НАДЁЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лекция 6:
Надёжность системы

ВМиК МГУ им. М.В. Ломоносова,
Кафедра АСВК, Лаборатория Вычислительных Комплексов
к.ф.-м.н., доцент Волканов Д.Ю.

О чём этот курс

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

План лекции

- Надёжность последовательных и параллельных систем
- Блок-схема расчёта надёжности
- Анализ эксплуатационной безопасности
- Анализ типов и последствий отказов
- Анализ дерева неисправностей

Эпиграф

- *В жизни нет гарантий, существуют одни вероятности*

  Том Клэнси

Что такое надежность?

- Надежность - вероятность того, что программная система будет работать по спецификации.
- Дефект – код, приводящий к ошибке.
- Ошибка – проявление дефекта.
- Ошибки приводят к отказам – неверной работе программы.
- Адрес страницы:
- http://lvk.cs.msu.su/~dimawolf/SoftwareReliability

Модели надёжности

- Граф надёжности (Reliability Graph)
- Дерево неисправностей (Fault Tree model)
- Блок-схема расчёта надёжности (Reliability Block Diagram)

Системная Надёжность (1)

- Система состоит из модулей
- Модуль состоит из компонент
- Компоненты могут иметь
  - Разную надёжность
  - Разную зависимость между ними
- Надёжность системы – это функция надёжности компонентов и зависимости между компонентами

Надежность последовательных систем

- Система состоит из n компонент, соединенных друг с другом
- Отказ каждого компонента приводит к отказу всей системы

- Надежность системы меньше надежности ее компонентов (т.к. $R_k \leq 1$).

Суммирование надежностей

- Надежность системы может быть выражена через надежности ее компонентов, если они отказывают независимо.
- Для систем из отдельных компонентов:

$$R = \prod_{k=1}^{Q_p} R_k$$

Q_p число компонентов

R_k надежность компонента

- Надежности компонентов (R_k) должны быть заданы в одном интервале.

Суммирование надежностей

- Используя связь между надежностью и частотой ошибок

$$\lambda = \frac{-\ln R}{\tau} \qquad or \quad R = e^{-\lambda\tau}$$

- Получаем:

$$\lambda = \sum_{k=1}^{Q_p} \lambda_k$$

- Т.е. общая частота есть сумма частот отдельных компонентов

Пример системы

- Система из 4 независимых компонентов
  - $R_1 = 0.95$
  - $R_2 = 0.87$
  - $R_3 = 0.82$
  - $R_4 = 0.73$

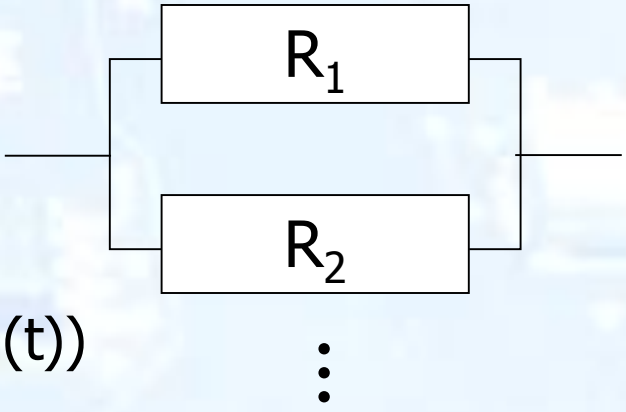
  $R_{system} = 0.95 \times 0.87 \times 0.82 \times 0.73 = 0.4947$
- Надежность системы меньше надежности каждого из компонентов

Надежность параллельных систем

- Система состоит из n компонентов, работающих параллельно
- Только отказ всех компонентов приводит к отказу всей системы.

$$\text{Unreliability } F_p(t) = \prod_{i=1}^{n} F_i(t)$$

$$\text{Reliability } R_p(t) = 1 - \prod_{i=1}^{n} F_i(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t))$$



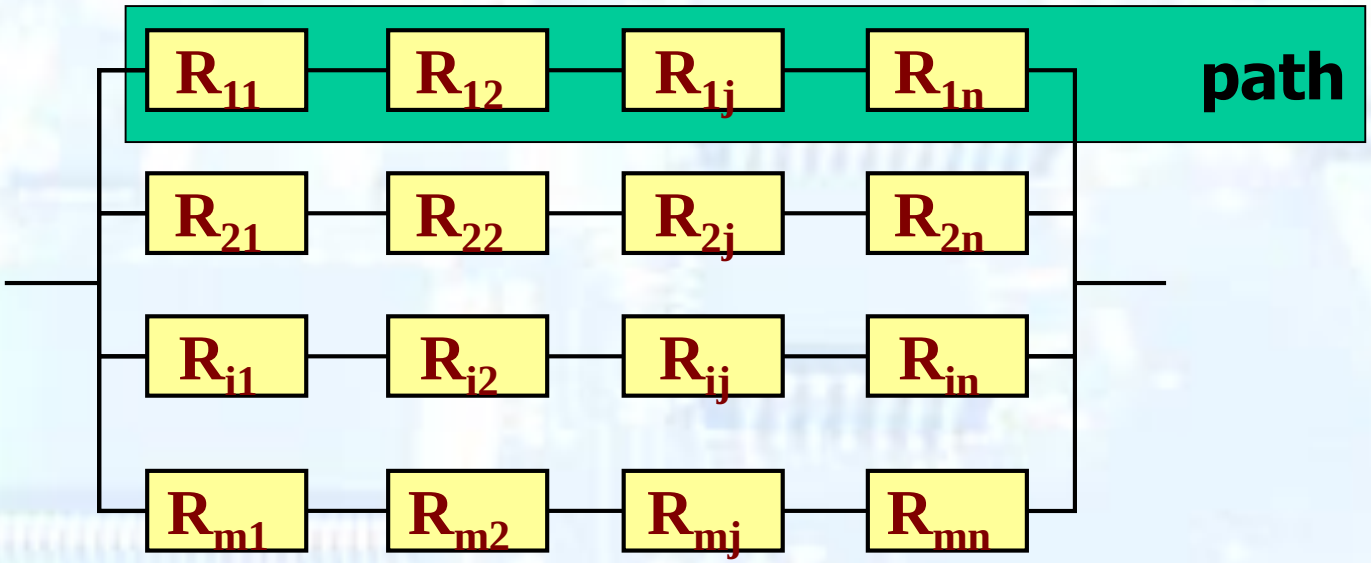
Пример параллельной системы

- Система состоит из 4 параллельно работающих компонентов
  - $R_1 = 0.95$
  - $R_2 = 0.87$
  - $R_3 = 0.82$
  - $R_4 = 0.73$

$$R_{system} = 1 - ((1 - 0.95) \times (1 - 0.87) \times (1 - 0.82) \times (1 - 0.73)) = 0.9996$$

- Надежность системы больше, чем надежности отдельных компонентов.

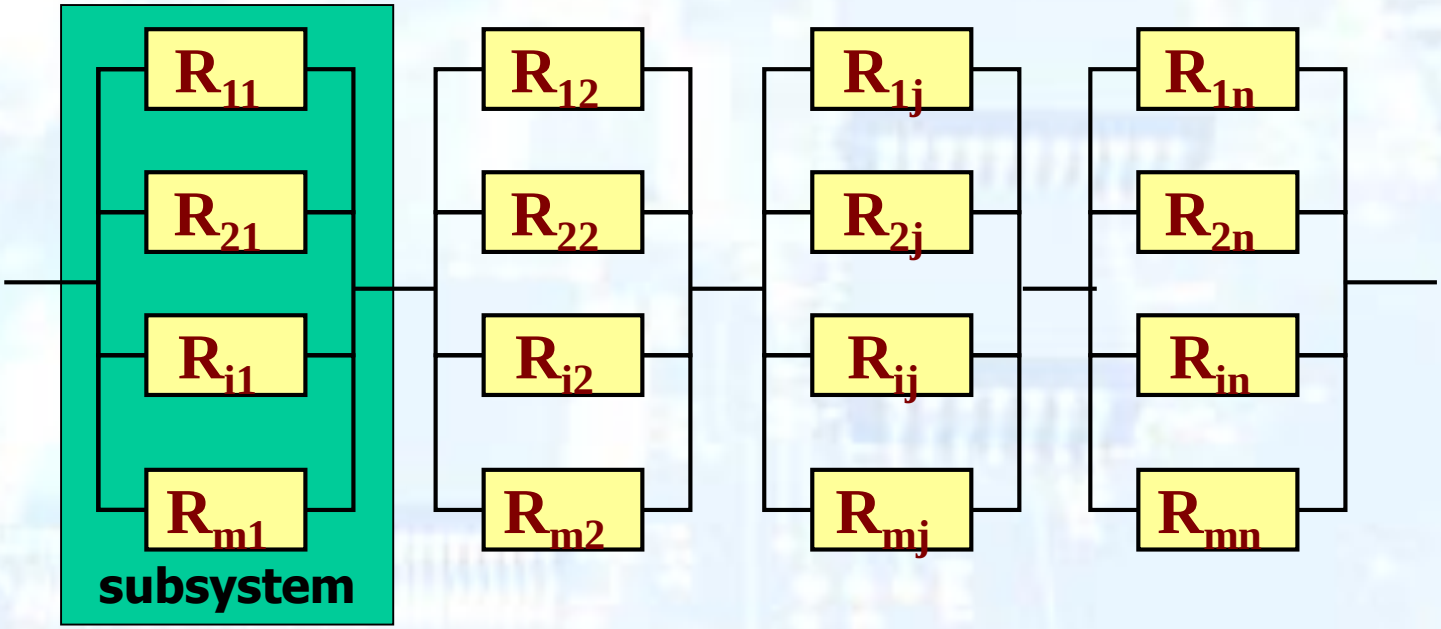
Параллельно-последовательные системы



$$\text{Path reliability } R_i = \prod_{j=1}^{n} R_{ij}$$

$$\text{System reliability } R = 1 - \prod_{i=1}^{m} (1 - R_i) = 1 - \prod_{i=1}^{m} \left(1 - \prod_{j=1}^{n} R_{ij}\right)$$

Параллельно-последовательные системы



Subsystem reliability $R_j = 1 - \prod_{i=1}^{m} (1 - R_{ij})$

System reliability $R = \prod_{j=1}^{n} R_j = \prod_{j=1}^{n} \left[1 - \prod_{i=1}^{m} (1 - R_{ij}) \right]$

Другие архитектуры

- Однополосный мост

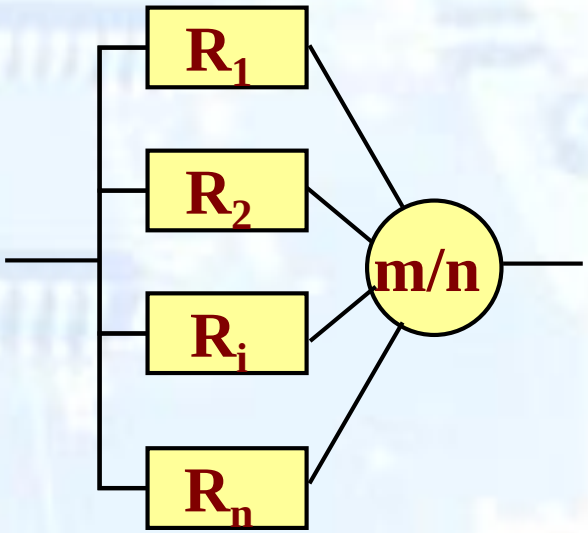
- Двуполосный мост

Горячее резервирование

- Использует параллельность.
- Все компоненты работают одновременно
- Каждый компонент достаточен для корректной работы
- Только один компонент необходим для корректной работы
- Каждый компонент удовлетворяет требованиям к надежности
- Система отказывает, если отказывают все компоненты.

Система m – из – n

- Система из n компонентов
- Для корректной работы нужно не меньше m компонент ($m \le n$).
  - m=n: последовательная
  - m=1: параллельная
- Например, самолет с 4 моторами может лететь при отказе двух моторов



Предположение: Все компоненты имеют одинаковую надежность

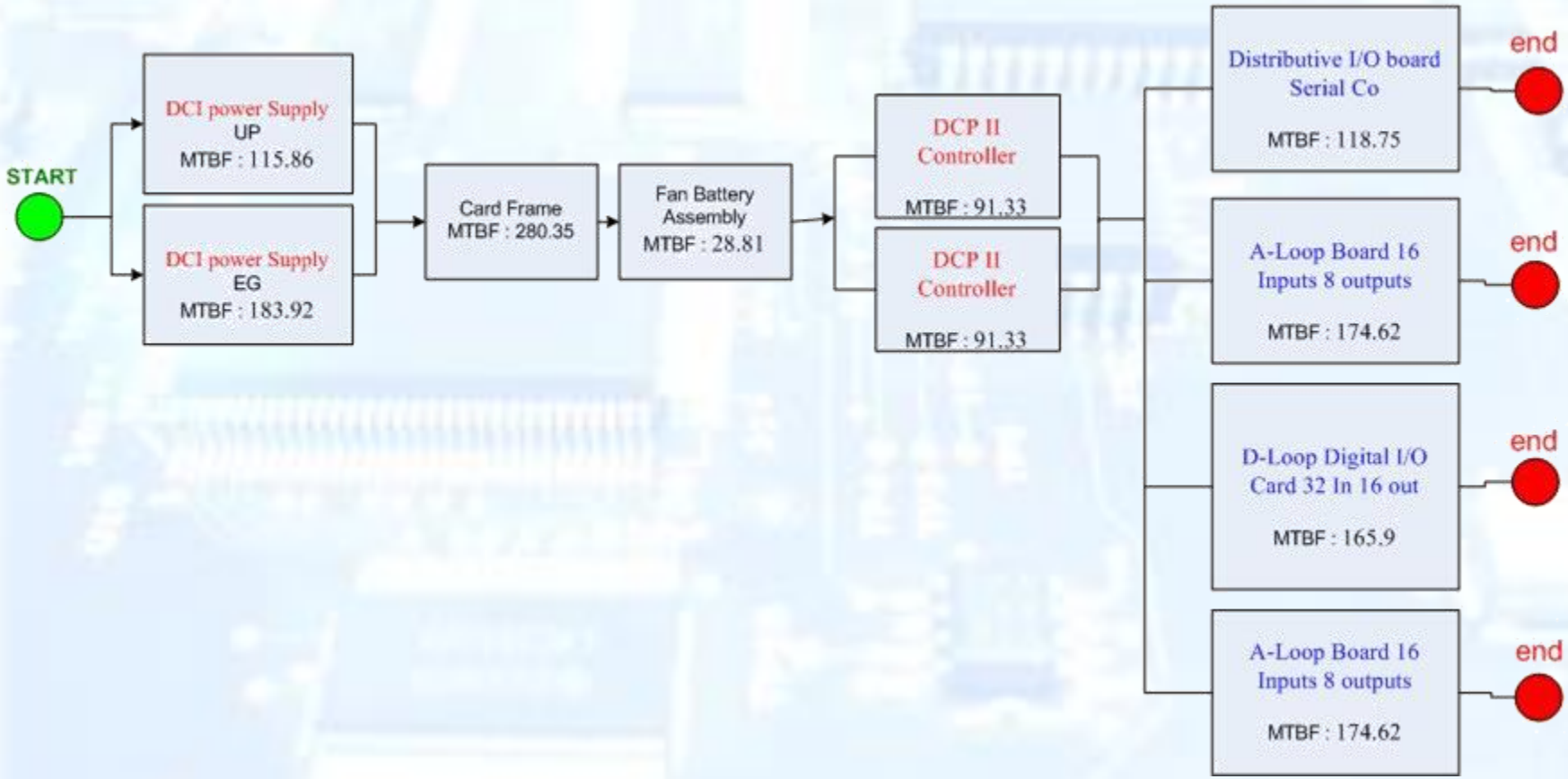
$$R_{sys}(t) = 1 - \sum_{i=0}^{m-1} \binom{n}{i} R(t)^i (1 - R(t))^{n-i}$$

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!}$$

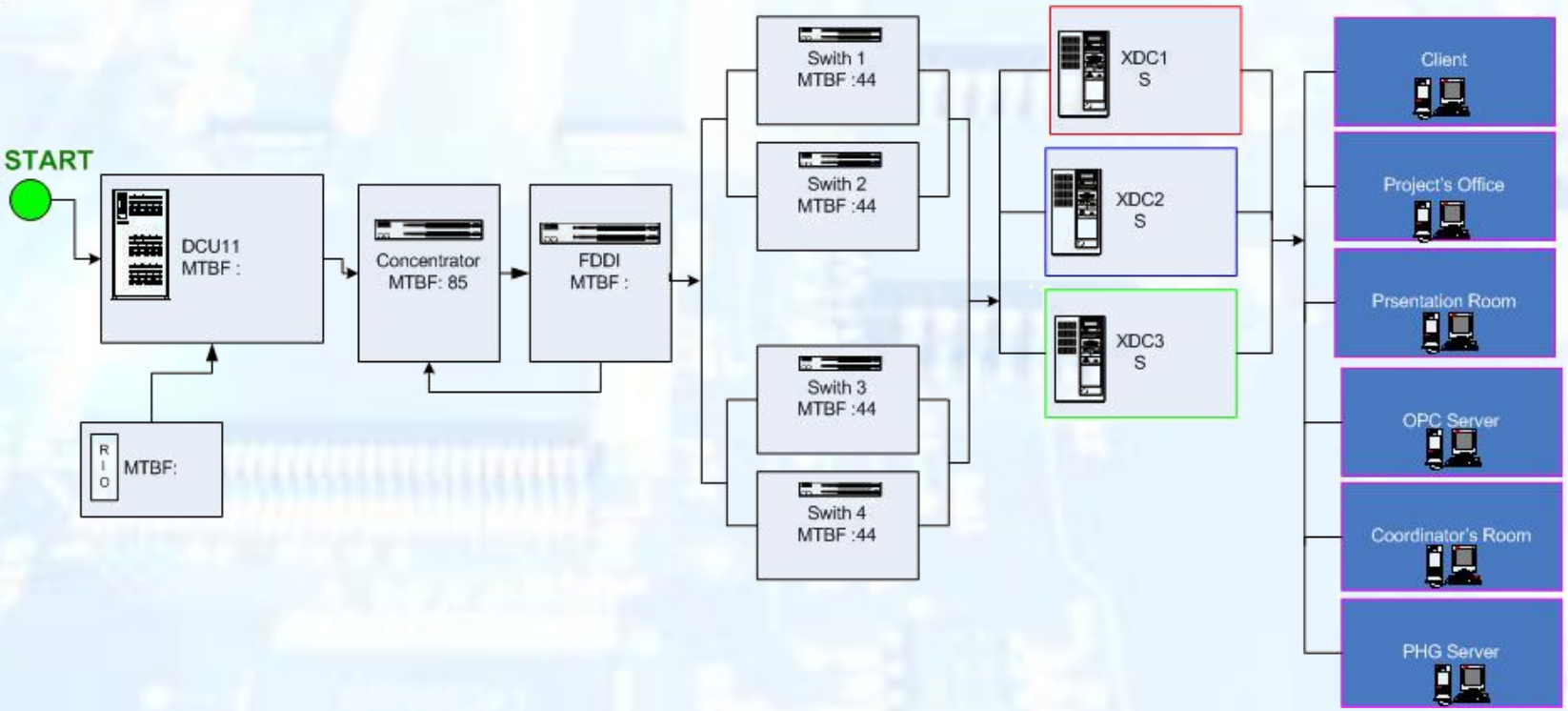
Блок-схема надежности (RBD)

- RBD показывает, как соединены компоненты системы для обеспечения надежности
- Самые частые конфигурации – последовательные и параллельные компоненты.
- В последовательной конфигурации отказ каждого компонента приводит к отказу системы. Общая надежность меньше надежности отдельных компонентов.
- В параллельной конфигурации отказ всех компонентов приводит к отказу системы. Общая надежность больше надежности отдельных компонентов.
- Система содержит некую комбинацию этих конфигураций
- Анализ диаграмм необходим для вычисления надежности систем.

RBD: пример /1


START
DCI power Supply
UP
MTBF : 115.86
DCI power Supply
EG
MTBF : 183.92
Card Frame
MTBF : 280.35
Fan Battery
Assembly
MTBF : 28.81
DCP II
Controller
MTBF : 91.33
DCP II
Controller
MTBF : 91.33
Distributive I/O board
Serial Co
MTBF : 118.75
A-Loop Board 16
Inputs 8 outputs
MTBF : 174.62
D-Loop Digital I/O
Card 32 In 16 out
MTBF : 165.9
A-Loop Board 16
Inputs 8 outputs
MTBF : 174.62
end
end
end
end

RBD: пример /2


START
DCU11
MTBF :
Concentrator
MTBF: 85
FDDI
MTBF :
R
I
O
MTBF:
Swith 1
MTBF :44
Swith 2
MTBF :44
Swith 3
MTBF :44
Swith 4
MTBF :44
XDC1
S
XDC2
S
XDC3
S
Client
Project's Office
Prsentation Room
OPC Server
Coordinator's Room
PHG Server

RBD: когда и как?

- Когда использовать RBD?
  - Когда нужно оценить надежность сложной системы и найти в системе уязвимые места
- Как использовать RBD?
  - Нарисовать схему
  - Найти надежность системы
  - Вычислить другие характеристики (доступность, время работы etc)
- Есть автоматизированные средства создания и анализа диаграмм, интегрированные с другими методами, например, с FTA.

RBD: плюсы и минусы

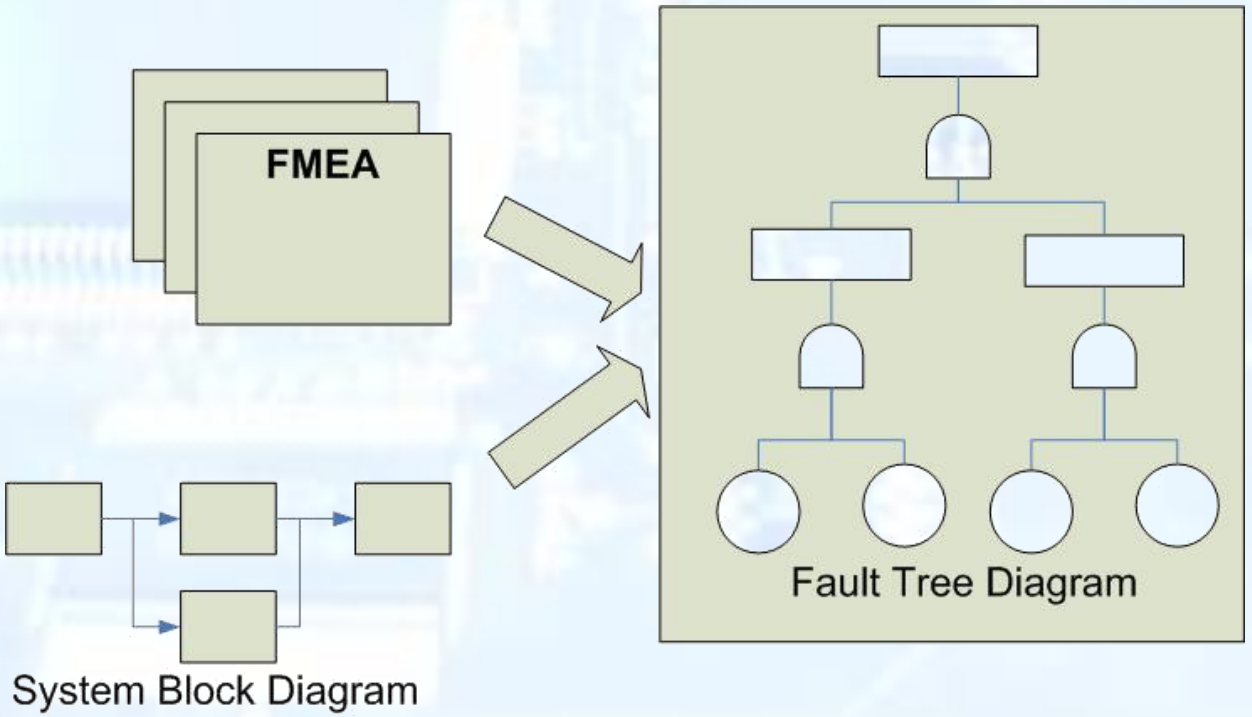
- RBD – самый простой метод визуализации характеристик надежности.
- Преимущества:
  - Помогает найти цели по надежности
  - Оценивает влияние отказа компонента на работу системы
  - Позволяет простой анализ различных сценариев работы
  - Подсчитывает MTBF системы
  - Уменьшает стоимость исправления проблем в больших системах
  - Позволяет анализировать разные конфигурации
  - Указывает на потенциальные проблемы в системе
  - Определяет чувствительность системы к отказам компонентов
- Недостаток в том, что сложные конфигурации (ожидание, ветвление, распределение нагрузки и.т.д.) не могут быть описаны RBD.

Дерево неисправностей (FTA)

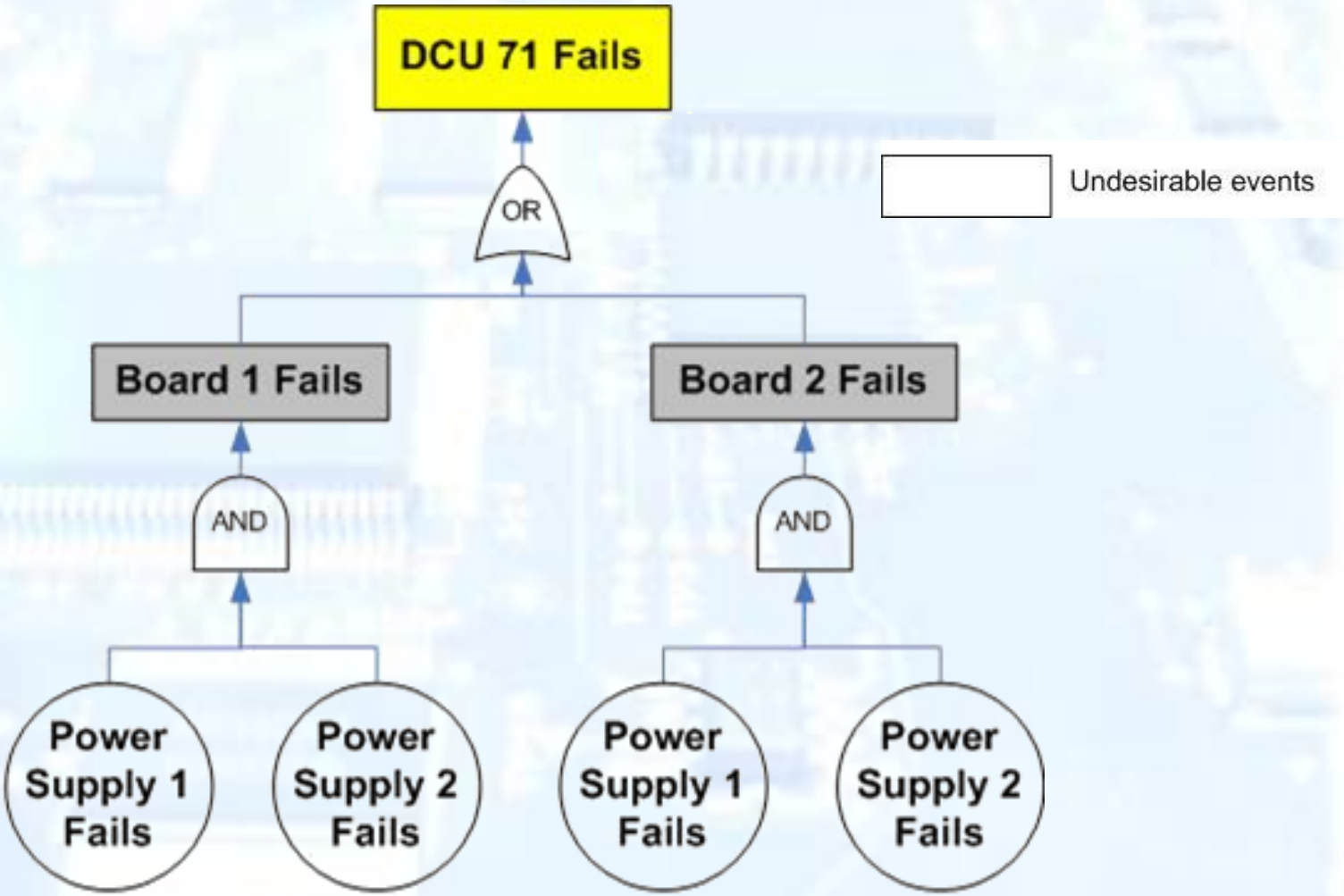
- Дерево неисправностей – графическое представление основных ошибок и отказов в системе, их причин и способов их устранения. Деревья позволяют находить уязвимые места при проектировании новой системы или анализе существующей. Они также помогают находить методы решения проблем.
- Дерево неисправностей можно рассматривать как графическую модель поведения системы, ведущего к нежелательному событию. Поведение определяется событиями и условиями и записывается логическим выражением.
- Анализ деревьев неисправностей полезен как при проектировании новых систем и программных продуктов, так и при поиске недостатков в существующих. Они помогают найти корень проблемы и исправить ее.

Шаги в FTA

- FMEA определяет условия, приводящие к нежелательным событиям. Затем определяется, какие локальные ошибки могут привести к отказу системы. FMEA отвечает на вопрос, “Что может пойти не так?” даже если спецификация выполнена. **Для работы FTA необходимо иметь FMEA.**



FTA: пример


DCU 71 Fails
OR
Undesirable events
Board 1 Fails
Board 2 Fails
AND
AND
Power Supply 1 Fails
Power Supply 2 Fails
Power Supply 1 Fails
Power Supply 2 Fails

FTA: плюсы и минусы

- **Плюсы:** FTA дают осмысленные данные, позволяющие оценить и улучшить надежность системы. Они также дают оценку необходимости и эффективности резервирования.
- **Минусы:** Ограничение FTA в том, что нежелательное событие необходимо предвидеть и знать основные источники проблемы. Это может быть очень долго и дорого. Успех зависит от способностей аналитика, работающего над этой задачей.

Методы обеспечения отказоустойчивости

- Обнаружение ошибок
- Диагностическое тестирование
- Изоляция ошибок
- Маскировка ошибок
- Корректирование ошибок
- Устранение ошибок

Требования к компонентам

Организация резервирования

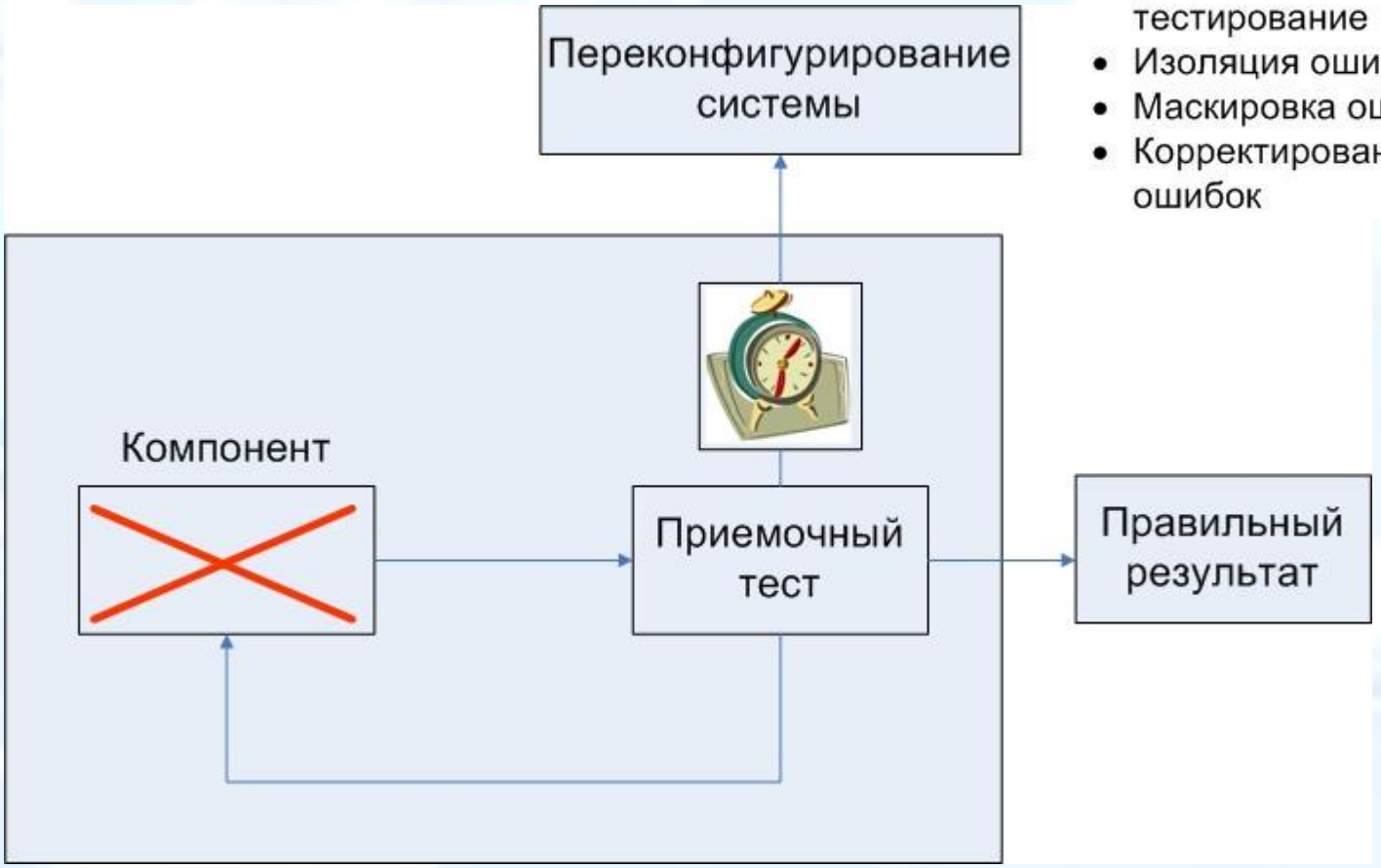
- Общие подходы
  - Организация глобального времени
  - Выделение изолированных регионов
  - Отказы в разделяемых компонентах
- Программное резервирование
  - Пространственное резервирование
  - Временное резервирование
    - Функциональный сдвиг во времени
    - Информационный сдвиг во времени
- Аппаратное резервирование
  - Пространственное резервирование
  - Кодирование

Механизмы обнаружения ошибок

- Программные
  - Приемочные тесты
  - Отказоустойчивые алгоритмы
  - Проверки
    - Временные
    - Кодовые
    - Реверсные
    - Семантические
    - Структурные
- Аппаратные
  - Диагностическое тестирование

Приемочные тесты

- Диагностическое тестирование
- Изоляция ошибок
- Маскировка ошибок
- Корректирование ошибок



Отказоустойчивые алгоритмы. Попарное тестирование

- Диагностическое тестирование
- Изоляция ошибок
- Маскировка ошибок
- Корректирование ошибок

Отказоустойчивые алгоритмы. Голосование

- Диагностическое тестирование
- Изоляция ошибок
- Маскировка ошибок
- Корректирование ошибок

Механизмы устранения ошибок

- Программные
  - Одноверсионное программирование
    - Контрольная точка и перезапуск
    - Парные прогоны
  - Многоверсионное программирование
    - Восстановление блоками
    - N-версионное программирование
    - N-самотестируемое программирование
- Аппаратные
  - Резервирование компонент
  - Переконфигурирование системы

Схемы голосования

голосователь сравнивает результаты двух и более функционально эквивалентных компонентов ПО и определяет корректныйрезультат

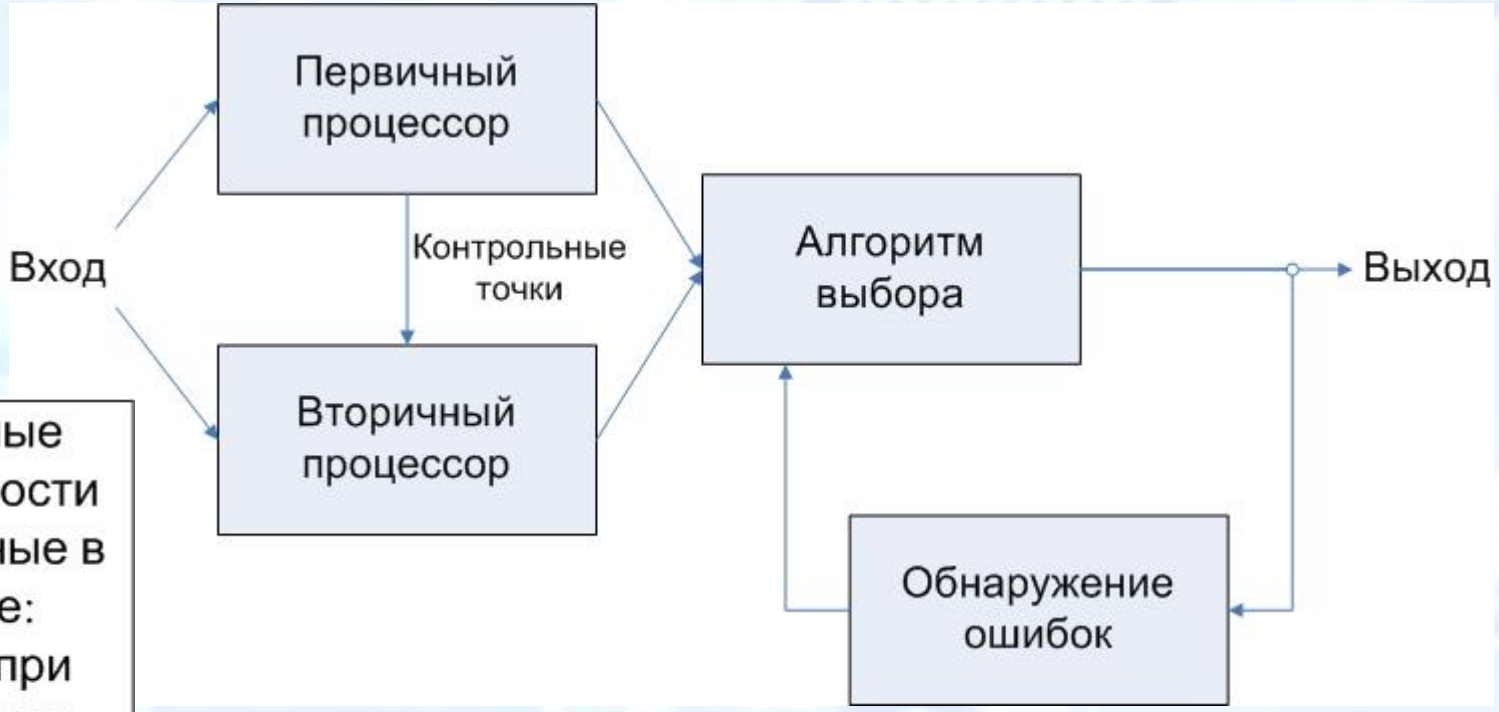
- Схемы голосования:
  - Большинство
  - Консенсус
  - 2-из-N

Контрольная точка и перезапуск

∀ временные неисправности:

- врем. э/м помеха
- приработка устройства

Парные прогоны



∀ временные неисправности + постоянные в аппаратуре:

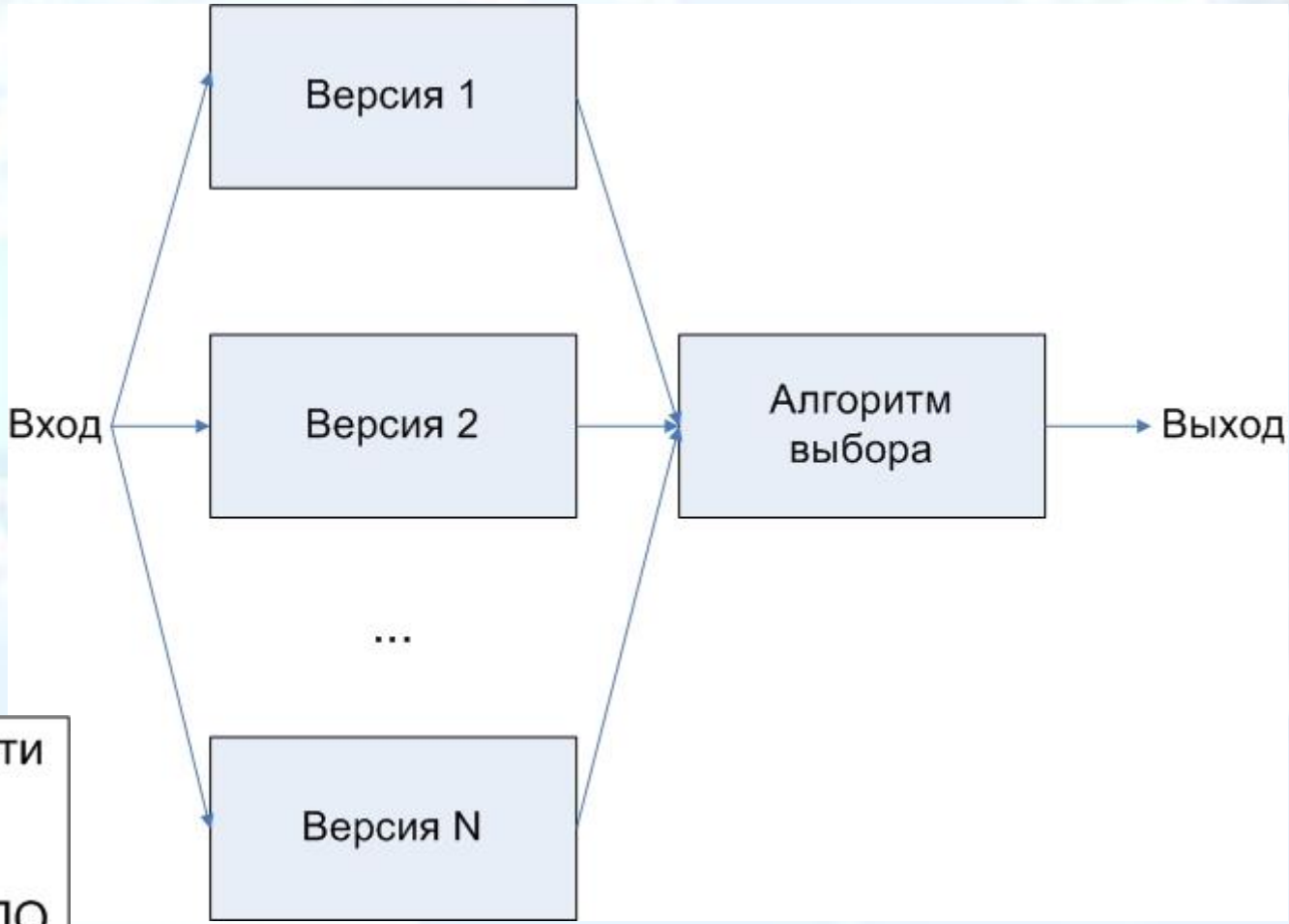
- ошибка при разработке аппаратуры
- ошибка ввода

Восстановление блоками

∀ временные неисправности + постоянные в ПО:

- ошибка при разработке ПО
- врем. э/м помеха

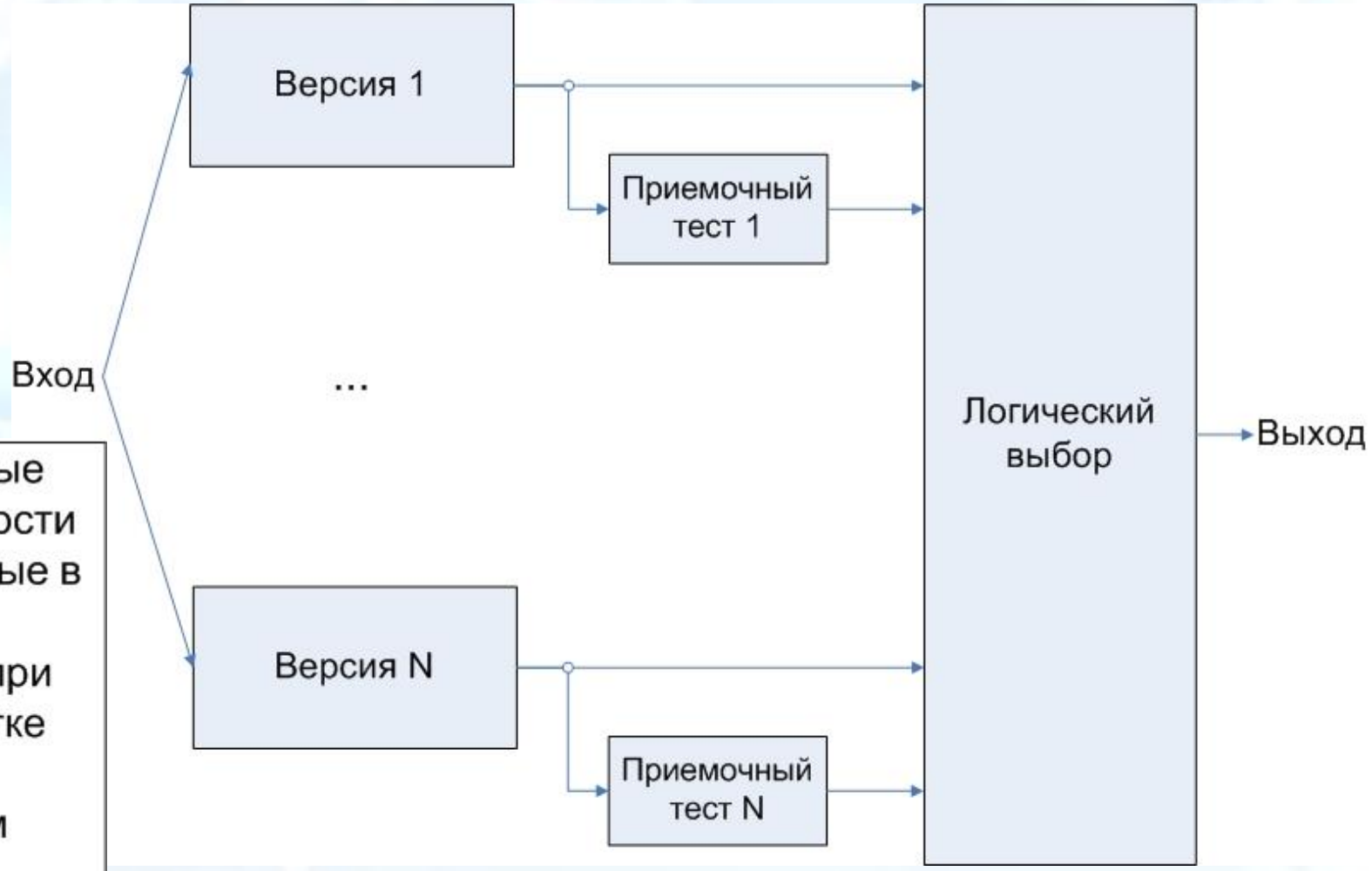
N-версионное программирование



∀ неисправности в ПО:

- ошибка при разработке ПО
- ошибка ввода

N-самотестируемое программирование



∀ временные неисправности + постоянные в ПО:

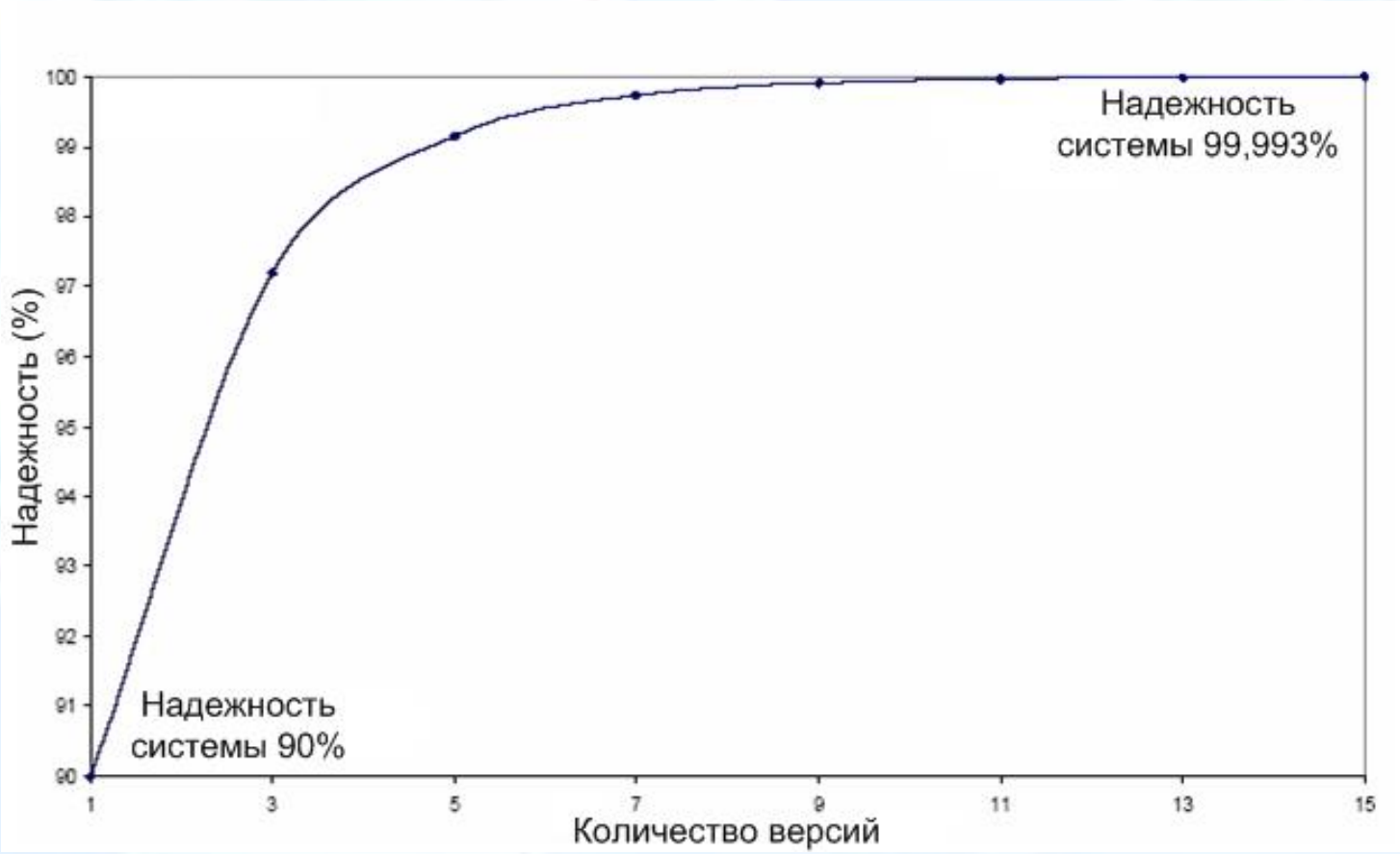
- ошибка при разработке ПО
- врем. э/м помеха

N-версионное программирование. Примеры

Год	Проект и спонсор	Кол-во версий	Требуемое разнообразие	ЯП
1975-76	Текстовый редактор Software Engineering	27	Разработчики	PL/1
1977-78	Решение ДУ Software Engineering	16	Разработчики + 3 алгоритма	PL/1
1979-83	Аэропортовый планировщик NSF	18	Разработчики + 3 спецификации	PL/1
1984-86	Отказоустойчивый датчик NASA	5	Разработчики	Pascal
1986-88	Беспилотная посадка самолета Sperry Flight Systems	6	Разработчики + 6 языков	Pascal, Ada, Modula-2, C, Prolog, T

Космический шаттл

Зависимость надежности от количества версий



Спасибо за внимание!