

# Качество программного обеспечения

# Тенденции индустрии разработки ПО

- ▶ Объем программ растет
- ▶ Время разработки новых версий ПО сокращается
- ▶ Все большее число задач решается программно
- ▶ ПО все больше используется при решении критически важных задач
- ▶ Существенная часть ПО является свободной и поставляется "as is"

# Известные примеры программных ошибок

- ▶ **США, 1962 год.** Гибель несущего аппарата "Маринер-1".
  - Причина – ошибка в одном символе программы
    - $DO\ 100\ I = 1, 10$
    - $DO100I = 1.10$
- ▶ **США, 1987 год.** Ускоритель Therac-25. Переоблучение пациентов онкокlinik.  
Причина – ошибка «race condition»
- ▶ **США, 1991 год.** Комплекс Patriot. Погибло 28 чел.
  - Причина – ошибка округления
- ▶ **Европа, 1996 год.** Ракета Ариан-5. Ущерб \$7 млрд.
  - Причина – использование унаследованного кода

# Известные примеры программных ошибок

- ▶ **США, 2003 год.** Сбой в энергосистеме (Blackout). Ущерб 7-10 млрд.\$.
  - Причина – ошибка «race condition»
- ▶ **Израиль.** Сбой навигационной системы самолетов F16.
  - Причина - высотомер выдавал значение  $\leq 0$ .
- ▶ **Голландия, 2000 год.** Остановка доменной печи 29 февраля. Гибель 6 человек.
  - Причина - ошибка в процедуре расчета даты.
- ▶ ...

# Что такое качественное ПО?

- ▶ Вариант 1: ПО, в котором отсутствуют ошибки
- ▶ Вариант 2: ПО, соответствующее требованиям



# Качество ПО

- ▶ ГОСТ Р ИСО/МЭК 9126 (ISO 9126):
  - **Качество ПО** – весь объем признаков и характеристик программной продукции, который относится к её способности удовлетворять установленным и предполагаемым свойствам
  - **Характеристики качества** – набор свойств программной продукции, по которым её качество описывается и оценивается

# Характеристики качества ПО

- ▶ Функциональность (Functionality)
- ▶ Надежность (Reliability)
- ▶ Практичность (Usability)
- ▶ Эффективность (Efficiencies)
- ▶ Сопровождаемость (Maintainability)
- ▶ Мобильность (Portability)

# Характеристики качества.

## Функциональность

- ▶ **Функциональность** - набор атрибутов характеризующий, соответствие функциональных возможностей ПО набору требуемой пользователем функциональности.
- ▶ Подхарактеристики:
  - Пригодность (соответствие требуемому набору функций)
  - Корректность (правильность, точность)
  - Способность к взаимодействию (с другими компонентами и системами)
  - Согласованность (соответствие стандартам)
  - Защищенность



# Характеристики качества.

## Надежность

- ▶ **Надежность** - набор атрибутов, относящихся к способности ПО сохранять свой уровень качества функционирования в установленных условиях за определенный период времени.
- ▶ Подхарактеристики:
  - Стабильность (число отказов при ошибках)
  - Устойчивость к ошибкам
  - Восстанавливаемость
  - Доступность/Готовность

# Характеристики качества.

## Практичность

- ▶ **Практичность** (удобство) - набор атрибутов, относящихся к объему работ, требуемых для исполнения и индивидуальной оценки такого исполнения определенным или предполагаемым кругом пользователей.
- ▶ Подхарактеристики:
  - Понятность (организации)
  - Изучаемость
  - Простота использования
  - Привлекательность

# Характеристики качества.

## Эффективность

- ▶ **Эффективность** - набор атрибутов, относящихся к соотношению между уровнем качества функционирования ПО и объемом используемых ресурсов при установленных условиях.
- ▶ Подхарактеристики:
  - Временная эффективность
  - Используемость ресурсов

# Характеристики качества.

## Сопровождаемость

- ▶ **Сопровождаемость** - набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций).
- ▶ Подхарактеристики:
  - Анализируемость
  - Изменяемость
  - Стабильность
  - Тестируемость



# Характеристики качества.

## Мобильность

- ▶ **Мобильность** (переносимость)- набор атрибутов, относящихся к способности ПО быть перенесенным из одного окружения в другое.
- ▶ Подхарактеристики:
  - Адаптируемость
  - Простота установки (внедрения)
  - Соответствие стандартам (подчинение стандартам или соглашениям, относящимся к мобильности)
  - Взаимозаменяемость

# Качество ПО.

## Заинтересованные лица

- ▶ Пользователь/заказчик
  - Функциональность
  - Надежность
  - Практичность
  - Эффективность
- ▶ Разработчик/руководитель
  - Сопровождаемость
  - Мобильность

# Стандарты качества ПО

- ▶ **Мировые стандарты**
  - ISO/IEC 9126. Software engineering - Product quality:
    - Part 1: Quality model
    - Part 2: External metrics
    - Part 3: Internal metrics
    - Part 4: Quality in use metrics
  - ISO/IEC 25000:2005. Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE)
- ▶ **Российские стандарты**
  - ГОСТ 28195-89 «Оценка качества программных средств»
  - ГОСТ Р ИСО/МЭК 9126-93 «Оценка программной продукции»

# Оценка качества ПО.

## Программометрика

- ▶ Программометрика - наука о количественном оценивании свойств программного обеспечения
- ▶ Программная метрика - мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций



# Использование метрик для оценки качества

- ▶ Функциональность – метрики тестирования
- ▶ Практичность – метрики эргономики
- ▶ Сопровождаемость – **метрики кода**
- ▶ Мобильность – **метрики кода**
- ▶ Надежность – метрики тестирования, динамические методы
- ▶ Эффективность – только динамические методы

# Метрики программного обеспечения

- ▶ Различные системы метрик позволяют оценивать различные характеристики ПО:
  - Локализация
  - Инкапсуляция
  - Информационная закрытость
  - Наследование
  - Абстракция
  - Связность объектов программы
  - Сложность
  - Размер
  - И т.п.

# Метрики программного обеспечения

- ▶ Существует множество систем метрик:
  - Метрики Холстеда
  - Метрики Л. Константейна и Э. Йордана
  - Метрики Л. Отта и Б. Мехра
  - Метрики Д. Биемена и Б. Кенга
  - Метрики С. Чидамбера и К. Кемерера
  - Метрики М. Лоренца и Д. Кидда
  - Метрики Ф. Абреу
  - Метрики Р. Байндера
  - И т.п.

# Метрики Ф. Абреу (MOOD)

- ▶ Набор метрик MOOD (Metrics of Object-Oriented Design)
- ▶ Разработаны Фернандо Абреу в 1994 г.
- ▶ Цели:
  - Описание ОО-механизмов: инкапсуляция, наследование, полиморфизм, обмен сообщений
  - Формализованность метрик
  - Независимость от размера ПО
  - Независимость от ЯП



# Метрики MOOD

- ▶ Фактор закрытости метода (MHF)
- ▶ Фактор закрытости свойства (AHF)
- ▶ Фактор наследования метода (MIF)
- ▶ Фактор наследования свойства (AIF)
- ▶ Фактор полиморфизма (POF)
- ▶ Фактор сцепления (COF)

# MOOD. Фактор закрытости метода

- ▶ MHF – Method Hiding Factor
- ▶ Показывает долю скрытых методов в программе
- ▶  $MHF = \sum_{1..N}(Mh_i) / \sum_{1..N}(Mh_i + Mv_i)$ 
  - $Mh_i$  – число скрытых неунаследованных методов класса  $i$
  - $Mv_i$  – число видимых неунаследованных методов класса  $i$

# MOOD. Фактор закрытости свойства

- ▶ AHF – Attribute Hiding Factor
- ▶ Показывает долю скрытых свойств в программе
- ▶  $AHF = \sum_{1..N}(Ah_i) / \sum_{1..N}(Ah_i + Av_i)$ 
  - $Ah_i$  – число скрытых неунаследованных свойств класса  $i$
  - $Av_i$  – число видимых неунаследованных свойств класса  $i$

# MOOD. Фактор наследования метода

- ▶ MIF – Method Inheritance Factor
- ▶ Показывает долю унаследованных непереопределенных методов в программе
- ▶  $MIF = \sum_{1..N}(MI_i) / \sum_{1..N}(MN_i + MI_i + MO_i)$ 
  - $MI_i$  – число унаследованных непереопределенных методов класса  $i$
  - $MN_i$  – число новых методов класса  $i$
  - $MO_i$  – число унаследованных переопределенных методов класса  $i$



# MOOD. Фактор наследования свойств

- ▶ AIF – Attribute Inheritance Factor
- ▶ Показывает долю унаследованных непереопределенных свойств в программе
- ▶  $AIF = \sum_{1..N}(AI_i) / \sum_{1..N}(AN_i + AI_i + AO_i)$ 
  - $AI_i$  – число унаследованных непереопределенных свойств класса  $i$
  - $AN_i$  – число новых свойств класса  $i$
  - $AO_i$  – число унаследованных переопределенных свойств класса  $i$

# MOOD. Фактор полиморфизма

- ▶ POF – Polymorphism factor
- ▶  $POF = \sum_{1..N}(MO_i) / \sum_{1..N}(MN_i * D_i)$ 
  - $MN_i$  – число новых методов класса  $i$
  - $MO_i$  – число унаследованных переопределенных методов класса  $i$
  - $D_i$  – количество потомков класса  $i$

# MOOD. Фактор сцепления

- ▶ COF – Coupling Factor
- ▶ Определяет долю пар классов, связанных отношением «клиент-поставщик»
- ▶  $COF = \sum_{i \in 1..N} \sum_{j \in 1..N} (C_{ij}) / (N \cdot (N-1))$ 
  - $C_{ij} = 1$ , если класс  $i$  имеет собственную ссылку на класс  $j$

# Аудит программного кода

- ▶ Аудит (review) – процесс контроля и оценки программного кода в процессе его эволюции
- ▶ Ручной аудит проводится экспертами в области программирования
- ▶ Автоматизированный аудит проводится на основе программных метрик
- ▶ Автоматизированный аудит является частью многих сред разработки



# Использование метрик

- ▶ Для проекта выбирается набор(ы) метрик
- ▶ Для каждой метрики:
  - Если метрика неоднозначна – доопределяется
  - Формируются эталонные значения
  - Значение метрик - нормируются
- ▶ Осуществляется непрерывный аудит программного кода
- ▶ Метрики, значения которых неудовлетворительны, выбираются для анализа
- ▶ Программный код модифицируется с целью улучшения метрик
- ▶ Накапливается статистика
- ▶ \* Возможно корректируются эталонные значения метрик

# Отображение метрик

The screenshot displays the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Model, Diagram, Run, Window, and Help. The toolbar contains various icons for file operations and development tools. The Model Navigator on the left shows a project structure with folders like Demo, DocMgmtBizProcess, DocMgmtPIM, and DocMgmtPSM, and classes like INomination, IPerson, and Nomination. The Properties view at the bottom left shows a 'Kiviat chart - DocMgmtPSM'. The main editor window displays the code for the testAddNomination method in Document.java. The Audit Metric table at the bottom right shows the following data:

NOO	NOOM	NOP	NOPA	NORM	PC	PF	PIS	PS	PUR	RFC	TCC	WCM	WMPC1	WOC
52	40			32		0				146			83	



# **Обеспечение качества программных систем**

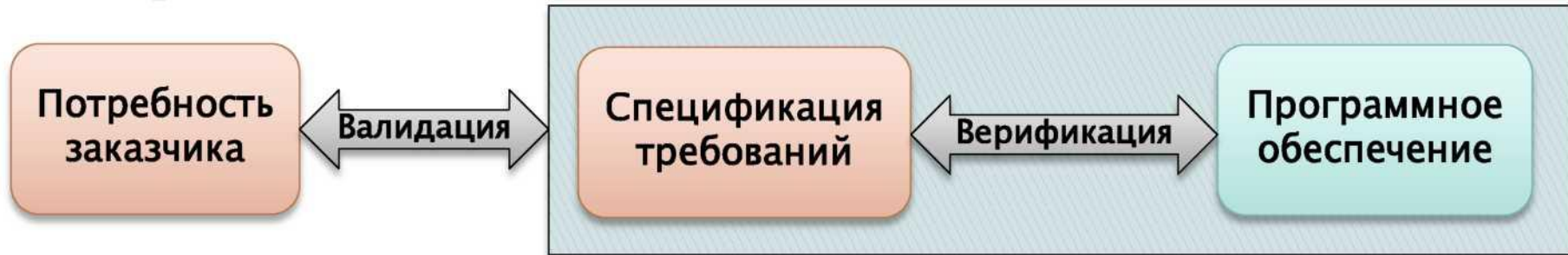


# Обеспечение качества ПО

- ▶ 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

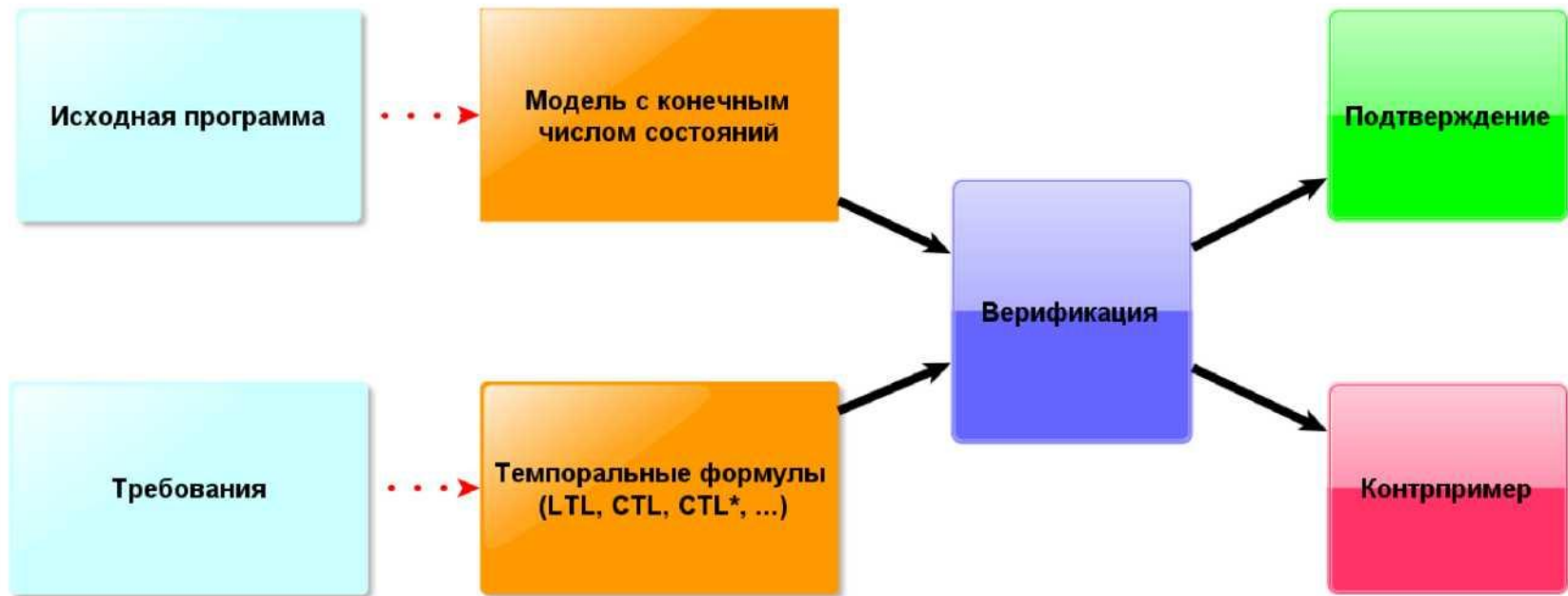
- ▶ Проверка модели, проверка на модели, model checking
- ▶ Метод формальной верификации для систем с конечным числом состояний
- ▶ Позволяет проверить, удовлетворяет или нет система некоторому свойству (требованию)

# 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.

# 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)

# Методы обеспечения качества ПО

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