

# **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. Высотомер выдавал значение  $\leq 0$ .
- ▶ **Голландия, 2000 год.** Остановка доменной печи 29 февраля. Гибель 6 человек. 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  $j$

# Метрики сложности

- ▶ Метрики размера программных систем
- ▶ Классические метрики сложности
  - Метрики Холстеда
- ▶ Метрики сложности потока управления
- ▶ Метрики сложности потока данных

# Метрики размера

- ▶ LOC – Lines Of Code. Число строк кода. Может вычисляться
  - как общее число всех строк
  - как число операторов программы
- ▶ Среднее число строк в модуле
- ▶ Среднее число строк в функции
- ▶ Степень комментирования программы (ЛОСС/ЛОС)
- ▶ Плотность комментирования

# Метрики Холстеда

- ▶ Характеристики программы
  - NUOprtr (Number of Unique Operators) — число уникальных операторов программы
  - NUOprnd (Number of Unique Operands) — число уникальных операндов программы (словарь операндов)
  - Noprtr (Number of Operators) — общее число операторов в программе
  - Noprnd (Number of Operands) — общее число операндов в программе
- ▶ Оценки
  - **Словарь программы** (Halstead Program Vocabulary, HPVoc):
    - $HPVoc = NUOprtr + NUOprnd$ ;
  - **Длина программы** (Halstead Program Length, HPLen):
    - $HPLen = Noprtr + Noprnd$ ;
  - **Объем программы** (Halstead Program Volume, HPVol):
    - $HPVol = HPLen * \log_2 HPVoc$ ;
  - **Сложность программы** (Halstead Difficulty, HDiff):
    - $HDiff = (NUOprtr/2) \times (Noprnd / NUOprnd)$ ;
  - **Усилия программиста при разработке:** (Halstead Effort):
    - $HEff = HDiff \times HPVol$ .

# Метрики сложности потока управления. Метрика Мак-Кейба

- ▶ Цикломатическая сложность
  - $C = A - V + 2$ 
    - A – число ребер в ГПУ
    - V- число вершин в ГПУ
  - В общем случае показывает число тестовых прогонов программы, для прохода всех ветвей

# Метрики сложности потока данных. Метрики Чепина

- ▶ **Множество «Р»** – входные переменные программы
- ▶ **Множество «М»** – модифицируемые или создаваемые внутри программы переменные
- ▶ **Множество «С»** – переменные, участвующие в управлении работой программного модуля (управляющие переменные)
- ▶ **Множество «Т»** – не используемые в программе переменные
- ▶  **$Q = P + 2M + 3C + 0.5T$**

# 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 main editor shows a Java class named `Document.java` with the following code:

```
public void testAddNomination(INomination aNomination) throws BusinessException
{
    if (this.isPublished())
    {
        throw new BusinessException("Document already published.");
    }

    INomination lastNomination = null;
    try
    {
        lastNomination = this.getLatestNomination();
    }
    catch (BusinessRuleException ex)
    {
        // lastNomination is null. ok to nominate
        return;
    }

    if (lastNomination.isStatusPending() || lastNomination.isStatusInReview)
    {
        throw new BusinessException("Nomination denied. Document has u");
    }
}
```

The Model Navigator on the left shows a project structure with a `casestudy` package containing `Document`, `IDocument`, `INomination`, `IPerson`, `IPersonProfile`, `ITeam`, `ITeamMember`, `ITeamMemberProfile`, and `Nomination`.

The Properties view at the bottom left displays a Kiviati chart for `DocMgmtPSM`. The chart is a circular radar plot with 16 axes, each representing a metric. The values for these metrics are shown in the table below.

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

# Отображение метрик

The screenshot displays the Borland Developer Studio 2006 interface. The central window shows a Kiviati Chart (Radar Chart) with 18 axes representing different metrics. A red circle highlights a specific data series. The bottom panel shows a Metrics table for the Win32OperatorOverload project.

	NO...	NO...	NOIS	NO...	NOM	NOO	NO...	NOP	NO...	NO...	PC	PF	PIS	PS	PUR	RFC	TCC	WCM	W...	WOC
<default>	0	-1	0	-1	103	98	0	-1	-1	42	300		0	2	0	303	-1	-1	146	-1
Vassbotn.Vcl.Complex	3	0	-1	56	47	0	-1	7	42							92	0	14	48	90
Vassbotn.Vcl.Comp	3	0	-1	103	98	0	-1	5	22							303	0	39	146	97