

Рабочая программа составлена на основании федерального государственного образовательного стандарта высшего образования по направлению (специальности): 09.04.04 Программная инженерия

ФГОС введен в действие приказом №1406 от 30.10.2014 г. , дата утверждения: 28.11.2014 г.

Место дисциплины в структуре учебного плана: Блок 1, вариативная, по выбору студента

Рабочая программа разработана на основе компетентностной модели выпускника по направлению (специальности): 09.04.04 Программная инженерия

Рабочая программа обсуждена на заседании кафедры

ВТ, протокол заседания кафедры №6 от 20.06.2017

Утверждена на совете факультета автоматики и вычислительной техники, протокол № 6 от 21.06.2017

Программу разработал:

доцент, к.т.н. Романов Е. Л.

Заведующий кафедрой:

доцент, к.т.н. Якименко А. А.

Ответственный за образовательную программу:

доцент Якименко А. А.

1. Внешние требования

Таблица 1.1

Компетенция ФГОС: ПК.6 пониманием существующих подходов к верификации моделей программного обеспечения; в части следующих результатов обучения:
31. методы верификации моделей проектирования и программного обеспечения
у1. применять методы верификации моделей проектирования и программного обеспечения
Компетенция НГТУ: ПК.22.В способность управлять средой функционирования объектов профессиональной деятельности; в части следующих результатов обучения:
34. типовые метрики программного обеспечения
36. основные методы измерения и оценки характеристик программного обеспечения
у3. оценивать работоспособность программного продукта
у4. применять методы и средства проверки работоспособности программного обеспечения
Компетенция НГТУ: ПК.23.В способность к управлению процессами жизненного цикла программного обеспечения; в части следующих результатов обучения:
35. методологии разработки программного обеспечения

2. Требования НГТУ к результатам освоения дисциплины

Таблица 2.1

Результаты изучения дисциплины по уровням освоения (иметь представление, знать, уметь, владеть)	Формы организации занятий
<i>Управление качеством разработки программного обеспечения</i>	
ПК.6.31 методы верификации моделей проектирования и программного обеспечения	
1.методы валидации программного обеспечения	Практические занятия; Самостоятельная работа
ПК.6.у1 применять методы верификации моделей проектирования и программного обеспечения	
2.применять методы валидации программного обеспечения	Практические занятия; Самостоятельная работа
ПК.22.В.34 типовые метрики программного обеспечения	
3.типовые метрики программного обеспечения	Практические занятия; Самостоятельная работа
ПК.22.В.36 основные методы измерения и оценки характеристик программного обеспечения	
4.основные методы измерения и оценки характеристик программного обеспечения	Практические занятия
ПК.22.В.у3 оценивать работоспособность программного продукта	
5.оценивать работоспособность программного продукта	Практические занятия
ПК.22.В.у4 применять методы и средства проверки работоспособности программного обеспечения	
6.применять методы и средства проверки работоспособности программного обеспечения	Практические занятия
ПК.23.В.35 методологии разработки программного обеспечения	
7.методологии разработки программного обеспечения	Практические занятия

3. Содержание и структура учебной дисциплины

Таблица 3.1

Темы практических занятий	Активные формы, час.	Часы	Ссылки на результаты обучения	Учебная деятельность
Семестр: 2				
Дидактическая единица: Управление качеством ПО				

1. Основные понятия и определения	0	2	7	Основные понятия и определения. Задача количественной оценки качества программного обеспечения. Основы метрологической оценки ПО. Задачи метрологии качества ПО. Стандарты управления качеством ПО. Сущность стандартизации, роль и место стандартизации в производстве и применении программного обеспечения, нормативные документы по стандартизации и виды стандартов
2. Характеристики качества ПО	2	4	3, 7	Характеристики качества ПО. Система качества стандарта ISO 9126: характеристики качества, показатели характеристик. Система качества ГОСТ 28195-89: факторы и критерии качества программного обеспечения, метрики и оценочные элементы. Понятие метрики. Классификация метрических шкал: относительные, интервальные, порядковые, категорийные шкалы. Метрики размера программ. Метрики стилистики и понятности программы, метрики Холстеда. Метрики сложности потока управления программы: цикломатическая метрика МакКейба, метрика Майерса, метрика Джилба, метрика граничных значений. Метрики сложности потока данных программы: метрика обращения к глобальным переменным, метрика Спена, метрика Чепина. Метрики инкапсуляции, наследования, полиморфизма
3. Виды сложности при разработке и эксплуатации ПО	2	4	4, 5	Виды сложности при разработке и эксплуатации ПО. Временная, программная, информационная сложности. Измерение и оценка сложности ПО

4. Виды корректности ПО	2	4	1, 2	Виды корректности ПО. Функциональная, детерминированная, стохастическая, динамическая корректности. Тестирование структуры ПО. Типы эталонов, методы измерений и проверки корректности ПО. Классификация ошибок ПО. Причины ошибок. Обнаружение и устранение ошибок. Спецификации программ, анализ корректности. Автоматизация верификации программ.
5. Основные понятия надежности ПО	2	2	5, 6	Основные понятия надежности ПО, методы измерения. Методы обеспечения надежности. Показатели надежности. Определение показателей надежности. Аналитические, имитационные, экспериментальные методы определения показателей надежности. Моделирование и обеспечение надежности при создании ПО. Тестирование программ. Инструментальные средства измерений и оценки качества программного обеспечения
6. Сертификация и система сертификации.	0	2	7	Сертификация и система сертификации. Добровольная и обязательная сертификация. Виды сертификационных испытаний ПО. Стандарты сертификации ПО. Формы подтверждения соответствия. Аккредитация органов сертификации и испытательных лабораторий.

4. Самостоятельная работа обучающегося

№	Виды самостоятельной работы	Ссылки на результаты обучения	Часы на выполнение	Часы на консультации
Семестр: 2				
1	РГЗ	2	40	4

Индивидуальное задание по материалам программных проектов, выполненных студентом (например, ВКРБ): комплексная оценка проекта на основе метрик программного обеспечения: - поиск и установка компонент получения метрик в IDE - сбор статистики метрик программного проекта - комплексная оценка качества проекта на основе анализа статистики - "ручное" вычисление метрики на отдельные части проекта при отсутствии средств автоматизации - определение узких мест проекта с точки зрения метрики ПО - комплексная оценка качества проекта и разработка рекомендаций по рефакторингу : Метрология и качество программного обеспечения : лабораторный практикум / Новосиб. гос. техн. ун-т ; [сост. М. Г. Зайцев]. - Новосибирск, 2008. - 19, [1] с. : табл.. - Режим доступа: http://www.library.nstu.ru/fulltext/metodics/2008/3500.rar				
2	Подготовка к занятиям	1, 3	20	0
: Метрология и качество программного обеспечения : лабораторный практикум / Новосиб. гос. техн. ун-т ; [сост. М. Г. Зайцев]. - Новосибирск, 2008. - 19, [1] с. : табл.. - Режим доступа: http://www.library.nstu.ru/fulltext/metodics/2008/3500.rar				
3	Подготовка к аттестации	1, 3	18	6
: Метрология и качество программного обеспечения : лабораторный практикум / Новосиб. гос. техн. ун-т ; [сост. М. Г. Зайцев]. - Новосибирск, 2008. - 19, [1] с. : табл.. - Режим доступа: http://www.library.nstu.ru/fulltext/metodics/2008/3500.rar				

5. Технология обучения

Для организации и контроля самостоятельной работы обучающихся, а также проведения консультаций применяются информационно-коммуникационные технологии (табл. 5.1).

Таблица 5.1

Деятельность	Информационно-коммуникационные технологии
Информирование	Социальные сети: YandexDisk vk.com/magister_cs; Чат: Skype
Размещение учебных материалов	Социальные сети: YandexDisk vk.com/magister_cs

Таблица 5.2

Активные и интерактивные формы проведения занятий

№	Наименование активных форм
1	Дискуссия
Краткое описание применения: Практическое занятие по предварительно объявленной теме с обсуждением найденных и подобранных студентами материалов, с определением назначения, структуры, особенностей и примеров и практик применения методов и средств управления качеством ПО	

6. Правила аттестации обучающихся по учебной дисциплине

Для аттестации обучающихся по дисциплине используется балльно-рейтинговая система (БРС), позволяющая выставять оценки по традиционной шкале и 15-уровневой ECTS.

Краткая информация о БРС приведена в табл. 6.1.

Таблица 6.1

Оцениваемые виды деятельности обучающихся	Мин. балл	Максимальный балл
---	-----------	-------------------

Семестр: 2		
<i>Практические занятия:</i>	20	40
<i>РГЗ:</i>	20	40
<i>Зачет:</i>	0	20
Контролирующие материалы - список вопросов		

В таблице 6.2 представлено соответствие форм контроля заявляемым требованиям к результатам освоения дисциплины.

Таблица 6.2

Коды компетенций ФГОС	Результаты обучения	Формы контроля	
		Защита РГЗ	Зачет
ПК.6	з1. методы верификации моделей проектирования и программного обеспечения		+
	у1. применять методы верификации моделей проектирования и программного обеспечения		+
	ПК.22.В з4. типовые метрики программного обеспечения		+
	ПК.22.В з6. основные методы измерения и оценки характеристик программного обеспечения	+	+
	ПК.22.В у3. оценивать работоспособность программного продукта	+	+
	ПК.22.В у4. применять методы и средства проверки работоспособности программного обеспечения	+	+
	ПК.23.В з5. методологии разработки программного обеспечения		+

Фонд оценочных средств по дисциплине представлен в приложении № 1 к рабочей программе.

7. Литература

Основная литература

1. Черкесов Г. Н. Надежность аппаратно-программных комплексов : учебное пособие по дисциплине "Надежность, эргономика и качество" [для вузов по направлениям 654600 "Информатика и вычислительная техника" и др.] / Г. Н. Черкесов. - СПб., 2005. - 478 с. : ил.. - Издательская программа 300 лучших учебников для высшей школы в честь 300-летия Санкт-Петербурга.
2. Зайцев М. . Метрология и качество программного обеспечения [Электронный ресурс] : конспект лекций / М. Г. Зайцев ; Новосиб. гос. техн. ун-т. - Новосибирск, [2009]. - Режим доступа: <http://moodle.ciu.nstu.ru/>. - Загл. с экрана.

Дополнительная литература

1. Липаев В. В. Методы обеспечения качества крупномасштабных программных средств / В. В. Липаев ; Ин-т систем. программирования. - М., 2003. - 510 с. : ил.

Интернет-ресурсы

1. ЭБС НГТУ : <http://elibrary.nstu.ru/>
2. ЭБС «Издательство Лань» : <https://e.lanbook.com/>
3. ЭБС IPRbooks : <http://www.iprbookshop.ru/>
4. ЭБС "Znaniy.com" : <http://znaniy.com/>
5. :

8. Методическое и программное обеспечение

8.1 Методическое обеспечение

1. Метрология и качество программного обеспечения : лабораторный практикум / Новосиб. гос. техн. ун-т ; [сост. М. Г. Зайцев]. - Новосибирск, 2008. - 19, [1] с. : табл.. - Режим доступа: <http://www.library.nstu.ru/fulltext/metodics/2008/3500.rar>

8.2 Специализированное программное обеспечение

1 Microsoft Office

2 Microsoft Windows

9. Материально-техническое обеспечение

Презентационное оборудование

№	Наименование	Назначение
1	Презентационное оборудование (мультимедиа-проектор, экран, компьютер для управления)	Проведение крактических занятий

Компьютерный класс

№	Наименование	Назначение
1	Компьютерный класс (Компьютеры объединены в локальную сеть с выходом в Internet)	Компьютеры объединены в локальную сеть с выходом в Internet

Характеристики и оценка качества кода

Цели получения метрических характеристик качества кода:

- поддержание стандартов кода (коллективное владение кодом, XP)
- обнаружение потенциально опасного кода
- обнаружение «узких мест» в структуре кода
- мониторинг разрабатываемого проекта (в системе контроля качества организации при наличии статистики)

Текущее состояние:

- прагматический подход: средства поддержания стандартов кодирования (стилистика + простые количественные метрики)
- научно-исследовательский подход: разработка метрик, отражающих свойства «правильного» кода
- сами по себе метрики не гарантируют эффективной реализации функционала
- собранная метрическая статистика нуждается в обработке и интерпретации



Характеристики и оценка качества кода

Качество кода – наличие у кода набора **свойств**, которые могут выступать показателями качества.

Свойства стиля:

- документированность
- читаемость
- устойчивость к потенциальным ошибкам

Свойства структуры кода:

- модульность
- сложность
- инкапсуляция (сокрытие свойств)
- управляемость (автономность, независимость, связность)

Метрика – количественная оценка (мера), создаваемая путем введения параметра, который может характеризовать одно из этих свойств

Виды метрик:

- количественные (размерности кода)
- сложности потока управления
- сложности потока данных
- сложности ПУ + ПД
- объектно-ориентированные



Рекомендации к стилю кода

<http://habrahabr.ru/post/112042/>

1. Комментарии/Javadoc: пишите их; используйте стандартный стиль.
2. Короткие методы: не пишите гигантских методов.
3. Поля: должны быть вверху файла, или прямо перед методом, который их использует.
4. Локальные переменные: ограничивайте область видимости.
5. Импорты: `android`; сторонние (в алфавитном порядке); `java(x)`
6. Отступы: 4 пробела, без табуляций.
7. Длина строки: 100 символов.
8. Имена полей: не `public` и не `static` поля начинаются с «m».
9. Фигурные скобки: открывающие фигурные скобки не находятся в отдельной строке.
10. Аннотации: используйте стандартные аннотации.
11. Сокращения: используйте сокращения как слова в именах, например, `XmlHttpRequest`, `getUrl()` и т.п.
12. Стиль TODO: «TODO: пишите описание здесь».
13. Согласованность: смотрите, что находится вокруг вас



Количественные метрики

Количественные метрики – количество строк исходного кода, элементов языка (операций, переменных), комментариев и т.п.

1. SLOC - количество строк исходного кода, операторов, комментариев

Удельная мера: на 1 модуль (файл), на 1 функцию, % комментариев

2. Мера Холстеда – применение мер теории информации для оценки количественного многообразия кода программы.

- **n1** – словарь действий - операторов (ключевые слова, знаки операций, операторы, символы-разделители)
- **n2** – словарь сущностей – операндов (имена типов данных, переменные, константы)
- **N1** – количество операторов
- **N2** – количество операндов
- **n1', n2'** – теоретический словарь программы
- **n1 + n2** – словарь программы
- **N1 + N2** – длина программы
- **V = N log₂(n)** – объем программы
- **N' = n1 log₂(n1) + n2 log₂(n2)** – теоретическая длина программы



Мера Холстеда

```
//-----Двоичный поиск в упорядоченном массиве
int binary(int c[], int n, int val){ // Возвращает индекс найденного
int a,b,m;                          // Левая, правая границы и
    for(a=0,b=n-1; a <= b;) {        // середина
        m = (a + b)/2;               // Середина интервала
        if (c[m] == val)             // Значение найдено -
            return m;                // вернуть индекс найденного
        if (c[m] > val)
            b = m-1;                  // Выбрать левую половину
        else
            a = m+1;                  // Выбрать правую половину
    }
return -1; }                          // Значение не найдено
```

n1 = 16 (binary,(),,,{,},=,<=,==,+,-,/,[],while,if,else,return, ;)

n2 = 10 (int,c,n,m,val,a,b,1,2,0)

N1 = 44 - количество операторов

N2 = 35 - количество операндов

n = n1 + n2 = 26 - словарь программы

N = 79 - длина программы

V = 373 - объем программы

N' = 97 - теоретическая длина программы (словарь)



Сложность потока управления

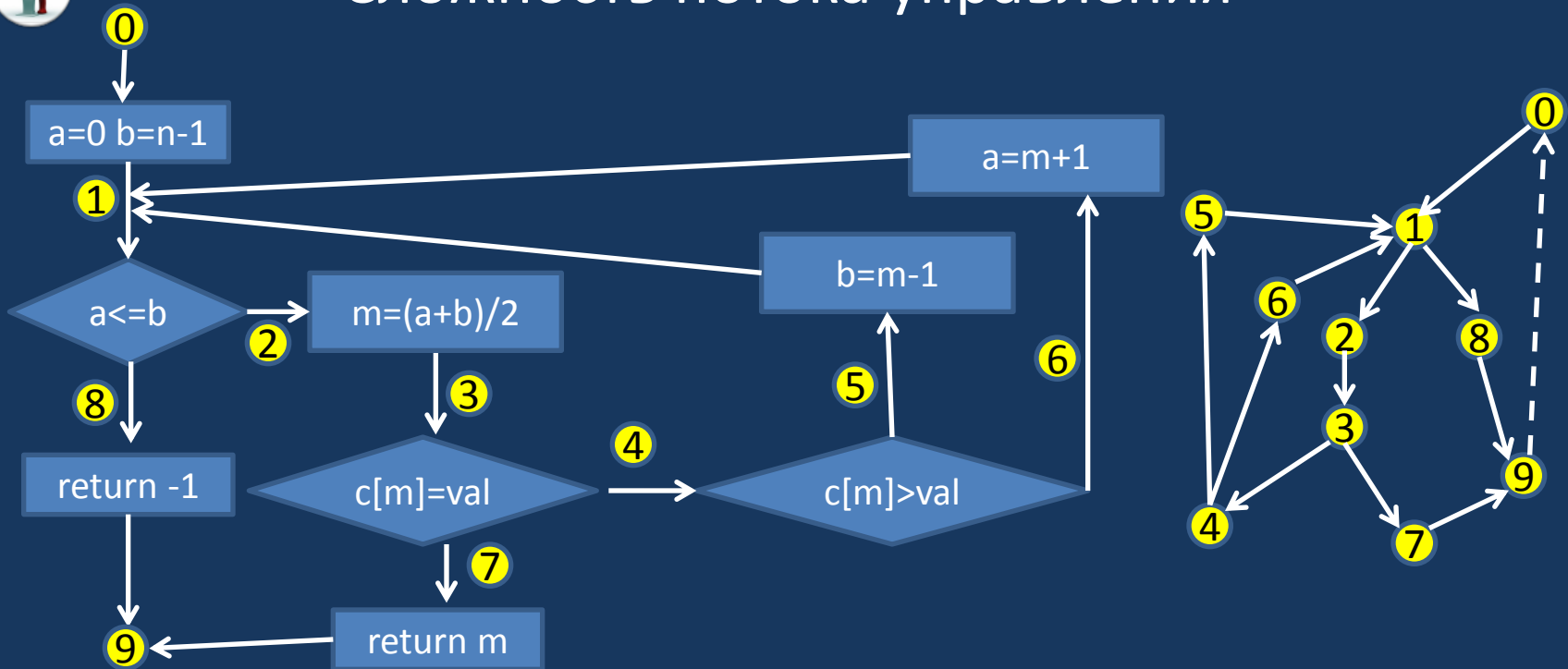
Цикломатическая сложность графа потока управления: $V = E - N + 2$

E – количество дуг, N – количество вершин, 2 – замыкание конечной вершины на начальную. Граф потока управления строится по блок-схеме, вершины графа - дуги блок-схемы («точки останова»), дуги соединяют входы и выходы элементов блок-схемы (1-1 для линейного блока и 1-2 – для ветвления)

```
//-----Двоичный поиск в упорядоченном массиве
int binary(int c[], int n, int val){ // Возвращает индекс найденного
int a,b,m;                          // Левая, правая границы и
    for(a=0,b=n-1; a <= b;) {        // середина
        m = (a + b)/2;               // Середина интервала
        if (c[m] == val)              // Значение найдено -
            return m;                // вернуть индекс найденного
        if (c[m] > val)
            b = m-1;                  // Выбрать левую половину
        else
            a = m+1;                  // Выбрать правую половину
    }
return -1; }                         // Значение не найдено
```



Сложность потока управления

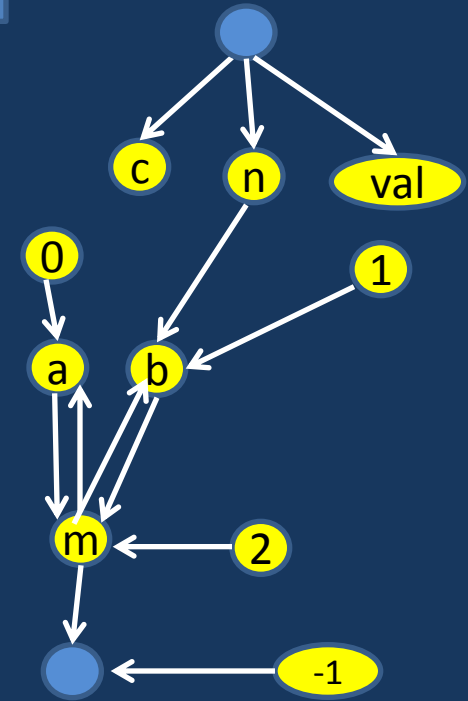
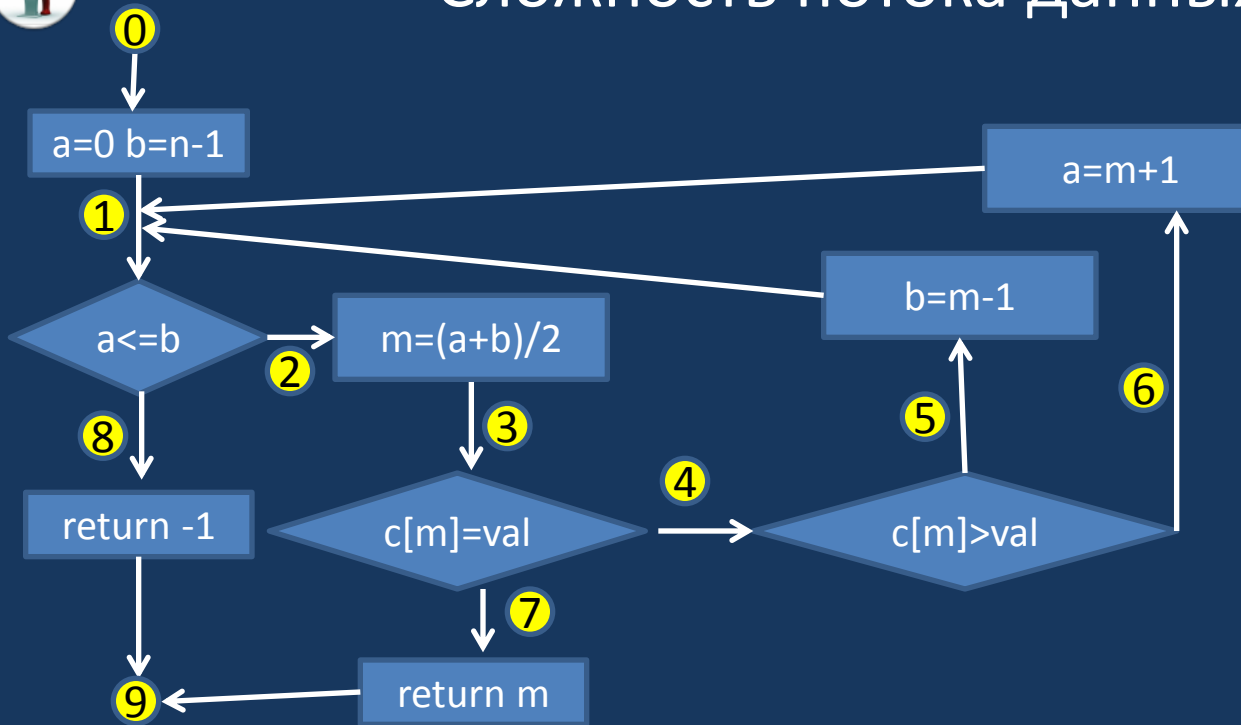


$$V = 12 - 9 + 2 = 5$$

- линейные участки не влияют
- для «пустой» программы $V = 1$



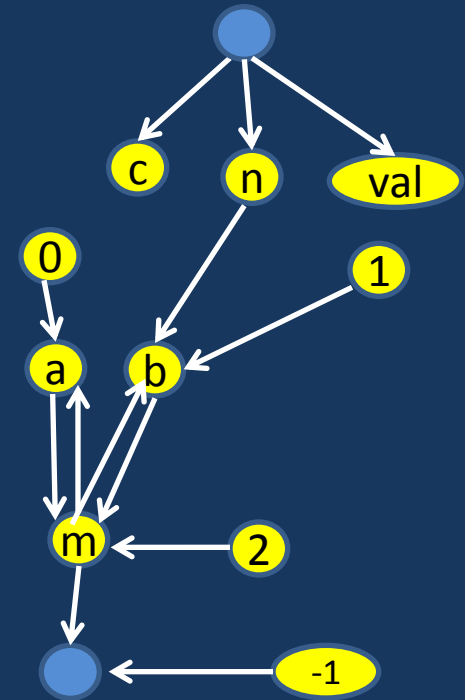
Сложность потока данных



Граф потока данных соединяет вершины данных, связанные **непосредственными** вычислениями, учитываются также входные/выходные данные и инициализация константами

$$V = E - N + 2 = 13 - 12 + 2 = 3$$

Для линейных вычислений $V = 1$



- граф потока команд
- дуги графа потока данных отдельно для каждой вершины потока команд (вершина **перед** элементом блок-схемы)
- начальные (конечные) вершины потоков команд и данных совпадают (точки входа и выхода)

$$V = V_c + E_d - N_d + 2 = 5 + 28 - 12 + 2 = 23$$



Характеристики связей модулей

Характеристика внутренних связей между компонентами модуля – **связность (cohesion)**, внешних связей – **сцепление (coupling)**.

Качественная характеристика (вид связи)

Виды связности:

- **совпадение (cc=0)** – ничего общего, кроме факта нахождения в общем модуле
- **логическая (cc=1)** – в рамках общего функционала, но не связаны между собой (например, обработка ошибок различных типов)
- **временная (cc=3)** - используются на одной фазе процесса, в один период времени (инициализация, завершение)
- **процедурная (cc=5)** – в рамках одного сценария, имеют определенный порядок вызова
- **коммуникативная (cc=7)** – работают с общей структурой данных
- **последовательная (cc=9)** – результат первого – вход второго
- **функциональная (cc=10)** – один модуль вызывает другой
- **объектная** - в рамках класса с общими свойствами и функциональностью



Характеристики связей модулей

Виды сцепления:

- **сцепление по данным (СЦ=1)** – вызов с параметрами – примитивными типами данных
- **сцепление по образцу (СЦ=3)** – вызов с параметрами-объектами (структурами данных)
- **сцепление по управлению (СЦ=4)** - модуль устанавливает флаги в другом модуле, управляя его поведением
- **сцепление по внешним ссылкам (СЦ=5)** - модули используют один и тот же глобальный элемент данных или ссылку на него
- **сцепление по общей области (СЦ=7)** - модули разделяют одну и ту же глобальную структуру данных или используют ссылку на общий объект
- **сцепление по содержанию (СЦ=9)** - один модуль прямо исполняет часть кода другого модуля



Объектно-ориентированные метрики

Метрика связности (cohesion) класса по данным

Токен (Token) – элемент данных класса (переменные, ссылки и константы)

Секция данных – набор токенов для вычисления одного из выходных параметров метода (результат метода или элемент данных класса)

Склеенный токен – используется более чем в 1 секции данных

Сильно склеенный токен – используется более чем во всех секциях данных

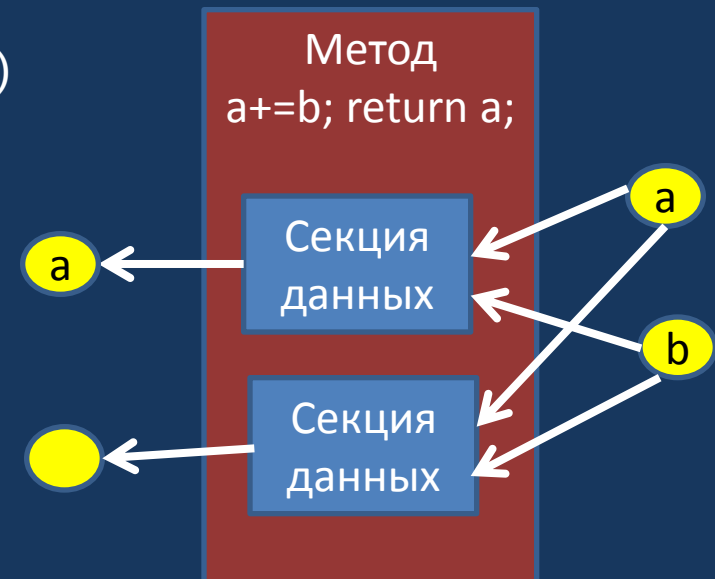
Сильная связанность по данным (SDC – Strong Data Cohesion) – доля сильно связанных токенов

Слабая связанность по данным (WDC – Weak Data Cohesion) – доля слабо связанных токенов

Клейкость данных (DA - Data Adhesiveness)

$$\frac{\text{кол-во дуг токен-секция}}{\text{кол-во секций} * \text{кол-во токенов}}$$

- Конструкторы, set/get-методы не учитываются
- при вызове метода А из метода В секции метода А учитываются в В





Метрика связности класса по данным. Примеры

Минимальная связность – набор независимых элементов данных и операций над ними

Токенов – $3(N)$

Секций (методов) – $6(2N)$

$SDC = 0$

$WDC = 3/3 = 1$

$DA = 2N / (4N * N) = 1/2N$

```
class MinCohesion{
    private int a,b,c;
    public MinCohesion(int aa, int bb, int cc){
        a=aa; b=bb; c=cc;      // Не считается
    }

    //-----
    public int incA(){ int aa=a; a++; return aa; }
    public int incA2(){ a++; return a; }
    public int incB(){ int aa=b; b++; return aa; }
    public int incB2(){ b++; return b; }
    public int incC(){ int aa=c; c++; return aa; }
    public int incC2(){ c++; return c; }
}
```

Максимальная связность – набор методов над единственным токеном

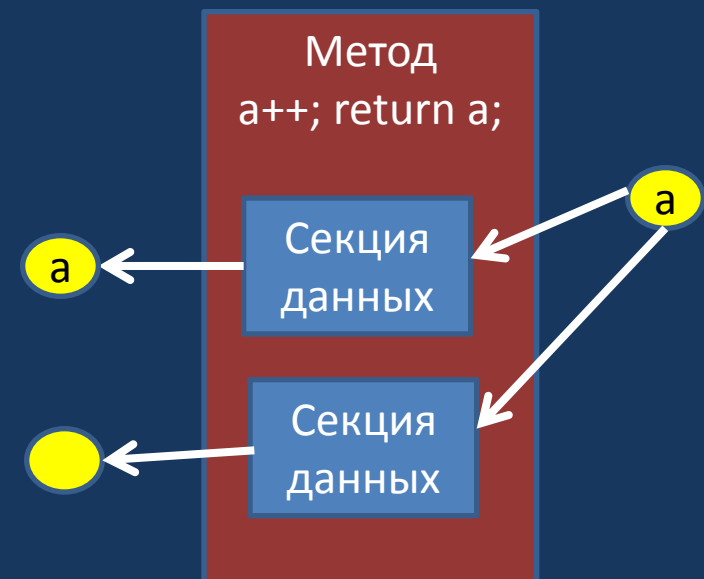
Токенов – 1

Секций (методов) – N

$SDC = 1$

$WDC = 1$

$DA = N / (N * 1) = 1$





Метрика связности по методам

прямо связанные методы – имеют общий элемент данных

косвенно связанные методы – имеют общий прямо связанный метод, в т.ч. вызывают его

$NP = N * (N-1) / 2$ – количество пар методов

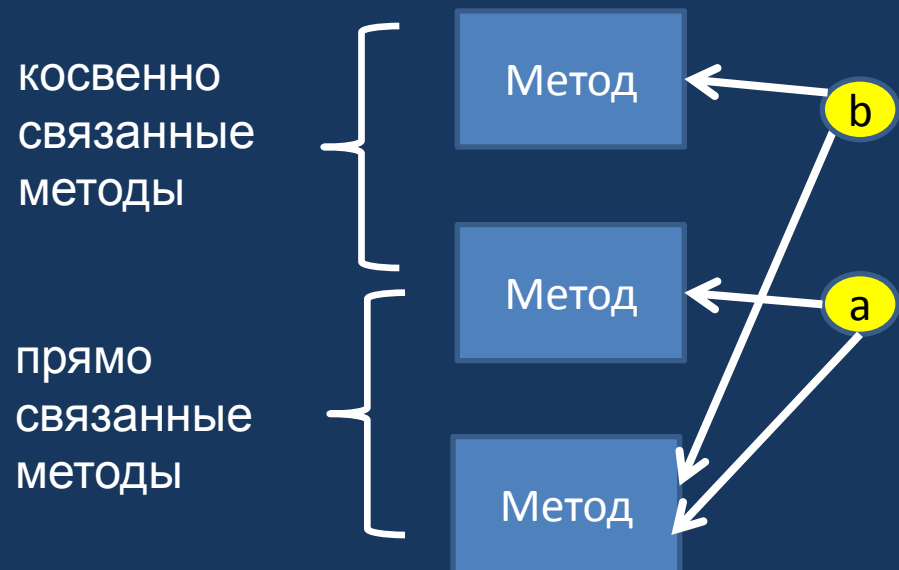
NT – количество прямо связанных пар

NL - количество косвенно связанных пар

$TCC = NT / NP$ сильная связность класса (Tight Class Cohesion)

$LCC = (NT + NL) / NP$ - слабая связность класса (Loose Class Cohesion)

- связность может определяться как с учетом наследования, так и без него (для текущего класса)





Прагматические ОО метрики

Метрики Чидамбера и Кемерера

1. Взвешенные методы на класс - WMC (Weighted Methods Per Class) – метрика количества и сложности методов: суммарная сложность кода методов (нормированная), общее количество методов

Варианты: учет унаследованных или только собственных методов

2. Высота дерева наследования - DIT (Depth of Inheritance Tree) - максимальная длина пути (количество вершин – классов) по дереву наследования

3: Количество детей - NOC (Number of children) – среднее количество прямых наследований класса

4. Сцепление между классами - CBO (Coupling between object classes) – общее количество вызовов методов и использования свойств объектов других классов в коде класса

5. Отклик класса - RFC (Response for a class) – общее кол-во методов в других классах, которые могут быть вызваны **при изменении** данных класса

6: Недостаток связности в методах LCOM (Lack of Cohesion in Methods) - кол-во пар **несвязанных** методов (не использующих совместно хотя бы 1 элемент данных класса) минус кол-во пар **связанных** методов (но >0).



Прагматические ОО метрики

Метрика Мартина

Категория классов – группа функционально связанных (сильно связанных) классов

Центростремительное сцепление C_a – количество классов вне категории, зависящих от классов этой категории

Центробежное сцепление C_e количество классов внутри категории, которые зависят от классов вне этой категории

Нестабильность $I = C_e / (C_a + C_e)$

Абстрактность A – доля абстрактных классов

$I + A = 1$ - сбалансированность между абстрактностью и нестабильностью (главная последовательность)

$D = |(A + I - 1) / \sqrt{2}|$ - расстояние до главной последовательности

$D_n = |A + I - 2|$ - нормализованной расстояние до главной последовательности



Прагматические ОО метрики

Метрики Лоренца и Кидда

- 1. Размер класса CS** (Class Size) – количество методов (в т.ч. унаследованных и закрытых) + количество свойств. $CS \leq 20$ методов
- 2. Количество методов, переопределяемых подклассом, NOO** (Number of Operations Overridden by a Subclass). $NOO \leq 3$
- 3. Количество операций, добавленных подклассом, NOA** (Number of Operations Added by a Subclass) Для $CS = 20$ и $DIT = 6$, $NOA \leq 4$
- 4. Индекс специализации SI** $= NOO * L / M$ (Specialization Index), L – уровень наследования, M – общее количество методов. $SI \leq 0,15$
- 5. Средний размер операции AOS** (Average Operation Size). $AOS \leq 9$, SLOC-метрика кода
- 6. Сложность операции ОС** (Operation Complexity) – цикломатическая сложность метода. Предлагается собственная калькуляция с весами операций
- 7. Среднее количество параметров на операцию ANP** (Average Number of Parameters per operation) $ANP = 0,7$
- 8. Количество описаний сценариев NSS** (Number of Scenario Scripts)
- 9. Количество ключевых классов NKC** (Number of Key Classes) $NKC > 0,2$ от общего количества классов системы
- 10. Количество подсистем NSUB** (Number of SUBsystem) $NSUB > 3$



Запутывающие преобразования

Часто встречающиеся формальные преобразования кода, не меняющие функционал, но изменяющие метрики кода:

- **запутывание форматирования исходного текста** (layout obfuscation) - стилистические преобразования (изменения имен, выравнивания, комментариев), оказывающие влияния только на SLOC-метрики
- **запутывание данных** (data obfuscation)
 1. преобразования размещения (локальные, глобальные, объектные) и кодирования
 2. преобразования агрегации
 3. преобразования упорядочения
- **запутывание управления** (control obfuscation)
 1. преобразование агрегации - вставка (inline) и вынос (outline) функции, клонирование (копипаст) кода, «раскрутка циклов», переупорядочение операторов
 2. преобразование вычислений - вставка «мертвого» кода, диспетчеризация
 3. табличная интерпретация – автоматные модели, переменные состояния
- **превентивные трансформации** (preventive transformation)



Средства контроля качества кода

sonarqube®

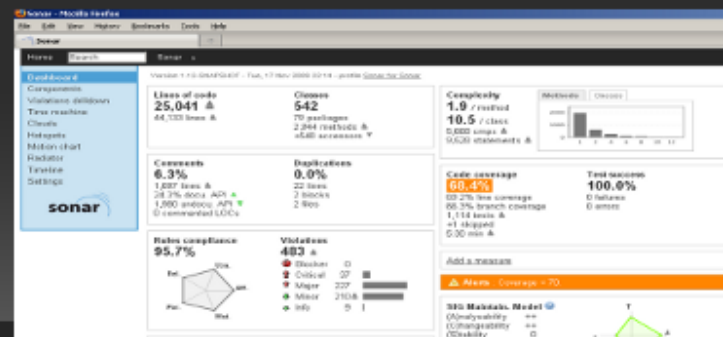
Download Features Get Support Get Involved Development Roadmap Resources Blog Company

Put your technical debt under control

Productivity is falling?

Confess your source code to clean it up!

► Mission ► Platform ► Figures



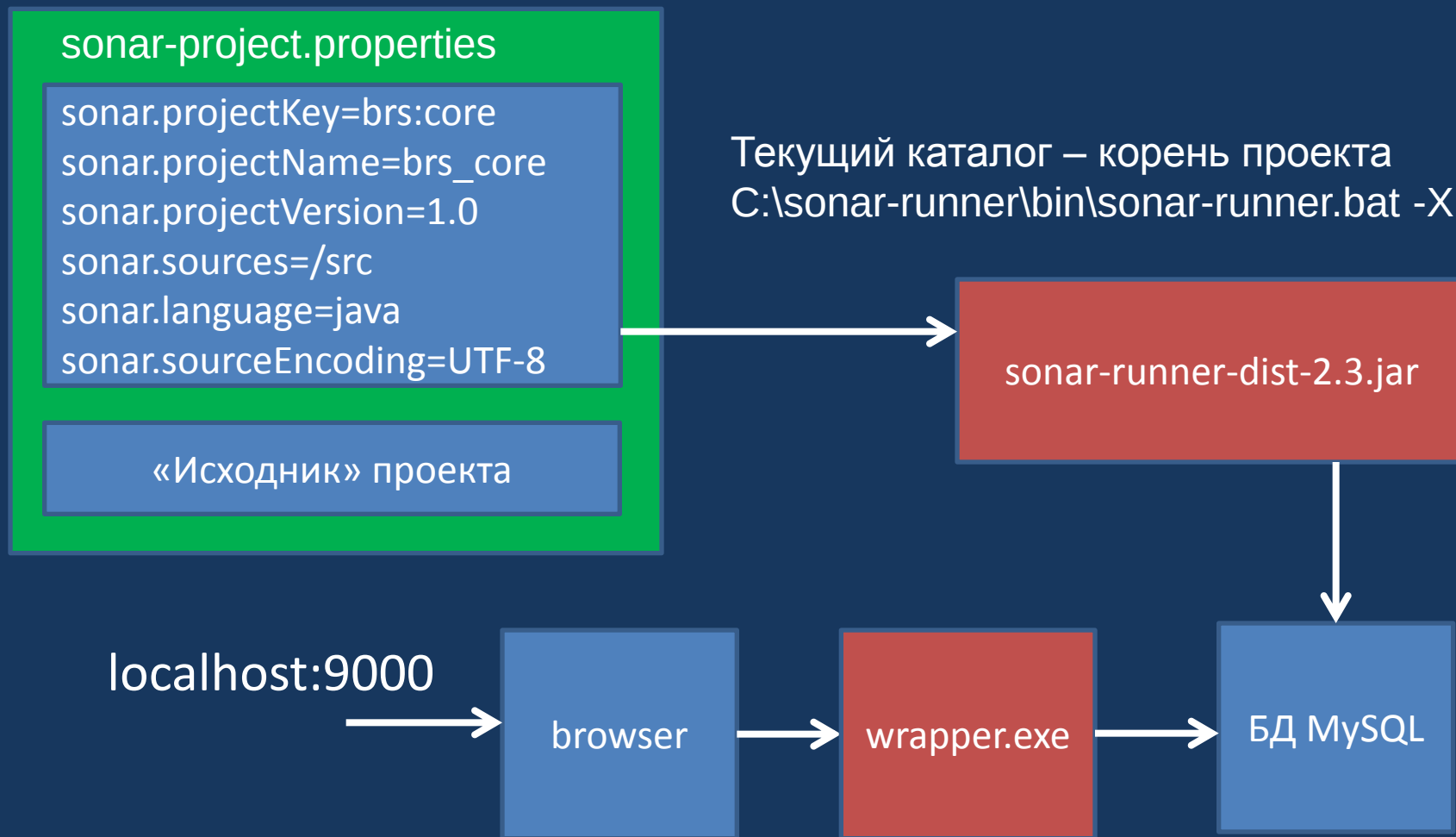
SonarQube (sonarqube.org) – средство контроля качества кода:

- комментарии
- дублирование кода (копипаст)
- правила кодирования
- Unit-тесты
- сложность кода
- потенциальные ошибки
- архитектура и конструирование



SonarQube

Установка и использование



C:\sonarqube\bin\windows-x86-32\StartSonar.bat



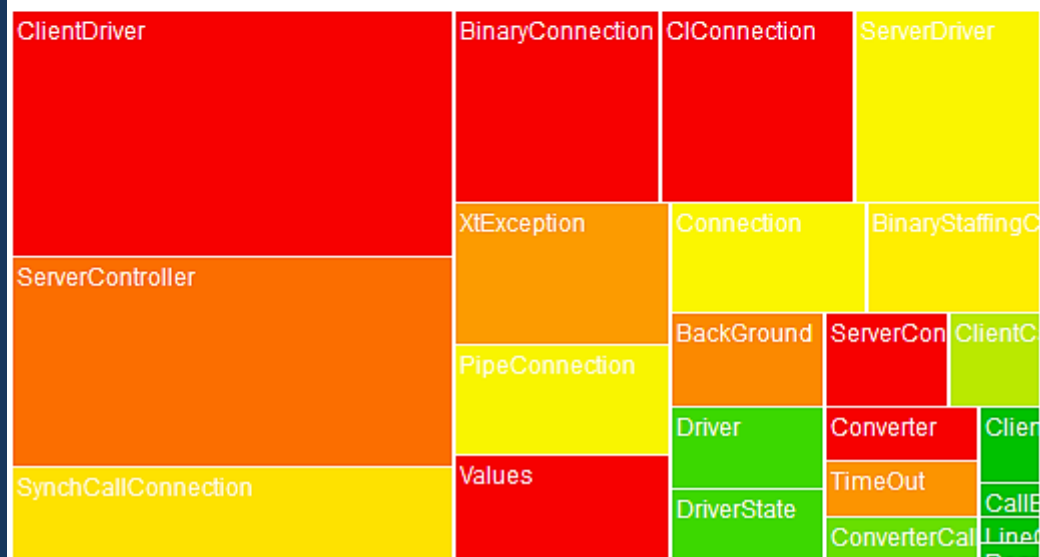
SonarQube

<u>A</u>	<u>Name</u> ^	<u>Version</u>	<u>LOCs</u>	<u>Technical Debt</u>	<u>Last Analysis</u>
	brs_android	1.0	2 825	26.5	02 марта 2014
	brs_core	1.0	6 707	73.6	02 марта 2014
	brs_desktop	1.0	5 340	54.2	02 марта 2014
	brs_desktop	1.0	5 340	54.2	02 марта 2014
	brs_web	1.0	684	4.3	02 марта 2014
	GUIWizard	1.0	2 031	16.6	30 марта 2014
	xtaxi:android	1.0	17 156	157.2	09:51
	xtaxi_core	1.0	9 948	95.8	09:59

8 results

Projects

Size: Lines of code Color: Rules compliance 0.0% 100.0%





SonarQube – результаты анализа кода

Version 1.0 - 03 нояб. 2014 09:59

Lines of code

9 948

11 898 lines
5 720 statements
109 files

Classes

113

9 packages
936 functions
216 accessors

Documentation

8.6% docu. API

919 public API
840 undocu. API

Comments

12.9%

1 470 lines

Duplications

4.4%

519 lines
44 blocks
14 files

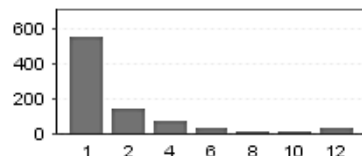
Complexity

2.5 /function

21.0 /class

21.8 /file

Total: 2 378



☒ Functions ☐ Files

Issues

3 581

Technical Debt

95.8 days

	Blocker	58	
	Critical	65	
	Major	2 311	
	Minor	1 044	
	Info	103	

Unit Tests Coverage

-

Unit test success

0 tests



SonarQube - копипаст

Duplications

4.4%

519 lines

44 blocks

14 files

Duplicated lines (%)

4.4%

Drilldown on 519 Duplicated lines

🔍 📁 xtaxi_core	291	🔍 📄 WtConnection	118
🔍 📁 xtaxi_core.waptaxi	118	🔍 📄 ClConnection	76
🔍 📁 xtaxi_core.clientline	58	🔍 📄 BinaryConnection	76
🔍 📁 xtaxi_core.binstream	28	🔍 📄 ServerController	62
🔍 📁 xtaxi_core.configserver	12	🔍 📄 PipeConnection	38
🔍 📁 xtaxi_core			

xtaxi_core

📄 xtaxi.core.ServerController

[Duplications](#) [Issues](#) [Source](#)

14.9%

Lines: 416

Duplicated lines: 62

Duplicated blocks: 4

Blocks	Nb Lines	From line	File	Details
3	11	184	ServerController	📄 xtaxi.core.ServerController
	11	227	<u>ServerController</u>	184 <code>@Override</code>
	11	63	<u>BackGround</u>	185 <code>public void onCancel() {</code> 186 <code>}</code> 187 <code>@Override</code> 188 <code>public void onError(XtException ee) {</code> 189 <code>System.out.println(ee.toString());</code>
				Expand



SonarQube – стилистика кода

The screenshot displays the SonarQube web interface. At the top, a tree view shows the project structure with folders like `xtaxi.core.waptaxi` (77.2), `xtaxi.core.data` (37.0), `xtaxi.core.binstream` (31.0), `xtaxi.core` (19.9), `xtaxi.core.configserver.data` (18.8), and `xtaxi.core.configserver` (13.0). On the right, a file list shows `WtConnection`, `WtConverter` (selected), `DateTime`, `ServerController`, `ClldVector`, and `ClientDriver`.

Below the tree, the breadcrumb path is `xtaxi_core > xtaxi.core.waptaxi.WtConverter`. There are tabs for `Duplications`, `Issues`, and `Source`. The `Issues` tab is active, showing a summary of 178 issues: 2 Blockers, 3 Critical, 91 Major, 81 Minor, 1 Info, and a Technical Debt of 3.1 days.

Filters include `Full source`, `Time changes...`, and a dropdown set to `Critical (3)`. The code view shows a snippet from `WtConverter` with lines 267 to 271. Line 271 is highlighted in red and contains a catch block: `} catch(Throwable ee) { System.out.println(ee); }`. Below the code, a message states: `Exception handlers should provide some context and preserve the original exception`, with an `Open` link and an update time of `около 2 часа`.

- технологические дефекты (обработка исключений)
- стилистика, минимизирующая ошибки (`if () {...обязательно...}`)
- стандарты оформления кода (`final static int MYCONST=55;`)



SonarQube – профили качества (правила)

Issues

3 243

Technical Debt

80.1 days



Blocker 51



Critical 60



Major 2 035



Minor 994



Info 103

☒ Major

"java.lang.Error" should not be extended

java.lang.Error and its subclasses represent abnormal conditions, such as OutOfMemoryError, which should only be used for exceptional situations. The following code snippet:

```
public class MyException extends Error { /* ... */ } // Non-Compliant
```

should be refactored into:

```
public class MyException extends Exception { /* ... */ } // Compliant
```

Repository: squid | Key: S1194 | Available since 01 anp. 2014

☒ Critical

Throwable.printStackTrace(...) should never be called

Throwable.printStackTrace(...) prints a throwable and its stack trace to some stream.

Loggers should be used instead to print throwables, as they have many advantages:

- Users are able to easily retrieve the logs.
- The format of log messages is uniform and allow users to browse the logs easily.

The following code:

```
try {
    /* ... */
} catch (Exception e) {
    e.printStackTrace(); // Non-Compliant
}
```



Средство оценки качества кода в MS Visual Studio

Code Metrics Results						
Filter: None Min: Max: [Icons]						
Hierarchy ▲		Maintainability In...	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
ClassLibrary (Debug)		62	280	3	4	383
ClassLibrary		62	280	3	4	383
A		83	2	1	0	3
A(int)		83	2		0	3
B		98	1	2	1	1
B(int)		98	1		1	1
C		5	277	3	4	379
C(int)		2	201		2	233
Method(int, DateTime, Guid) : int		14	76		3	146

Maintainability Index – комплексный показатель качества кода.

$$MI = \text{MAX}(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * 100 / 171)$$

HV – Halstead Volume, вычислительная сложность

Class Coupling – сцепление классов

CC (Cyclomatic Complexity) – цикломатическая сложность кода

Depth of Inheritance – глубина наследования

LoC (Lines of Code) – количество строк кода (исключая пустые строки, комментарии, строки со скобками, объявление типов и пространств имен)



Online-оценка кода <http://www.codenforcer.com>

- нормализованные ОО метрики - $0 \leq V \leq 1$
- 7 метрик для пакета, 2 – для класса
- рекомендации по исправлению кода
- оценка стилистики кода
- разработка документации к проекту



Projects | My Profile | Messages | Support | Help

Logged in as [romanow \[Anonymous\]](#) 

Projects

C# | JAVA | C++ | PHP | ALL

Start your work with [Creating of New Project](#). Click on project to see details. Click on language name to view projects for this language

[New Project](#)

Projects per page 20

		Project Name	Project type	Project Owner	Number of Files	Number of Lines	Number of Types	Number of Assemblies
		brs_core	Java	romanow [Anonymous]	91	8705	95	1
		GUIWizard	Java	romanow [Anonymous]	61	5796	72	1

Регистрация. Заливка архива с кодом



Метрики кода <http://www.codenforcer.com>

Варианты применения ОО метрик

#	Metrics	Application	Assembly	Namespace	Package	Class	Interface	Structure	Enumeration
1	Coupling			✓+✓	✓	✓	✓	✓✓	✓
2	Afferent Coupling			+ + +	+	✓	✓	✓✓	✓
3	Efferent Coupling			+ + +	+	✓	✓	✓✓	✓
4	Instability			+ + +	+	✓	✓	✓✓	✓
5	Relational Cohesion			+ + +	+				
6	Distance from the Main Sequence			+ + +	+				
7	Abstractness			+ + +	+	+	+	+ +	+
8	Association Between Classes			+ + +	+	+	+	+ +	+
9	Cohesion of LCOM					✓	✓	✓✓	
10	Cohesion of LCOM HS					✓	✓	✓✓	
11	Modularity					+ +	+ +	+	
12	OOP Level For Types	+ +	+						
13	OOP Level For Methods	+ +	+	+ +					
14	OOP Level For Fields	+ +	+	+ +					

+ - available in codEnforcer metrics ✓ - metrics under development
 + ✓ - metric applicable for Java, C++, C# and PHP + ✓ - specific for Java
 + ✓ - specific for C# + ✓ - specific for PHP + ✓ - specific for C++

Сцепление (coupling)

Общий показатель сцепления кода класса

$$C = r_t \left(\sum_{i=0}^n r_i \right)^{-1}$$

r_i - количество ссылок на класс (ссылка является элементом данных, локальной переменной, результатом или параметром метода)

r_t - количество ссылок на класс, для которого рассчитывается метрика;

n - число классов



Метрики кода

см. метрика Мартина

1. Центростремительное сцепление (afferent coupling) – количество классов вне категории, зависящих от классов этой категории

$$f(N) = \begin{cases} 1, & R \geq N \\ \frac{R}{N}, & R < N \end{cases}$$

$$C_a = 0.35 \frac{R}{PC} + 0.65 f(N),$$

R – кол-во связей из внешних пакетов к классам анализируемого пакета

N – кол-во классов в анализируемом пакете

PC – общее кол-во классов

2. Центробежное сцепление (efferent coupling) – количество классов, от которых зависят классы этой категории

$$C_e = \frac{R}{N}$$

R – кол-во классов анализируемого пакета, которые ссылаются на другие классы

N – кол-во классов в анализируемом пакете

3. Нестабильность (I=1 – максимально стабильный пакет, класс)

$$I = \frac{C_e}{C_a + C_e}$$

4. Недостаток связности методов (lack of cohesion) - $LCOM < 1$,
 $LCOM_{HS} < 2$ (Henderson-Sellers) ($LCOM=0$ – абсолютная связность)

$$LCOM = 1 - \frac{1}{M * F} \sum_{i=0}^n MF,$$

$$LCOM_{HS} = \frac{1}{M - 1} \left(M - \frac{1}{F} \sum_{i=0}^n MF \right)$$

M – кол-во методов

F – кол-во свойств класса

MF – кол-во методов,

использующих i -ое свойство класса

n – кол-во свойств класса

(отношение кол-ва пар «метод-свойство» к возможному числу)

5. Относительная связность (relational cohesion)

$$H = \begin{cases} \frac{1}{N}, & R = 0 \\ \frac{R}{N}, & R > 0 \end{cases}$$

R – кол-во автономных классов (не имеют ссылок и на них не ссылаются вне пакета)

N – кол-во классов в анализируемом пакете

6. Абстрактность

$$A = \frac{M}{N}$$

M – кол-во абстрактных классов

N – кол-во классов в анализируемом пакете

7. Расстояние до главной последовательности

$$DMS = \begin{cases} I - A, & A \in [0, 0.35) \quad I \in [0.35, 1) \\ f(|A - I|, 0.35, 0.7), & A, I \in [0, 0.35) \\ \frac{A + I}{2}, & A, I \in [0.7, 1] \text{ или } A, I \in [0, 0.35) \end{cases}$$

I - нестабильность

A - абстрактность

$$DMS_{norm} = (\max - DMS) \frac{\max - \min}{\max} + \min$$

8. Ассоциация между классами - кол-во случаев использования данным классом других классов

$$ABC_X = \sum_{i=0}^N i \mid (F \notin X)$$

i-количество обращений методам, структурам, интерфейсам или перечислениям F, не принадлежащим данному классу X

N – кол-во классов в пакете



Online-оценка кода <http://www.codenforcer.com>

Общая статистика кода

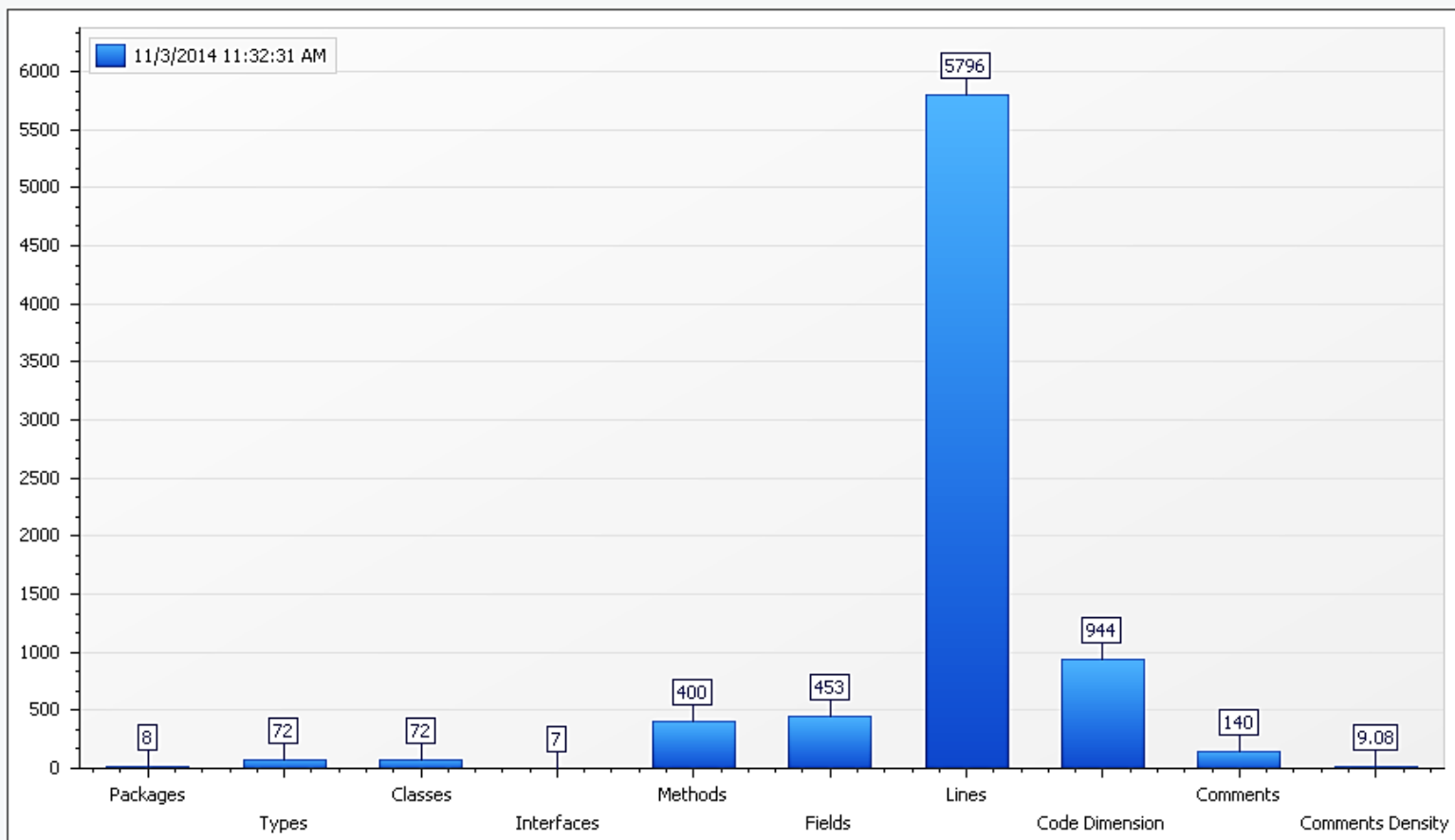
	Status			Date	Number of Files	Number of Lines	Number of Types	Number of
☒		View snapshot		11/3/2014 11:32:31 AM	61	5796	72	1

Source Code Statistics

Statistical information

Code review status

Optimization status





Рекомендации для пакета

Recommendation and Metric

Recommendation

Metric details

Package is rather concrete. It might make sense to add some number of abstract classes and interfaces to increase code flexibility. The level of abstractness of any package reflects the level of application of such principle of object-oriented programming as polymorphism. This indicates at what extend different packages can be resilient to various changes. If value of this metrics is close to 0, this means that there are no abstract classes within the package at all, and therefore it does not describe any entities that can be used in the system to realize different complex objects. Low level of abstractness also indicates that source code is too specific, and changing it might be complicated. Also low abstractness indicates that more likely no design patterns were applied while designing and developing the code.

List of non abstract types:

me.romanow.guiwizard.zview.ZButton
me.romanow.guiwizard.zview.ZViewGroup
me.romanow.guiwizard.zview.ZRelativeLayout
me.romanow.guiwizard.zview.ZEditText
me.romanow.guiwizard.zview.ZImageView
me.romanow.guiwizard.zview.ZLinearLayout
me.romanow.guiwizard.zview.ZView
me.romanow.guiwizard.zview.ZTextView

Close



Online-оценка кода <http://www.codenforcer.com>

Анализ стилистики кода. Редактирование

Source code view

Convention Checking

User's Rules Checking

View

Edit

```
19.
20.
21. public class ZViewGroup extends ZView{
22.     Vector<ZView> childs=new Vector();
23.     public ZViewGroup() {
24.         setHasChilds(true);
25.         setCanCreate(false);
26.     }
27.     public ZViewGroup(ViewGroup orig) {
28.         super(orig);
29.         setHasChilds(true);
30.         setCanCreate(false);
31.     }
32.     @Override
33.     public void addChild(ZView child) {
34.         childs.add(child);
35.     }
36.
37.     @Override
38.     public ZView findChildAt(int index) {
39.         if (pp==getOnScreenChilds())
40.             for (int i=0; i<childs.size(); i++)
41.                 if (childs.get(i).isOnScreen())
42.                     return childs.get(i);
43.         return null;
44.     }
45. }
```

Warnings

- | | |
|----|---|
| 40 | Each line should contain at most one statement. |
| 40 | All binary operators except \. \ should be separated from their operands by spaces. |
| 41 | All binary operators except \. \ should be separated from their operands by spaces. |
| 42 | All binary operators except \. \ should be separated from their operands by spaces. |
| 48 | The element must have a documentation header. |
| 49 | Each line should contain at most one statement. |
| 49 | Each line should contain at most one statement. |
| 49 | All binary operators except \. \ should be separated from their operands by spaces. |
| 50 | All binary operators except \. \ should be separated from their operands by spaces. |



Online-оценка кода <http://www.codenforcer.com>

Документирование проекта

Project Documentation

Save



Export Pdf



Export Chm



Export WebHelp

Auto comments

me.romanow.guiwizard.zview package

Classes:

[me.romanow.guiwizard.zview.ZButton](#)

[me.romanow.guiwizard.zview.ZEditText](#)

[me.romanow.guiwizard.zview.ZForm](#)

[me.romanow.guiwizard.zview.ZFormView](#)

[me.romanow.guiwizard.zview.ZFormView2](#)

[me.romanow.guiwizard.zview.ZFormView3](#)

[me.romanow.guiwizard.zview.ZFormView4](#)

[me.romanow.guiwizard.zview.ZFormView5](#)

User comments



Normal



(Font Name)



(Font Size)



B

I

U

-S-

≡

≡

≡

ab



A



aa-a-a-a-a-a-a-a-a-a



Plugin-ы оценки кода

<http://plugins.netbeans.org/plugin/42970/sourcecodemetrics>

WaitWindowSample.java BRSUITest1.java Client.java Source Code Metrics of brs_core

All Package Class 1 Class 2 Method Collapse all Expand all

Tree	LCC	LCOM1	LCOM2	LCOM3	LCOM4	LCOM5	NAK	LOC	LOCm
brs_core	0.35	114	71	5	4	0.37	0.0	8618	800
me.romanow.brs	0.39	332	170	6	3	0.4	0.0	340	14
DateTime	0.79	665	340	12	6	0.8	0.0	310	4
Values	0.0	0	0	0	0	0.0	0.0	12	1
me.romanow.brs.ciu	0.0	2	2	1	1	0.23	0.0	207	65
CIUConnection	0.0	10	10	5	5	1.15	0.0	106	11
CIUError	0.0	0	0	0	0	0.0	0.0	3	0
CIUKafedra	0.0	0	0	0	0	0.0	0.0	5	0
CIUStudent	0.0	0	0	0	0	0.0	0.0	14	0
CIUTeacher	0.0	0	0	1	1	0.0	0.0	10	0
me.romanow.brs.connect	0.39	76	26	5	4	0.47	0.0	927	44

WaitWindowSample.java BRSUITest1.java Client.java Source Code Metrics of brs_core

All Package Class 1 Class 2 Method Collapse all Expand all

Tree	A	AC	C	D	EC	I	NCP	NIP	LOC	LOCm
brs_core	0.16	2	0.41	0.35	2	0.47	10	0	8618	800
me.romanow.brs	0.0	3	0.5	0.71	0	0.0	2	0	340	14
me.romanow.brs.ciu	0.0	1	0.8	0.14	4	0.8	5	0	207	65
me.romanow.brs.connect	0.0	1	0.0	0.18	3	0.75	5	0	927	44
me.romanow.brs.controller	0.5	0	0.0	0.35	3	1.0	1	1	452	30



java : netbeans : sourcecodemetrics

f_x	0																	
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Class	C	LCC	LCOM1	LCOM2	LCOM3	LCOM4	LCOM5	NAK	NOC	NOF	NOM	NOSF	NOSM	NTM	TCC	WMC	LOC	LOCm
DateTime	0	0,79	665	340	12	6	0,80	0	0	9	45	1	2	1	0,46	81	310	4
Values	1	-	0	0	0	0	-	0	0	7	0	7	0	0	0	0	12	1
CIUConnection	1	-	10	10	5	5	1,15	0	0	12	5	0	2	0	0	11	106	11
CIUError	1	-	0	0	0	0	-	0	0	0	0	0	0	0	0	0	3	0
CIUKafedra	0	-	0	0	0	0	-	0	0	2	0	0	0	0	0	0	5	0
CIUStudent	1	-	0	0	0	0	-	0	0	11	0	0	0	1	0	0	14	0
CIUTeacher	1	-	0	0	1	1	-	0	0	6	1	0	0	1	0	1	10	0
DBConnection	0	0,47	216	26	10	10	0,63	0	0	2	29	0	0	0	0,47	73	346	13
DBLiteConnection	0	-	15	15	6	6	-	0	0	0	6	0	0	0	0	6	150	0
DBSQLiteJDBC	0	0,71	53	28	4	3	0,83	0	0	5	13	0	0	1	0,4	27	155	2
DBServerJDBC	0	0,77	98	60	4	3	0,88	0	0	9	17	0	0	1	0,32	36	216	5
AliveThread	0	-	1	1	2	2	-	0	0	0	2	0	0	0	0	6	26	1
ViewController	0	0,81	213	0	4	4	0,85	0	0	14	31	4	0	2	0,54	90	396	18
ViewControllerLister	0	-	276	276	24	24	-	0	0	0	24	0	0	0	0	24	27	0
Base64Coder	0	-	78	78	13	13	1,00	0	0	3	13	3	12	0	0	31	203	84
DBArchFile	0	-	1	1	2	2	-	0	0	0	2	0	0	0	0	2	7	0
DBArray	0	1,00	0	0	1	1	-	0	0	1	2	0	0	0	1	4	15	0
DBBasePermission	0	-	1	1	2	2	-	0	0	0	2	0	0	0	0	2	9	0
DBBases	0	0,50	3	0	2	2	0,33	0	0	1	4	0	0	0	0,5	4	15	0
DBCell	1	0,58	63	48	4	4	0,99	0	1	12	13	8	0	0	0,19	13	53	3



java : netbeans : sourcecodemetrics

	A	B	C	D	E	F	G
1	Symbol	Level	Name	Minimum	Maximum	Average	Total
2	A	package	Abstractness	-	0,8	0,16	
3	AC	package	Afferent Coupling	-	6	2,00	
4	C	package	Coverage	-	1	0,41	
5	D	package	Distance	0,1	0,707107	0,35	
6	EC	package	Efferent Coupling	-	4	2,00	
7	I	package	Instability	-	1	0,47	
8	LOC	package	Lines Of Code	-	3700		8618
9	LOCm	package	Lines Of Comments	-	317		800
10	NCP	package	Number Of Classes in Package	-	33		10
11	NIP	package	Number Of Interfaces in Package	-	4		0
12	LCC	class	Loose Class Coupling	1,0	0	0,35	
13	LCOM1	class	Lack Of Cohesion in Methods 1	665,0	12	114,00	
14	LCOM2	class	Lack Of Cohesion in Methods 2	340,0	9	71,00	
15	LCOM3	class	Lack Of Cohesion in Methods 3	12,0	4	5,00	
16	LCOM4	class	Lack Of Cohesion in Methods 4	6,0	4	4,00	
17	LCOM5	class	Lack Of Cohesion in Methods 5	0,8	0,95	0,37	
18	LOC	class	Lines Of Code	3,0	731		
19	LOCm	class	Lines Of Comments	-	84		
20	NAK	class	Number of Assertions per KLOC	-	0	-	
21	NOC	class	Number Of Children	-	11		44
22	NOF	class	Number Of Fields	9,0	5		449
23	NOM	class	Number Of Methods	45,0	6		946
24	NOSF	class	Number Of Static Fields	1,0	2		119
25	NOSM	class	Number Of Static Methods	-	0		946
26	NTM	class	Number of Test Methods	-	8		30
27	TCC	class	Tight Class Coupling	-	1	0,26	
28	WMC	class	Weighted Method Count	-	241		1905
29	LOC	method	Lines Of Code	-	-		
30	LOCm	method	Lines Of Comments	-	8		
31	NBD	method	Nested Block Depth	-	5	-	
32	NOP	method	Number Of Parameters	9,0	9		922
33	VG	method	McGabe's Cyclomatic Complexity	1,0	19	1,00	
34							

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»

Кафедра вычислительной техники

“УТВЕРЖДАЮ”
ДЕКАН АВТФ
к.т.н., доцент И.Л. Рева
“ ” _____ Г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

УЧЕБНОЙ ДИСЦИПЛИНЫ

Управление качеством разработки программного обеспечения

Образовательная программа: 09.04.04 Программная инженерия, магистерская программа:

Разработка программного обеспечения информационных систем

1. Обобщенная структура фонда оценочных средств учебной дисциплины

Обобщенная структура фонда оценочных средств по дисциплине Управление качеством разработки программного обеспечения приведена в Таблице.

Таблица

Формируемые компетенции	Показатели сформированности компетенций (знания, умения, навыки)	Темы	Этапы оценки компетенций	
			Мероприятия текущего контроля (курсовой проект, РГЗ(Р) и др.)	Промежуточная аттестация (экзамен, зачет)
ПК.22.В способность управлять средой функционирования объектов профессиональной деятельности	34. типовые метрики программного обеспечения	Характеристики качества ПО		Зачет, вопросы 3-6
ПК.22.В	36. основные методы измерения и оценки характеристик программного обеспечения	Виды сложности при разработке и эксплуатации ПО	РГЗ	Зачет, вопрос 7
ПК.22.В	у3. оценивать работоспособность программного продукта	Виды сложности при разработке и эксплуатации ПО Основные понятия надежности ПО	РГЗ	Зачет, вопросы 7,11,12
ПК.22.В	у4. применять методы и средства проверки работоспособности программного обеспечения	Основные понятия надежности ПО	РГЗ	Зачет, вопросы 11,12
ПК.23.В способность к управлению процессами жизненного цикла программного обеспечения	35. методологии разработки программного обеспечения	Основные понятия и определения Сертификация и система сертификации. Характеристики качества ПО		Зачет, вопросы 14,15
ПК.6/НИ пониманием существующих подходов к верификации моделей программного обеспечения	31. методы верификации моделей проектирования и программного обеспечения	Виды корректности ПО		Зачет, вопрос 8
ПК.6/НИ	у1. применять методы верификации моделей проектирования и программного обеспечения	Виды корректности ПО		Зачет, вопрос 8

2. Методика оценки этапов формирования компетенций в рамках дисциплины.

Промежуточная аттестация по дисциплине проводится в 2 семестре - в форме зачета, который направлен на оценку сформированности компетенций ПК.22.В, ПК.23.В, ПК.6/НИ.

Зачет проводится в устной форме по билетам

Кроме того, сформированность компетенций проверяется при проведении мероприятий текущего контроля, указанных в таблице раздела 1.

В 2 семестре обязательным этапом текущей аттестации является расчетно-графическое задание (работа) (РГЗ(Р)). Требования к выполнению РГЗ(Р), состав и правила оценки сформулированы в паспорте РГЗ(Р).

Общие правила выставления оценки по дисциплине определяются балльно-рейтинговой системой, приведенной в рабочей программе дисциплины.

На основании приведенных далее критериев можно сделать общий вывод о сформированности компетенций ПК.22.В, ПК.23.В, ПК.6/НИ, за которые отвечает дисциплина, на разных уровнях.

Общая характеристика уровней освоения компетенций.

Ниже порогового. Уровень выполнения работ не отвечает большинству основных требований, теоретическое содержание курса освоено частично, пробелы могут носить существенный характер, необходимые практические навыки работы с освоенным материалом сформированы не достаточно, большинство предусмотренных программой обучения учебных заданий не выполнены или выполнены с существенными ошибками.

Пороговый. Уровень выполнения работ отвечает большинству основных требований, теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые виды заданий выполнены с ошибками.

Базовый. Уровень выполнения работ отвечает всем основным требованиям, теоретическое содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные программой обучения учебные задания выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые из выполненных заданий, возможно, содержат ошибки.

Продвинутый. Уровень выполнения работ отвечает всем требованиям, теоретическое содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному.

Паспорт зачета

по дисциплине «Управление качеством разработки программного обеспечения», 2 семестр

1. Методика оценки

Зачет проводится в устной форме, по билетам. Билет формируется по следующему правилу: первый вопрос выбирается из диапазона вопросов 1-7, второй вопрос из диапазона вопросов 8-15 (список вопросов приведен ниже). В ходе зачета преподаватель вправе задавать студенту дополнительные вопросы из общего перечня (п. 4).

Форма билета для зачета

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Факультет АВТФ

Билет № _____

к зачету по дисциплине «Управление качеством разработки программного
обеспечения»

Вопрос 1. Понятие метрики. Классификация метрических шкал: относительные, интервальные, порядковые, категорийные шкалы. Метрики размера программ

Вопрос 2. Аналитические, имитационные, экспериментальные методы определения показателей надежности. Моделирование и обеспечение надежности при создании ПО.

Утверждаю: зав. кафедрой _____ должность, ФИО
(подпись) (дата)

2. Критерии оценки

Согласно положению о балльно-рейтинговой системе НГТУ, базовый балл рейтинга за зачет составляет 20 баллов. Соответственно, критерий оценки определяется в процентах к этому баллу:

- Ответ на билет для зачета считается **неудовлетворительным**, если студент при ответе на вопросы не дает определений основных понятий, не способен показать причинно-следственные связи явлений, при решении задачи допускает принципиальные ошибки, оценка составляет менее 50% базовой
- Ответ засчитывается на **пороговом** уровне, если в теоретических вопросах даны только основные определения - оценка составляет не более 50% базовой
- Ответ засчитывается на **базовом** уровне, если в теоретических вопросах отражена структура вопроса (определения, свойства, правила) - оценка составляет 50-80% базовой

- Ответ засчитывается на **продвинутом** уровне, если дан развернутый ответ на теоретический вопрос и доп. вопросы - оценка составляет 80-100% базовой

3. Шкала оценки

В общей оценке по дисциплине баллы за зачет учитываются в соответствии с правилами балльно-рейтинговой системы, приведенными в рабочей программе дисциплины.

4. Вопросы к зачету по дисциплине «Управление качеством разработки программного обеспечения»

Список вопросов

1. Основные понятия и определения. Задача количественной оценки качества программного обеспечения. Основы метрологической оценки ПО. Задачи метрологии качества ПО.
2. Стандарты управления качеством ПО. Сущность стандартизации, роль и место стандартизации в производстве и применении программного обеспечения, нормативные документы по стандартизации и виды стандартов.
3. Характеристики качества ПО. Система качества стандарта ISO 9126: характеристики качества, показатели характеристик. Система качества ГОСТ 28195-89: факторы и критерии качества программного обеспечения, метрики и оценочные элементы.
4. Понятие метрики. Классификация метрических шкал: относительные, интервальные, порядковые, категориальные шкалы. Метрики размера программ.
5. Метрики стилистики и понятности программы, метрики Холстеда. Метрики сложности потока управления программы: цикломатическая метрика МакКейба, метрика Майерса, метрика Джилба, метрика граничных значений.
6. Метрики сложности потока данных программы: метрика обращения к глобальным переменным, метрика Спена, метрика Чепина. Метрики инкапсуляции, наследования, полиморфизма.
7. Виды сложности при разработке и эксплуатации ПО. Временная, программная, информационная сложности. Измерение и оценка сложности ПО.
8. Виды корректности ПО. Функциональная, детерминированная, стохастическая, динамическая корректности.
9. Тестирование структуры ПО. Типы эталонов, методы измерений и проверки корректности ПО. Классификация ошибок ПО. Причины ошибок. Обнаружение и устранение ошибок.
10. Спецификации программ, анализ корректности. Автоматизация верификации программ
11. Основные понятия надежности ПО, методы измерения. Методы обеспечения надежности. Показатели надежности. Определение показателей надежности.
12. Аналитические, имитационные, экспериментальные методы определения показателей надежности. Моделирование и обеспечение надежности при создании ПО.
13. Тестирование программ. Инструментальные средства измерений и оценки качества программного обеспечения.
14. Сертификация и система сертификации. Добровольная и обязательная сертификация. Виды сертификационных испытаний ПО.
15. Стандарты сертификации ПО. Формы подтверждения соответствия. Аккредитация органов сертификации и испытательных лабораторий.

Паспорт расчетно-графического задания (работы)

по дисциплине «Управление качеством разработки программного обеспечения», 2 семестр

1. Методика оценки

Индивидуальное задание по материалам программных проектов, выполненных студентом (например, ВКРБ): комплексная оценка проекта на основе метрик программного обеспечения:

- поиск и установка компонент получения метрик в IDE
- сбор статистики метрик программного проекта
- комплексная оценка качества проекта на основе анализа статистики
- "ручное" вычисление метрики на отдельные части проекта при отсутствии средств автоматизации
- определение узких мест проекта с точки зрения метрики ПО
- комплексная оценка качества проекта и разработка рекомендаций по рефакторингу

2. Критерии оценки

Согласно положению о балльно-рейтинговой системе НГТУ, базовый балл рейтинга за РГР определен в рабочей программе. Соответственно, критерий оценки определяется в процентах к этому баллу:

- РГР считается **не выполненной**, если выполнены не все части РГЗ(Р), оценка составляет менее 30% базовой.
- РГР засчитывается на **пороговом** уровне, содержание не полностью раскрывает задание, список источников обзора ограничен, пояснительная записка оформлена со значительными структурными, стилистическими и грамматическими ошибками - оценка составляет 30-50% базовой
- РГР засчитывается на **базовом** уровне, если содержание работы соответствует заданию, пояснительная записка оформлена в целом грамотно - оценка составляет 50-80% базовой
- РГР засчитывается на **продвинутом** уровне, если содержание раскрыто полностью и разнообразно, количество источников обзора более 30, в пояснительной записке отражены все аспекты тематики, имеется аналитическая часть - оценка составляет 80-100% базовой

3. Шкала оценки

В общей оценке по дисциплине баллы за РГЗ(Р) учитываются в соответствии с правилами балльно-рейтинговой системы, приведенными в рабочей программе дисциплины.

4. Примерный перечень тем РГЗ(Р)

Темы определяются индивидуально из проектов, выполненных студентом, либо из проектов с открытым кодом с репозиториях типа github, sourceforce