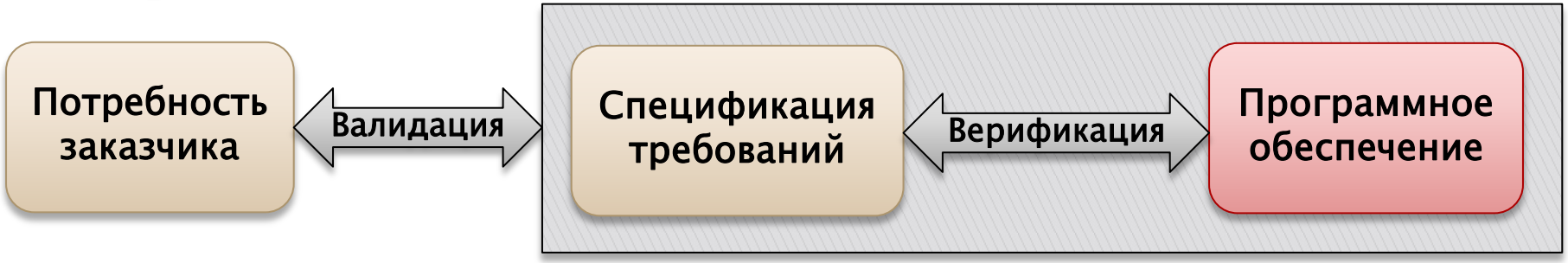
Обеспечение качества программных систем

Обеспечение качества ПО

- Методы, направленные на проектирование качественного ПО
  - Формальные спецификации
  - Синтез ПО на основе спецификаций и моделей (MDD, etc)
  - Контрактное программирование (Design by contracts)
  - И т.п.
- Методы, направленные на обеспечение качества существующего ПО

Обеспечение качества ПО. Терминология



- **Верификация** - подтверждение на основе представления объективных свидетельств того, что установленные требования были выполнены
- **Валидация** - подтверждение на основе представления объективных свидетельств того, что требования, предназначенные для конкретного использования или применения, выполнены, декларируемые свойства и характеристики подтверждаются, а поставленная цель (предназначение системы, комплекса, устройства и т. д.) достигнута.

Методы обеспечения качества ПО

- По используемым формализмам
  - Формальные методы
  - Неформальные методы
- По необходимости запуска анализируемой программы
  - Динамические
  - Статические
  - Гибридные
- По уровню автоматизации
  - Ручные
  - Автоматизированные
  - Автоматические

Методы обеспечения качества

- Динамические методы
  - Тестирование
  - Профилирование
  - Динамический анализ
    - Мониторинг
    - Анализ трасс исполнения
    - Контрактное программирование
  - …
- Статические методы
  - Формальная верификация
    - Дедуктивная верификация
    - Model checking (методы проверки модели)
  - Статический анализ
  - Трансформации программ
    - Рефакторинги
    - Модификации
  - Аудит

Формальная (дедуктивная) верификация

- Верификация - подтверждение соответствия конечного продукта функциональной спецификации
- Формальная верификация – доказательство корректности с помощью формальных методов
- Используемые методы и мат. аппарат
  - Пропозициональные логики
  - Темпоральные логики
  - Формальные семантики
  - Формальные преобразования программ
  - Формальные спецификации
  - Логика Хоара
  - Сепарационная логика (separation logic)
  - И т.п.
- Наиболее известные подходы:
  - Верификация методом Хоара (на основе троек Хоара)
  - Верификация по Флойду

Формальная (дедуктивная) верификация

- Достоинства:
  - В случае успеха – в программе нет ошибок!
- Недостатки:
  - Формальные спецификации на порядок сложнее программ
  - Для большинства программ задача формального доказательства корректности – очень трудоёмка
  - Для некоторых случаев – задача формального доказательства корректности – неразрешима
- В реальных системах при формальной верификации рассматривают часть системы и частичные спецификации
- Редко применяется для обеспечения качества программных систем общего назначения

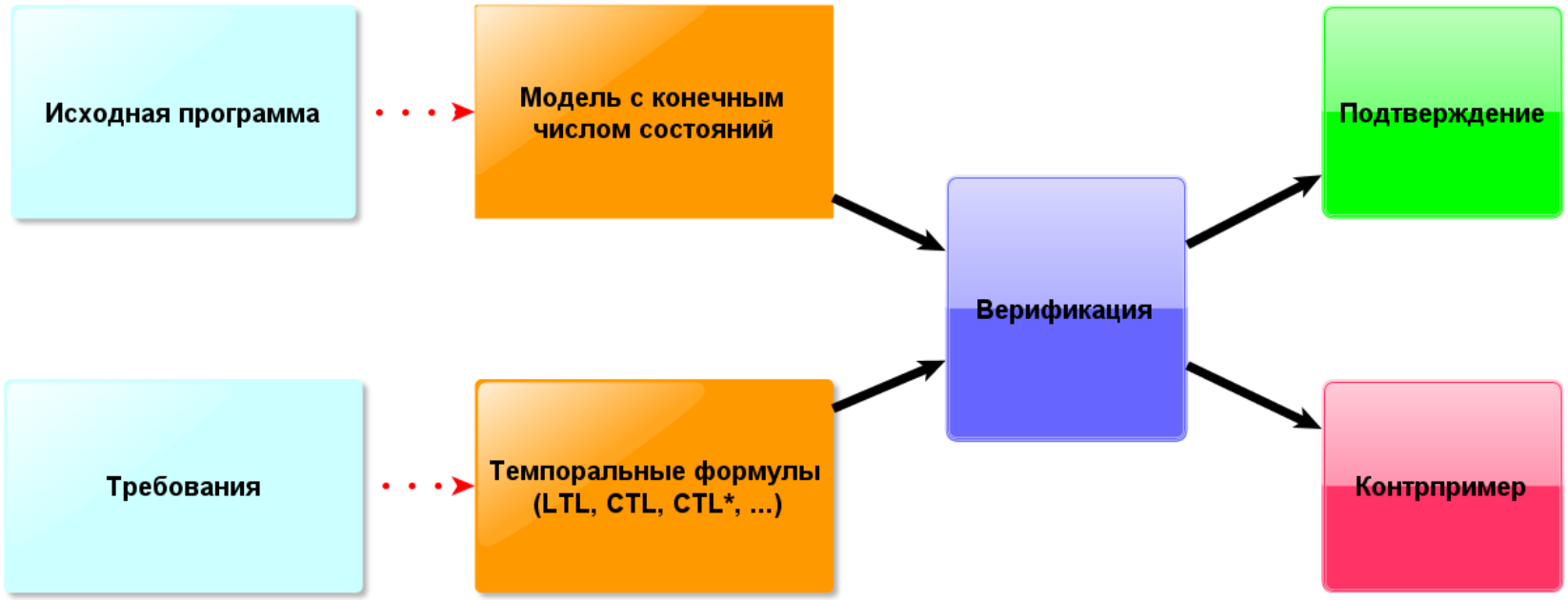
Метод проверки моделей

- Проверка модели, проверка на модели, model checking
- Метод формальной верификации для систем с конечным числом состояний
- Позволяет проверить, удовлетворяет или нет система некоторому свойству (требованию)

Верификация по методу Model Checking

- Исследуемая система приводится к модели с конечным числом состояний (например, модель Крипке)
- Проверяемые свойства представляются формулами темпоральной логики (LTL, ALTL, CTL, CTL* и т.д.)
- Проверка модели – формальная проверка выполнимости формулы на модели
  - Результат проверки:
    - Формула выполняется
    - Формула не выполняется. Контрпример.
  - Существуют методы проверки систем с $10^{100\text{-}200}$ состояний

Верификация по методу Model Checking



Верификация по методу Model Checking

- Ограничения
  - Проверяются свойства, связанные только с корректностью смены состояний
  - Не все свойства представляются в виде темпоральных формул
  - В общем случае задача - NP-полная
  - В общем случае неформализуется переход от реальной системы к модели с конечным числом состояний
- Программные средства:
  - SPIN
  - NuSMV
  - …

Обеспечение качества ПО путем обнаружения ошибок

- Ошибки
  - Функциональные ошибки
  - Нефункциональные ошибки (дефекты)
- Проявление дефектов:
  - Сбои ПО
  - Зависания ПО
  - Аварийное завершение ПО
  - Уязвимости
  - Отсутствие проявлений
  - …

Обнаружение программных дефектов

- Динамические методы:
  - Тестирование
  - Динамический анализ
- Статические методы:
  - Статический анализ
  - Верификация (частично)

Статический анализ

- Использует исходный код ПО для анализа
- Применяется для
  - Форматирования программ
  - Вычисления программных метрик
  - Оптимизации программ
  - Распараллеливания программ
  - Преобразования программ
  - Обфускации программ
  - Деобфускации программ
  - **Обнаружения дефектов**
  - …

Статический анализ

- Цель – обнаружение дефектов в программном коде
- Использует исходный код ПО для анализа
- Позволяет проанализировать все возможные трассы исполнения
- Позволяет проанализировать все наборы входных данных
- Может быть полностью автоматизирован
- Позволяет обнаружить *нефункциональные* дефекты

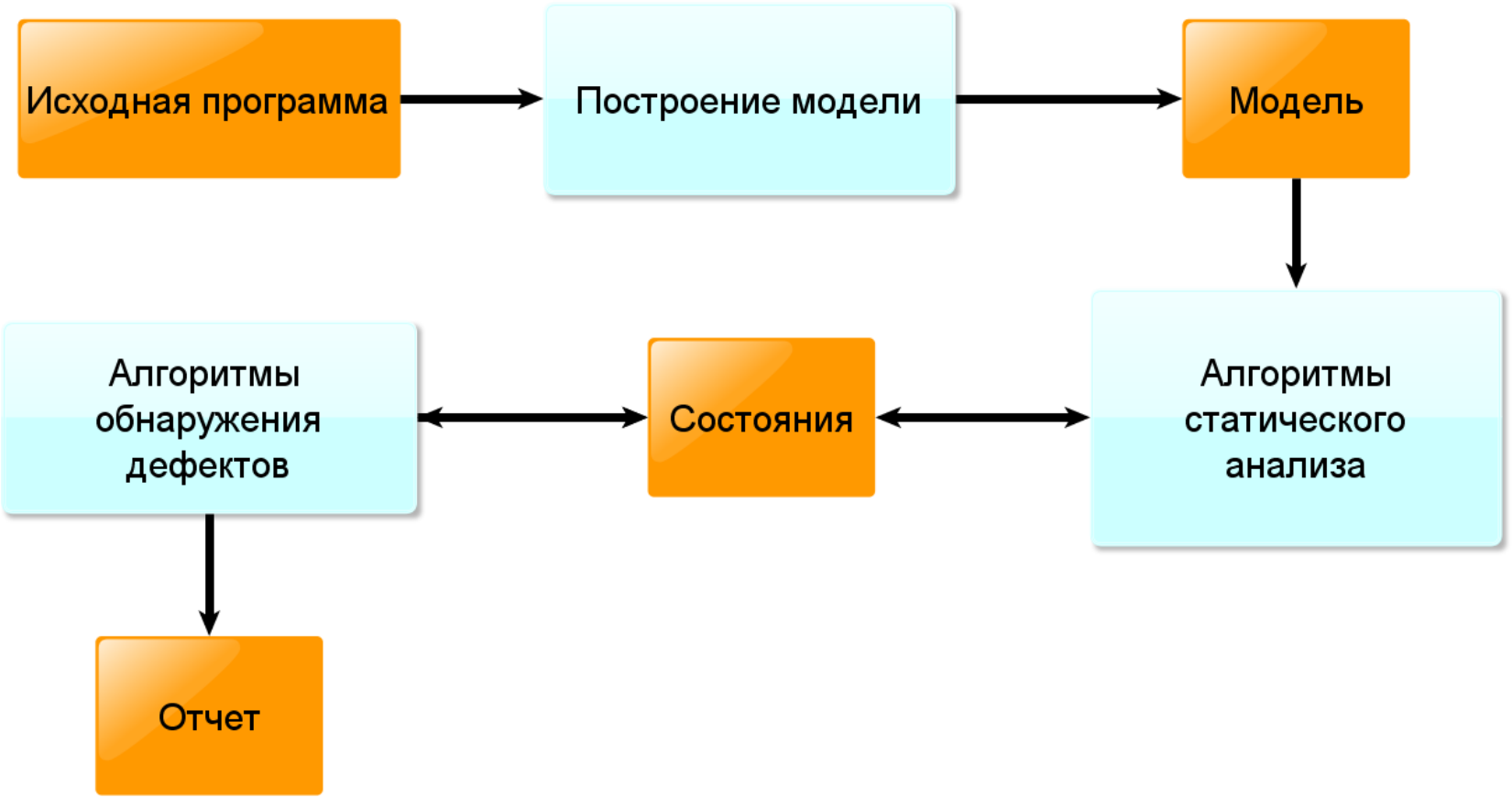
Программные дефекты

- Основные виды дефектов:
  - Неправильная работа с буферами:
    - Переполнение буферов
    - Выход за границу массива
    - …
  - Неправильная работа с динамической памятью:
    - Утечки памяти
    - «Висячие» указатели
    - Разыменование нулевого указателя
    - …
  - Использование неинициализированных переменных
  - Ошибки работы с объектами
  - Ошибки работы с библиотечными функциями
  - Ошибки работы со строками
  - Арифметические ошибки
  - И т.п.

Статический анализ

- Используемые методы
  - Интервальный анализ
  - Поиск достижимости
  - Анализ указателей
  - Ресурсный анализ
  - Сигнатурный анализ
  - …

Схема проведения СА



Статический анализ

- Достоинства СА:
  - Обнаружение дефектов на ранних стадиях
  - Сокращение стоимости разработки, отладки, тестирования, сопровождения
- Недостатки СА:
  - Невозможность обнаруживать функциональные ошибки
  - Недостаточность информации о путях выполнения -> наличие ложных обнаружений
  - Невозможность обнаружить все ошибки статически
  - Высокие требования к вычислительным ресурсам

Статический анализ

- Программные средства анализа кода и поиска дефектов:
  - IBM Rational Code Analyzer
  - Coverity Prevent
  - Fortify 360
  - Klocwork
  - Flexlint/PCLint
  - Splint
  - Microsoft PREFix/PreFast
  - ParaSoft C++Test
  - Frama-C
  - Aegis (http://digiteklabs.ru/aegis)
  - …(более 20)

Тестирование

- Тестирование – процесс выполнения программы с целью выявления ошибок
- Полная проверка программы – исчерпывающее тестирование. В реальности практически не осуществимо
- Тестирование обеспечивает:
  - Обнаружение ошибок
  - Демонстрацию соответствия функций программы ее назначению
  - Демонстрацию реализации требований к характеристикам программы
  - Отображение надежности как одной из характеристик качества
- **Тестирование не может показать отсутствие ошибок!**
- **Тестирование может только показать наличие ошибок!**

- *«Тестирование программ может служить для доказательства наличия ошибок, но никогда не докажет их отсутствия!» Э.Дейкстра. Заметки по структурному программированию*

Тестирование

- Виды тестирования
  - Функциональное тестирование
    - Тестирование «черного ящика»
  - Структурное тестирование
    - Тестирование «белого ящика»
  - Смешанное тестирование
    - Тестирование «серого ящика»

Структурное тестирование

- Тестирование «белого ящика»
- Тестирование производится на основе знания внутренней структуры программы
- Проверяется корректность построения элементов программы и их взаимодействие друг с другом

Функциональное тестирование

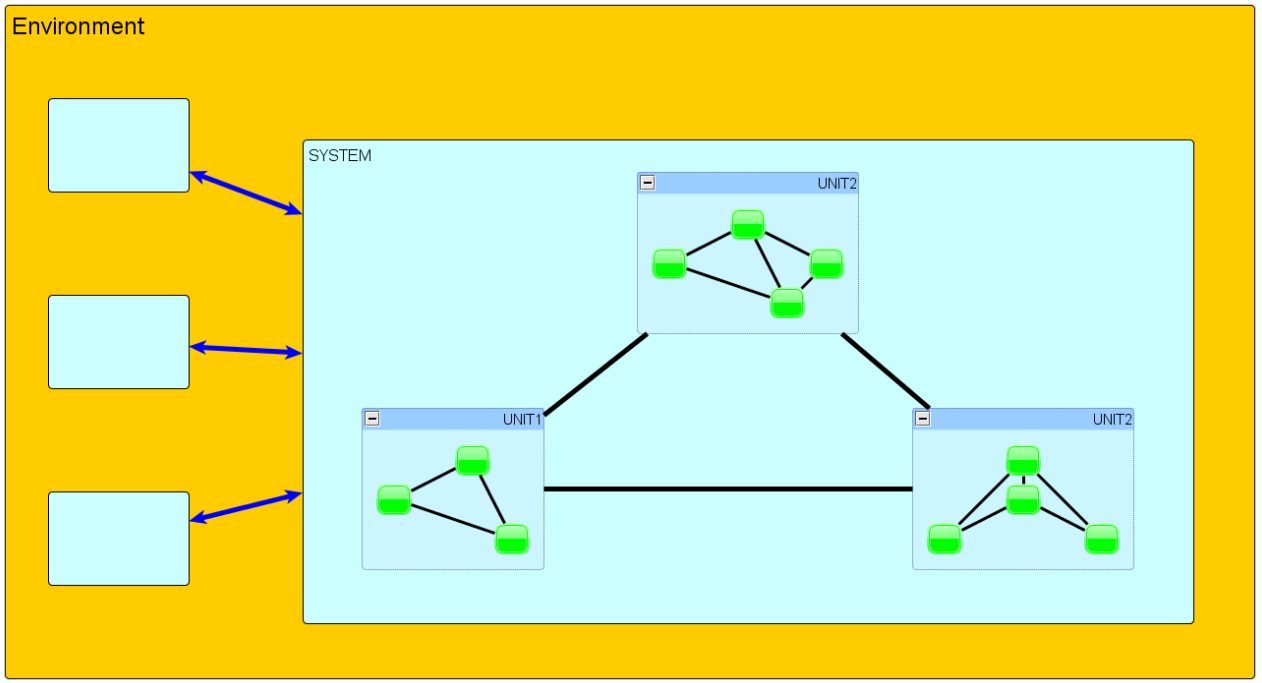
- Тестирование «черного ящика»
- Проверка соответствия проверяемого объекта спецификации
- Поведение программной системы оценивается путем подачи воздействий и анализа реакции
- Желательно найти
  - Набор входных данных, при которых система ведет себя аномально
  - Набор выходных данных, демонстрирующих дефекты программы

Функциональное тестирование

- Позволяет обнаруживать следующие ошибки:
  - Некорректные или отсутствующие функции
  - Ошибки интерфейса
  - Ошибки во внутренних структурах данных или доступе к внешним БД
  - Ошибки характеристик ПО
  - Ошибки инициализации
  - Ошибки завершения
- Обычно тестирование «чёрного ящика» применяют на поздних стадиях тестирования

Организация процесса тестирования

- Основные составляющие процесса:
  - Тестирование элементов
  - Тестирование интеграции
  - Системное тестирование
  - Тестирование правильности



Тестирование элементов

- Тестирование модулей, unit-тестирование
- Обычно проводится **разработчиком**
- Основные подходы
  - *Тестирование «белого ящика»*
  - Тестирование «чёрного ящика»

Тестирование элементов

- Тестированию должны быть подвергнуты
  - Внутренние структуры данных
  - Независимые траектории функционирования
  - Циклы
  - Ветвления
  - Рекурсии
  - Траектории обработки ошибочных ситуаций
  - Граничные условия
  - Интерфейс модуля
  - И т.п.

Тестирование элементов

- Необходимо обеспечить полное тестовое покрытие – каждый участок кода выполняется хотя бы одним тестовым случаем
- Полное тестовое покрытие не гарантирует выполнение всех путей программы!
- Разработка модульных тестов может:
  - Вестись после кодирования
  - Вместе с кодированием
  - До кодирования (*)

Тестирование интеграции

- Цель – протестировать ошибки интерфейсов между компонентами
  - Потеря данных между модулями
  - Взаимное влияние модулей
  - Невыполнение общей функции системы
  - Накапливание ошибок при интеграции
  - Несоблюдение протокола
  - И т.п.
- Используется тестирование «черного ящика»

Системное тестирование

- Тестирование системных функций
- Тестирование нефункциональных требований
  - Тестирование восстановления
  - Тестирование безопасности
  - Стресс-тестирование
  - Тестирование производительности

Тестирование правильности

- Цель – подтвердить выполнение спецификации требований
- Используется тестирование «черного ящика»
- Проверяются выполнение всех требований. Создается список несоответствий (недостатков)
- Каждый недостаток сопровождается описанием его проявления

Тестирование правильности

- Альфа-тестирование
  - Тестированию подвергается альфа-версия программной системы
  - Тестирование производится разработчиком при участии представителей заказчика
- Бета-тестирование
  - Тестированию подвергается бета-версия программной системы
  - Тестирование производится заказчиком или специально отобранными конечными пользователями

Регрессионное тестирование

- Регрессионное тестирование – повторный запуск ранее созданных тестов
  - для проверки того, что старые ошибки не проявились вновь
  - для проверки того, что новые изменения не испортили функциональность программы
- У ошибок есть свойство повторяться. Причины повторения
  - Не до конца исправлена
  - Ошибочно появилась при работе с СКВ
  - Код был переписан и ошибка повторена
  - Проявилась при других внешних условиях
  - И т.п.
- Поэтому – тестовые случаи накапливаются и прогоняются при изменениях программы

Программные средства тестирования

- HP Mercury
- IBM Rational
- Borland
- Compuware
- UniTesk
- …

Методы обеспечения качества ПО

	Х-ка качества	**Проблема**	**Обеспечение качества**
1	Функциональность	Функциональные ошибки, несоответствие спецификации	Верификация Тестирование
2	Надежность	Низкая надежность Наличие уязвимостей	Статический анализ Тестирование
3	Практичность	Сложность использования	Тестирование*
4	Эффективность	Проблемы с производительностью, ресурсами	Тестирование Профилирование Динамический/статический анализ
5	Сопровождаемость	Сложность сопровождения, модификации	Рефакторинг Документирование
6	Мобильность	Несоответствие стандартам, Сложность адаптации	Аудит, рефакторинг Статический анализ