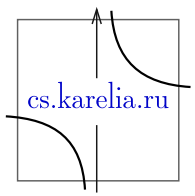
Петрозаводский государственный университет
Институт Математики и информационных технологий
Кафедра Информатики и математического обеспечения

Модели жизненного цикла. Зрелость процесса разработки.

Лектор:

к.т.н., доцент Богоявленский Ю. А.
ybgv@cs.karelia.ru



Содержание

Содержание

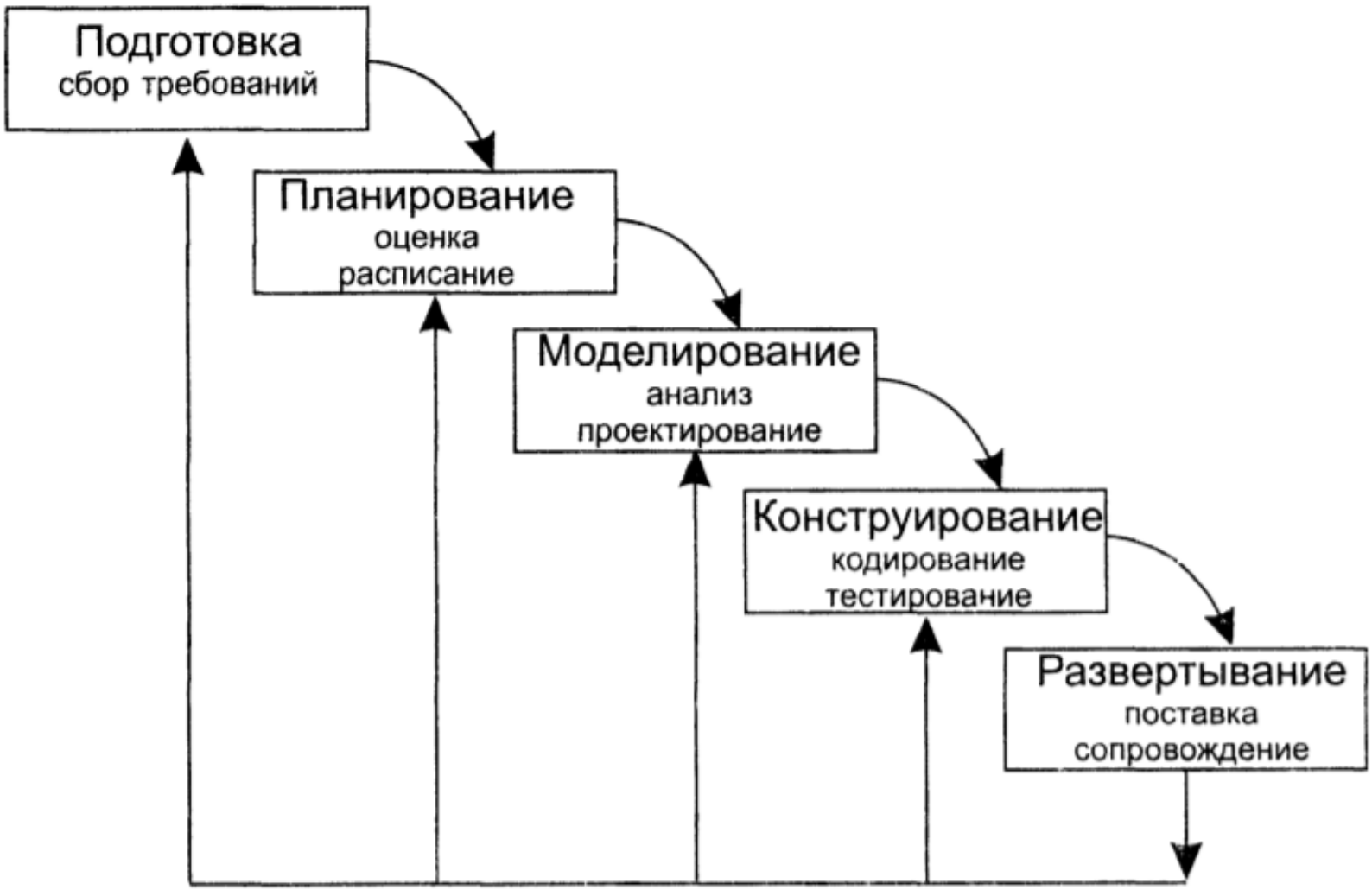
- Два семейства процессов разработки.
- Тяжеловесные и облегченные процессы.
- Обзор agile моделей.
  - eXtreme Programming (XP).
  - Crystal Methodologies.
  - Dynamic Software Development Method (DSDM).
  - Feature Driven Development (FDD).
  - Scrum.
    - ★ Scrum. Общие положения.
    - ★ Подход и ценности Scrum.
    - ★ Комнда Scrum.
    - ★ Спринт Srum.
    - ★ Ежедневные встречи.

Содержание

- Зрелость процесса разработки
  - Модель стандарта ISO 9001:2000.
  - CMM модель.
  - Уровни зрелости CMM.

Модель «классический жизненный цикл»

Старейшая модель процесса разработки ПО — классический жизненный цикл.



Основные этапы: Подготовка, планирование, моделирование

- **Подготовка** (при активное взаимодействии с потенциальным заказчиком:
  - оформление контракта;
  - сбор и формирование требований, определяющих характеристики и функции будущей ПС.
- **Планирование**. Выполняются:
  - определение объема будущих работ и их риск;
  - определение необходимых трудозатрат;
  - формирование рабочих задач и расписание.
- **Моделирование**.Выполняются:
  - анализ требований;
  - проектирование.
  - Результат: Модели, представляемые на графических языках моделирования.

Основные этапы: Моделирование (анализ требований, проектирование)

Анализ требований состоит из:

- обработки набора требований к ПО, сформированного на этапе подготовки;
- уточнения и детализации функций, характеристик и интерфейса ПО.

Все текстовые определения и модели документируются в *спецификации анализа*.

Проектирование включает в себя создание представлений:

- архитектуры ПО;
- структурной и поведенческой организации частей архитектуры ПО;
- входного и выходного интерфейсов частей архитектуры.

Исходные данные для проектирования содержатся в *спецификации анализа*. В ходе проектирования выполняется трансляция требований к ПО во множество проектных представлений.

Основные этапы: Конструирование и Развертывание

Конструирование включает в себя:

- *кодирование*, состоящее в переводе результатов проектирования в текст на языке программирования;
- *тестирование*, представляющее собой выполнение программы для выявления ошибок в реализации программного продукта.

Этап **развертывания** нацелен на *поставку* разработанного продукта заказчику и *сопровождение* процесса эксплуатации этого продукта.

Достоинства и недостатки классической модели

Достоинства:

- дает план и временной график по всем этапам проекта;
- упорядочивает ход разработки.

Недостатки:

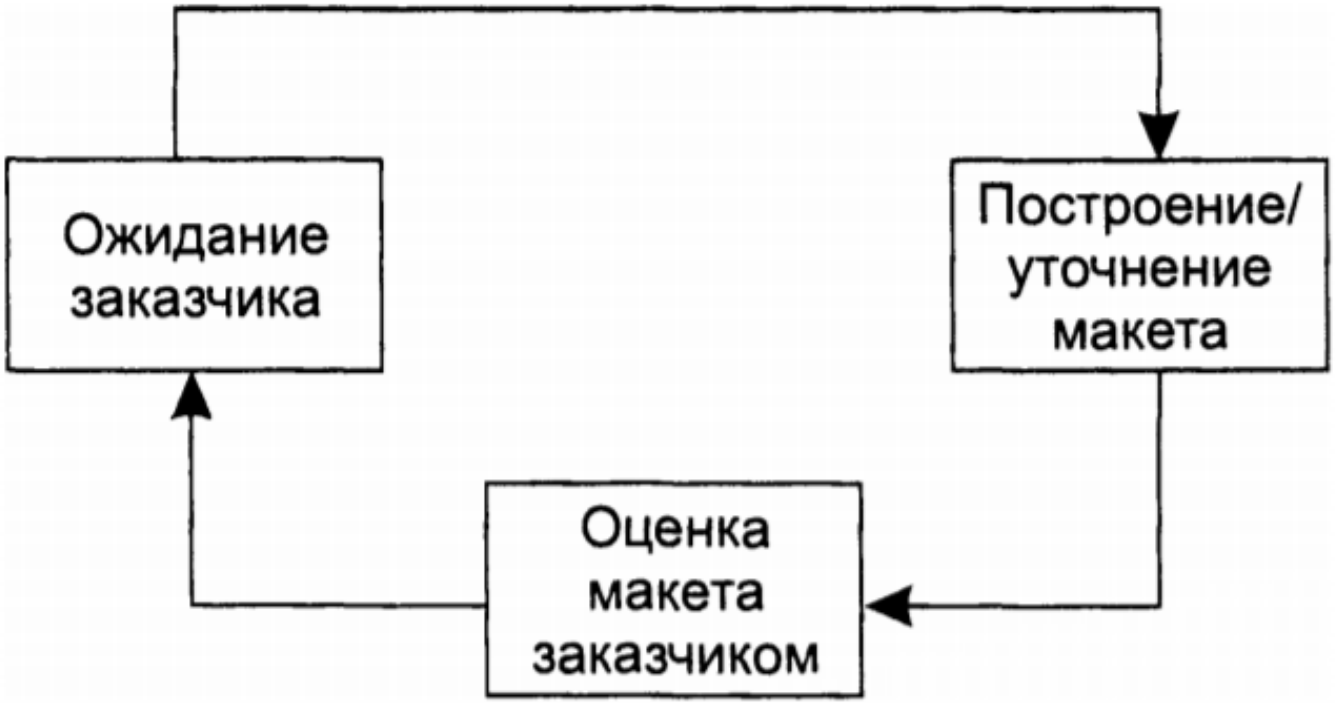
- реальные проекты часто требуют отклонения от стандартной последовательности шагов;
- цикл основан на точной формулировке исходных требований к ПО;
- результаты проекта доступны заказчику только в конце работы.

Макетирование: Определение и итерации

Макетирование (прототипирование) — это процесс создания модели требуемого программного продукта.

Основная цель макетирования: снять неопределенности в требованиях заказчика.

Макетирование основывается на многократном повторении итераций, в которых участвуют заказчик и разработчик.



Макетирование: Последовательность действий

Макетирование: Достоинства и недостатки

Достоинство:

обеспечивает определение полных требований к ПО.

Недостатки:

- заказчик может принять макет за продукт;
- разработчик может принять макет за продукт.

Стратегии разработки ПО

1. Однократный проход (водопадная стратегия) — линейная последовательность этапов разработки
2. Инкрементная стратегия. В начале процесса определяются все пользовательские и системные требования, оставшаяся часть разработки выполняется в виде последовательности версий.
3. Эволюционная стратегия. Система строится в виде последовательности версий, но требования уточняются в результате разработки каждой версий.

Стратегия	Определение требования в начале обработки	Количество циклов разработки	Распростр. промежут. ПО
Однократый подход	Да	Нет	Нет
Инкрементная	Да	Да	Мб
Эволюционная	Нет	Да	Да

Модели разработки ПО. Инкрементная модель

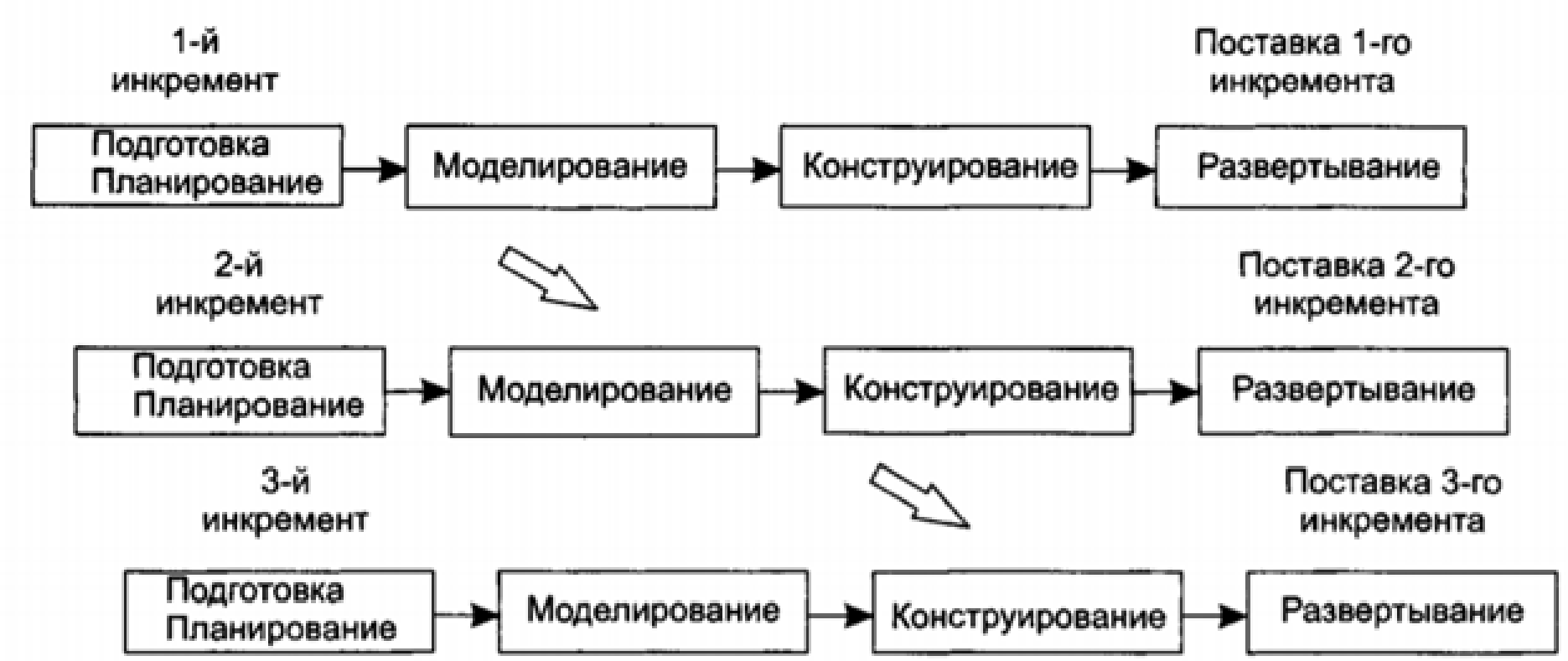
1. Инкрементная модель
2. Спиральная модель
3. Компонентно-ориентированная модель

Инкрементная модель является классическим примером инкрементной стратегии разработки.

Каждая линейная последовательность здесь вырабатывает поставляемый инкремент (версию) ПО.

По своей природе инкрементный процесс итеративен, но в отличие от макетирования инкрементная модель обеспечивает на каждом инкременте работающий продукт.

Инкрементная модель



Первый инкремент приводит к получению базового продукта, реализующего базовые требования. План следующего инкремента предусматривает модификацию базового продукта, обеспечивающего дополнительные характеристики и функциональность.

Спиральная модель

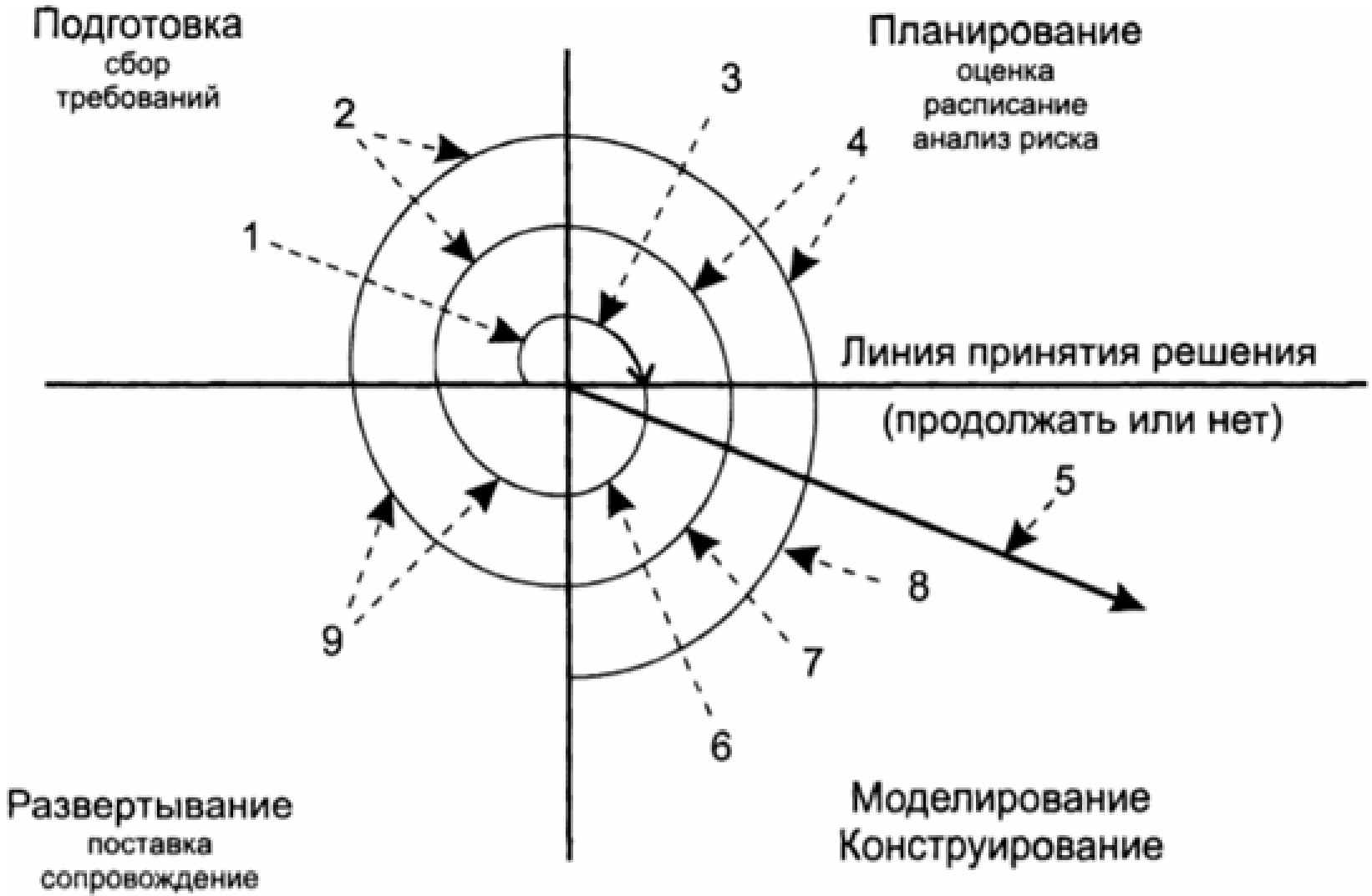
Спиральная модель — классический пример применения эволюционной стратегии разработки.

Спиральная модель (автор Барри Боэм, 1988) базируется на лучших свойствах классического жизненного цикла и макетирования, к которым добавляется новый элемент — анализ риска.

Модель определяет четыре действия:

1. Подготовка — сбор требований и ограничений
2. Планирование — формирование плана проекта и анализ риска
3. Моделирование и конструирование — подготовка моделей и реализация продукта следующего уровня
4. Развертывание — оценка заказчиком текущей версии продукта

Спиральная модель



Достоинства и недостатки спиральной модель

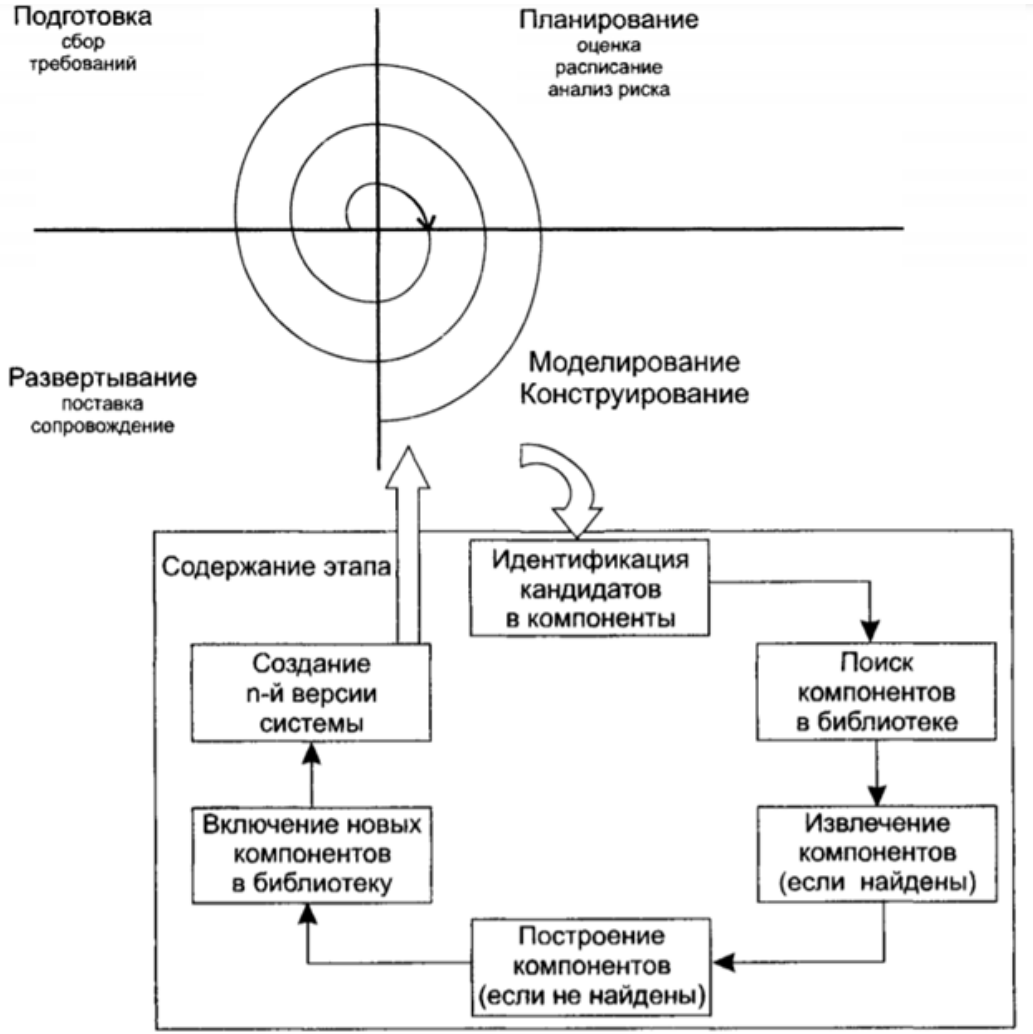
Достоинства спиральной модели:

1. наиболее реально (в виде эволюции) отображает разработку программного обеспечения
2. позволяет явно учитывать риск на каждом витке эволюции разработки
3. включает возможность оценки системы в итерационную структуру разработки
4. использует моделирование для уменьшения риска и совершенствования программного изделия

Недостатки спиральной модели:

1. повышенные требования к заказчику
2. трудности контроля и управления временем разработки

Компонентно-ориентированная модель



Компонентно-ориентированная модель

Это развитием спиральной модели на базе эволюционной стратегии разработки.

Изменено содержание квадранта моделирования-конструирования который включена идентификация подходящих компонентов.

Программные компоненты, созданные в ранее реализованных программных проектах, хранятся в библиотеке. Если компоненты можно использовать в новом проекте, то они подключаются из библиотеки. Разработанные заново компоненеты также включаются в библиотеку.

Достоинства компонентно-ориентированной модели:

1. уменьшает на 30% время разработки программного продукта
2. уменьшает стоимость программной разработки до 70%
3. увеличивает в полтора раза производительность разработки

Два семейства процессов разрааботки

В программной инженерии существует два семейства процессов разработки:

- Семейство прогнозирующих (heavyweight — тяжеловесных) процессов обобщенно называемых водопадными.
- Семейство адаптивных (lightweight — облегченных) процессов.

У каждого семейства есть свои достоинства, недостатки и область применения:

- Прогнозирующий процесс применяют при фиксированных требованиях и многочисленной группе разработчиков разной квалификации.
- Адаптивный процесс используют при частых изменениях требований, малочисленной группе высококвалифицированных разработчиков и грамотном заказчике, который согласен участвовать в разработке.

Тяжеловесные и agile процессы

- Тяжеловесные (heavyweight) процессы.
  Прогнозируется ВЕСЬ объем предстоящих работ (predictive) процесс. Строгий порядок действий разработчика, большой объем документации. Ситаются чрезмерно регулируемыми, планируемыми и микроуправляемыми.
- Облегченные agile процессы.
  В 1990-е годы в противовес тяжеловесным были разработаны:

1991. RAD — Rapid Application Development;

1994. DSDM — Dynamic Systems Development Method;

1995. SCRUM — схватка (в регби);

1996. Crystal Clear;

1996. XP — eXtreme programming (XP);

1997. FDD — feature-driven development.

Хотя они возникли до публикации Agile Manifesto, теперь они все вместе называются agile методами разработки программного обеспечения. Agile — гибкий, проворный, подвижный.

Обзор agile моделей

История Agile начинается с публикации в 2001 году «Манифеста гибкой разработки ПО», состоящего из 12 принципов. Конечно, отдельные положения Agile-подхода появлялись и до этого, но только этот документ систематизировал и изложил их в достаточной для использования мере.

Agile — итеративная модель разработки, в которой программное обеспечение создают инкрементально с самого начала проекта, в отличии от каскадных моделей, где код доставляется в конце рабочего цикла.

Основа гибкой методологии — разбиение проектов на маленькие рабочие кусочки, называемые пользовательскими историями. Согласно приоритетности задачи решают в рамках коротких двухнедельных циклов (итераций).

Обзор agile моделей

12 принципов, которые составляют Agile Methodology, можно поделить на 4 главные идеи:

- Приоритет людей и общения над инструментами и процессами.
- Приоритет работающего продукта над полной документацией.
- Приоритет сотрудничества с заказчиком над утверждением контракта.
- Приоритет у готовности меняться выше, чем у следования первоначально созданному плану.

eXtreme Programming (XP)

Разработчик методики, Кент Бек, создал метод экстремального программирования, цель которого — справиться с постоянно меняющимися требованиями к программному продукту и повысить качество разработки.

Он применим исключительно в сфере разработки ПО, и строится вокруг 4 процессов:

- **Кодирование** — согласно единым в команде стандартам оформления.
- **Тестирование** — тесты пишутся самими программистами до написания кода, который будут тестировать.
- **Планирование** — как финального билда, так и отдельных итераций. Последнее проходит в среднем раз в две недели.
- **Слушание** — как разработчиков, так и клиента, в ходе которого исчезают неясности, определяются требования и ценности.

Crystal Methodologies

Семейство методологий, разработанное Алистером Кокберном, одним из автором «Манифеста гибкой разработки ПО». Классификацию Кокберн предлагает проводить по цветам за критерием количества человек в команде: от 2 (Crystal Clear) до 100 (Crystal Red). Под более масштабные проекты выделены цвета Maroon, Blue и Violet.

3 основных показателя Crystal-проектов:

- **Быстрая доставка рабочего кода** — развитие идеи итеративной модели разработки Agile.
- **Совершенство через рефлексию** — новая версия ПО улучшается на основе данных о предыдущей.
- **«Осмотическое» взаимодействие** — нововведение Алистера, метафора коммуникации и обмена информацией между разработчиками ПО в одной комнате.

Dynamic Software Development Method (DSDM)

Над разработкой DSDM трудился не один человек и даже не команда, а консорциум из 17 английских компаний. DSDM, как и экстремальное программирование, используется преимущественно для создания программного обеспечения.

Особая роль отводится участию конечного потребителя (пользователя) в процессе разработки. Помимо этого принципа, к базовым относятся:

- Главный критерий - как можно более быстрая поставка программного обеспечения, которое удовлетворяет текущим потребностям рынка.
- Автономность разработчиков в плане принятия решений.
- 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.
- Тестирование на протяжении всего рабочего цикла.

Scrum. Общие положения

Scrum — это метод управления проктами по разработке, поставке и поддержк сложных продуктов.

Создателями Скрама являются Кен Швабер и Джефф Сазерленд.

Оласти применения Scrum

- Исследование и выявление жизнеспособных рынков, технологий и возможностей продуктов.
- Разработка продуктов и их улучшения.
- Выпуск продуктов и их обновления по несколько раз в день.
- Разработка и поддержка облачных технологий.
- Поддержка и обновление продуктов.

Суть Scrum небошая командв разработчиков. Каждая отдельная команда чрезвычайно гибка и адаптивна. Эти преимущества проявляются, распространяясь на любое количество команд в организации.

Подход и ценности Scrum

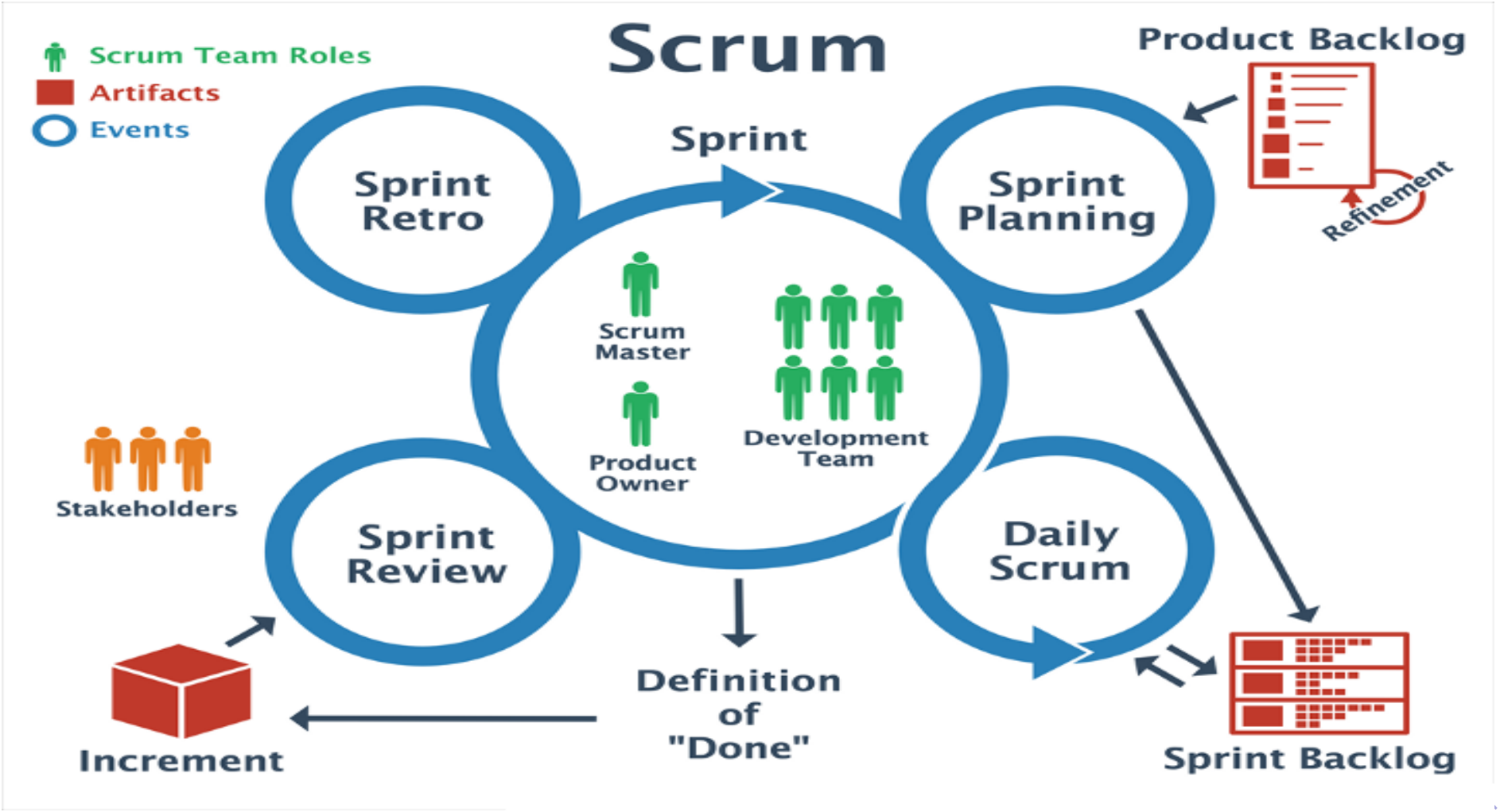
Scrum основан на теории эмпирического управления (эмпиризме). Согласно этой теории, источником знаний является опыт, а источником решений – реальные данные.

- Прозрачность.
- Инспекция.
- Адаптация.
- Преданность.
- Смелость.
- Сфокусированность.
- Открытость.
- Уважение.

Команда Scrum

Спринт Scrum

Спринт — временной отрезок длительностью месяц или меньше, в течение которого создается «готовый», то есть пригодный к использованию и выпуску инкремент продукта.



Спринт Scrum

- По результатам Планировани Спринта Скрам-команда решает:
  - каким будет Инкремент в конце Спринта;
  - как организовать работу, чтобы получить готовый Инкремент Продукта.
- Спринт может быть отменен досрочно только Владельцем Продукта.
- Спринт может быть отменен, если он потерял смысл в контексте сложившихся обстоятельств.

Обзор и ретроспектива спринта

- Обзор Спринта — не статусная встреча, а неформальная. На ней проводится демонстрация инкремента Продукта для получения обратной связи и развития сотрудничества.
- Ретроспектива Спринта — это возможность для команды провести инспекцию, направленную на себя, и создать план улучшений командной работы в следующем Спринте.

Ежедневные встречи — Daily Scrum

Короткие встречи (обычно 15 минут) членов проводятся ежедневно. На встрече руководитель задает всем членам команды три ключевых вопроса:

- Что я сделал вчера, что помогло Команде Разработки приблизиться к Цели Спринта?
- Что я сделаю сегодня, чтобы помочь Команде Разработки достичь Цели Спринта?
- Вижу ли я какие-либо препятствия, которые могут помешать мне или Команде Разработки достичь Цели Спринта?

Руководитель команды (Scrum Master), ведет встречу и оценивает ответы. Scrum-обсуждение способствуют раннему обнаружению потенциальных проблем и распространению индивидуальных знаний на всю команду.

CMM Модель качества процессов разработки

В современных условиях, условиях жесткой конкуренции, очень важно гарантировать высокое качество вашего процесса разработки ПО. Такую гарантию дает сертификат качества процесса, подтверждающий его соответствие принятым международным стандартам. Каждый такой стандарт фиксирует свою модель обеспечения качества.

Модели стандартов:

- ISO 9001:2000
- ISO/IEC 15504
- Capability Maturity Model — CMM

Модель стандарта ISO 9001:2000

Модель стандарта ISO 9001:2000 ориентирована на процессы разработки из любых областей человеческой деятельности. Стандарт ISO/IEC 15504 специализируется на процессах программной разработки и отличается более высоким уровнем детализации.

CMM модель

Базовым понятием модели CMM считается зрелость компании. Незрелой называют компанию, где процесс разработки ПО и принимаемые решения зависят только от таланта конкретных разработчиков. Как следствие, здесь высока вероятность превышения бюджета или срыва сроков окончания проекта.

Напротив, в зрелой компании работают ясные процедуры управления проектами и построения программных продуктов. По мере необходимости эти процедуры уточняются и развиваются. Оценки длительности и затрат разработки точны, основываются на накопленном опыте.

CMM модель

Кроме того, в компании имеются и действуют корпоративные стандарты на процессы взаимодействия с заказчиком, процессы анализа, проектирования, программирования, тестирования и внедрения программных продуктов.

Модель CMM фиксирует критерии для оценки зрелости компании и предлагает рекомендации для улучшения существующих в ней процессов.

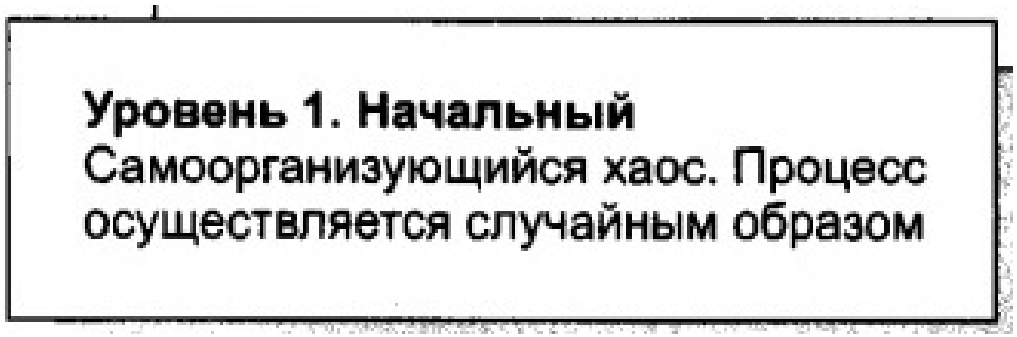
Зрелость компании описывается CMM уровнями, каждый из которых характеризуется областью ключевых процессов (ОКП), причем считается, что каждый последующий уровень включает в себя все характеристики предыдущих уровней.

Наример, уровень 3 подразумевает, что процессы компании характеризуются ОКП уровней 1, 2 и 3 и при совместном выполнении приводят к достижению определенного набора целей.

Уровни зрелости CMM

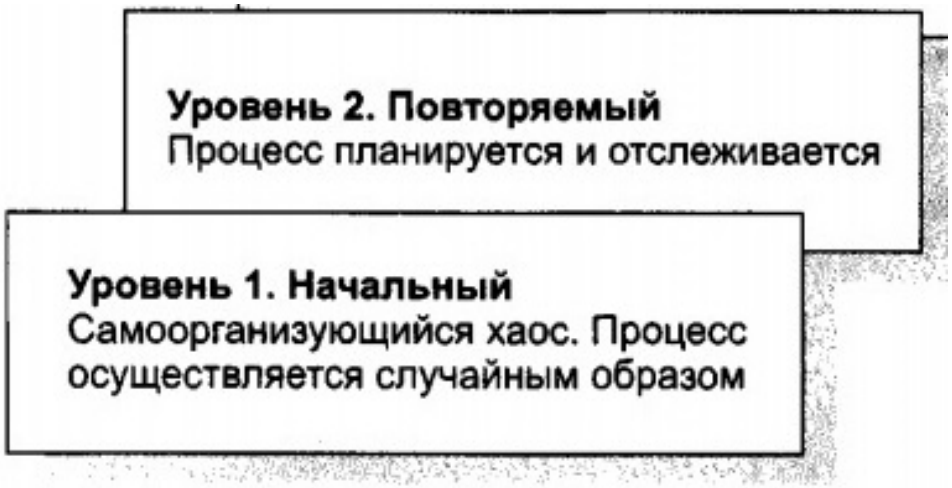
Уровни зрелости CMM

Начальный уровень (уровень 1) означает, что процесс в компании не формализован. Он не может строго планироваться и отслеживаться, его успех носит случайный характер. Результат работы целиком и полностью зависит от личных качеств отдельных сотрудников. При увольнении таких сотрудников проект останавливается.



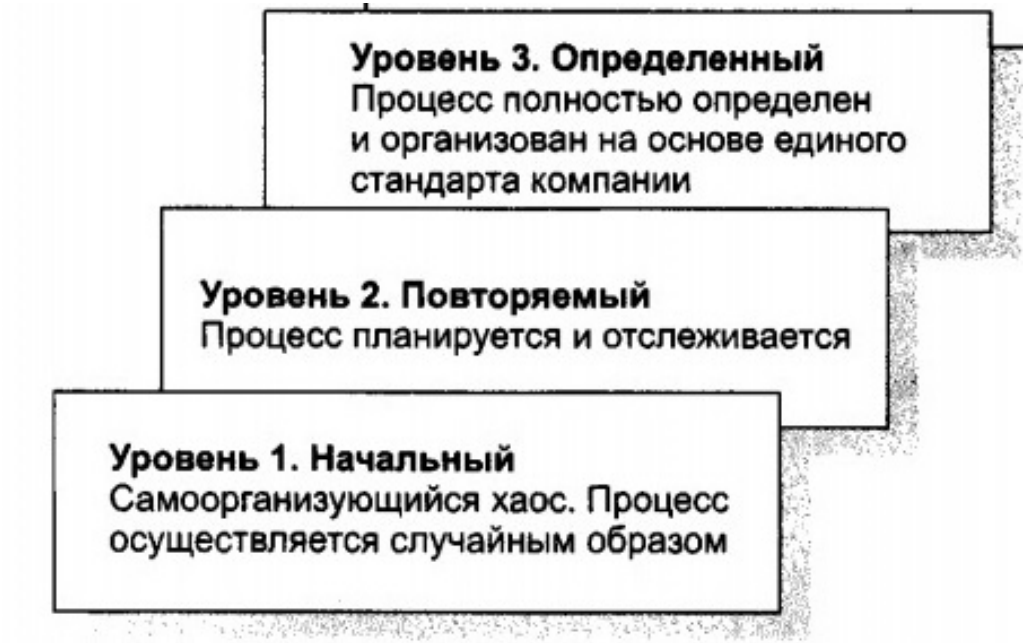
Уровни зрелости CMM

Для перехода на повторяемый уровень (уровень 2) необходимо внедрить формальные процедуры для выполнения основных элементов процесса разработки. Результаты выполнения процесса соответствуют заданным требованиям и стандартам. Основное отличие от уровня 1 состоит в том, что выполнение процесса планируется и контролируется. Применяемые средства планирования и управления дают возможность повторения ранее достигнутых успехов.



Уровни зрелости CMM

Следующий, определенный уровень (уровень 3) требует, чтобы все элементы процесса были определены, стандартизованы и задокументированы. Основное отличие от уровня 2 заключается в том, что элементы процесса уровня 3 планируются и управляются на основе единого стандарта компании. Качество разрабатываемого ПО уже не зависит от способностей отдельных личностей.



Уровни зрелости CMM

С переходом на управляемый уровень (уровень 4) в компании принимаются количественные показатели качества как программных продуктов, так и процесса. Это обеспечивает более точное планирование проекта и контроль качества его результатов. Основное отличие от уровня 3 состоит в более объективной, количественной оценке продукта и процесса

Уровни зрелости CMM

Уровень 5 подразумевает, что главной задачей компании становится постоянное улучшение и повышение эффективности существующих процессов, ввод новых технологий.

Основное отличие от уровня 4 — планомерное и последовательное совершенствование технологий разработки..

ОКП 5-го уровня характеризуется процессами:

- предотвращения дефектов
- управления изменениями технологии
- управления изменениями процесса

Спасибо за внимание