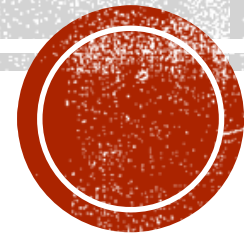


ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И ЭКСПЛУАТАЦИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Лекция 2

21 февраля 2019 г.



ИНФОРМАЦИОННАЯ СИСТЕМА

Опр. 1 (ISO/IEC 2382:2015): Information system: An information processing system, together with associated organizational resources such as human, technical, and financial resources, that provides and distributes information.

Опр. 1 (рус): Информационная система: система обработки информации, а также связанные с ней организационные ресурсы, такие как человеческие, технические и финансовые ресурсы, которые обеспечивают и распространяют информацию.

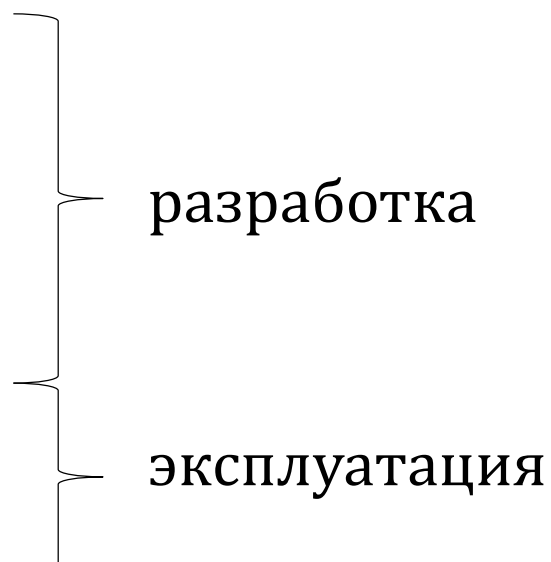
Опр. 1*: Информационной системой называется комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей.



ЖИЗНЕННЫЙ ЦИКЛ ПО

Опр. 2: Жизненный цикл программного обеспечения (ПО) — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Этапы жизненного цикла

- Изучение предметной области
 - Проектирование
 - Разработка
 - Тестирование
 - Внедрение
 - Сопровождение и эксплуатация
 - Вывод из эксплуатации
- 
- The diagram uses curly braces to group the stages of the software lifecycle into two main phases:
- разработка** (development): This phase includes the first five stages: Изучение предметной области, Проектирование, Разработка, Тестирование, and Внедрение.
 - эксплуатация** (operation): This phase includes the last two stages: Сопровождение и эксплуатация, and Вывод из эксплуатации.



МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ПО

Опр. 3: Модель жизненного цикла ПО — структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла ПО.

Модели ЖЦ и методологии разработки ПО – разные понятия.

- Scrum, TDD, XP, ... – методологии
- Каскадная, спиральная, итерационная, ... – модели



ЭВОЛЮЦИЯ МОДЕЛЕЙ ЖИЗНЕННОГО ЦИКЛА ПО

Первая модель – «закодировать и пофиксить» (`code-and-fix model`)

Два шага:

- Написать некоторый код
- Исправить ошибки

Сначала – кодировать, потом – думать.

Достоинства:

- готовый «продукт» получается супер-быстро

Недостатки (одни сплошные):

- После многочисленных правок код превращается в месиво
- Получается не то, что хочет заказчик, а то, как это понял программист
- Код очень накладно поддерживать, т.к. он это не предусматривает



ЭВОЛЮЦИЯ МОДЕЛЕЙ ЖИЗНЕННОГО ЦИКЛА ПО

Еще одна ранняя модель – эволюционная.

То же самое, что и первая модель, но процесс происходит малыми шагами, после каждого результат показывается заказчику, т.е. продукт «эволюционирует».

"I can't tell you what I want, but I'll know it when I see it."

Достоинства:

- готовый «продукт» получается более менее таким, как хочет заказчик.

Недостатки (одни сплошные):

- Те же, что и в первой модели



КАСКАДНАЯ И ИТЕРАЦИОННАЯ МОДЕЛИ

На смену вышеизложенным моделям пришли другие, которые можно разделить на два класса:

Линейная (каскадная)

Один проход по всем стадиям жизненного цикла.

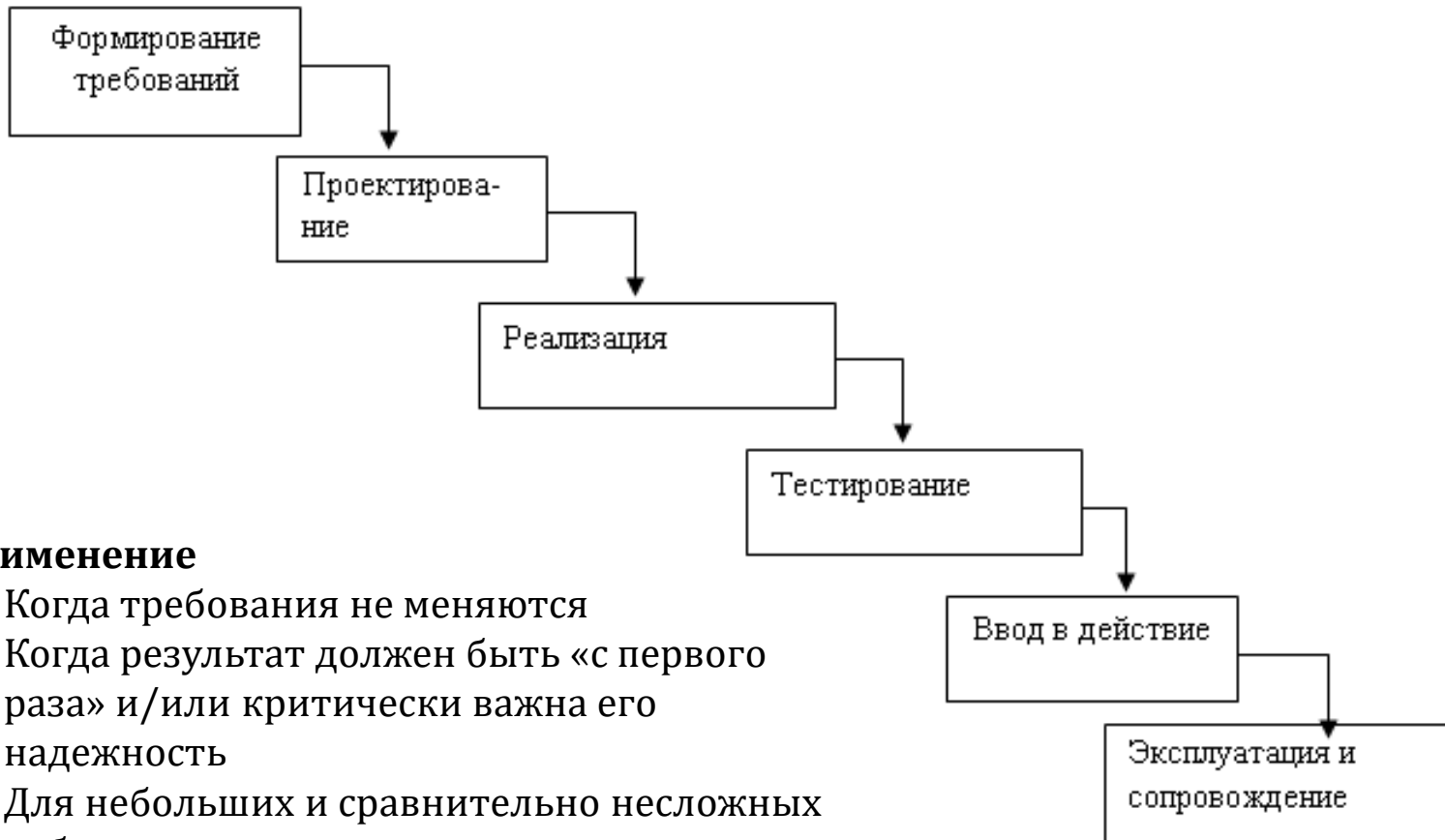
Итерационные

Все или часть этапов ЖЦ повторяются в цикле до тех пор, пока приемлемый результат не будет достигнут



КАСКАДНАЯ МОДЕЛЬ ЖЦ

Один проход, все стадии идут одна за другой, каждая начинается строго после окончания предыдущей.



Применение

- Когда требования не меняются
- Когда результат должен быть «с первого раза» и/или критически важна его надежность
- Для небольших и сравнительно несложных работ

Преимущества:

- Децентрализация и контроль
- Понятно и легко использовать
- Легко управлять из-за жесткости модели
- Хорошо работает для проектов, где требования очень хорошо понятны изначально

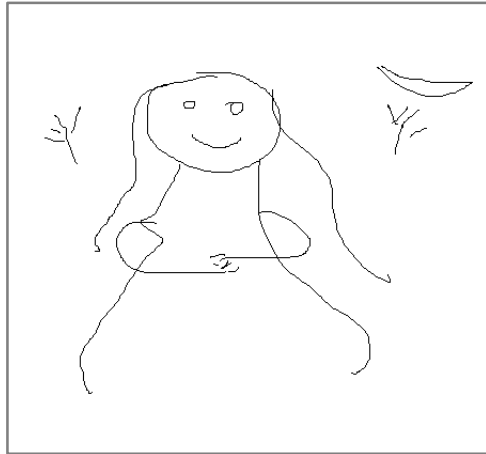
Недостатки:

- Трудно оценить время и стоимость каждого этапа
- Сложно «вернуться назад»
- Не подходит для проектов, где риск изменения требований высокий



ПРОТОТИПНАЯ МОДЕЛЬ ЖЦ

0



1



2



3



Преимущества:

- Иногда помогает продемонстрировать концепцию потенциальным инвесторам для получения финансирования.
- Снижает риск неудачи, так как потенциальные риски могут быть выявлены на ранней стадии.
- Пользователи получают лучшее представление о разрабатываемой системе и могут предлагать изменения.
- Пользовательская обратная связь доступна на ранней стадии, что обеспечивает лучшее решение.

Недостатки:

- Если в модельном продукте требуется слишком много изменений, это может нарушить ритм разработки.
- Может увеличить сложность системы, поскольку объем системы может расширяться за исходные границы.
- Прототипирование обычно выполняется за счет разработчика. Поэтому это должно быть сделано с использованием минимальных ресурсов.



- Оценка стоимости становится легкой
- Первые результаты получаются раньше
- Постоянная обратная связь с заказчиком

- Дорогостоящая модель
- Для оценки риска требуются эксперты
- Тщательное соблюдение всех процедур

- Для средних и крупных проектов.
- Для проектов с высоким уровнем риска.
- Пользователи не уверены в своих потребностях.
- Требования сложные.
- Если в проекте требуются частые изменения.

ИНКРЕМЕНТНАЯ МОДЕЛЬ ЖЦ

Продукт разрабатывается, внедряется и тестируется постепенно. Еще немного добавляется каждый раз, пока продукт не будет закончен.

Преимущества:

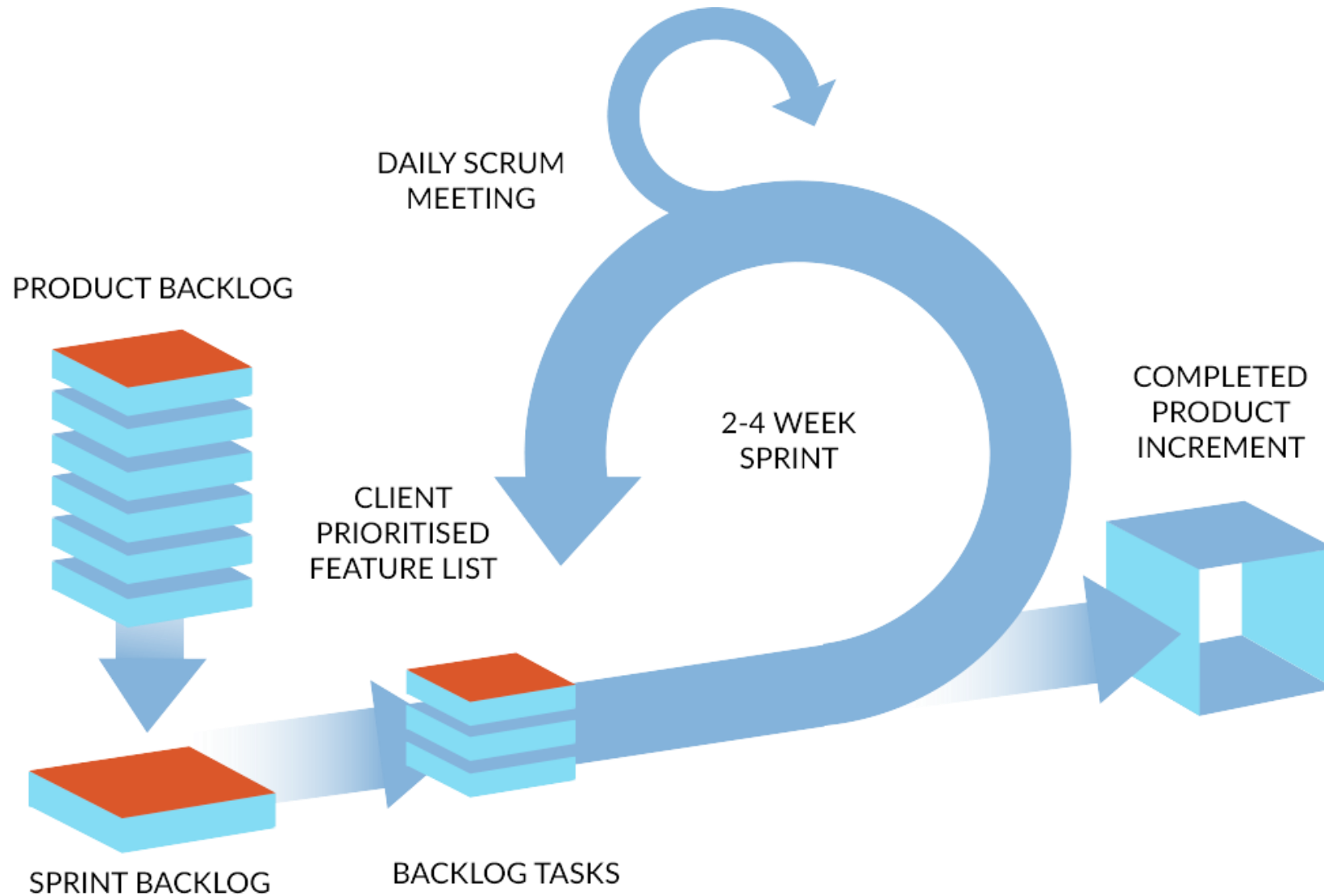
- рабочее программное обеспечение создается быстро
- более гибкая и менее дорогостоящая при изменении объема и требований
- легче тестировать и отлаживать, поскольку на каждой итерации происходят небольшие изменения
- Постоянная реакция от клиента

Недостатки:

- на каждом цикле могут возникнуть проблемы, связанные с архитектурой системы, которые не было видно на ранних стадиях
- нуждается в хорошем планировании на каждом шагу
- требуется четкое и полное определение всей системы, прежде чем она может быть разбита по частям



ИНКРЕМЕНТНАЯ МОДЕЛЬ ЖЦ



ГИБКАЯ МОДЕЛЬ ЖЦ

Гибкая модель (**agile**) – модель ЖЦ в основном **разработки** ПО.

Семейство методологий, придерживающихся **Manifesto for Agile Software Development**

Основная идея – требования будут меняться.

Люди и их взаимодействие

важнее, чем

Процессы и инструменты

Работающее программное обеспечение

важнее, чем

Исчерпывающая документация

Взаимодействие с заказчиком

важнее, чем

Обсуждение контракта

Реагировать на изменения

важнее, чем

Следовать плану

