МИНОБРНАУКИ РОССИИ
Белгородский государственный технологический университет
им. В.Г. Шухова

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

МЕТРОЛОГИЯ, СТАНДАРТИЗАЦИЯ И СЕРТИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания к выполнению лабораторных работ для студентов направления 09.03.04 — Программная инженерия

Белгород
2019

УДК 004
ББК 32.972
М54

Составитель: ст. преп. *Т. В. Бондаренко*

М54 **Метрология, стандартизация и сертификация программного обеспечения**: методические указания, предназначенные для студентов направления 09.03.04 — Программная инженерия / сост.: Т. В. Бондаренко — Белгород: Изд-во БГТУ, 2019. — 33 с.

В методических указаниях представлены основные теоретические сведения по основным разделам метрологии программного обеспечения, достаточные для закрепления материала и выполнения лабораторных работ студентами, а также варианты индивидуальных заданий к лабораторным работам, посвященным метрологии программного обеспечения.

Издание предназначено для студентов направления 09.03.04 — Программная инженерия.

Данное издание публикуется в авторской редакции.

УДК 004
ББК 32.972

Содержание

Лабораторная работа №1
Размерно-ориентированные метрики программного обеспечения

Цель работы: изучить размерно-ориентированные метрики программного обеспечения, получить практические навыки их вычисления и использовать полученные значения метрик программного обеспечения для оценки программного проекта.

Задание к работе

1. Используя данные соответствующего варианта задания, приведенные в таблице 1 выполнить:

– для каждого проекта A, B, C из метрического базиса фирмы вычислить размерно-ориентированные метрики программного обеспечения: производительность, качество, удельную стоимость, документированность;

– рассчитать предполагаемые затраты и стоимость для оцениваемого программного проекта.

2. Составить собственный метрический базис на основании выполненных курсовых работ, расчетно-графических заданий или лабораторных работ и с его помощью оценить свой собственный проект, например, курсовой проект по дисциплине «Базы данных».

Замечание. Для определения производительности использовать 1-й и 2-й подход (см. лекции).

Таблица 1. Метрический базис фирмы

№	Метрический базис фирмы						Оцениваемый проект			
1	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	14,82	2	1,764	163	451	1	15,37	19,98	27,66
	B	15,58	23	2,432	91	691	2	25,40	33,03	45,73
	C	42,52	17	4,436	130	344	3	31,20	40,56	56,15
2	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	10,65	12	2,378	56	54	1	2,34	3,05	4,22
	B	28,27	18	1,366	83	956	2	33,63	43,72	60,53
	C	49,23	10	2,786	70	185	3	7,11	9,25	12,80
3	Проект	LOC,	Затраты, чел-	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш	LOCвер, тыс.	LOCхудш, тыс.

№	Метрический базис фирмы						Оцениваемый проект			
		тыс. строк	мес.					, тыс. строк	строк	строк
	A	3,63	12	3,196	85	73	1	16,62	21,60	29,91
	B	40,01	20	3,048	15	21	2	49,80	64,74	89,64
	C	45,68	11	3,389	138	524	3	40,18	52,24	72,33
4	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	20,42	14	1,315	56	529	1	32,61	42,40	58,70
	B	13,81	15	3,352	127	122	2	7,25	9,43	13,06
	C	38,98	13	1,586	54	449	3	30,64	39,83	55,15
5	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	41,50	12	4,914	41	127	1	4,15	5,40	7,48
	B	22,81	1	4,717	139	464	2	21,32	27,72	38,38
	C	32,19	15	3,666	51	93	3	35,40	46,02	63,73
6	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	21,31	1	2,525	191	955	1	27,82	36,16	50,07
	B	12,84	1	0,387	19	167	2	32,57	42,34	58,63
	C	43,37	7	2,808	155	399	3	12,92	16,80	23,26
7	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	2,03	5	4,317	185	349	1	23,99	31,19	43,19
	B	45,27	10	2,610	103	807	2	10,23	13,30	18,42
	C	5,77	5	4,374	186	600	3	45,69	59,40	82,25

№	Метрический базис фирмы						Оцениваемый проект			
8	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	1,16	18	1,670	109	336	1	3,10	4,03	5,57
	B	38,23	21	3,722	45	959	2	32,78	42,62	59,01
	C	25,21	15	0,629	88	946	3	3,98	5,18	7,17
9	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	42,61	4	3,161	71	164	1	13,55	17,62	24,40
	B	23,49	1	1,457	112	594	2	45,25	58,82	81,45
	C	0,62	2	2,010	164	344	3	35,32	45,91	63,57
10	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	32,34	18	2,287	173	391	1	19,66	25,56	35,39
	B	18,09	24	3,427	171	44	2	27,94	36,32	50,29
	C	44,05	23	4,384	12	550	3	44,42	57,75	79,96
11	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	13,82	3	1,564	160	460	1	14,37	18,98	27,66
	B	18,58	25	2,532	95	700	2	28,40	31,03	46,73
	C	46,52	18	5,436	125	360	3	30,20	39,56	55,15
12	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	11,64	11	2,478	55	53	1	2,24	3,15	4,23
	B	27,2	17	1,466	81	955	2	32,53	42,72	59,53

№	Метрический базис фирмы						Оцениваемый проект			
		6								
	C	48,25	9	2,676	69	184	3	8,17	8,25	11,80
13	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	3,67	13	3,496	83	70	1	17,62	22,60	30,91
	B	40,09	21	3,248	18	25	2	50,80	65,74	90,64
	C	44,65	12	3,589	130	521	3	41,18	53,24	73,33
14	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	19,42	13	1,215	46	429	1	30,61	41,40	57,70
	B	12,81	14	3,252	117	112	2	8,25	10,43	12,06
	C	37,98	12	1,486	44	439	3	29,64	38,83	54,15
15	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	51,50	13	5,014	51	227	1	5,15	6,40	8,48
	B	32,81	3	5,017	189	564	2	22,32	28,72	39,38
	C	42,19	16	4,655	61	103	3	36,40	47,02	64,73
16	Проект	LOC, тыс. строк	Затраты, чел-мес.	Стоимость, т.$	Ошибки, шт	Документация, страниц	Функция	LOC лучш, тыс. строк	LOCвер, тыс. строк	LOCхудш, тыс. строк
	A	20,31	2	2,725	180	905	1	28,82	36,26	49,07
	B	11,04	1	0,287	23	171	2	30,57	41,34	57,63
	C	41,17	6	2,708	165	499	3	13,92	17,80	22,26

Основные теоретические сведения

Размерно-ориентированные метрики

Размерно-ориентированные метрики прямо измеряют программный продукт и процесс его разработки.

Размерно-ориентированные метрики основываются на LOC-оценках (Lines Of Code).

LOC-оценка — это количество строк в коде программного продукта.

Исходные данные для расчета размерно-ориентированных метрик сводятся в таблицу по мере накопления опыта организации.

На основе LOC-оценки могут быть вычислены следующие размерно-ориентированные метрики:

$$\text{Производительность} = \frac{\text{Длина}}{\text{Затраты}} \quad \left[\frac{\text{тыс. LOC}}{\text{чел.}-\text{мес}}\right]$$

$$\text{Качество} = \frac{\text{Ошибки}}{\text{Длина}} \quad \left[\frac{\text{единиц}}{\text{тыс. LOC}}\right]$$

$$\text{Удельная стоимость} = \frac{\text{Стоимость}}{\text{Длина}} \quad \left[\frac{\text{у. е.}}{\text{тыс. LOC}}\right]$$

$$\text{Документированность} = \frac{\text{СтраницДокумента}}{\text{Длина}} \quad \left[\frac{\text{страниц}}{\text{тыс. LOC}}\right]$$

Достоинства размерно-ориентированных метрик:

- широко распространены;
- просты и легко вычисляются.

Недостатки размерно-ориентированных метрик:

- зависимы от языка программирования;
- требуют исходных данных, которые трудно получить на начальной стадии проекта;
- не приспособлены к непроцедурным языкам программирования (например, prolog).

Пример вычисления LOC-метрик.

Таблица 2 содержит данные о проекте за последние несколько лет.

Таблица 2. Исходные данные для расчета LOC-метрик

Проект	Затраты чел.-мес	Стоимость, тыс	KLOC, тыс. LOC	Прогр. док-ты стр	Ошибки	Люди
A1	24	168	12,1	365	29	3
B2	62	440	27,2	1224	86	5
C3	43	314	20,2	1050	64	6

Оценка программного проекта может быть выполнена на основе LOC-метрик. Шаги процесса оценки:

1. Область назначения проектируемого продукта разбивается на ряд функций $f_1,\ldots, f_n$, каждую из которых можно оценить независимо;

2. Для каждой функции f_i планировщик формирует лучшую LOC-оценку ($LOC_{лучш.i}$), худшую (LOC-оценку ($LOC_{худ.i}$) и вероятную LOC-оценку ($LOC_{вер.i}$), при этом используются опытные данные или интуиция.

3. Для каждой функции f_i вычисляется ожидаемое значение LOC-оценки по формуле:

$$LOC_{ожид.i} = \frac{LOC_{лучш.i} + LOC_{худ.i} + 4LOC_{вер.i}}{6}$$

4. Определяется значение производительности разработки функций.

При этом может быть использован один из подходов:

1). для всех функций принимается одна и та же метрика средней производительности, взятая из метрического базиса, обозначаемая $ПРОИЗВ_{ср}$.

2). для каждой функции вычисляется настраиваемая величина производительности по формуле:

$$ПРОИЗВ_i = ПРОИЗВ_{ср} \frac{LOC_{ср.}}{LOC_{ожид.i}}$$

где $LOC_{ср}$ — средняя LOC-оценка, взятая из метрического базиса и соответствующая средней производительности.

3). для каждой функции настраиваемая величина производительности вычисляется по аналогу, взятому из метрического базиса:

$$ПРОИЗВ_i = ПРОИЗВ_{ан.i} \frac{LOC_{ан.i}}{LOC_{ожид.i}}$$

5. Вычисляется общая оценка затрат на проект, по каждому из 3 подходов.

Подход 1):

$$ЗАТРАТЫ = \frac{\sum_{i=1}^{n} LOC_{ожид.i}}{ПРОИЗВ_{ср}} \quad [чел. - мес]$$

Подход 2) и 3):

$$ЗАТРАТЫ = \sum_{i=1}^{n} \frac{LOC_{ожид.i}}{ПРОИЗВ_i} \quad [чел. - мес]$$

6. Вычисление общей оценки стоимости проекта, по каждому из 3 подходов:

Подход 1) и 2):

$$СТОИМОСТЬ = УД.СТОИМОСТЬ_{ср} \sum_{i=1}^{n} LOC_{ожид.i}$$

УД. $СТОИМОСТЬ_{ср}$ — метрика средней стоимости одной строки кода, взятая из метрического базиса.

Подход 3):

$$СТОИМОСТЬ = \sum_{i=1}^{n} LOC_{ожид.i} \times УД.СТОИМОСТЬ_{ср}$$

Контрольные вопросы

1. Понятие метрологии, метрики, меры и измерения.
2. Размерно-ориентированные метрики программного обеспечения: производительность.

3. Размерно-ориентированные метрики программного обеспечения: качество.
4. Размерно-ориентированные метрики программного обеспечения: удельная стоимость.
5. Шкала отношений: понятие, свойства, пример.
6. Размерно-ориентированные метрики программного обеспечения: достоинства и недостатки.
7. Размерно-ориентированные метрики программного обеспечения: производительность.
8. Размерно-ориентированные метрики программного обеспечения: качество.
9. Размерно-ориентированные метрики программного обеспечения: удельная стоимость.
10. Размерно-ориентированные метрики программного обеспечения: документированность.

Лабораторная работа №2.
Функционально-ориентированные метрики программного обеспечения

Цель работы: изучить функционально-ориентированные метрики программного обеспечения; получить практические навыки использования функционально-ориентированные метрики ПО для оценки программного проекта.

Задание к работе

1. Проанализировать возможности программного продукта, выбранного в соответствии с вариантом задания.

Сделать скриншот необходимых экранных форм, характеризующих возможности программного продукта.

2. Выделить в рассматриваемом приложении элементарные процессы и логические файлы.

3. Классифицировать элементарные процессы по типу: внешний ввод, внешний запрос, внешний вывод. Установить ранг сложности программного продукта.

4. Классифицировать файлы по типу: внутренний логический файл, внешний интерфейсный файл. Установить ранг сложности программного продукта.

5. Сводные данные об информационных характеристиках рассматриваемого программного продукта представить в виде таблицы (см. табл. 2.1).

Вместо символа «[]» необходимо подставить соответствующие числовые значения.

Таблица 2.1. Сводные данные об информационных характеристиках приложения

Имя характеристики	Ранг, сложность, количество			
	Низкий	Средний	Высокий	Итого
Внешние вводы	[] x 3 = []	[] x 4 =[]	[] x 6 = []	= []
Внешние выводы	[] x 4 =[]	[] x 5 = []	[] x 7 = []	= []
Внешние запросы	[] x 3 = []	[] x 4 = []	[] x 6 = []	= []
Внутренние логические файлы	[] x 7 =[]	[] x 10 = []	[] x 15 = []	= []
Внешние интерфейсные файлы	[] x 5 =[]	[] x 7 = []	[] x 10 = []	= []
Общее количество				= []

6. Выполнить оценку системных параметров приложения, результат представить в виде таблицы (см. табл. 2.2).

Значения выбираются эмпирически в результате ответа на 14 вопросов, которые характеризуют параметры приложения.

Таблица 2.2. Определение системных параметров приложения

№	Системный параметр	Описание	Значение параметра
1	Передачи данных	Сколько средств связи требуется для передачи или обмена информацией с приложением или системой?	[]
2	Распределенная	Как обрабатываются распределенные данные и	[]

	обработка данных	функции обработки?	
3	Производительность	Нуждается ли пользователь в фиксации времени ответа или производительности?	[]
4	Распространенность используемой конфигурации	Насколько распространена текущая аппаратная платформа, на которой будет выполняться приложение?	[]
5	Скорость транзакций	Как часто выполняются транзакции? (каждый день, каждую неделю, каждый месяц)	[]
6	Оперативный ввод данных	Какой процент информации надо вводить в режиме онлайн?	[]
7	Эффективность работы конечного пользователя	Приложение проектировалось для обеспечения эффективной работы конечного пользователя?	[]
8	Оперативное обновление	Как много внутренних файлов обновляется в онлайновой транзакции?	[]
9	Сложность обработки	Выполняет ли приложение интенсивную логическую или математическую обработку?	[]
10	Повторная используемость	Приложение разрабатывалось для удовлетворения требований одного или многих пользователей?	[]
11	Легкость инсталляции	Насколько трудны преобразование и инсталляция приложения?	[]
12	Легкость эксплуатации	Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления?	[]
13	Разнообразные условия размещения	Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций?	[]
14	Простота изменений	Была ли спроектирована, разработана и поддержана в приложении простота изменений?	[]
Сумма F_i ($\sum_1^{14} F_i$)			[]

Вместо символа «[]» необходимо подставить соответствующие числовые значения.

7. Вычислить метрику количество функциональных точек FP (function points) рассматриваемого приложения по формуле:

$$FP = \text{Общее количество} \cdot (0{,}65 + 0{,}01 \cdot \text{Сумма } F_i)$$

8. Выполнить пункты 1-7 для собственного программного проекта, который был рассмотрен в предыдущей лабораторной работе, например, курсового проекта по дисциплине «Базы данных».

Основные теоретические сведения

Функционально-ориентированные метрики

Функционально-ориентированные метрики косвенно измеряют программный продукт и процесс его разработки.

Вместо подсчета LOC-оценки при этом рассматривается не размер, а функциональность или полезность продукта.

Используется 5 информационных характеристик.

1. Количество внешних вводов.
2. Количество внешних выводов.
3. Количество внешних запросов.
4. Количество внутренних логических файлов.
5. Количество внешних интерфейсных файлов.

Внешний ввод — элементарный процесс, перемещающий данные из внешней среды в приложение. Данные могут поступать с экрана ввода или из другого приложения. Данные могут использоваться для обновления внутренних логических файлов.

Внешний вывод — элементарный процесс, перемещающий данные, вычисленные в приложении, во внешнюю среду. Кроме того, в этом процессе могут обновляться внутренние логические файлы. Данные создают отчеты или выходные файлы, посылаемые другим приложениям.

Внешний запрос — элементарный процесс, работающий как с вводимыми, так и с выводимыми данными. Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Его результат — данные, возвращаемые из внутренних логических файлов и внешних интерфейсных файлов.

Внутренний логический файл — распознаваемая пользователем группа логически связанных данных, которая размещена внутри приложения и обслуживается через внешние вводы.

Внешний интерфейсный файл - распознаваемая пользователем группа логически связанных данных, которая размещена внутри другого приложения и поддерживается им.

Каждой из выявленных характеристик ставится в соответствие сложность. Для этого характеристике назначается низкий, средний или высокий ранг, а затем формируется числовая оценка ранга.

Для файлов ранжирование основано на количестве типов элементов-записей и типов элементов данных, входящих в файл.

Тип элемента-записи — подгруппа элементов данных, распознаваемая пользователем в пределах файла.

Тип элемента данных — уникальное не рекурсивное (неповторяемое) поле, распознаваемое пользователем.

Варианты задания

Вариант	Программный продукт
1.	Язык и стандарты ОС Windows
2.	Свойства браузера ОС Windows
3.	Звук ОС Windows
4.	Экран ОС Windows
5.	Параметры меню «Пуск» ОС Windows
6.	Параметры папок ОС Windows
7.	Учетные записи пользователей ОС Windows

8.	Электропитание ОС Windows
9.	Клавиатура ОС Windows
10.	Мышь ОС Windows

Замечание. В качестве оцениваемого программного продукта можно выбрать любой другой стандартный программный продукт, входящий в состав операционной системы.

Контрольные вопросы.

1. Функционально-ориентированные метрики: понятие, достоинства и недостатки.
2. Внешний ввод: понятие, пример.
3. Внешний вывод: понятие, пример.
4. Внешний запрос: понятие, пример.
5. Внутренний логический файл: понятие.
6. Внешний интерфейсный файл: понятие.
7. Тип элемента запись: понятие, пример.
8. Тип элемента данных: понятие, пример.
9. Ранг сложности: понятие, способ оценки.
10. Определение системных параметров приложения.
11. Общее количество функциональных точек: вычислительная формула.
12. Качество, Производительность, Удельная стоимость, Документированность: вычислительные формулы с использованием FP.

Лабораторная работа №3.
Сложность программной системы

Цель работы: изучить теоретические сведения и получить практические навыки построения потокового графа подпрограмм и вычисления метрики цикломатической сложности.

Задание к работе

1. Реализовать программные реализации всех подпрограмм в соответствии с вариантом задания;
2. Для каждой из реализованных подпрограмм выполнить построение потокового графа;
3. Определить базовое множество независимых путей в каждом построенном потоковом графе;
4. Определить цикломатическую сложность для каждой подпрограммы;
5. Определить наборы тестов для каждой подпрограммы, инициирующие выполнение каждого пути из базового множества.
6. Выполнить пункты 1-5 для собственного программного проекта, который был рассмотрен в предыдущей лабораторной работе, например, курсового проекта по дисциплине «Базы данных». В качестве объекта исследование необходимо выбрать одну из подпрограмм собственного проекта.

Основные теоретические сведения

Одной из метрик, характеризующих структурную сложность программ, является цикломатическая сложность.

Цикломатическая сложность — это метрика ПО, которая обеспечивает количественную оценку логической сложности программы. Для ее определения введем вспомогательное определение потокового графа программы.

Потоковый граф — это способ представления программы.

Особенности потоковых графов.

1. Граф строится отображением управляющей структуры программы. В ходе отображения закрывающие скобки условных операторов и операторов циклов рассматриваются как отдельные (фиктивные) операторы.
2. Узлы (вершины) потокового графа соответствуют линейным участкам программы, включают один или несколько операторов программы.
3. Дуги потокового графа отображают поток управления в программе (передачи управления между операторами). Дуга — это ориентированное ребро.
4. Различают операторные и предикатные узлы. Из операторного узла выходит одна дуга, а из предикатного — две дуги.
5. Предикатные узлы соответствуют простым условиям в программе. Составь условие программы отображается в несколько предикатных узлов. Составным называют условие, в котором используется одна или несколько булевых операций (OR, AND).

Пример. Для заданного фрагмента кода выполнить построение потокового графа.

```
if a or b
```

then x
else;

Выполним прямое отображение фрагмента кода в потоковый граф.

Результат показан на рисунке 3.1.

Этот вариант является ошибочным, так как построенный граф не отвечает требованиям, предъявляемым к потоковым графам.

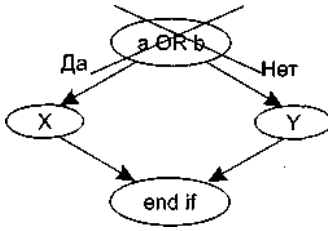

Рис. 3.1. Прямое отображение кода в потоковый граф

Преобразуем полученный граф, разделив составное логическое условие на два простых.

Результат показан на рисунке 3.2.

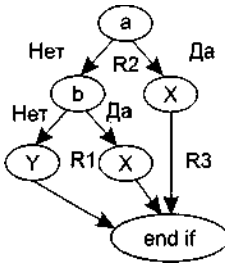

Рис. 3.2 Преобразованный потоковый граф

6. Замкнутые области, образованные дугами и узлами, называют регионами.
7. Окружающая граф среда рассматривается как дополнительный регион.

Потоковый граф, представленный на рисунке 3.2. образует три региона R1, R2, R3. Регион R3 образован окружающей средой графа.

Свойства базового множества:

– тесты, обеспечивающие проверку базового множества, гарантируют: однократное выполнение каждого оператора и выполнение каждого условия по True-ветви и по False-ветви;

– мощность базового множества равна цикломатической сложности потокового графа.

Цикломатическая сложность V(G) потокового графа вычисляется каждым из трех способов:

1) V(G) = число регионов потокового графа;
2) V(G) = число дуг - число узлов + 2;
3) V(G) = число предикатных узлов +1.

Варианты заданий

№	Подпрограмма №1	Подпрограмма №2
1	Сортировка вставками	Бинарный поиск
2	Сортировка выбором	Блочный поиск
3	Улучшенная сортировка обменом 1	Линейный поиск
4	Улучшенная сортировка обменом 2	Быстрый линейный поиск
5	Сортировка Шелла	Бинарный поиск
6	Сортировка вставками	Блочный поиск
7	Сортировка выбором	Линейный поиск
8	Улучшенная сортировка обменом 1	Быстрый линейный поиск
9	Улучшенная сортировка обменом 2	Бинарный поиск
10	Сортировка Шелла	Блочный поиск
11	Сортировка вставками	Линейный поиск
12	Сортировка выбором	Быстрый линейный поиск
13	Улучшенная сортировка обменом 1	Бинарный поиск
14	Улучшенная сортировка обменом 2	Блочный поиск
15	Сортировка Шелла	Линейный поиск
16	Сортировка вставками	Быстрый линейный поиск
17	Сортировка выбором	Бинарный поиск
18	Улучшенная сортировка обменом 1	Блочный поиск
19	Улучшенная сортировка обменом 2	Линейный поиск
20	Сортировка Шелла	Быстрый линейный поиск
21	Сортировка вставками	Бинарный поиск
22	Сортировка выбором	Блочный поиск
23	Улучшенная сортировка обменом 1	Линейный поиск
24	Улучшенная сортировка обменом 2	Быстрый линейный поиск
25	Сортировка Шелла	Бинарный поиск
26	Сортировка вставками	Блочный поиск
27	Сортировка выбором	Линейный поиск
28	Улучшенная сортировка обменом 1	Быстрый линейный поиск
29	Улучшенная сортировка обменом 2	Бинарный поиск
30	Сортировка Шелла	Блочный поиск

Контрольные вопросы

1. Понятие сложности программной системы.
2. Виды сложности программных систем.
3. Структурная сложность программ: понятие.

4. Статистическая сложность программных систем: понятие.
5. Сложность структур данных, информационная сложность: понятие.
6. Временная и программная сложность: понятие.
7. Цикломатическая сложность программ: понятие, способы вычисления.
8. Потоковый граф: определение, структура.
9. Потоковый граф: построение.
10. Операторный и предикатный узел потокового графа: понятие, изображение.
11. Независимый путь в потоковом графе: понятие, пример.
12. Базовое множество потоковых путей: понятие, свойства.

Лабораторная работа №4.
Метрики объектно-ориентированных программных систем

Цель работы: изучить теоретические сведения и получить практические навыки оценки иерархии классов объектно-ориентированных программных систем.

Задание к работе

1. Реализовать диаграмму классов собственной объектно-ориентированной программной системы.
2. Для каждого класса указать все его свойства и методы, кратко охарактеризовать их назначение и смысл.
3. Определить значения метрик из набора метрик Чидамбера и Кемерера.
4. Сформулировать рекомендации по модификации составленной иерархии классов на основании вычисленных значений метрик Чидамбера и Кемерера.
5. Определить значения метрик из набора метрик Лоренца и Кидда.
6. Сформулировать рекомендации по модификации составленной иерархии классов на основании вычисленных значений метрик Лоренца и Кидда

Основные теоретические сведения

Набор метрик Чидамбера и Кемерера.

В 1994 году С.Чидамбер и К.Кемерер (Chidamber и Kemerer) предложили шесть проектных метрик, ориентированных на классы.

Рассмотрим каждую из метрик набора.

1. *Взвешенные методы на класс WMC* (Weighted Methods Per Class). Пусть в классе определены n методов со сложностью $c_1, c_2,\ldots,c_n$. . Для оценки сложности может быть выбрана любая метрика сложности (например, цикломатическая сложность). В этом случае

$$WMC = \sum_{i=1}^{n} c_i$$

Очень часто применяют упрощенную версию метрики. При этом полагают c_i=1, и тогда WMC равно количеству методов в классе.

Возможны два противоположных варианта учета методов в классе.

1. Подсчитывать только методы текущего класса. Унаследованные методы игнорируются.
2. Подсчитываются методы, определенные в текущем классе, и все унаследованные методы. Этот подход подчеркивает важность пространства состояний в понимании класса (а не инкрементности класса).
3. *Высота дерева наследования DIT* (Depth of Inheritance Tree).

DIT определяется как максимальная длина пути от листа до корня дерева наследования классов.

Соответственно, для отдельного класса DIT – это длина максимального пути от данного класса до корневого класса в иерархии классов.

4. *Количество детей NOC* (Number of children).

Подклассы, которые непосредственно подчинены суперклассу, называются его детьми. Значение NOC равно количеству непосредственных наследников класса в иерархии классов.

По мнению Г. Буча, следует строить сбалансированные по высоте и ширине структуры наследования: обычно не выше, чем 7 ± 2 уровня, и не шире, чем 7 ± 2 ветви.

5. *Сцепление между классами объектов CBO* (Coupling between object classes).

CBO – это количество сотрудничеств, предусмотренных для класса, то есть количество классов, с которыми он соединен. Соединение означает, что методы данного класса используют методы или экземплярные переменные другого класса.

6. *Отклик для класса RFC* (Response For a Class).

Множество отклика класса RS – это множество методов, которые могут выполнятся в ответ на прибытие сообщений в объект этого класса.

Формула для определения RS имеет вид:

$$RS = M \bigcup_{i=1}^{n} R_i,$$

где R_i — множество методов, вызываемых методом i; M — множество всех методов в классе, n – количество методов в классе.

Метрика RFC равна количеству методов во множестве отклика, то есть равна мощности этого множества:

$$RFC = card\{RS\}.$$

RFC – это количество методов класса плюс количество методов других классов, вызываемых из данного класса.

7. *Недостаток связности в методах LCOM* (Lack of Cohesion in Methods).

Каждый метод внутри класса обращается к одному или нескольким свойствам (экземплярным переменным). Метрика LCOM показывает, насколько методы не связаны друг с другом через свойства (переменные).

Если все методы обращаются к одинаковым свойствам, то LOCM = 0.

Введем обозначения:

- НЕ СВЯЗАНЫ – количество пар методов без общих экземплярных переменных;
- СВЯЗАНЫ – количество пар методов с общими экземплярными переменными.
- I_j – набор экземплярных переменных, используемых методом M_j.

Очевидно, что

НЕ СВЯЗАНЫ = $card\{I_{ij} \mid I_i \cap I_j = 0\}$,

СВЯЗАНЫ = $card\{I_{ij} \mid I_i \cap I_j \neq 0\}$.

Тогда формула для вычисления недостатка связности в методах примет вид

LCOM = НЕ СВЯЗАНЫ – СВЯЗАНЫ,
если (НЕСВЯЗАНЫ > СВЯЗАНЫ);
LCOM = 0 – в противном случае.

Можно определить метрику по-другому: LCOM – это количество пар методов, не связанных по свойствам класса, минус количество пар методов, имеющих такую связь.

Набор метрик Лоренца и Кидда.

Коллекция метрик М.Лоренца (Lorenz) и Д.Кидда (Kidd) — результат практического, промышленного подхода к оценке ОО-проектов.

1. *Размер класса CS* (Class Size).

Общий размер класса определяется как сумма общего количества операций (вместе с приватными и наследуемыми экземплярными операциями), которые инкапсулируются внутри класса и общего количества свойств (вместе с приватными и наследуемыми экземплярными свойствами), которые инкапсулируются классом.

Рекомендуемое значение CS<=20.

2. *Количество операций, переопределяемых подклассом, NO*O (Number of Operations Overridden by a Subclass).

Переопределением называют случай, когда подкласс замещает операцию, унаследованную от суперкласса, своей собственной версией.

Рекомендуемое значение NOO <= 3 методов.

3. *Количество операций, добавленных подклассом, NOA* (Number of Operations Added by a Subclass).

Для рекомендуемых значений CS=20 и DIT=6 рекомендуемое значение NOA<=4 методов (для класса-листа).

4. *Индекс специализации SI* (Specialization Index).

Обеспечивает грубую оценку степени специализации каждого подкласса. Специализация достигается добавлением, удалением или переопределением операций:

$$SI = (NOO \times Уровень)/M_{общ}$$

где Уровень - номер уровня в иерархии, на котором находится подкласс,

$M_{общ}$ - общее количество методов класса.

Рекомендуемое значение SI <=0,15.

5. *Средний размер операции* OS_{AVG} (Average Operation Size).

В качестве индикатора размера может использоваться количество строк программы, однако LOC-оценки приводят к известным проблемам. Альтернативный вариант — «количество сообщений, посланных операцией».

Рекомендуемое значение OS_{AVG}<=9.

6. *Сложность операции* OC_{AVG} (Average Operation Complexity).

Сложность операции может вычисляться с помощью стандартных метрик сложности, то есть с помощью LOC- или FP-оценок, метрики цикломатической сложности.

Рекомендуемое значение OC<=65 (для предложенного суммирования).

М. Лоренц и Д. Кидд предлагают вычислять OC суммированием оценок с весовыми коэффициентами, приведенными в табл. 4.1.:

Таблица 4.1 - Весовые коэффициенты для метрики OC

Параметр	Вес
Вызовы функций API	5,0
Присваивания	0,5
Арифметические операции	2,0
Сообщения с параметрами	3,0
Вложенные выражения	0,5
Параметры	0,3
Простые вызовы	7,0
Временные переменные	0,5
Сообщения без параметров	1,0

7. *Среднее количество параметров на операцию* NP_{AVG} (Average Number of Parameters per Operation).

Рекомендуемое значение NP_{AVG}=0,7.

8. *Количество описаний сценариев NSS* (Number of Scenario Scripts).

Рекомендуемое значение NSS — не менее одного сценария на публичный протокол подсистемы, отражающий основные функциональные требования к подсистеме.

9. *Количество ключевых классов NKC* (Number of Key Classes)

Ключевой класс прямо связан с коммерческой проблемной областью, для которой предназначена система. Маловероятно, что ключевой класс может появиться в результате повторного использования существующего класса. Поэтому значение NKC достоверно отражает предстоящий объем разработки. М. Лоренц и Д. Кидд предполагают, что в типовой ОО-системе на долю ключевых классов приходится 20-40% от общего количества классов. Рекомендуемое значение: если NKC<0,2 от общего количества классов системы, следует углубить исследование проблемной области (для обнаружения важнейших абстракций, которые нужно реализовать).

9. *Количество подсистем NSUB* (Number of SUBsystem). Количество подсистем обеспечивает понимание следующих вопросов: размещение ресурсов, планирование (с акцентом на параллельную разработку), общие затраты на интеграцию.

Рекомендуемое значение: NSUB > 3.

Контрольные вопросы.

1. Набор метрик Чидамбера и Кемерера.
2. Взвешенные методы на класс WMC: понятие, формула вычисления.
3. Высота дерева наследования DIT: понятие, пример вычисления.
4. Количество детей NOC: понятие, пример вычисления.
5. Сцепление между классами объектов CBO: понятие, пример вычисления.
6. Отклик для класса RFC: понятие, формула вычисления.
7. Связные и несвязные методы. Недостаток связности в методах LCOM.
8. Набор метрик Лоренца и Кидда.
9. Размер класса CS: понятие, вычисление.
10. Количество операций, переопределяемых подклассом, NOO: понятие, вычисление.
11. Количество операций, добавленных подклассом, NOA: понятие, вычисление.
12. Индекс специализации SI: понятие, вычисление.
13. Средний размер операции OS_{AVG}: понятие, вычисление.
14. Сложность операции OC_{AVG}: понятие, вычисление

Лабораторная работа №5
Характеристики качества программных средств

Цель работы: изучить характеристики и атрибуты качества программных средств. Использовать ГОСТ Р ИСО/МЭК 9126-93 для оценки качества программного средства.

Задание к работе:

1. Выбрать два любых программных продукта, соответствующих типу, указанному в варианте задания (см. табл. 5.1)
2. Оценить качество выбранных программных продуктов в соответствии с ГОСТ Р ИСО/МЭК 9126-93 (для оценки выбранного ПС могут использоваться не все указанные в стандарте показатели).
3. Сравнить результаты оценки первого и второго программного продукта.
4. Сделать выводы по результату проделанной работы.

Содержание отчета:

1. Титульный лист
2. Тема, цель, задание лабораторной работы.
3. Значения характеристик качества для программного средства №1.
4. Значения характеристик качества для программного средства №2.
5. Выводы по работе.

Варианты задания

№ вар.	*Тип программного средства*
1	Программные средства просмотра графических изображений
2	Программные средства просмотра видео-файлов
3	Программные средства проигрывания музыкальных файлов
4	Программные средства просмотра html-страниц (веб-браузеры)
5	Текстовые редакторы
6	Графические редакторы
7	Редакторы электронных таблиц
8	Программные средства для подготовки презентаций
9	Архиваторы
10	Программные средства для общения через Интернет

Основные теоретические сведения

Основой регламентирования показателей качества программных систем (ПС) является международный стандарт **ISO 9126:1991** (ГОСТ Р ИСО/МЭК 9126-93) *Информационная технология. Оценка программного продукта. Характеристики качества и руководство по ах применению.*

Развитие этого стандарта проводится в направлении уточнения, детализации и расширения содержания характеристик качества комплексов программ.

Все множество показателей качества ПС может быть классифицировано в виде иерархического дерева характеристик и субхарактеристик.

Самый высший уровень этой структуры состоит из характеристик качества, а самый нижний уровень — из их атрибутов.

Таблица 5.1. Атрибуты качества программных средств

Категорийно-описательные метрики	
Функциональные возможности	Функциональная пригодность
	Корректность – правильность
	Способность к взаимодействию
	Защищенность
Количественные метрики	
Надежность	Завершенность
	Устойчивость к дефектам
	Восстанавливаемость
	Доступность – готовность
Эффективность	Временная эффективность
	Используемость ресурсов
Качественные метрики	
Практичность	Понятность
	Простота использования
	Изучаемость
	Привлекательность
Сопровождаемость	Анализируемость
	Изменяемость
	Стабильность
	Тестируемость
Мобильность	Адаптируемость
	Простота установки
	Сосуществование – соответствие
	Замещаемость

Стандартизированные характеристики качества программных средств

Первая часть стандарта **ISO 9126-1** распределяет атрибуты качества ПС по шести характеристикам, которые используются в остальных частях стандарта. (см. табл. 5.1):

− категорийно-описательные (номинальные) метрики, которым наиболее адекватны функциональные возможности ПС;

− количественные метрики, применимые для измерения характеристик надежности и эффективности сложных комплексов программ;

− качественные метрики, в наибольшей степени соответствующие практичности, сопровождаемости и мобильности ПС.

Функциональные возможности — способность ПС обеспечивать решение задач, удовлетворяющих сформулированным потребностям заказчиков и пользователей при применении комплекса программ в заданных условиях.

Данная характеристика связана с тем, какие функции и задачи решает ПС для удовлетворения потребностей, в то время как другие характеристики главным образом связаны с тем, как и при каких условиях они могут выполняться.

Субхарактеристики и атрибуты функциональной возможности можно характеризовать в основном категориями и качественным описанием функций, для которых трудно определить количественные меры и шкалы. Поэтому они отнесены в отдельную группу номинальных, категорийно-описательных метрик.

Функциональная пригодность **—** это набор свойств и описаний субхарактеристики и ее атрибутов, определяющие назначение, номенклатуру, основные, необходимые и достаточные функции ПС, заданные техническим заданием и спецификациями требований заказчика или потенциального пользователя.

В процессе проектирования атрибуты функциональной пригодности должны конкретизироваться в спецификациях на компоненты и на ПС в целом.

Атрибутами этой субхарактеристики могут быть функциональная полнота решения заданного комплекса задач, степень покрытия функциональных требований спецификаций и их стабильность при развитии ПС.

Некоторые атрибуты можно представить численно: точностью результатов, относительным числом поэтапно изменяемых функций, числом реализуемых требований спецификаций заказчиков и т. д.

В наибольшей степени функциональная пригодность проявляется в ***корректности и надежности ПС.*** Критериями могут быть: способность компонентов к взаимодействию и степень стандартизации интерфейсов, мобильность программ и их защищенность от негативных внешних воздействий. В зависимости от назначения ПС почти каждый из показателей может стать в значительной степени доминирующим или почти полностью определяющим функциональную пригодность ПС.

Правильность — корректность — способность ПС обеспечивать правильные или приемлемые результаты и внешние эффекты для пользователя.

Эталонами для сравнения могут быть конструктивные требования — правила структурного построения модулей, компонентов и комплекса программ, организация их взаимодействия и интерфейсов, а также взаимоувязанные требования к функциям комплекса, компонентов модулей программ.

Эти требования должны быть прослежены сверху вниз до модулей и использоваться как эталоны при оценивании корректности соответствующих компонентов. Данное понятие включает получение ожидаемых данных с необходимой степенью точности расчетных значений в соответствии с требованиями технического задания и спецификаций.

В процессе проектирования модулей и групп программ применяются частные конструктивные критерии корректности, которые включают корректность структуры программ, обработки данных и межмодульных интерфейсов.

Корректность программных модулей включает функциональную и конструктивную корректность.

Конструктивная корректность модулей заключается в соответствии их структуры общим правилам структурного программирования и конкретным правилам оформления и внутреннего строения программных модулей в данном проекте.

Функциональная корректность модулей определяется корректностью обработки исходных данных и получения результатов, соответствующих требованиям спецификаций.

Конструктивная корректность обработки данных определяется правилами их структурирования и упорядочения в проекте. Эти правила могут быть достаточно полно формализованы без учета конкретных особенностей функций программ. Назначение и область применения программ определяют выбор используемых структур данных и конкретных дисциплин их упорядочения. Функциональная корректность обработки данных связана в основном с конкретизацией их содержания в процессе исполнения программ, а также при подготовке данных для внешних абонентов.

Корректность структуры комплексов программ определяется корректностью структур модулей и функциональных групп программ, построенных из модулей. Структурная корректность ПС должна поддерживать основную составляющую

корректности — функциональную. Степень соответствия программных компонентов и данных функциональным требованиям на каждом иерархическом уровне определяют интегральную корректность ПС.

Способность к взаимодействию — свойство ПС и их компонентов взаимодействовать с одной или большим числом указанных компонентов внутренней и внешней среды.

Способность программных и информационных компонентов к взаимодействию можно оценивать объемом технологических изменений в ПС, которые необходимо выполнить при дополнении или исключении некоторой функции, когда отсутствуют изменения операционной, аппаратной или пользовательской среды.

С этим показателем качества связаны корректность и унифицированность межмодульных интерфейсов, которые определяются двумя видами связей: по управлению и информации.

Связи по управлению составляют вызовы программных модулей и возвраты в вызывавшие модули. Взаимодействие модулей по информации может происходить через обменные переменные, непосредственно подготавливаемые и используемые соседними модулями, или через глобальные переменные между более крупными компонентами.

Защищенность — свойство компонентов ПС защищать программы и информацию от любых негативных воздействий.

В критериях защиты и обеспечения безопасности для конкретного ПС сосредоточиваются разнообразные характеристики, которые в ряде случаев трудно или невозможно описать количественно и приходится оценивать экспертно или по балльной системе.

Основное внимание в практике обеспечения безопасности применения информационных систем (ИС) сосредоточено на защите от злоумышленных разрушений, искажений и хищений программных средств и информации баз данных. При этом подразумевается наличие лиц, заинтересованных в доступе к конфиденциальной, или полезной информации с целью ее незаконного использования, искажения или уничтожения. Основой такой защиты является аудит доступа и контроль организации ограничений доступа. В реальных сложных системах возможны катастрофические последствия и аномалии функционирования, отражающиеся на безопасности применения, при которых их источниками являются случайные, непредсказуемые, дестабилизирующие факторы и отсутствуют непосредственно заинтересованные в подобных нарушениях лица.

Наиболее полно качество защиты ИС характеризуется величиной предотвращенного ущерба, а также средним временем между возможными проявлениями угроз, нарушающих безопасность. Однако формализовано описать и измерить возможный ущерб при нарушении безопасности для критических ПС разных классов практически невозможно. Поэтому факты реализации угроз целесообразно характеризовать интервалами времени между их проявлениями или наработкой на отказы, отражающимися на безопасности. Это сближает понятия и характеристики степени безопасности с показателями надежности ПС. Принципиальное различие состоит в том, что в показателях надежности учитываются все реализации отказов, а в характеристиках защиты следует регистрировать только те отказы, которые отразились на безопасности функционирования. Достаточно универсальным измеряемым параметром при этом остается длительность восстановления нормальной работоспособности ПС и информационной системы. Приближенно такие катастрофические отказы в восстанавливаемых ПС можно выделять по превышению некоторой допустимой длительности восстановления работоспособности. В требованиях к качеству программ,

обеспечивающих защиту, следует отражать все аспекты, необходимые для удовлетворения согласованных потребностей заказчика по общей безопасности ИС.

Надежность — свойства комплекса программ, которые обеспечивают достаточно низкую вероятность отказа в процессе функционирования ПС в реальном времени. При применении понятий надежности к программным средствам следует учитывать следующие особенности и отличия этих объектов от традиционных технических систем, для которых первоначально разрабатывалась теория надежности:

− при разработке и оценке качества отдельных программных компонентов к ним не применимы понятия надежности функционирования, если при обработке информации они не используют значения реального времени и не взаимодействуют динамически с внешней средой;

− относительно редкое разрушение программных компонентов и необходимость их физической замены приводит к принципиальному изменению понятий сбоя и отказа программ и к разделению их по длительности восстановления относительно некоторого времени неработоспособного простоя, допустимого для функционирования информационной системы;

− для повышения надежности комплексов программ особое значение имеют методы автоматического сокращения длительности восстановления и преобразования отказов в кратковременные сбои путем введения в программные средства временной, программной и информационной избыточности для реализации рестарта.

Основным ***принципам классификации сбоев и отказов*** в программах при отсутствии их физического разрушения является разделение по временному показателю длительности восстановления после любого искажения программ, данных или вычислительного процесса, регистрируемого как нарушение работоспособности. Классификация программных сбоев и отказов по длительности восстановления приводит к необходимости анализа и учета динамических характеристик абонентов, являющихся источниками и потребителями данных, обработанных исследуемым ПС, а также временных характеристик функционирования программ. Пороговое время восстановления работоспособного состояния системы, при превышении которого следует фиксировать отказ, близко к периоду решения задач и подготовки информации соответствующему абоненту.

Надежность функционирования ПС наиболее широко характеризуется устойчивостью или способностью к безотказному функционированию и восстанавливаемостью работоспособного состояния после произошедших сбоев или отказов. В свою очередь, устойчивость зависит от уровня не устранённых дефектов и ошибок (завершенности) и способности ПС реагировать на их проявления так, чтобы это не отражалось на показателях надежности. Некорректная программа может функционировать совершенно надежно, следовательно, надежность функционирования программ является понятием динамическим и существенно отличается от понятия статической корректности программ.

Завершенность — свойство ПС не попадать в состояния отказов вследствие ошибок и дефектов в программах и данных. Завершенность можно характеризовать наработкой (длительностью) на отказ при отсутствии автоматического восстановления — рестарта, измеряемой обычно часами. На эту субхарактеристику надежности влияют только отказы вследствие проявившихся дефектов. Они могут быть обусловлены неполным тестовым покрытием при испытаниях компонентов и ПС в целом, а также недостаточной завершенностью их тестирования.

Устойчивость к дефектам и ошибкам — свойство ПС автоматически поддерживать заданный уровень качества функционирования в случаях проявления дефектов и ошибок или нарушения установленного интерфейса. Для этого в ПС должна вводиться временная, программная и информационная избыточность, реализующая оперативное обнаружение дефектов и ошибок функционирования, их идентификацию и автоматическое

восстановление (рестарт) нормального функционирования ПС. Эффективное оперативное устранение проявления дефектов, ошибок и некорректного взаимодействия с операционной и внешней средой определяет субхарактеристику — устойчивость комплексов программ.

Восстанавливаемость — свойство ПС в случае отказа возобновлять заданный уровень качества функционирования, а также поврежденные программы и данные. После отказа ПС иногда бывает неработоспособно в течение некоторого периода времени, продолжительность которого определяется его восстанавливаемостью. Для прерывания неработоспособного состояния, диагностики причин отказа, а также реализации процессов восстановления необходимы вычислительные ресурсы и время. Основными показателями процесса восстановления являются длительность и ее вероятностные характеристики. Восстанавливаемость характеризуется также полнотой восстановления нормального функционирования программ в процессе ручного или автоматического их перезапуска — рестарта. Перезапуск должен обеспечивать возобновление нормального функционирования ПС, на что требуются ресурсы ЭВМ и время. Поэтому полнота и длительность восстановления функционирования после сбоев и отказов определяют надежность ПС и возможность его использования по прямому назначению.

Доступность или готовность — свойство ПС быть в состоянии выполнять требуемую функцию в данный момент времени при заданных условиях использования. Внешне доступность может оцениваться относительным временем, в течение которого ПС находится в работоспособном состоянии, в пропорции к общему времени применения. Следовательно, доступность — комбинация завершенности (от которой зависит частота отказов), устойчивости к ошибкам и восстанавливаемости, которые в совокупности обусловливают длительность простоя после каждого отказа, а также длительность наработки на отказ. Для определения доступности измеряется время работоспособного состояния системы между последовательными отказами или началами нормального функционирования системы после них. Характеристики отказов и восстановления обобщаются в критерии ***коэффициент готовности.*** Этот показатель отражает вероятность наличия восстанавливаемой системы в работоспособном состоянии в произвольный момент времени.

Эффективность — свойства ПС, обеспечивающие требуемую производительность решения функциональных задач с учетом количества используемых вычислительных ресурсов в установленных условиях.

Временная эффективность — свойства ПС, обеспечивающие требуемые времена отклика и обработки заданий, а также пропускную способность при выполнении его функций в заданных условиях. Временная эффективность ПС определяется длительностью выполнения заданных функций и ожидания результатов в средних и наихудших случаях. Она зависит от скорости обработки данных, влияющей непосредственно на интервал времени завершения конкретного вычислительного процесса, и от пропускной способности — производительности, т. е. от числа заданий, которое можно реализовать на данной ЭВМ в заданном интервале времени. Эти показатели качества тесно связаны с временем реакции (отклика) ПС на запросы при решении основных, функциональных задач. Величина этого времени зависит от длительности решения задачи центральным процессором ЭВМ, от затрат времени на обмен с внешней памятью, на ввод и вывод данных и от длительности ожидания в очереди до начала решения задачи.

Используемость ресурсов — степень использования доступных вычислительных ресурсов в течение заданного времени при выполнении функций ПС в установленных условиях. Ресурсная экономичность отражает полноту занятости ресурсов центрального

процессора, оперативной, внешней и виртуальной памяти, каналов ввода-вывода, терминалов и каналов сетей связи. Этот критерий определяется структурой и функциями ПС, а также архитектурными особенностями и доступными ресурсами ЭВМ. В зависимости от конкретных особенностей ПС и ЭВМ при выборе критериев качества может доминировать либо величина абсолютной занятости ресурсов различных видов, либо относительная величина использования ресурсов каждого вида при нормальном функционировании ПС. Ресурсная экономия влияет не только на стоимость решения функциональных задач, но часто, особенно для встраиваемых ЭВМ, определяет принципиальную возможность полноценного функционирования конкретного ПС в условиях реально ограниченных вычислительных ресурсов.

Практичность — свойства ПС, обусловливающие сложность его понимания, изучения и использования, а также привлекательность для квалифицированных пользователей при применении в указанных условиях. В число пользователей могут быть включены операторы, конечные пользователи и косвенные пользователи, которые находятся под влиянием или зависят от качества функционирования ПС. В практичности следует учитывать все разнообразие характеристик внешней среды пользователей, на которые может влиять ПС, включая возможную подготовку к использованию и оценку результатов функционирования программ. Применимость (практичность) использования ПС — понятие достаточно абстрактное и трудно формализуемое, однако часто определяющее функциональную пригодность и полезность применения ПС. В эту группу показателей входят критерии, с различных сторон отражающие функциональную понятность, удобство освоения, техническую эффективность или простоту использования. Некоторые из субхарактеристик можно оценивать технико- экономическими показателями — затратами труда и времени на реализацию некоторых функций.

Понятность — свойства ПС, обеспечивающие пользователю понимание того, является ли ПС пригодным для его целей, и того, как ПС можно использовать для конкретных задач и условий использования. Понятность зависит от качества документации и первичных впечатлений от характеристик ПС. Понятность ПС можно описать четкостью функциональной концепции, широтой демонстрационных возможностей, полнотой, комплектностью и наглядностью представления в документации возможных функций.

Простота использования — свойства ПС, обеспечивающие пользователю возможность удобной и комфортной его эксплуатации и управления им. Изменяемость, адаптируемость и легкость инсталляции могут быть предпосылками для простоты использования. Она соответствует управляемости, устойчивости к ошибкам и согласованности с ожиданиями и навыками пользователя. Этот критерий учитывает физические и психологические характеристики пользователей и отражает уровень контролируемости и комфортности условий эксплуатации ПС, возможность предотвращения ошибок пользователей. Кроме того, удобство использования характеризуется рядом динамических параметров: временем ввода и отклика на задание, длительностью решения типовых задач, временем на регистрацию результатов.

Изучаемость — свойства ПС, обеспечивающие удобное изучение его применения достаточно квалифицированными пользователями. Может определяться трудоемкостью и длительностью подготовки пользователя к полноценной эксплуатации ПС. Эти атрибуты зависят от возможности предварительного обучения и совершенствования знаний в процессе эксплуатации, от возможностей оперативной помощи и подсказки (Help) при использовании ПС, а также от полноты, доступности и

удобства использования руководств и инструкций по эксплуатации. Изучаемость может также; характеризоваться объемом эксплуатационной документации и/или объемом и качеством электронных учебников.

Привлекательность — субъективное свойство ПС нравиться заказчикам, покупателям и/или пользователям. Оно связано с внешними атрибутами и эстетикой оформления ПС, интерфейсов с пользователями и эксплуатационной документации, обусловливающими большую или меньшую его привлекательность для потребителей.

Сопровождаемость — приспособленность ПС к модификации и изменению конфигурации и функций. Модификации могут включать исправления, усовершенствования или адаптацию ПС к изменениям во внешней среде применения, а также в требованиях и функциональных спецификациях заказчика. Простота и трудоемкость модификаций определяются внутренними характеристиками качества комплекса программ, которые отражаются на внешнем качестве и на качестве в использовании, а также на сложности управления конфигурацией и модификацией ПС.

Анализируемость (удобство для анализа) — пригодность ПС к диагностике его дефектов или причин отказов, а также к идентификации и выделению его компонентов для модификации. Эта субхарактеристика зависит от стройности архитектуры, унифицированности интерфейсов, полноты и корректности технологической и эксплуатационной документации на ПС.

Изменяемость — приспособленность ПС к простой реализации специфицированных изменений и к управлению конфигурацией. Реализация модификаций включает кодирование, проектирование и документирование изменений. Для этого требуется определенная трудоемкость и время, связанные с исправлением дефектов и/или модернизацией функций, а также с изменением процессов эксплуатации. В оценках этой субхарактеристики учитываются влияние структуры, интерфейсов и технических особенностей ПС и не рассматриваются воздействия фундаментальных, принципиальных изменений его функций. Изменяемость зависит не только от внутренних свойств ПС, но также от организации и инструментальной оснащенности процессов сопровождения и конфигурационного управления, на которые ориентированы архитектура, внешние и внутренние интерфейсы комплекса программ.

Стабильность — способность ПС предотвращать и минимизировать непредвиденные негативные эффекты от его изменений, возможность локализовать и ограничивать область изменения программ и данных. Эта внутренняя субхарактеристика определяется архитектурой ПС, корректностью технологической документации и может существенно влиять на функциональную пригодность, надежность и адекватность поведения комплекса программ при изменениях и использовании.

Тестируемость — свойство ПС, обеспечивающее простоту проверки качества изменений и приемки модифицированных компонентов программ. Эта субхарактеристика зависит от величины области влияния изменений, которые необходимо тестировать при модификациях программ и данных, от сложности тестов для проверки характеристик процессов рестарта. Ее атрибуты зависят от четкости правил структурного построения компонентов и всего комплекса программ, от унификации межмодульных и внешних интерфейсов, от полноты и корректности технологической документации. Возможность локализации изменений и унификация интерфейсов компонентов с некорректируемой

частью ПС позволяет сокращать сложность, трудоемкость и длительность их тестирования, упрощает подготовку тестов и анализ результатов.

Мобильность — подготовленность ПС к переносу из одной аппаратно-операционной среды в другую. Переносимость программ и данных на различные аппаратные и операционные платформы является важным показателем функциональной пригодности для многих современных ПС. Это свойство может оцениваться объемом, трудоемкостью и длительностью необходимых доработок компонентов ПС и операций по адаптации, которые следует выполнить для обеспечения полноценного функционирования ПС после переноса на иную платформу. Мобильность может осуществляться на уровне исходных текстов программ или на уровне объектного кода. Она зависит от структурированности и расширяемости комплексов программ и данных, а также от дополнительных ресурсов, необходимых для реализации переносимости и модификации компонентов при их переносе.

Адаптируемость — способность программ и информации баз данных к модификации для эксплуатации в различных аппаратных и операционных средах без применения других действий или средств, нежели те, что предназначены для этой цели в рассматриваемом ПС. Она зависит от свойств и структуры аппаратной и операционной среды, от методов и средств подготовки ПС к переносу на новые платформы. Адаптируемость включает масштабируемость внутренних возможностей (например, экранных полей, размеров таблиц, объемов транзакций, форматов отчетов и т. д.). Если ПС должно адаптироваться конечным пользователем, адаптируемость соответствует пригодности для индивидуализации и может быть компонентом для простоты использования.

Простота установки — способность ПС к простому внедрению (инсталляции) в новой аппаратной и операционной среде заказчика или пользователя. Если ПС должно устанавливаться конечным пользователем, легкость установки будет предпосылкой для удобства использования. Также, как и адаптируемость, она может измеряться трудоемкостью и длительностью процедур установки, а также степенью удовлетворения требований заказчика и пользователей к характеристикам инсталляции.

Сосуществование (соответствие) — способность ПС сосуществовать и взаимодействовать с другими независимыми программами в общей вычислительной среде, разделяя общие ресурсы. Эта субхарактеристика зависит от степени стандартизации интерфейсов ПС с операционной и аппаратной средой применения, от совместимости функций и данных и может оцениваться экспертно.

Замещаемость — приспособленность каждого компонента ПС к относительно простому использованию вместо другого заменяемого компонента. Замещаемость не предполагает, что заменяемый компонент ПС способен полностью выполнять функции предшествующего. Она может включать атрибуты, как простоты установки, так и адаптируемости. Большую роль для этого свойства играют четкая структурированность архитектуры и стандартизация внутренних и внешних интерфейсов ПС. Это свойство отражается на трудоемкости и длительности замены в основном крупных компонентов ПС. Данное понятие было представлено в стандарте как отдельная субхарактеристика вследствие ее важности.

Библиографический список

1. Крылова Г.Д. Основы стандартизации, сертификации, метрологии. – М.: ЮНИТИ-ДАНА, 2000. – 711 с.
2. Липаев В.В. Качество программного обеспечения. – М.: Финансы и статистика, 2002. – 263 с.
3. Липаев В.В. Отладка сложных программ: Методы средства, технология. М.: Энергоатомиздат, 1993. – 384 с.
4. Никитин В.А., Филончева В.В. Управление качеством на базе стандартов ИСО 9000:2000. – СПб.: Питер, 2004. – 127 с.
5. Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем. – СПб.: Питер, 2002. – 464 с.
6. Ермоленко, Д. Н. Метрология, стандартизация и сертификация программного обеспечения: учеб. пособие для студентов специальности 230105. Белгород: Изд-во БГТУ им. В. Г. Шухова, 2008. — 82 с.
7. Стандарт ISO 9126:1991 (ГОСТ Р ИСО/МЭК 9126-93) Информационная технология. Оценка программного продукта. Характеристики качества и руководство по ах применению.

Учебное издание

МЕТРОЛОГИЯ, СТАНДАРТИЗАЦИЯ И СЕРТИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания к выполнению лабораторных работ для студентов направления 09.03.04 — Программная инженерия

Составитель: **Бондаренко** Татьяна Владимировна

Подписано в печать Формат 60×84/16. Усл.печ.л.. Уч.-изд.л.
Тираж экз. Заказ Цена
Отпечатано в Белгородском государственном технологическом университете им. В. Г. Шухова
308012, г. Белгород, ул. Костюкова, 46