Информационное общество и проблемы прикладной информатики

Задание на расчётно-графическую работу по дисциплине

Составил: М.А. Бакаев, к.т.н., доцент, АВТФ НГТУ

Цель выполнения РГР: освоить на практике осуществление программометрических расчётов: определение размера словаря и основных метрических характеристик программных продуктов.

Расчётно-графическая работа соответствует экспресс-обследованию, проводимому перед началом проектов по созданию программного обеспечения. Подробная информация по математическому базису и методологической основе содержится в [1].

1. Проектирование структур данных и расчет словаря программного продукта

1.1. Теоретическая справка

Основным показателем, характеризующим сложность программного обеспечения (ПО) является объем (длина текста) его кода. М. Холстед в 1970-х годах показал, что количество слов в тексте программ (N) с высокой степенью точности связано с размером словаря этого текста (n), т.е. количеством **различных** слов в тексте:

$$N \approx n \log_2 n \tag{1}$$

Позднее были предложены методики [1], позволяющие оценить размер полного словаря ПО на основе «сжатого» словаря входных[1] и выходных данных (n^*):

$$n \approx n^* \log_2 n^* \tag{2}$$

Важным преимуществом методики, описанной в [1], является то, что она позволяет оценить ожидаемые характеристики будущего программного продукта – прежде всего, объем его кода – ещё на предварительном этапе проекта (оценка целесообразности разработки). Кроме этого, может быть также рассчитана оптимальная структура ПО: количество и иерархия программных модулей, что может быть полезно в т.ч. для декомпозиции работ по разработчикам:

$$k = \frac{n^*}{\log_2 2n^*} \tag{3}$$

или

$$k_{\text{иер}} = \frac{n^*}{\log_2 n^*} \tag{4}$$

где k – количество модулей без использования иерархической структуры (обычно для небольших программных продуктов с $n^*<64$), $k_{\text{иер}}$ – количество модулей при использовании иерархической структуры.

Для всего этого должны быть предварительно спроектированы структуры данных, которые будут использоваться в работе программного продукта. Они определяются спецификой предметной области, а также типом программного продукта, например:

- Для **информационных систем**, основной функцией которых является преобразование документов, сжатый словарь это в основном названия и реквизиты входных и выходных документов.

[1] К входным данным следует относить не только непосредственно вводимые пользователем значения, но также и нормативно-справочную (условно-постоянную информацию), имена и параметры внешних библиотек/функций, имена и свойства сущностей из внешних баз данных и знаний (в т.ч. согласно форматам взаимодействия с ними по API) и т.п.

- Для **библиотек** (компонентов, предназначенных для повторного использования), которые часто разрабатываются с применением объектно-ориентированного подхода, сжатый словарь – это имена глобальных переменных + имена классов + имена переменных в классах + имена методов. При этом следует учитывать наследование в структуре классов, которое позволяет уменьшить словарь.
- Для различных **автоматизированных рабочих мест** (АРМ) следует производить декомпозицию по функциям (задачам) и вычислять словарь по каждой из таких подсистем отдельно.
- Для **трансляторов и компиляторов** (по оценке Ф. Брукса, они в среднем в 3 раза сложнее обычных прикладных программных продуктов), словарь определяется синтаксисом входного и выходного языка.
- Для операционных систем (по оценке Ф. Брукса, они в среднем в 3 раза сложнее, чем трансляторы и компиляторы), словарь определяется разрядностью регистра состояний.

Для перехода от структур данных к размеру словаря следует руководствоваться следующими принципами:

- +1 элемент словаря для переменной или константы простого типа данных: целочисленный, вещественный, символьный, строковый, логический, …
- +1 по значениям переменной, если они сравниваются с константами и обрабатываются особо
- +3 элемента словаря для массива (+1 имя массива, +1 тип данных, +1 количество элементов массива или +1 начальный адрес памяти, +1 конечный адрес памяти, +1 байтов на элемент)
- +6 элементов словаря для двумерного массива или однородно обрабатываемой таблицы (массив массивов)
- Для типа «запись» (сущности, класса, таблицы реляционной БД) – по количеству и типам данных (не забыть +1 элемент словаря на название самого объекта)

1.2. Ход работы

В рамках первой части РГР студентам необходимо выполнить следующие задачи:

1. Определить примерный функционал программного продукта, создание которого планируется в рамках магистерской диссертации. Представить описание функционала (при необходимости, по подсистемам) в отчете по РГР, указав в частности назначение и пользователя ПО. Пример сопутствующей диаграммы представлен на Рис. 1.
   Объем: не менее 2500 знаков[2].

Рис. 1. Пример информационной модели автоматизированной системы оценки деятельности (фрагмент) [2].

2. Определить входные, выходные и нормативно-справочные (условно-постоянные) данные для ПО (при необходимости, по подсистемам). Спроектировать структуры данных (простые переменные и константы, массивы, записи/таблицы и др.) и отразить их в отчёте по РГР (см. примеры на Рис. 2, Рис. 3). Описать внешние сущности, к которым будет обращаться программный продукт (готовые решения, базы данных и т.п.) и интерфейсы взаимодействия с ними. Подсчитать размер словаря для каждой структуры данных и общий «сжатый» размер словаря создаваемого ПО (n^*).
   Объем: **не менее 50** элементов словаря (без учёта справочников и внешних сущностей).

[2] Здесь и далее: 1 рисунок или диаграмму считать эквивалентными 500 знакам.

Наименование потока	Наименование реквизитов	Тип данных	Количество
1	2	3	4
Выходные потоки			
Результаты оценки деятельности сотрудников	Код сотрудника	int	21
	Код отдела	int	
	Код проекта	Int	
	Дата начала деятельности	datetime	
	Дата окончания деятельности	datetime	
	Код цели сотрудника	int	
	Код показателя	int	
	Фактическое значение	int	
	Плановое значение	int	
	Веса	Array (int)	
	Оценки	Array (int)	
	Процент выполнения	float	
	Итог по показателю	float	
	Баллы	float	
	Размер премий	money	
	Рейтинг	int	

...

Нормативно-справочная информация			
Справочник отделов	Код отдела	int	4
	Название отдела	varchar	
	Код организации	int	
Справочник сотрудников	Код сотрудника	int	11
	ФИО	varchar	
	Логин в ИС	varchar	
	Пароль в ИС	varchar	
	Адреса электронной почты	Array (varchar)	
	Код роли в проекте	int	
	Код должности в отделе	int	
	Оклад	money	
Справочник целей	Код цели	int	5
	Тип цели	Array (varchar)	
	Описание цели	text	

Рис. 2. Пример расчета словаря для автоматизированной системы оценки деятельности (фрагмент) [2].

Наименование справочников и документов	**Наименование переменных**	**Описание переменной, тип**	**Количество слов**
Автомобили аэропорта	**№машины***	Integer	1
	Марка машины	String	1
	Цвет	String	1
Автомобильная стоянка (документ)	**№ документа***	Integer	1
	Дата документа	Data	1
	Срок стоянки	Integer	1
	Стоимость стоянки	Integer	1
	№ места	Integer	1
	Просрочка	Data	1
	Штраф	Integer	1
	Код клиента	Integer	1
	№ стоянки	Integer	1
	Код аэропорта	Integer	1

Рис. 3. Пример расчета словаря для информационной системы аэропорта (фрагмент) [3].

2. Расчет программометрических характеристик и экономических показателей проекта

2.1. Алгоритм программометрического расчета

Алгоритм расчета программометрических характеристик на основе размера «сжатого» словаря (n^*), определенного в предыдущей части РГР, может быть представлен в следующем виде (подробнее см. [1, с. 115-119])[3].

1. Определение количества (k) и структуры модулей на основе формулы (3) либо (4).
   - в случае использования (4) следует дополнительно определить оптимальное количество иерархических уровней в структуре модулей[4]:

$$i_{опт} = \left[\frac{\log n^*}{\log 8} \right] + 1 \tag{5}$$

2. Определение «сжатого» (n^*_M) и полного (n_M) словаря модуля:

$$n^*_M = \left[n^* / k \right] \approx 6..8 \tag{6}$$

$$n_M = \left[n^*_M \log n^*_M \right] \tag{7}$$

3. Определение мат. ожидания длины программного кода (на ассемблере) модуля (N_M) и программы в целом (N_{AC}), а также точности оценки N_{AC} (относительного отклонения от мат. ожидания):

$$N_M = \left[2 n_M \log n_M \right] \tag{8}$$

$$N_{\text{AC}} = \left[k N_M + 2k \log k \right] \tag{9}$$

$$\delta \approx \pm \frac{1}{\sqrt{6k}} \tag{10}$$

4. Определение объема программы на ассемблере (V_{AC}) и на алгоритмическом языке ($V_{АЯ}$)[5]:

$$V_{\text{AC}} = \left[N_{\text{AC}} \log n_M \right] \tag{11}$$

$$V_{\text{АЯ}} = \left[\frac{\lambda_{AC}}{\lambda_{АЯ}} V_{AC} \right] = \left[\frac{0{,}88}{1{,}5} V_{AC} \right] \tag{12}$$

5. Определение эквивалентного количества машинных команд на ассемблере (P_{AC}) и на алгоритмическом языке ($P_{АЯ}$):

$$P_{\text{AC}} \approx \left[0{,}5 N_{\text{AC}} \right] \tag{13}$$

$$P_{\text{АЯ}} \approx \left[\frac{\lambda_{AC}}{\lambda_{АЯ}} P_{\text{AC}} \right] \tag{14}$$

---

[3] Во всех формулах используются логарифмы по основанию 2, если не указано иное.
[4] Квадратные скобки означают округление.
[5] Уровень алгоритмического языка высокого уровня принять за λ=1,5, если не обосновано другое значение.

6. Определение начального количества ошибок перед комплексной отладкой в программе, написанной на алгоритмическом языке (B_0):

$$B_0 = \left[\frac{\alpha V_{АЯ}}{E_{rate}} \right] \quad (15)$$

где коэффициент сложности программного кода (α):

$$\alpha = \frac{\log^3 \frac{n^*}{7}}{250} \quad (16)$$

а E_{rate} – статистический коэффициент, показывающий на сколько бит программного кода программист делает одну ошибку[6].

2.2. Определение трудоемкости и стоимости разработки

Трудоемкость кодирования и отладки может быть оценена на основе статистики производительности труда программистов в зависимости от сложности проектов, приведённой в [1]. Так, количество **отлаженных** команд в день для одного **квалифицированного** программиста (C_D) составляет:

- Для операционных систем: 1,4…5
- Для ПО средней сложности (трансляторы и т.п.): 9
- Для простых прикладных программ: 30
- В среднем (за примерно 20 лет): 12

Таким образом, трудоемкость кодирования и отладки программного продукта ($A_{АЯ}$, в человеко-днях) может быть оценена следующим образом:

$$A_{АЯ} = \frac{P_{АЯ}}{C_D} \quad (17)$$

Для небольшого количества разработчиков в команде (M), т.е. без учета времени на коммуникацию, срок кодирования и отладки ($T_{АЯ}$, в рабочих днях) может быть оценен как:

$$T_{АЯ} = \frac{A_{АЯ}}{M} \quad (18)$$

Учитывая, что, согласно статистике, накопленной в сфере управления ИТ-проектами, на стадии кодирования и отладки приходится около 50% трудоемкости[7], общую трудоемкость реализации проекта ($A_К$, в человеко-днях) можно оценить как:

$$A_К = 2A_{АЯ} \quad (19)$$

Предполагая, что на каждой стадии проекта количество профильных ИТ-специалистов остаётся в среднем равным M, можно оценить итоговый срок разработки ($T_К$, в рабочих днях):

$$T_К = 2T_{АЯ} \quad (20)$$

---

[6] Для высококвалифицированных программистов составляет около 3000 бит, для низкоквалифицированных принять значение равным 1000 бит.

[7] На стадии анализа требований и проектирования приходится остальные 50% трудоемкости, внедрение и сопровождение не учитываются.

После этого, выбрав долю трудоемкости A_{test}, отводимую на тестирование и отладку (в крупных проектах эта доля достигает 40% от $A_к$, однако в небольших, как правило, меньше) можно оценить надежность программного продукта через время наработки на отказ ($t_Н$), т.е. между двумя последовательными проявлениями неизбежно оставшихся в нём ошибок:

$$t_{\text{Н}} = \frac{A_{\text{test}}}{\ln(\text{B}_0 + 1)} \tag{21}$$

Альтернативно, общая трудоемкость ($A_{ММ}$, в человеко-месяцах) и срок разработки ($T_М$, в месяцах) программного продукта могут быть определены с использованием методики COCOMO I (базовый уровень):

$$A_{ММ} = a * (SIZE)^b \tag{22}$$

$$T_M = c * (PM)^d \tag{23}$$

где SIZE – объем программного продукта в **тысячах** строк кода (LOC). Для алгоритмических языков высокого уровня количество логических строк кода может быть приравнено к количеству команд ($P_{АЯ}$).

Значения коэффициентов COCOMO I (базовый уровень) даны в Табл. 1.

Таблица 1. Значения коэффициентов COCOMO в зависимости от типа проекта.

Тип проекта	***a***	***b***	***c***	***d***
Распространенный	2,4	1,05	2,5	0,38
Полунезависимый	3,0	1,12	2,5	0,35
Встроенный	3,6	1,20	2,5	0,32

Типы проектов определяются в COCOMO I на основе экспертной оценки:

- Распространенный – маленькие команды с хорошим опытом работы и не жесткими требованиями к разработке.
- Полунезависимый – средние по размеру команды со смешанным опытом разработки и со смешанными требованиями.
- Встроенный – с учетом множества жестких ограничений (по аппаратному, программному, операционному обеспечению и т.д.)

2.3. Ход работы

В рамках второй части РГР (**после согласования с преподавателем** успешного выполнения первой части) студентам необходимо выполнить следующие задачи:

3. Рассчитать основные программометрические характеристики будущего ПО, включая количество команд (логических строк кода) и начальное количество ошибок. Отразить ход расчёта в отчёте по РГР.
4. Задав экспертным путем необходимые значения, оценить трудоемкость и сроки разработки, используя:
    a. статистическую методику Г.И. Кайгородцева [1] (дополнительно определить надежность ПО),
    b. методику COCOMO I.

Сравнить полученные двумя способами значения, **сделать выводы**.

Дать итоговую оценку трудоемкости и сроков разработки.

5. Исходя из итоговой оценки трудоемкости, определить итоговый бюджет проекта на основе текущей средней заработной платы программистов соответствующей квалификации (с начислениями на фонд оплаты труда и налогами) и накладных расходов[8].

[8] Принять как +20% от прямых производственных расходов, если не обосновано другое значение.

Рекомендуемая литература

1. Г.И. Кайгородцев. Введение в курс метрической теории и метрологии программ. // Учебник. – Новосибирск, издательство НГТУ, 2011. – 192 с.
2. Е. Мезенцева (2014). РГР по ИОиППИ. НГТУ. – 13 с.
3. Д. Пушкарь (2013). РГР по ИОиППИ. НГТУ. – 7 с.