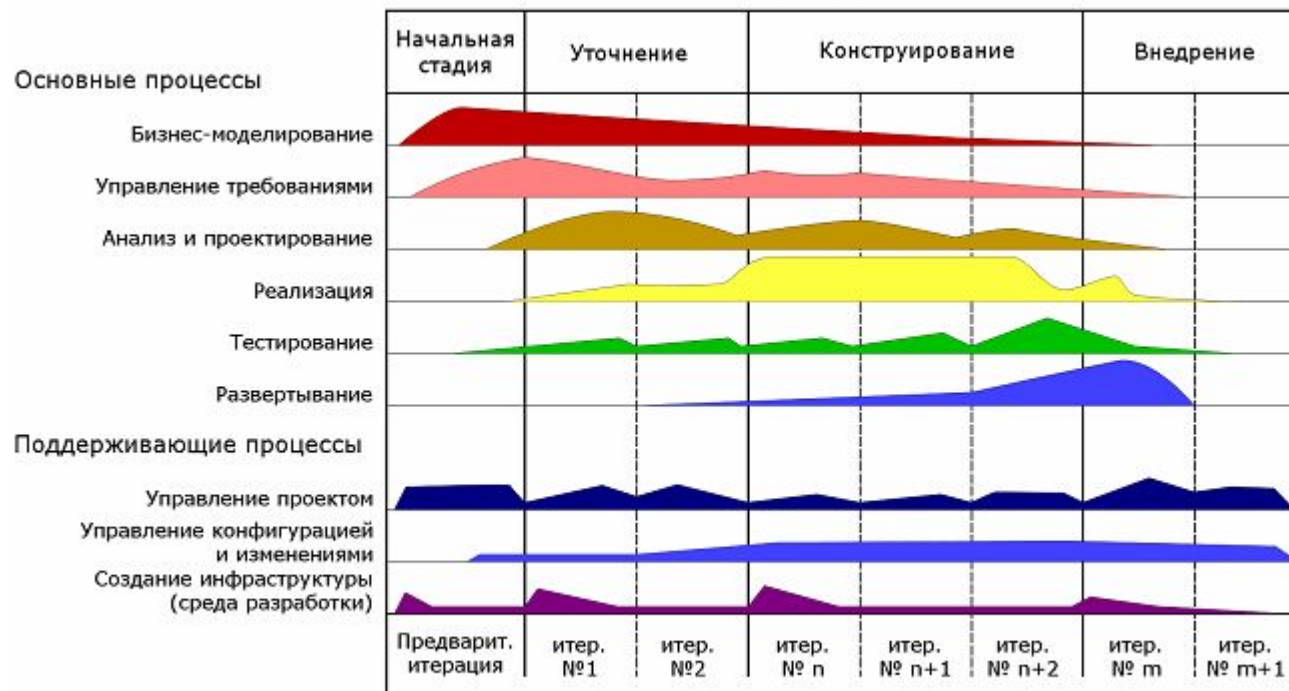


Rational Unified Process

- * Ранняя идентификация и непрерывное (до окончания проекта) устранение основных рисков.
- * Концентрация на выполнении требований заказчиков к исполняемой программе (анализ и построение модели прецедентов (вариантов использования)).
- * Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки.
- * Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта.
- * Постоянное обеспечение качества на всех этапах разработки проекта (продукта).
- * Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

Рабочие процессы

Стадии



Итерации

Microsoft Solutions Framework

- * **модели:**
- * модель проектной группы
- * модель процессов

Принципы

- * Распределение ответственности при фиксации отчетности
- * Наделяйте членов команды полномочиями
- * Концентрируйтесь на бизнес-приоритетах
- * Единое видение проекта
- * Проявляйте гибкость — будьте готовы к переменам
- * Поощряйте свободное общение

Концепции

- * Команда соратников
- * Сфокусированность на нуждах заказчика
- * Нацеленность на конечный результат
- * Установка на отсутствие дефектов
- * Стремление к самосовершенствованию
- * Заинтересованные команды работают эффективно

ролевые кластеры

- * управление программой
- * управление продуктом
- * разработка
- * тестирование
- * управление релизом
- * удовлетворение потребителя

Модель процессов

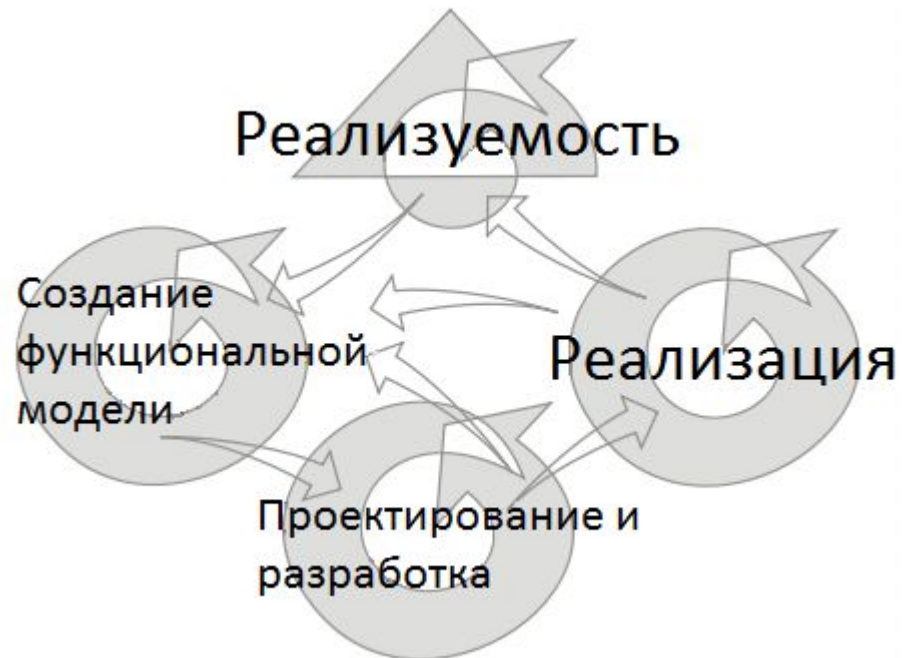
- * Выработка концепции (Envisioning)
- * Планирование (Planning)
- * Разработка (Developing)
- * Стабилизация (Stabilizing)
- * Внедрение (Deploying)



DSDM

- * Вовлечение пользователя - это основа ведения эффективного проекта, где разработчики делят с пользователями рабочее пространство и поэтому принимаемые решения будут более точными.
- * Команда должна быть уполномочена принимать важные для проекта решения без согласования с начальством.
- * Частая поставка версий результата, с учётом такого правила, что «поставить что-то хорошее раньше - это всегда лучше, чем поставить всё идеально сделанное в конце». Анализ поставок версий с предыдущей итерации учитывается на последующей.
- * Главный критерий как можно более быстрая поставка программного обеспечения, которая удовлетворяет текущим потребностям рынка. Но в то же время поставка продукта, который удовлетворяет потребностям рынка, менее важно, чем решение критических проблем в функционале продукта.
- * Разработка - итеративная и инкрементная. Она основывается на обратной связи с пользователем, чтобы достичь оптимального с экономической точки зрения решения.
- * Любые изменения во время разработки - обратимы.
- * Требования устанавливаются на высоком уровне прежде, чем начнётся проект.
- * Тестирование интегрировано в жизненный цикл разработки.
- * Взаимодействие и сотрудничество между всеми участниками необходимо для его эффективности.

- * Исследование реализуемости
- * Исследование экономической целесообразности
- * Создание функциональной модели
- * Проектирование и разработка
- * Реализация



Разработка через тестирование TDD

- * **Добавление теста**
- * **Запуск всех тестов: убедиться, что новые тесты не проходят**
- * **Написать код**
- * **Запуск всех тестов: убедиться, что все тесты проходят**
- * **Рефакторинг**
- * **Повторить цикл**