МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Кафедра информационных и автоматизированных
производственных систем

Составитель
И. В. Чичерин

УПРАВЛЕНИЕ ПРОЕКТАМИ

Методические материалы

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

Кемерово 2018

Рецензенты:

Ванеев О. Н. – кандидат технических наук, доцент кафедры информационных и автоматизированных производственных систем ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Сыркин И. С. – кандидат технических наук, доцент кафедры информационных и автоматизированных производственных систем ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Чичерин Иван Владимирович

Управление проектами [Электронный ресурс]: методические материалы для обучающихся специальности СПО 09.02.07 Информационные системы и программирование очной формы обучения / сост. И. В. Чичерин; КузГТУ. – Электрон. издан. – Кемерово, 2018.

Приведен теоретический и практический материал, необходимый для успешного изучения дисциплины.

Методические материалы дисциплины «Управление проектами» описывают содержание практических и самостоятельных занятий.

ПРЕДИСЛОВИЕ

Целью освоения дисциплины «Управление проектами» является приобретение обучающимися знаний в области измерительных методов оценки программ, исследования программного кода на предмет ошибок и отклонения от алгоритма.

Основными задачами изучения дисциплины «Управление проектами», являются:

1. Измерительные методы оценки программ: назначение, условия применения.
2. Корректность программ. Эталоны и методы проверки корректности
3. Метрики, направления применения метрик. Метрики сложности. Метрики стилистики
4. Исследование программного кода на предмет ошибок и отклонения от алгоритма
5. Программные измерительные мониторы
6. Применение отладчиков и дизассемблера (например OllyDbg, WinDbg, IdaPro)
7. Зашита программ от исследования
8. Исследование кода вредоносных программ

СОДЕРЖАНИЕ ДИСЦИПЛИНЫ В СООТВЕТСТВИИ С УЧЕБНЫМ ПЛАНОМ

В соответствии с учебным планом изучение дисциплины «Управление проектами» предусматривает проведение лекционных, практических занятий и самостоятельной работы обучающимися очной формы обучения.

Общая трудоемкость дисциплины составляет 80 часов.

Промежуточный контроль – экзамен (7 семестр).

СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

При подготовке к практическим занятиям обучающиеся самостоятельно изучают основную и дополнительную литературу, готовят конспекты по темам, предложенным преподавателем.

На практических занятиях преподаватель осуществляет контроль подготовки качества знаний обучающегося, используя: опрос,

обсуждение вопросов по темам изучаемой дисциплины, письменный опрос при текущем контроле и предоставление отчетов по практическим занятиям.

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.

Для выполнения практической работы № 1 студент должен изучить приведенный ниже теоретический материал. Для вычисления параметров метрики потока данных программ необходимо использовать формулы расчета для метрики сложности программ. Отчет сдается в распечатанном и электронном (файл Word) видах.

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 $h=h1 + h2$
Длину программы $N=N1+N2$
Объем программы $V=N \log_2 h$

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),

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

Теоретический словарь в этом случае будет состоять из
n1*: {CALL, SIMPLE (...)} n1*=2;
n2*: {X,Y}, h2*=2;

а его длина, определяемая как
h* = h1* + h2* будет равна 4.

Используя h*, Холстед вводит оценку V*: V* = h*log2h*, с помощью которой описывается потенциальный объем программы, соответствующий 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 Aup говорит о том, сколько раз модули Up действительно получили доступ к глобальным переменным, а число Pup – сколько раз они могли бы получить доступ.

Отношение числа фактических обращений к возможным определяется

Rup=Aup/Pup

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 Pup=9. Далее пусть первая подпрограмма обращается к одной переменной, вторая – двум, а третья не обращается ни к одной переменной. Тогда Aup=3, Rup=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 контролирующих утверждений, вводимых в тело программы при построении трассы программы по этому идентификатору в процессе тестирования и отладки.

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).

Не используемые 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:

$$Q = a1P + a2M + a3C + a4T,$$

где a1, a2, a3, a4 – весовые коэффициенты.

Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный трем,

имеет функциональная группа С, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: a1=1; a2=2; a4=0.5. Весовой коэффициент группы Т не равен нулю, поскольку «паразитные» переменные не увеличивают сложности потока данных программы, но иногда затрудняют ее понимание. С учетом весовых коэффициентов выражение примет вид

$$Q = P + 2M + 3C + 0.5T .$$

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

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.

Для выполнения практической работы № 2 студент должен изучить приведенный ниже теоретический материал. Для вычисления параметров метрики потока данных программ необходимо использовать формулы расчета для метрики стилистики программ. Отчет сдается в распечатанном и электронном (файл Word) видах.

Метрики стилистики программ

За время своего существования программирование перестало быть искусством отдельных исполнителей, превратившись в предмет коллективной производственной деятельности. Язык програм-

мирования 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:

$$F = N_{ком}/N_{стр},$$

где Nком – количество комментариев в программе; Nстр – количество строк или операторов исходного текста.

Таким образом, метрика F отражает насыщенность программы комментариями.

Исходя из практического опыта принято считать, что $F \geq 0.1$, 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 приведенная 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. По-

этому не следует забывать о некоторой условности группировки метрик программ.

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?

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

Обучающиеся должны изучить интернет-ресурсы и литературу по вопросам, представленным в таблице 2.

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. Рудаков, А. В. Технология разработки программных продуктов [Электронный ресурс] : учебник для студентов учреждений среднего профессионального образования, обучающихся по специальности «Программное обеспечение вычислительной техники и автоматизированных систем» : [профессиональный модуль ПМ.03 «Участие в интеграции программных модулей» (МДК.03.01)] / А. В. Рудаков. – Москва : Академия, 2017. – 208 с. – Режим доступа: http://www.academia-moscow.ru/catalogue/4831/362819/. – Загл. с экрана.

Дополнительная литература

1. Рыбалова, Е. А. Управление проектами [Электронный ресурс]. – Томск : Факультет дистанционного обучения ТУСУРа, 2015. – 206 с. – Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=480900. – Загл. с экрана.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И ИНТЕРНЕТ-РЕСУРСЫ

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

2. 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://biblio-online.ru/catalog/spo

3. Информатика и информационные технологии: конспект лекций [Электронный ресурс]. – Режим доступа: http://fictionbook.ru

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

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

6. Единое окно доступа к информационным ресурсам [Электронный ресурс]. – Режим доступа: http://window.edu.ru/, свободный. – Загл. с экрана.

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

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