ЖИЗНЕННЫЙ ЦИКЛ СОЗДАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Постоев Дмитрий

Часть 6:
Разработка (программирование)

Определения

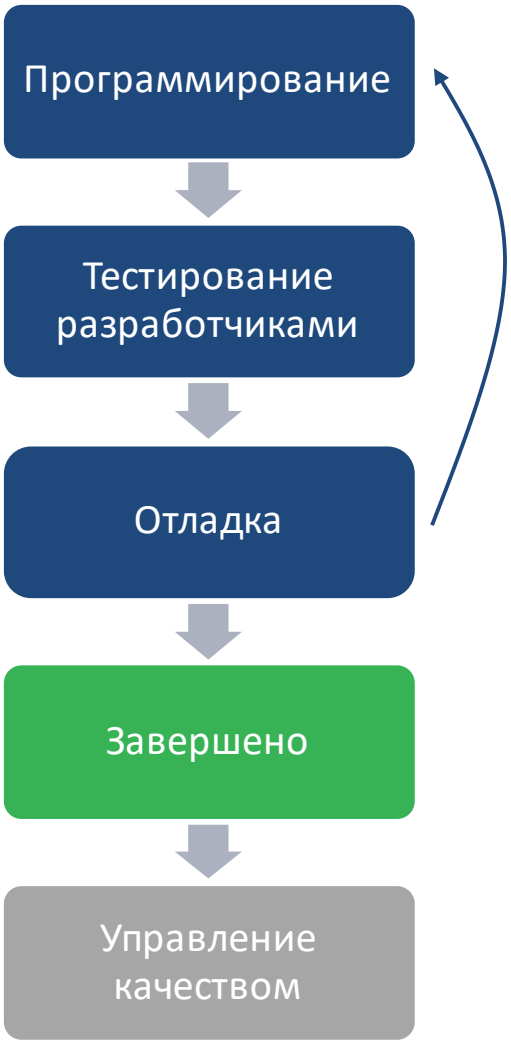
Разработка (конструирование) – детальное создание программного обеспечения посредством комбинации кодирования, верификации, модульного тестирования, интеграционного тестирования и отладки.[1]

[1] IEEE SWEBOK Guide V3.0, Ch. 3

Ключевые моменты разработки

1. Минимизация сложности (Keep It Stupid Simple)
2. Повторное использование
3. Масштабируемость
4. Производительность
5. Тестируемость кода
6. TDD
7. Скорость реакции
8. Социальные аспекты

Процесс конструирования программы



Программирование

– процесс написание инструкций на специальном языке (программы) для решения поставленной задачи.

Аспекты программирования:

- Инструментарий
- Стандарты кодирования
- Инспекция кода (Code Review)
- Парное программирование
- Рефакторинг
- Безопасное программирование

Инструментарий

1. **Текстовые редакторы**

Терминалы: Vim, Emacs

Графические: Sublime Text, Notepad++

2. **IDE**

Open-source: Eclipse, Netbeans

Proprietary: Visual Studio, Clion, Xcode

3. **Профилировщики**
4. **Статические/динамические анализаторы**
5. **Командная оболочка (Command-line shell)**

Примеры: bash, powershell

6. **Прочее**

Примеры: cmake, diff, valgrind, gdb

Стандарты кодирования

– набор правил и соглашений, используемых при написании исходного кода на некотором языке программирования.

Что включает в себя:

- Стиль наименования переменных, функций и других идентификаторов
- Стиль отступов
- Расстановка скобок, пробелов
- Комментирование
- Расположение и структурирование кода
- Правила хорошего кодирования

Примеры:

- Google C++ Coding Style (https://google.github.io/styleguide/cppguide.html)
- GNU C Coding Style (https://www.gnu.org/prep/standards/html_node/Writing-C.html)
- NASA C Style Guide (http://homepages.inf.ed.ac.uk/dts/pm/Papers/nasa-c-style.pdf)

Инспекция кода (Code Review)

- систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки.

Зачем:

- Улучшение качества кода
- Совершенствование навыков разработчиков
- Быстрое нахождение ошибок

Парное программирование

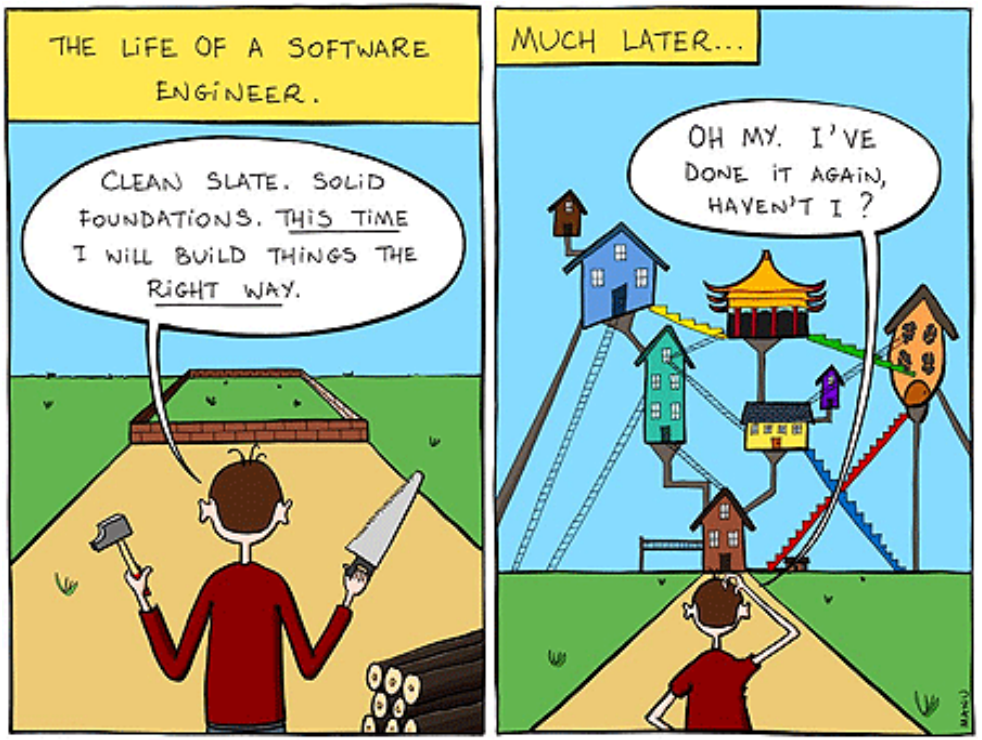
- техника программирования, при которой исходный код создаётся парами людей, программирующих одну задачу, сидя за одним рабочим местом.

Зачем:

- Улучшение качества кода
- Совершенствование навыков разработчиков
- Быстрое нахождение ошибок
- Повышение командного духа

Рефакторинг

- изменения внутренней структуры программы, с целью упрощения дальнейших изменений и понимания работы, без изменения ее внешнего поведения.



Безопасное программирование

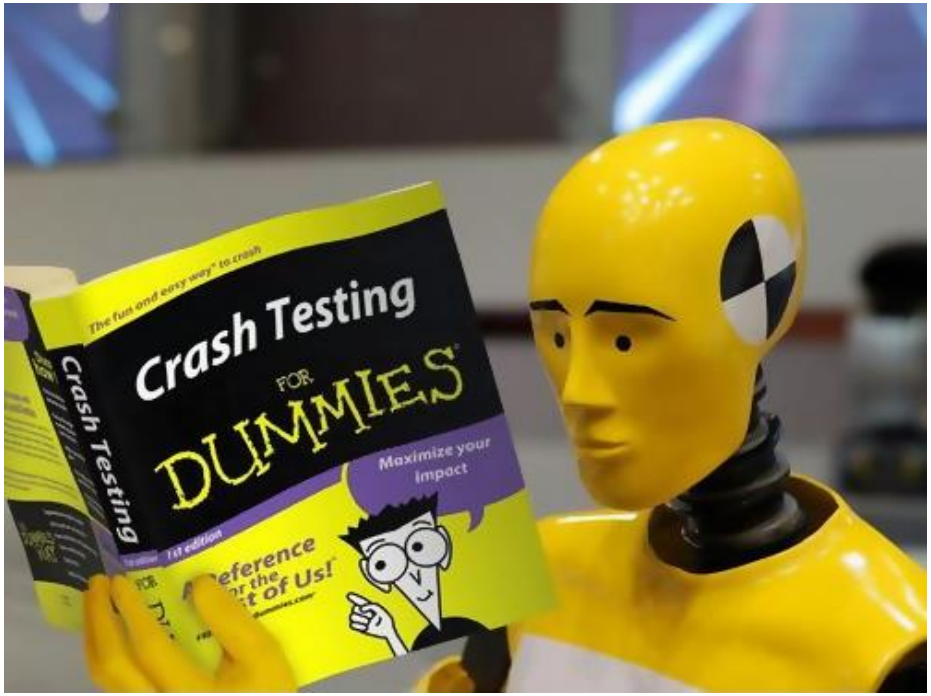
- разновидность безопасного дизайна, направленная на корректное работу функций и модулей в непредвиденных обстоятельствах.

Примеры правил:

- Проверка вводимых пользователем данных (как можно раньше)
- Использовать **assert**, которые должны быть истинны, иначе программа завершает свою работу
- Использовать константные выражения вместо переменных
- Инициализировать переменные
- Использовать **strn*** вместо **str*** (во избежание переполнения буфера)
- Использование анализаторов кода
- TDD
- Дублирование и резервирование систем (watchdog timer)

Тестирование разработчиками

1. Модульное
2. Интеграционное
3. Системное



Модульное тестирование

- процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Виды модульных тестов:

- Позитивные
- Негативные
- Граничные
- Обработка «ошибки на единицу»

Фреймворки:

CUnit, Google C++ Testing Framework, Boost Test, CppUnit

Интеграционное тестирование

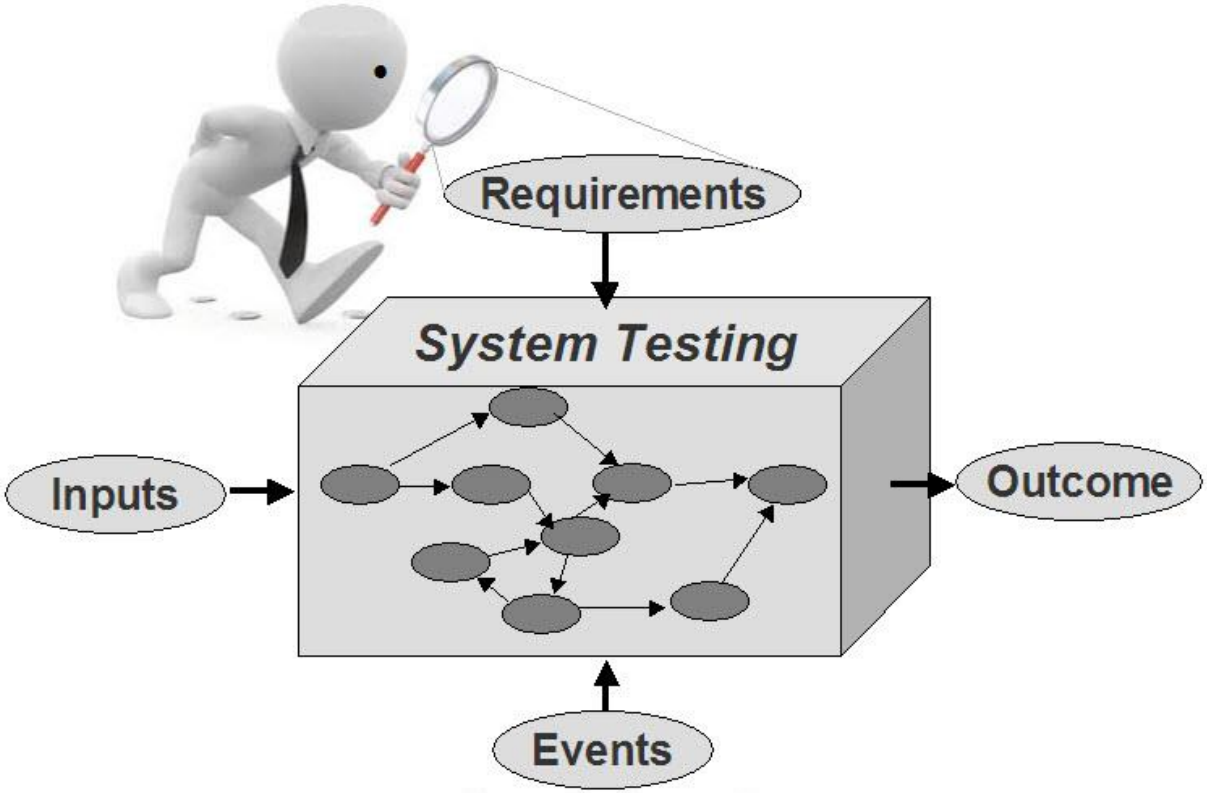
- это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям.

Smoke Test (дымовое тестирование)

– минимальный набор тестов на явные ошибки.

Системное тестирование

- это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям.



Отладка

- этап разработки программного обеспечения, на котором обнаруживают, локализуют и устраняют ошибки.

Алгоритм:

1. Воспроизвести проблему
2. Упрощение входных данных (сокращение места поиска)
3. Использовать инструмент отладки для анализа подозрительного места

Техники отладки:

- Интерактивный
- Вывод на экран
- Посмертная отладка (анализ crash dump, журналов)
- Удаленная

Литература

1. «Code Complete», Steve McConnell
2. «Test Driven Development: By Example», Kent Beck
3. «Rapid Development», Steve McConnell
4. «Refactoring: Improving the Design of Existing Code», Martin Fowler
5. «The Pragmatic Programmer: From Journeyman to Master», Andrew Hunt and David Thomas

Домашнее задание

1. **Руководитель проекта** совместно с **Разработчиками** разбивает проект на модули, уточняет интерфейсы взаимодействия между ними согласно Архитектуре и назначает ответственного за модуль **Разработчика.**
2. **Руководитель проекта** совместно с **Разработчиком** соответствующего модуля подготавливают план по его реализации, разбивая на мелкие задачи. **Руководитель проекта** назначает на этого **Разработчика** полученные задачи, создавая *Issue* с необходимым описанием.
3. **Разработчики** подготавливают свои локальные репозитории и IDE в соответствии с рекомендациями.
4. **Разработчики** реализуют функции согласно плану, согласованному с **Руководителем проекта** (и назначенной на них *Issue*). Изменения добавляются (в ходе реализации) в ветку с названием, соответствующем их модулю.

Примечание: Задачи назначаются и оформляются на Github