



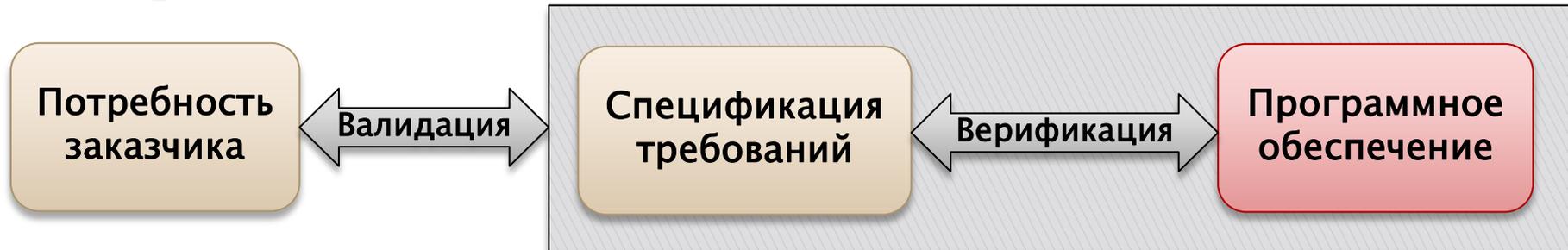
Обеспечение качества программных систем

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 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 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

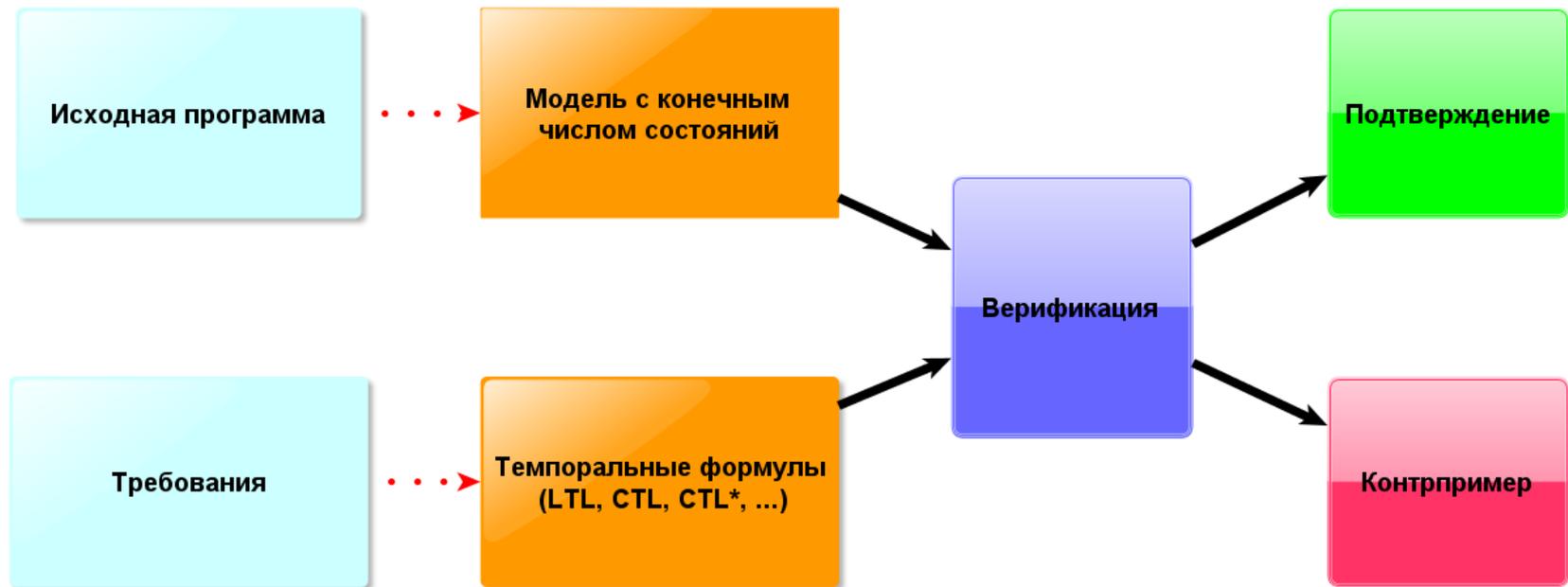
Метод проверки моделей

- ▶ 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, STL, STL* и т.д.)
- ▶ Проверка модели – формальная проверка выполнимости формулы на модели
 - Результат проверки:
 - Формула выполняется
 - Формула не выполняется. Контрпример.
 - Существуют методы проверки систем с $10^{100-200}$ 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

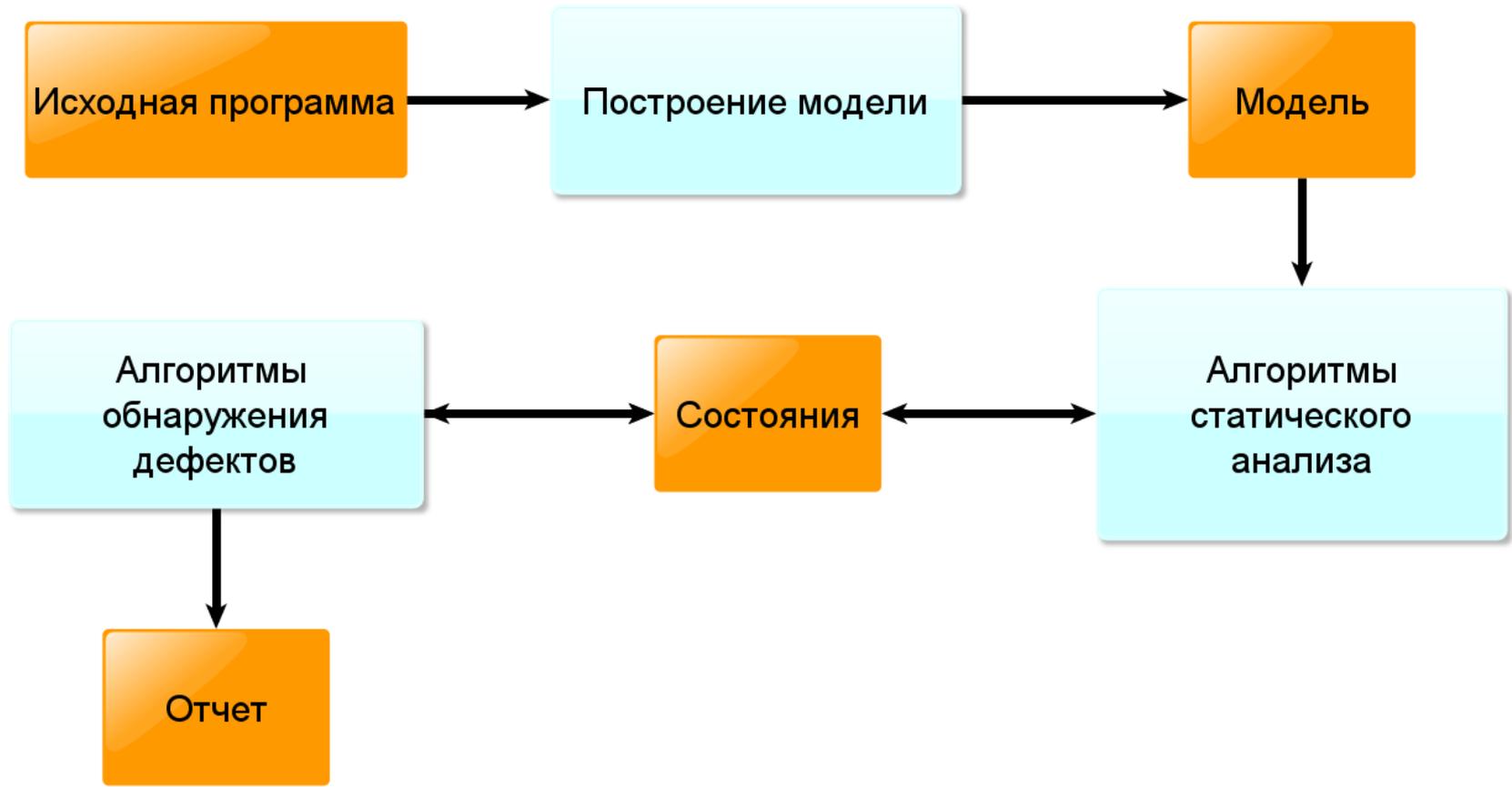
Программные дефекты

- ▶ Основные виды дефектов:
 - Неправильная работа с буферами:
 - Переполнение буферов
 - Выход за границу массива
 - ...
 - Неправильная работа с динамической памятью:
 - Утечки памяти
 - «Висячие» указатели
 - Разыменованное 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://digitek-labs.ru/aegis>)
 - ...(более 20)

Трансформации программ

- ▶ Формальные трансформации программ:
 - Эквивалентные трансформации –рефакторинг
 - Неэквивалентные трансформации (модификации, эволюции)

Рефакторинг

- ▶ **Рефакторинг** – преобразование программной системы:
 - изменяющее (улучшающее) внутреннюю структуру программного
 - не меняющее его видимое поведение
- ▶ Цель рефакторинга – улучшить внутреннюю структуру программы, избавиться от неудачного, устаревшего или трудно поддерживаемого кода
- ▶ Наиболее часто используется объектно-ориентированный рефакторинг – рефакторинг ОО-систем
- ▶ Рефакторинг – сложная рутинная операция, потенциально способная вносить дополнительные ошибки в программный код

Автоматизированный рефакторинг

- ▶ **Автоматизированный рефакторинг** – набор процедур, встроенных в среду разработки, позволяющих автоматизировано осуществлять преобразование программ с одновременным контролем корректности модификации
- ▶ Автоматизированный рефакторинг реализован:
 - MS Visual Studio
 - Borland Developer Studio (Delphi, C++)
 - NetBeans
 - Idea
 - Eclipse
 - И т.п.
- ▶ Перспективное направление – проведение рефакторинга на основе метрического покрытия программы

Методы обеспечения качества ПО

	X-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