# НАДЁЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

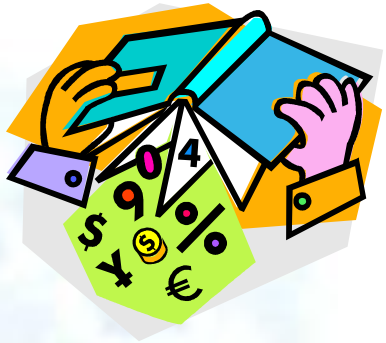
## Лекция 10:
## *Определение необходимого уровня надёжности и построение функционального среза*

ВМиК МГУ им. М.В. Ломоносова,
Кафедра АСВК, Лаборатория Вычислительных Комплексов
Ассистент Волканов Д.Ю.

# О чём этот курс

- Концепции и связи
- Аналитические модели и средства поддержки
- Методы для увеличения надёжности ПО:
  - Верификация
  - Статический анализ
  - Расчёт надёжности
  - **Тестирование**
  - *Анализ статистики ошибок*
  - *Безопасность*

# План лекции



- Определение необходимого уровня надёжности программ
- Классы тяжести отказов
- Интенсивность отказов
- Стратегии уменьшения числа отказов
- Операция, функция, прогон и его типы
- Функциональный срез
- Разрабока функционального среза



3

# Эпиграф

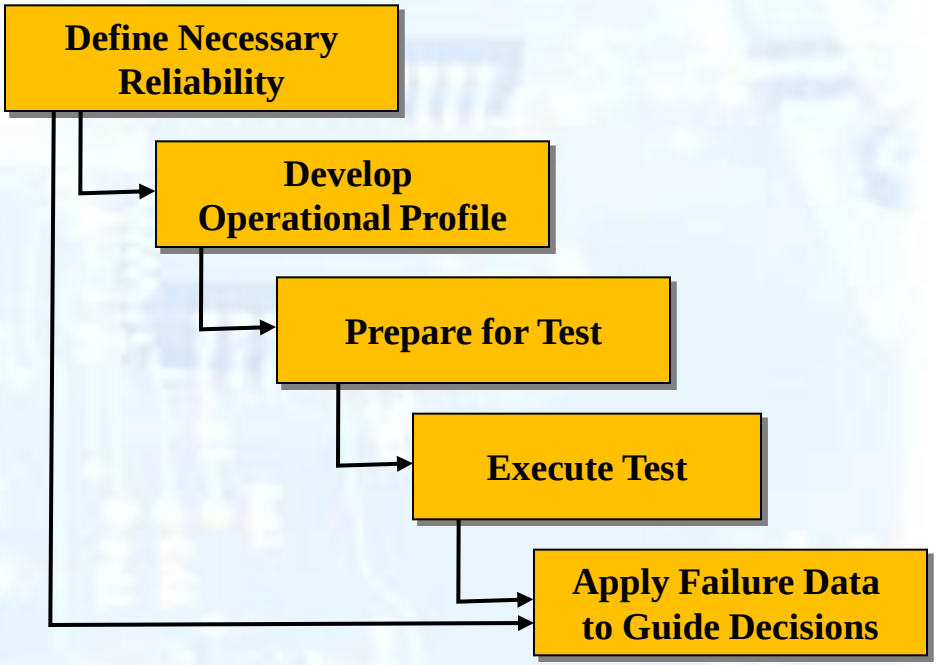
- *Да, все голосуют за качество, но если оно стоит лишнюю копейку, вы начинаете быстро познавать настоящее отношение к качеству со стороны тех, кто платит.*

***Том Демарко***

***и Тимоти Листер***

# SRE: процесс

- 5 шагов в SRE процессе:
  - **Определить необходимый уровень надежности**
  - Разработать функциональный срез
  - Подготовить для тестирования
  - Выполнить тестирование
  - Проанализировать данные об ошибках и использовать их для принятия решений



# Как определить требуемый уровень надёжности программы?

# Надёжность и Риск

- Требуемый уровень надёжности зависит от риска. Больший риск требует большего уровня надёжности

- Литература:
- Software Testing Fundamentals: Methods and Metrics
- Marnie L. Hutcheson

ISBN:047143020X John Wiley & Sons


7

# Как найти нужный уровень надёжности:

1) Разбить возможные отказы по классам тяжести
2) Установить целевую интенсивность отказов для каждой тестируемой системы
3) Выбрать единый масштаб для всех тестируемых систем
4) Найти текущий показатель интенсивности отказов
5) Выбрать стратегию для достижения поставленного уровня интенсивности


8

# 1. Классы тяжести отказов

- Отказы различаются между собой по уровню их воздействия на тестируемую систему
- Каждый класс тяжести отказов объединяет отказы с одинаковым влиянием на пользователя [по заданным критериям]
- Примеры критериев классификации:
  - Стоимость ущерба, угроза жизни человека
- Тяжесть отказа ≠ сложность отказа
- Тяжесть отказа может зависеть от времени его наступления


9

# Критерии классификации тяжести отказов

- **Стоимость**
- Какой ущерб способен нанести отказ в терминах стоимости устранения, восстановления, простоя системы, …
- Классы тяжести по стоимости ущерба обычно различаются на один порядок
- На практике достаточно использовать 4 группы классов

| Класс тяжести | Стоимость ($) |
|---|---|
| 1 | > 100,000 |
| 2 | 10,000 –100,000 |
| 3 | 1,000 – 10,000 |
| 4 | < 1,000 |


10

# Критерии классификации тяжести отказов

- **Функционирование системы**
- Включает такие факторы как потерю данных, время простоя системы, возможность её восстановления, …

| Класс тяжести | Ущерб функционированию |
|---|---|
| 1 | Перерыв в работе сервиса |
| 2 | Ошибки в работе сервиса |
| 3 | Неудобства, требующие срочного исправления |
| 4 | Неудобства в работе, исправление которых может подождать |


11

# Критерии классификации тяжести отказов

- **Окружающая среда**
- Может привести к нанесению вреда экологии, исчезновению видов растений и животных, …
- Широко применяется в атомной и химической промышленности

| Класс тяжести | Ущерб экологии |
|---|---|
| 1 | Значительный и невосстановимый |
| 2 | Значительный и частично восстановимый |
| 3 | Незначительный, но невосстановимый |
| 4 | Незначительный и восстановимый |


12

# Критерии классификации тяжести отказов

- **Жизнь людей**
- Отказ может привести к нанесению вреда окружающей среде, потере человеческих жизней
- Применяется в аэрокосмической, медицинской, военной и других отраслях

| Класс тяжести | Определение |
|---|---|
| 1 | Возможны людские потери |
| 2 | Значительные повреждения здоровья людей |
| 3 | Незначительные повреждения здоровья людей |
| 4 | Восстановимые нарушения здоровья |


13

# Как определить уровень тяжести отказа?

- На основе опыта: попросить пользователей, заказчиков и разработчиков и сравнить с аналогичными продуктами
- Перечислить всевозможные факторы ущерба, вызванные отказом системы
- Сузить список до наиболее важных факторов
- Некоторые факторы могут быть трудноизмеримыми [репутация компании]


14

# Противоречивые оценки

- Пользователи и разработчики могут иметь разные представления о тяжести отказов
- Мнения отдельных групп должны быть согласованы до установки показателя целевой интенсивности отказов
- Разрешить противоречия в тяжести отказов часто помогает исследование аналогичных продуктов


15

# Документирование уровней тяжести отказов

| *Способ использования* | *Критерии классификации* | *Список возможных отказов* | | | |
|---|---|---|---|---|---|
| | | *Класс 1* | *Класс 2* | *Класс 3* | *Класс 4* |
| | **Стоимость** | | | | |
| | **Функционирование системы** | | | | |
| | **Человеческие жизни** | | | | |
| | **Окружающая среда** | | | | |
| | **Другое (указать отдельно)** | | | | |

**Классы тяжести отказов определяются независимо по каждому критерию**



16

# 2. Целевая интенсивность отказов

- Отражает ожидаемое число «багов», оставшихся в продукте после завершения его разработки
- Альтернативный способ измерения надёжности


17

# Целевая интенсивность отказов

- Интенсивность отказов обычно определяется как число случившихся отказов системы за заданное время (или аналогичными метриками):
  - 3 сигнала тревоги за 100 часов работы.
  - 5 отказов за время печати 1000 документов.
- Интенсивность отказов системы – сумма интенсивностей отказов всех компонентов системы


18

# Как установить целевую интенсивность отказов?

- Оценка основывается на опыте и субъективно зависит от проекта
- Зависит от компромисса между качеством продукта (временем и стоимостью разработки) и его функциональностью
- **Правило большого пальца:** Устанавливай целевую интенсивность отказов исходя из расчётной итоговой стоимости проекта



19

# Как установить целевую интенсивность отказов?

- Типичные целевые интенсивности отказов для разных проектов:

| **Стоимость отказа** | **Целевая интенсивность отказов (λ)** | **Время между отказами** |
|---|---|---|
| > 1,000,000,000 $ | 1 в 1,000,000,000 часов | 114,000 лет |
| > 1,000,000 $ | 1 в 1,000,000 часов | 114 лет |
| > 1,000 $ | 1 в 1,000 часов | 6 недель |
| > 100 $ | 1 в 100 часов | 100 часов |
| > 10 $ | 1 в 10 часов | 10 часов |
| > 1 $ | 1 в час | 1 час |


20

# Целевая интенсивность отказов и надёжность

- Зависимость между целевой интенсивностью и надёжностью

$$\lambda = \frac{-\ln R}{t} \quad \textit{или} \quad \lambda \approx \frac{(1-R)}{t} \ \textit{для} \ R \geq 0.95$$

$\lambda$ - интенсивность отказов

*R - надёжность*

*t* - единица измерения (время, ...)

- При R = 0.992 за 8 часов, $\lambda$ = 0.001



21

# Надёжность & Интенсивность отказов

| *Надёжность для 1 часа работы* | *Интенсивность отказов* |
|---|---|
| 0.36800 | 1 отказ в час |
| 0.90000 | 105 отказов в 1000 часов |
| 0.95900 | 1 отказ в день |
| 0.99000 | 10 отказов в 1000 часов |
| 0.99400 | 1 отказ в неделю |
| 0.99860 | 1 отказ в месяц |
| 0.99900 | 1 отказ в 1000 часов |
| 0.99989 | 1 отказ в год |



22

# Целевая интенсивность отказов и доступность

- Зависимость между целевой интенсивностью отказов и доступностью:

$$A(t) = \frac{1}{1 + t_m \lambda(t)} \quad or \quad \lambda(t) = \frac{1 - A(t)}{t_m A(t)}$$

- $t_m$ $\lambda$ - интенсивность отказов
  - время простоя на ошибку
- Если система должна быть доступна 99% времени работы и время простоя 6 минут, тогда целевая интенсивность отказов равна 1 отказ в 10 часов


23

# Целевая интенсивность отказов и среднее время безотказной работы

- Ещё одно определение доступности:

$$A = \frac{MTTF}{MTTF + MTTR} = \frac{MTTF}{MTBF}$$

$$\lambda = \frac{1 - \frac{MTTF}{MTTF + MTTR}}{\frac{MTTR \times MTTF}{MTTF + MTTR}} = \frac{1}{MTTF}$$

| | |
|---|---|
| $\lambda$ | интенсивность отказов |
| *MTTR* | среднее время восстановления после отказа |
| *MTTF* | среднее время безотказной работы |


24

# Целевая интенсивность отказов и вероятность первого отказа

- Вероятность первого отказа $z(t)$ – вероятность сбоя в заданном временном интервале при условии отсутствия сбоев до его наступления
- Вероятность первого отказа 0.05 означает, что первый отказ с вероятностью 5% произойдёт в заданном интервале и не раньше его наступления
- Для экспоненциального распределения $z(t) = \lambda$



25

# Целевая интенсивность отказов и прибыль

- Расчёт основывается на опыте аналогичных систем с помощью сравнения их основных качественных характеристик и отношению к ним пользователя
- Сравниваются тенденции изменения приоритетов развития компромисса между прибылью и интенсивностью отказов


26

# Пример

- Совет: выбирайте диапазон на границе высоких прибылей

| Интенсивность отказов | Прибыль |
|---|---|
| 1000 | 0,6 |
| 500 | 1,5 |
| 250 | 5,8 |
| 100 | 17,2 |
| 50 | 21,4 |
| 25 | 19,8 |
| 10 | 16,3 |



27

# Надёжность vs. Доступность

- Зачем задавать надёжность, если доступность более интуитивна и проще для понимания?
- Доступность субъективна для каждого пользователя и обычно существуют способы изменения доступности системы без изменения её внутренней надёжности
- **Пример:** дублирование серверов DNS повышает доступность системы, но не повышает надежность серверного ПО


28

# Разрабатываемое ПО

- *Создаваемый продукт – лишь часть системы*




29

# 3. Выбор общего масштаба

- Для разных целевых интенсивностей отказов различных частей проекта могут использоваться разные масштабы:
  - Целевая интенсивность отказов системы

    30 отказов на 1 миллион операций
  - Среднее время между отказами ОС
    - 3,000 отказов на 10 миллионов операций
  - Среднее время между отказами оборудования
    - 1 на 30 часов работы
- Нужно выбрать общий масштаб



30

# Как найти целевую интенсивность отказов разрабатываемого ПО?

1. Установить общесистемную целевую интенсивность отказов
2. Перевести системы измерения интенсивностей отказов компонентов в единую систему измерения
3. Вычесть из целевой интенсивности системы расчётную интенсивность отказов приобретённых компонентов
4. Вычесть расчётную интенсивность отказов окружения (ОС и интерфейсных систем), в котором будет работать разработанное ПО
5. Остаток – целевая интенсивность для разрабатываемого ПО


31

# Вычисление целевой интенсивности отказов для разрабатываемого ПО (1)

- **Пример 1:**
  - Общесистемная целевая интенсивность отказов
    100 отказов на 1 миллион операций
  - Интенсивность отказов «железа»
    0.1 отказа в час
  - Число отказов ОС на 100,000 операций
    0.4 отказа в час

- Целевая интенсивность ПО
- 95 отказов на 1 миллион операций


32

# Вычисление целевой интенсивности отказов для разрабатываемого ПО (2)

- **Пример 2: БД на Win 2K:**
  - Общая целевая интенсивность отказов
    - 30 отказов на 1 миллион операций
  - Среднее время между отказами для Win 2K
    - 3,000 часов на 10 миллионов операций
  - Среднее интенсивность отказов «железа»
    - 1 отказ в 30 часов
  - Интенсивность отказов других подсистем
    - 9 отказов на 1 миллион операций
- Какова целевая интенсивность отказов для разрабатываемого ПО?


33

# Вычисление целевой интенсивности отказов для разрабатываемого ПО (3)

$$\lambda_{os} = \frac{1}{MTTF} = 1/3000$$

$$\lambda_{hardware} = \frac{1}{30} = 100/3000$$

$$\lambda_{os} + \lambda_{hardware} = 101/3000 \quad for \quad 10,000,000\ transactions$$

$$\lambda_{other} = 90 \quad for \quad 10,000,000\ transactions$$

$$\lambda_{total} = 191 \quad for \quad 10,000,000\ transactions$$

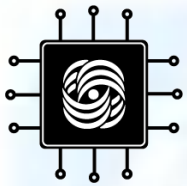
$$\lambda_{F} = 300 \quad for \quad 10,000,000\ transactions$$

*therefore*

$$\lambda_{developed\_software} = 300 - 191 = 109 \quad for \quad 10,000,000\ transactions$$



34



# 4. Стратегии уменьшения числа отказов программ

- 4 основных стратегии:
  - Недопущение
  - Устранение
  - Невосприимчивость
  - Прогнозирование



35

# Недопущение отказов

- Избежание отказов по построению.
- **Действия:**
  - Перепроверка требований
  - Пересмотр архитектуры и очистка кода
  - Использование стандартов (ISO 9000-3, etc.)
  - Использование CASE-средств со встроенными механизмами контроля
- **Эффективность применения:**
  - Пропорция отказов после выполнения указанных действий не меняется.



36

# Устранение отказов

- Ошибки в коде устраняются с помощью верификации и валидации.
- **Действия:**
  - Семантическая проверка кода
  - Тестирование
- **Эффективность применения:**
  - Уменьшение числа ошибок в коде.
  - Уменьшение числа отказов ПО.


37

# Тестирование vs. Инспектирование

- Инспеция – детальное изучение спецификаций, архитектуры проекта, его кода, …
- Инспектирование эффективнее тестирования более чем в 20 раз
- Пересмотр кода позволяет найти в 2 раза больше ошибок, чем его тестирование
- 80% ошибок находятся во время инспектирования
- Инспектирование уменьшает стоимость разработки в 10 раз


38

# Можно ли заменить тестирование инспектированием?

- НЕТ. Иногда инспектирование не позволяет получить информацию, доступную с помощью тестирования:
  - Реальное поведения системы
  - Показатели производительности, удобства использования, …
  - Показатель надёжности системы



39

# Невосприимчивость к отказам

- Обеспечение сервиса несмотря на отказы ПО с помощью избыточности
- **Действия:**
  - Проектирование и обеспечение избыточности
- **Эффективность применения:**
  - Уменьшение числа отказов системы из-за её избыточности



40

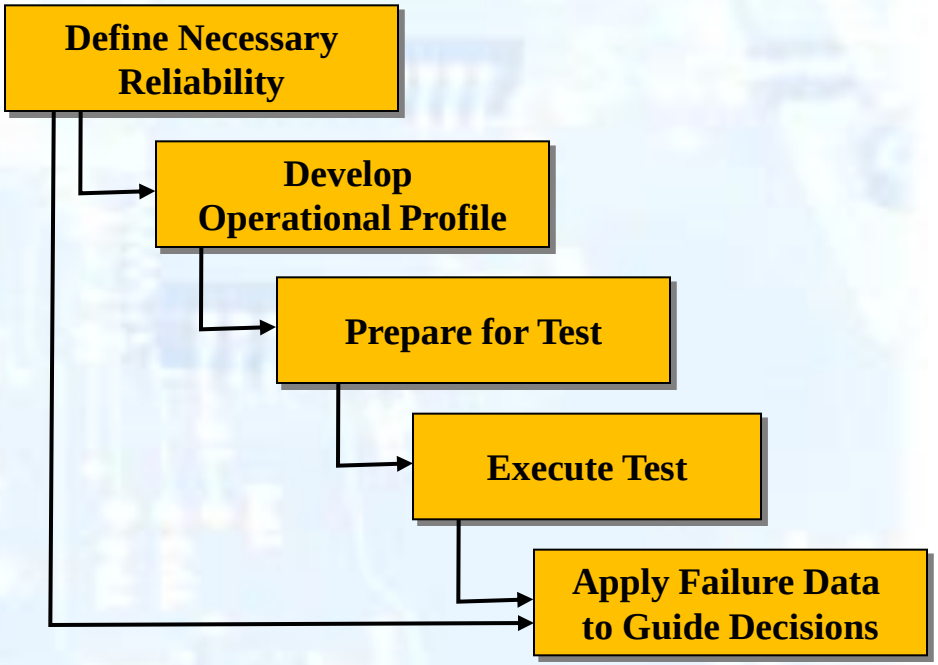
# Прогнозирование отказов

- Построение оценки вероятности возникновения отказов
- **Действия:**
  - Построение модели надёжности
  - Сбор информации об отказах
  - Анализ результатов
- **Эффективность применения:**
  - Уменьшение интенсивности возникновения ошибок за счёт управления надёжностью


41

# SRE: процесс

- 5 шагов в SRE процессе:
  - Определить необходимый уровень надежности
  - **Построить функциональный срез**
  - Подготовиться к тестированию
  - Выполнить тестирование
  - Проанализировать данные об ошибках и использовать их для принятия решений



# Профиль

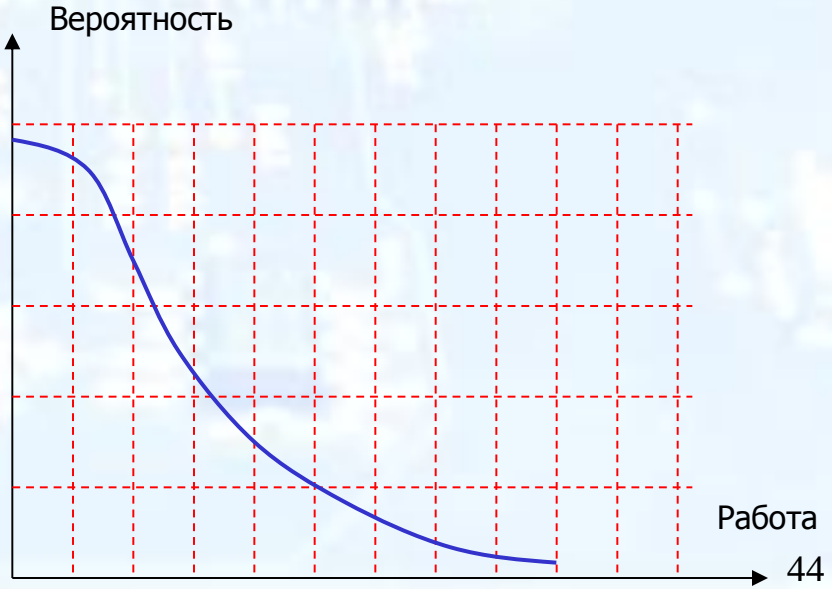
- Профиль - набор альтернативных сценариев вместе с вероятностями их выполнения
- Пример:
  - Если сценарий A выполняется в 30% случаев, сценарий B – в 20%, а C – в 50%, то профиль имеет следующий вид:

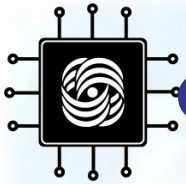
| | |
|---|---|
| A | 0.3 |
| B | 0.2 |
| C | 0.5 |

# Функциональный срез

- Функциональный срез – полный набор сценариев выполнения вместе с их вероятностями (во время работы заданного ПО)
- Функциональный срез описывает распределение событий, которые происходят во время работы программы
- Функциональный срез программы отражает то, как она будет использоваться на практике

Функциональный срез зависит от пользователя, времени, используемой функциональности т.д.





# Функциональный срез

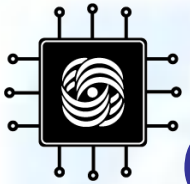
Функциональный срез – полный набор операций (имя и частота) и вероятностей их использования

## Пример:

| Operation | Occurrence rate (operations per hour) |
|---|---|
| Phone number entry | 10,000 |
| Add subscriber | 50 |
| Delete subscriber | 50 |
| Process voice call, no pager, answer | 18,000 |
| Process voice call, no pager, no answer | 17,000 |
| Process voice call, pager, answer | 17,000 |
| Process voice call, pager, answer on page | 12,000 |
| Process voice call, pager, no answer on page | 10,000 |
| Process fax call | 15,000 |
| Audit section of phone number database | 900 |
| Recover from hardware failure | 0.1 |
| Total | 100,000 |

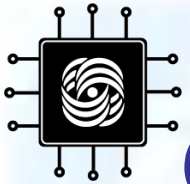
# Почему это полезно?

- Профиль функционирования программы даёт информацию о том, как её будут использовать, и позволяет разработчикам сфокусироваться на решении действительно актуальных задач
- С помощью профиля разработчики могут:
  - Увеличить эффективность разработки и тестирования.
  - Сделать тесты более реалистичными.
- С точки зрения эффективного управления разработку всегда нужно начинать с изучения приложений программы и составления соответствующего профиля функционирования



# Операция (1)

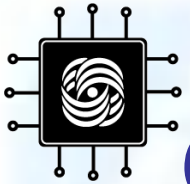
- Неформальное определение:
  - Операция представляет собой задачу, выполняемую программой в заданном окружении. Причём результат выполнения задания должен быть обозримым для пользователя
- Более формальное определение:
  - Операция – крупное логическое краткосрочное задание, выполнение которого значительно отличается от других заданий



# Операция (2)

- *Крупное* означает, что операция должна относиться к функциональным требованиям, предъявляемым к разрабатываемой программе, а не к подзадаче, решаемой в её реализации

- Операция – *логическая* концепция в том смысле, что она может включать набор программных, аппаратных и пользовательских действий



# Операция (3)

- *Кратковременность* означает, что при нормальном функционировании программы, она может выполнять тысячи операций в час

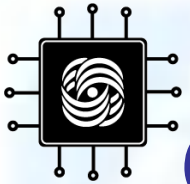
- *Значительно отличающееся выполнение* означает, что с высокой долей вероятности отказ внутри операции не произойдёт при выполнении других операций

# Прогон и его типы (1)

- Прогон - наименьшая часть работы, которая может быть вызвана внешним воздействием
- Примеры:
  - Звонок в телекоммуникационной системе
  - Команда в интерактивном приложении
  - Транзакция в интернет-магазине
  - etc.

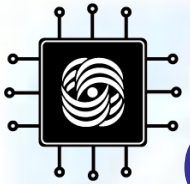
# Прогон и его типы (2)

- Прогоны связаны со входным состоянием, включающим полный набор входных переменных (любых данных, полученных из вне)
- Входные переменные могут быть:
  - Частично неизвестными на стадии проектирования программы
  - Скорее логическими, чем физическими [например адрес в памяти]
- Прогоны с одинаковым входным состоянием принадлежат к прогона одного типа



# Операции vs. Функции (1)

- Функция, заданная требованиями к программе, потенциально может стать операцией или набором операций на стадии её разработки
- Функция – группа типов прогона, заданных требованием к программе и наблюдаемых пользователем
- Операция – группа типов прогона заданных во время разработки программы и рассматриваемых с точки зрения её проектирования



# Операции vs. Функции (2)

| Анализ | Проектирование |
|---|---|
| Концентрация на понимании проблемы | Концентрация на понимании решения |
| Идеальный случай | Близко к реальному коду |
| Поведение | Операции и атрибуты |
| | Жизненный цикл объектов |

Пример:

Перемещение объекта в памяти (функция) на стадии определения требований может превратиться в операции создания, копирования и удаления на стадии разработки

# Как искать операции?

# Представление

- Описывает функциональный профиль:
  - Табличное представление
  - Графическое представление

# Representation: Tabular

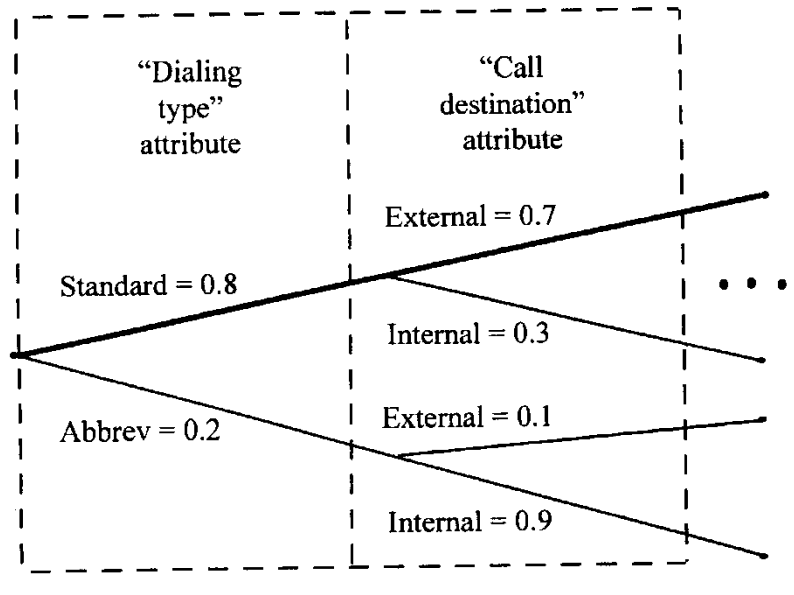
- Табличное представление включает список имён операций и вероятностям их возникновения

| Command | Transactions per month | Occurrence probability |
|---|---|---|
| relocate | 780 | 0.0153 |
| remove | 90 | 0.0017 |
| add -s staff | 70 | 0.0014 |
| update | | 0.0010 |
| add -s manager | 25 | 0.0005 |
| add -s secretary | 5 | 0.0001 |

Таблица из *Reliability Engineering Handbook*

# Графическое представление

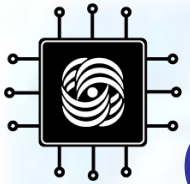
- Графическое представление состоит из:
  - *Узлов* – атрибутов операций
  - *Ветвей* – значения атрибутов
  - *Вероятность использования:* ассоциируется с ветвями




**Графическое представление может быть получено из табличного и наоборот.**

# Режим функционирования

- Режим функционирования – шаблон использования системы и/или условий окружения которые нужно различать во время тестирования, так как они могут приводить к отказам
- Пример: пиковые и свободные режимы работы телекоммуникационной системы
- Одна и та же операция может использоваться в разных режимах функционирования с разными вероятностями
- Профиль функционирования системы должен включать все важные режимы функционирования



# Сколько это стоит?

- Средняя стоимость построения профиля функционирования для проекта среднего размера (10 разработчиков, 100 тыс. сток кода, интервал разработки в 18 месяцев) составляет около 1 человеко-месяца
- Один и тот же профиль функционирования может быть использован (с минимальными изменениями) разными версиями одной программы без увеличения стоимости

# Построение профиля функционирования

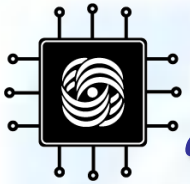
1. Выявить режимы функционирования
2. Выявить инициатора операции
3. Выбрать представление
4. Построить список функций и операций
5. Определить частоты использования каждой операции
6. Задать вероятности использования каждой операции, разделив частоты на общее число вызовов

# 1. Режим функционирования (1)

- Определить все возможные режимы функционирования и выбрать только наиболее важные на основании таких критериев как:
  - Время (время года, день недели, время дня)
  - Тип пользователя (покупатель или пользователь)
  - Опыт пользователя (новичок или эксперт)
  - Условия работы (режим перегрузки, обычный)
  - Ограничения функциональности
  - Критичность (режим остановки атомных станций)

# 1. Режим функционирования (2)

- Примеры:
  - Время (часы пика и простоя телефонной станции)
  - Различные типы пользователей (пользователь, системный администратор, …)
  - Опыт пользователя (использование различных наборов команд новичками и экспертами в системе редактирования видео)
  - Ограничения функциональности (такие операции как вызов справки в банкомате или защита соединения в транзакционной системе должны быть доступны всегда)



# 2. Инициаторы операций

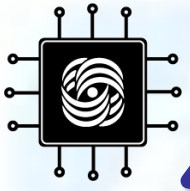
- Возможные инициаторы (субъекты из схем случаев использования):
  - Пользователи систем (любой, кто инициирует событие, включая системного администратора и пользователей)
  - Внешние системы (другие системы, взаимодействующие с разрабатываемой системой)
  - Собственные подсистемы (задачи поддержки и администрирования: аудит, резервное копирование данных, …)

# 3. Выбор представления

- Используйте табличное представление если каждой операции соответствует небольшое количество атрибутов
- Используйте графическое представление если большинство операций описывается несколькими атрибутами
- На практике может быть удобным использование смешанного представления или конвертации одного представления в другое

# Пример представления

- Пусть для разрабатываемого приложения выполняется следующее:
  - 80% используют систему для личных нужд, 20% - для работы.
  - 20% пользователей подписаны на бесплатные обновления в течение года,  5% - в течение 3х лет. Остальные на обновление не подписаны.
  - 90% пользователей установило обновление безопасности, остальные 10% этого не сделали.
- Каким будет функциональный профиль системы?

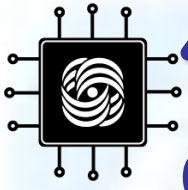


# 4. Список операций

- **Разбейте каждый режим функционирования на отдельные операции.**
  1. **Число операций**
  2. **Явные и неявные операции**
  3. **Изначальный список операций**
  4. **Переменные окружения**
  5. **Финальные список операций**

# 4.1 Число операций

- Стоимость подсчёта надёжности и тестирования прямо пропорциональна общему числу операций
- Обычно от нескольких десятков до сотен
- Число операций увеличивается вместе с размером проекта, числом режимов функционирования, числом ключевых входных переменных, …
- Обычно лучше оставить только те операции, которые покрывают все значительно отличающиеся виды обработки



# 4.2 Явные или неявные операции (1)

- Ключевая входная переменная - входная переменная сразу для нескольких операций, чьё значение важно для возможности отличать эти операции между собой
- Неявный профиль выражается как множество подпрофилей, каждый из которых включает ключевую входную переменную и соответствующую ей вероятность
- Явный профиль – множество наборов значений для всех ключевых переменных
- Обычно профиль легче выразить в неявной форме

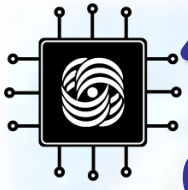
# 4.2 Явные или неявные операции (2)

- Пример: **неявный профиль**

| Входная переменная A | | Входная переменная B | |
|---|---|---|---|
| значение | вероятность | Значение | Вероятность |
| A1 | 0.6 | B1 | 0.7 |
| A2 | 0.3 | B2 | 0.2 |
| A3 | 0.1 | B3 | 0.1 |

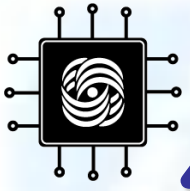
- Пример: **явный профиль:**

| | | | | | |
|---|---|---|---|---|---|
| A1B1 | 0.42 | A1B2 | 0.12 | A1B3 | 0.06 |
| A2B1 | 0.21 | A2B2 | 0.06 | A2B3 | 0.03 |
| A3B1 | 0.07 | A3B2 | 0.02 | A3B3 | 0.01 |



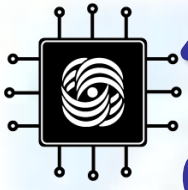
# 4.3 Изначальный список операций

- Изначальный список операций – пробный список, обычно разделённый по режимам функционирования и инициаторам операций
- Источники, которые могут быть полезны при составлении списка начальных операций:
  - Требования к системе(основной источник)
  - Случаи использования, модели работы
  - Диаграммы процесса работы
  - Черновики инструкций для пользователей
  - Предыдущие версии программы
  - Прототипы



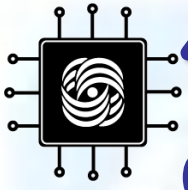
# 4.4 Переменные окружения

- Переменные окружения описывают условия, влияющие на процесс выполнения программы
- Пример: сетевой траффик, конфигурация оборудования и операционной системы
- Некоторые переменные окружения могут отображаться на режимы функционирования
- Перечислите все переменные окружения, которые могут привести к разной реакции программы, а затем выберете из них те, чьё влияние на поведение программы наиболее велико



# 4.5 Финальный список операций (1)

- Финальный список операций основан на сужении изначального списка операций с помощью следующих критериев:
  - Режим функционирования
  - Переменные окружения
  - Ключевые входные переменные
  - Зависимости между переменными
- Если между отдельными операциями существует связь (одна операция происходит, если происходит другая), то лучше объединить эти операции



# 4.5 Финальный список операций (2)

- Между функциями и операциями могут появиться значительные отличия
- Некоторые функции (решающие несвязанные задачи) могут отображаться на одни и те же операции (реализующие разную обработку)
- Пример: Перемещение объекта в памяти (функция) может быть отображена в операции создания, копирования и удаления

# 5. Частоты операций (1)

- Частоты использования операций строятся независимо для каждого режима функционирования на основе данных системного журнала, ручной сборки данных и опыта
- Если на этой стадии обнаруживаются связанные операции, их лучше объединить

# 5. Частоты операций (2)

- Пример: **Система управления звонками**

| Operation | Occurrence rate (operations per hour) |
|---|---|
| Phone number entry | 10,000 |
| Add subscriber | 50 |
| Delete subscriber | 50 |
| Process voice call, no pager, answer | 18,000 |
| Process voice call, no pager, no answer | 17,000 |
| Process voice call, pager, answer | 17,000 |
| Process voice call, pager, answer on page | 12,000 |
| Process voice call, pager, no answer on page | 10,000 |
| Process fax call | 15,000 |
| Audit section of phone number database | 900 |
| Recover from hardware failure | 0.1 |
| Total | 100,000 |

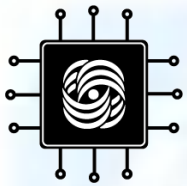
# 6. Вероятности операций

- **Вероятности операций определяются с помощью деления индивидуальных частот операций на общее число их вызовов.**

| Operation | Occurrence probability |
|---|---|
| Process voice call, no pager, answer | 0.18 |
| Process voice call, no pager, no answer | 0.17 |
| Process voice call, pager, answer | 0.17 |
| Process fax call | 0.15 |
| Process voice call, pager, answer on page | 0.12 |
| Process voice call, pager, no answer on page | 0.10 |
| Phone number entry | 0.10 |
| Audit section of phone number database | 0.009 |
| Add subscriber | 0.0005 |
| Delete subscriber | 0.0005 |
| Recover from hardware failure | 0.000001 |
| Total | 1.0 |

# Документация

| *Режим функционирования* | *Инициатор / ключевая входная переменная / Переменная окружения* | *Список операций* | *Частота / вероятность операции* |
|---|---|---|---|
| | | **1.** | |
| | | **2.** | |
| | | **3.** | |
| | | **1.** | |
| | | **2.** | |
| | | **1.** | |
| | | **2.** | |
| | | **3.** | |
| | | **4.** | |



# Спасибо за внимание!