

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кузбасский государственный технический университет
имени Т. Ф. Горбачева»

Институт профессионального образования
Кафедра информатики и информационных систем

Константин Андреевич Кулиничев

УПРАВЛЕНИЕ ПРОЕКТАМИ

Методические материалы к практическим занятиям,
лабораторным и самостоятельным работам

Рекомендовано цикловой методической комиссией по
специальности СПО 09.02.07 Информационные системы
и программирования в качестве электронного издания
для использования в образовательном процессе

Кемерово 2024

Рецензенты: Ерошевич К.В. – преподаватель кафедры информатики и информационных систем ИПО ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Кулиничев, А.К. Управление проектами: методические материалы к практическим занятиям, лабораторным и самостоятельным работам для обучающихся специальности СПО 09.02.07 «Информационные системы и программирование» / сост. К. А. Кулиничев, Кузбасский государственный технический университет имени Т. Ф. Горбачева. – Кемерово, 2024. – Текст: электронный.

Методические материалы к практическим занятиям, лабораторным и самостоятельным работам для дисциплины «Управление проектами» описывают содержание практических, лабораторных занятий и самостоятельной работы, перечень вопросов на защиту выполненных работ.

© Кузбасский государственный
технический университет
имени Т. Ф. Горбачева,
2024

© Кулиничев К. А.,
составление, 2024

Оглавление

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1. Использование метрик программного продукта	4
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2. Использование метрик стилистики	9
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 3. Выполнение измерений характеристик кода в среде VisualStudio	11
ЛАБОРАТОРНАЯ РАБОТА № 1. Проверка целостности программного кода	16
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4. Анализ потоков данных	18
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 5. Выполнение измерений характеристик кода в среде (например, Eclipse C/C++ и др.)	20
СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ	25
УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ	26

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1.

Использование метрик программного продукта

Целью работы является изучение метрик потока данных компьютерных программ. Результатом практической работы является отчет, в котором должны быть приведены метрические параметры потока данных программ.

Метрики сложности программ

При оценке сложности программ, как правило, выделяют три основные группы метрик: метрики размера программ, метрики сложности потока управления программ и метрики сложности потока данных программ.

Оценки первой группы наиболее просты и поэтому получили широкое распространение. Традиционной характеристикой размера программ является количество строк исходного текста. Под строкой понимается любой оператор программы.

Непосредственное измерение размера программы, несмотря на свою простоту, дает хорошие результаты. Оценка размера программы недостаточна для принятия решения о ее сложности. Но вполне применима для классификации программ, существенно различающихся объемами. При уменьшении различий в объеме программ на первый план выдвигаются оценки других факторов, оказывающих влияние на сложность. Таким образом, оценка размера программы есть оценка по номинальной шкале, на основе которой определяются только категории программ без уточнения оценки для каждой категории.

К группе оценок размера программ можно отнести также метрику Холстеда. За базу принят подсчет количества операторов и операндов, используемых в программе, т.е. определение размера программы.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы: $h1$ – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов); $h2$ – число уникальных операндов программы (словарь операндов); $N1$ – общее число операторов в программе $N2$ – общее число операндов в программе.

Опираясь на эти характеристики, получаемые непосредственно при анализе исходных текстов программ, Холстед вводит следующие оценки:

Словарь программы:

$$h = h_1 + h_2 \quad (1)$$

Длину программы:

$$N = N_1 + N_2 \quad (2)$$

Объем программы:

$$V = N \log_2 h \quad (3)$$

Смысл оценок h и N достаточно очевиден, поэтому подробно рассмотрим только характеристику V .

Количество символов, используемых при реализации некоторого алгоритма, определяется в числе прочих параметров и словарей программы h , представляющим собой минимально необходимое число символов, обеспечивающих реализацию алгоритма.

Далее Холстед вводит h^* – теоретический словарь программы, т.е. словарный запас, необходимый для написания программы с учетом того, что необходимая функция уже реализована в данном языке и, следовательно, программа сводится к вызову этой функции. Например, согласно Холстеду, возможное осуществление процедуры выделения простого числа могло бы выглядеть так:

$$CALL SIMPLE (X, Y), \quad (4)$$

где Y – массив численных значений, содержащих искомое число X .

Теоретический словарь в этом случае будет состоять из

$n_1^*: \{CALL, SIMPLE (...)\}, n_1^*=2;$

$n_2^*: \{X, Y\}, n_2^*=2;$

а его длина, определяемая как

$$h^* = h_1^* + h_2^* \quad (5)$$

будет равна 4.

Используя h^* , Холстед вводит оценку V^* :

$$V^* = h^* \log_2 h^* \quad (6)$$

с помощью которой описывается потенциальный объем программы, максимально компактно реализующей данный алгоритм.

Другая группа метрик сложности программ – метрика сложности потока данных, то есть использования, конфигурации и размещения данных в программах.

Пара «модуль – глобальная переменная» обозначается как (p, r) , где p – модуль, имеющий доступ к глобальной переменной r . В зависимости от наличия в программе реального обращения к переменной r формируются два типа пар «модуль – глобальная переменная»: фактические и возможные. Возможное обращение к r с помощью p показывает, что область существования r включает в себя p .

Характеристика A_{ip} говорит о том, сколько раз модули U_p действительно получили доступ к глобальным переменным, а число R_{ip} – сколько раз они могли бы получить доступ.

Отношение числа фактических обращений к возможным определяется

$$R_{ip} = A_{ip}/P_{ip}. \quad (7)$$

Эта формула показывает приближенную вероятность ссылки произвольного модуля на произвольную глобальную переменную. Очевидно, чем выше эта вероятность, тем выше вероятность «несанкционированного» изменения какой-либо переменной, что может существенно осложнить работы, связанные с модификацией программы.

Покажем расчет метрики «модуль – глобальная переменная». Пусть в программе имеются три глобальные переменные и три подпрограммы. Если предположить, что каждая подпрограмма имеет доступ к каждой из переменных, то мы получим девять возможных пар, то есть $R_{ip}=9$. Далее пусть первая подпрограмма обращается к одной переменной, вторая – к двум, а третья не обращается ни к одной переменной. Тогда, $A_{ip} = 3, R_{ip} = 3/9$.

Еще одна метрика сложности потока данных – спен.

Определение спена основывается на локализации обращения к данным внутри каждой программной секции. Спен – это число утверждений, содержащих данный идентификатор, между его первым и последним появлением в тексте программы. Идентификатор, появившийся n раз, имеет спен, равный $n-1$.

Спен определяет количество контролируемых утверждений, вводимых в тело программы при построении трассы программы по этому идентификатору в процессе тестирования и от-

ладки.

Следующей метрикой сложности потока данных программ является метрика Чепина. Существует несколько ее модификаций. Рассмотрим более простой, а с точки зрения практического использования – достаточно эффективный вариант этой метрики.

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы:

- P – вводимые переменные для расчетов и для обеспечения вывода. Примером может служить используемая в программах лексического анализатора переменная, содержащая строку исходного текста программы, то есть сама переменная не модифицируется, а только содержит исходную информацию.
- M – модифицируемые или создаваемые внутри программы переменные.
- C – переменные, участвующие в управлении работой программного модуля (управляющие переменные).
- T – неиспользуемые в программе («паразитные») переменные.

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе.

Далее вводится значение метрики Чепина:

$$Q = a_1P + a_2M + a_3C + a_4T, \quad (8)$$

где a_1, a_2, a_3, a_4 – весовые коэффициенты.

Весовые коэффициенты используются для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный трем, имеет функциональная группа C , так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: $a_1=1$; $a_2=2$; $a_4=0.5$.

Весовой коэффициент группы T не равен нулю, поскольку «паразитные» переменные не увеличивают сложности потока данных программы, но иногда затрудняют ее понимание. С учетом весовых коэффициентов выражение примет вид:

$$Q = P + 2M + 3C + 0.5T. \quad (9)$$

Следует отметить, что рассмотренные метрики сложности программы основаны на анализе исходных текстов программ, что обеспечивает единый подход к автоматизации их расчета.

Задание

1. Необходимо произвести вычисления параметров метрики потока данных программ, для этого необходимо использовать формулы расчета для метрики сложности программ.

2. Оформить отчет с подробным описанием хода выполнения работы.

3. Отчет сдается в распечатанном и электронном (файл Word) видах.

Контрольные вопросы

1. Какие метрики оценки сложности программ существуют?
2. Какие характеристики составляют метрику Холстеда?
3. Что такое метрика сложности потока данных?
4. Что такое спен?
5. Как рассчитывается метрика Чепина?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2.

Использование метрик стилистики

Целью работы является изучение метрик стилистики компьютерных программ. Результатом практической работы является отчет, в котором должны быть приведены метрические параметры стилистики.

Метрики стилистики программ

За время своего существования программирование перестало быть искусством отдельных исполнителей, превратившись в предмет коллективной производственной деятельности. Язык программирования и сами программы являются не только средством общения программиста с компьютером, но и средством общения программистов между собой. Появляются новые требования к читаемости и воспринимаемости программ, соблюдение которых упрощает контакт внутри группы разработчиков, а также позволяет обслуживать и корректировать программы без участия непосредственных разработчиков.

Наиболее простой метрикой стилистики и понятности является оценка уровня комментированности программы F :

$$F = N_{kom}/N_{str}, \quad (10)$$

где N_{kom} – количество комментариев в программе; N_{str} – количество строк или операторов исходного текста.

Таким образом, метрика F отражает насыщенность программы комментариями.

Исходя из практического опыта принято считать, что $F > 0.1$, т.е. на каждые десять строк программы должен приходиться минимум один комментарий. Как показывают исследования, комментарии распределяются по тексту программы неравномерно: в начале программы их избыток, а в середине или в конце – недостаток. Это объясняется тем, что в начале программы, как правило, расположены операторы описания идентификаторов, требующие более «плотного» комментирования. Кроме того, в начале программы также расположены «шапки», содержащие общие сведения об исполнителе, характере, функциональном назначении программы и т.п. Такая насыщенность компенсирует недостаток комментариев в теле программы, и поэтому приве-

денная формула недостаточно точно отражает комментированность функциональной части текста программы.

Более удачен вариант, когда вся программа разбивается на n равных сегментов и для каждого из них определяется Fi .

Уровень комментированности программы считается нормальным, если выполняется условие: $F = n$. В противном случае какой-либо фрагмент программы дополняется комментариями до нормального уровня.

Необходимо подчеркнуть, что стилистика и понятность программ тесно связана и с размером, и со сложностью программ. Поэтому не следует забывать о некоторой условности группировки метрик программ.

Задание

1. Необходимо произвести вычисления параметров метрики потока данных программ, для этого необходимо использовать формулы расчета для метрики стилистики программ

2. Оформить отчет с подробным описанием хода выполнения работы.

3. Отчет сдается в распечатанном и электронном (файл Word) видах.

Контрольные вопросы

1. Для чего рассчитываются метрики стилистики?
2. Какие метрики стилистики существуют?
3. Как рассчитывается оценка уровня комментированности программы?
4. Какой уровень комментированности считается нормальным?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 3.

Выполнение измерений характеристик кода в среде VisualStudio

Целью работы является выполнение измерений характеристик кода в среде VisualStudio. Результатом практической работы является отчет, в котором должны быть приведены результаты измерений.

Метрики кода

Метрики кода, вычисляемые Visual Studio

- **Индекс удобства обслуживания** – вычисляет значение индекса от 0 до 100, которое представляет относительную простоту обслуживания кода.

Высокое значение означает лучшую поддержку. Цветовые оценки можно использовать для быстрого выявления проблемных мест в коде. Зеленая оценка составляет от 20 до 100 и указывает, что код имеет хорошую поддержку. Желтая оценка составляет от 10 до 19 лет и указывает, что код умеренно поддерживается. Красная оценка – это оценка от 0 до 9 и указывает на низкую поддержку.

- **Цикломатическая сложность** – измеряет структурную сложность кода. Он создается путем вычисления количества различных путей кода в потоке программы. Программа, которая имеет сложный поток управления, требует больше тестов для достижения хорошего объема протестированного кода и менее поддерживается.

- **Глубина наследования** – указывает количество различных классов, наследуемых друг от друга, вплоть до базового класса. Глубина наследования аналогична взаимозависимости классов, что изменение базового класса может повлиять на любой из унаследованных классов. Чем выше это число, тем глубже наследование и чем выше потенциал для изменений базового класса, что приведет к критическому изменению. Для глубины наследования низкое значение хорошо, и высокое значение плохо.

- **Объединение классов** – измеряет связь с уникальными классами с помощью параметров, локальных переменных, воз-

вращаемых типов, вызовов методов, универсальных или шаблонных экземпляров, базовых классов, реализаций интерфейсов, полей, определенных во внешних типах и оформлении атрибутов. Хорошая конструкция программного обеспечения определяет, что типы и методы должны иметь высокую сплоченность и низкую взаимозависимость. Высокая взаимозависимость указывает на структуру, которую трудно использовать и поддерживать из-за многих взаимозависимостей на других типах.

- **Строки исходного кода** – указывает точное количество строк исходного кода, присутствующих в исходном файле, включая пустые строки. Эта метрика доступна начиная с Visual Studio 2019 версии 16.4 и Microsoft.CodeAnalysis.Metrics 2.9.5.

- **Строки исполняемого кода** – указывает приблизительное количество строк или операций исполняемого кода. Это число операций в исполняемом коде. Эта метрика доступна начиная с Visual Studio 2019 версии 16.4 и Microsoft.CodeAnalysis.Metrics 2.9.5. Значение обычно является близким совпадением с предыдущей метрикой, **строками исходного кода**, которая является метрикой на основе инструкций MSIL, используемой в устаревшем режиме.

Отображение результатов метрик кода

Окно "**Результаты метрик кода**" отображается автоматически при создании результатов метрик кода. Окно также можно отобразить в любое время.

Окно результатов метрик кода можно отобразить с помощью одной из следующих последовательностей меню:

- В меню "**Анализ**" выберите "**Результаты метрики кода Windows**".
- В меню "**Вид**" выберите "**Другие метрики кода Windows**".

Откроется окно "**Результаты метрик кода**", даже если оно не содержит результатов.

Просмотр сведений о метриках кода

Если были созданы результаты метрик кода, разверните дерево в столбце **Hierarchy**.

Фильтрация результатов метрик кода

Вы можете отфильтровать результаты, отображаемые в окне **"Результаты метрик кода"**, с помощью панели инструментов в верхней части экрана. Например, может потребоваться просмотреть только результаты с индексом удобства обслуживания ниже 65.

Раскрывающийся список **"Фильтр"** содержит имена столбцов результатов. При определении фильтра он добавляется в нижнюю часть списка вместе с отступом. Список может содержать последние 10 фильтров, которые были определены.

Для фильтрации результатов метрик кода необходимо:

1. В списке фильтров выберите имя столбца.
2. В поле **Min** введите минимальное значение, которое будет отображаться.
3. В поле **Max** введите максимальное значение, которое будет отображаться.
4. Нажмите кнопку **"Применить фильтр"**.
5. Чтобы просмотреть сведения о результатах, разверните дерево иерархии.

Добавление, удаление и изменение порядка столбцов данных

Столбцы результатов можно добавлять или удалять из окна **"Результаты метрик кода"**. Кроме того, можно изменить порядок столбцов результатов, чтобы они отображались в нужном порядке.

Для добавления или удаления столбца необходимо:

1. Нажмите кнопку **"Добавить или удалить столбцы"** или щелкните правой кнопкой мыши любой заголовок столбца и выберите команду **"Добавить или удалить столбцы"**.

2. В диалоговом окне **"Добавление и удаление столбцов"** установите или снимите флажок для столбца, который требуется добавить или удалить, и нажмите кнопку **"ОК"**.

Для изменения порядка столбцов необходимо:

1. Нажмите кнопку **"Добавить и удалить столбцы"**.
2. В диалоговом окне **"Добавление и удаление столбцов"** выберите столбец, который требуется переместить, а затем щелкните стрелку вверх или стрелку вниз.

3. Если столбец расположен в нужном месте, нажмите кнопку **"ОК"**.

Копирование данных в буфер обмена или Excel

Вы можете выбрать и скопировать выбранную строку данных метрик кода в буфер обмена в виде текстовой строки, содержащей одну строку для имени и значения каждого столбца данных. Вы также можете щелкнуть **"Открыть выбор"** в **Microsoft Excel**, чтобы экспортировать все результаты метрик кода в электронную таблицу Excel.

Создание рабочего элемента на основе результата

1. Щелкните результат правой кнопкой мыши.
2. Наведите указатель мыши на **создание рабочего элемента** и выберите тип рабочего элемента, который нужно создать (ошибка, задача и т. д.).
3. Заполните форму рабочего элемента, заполнив все обязательные поля.
4. В меню **"Файл"** нажмите кнопку **"Сохранить все"**, чтобы сохранить рабочий элемент.

Создание ошибки на основе результата

1. Щелкните результат, чтобы выбрать его.
2. Нажмите кнопку **"Создать рабочий элемент"**.
3. Заполните форму рабочего элемента, заполнив все обязательные поля.
4. В меню **"Файл"** нажмите кнопку **"Сохранить все"**, чтобы сохранить рабочий элемент.

Задание

1. Провести измерение известных метрик кода на тестовом примере.
2. Оформить отчет с подробным описанием хода выполнения работы.

Контрольные вопросы

1. Какие существуют метрики кода, вычисляемые Visual Studio?
2. Что представляет собой метрика «Индекс удобства обслуживания»?
3. Что представляет собой метрика «Цикломатическая сложность»?
4. Что представляет собой метрика «Глубина наследования»?
5. Что представляет собой метрика «Объединение классов»?
6. Что представляет собой метрика «Строки исходного кода»?
7. Что представляет собой метрика «Строки исполняемого кода»?
8. Как произвести фильтрацию результатов метрик кода?

ЛАБОРАТОРНАЯ РАБОТА № 1. Проверка целостности программного кода

Целью работы является выполнение проверки целостности программного кода. Результатом практической работы является отчет, в котором должны быть приведены результаты проверки целостности программного кода.

Порядок выполнения работы

Исследование выполняется в несколько этапов:

1. В первой части работы студенту предлагается написать простое консольное приложение на языке программирования C#, в котором будет реализовано обращение к собственному методу, например вычисляющему несложное арифметическое выражение, создание объектов, и осуществляться вывод текстовых сообщений. (Возможно использование текстов программ, разработанных студентами в рамках другие дисциплин или примеров программ, распространяющихся вместе с учебной литературой.)

2. Для вычисления хеш-сверток предлагается использовать программы WinMD5deep или WinSHA-1Sum, которые можно бесплатно скачать из Интернета.

3. В текстовом редакторе Microsoft Word необходимо создать таблицу для занесения результатов экспериментов, в которую в первый столбец заносится описание указываемого действия, во втором – хеш-свертка, в третий – выводы (табл. 1).

Таблица 1 – Результаты эксперимента

№ п/п	Действие	Значение хеш-суммы	Вывод
1	Исходная программа	ed1cf0dea20831fb26661c10ca65340e3a3 ea6 16	Не влияет

4. Студент формирует исполняемый файл при помощи указанного программного обеспечения, вычисляет его хеш-сумму, заносит результат в таблицу. Для продолжения выполняет следующие действия:

- изменяет текст выводимых сообщений, формирует исполняемый файл, вычисляет его хеш-сумму, заносит результат в таблицу, проводит сравнение хешей полученного результата и исходной программы и заносит результат сравнения в таблицу;
- выполняет действия, аналогичные предыдущему пункту, изменив имя вызываемого метода;
- выполняет действия, аналогичные предыдущему пункту, изменив имя класса;
- выполняет действия, аналогичные предыдущему пункту, изменив числовые значения переменных в вычисляемом выражении;
- выполняет действия, аналогичные предыдущему пункту, изменив идентификаторы переменных;
- выполняет действия, аналогичные предыдущему пункту, изменив типы используемых в вычисляемом выражении переменных.

5. Во второй части задания студенту предлагается скачать с официального сайта дистрибутив ОС Linux Debian для установки по сети (из-за меньшего объема скачиваемого файла). Перед скачиванием сохранить рассчитанную по алгоритму SHA-1 хеш-сумму. Затем вычислить хеш-сумму скачанного файла и выполнить сравнение хеш-сумм, полученных в результате эксперимента и предоставленной на сайте поставщика дистрибутива.

6. Оформить отчет, сделав выводы о проделанной работе.

Задание

Необходимо выполнять задание и предоставить отчет, содержащий описание выполняемых действий, снимки экрана, подтверждающие факт выполнения работы.

Контрольные вопросы

1. Какие этапы нужно выполнить, чтобы произвести проверку целостности программного кода?
2. Какие нужно использовать программы для вычисления хеш-сверток?
3. Что такое хеш-сумма?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4. Анализ потоков данных

Целью работы является выполнение анализа потоков данных. Результатом практической работы является отчет, в котором должны быть приведены результаты анализа потоков данных.

Спены

Спеном идентификатора называется число повторных появлений идентификатора в тексте программы. Например, если идентификатор встречается в программе n раз, то его спен равен $n - 1$.

Значение спена идентификатора связано со сложностью тестирования программы, так как при трассировке программы по этому идентификатору придется ввести 10 контрольных точек.

Рассмотрим вычисление метрик потока данных на примере программы, вычисляющей выражение $y = \sin(x)$ (табл. 2).

Таблица 2 – Подсчет суммарного спена программы

Идентификатор	x	n	s	st	y	Суммарный спен программы
Спен	6	5	4	3	2	

Проведем подсчет переменных, входящих в группы, необходимые для вычисления метрики Чепина.

Для расчета метрики ввода вывода Чепина используются только переменные ввода-вывода, поэтому относительно нашего примера используются только переменные x , y и константа cst . Все результаты заносятся в табл. 3, представленную ниже.

Исследование выполняется в несколько этапов:

1. На основании выданного преподавателем задания написать программу на заданном языке программирования. В ходе написания программы реализовать вывод всех входных и выходных данных. Программа должна быть хорошо прокомментирована и описана. В описании должны быть четко указаны назначение и состав используемых входных, выходных и внутренних переменных, а также блоков программы.

2. Используя исходный текст программы, необходимо рассчитать и занести в таблицу аналогично приведенному приме-

ру:

- Полный спен программы
- Полную метрику Чепина
- Метрику Чепина ввода-вывода

3. Сделать выводы по проделанной работе и оформить отчет.

Таблица 3 – Результаты выполнения работы

Переменные	Полная метрика Чепина				Метрика ввода-вывода Чепина			
Группы переменных	<i>P</i>	<i>M</i>	<i>C</i>	<i>T</i>	<i>P</i>	<i>M</i>	<i>C</i>	<i>T</i>
Переменные в группе	<i>x</i>	<i>y, n</i>	<i>bs, cns</i>	–	<i>x</i>	<i>y, n</i>	<i>cns</i>	–
Значения переменных	1	2	2	0	1	1	1	0
Значение метрики	$Q = 1*1 + 2*2 + 3*2 + 0,5*0 = 11$				$Q = 1*1 + 2*1 + 3*1 = 6$			

Задание

Выполнить исследование, согласно примеру практической работы и предоставить отчет, содержащий описание выполняемых действий, снимки экрана с подробным описанием выполненной работы.

Контрольные вопросы

1. Дайте определение понятию спена идентификатора.
2. Какие данные необходимые для вычисления метрики Чепина?
3. О чем говорит показатель размера спена?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 5. Выполнение измерений характеристик кода в среде (например, Eclipse C/C++ и др.)

Дублирование кода с помощью CPD

Плагин Eclipse PMD предоставляет функцию CPD (или детектор копирования и вставки) для поиска дублированного кода.

Чтобы использовать этот удобный инструмент в Eclipse, вам необходимо установить плагин Eclipse с PMD, который имеет функциональность CPD.

Чтобы найти повторяющийся код, щелкните правой кнопкой мыши проект Eclipse и выберите **PMD|Find Suspect Cut and Paste**, как показано на рисунке 1.1:

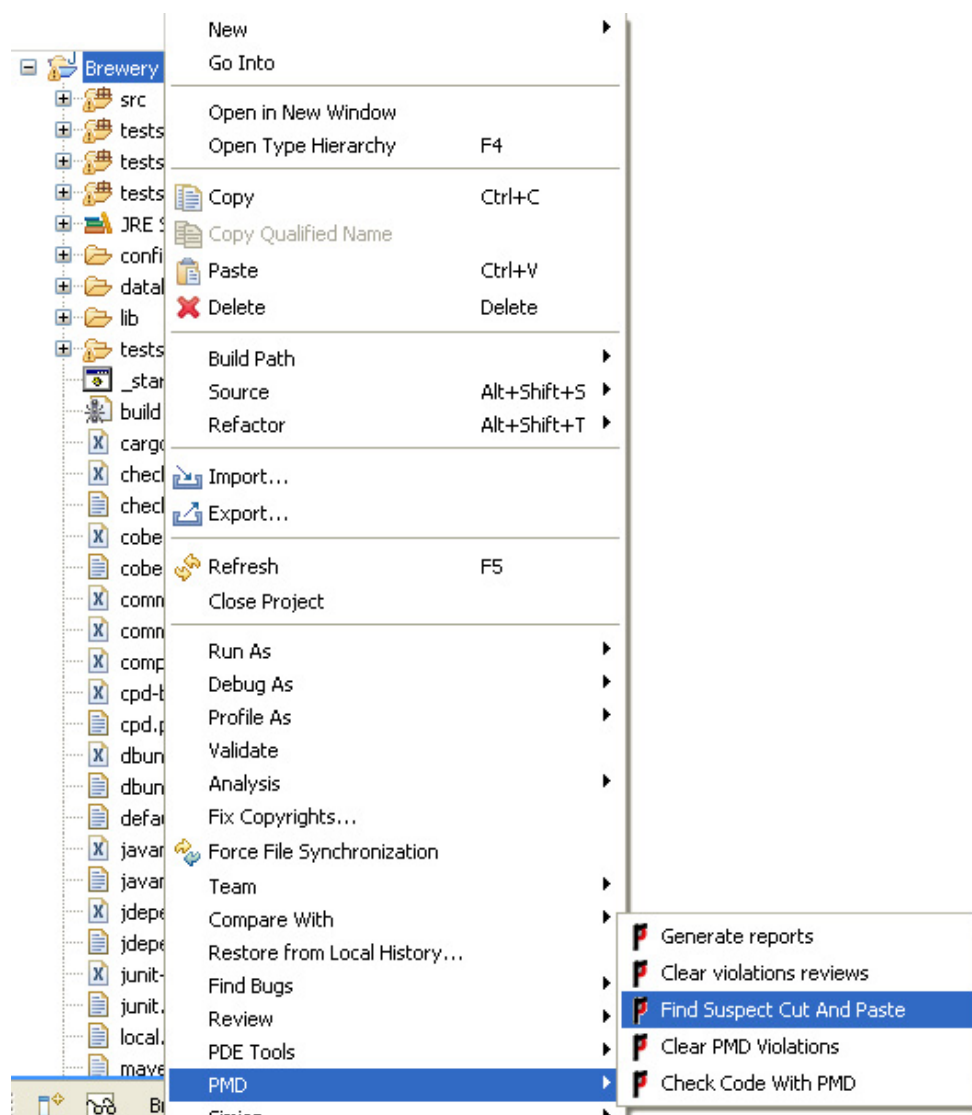


Рисунок 1.1 – Использование плагина CPD для запуска проверки копирования и вставки

После запуска CPD в вашем корневом каталоге Eclipse будет создана папка отчета, в которой будет находиться файл `cpd.txt`, в котором перечислены все дубликаты кода. На рисунке 1.2 показан пример файла `cpd.txt`:

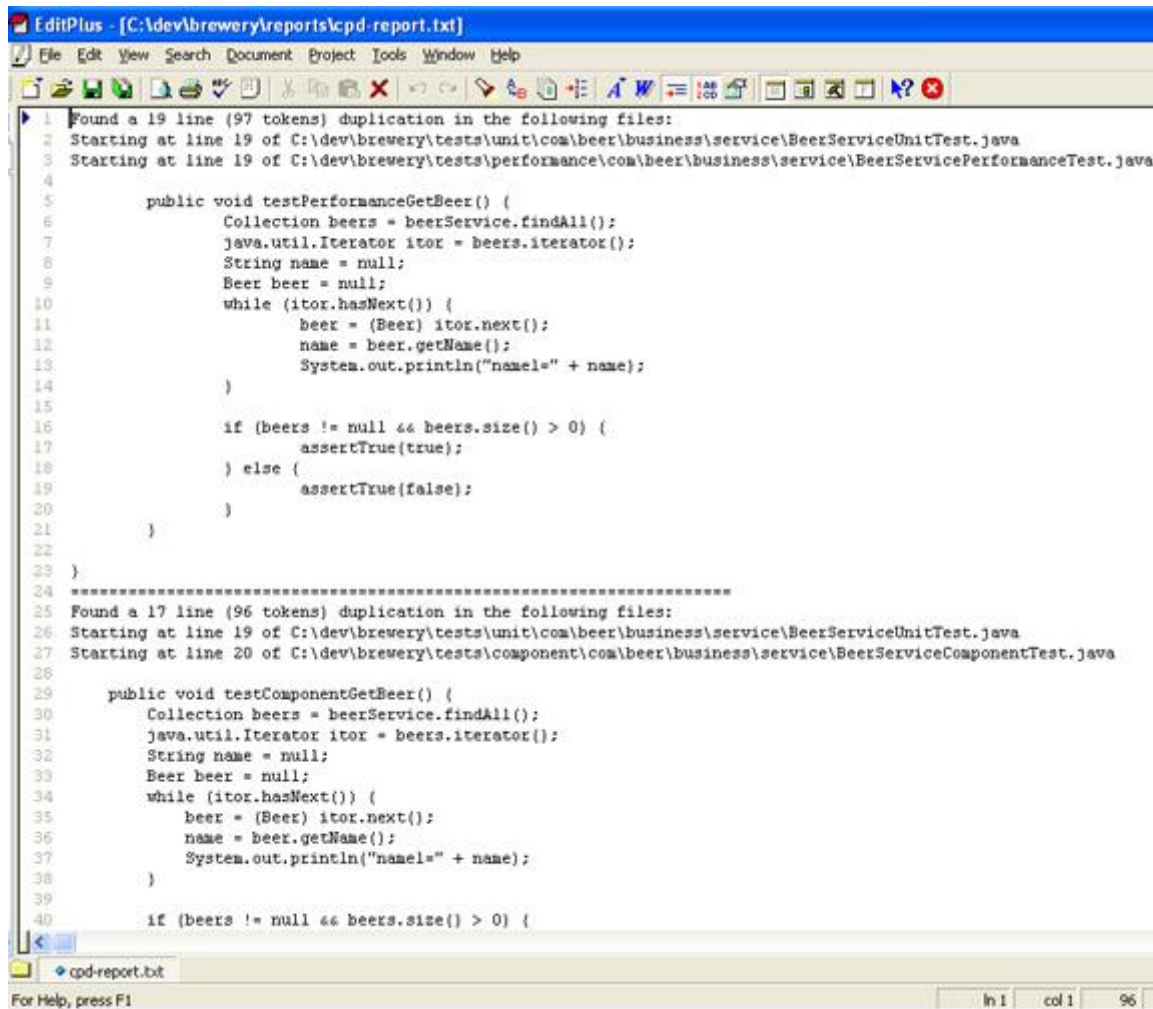


Рисунок 1.2 – Текстовый файл CPD, сгенерированный плагином Eclipse

Поиск дублированного кода вручную – сложная задача, но использование таких плагинов, как CPD, облегчает поиск дублированного кода при кодировании.

Проверка зависимостей с использованием JDepend

JDepend – это свободно доступный инструмент с открытым исходным кодом, который предоставляет объектно-ориентированные метрики для зависимостей пакетов, чтобы ука-

зать устойчивость кода. Другими словами, JDepend может эффективно измерять надежность (и наоборот) архитектуры.

В дополнение к плагину Eclipse JDepend также предоставляет задачу Ant, плагин Maven и приложение Java для получения этих метрик. Для одной и той же информации у них разные механизмы доставки, но особое и соответствующее преимущество подключаемого модуля Eclipse заключается в том, что он может доставлять эту информацию способом, который ближе к исходному коду (т.е. при кодировании).

На рисунке 1.3 показано, как использовать подключаемый модуль Eclipse JDepend: щелкнув правой кнопкой мыши исходную папку и выбрав «Запустить анализ JDepend». Обязательно выберите исходную папку с исходным кодом, иначе вы не увидите этот пункт меню.

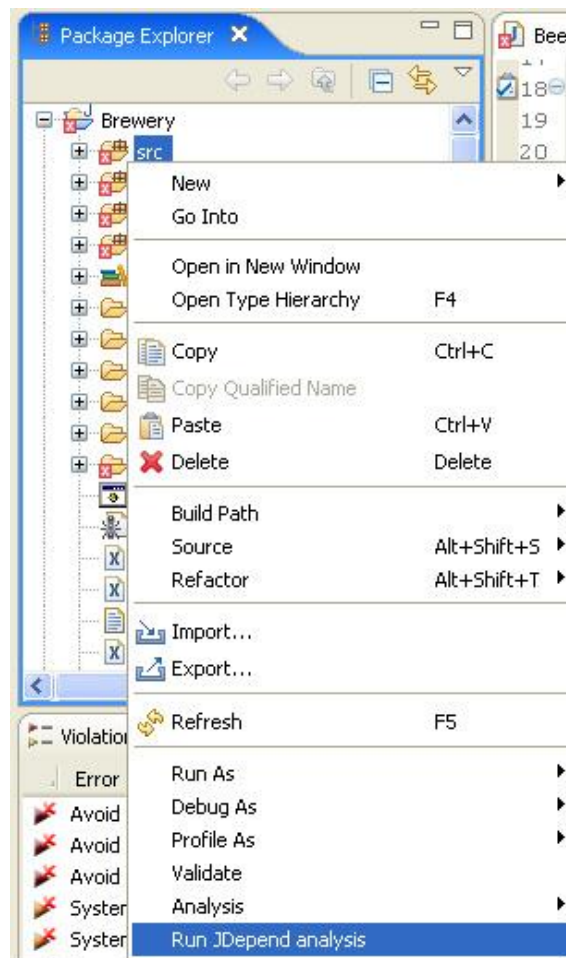


Рисунок 1.3 – Анализ кода с использованием JDepend Analysis

На рисунке 4 показан отчет, сгенерированный при запуске JDepend Analysis. Пакеты отображаются слева, а метрики зависи-

мостей для каждого пакета отображаются справа.

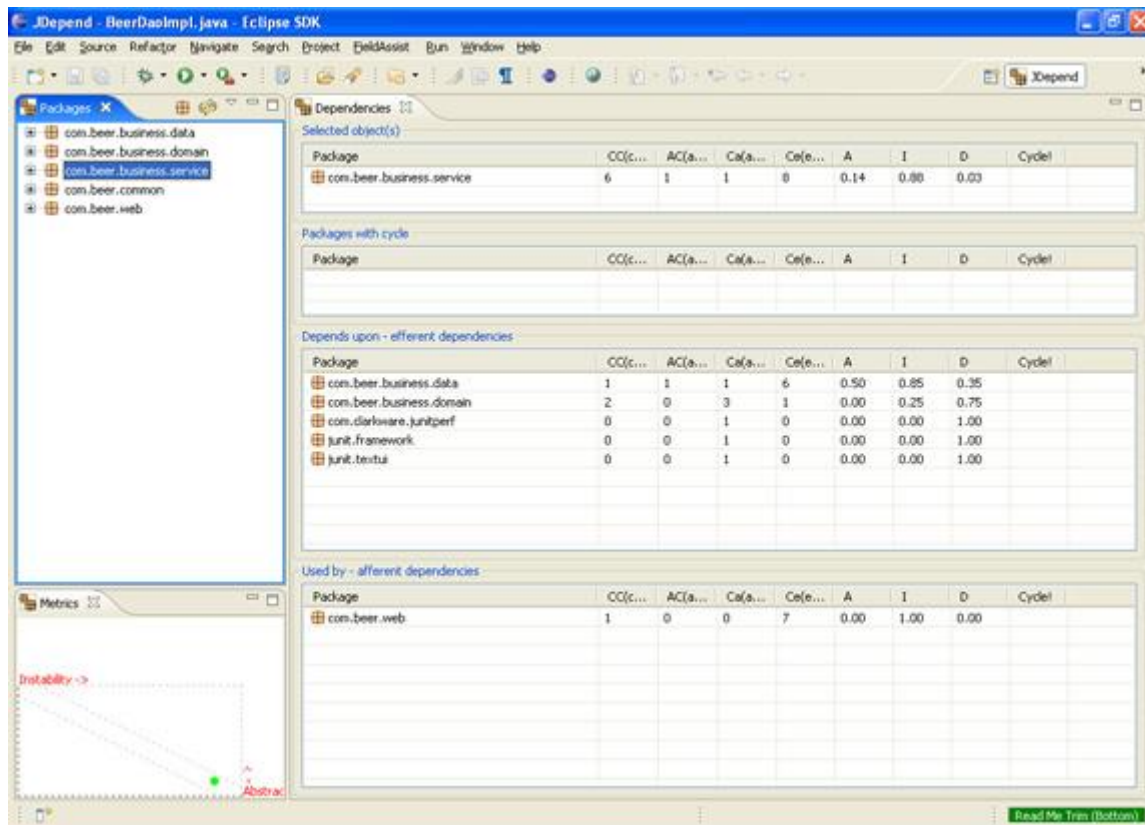


Рисунок 1.4 – Зависимости пакетов в проекте Eclipse

Плагин JDepend предоставляет много информации, которая помогает постоянно наблюдать за изменениями в поддерживаемой архитектуре – самое большое преимущество этого в том, что вы можете видеть эти данные во время кодирования.

Метрики для измерения сложности

Eclipse предоставляет плагин под названием Metrics, который можно использовать для выполнения многих полезных измерений кода, включая измерение цикломатической сложности, который используется для измерения количества уникальных путей в методе.

Установите плагин Metrics и перезапустите Eclipse, затем выполните следующие действия:

1. Щелкните правой кнопкой мыши по вашему проекту и выберите меню «Свойства». В окне результатов установите флажок «Включить метрику» и нажмите «ОК», как показано на рисунке 1.5:

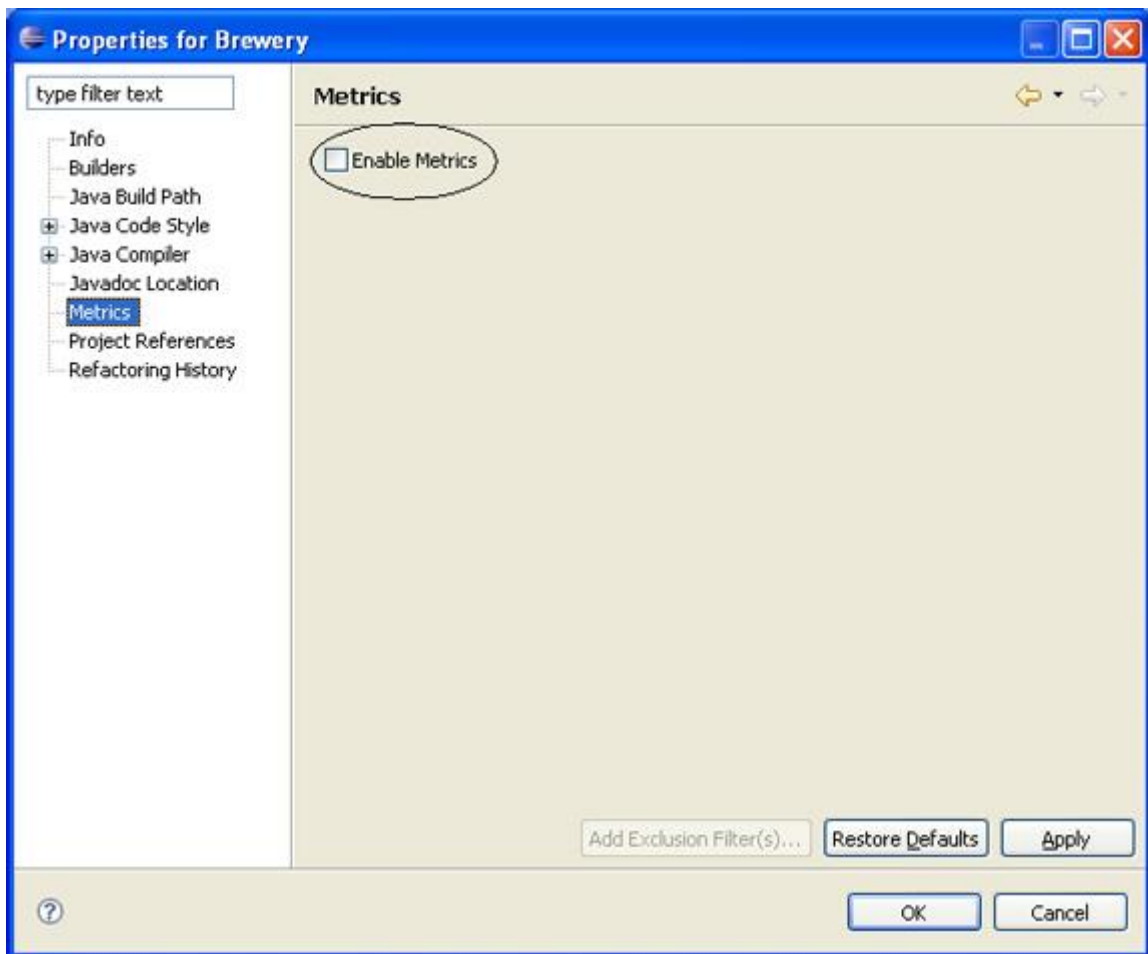


Рисунок 1.5 – Настройка метрик для проекта

2. В Eclipse выберите меню «**Окно**», чтобы открыть представление «**Метрики**», а затем выберите «**Показать представление**» – «**Другое**» ...

3. Выберите **Metrics | Metrics View**, чтобы открыть окно, показанное на рисунке 1.6. Вам нужно использовать перспективу Java и перестроить проект для отображения этих метрик.

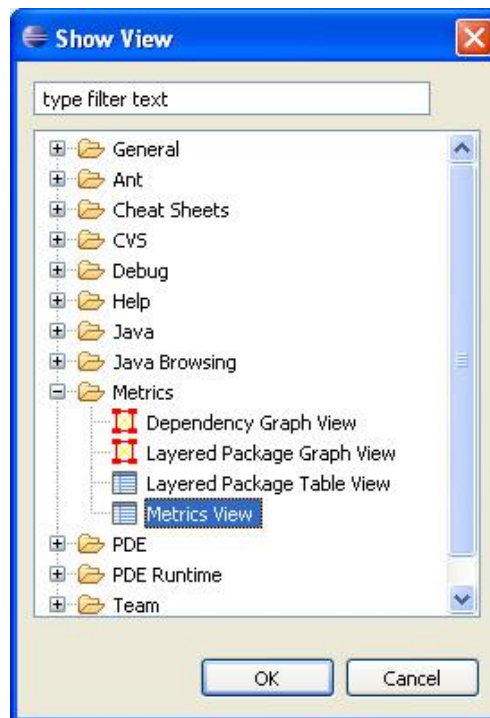


Рисунок 1.6 – Открытое представление метрик в Eclipse

4. Нажмите «ОК», чтобы отобразить окно, показанное на рисунке 1.7. В этом примере рассматривается цикломатическая сложность одного метода. Для нужно дважды щелкнуть метод в списке метрик, и плагин откроет исходный код этого метода в редакторе Eclipse. Это делает коррекцию очень простой (при необходимости).

Metric	Total	Mean	Std. D...	Maximum	Resource causing Maximum
Number of Static Attributes (avg/max per type)	10	0.833	1.462	4	/Brewery/src/com/beer/common/Constants.java
Nested Block Depth (avg/max per method)	1.429	0.695		3	/Brewery/src/com/beer/business/data/BeerDaoImpl.java
Number of Methods (avg/max per type)	38	3.167	2.115	8	/Brewery/src/com/beer/business/domain/Beer.java
Lack of Cohesion of Methods (avg/max per type)	0.127	0.287	0.857		/Brewery/src/com/beer/business/domain/Beer.java
McCabe Cyclomatic Complexity (avg/max per meth...	1.81	1.384		6	/Brewery/src/com/beer/web/ServletController.java
src	1.808	1.442		6	/Brewery/src/com/beer/web/ServletController.java
com.beer.web	4	2.16		6	/Brewery/src/com/beer/web/ServletController.java
com.beer.common	2.5	1.658		5	/Brewery/src/com/beer/common/BaseDao.java
com.beer.business.data	2.25	0.829		3	/Brewery/src/com/beer/business/data/BeerDaoImpl.java
BeerDaoImpl.java	2.25	0.829		3	/Brewery/src/com/beer/business/data/BeerDaoImpl.java
BeerDaoImpl	2.25	0.829		3	/Brewery/src/com/beer/business/data/BeerDaoImpl.java
findAll	3				
findAllStates	3				
create	2				
BeerDaoImpl	1				
BeerDao.java		0	0		

Рисунок 1.7 – Цикломатическая сложность метода просмотра

Задание

1. Выполнить дублирование кода с помощью CPD
2. Произвести проверку зависимостей с использованием JDepend
3. Выполнить измерение цикломатической сложности
4. Оформить отчет, который должен содержать описание выполняемых действий, снимки экрана, подтверждающие факт выполнения работы.

Контрольные вопросы

1. Какие программные продукты можно использовать для измерения характеристик кода?
2. Как выполнить дублирование кода с помощью CPD?
3. Как произвести проверку зависимостей с использованием JDepend?
4. Какие знаете метрики измерения сложности кода?

СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Цель самостоятельной работы обучающихся – получить новые знания по дисциплине «Управление проектами».

Самостоятельная работа необходима для формирования у обучающихся способности самостоятельно решать задачи профессиональной деятельности, формирования умения и навыков планирования времени, формирования стремления развиваться и совершенствоваться.

Виды самостоятельной работы обучающихся указаны в таблице 1.

Таблица 1. Виды самостоятельной работы

№ п/п	Вид СРС
1	Изучение инструментов анализа, поиск и установка вновь разработанных свободно распространяемых программных утилит для анализа
2	Оформление отчетов по практическим работам

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем: учебник для СПО / Черткова Е. А.. – 2-е изд., испр. и доп. – Москва: Юрайт, 2023. – 147 с. – ISBN 978-5-534-09823-5. – URL: <https://urait.ru/book/programmnyaya-inzheneriya-vizualnoe-modelirovanie-programmnyh-sistem-515393> (дата обращения: 09.04.2024). – Текст: электронный.

Программное обеспечение и интернет-ресурсы

2. Официальный сайт Кузбасского государственного технического университета имени Т.Ф. Горбачева. Режим доступа: www.kuzstu.ru

3. Электронные библиотечные системы:

- Университетская библиотека онлайн. Режим доступа: www.biblioclub.ru;

- Лань. Режим доступа: <http://e.lanbook.com>

- Электронно-библиотечная система Znanium.com

- Электронная библиотека издательства Юрайт <https://biblionline.ru/catalog/spo>

4. Информатика и информационные технологии: конспект лекций [Электронный ресурс]. - Режим доступа: <http://fictionbook.ru>

5. Современные тенденции развития компьютерных и информационных технологий: [Электронный ресурс]. - Режим доступа: <http://www.do.sibsutis.ru>

6. Единая коллекция Цифровых образовательных ресурсов [Электронный ресурс]. - Режим доступа: <http://school-collection.edu.ru/>, свободный. - Загл. с экрана.

7. Единое окно доступа к информационным ресурсам [Электронный ресурс]. - Режим доступа: <http://window.edu.ru/>, свободный. - Загл. с экрана.

8. Информационно-коммуникационные технологии в образовании [Электронный ресурс]. - Режим доступа: <http://www.ict.edu.ru/>, свободный. - Загл. с экрана.

9. Федеральный центр информационно-образовательных ресурсов [Электронный ресурс]. - Режим доступа: <http://fcior.edu.ru/>, свободный. - Загл. с экрана.