Тестирование ПО

Лекция 2. Тестирование и жизненный цикл ПО. CMM

ШТАНЮК А.А., 2019

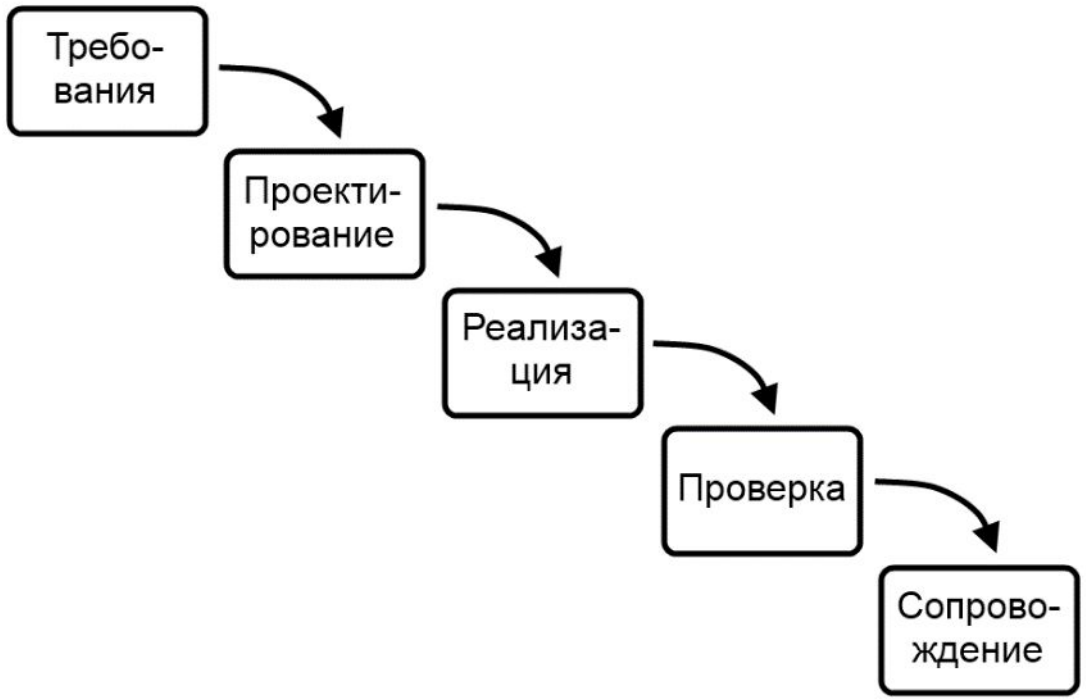
Процесс разработки ПО

Процесс разработки программного обеспечения: набор дисциплин в виде задач/подпроцессов + правила переходов

Примеры подпроцессов:

- Анализ рынка
- Анализ требований
- Бизнес-моделирование
- Планирование продукта
- Разработка архитектуры
- Кодирование
- Тестирование и отладка
- Документирование
- Внедрение
- Сопровождение

Каскадная модель



Каскадная модель

Модель базируется на том, что человек никогда не ошибается

Является идеалом, который, к сожалению, недостижим

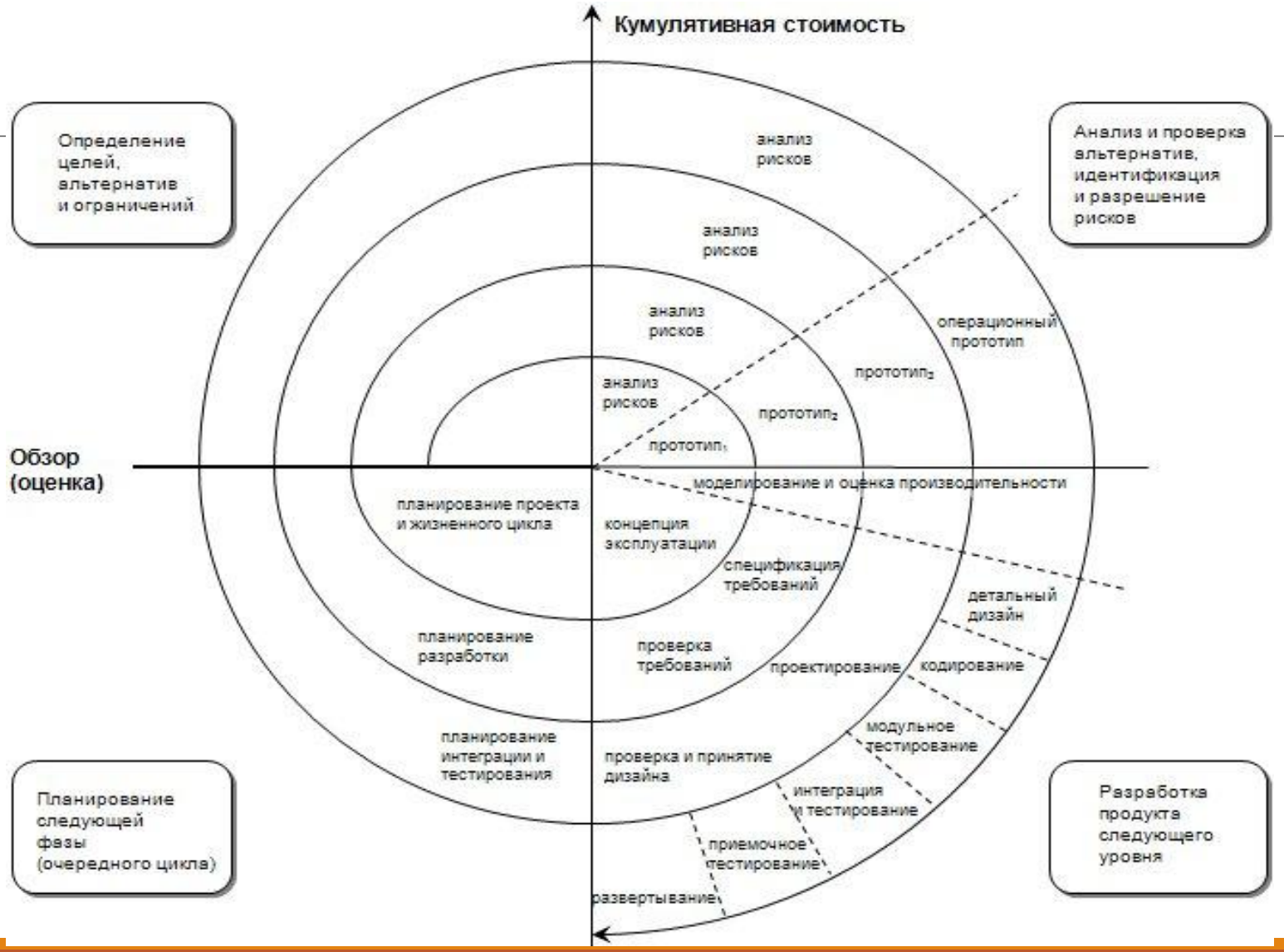
Следуя каскадной модели, разработчик переходит от одной стадии к другой строго последовательно

Следуя каскадной модели, разработчик переходит от одной стадии к другой строго последовательно

Спустя непродолжительное время после своего появления на свет каскадная модель была доработана Уинстоном Ройсом с учетом взаимозависимости этапов и необходимости возврата на предыдущие ступени, что может быть вызвано, например, неполнотой требований или ошибками в формировании задания. В таком "обратимом" виде (модель водоворота) эта модель просуществовала долгое время и явилась основой для многих проектов

Каскадная модель с обратной связью

Спиральная модель



Спиральная модель

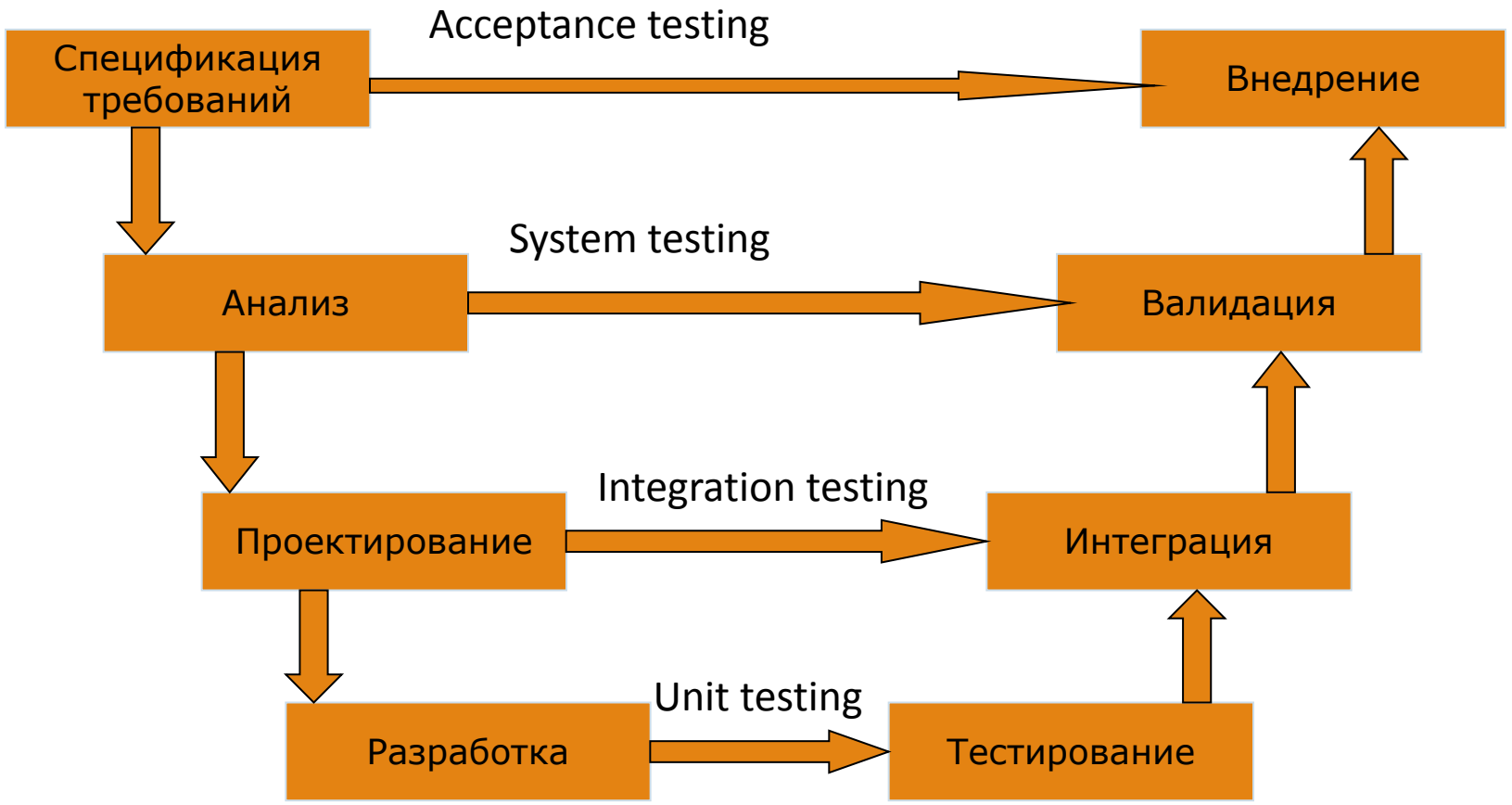
- Спиральная модель Боэма (1988) сфокусирована на проектировании. Собственно разработка ПО происходит лишь на последнем витке спирали по обычной каскадной модели, однако этому предшествует несколько итераций проектирования на основе создания прототипов – при этом каждая итерация включает стадию выявления и анализа рисков и наиболее сложных задач.

- Поскольку спиральная модель в основном охватывает именно проектирование, то в первоначальном виде она не получила широкого распространения в качестве метода управления всем жизненным циклом создания ПО. Однако главная ее идея, заключающаяся в том, что процесс работы над проектом может состоять из циклов, проходящих одни и те же этапы, послужила исходным пунктом для дальнейших исследований и стала основой большинства современных моделей процесса разработки ПО.

Современные модели

- V-model
- Инкрементная модель
- Итерационная модель

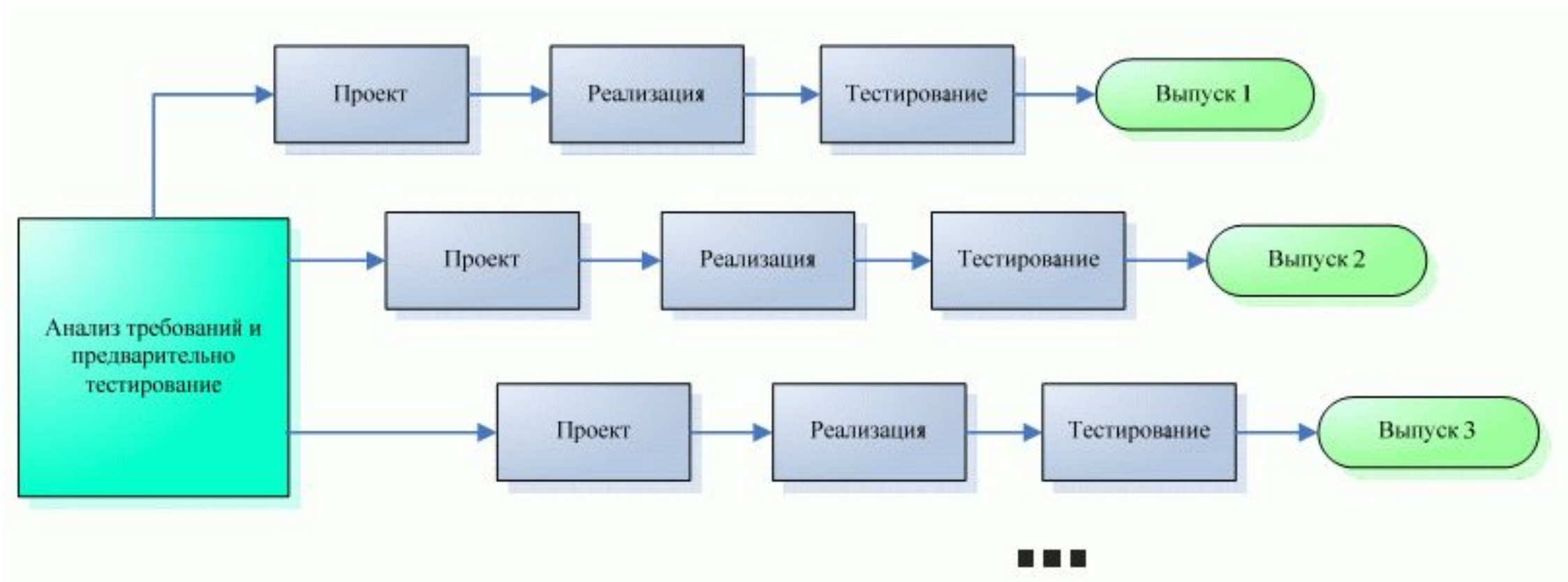
V-модель


Acceptance testing
Спецификация требований
Внедрение
System testing
Анализ
Валидация
Integration testing
Проектирование
Интеграция
Unit testing
Разработка
Тестирование

Инкрементная модель

- **Инкрементная разработка** представляет собой процесс частичной реализации всей системы и медленного наращивания функциональных возможностей.

Инкрементная модель

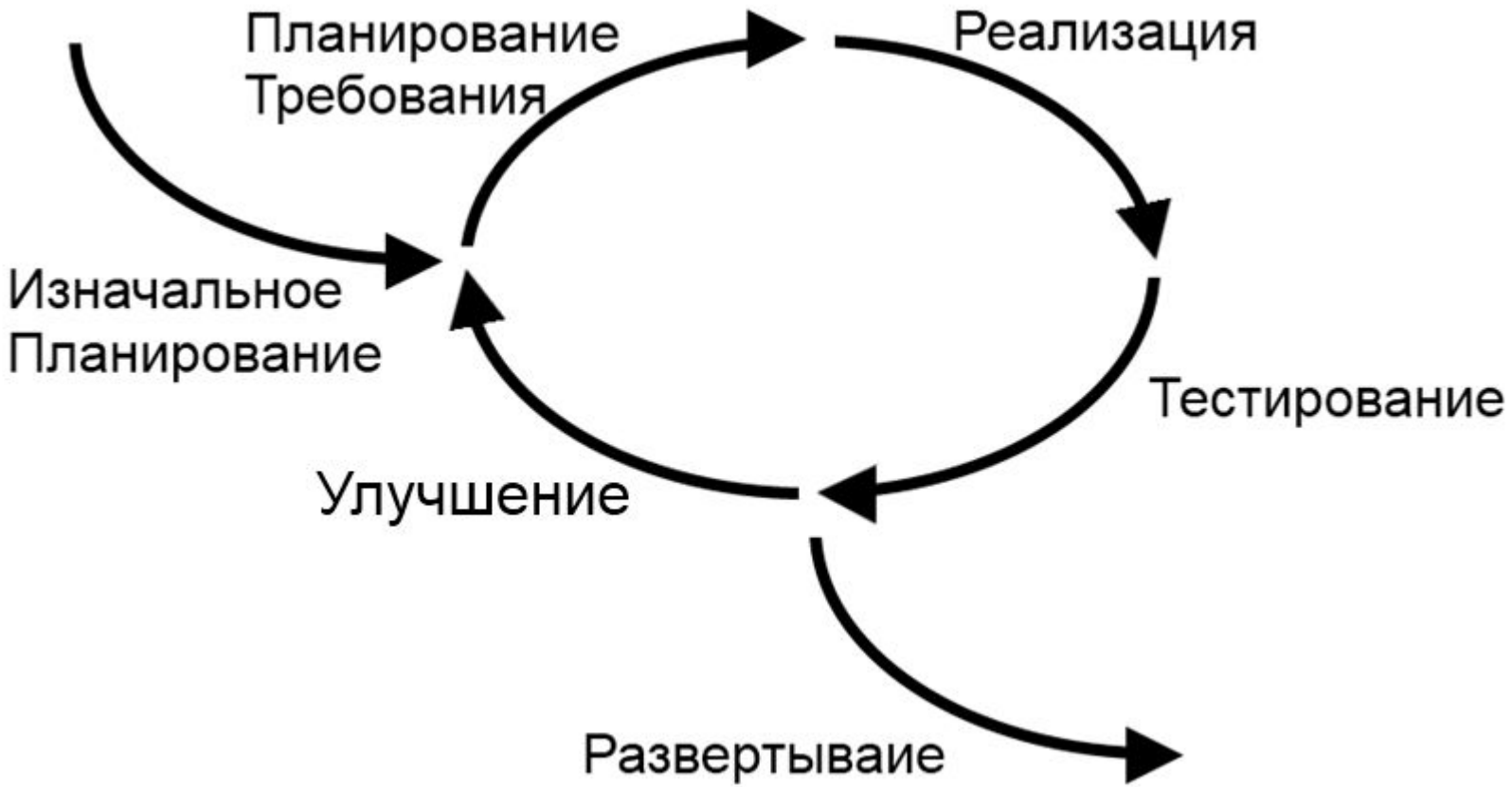


Итеративная модель

Сложность приложений и их объем повысились еще на несколько порядков, так же, как и стоимость разработки. А одной из основных тенденций стала не просто разработка качественного продукта, а возврат инвестиций от него.

Более того, практика показала ограниченность применявшихся ранее инкрементальной и спиральной моделей, и на смену им появилась и была почти повсеместно принята итеративная модель разработки ПО. Стоит отметить, что большинство успешных проектов было создано именно на ее основе.

Итеративная модель



Итеративная модель

Впервые предложенная Филиппом Крачтеном в 1995 г., данная модель объединяет главные преимущества спиральной, инкрементной, каскадной моделей, а также методов разработки на основе создания прототипов и объектно-ориентированного подхода. Она завоевала большую популярность и в том или ином виде используется во многих современных проектах.

На каждой фазе проект проходит множество итераций, приводящих к созданию работоспособных версий: на начальных создаются прототипы, уточняются требования, прорабатываются наиболее сложные проблемы; конечные приводят к созданию продукта, его совершенствованию и расширению функциональности.

Итеративная модель

Преимущества итеративного подхода:

- снижение воздействия серьёзных 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;

Итеративная модель

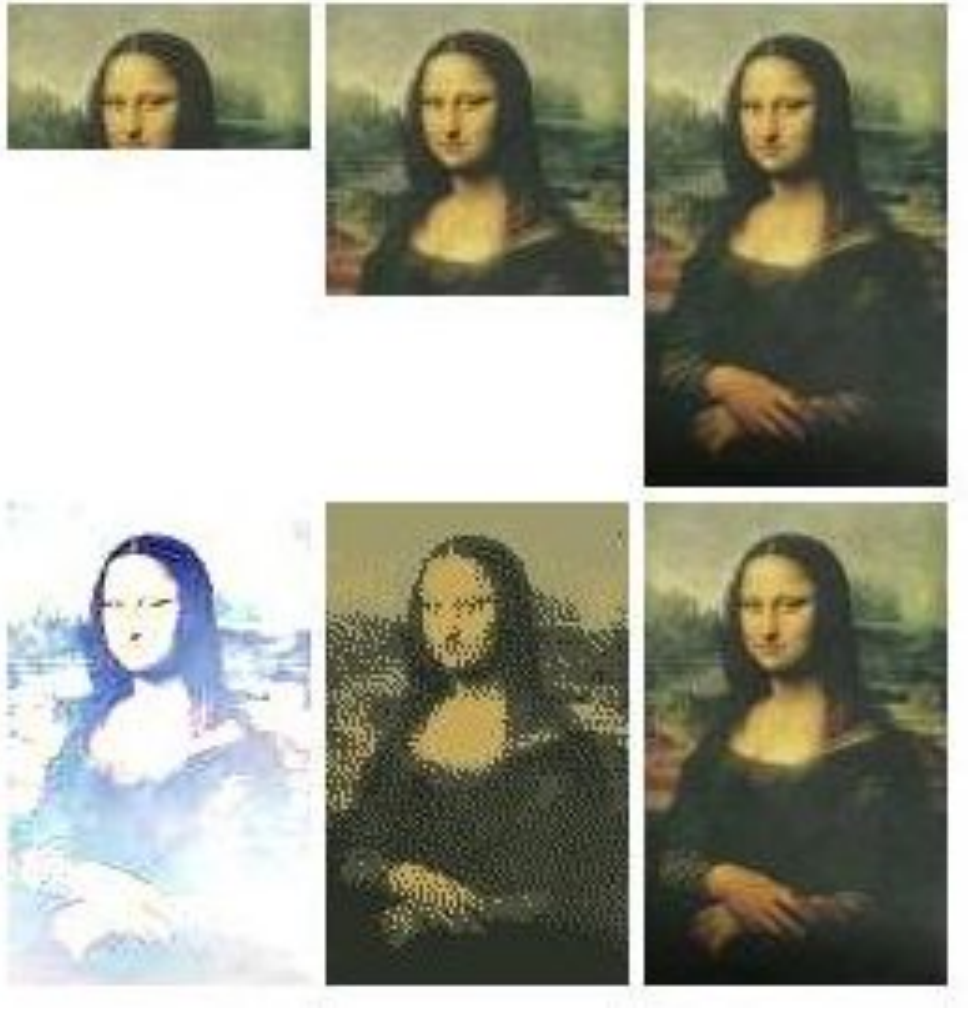
Преимущества итеративного подхода (продолжение):

- 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.
- затраты распределяются по всему проекту, а не группируются в его конце

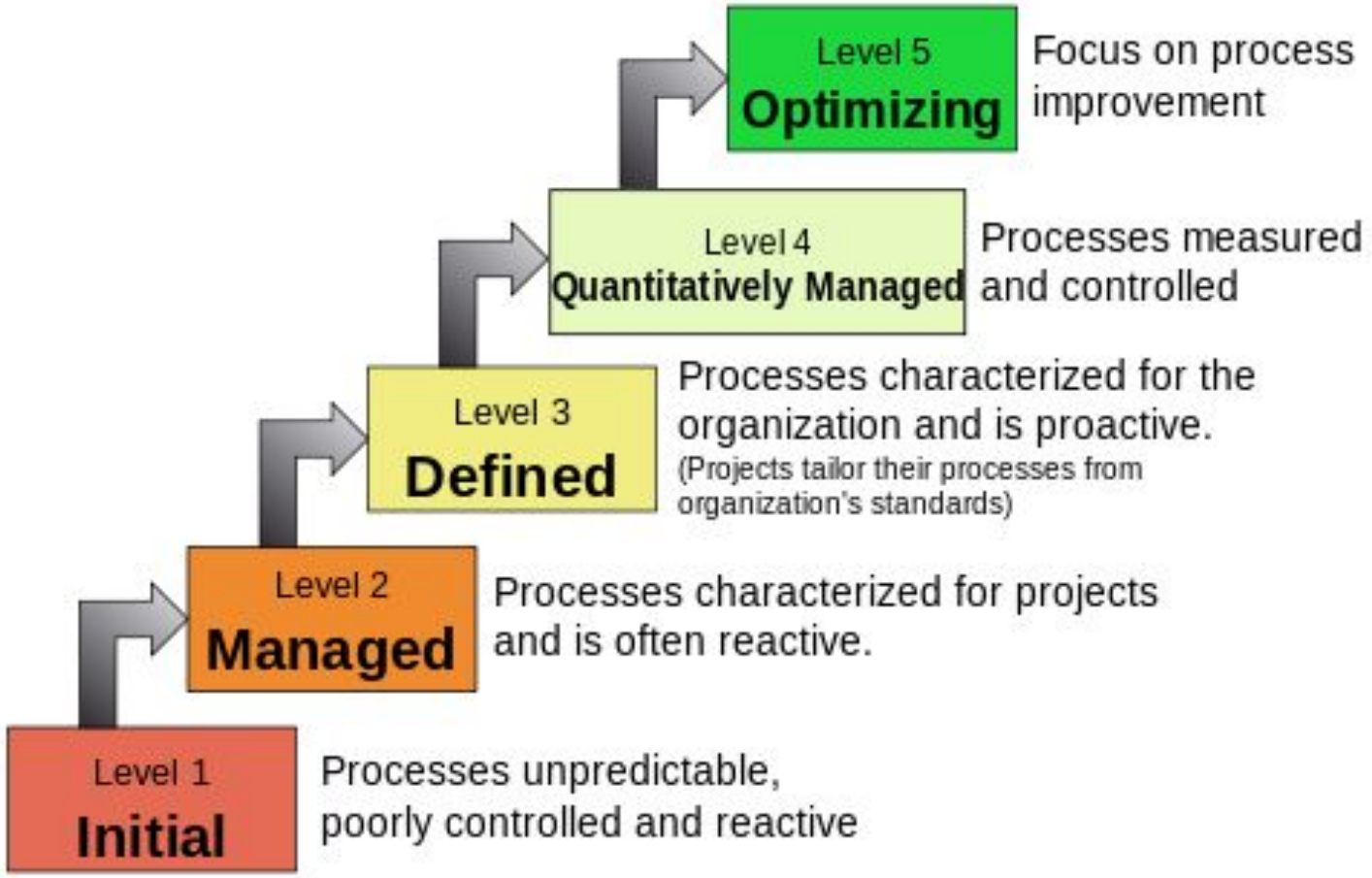
.

Различия

- Инкрементная модель
- Итеративная модель



Модель CMM (ISO 90003:2004 и ISO/IEC 15504)



Модель CMM

Начальный. Самый примитивный статус организации. Организация способна разрабатывать ПО. Организация не имеет явно осознанного процесса, и качество продукта целиком определяется индивидуальными способностями разработчиков. Один проявляет инициативу, и команда следует его указаниям. Успех одного проекта не гарантирует успех другого. При завершении проекта не фиксируются данные о трудозатратах, расписании и качестве

Модель CMM

Повторяемый. В некоторой степени отслеживается процесс. Делаются записи о трудозатратах и планах. Функциональность каждого проекта описана в письменной форме. В середине 99 лишь 20 % организаций имели 2-й уровень или выше.

Установленный. Имеют определённый, документированный и установленный процесс работы, не зависящий от отдельных личностей. То есть вводятся согласованные профессиональные стандарты, а разработчики их выполняют. Такие организации в состоянии достаточно надёжно предсказывать затраты на проекты, аналогичные выполненным ранее

Модель CMM

Управляемый. Могут точно предсказать сроки и стоимость работ. Есть база данных накопленных измерений. Но нет изменений при появления новых технологий и парадигм

Оптимизированный. Есть постоянно действующая процедура поиска и освоения новых и улучшенных методов и инструментов

Современные методики

- Continuous Integration
- Continuous Testing
- DevOps

Continuous Integration

Цель: уменьшить время жизни бага в коде

- Раннее обнаружениие: per commit testing
- Быстрое исправление ~~Как-нть потом поправим~~

Простая организация и представление

- Статус всех стадий «в одном окне»
- Автоматический запуск шагов по цепочке (триггеры)

Continuous Testing

Цель: проверка каждого изменения на стабильность продукта

Быстрые циклы тестирования

- Автоматические тесты
- Автономная инфраструктура
- Дифференциация наборов: unit, pre- & per-commit, daily …

Процент прошедших тестов (pass rate)

DevOps

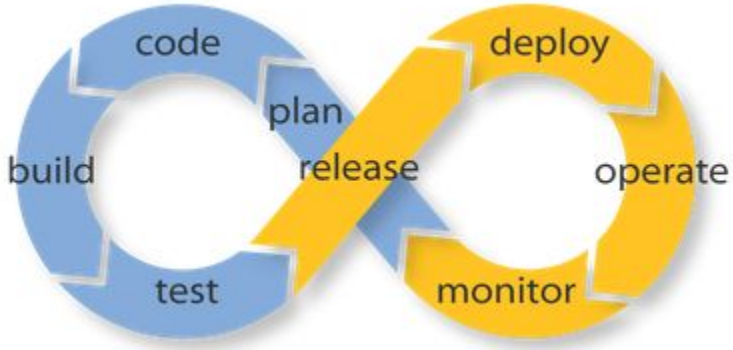
Development + Operations

Разработка

Тестирование

Развертывание

Ключевые моменты:

Частые р 

жки, изменение бизнес-процессов