Организация проектирования информационных систем с использованием гибких методологий

Признаки успешного проекта

- **Реализованы требования пользователя**
- **Проект выполнен в установленный срок**
- **Бюджет проекта не превышен**

Понятие методологии

Методология– последовательностьвыполненияработ, правилвыбораметодовирешенийнаразныхэтапахразработки**.**

Задачи методологии разработки:

Организация ролей (ответственности членов проектной команды), детализация этапов жизненного цикла и процессов, определение активов (артефактов), значимых на разных этапах проекта, практики анализа и предупреждения рисков.

Взаимосвязь стандартов, моделей и методологий

Стандарт	Модель	Методология
Стандарт регламентирует **состав** стадий и процессов жизненного цикла ИС.	Определяет соотношение между процессами и стадиями проектирования.	Организация ролей, детализация этапов жизненного цикла и процессов, определение активов , методов анализа и предупреждения рис*ков*

Как использовать шаблон

Конкретная технология проектирования

Виды методологий конструирования ИС

- *Прогнозирующие (тяжеловесные) методологии*

1.Каноническая.

2.Прототипирование (макетирование).

3.Инкрементная

4.RUP.

5.MSF.

6.DATARUN

7.OracleDesigner

- *Адаптивные методологии (Agile)*

1.Экстремальное программирование (XP)

2.SCRUM

3.Kanban

4.Feature Driven Development(FDD),

5.Dynamic Systems Development Method (DSDM) и др.

Классическая методологии

Основные принципы

- **Строго последовательное выполнение фаз**
  - Каждая последующая фаза начинается лишь тогда, когда полностью завершено выполнение предыдущей фазы
  - Каждая фаза имеет определенные критерии входа и выхода: входные и выходные данные.
  - Каждая фаза полностью документируется
  - Переход от одной фазы к другой осуществляется посредством формального обзора с участием заказчика
- **Основа модели – сформулированные требования (ТЗ), которые меняться не должны**
- **Критерий качества результата – соответствие продукта установленным требованиям.**

преимущества:

- Проста и понятна заказчикам, т.к часто используется другими организациями для отслеживания проектов, не связанных с разработкой ПО
- Проста и удобна в применении:
  - процесс разработки выполняется поэтапно;
  - ее структурой может руководствоваться даже слабо подготовленный в техническом плане или - неопытный персонал;
  - она способствует осуществлению строгого контроля менеджмента проекта;
- Каждую стадию могут выполнять независимые команды (все документировано).
- Позволяет достаточно точно планировать сроки и затраты.

недостатки:

- Попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо проблему или недостаток, приведет к значительному увеличению затрат и сбою в графике.
- Интеграция компонент, на которой обычно выявляется большая часть ошибок, выполняется в конце разработки, что сильно увеличивает стоимость устранения ошибок.
- Запаздывание с получением результатов – если в процессе выполнения проекта требования изменились, то получится устаревший результат

Применимость методологии

• Требования и их реализация максимально четко определены и понятны; используется неизменяемое определение продукта и вполне понятные технические методики. Это задачи типа:

- научно-вычислительного характера (пакеты и библиотеки научных программ типа расчета несущих конструкций зданий, мостов, …);
- операционные системы и компиляторы;
- системы реального времени управления конкретными объектами;

• Повторная разработка типового продукта (автоматизированного бухгалтерского учета, начисления зарплаты, …).

• Выпуск новой версии уже существующего продукта, если вносимые изменения вполне определены и управляемы (перенос уже существующего продукта на новую платформу).

Прототипирование (макетирование)

Под ***прототипом*** понимается действующий программный компонент, реализующий отдельные функции.

Макет может быть реализован как:
- бумажный макет или макет, реализованный на ПК (изображает или рисует человеко – машинный диалог),
- работающий макет (выполняет некоторую часть требуемых функций),
- существует программа, характеристики которой затем должны быть улучшены.

Прототипирование (макетирование)

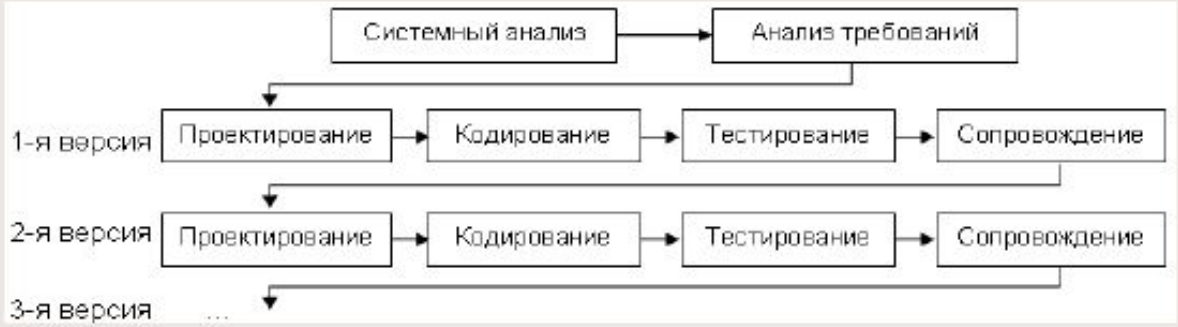
- Применятся, когда имеются не все требования
- Позволяет быстро увидеть некоторые свойства продукта
  - Удобство
  - Внешний вид
  - Применимость
- Часто применятся при проектировании
  - Информационных систем
  - Программных продуктов с ГПИ

Алгоритм работы.

1. Сбор и уточнение требований
2. Быстрое проектирование
3. Построение макета
4. Оценка макета заказчиком
   - Заказчик не удовлетворен
     - Уточнение требований
     - Переход к п. 2
   - Заказчик удовлетворен
     - Переход к п. 5
5. Конструирование продукта

Инкрементная методология

Инкрементная стратегия (англ. increment – увеличение, приращение) подразумевает разработку информационной системы с линейной последовательностью стадий, но в несколько инкрементов (версий), т. е. с запланированным улучшением продукта



- Объединяет классический подход и макетирование
- Весь проект делится на инкременты – версии продукта с определенной функциональностью
- Для каждого инкремента выполняется:
  - Анализ
  - Проектирование
  - Кодирование
  - Тестирование
- Результат каждого инкремента – работающий продукт

Инкрементная методология

Причины разработки версиями:

- отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;
- отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;
- требований поэтапного внедрения и освоения продукта конечными пользователями.

Достоинства:

- Имеется план и график по всем этапам конструирования
- Промежуточные версии доступны заказчику

Недостатки:

- Часто всех требований на начальном этапе нет
- Не всегда можно заранее спланировать содержание версий
- Отсутствует гибкость

RUP Rational Unified Process

ALM (application lifecycle management) - комплексные платформы и методологи управления жизненным циклом приложений.

Rational Unified Process (RUP) - один из самых известных процессов, использующих итеративную модель разработки.

Цель состоит в том, чтобы гарантировать высокое качество программного продукта, отвечающего потребностям конечных пользователей, в пределах предсказуемого временного графика и бюджета.

Создан во второй половине 1990-х годов в компании Rational Software.

- Начало разработки -1995 г.
- Первая версия RUP -1998 г.

Разработчики :

- Филипп Крачтен (Philippe Kruchten),
- Грейди Буч (Grady Booch),
- Джеймс Рамбо (James Rumbaugh)
- Айвар Якобсон (Ivar Jacobson).

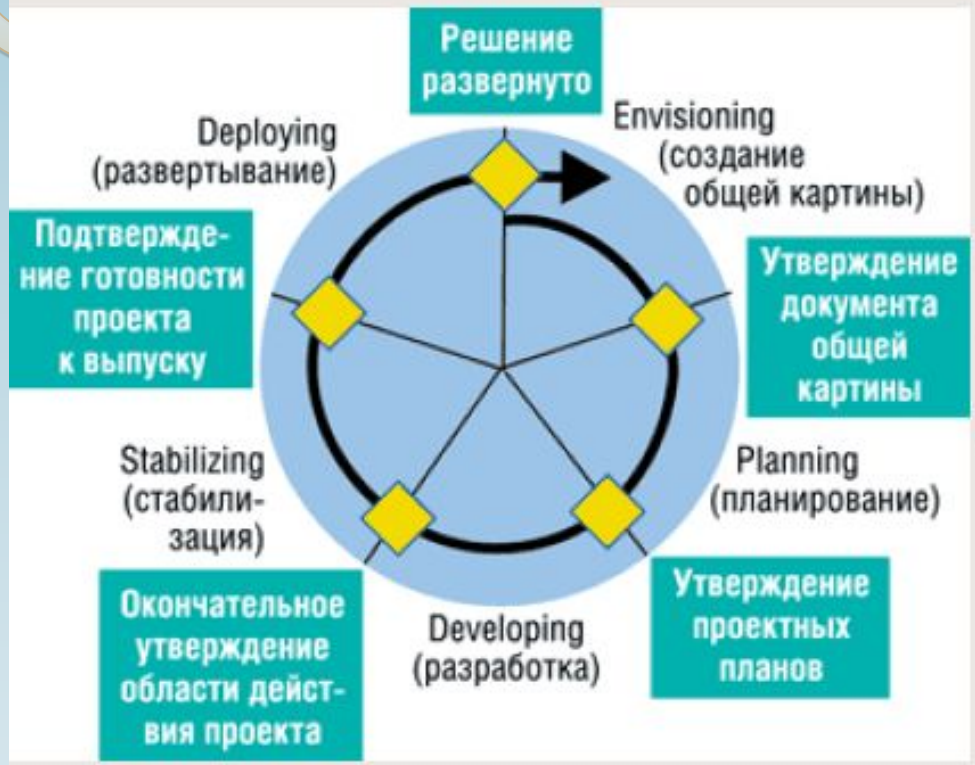
Microsoft Solutions Framework MSF

Microsoft разработал и развивает собственную ALM-методологии (application lifecycle management) под названием Microsoft Solutions Framework (MSF)

Microsoft предоставляет разработчикам унифицированный набор рекомендаций по эффективному проектированию, разработке, развёртыванию, использованию и поддержке решений.

При построении MSF сочетается водопадная и спиральная модели разработки: проект реализуется поэтапно, с наличием соответствующих контрольных точек, а сама последовательность этапов может повторяться по спирали

Этапы методологии MSF



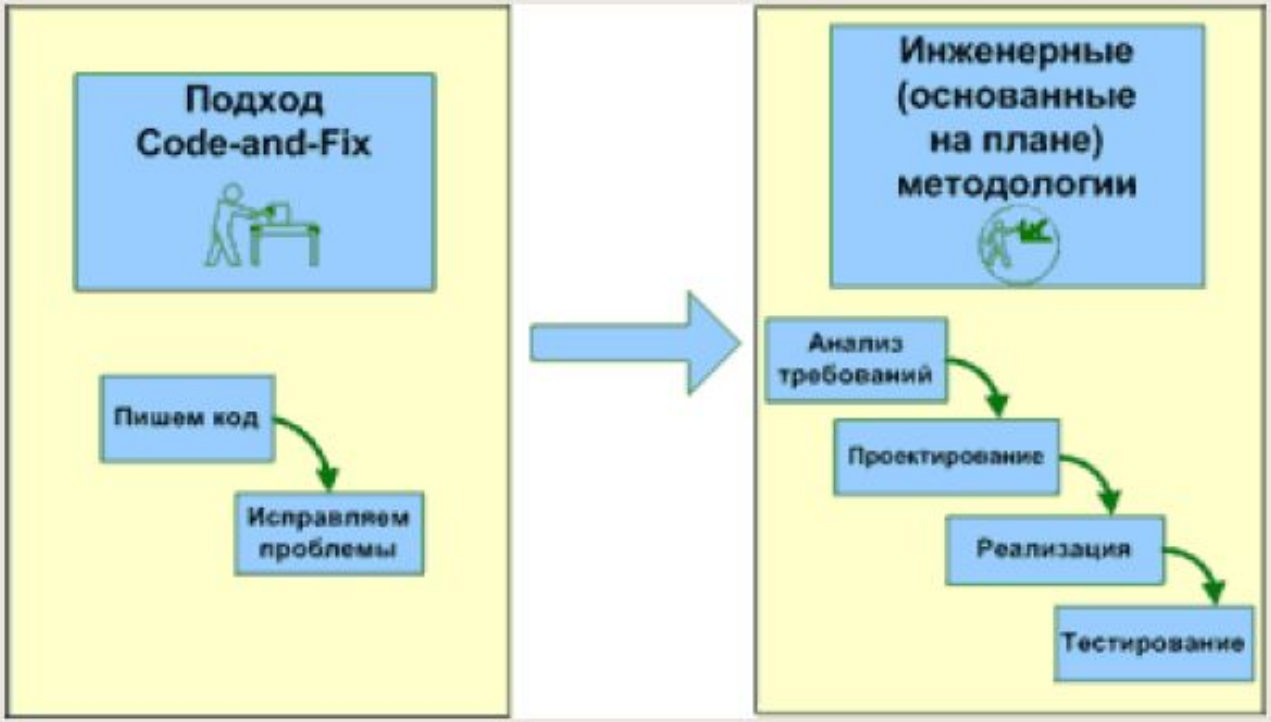
Модель процессов MSF состоит из пяти этапов:

1. создания общей картины приложения;
2. планирования;
3. разработки;
4. стабилизации;
5. развертывания.

Каждый этап завершается контрольной точкой.

Гибкая методология разработки (англ. *Agile software development*), ***agile-методы*** — серия подходов к разработке программного обеспечения, ориентированных на использование **итеративной** разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.

Переход к инженерным методологиям

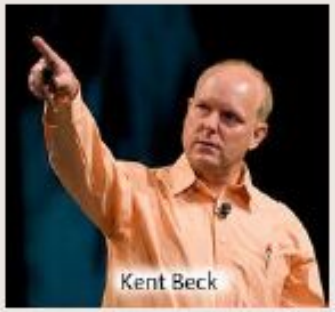


Конфликт уникальности программного обеспечения с подходами инженерных методологий :

- Недооценка роли человеческого фактора.
- Несоответствие инженерного подхода к описанию жизненного цикла проекта природе программного обеспечения.

Активное вхождение Agile в массы началось после подписания Agile Manifesto 11-13 февраля 2001 года на лыжном курорте в штате Юта США. Этот манифест подписали представители таких методологий как :

Scrum, Crystal Clear, Extreme Programming, Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM).



Принципы Agile

- Люди и взаимодействие важнее процессов и инструментов.
- Работающий продукт важнее исчерпывающей документации.
- Сотрудничество с заказчиком важнее согласования условий контракта.
- Готовность к изменениям важнее следования первоначальному плану.

Принципы гибкой разработки программного обеспечения, сформулированные в Agile Манифесте

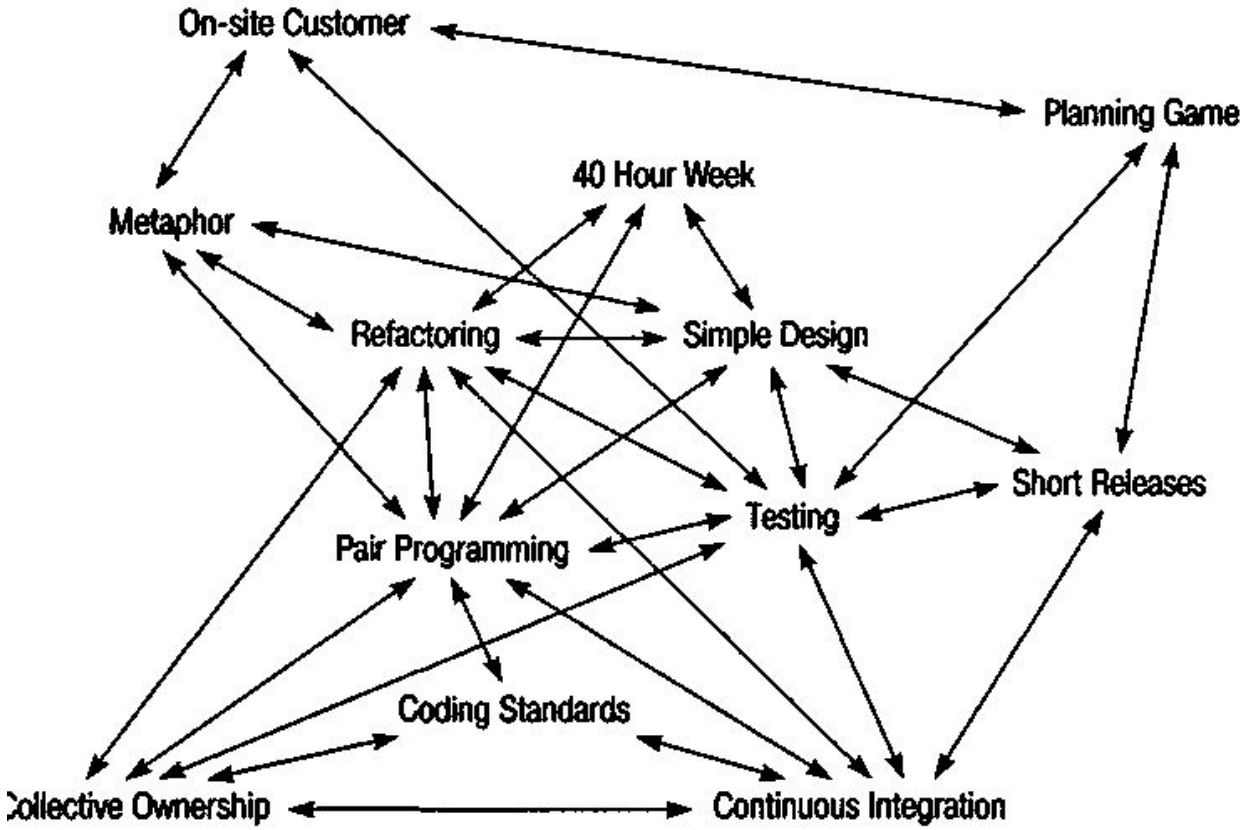
1. Нашим главным приоритетом является удовлетворение заказчика посредством ранней и непрерывной поставки работоспособного программного обеспечения.
2. Приветствуйте меняющиеся требования даже на поздних стадиях разработки. Гибкие процессы используют изменения как средство получить конкурентные преимущества для заказчика.
3. Поставляйте работоспособное программное обеспечение часто: от раза в несколько недель, до раза в несколько месяцев, отдавая предпочтение коротким интервалам.
4. Представители бизнеса и разработчики должны работать вместе в течение всего проекта.
5. Стройте проекты вокруг мотивированных личностей. Предоставьте им среду и поддержку, в которой они нуждаются, и доверьте им самим сделать работу.
6. Наиболее эффективный способ передать информацию в команду проекта (а также передавать её внутри команды) - это непосредственное живое общение.

7.Основной мерой прогресса проекта является работоспособное программное обеспечение.
8.Гибкие процессы поощряют разработку с постоянной скоростью. Спонсоры проекта, разработчики и пользователи должны быть способны поддерживать постоянную скорость на неограниченной дистанции.
9.Непрерывное внимание техническому совершенству и хорошему дизайну увеличивает степень гибкости.
10.Простота - искусство максимизации работы, которую не надо делать, - является существенным фактором.
11.Наилучшие архитектуры, требования и дизайн создаются самоорганизующимися командами.
12.Через регулярные промежутки времени команда должна проводить анализ того, как стать более эффективной и улучшать свой процесс работы.

1. **agilemanifesto.org** - Манифест гибкой разработки программного обеспечения
2. **agilemanifesto.org/principles.html** - принципы гибкой разработки программного обеспечения.
3. **www.ddj.com/architect/184414755** - Статья соавторов Манифеста М. Фоулера и Дж. Хайсмита, содержащая анализ Манифеста и Принципов гибкой разработки.

XP (Extreme Programming)

Экстремальное программирование (англ. Extreme Programming, XP) — одна из гибких методологий разработки программного обеспечения.



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, Refactoring)
- — *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)