Основы создания качественных программных систем

(продолжение)

Таблица 7.1 – Международные стандарты.

Стандарт	Описание
ISO 09126:1991	Оценка программного продукта. Характеристики качества и руководство по их применению
DOD-STD-2168	Программа обеспечения качества оборонных программных средств
ISO 09000-:1991	Общее руководство качеством и стандарты по обеспечению качества. Ч.3: Руководящие указания по применению ISO 09001 при разработке, поставке и обслуживанию программного обеспечения
ISO 12207:1995	Процессы жизненного цикла программных средств
DOD-STD 2167A:1988	Разработка программных средств для систем военного назначения
ISO 09646-1-6:1991.ИТ.ВОС	Методология и основы аттестационного тестирования ВОС
ANSI/IEEE 829-83	Документация при тестировании программ
ANSI/IEEE 1008-86	Тестирование программных модулей и компонент программных средств
ANSI/IEEE 1012-86	Планирование проверки (оценки) (verification) и подтверждения достоверности (validation) программных средств

Качество ПО

Оценка качества ПО в соответствии со стандартом.

Методика оценки качества программного обеспечения основывается на требованиях стандарта ГОСТ 28195-89.

Процесс оценки качества программного обеспечения производится для каждой стадии жизненного цикла и включает:

- выбор показателей качества;
- определение значений показателей;
- сравнение полученных значений с базовыми значениями показателей качества.

Качество ПО

В соответствии с ГОСТ 28195-89 период жизненного цикла программного обеспечения подразделяется на временные промежутки (фазы):

1 Анализ – этап определения требований к программному обеспечению, спецификация требований и формирования технического задания на проектирование программы.

2 Проектирование – этап разработки технического проекта.

3 Реализация – этап разработки программного обеспечения, средств тестирования и документации.

4 Тестирование – этап испытания программы и устранения недостатков.

Качество ПО

5 Изготовление – этап преобразования программного обеспечения в форму, готовую для поставки, завершение формирования документации.

6 Внедрение – этап подтверждения стабильной работы программного обеспечения и ввод в стадию использования.

7 Эксплуатация – этап применения программного обеспечения по назначению.

8 Сопровождение – этап устранения дефектов в процессе эксплуатации, усовершенствование и модификация программного обеспечения.

Качество ПО

Оценка качества программного обеспечения на стадиях жизненного цикла осуществляется на основе четырёхуровневой системы показателей.

Показатели первого уровня (факторы качества) характеризуют потребительски-ориентированные свойства программных средств, которые соответствуют потребностям пользователей. Факторы качества определяют значимые свойства программы. Фактор представляет собой интегральную оценку, которой соответствует несколько критериев качества (комплексных показателей второго уровня).

Качество ПО

Стандарт ISO 9126

«Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению» является международным стандартом, определяющим оценочные характеристики качества ПО.

Стандарт ISO 9126 состоит из четырех частей, в которых излагаются следующие категории:

- модель качества;
- внешние метрики;
- внутренние метрики;
- метрики качества в использовании.

Качество ПО

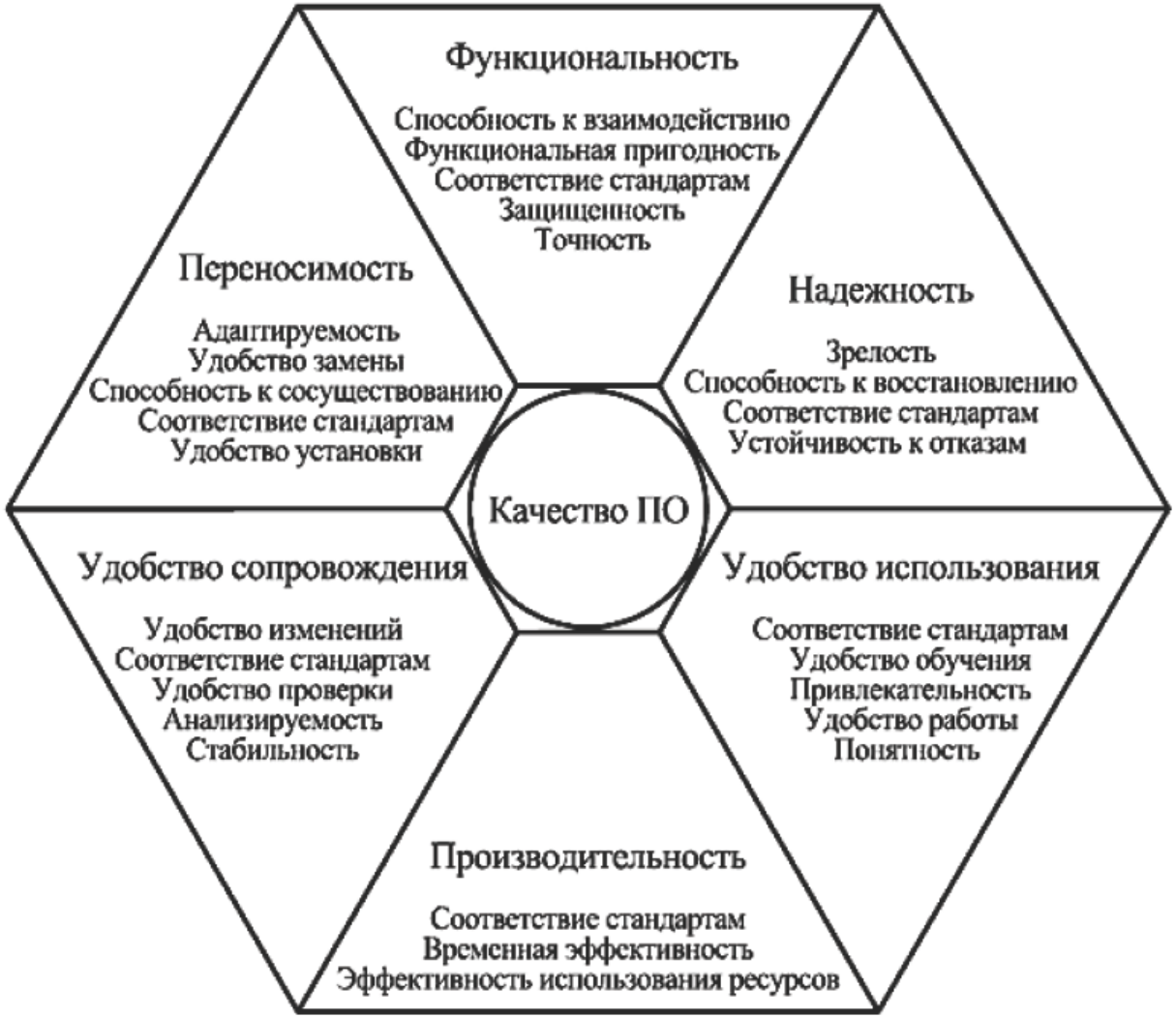

Рис. 7.3 - Показатели качества по ISO 9126

Качество ПО

Модель качества классифицирует качество ПО с шестью структурными наборами характеристик – показателей качества ПО. Эти показатели, в свою очередь, детализируются атрибутами (суб-характеристиками), как показано на рис. 7.3.

К основным характеристикам, определяющим модель качества программной системы, относятся:

1) функциональность – соответствие функциональных возможностей ПО набору требуемого пользователем предназначения;
2) надёжность – способность ПО сохранять необходимый уровень качества функционирования в заданных условиях за определенный период времени;

Качество ПО

3) практичность (применимость) – характеризует объемы работ, требуемых для исполнения и индивидуальной оценки такого исполнения определенным или предполагаемым кругом пользователей;

4) эффективность – оценивает соотношение между уровнем качества функционирования ПО и временной производительностью (скоростью работы) в сочетании с объемом используемых ресурсов;

5) сопровождаемость – показатель, определяемый объемом работ, требуемых для проведения конкретных изменений (модификаций) в процессах сопровождения и продолжающейся разработки;

Качество ПО

6) мобильность – способность ПО к перенесению из одного аппаратно-программного окружения в другое.

В целом представление качества ПО является иерархически многоуровневым, где:

− первый уровень отражает комплекс перечисленных выше шести характеристик (показателей) качества ПО;

− второму уровню соответствуют атрибуты для каждого показателя качества, детализирующие его разные аспекты и использующиеся при оценке качества (рис. 7.2);

Качество ПО

− третий уровень определен для измерения и оценки качества с помощью метрик, каждая из которых определяется как комбинация метода измерения атрибута и шкалы измерения значений атрибутов;

− четвертый уровень представляет собой оценочные элементы метрики (веса), используемые для получения количественного значения или качественной оценки конкретного атрибута показателя ПО.

Приведенные характеристики и атрибуты качества ПО используются для систематического описывания требований к нему, определяя, какие конкретные свойства программ хотят обеспечить заинтересованные стороны по заданной хар-ке.

Качество ПО

Эти требования должны определять следующие аспекты качества:

– что ПО должно делать, например позволять клиенту оформлять заказы и обеспечить их доставку;

– обеспечивать контроль качества строительства и контролировать проблемные места;

– насколько оно должно быть надежно, например работать 7 дней в неделю и 24 часа в сутки;

– никакие введенные пользователями данные при отказе не должны теряться;

– насколько этим ПО должно быть удобно пользоваться, например пользователь, зная название товара и имея средние навыки работы в

Качество ПО

сети, должен находить нужный ему товар за не более чем 1 мин;

- насколько ПО должно быть эффективно, например поддерживать обслуживание до 10000 запросов в секунду, или время отклика на запрос при максимальной загрузке не должно превышать 2 с;
- насколько удобным должно быть его сопровождение, например добавление в систему нового вида запросов не более 3 человеко-дней;
- насколько оно должно быть переносимо, например ПО должно работать на операционных системах Linux, Windows и MacOS X;
- приложение должно взаимодействовать с документами в популярных форматах.

Качество ПО

Сертификация программного обеспечения

Основные понятия сертификации

Основные понятия и определения приведены в стандарте ISO/IEC 00002 – Общие термины и определения в области стандартизации и смежных видов деятельности.

Сертификация соответствия – действие третьей стороны, доказывающее, что обеспечивается необходимая уверенность в том, что должным образом идентифицированная продукция, процесс или услуга соответствует конкретному стандарту или нормативному документу.

Качество ПО

Сертификат соответствия – документ, изданный в соответствии с правилами системы сертификации, удостоверяющий, что обеспечивается необходимая уверенность в том, что должным образом идентифицированная продукция, процесс или услуга соответствует конкретным стандартам или нормативным документам.

Цель сертификации технологий проектирования и производства систем и программных средств – защита интересов пользователей, государственных и ведомственных интересов на основе контроля качества продуктов, обеспечения высоких потребительских свойств, повышения эффективности затрат в сфере их производства,

Качество ПО

эксплуатации и сопровождения, повышения объективности оценок характеристик и обеспечения конкурентоспособности конечного продукта.

При анализе и организации процессов сертификационных испытаний и/или объектов системы и комплекса программ следует учитывать базовые компоненты методологии сертификации:

- цели сертификации – правовые, экономические, формальные;
- проблемы, которые необходимо решать для обеспечения высокой эффективности и достоверности результатов сертификационных испытаний;

Качество ПО

- исходные данные и документы, необходимые для проведения сертификации: стандарты и нормативные документы, их структура и содержание;
- характеристики и классификацию продуктов и/или процессов сертификации и их показатели качества;
- ресурсы обеспечения и испытаний – финансовые, кадры специалистов, их аппаратурная оснащённость, нормативные и инструментальные средства.

Виды сертификации:

- обязательная сертификация;
- добровольная сертификация.

Качество ПО

Обязательная сертификация необходима для программных продуктов и их производства, выполняющих ответственные функции, в которых недостаточное качество, ошибки или отказы могут нанести большой ущерб или опасны для жизни и здоровья людей. Обязательная сертификация программных продуктов способствует значительному снижению риска заказчика и повышению безопасности функционирования программного продукта у потребителя до необходимого уровня.

Добровольная сертификация применяется с целью повышения конкурентоспособности продукции, расширения сферы её использования и получения экономических преимуществ.

Качество ПО

Экономическими целями сертификации являются большие тиражи изделий при производстве, большая длительность жизненного цикла с множеством версий, снижение налогов за высокое качество, увеличение прибыли разработчиков и поставщиков программного продукта, сокращение рекламаций пользователей.

Требования к качеству функционирования ПП

Сертификация программного продукта должна обеспечивать высокое качество результатов производства и полное соответствие требованиям заказчика и пользователей.

Качество ПО

Сертификационные испытания включают этапы:

- формирование функциональных и конструктивных требований к программным продуктам, требования к допустимым рискам и проверку корректности;
- организация сертификационных испытаний программных продуктов на соответствие требованиям, предварительный выбор стратегии, планирование и оценки затрат на испытания;
- подготовка сертификационных испытаний программных продуктов, выбор методов, требования к тестам и процессы генерации динамических тестов внешней среды в реальном времени;

Качество ПО

- программы, методики и процессы испытаний программных продуктов, а также эксплуатационной документации на соответствие требованиям к программным продуктам;
- завершение испытаний, оформление отчётной документации и утверждение акта результатов сертификационных испытаний программных продуктов на соответствие требованиям, тиражирование и внедрение сертификационных версий программных продуктов.

Для сертификации программных продуктов необходимо определить свойства, выделить и формализовать характеристики.

Качество ПО

Основные требования к качеству комплексов программ для сертификации определяют базовые международные стандарты.

Методики оценки качества ПС при сертификации

В качестве объекта оценки качества работы программно-технических средств используется информационная система.

Методики оценки качества программно-технических средств приведены на рисунке 7.4.

Качество ПО

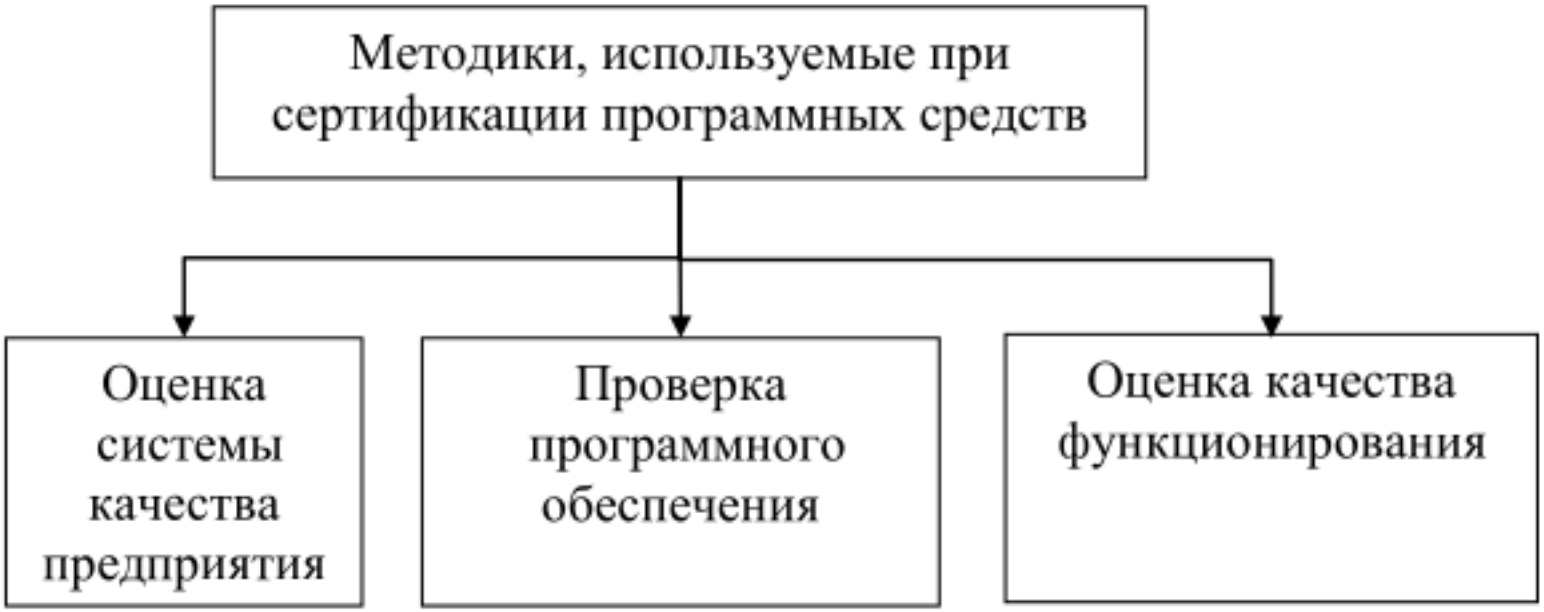

Рисунок 7.4 – Методики оценки качества

Качество ПО

Виды проверки программного обеспечения:

- проверка на отсутствие опасных закладных элементов;
- проверка на соответствие функциональных возможностей декларируемым в документации.

Оценка качества функционирования включает:

- надёжность представления выходной информации;
- своевременность представления выходной информации;
- полноту отражения в базе данных объектов учёта предметной области;

Качество ПО

- безошибочность входной информации;
- защищённость ресурсов от программных вирусов;
- актуальность базы данных;
- защищённость ресурсов от несанкционированного доступа.

Разработанные методики требуют большого перечня входной информации для оценки качества программно-технических средств. Подготовка большого объёма входных данных затрудняет процесс оценки качества программно-технических средств.

Качество ПО

Критерии оценки качества

Для упрощения процесса оценки качества программно-технических средств выбирается значимый критерий – своевременность представления запрашиваемых выходных данных информационных систем.

Метод определяет вероятностно-временные характеристики функционирования программно-технических средств. Получение данных производится по запросам с фиксированным временем выполнения. Моменты поступления запросов носят случайный характер.

Для оценки качества информационных систем применяются два критерия:

Качество ПО

- своевременное представление информации, если среднее время T_{ij} обработки запроса на получение выходного документа i -го типа j –го приоритета не более заданного. В этом случае выполняется условие

$$T_{ij} \le t_i^{зад} , \qquad (6.1)$$

где T_{ij} – среднее время T_{ij} обработки запроса на получение выходного документа i -го типа j -го приоритета;

$t_i^{зад}$ – заданное время.

Качество ПО

- своевременное представление информации, если вероятность представления выходного документа i -го типа j -го приоритета за заданное время окажется не менее установленной. В этом случае выполняется условие

$$P\left(T_{ij} \le t_i^{зад}\right) \ge P_i^{треб} \qquad (6.2)$$

где $P(T_{ij} \le t_i^{зад})$ – вероятность представления выходного документа i -го типа j -го приоритета за заданное время;

$P_i^{треб}$ – заданная вероятность.

Качество ПО

Для оценки своевременности представления информации по критерию (6.1) используются модели:

- модель бесприоритетного обслуживания;
- модель обслуживания с относительными приоритетами;
- модель обслуживания с абсолютными приоритетами.

Модель бесприоритетного обслуживания

При поступлении нового запроса не происходит прерывания обработки предыдущего запроса. Запросы обслуживаются в порядке поступления в информационную систему.

Качество ПО

При решении задачи определяется показатель качества – среднее время обработки запроса. Поступление запросов последовательное, по одному запросу каждого типа. Обслуживание запросов осуществляется в порядке появления в очереди.

Алгоритм решения задачи бесприоритетного обслуживания запросов выходных данных информационной системы:

1 Построение диаграммы поступления и обслуживания запросов выходной информации.

2 Определение T_i моментов времени окончания обслуживания запросов выходной информации. Переменная i – номер запроса выходной информации, i = 1,…, n.

Качество ПО

Величина n – количество запросов выходной информации.

3 Определение суммарного времени T выполнения запросов выходной информации по формуле

$$T = \sum_{i=1}^{n} T_i \,, \qquad (6.3)$$

где T_i – момент времени окончания обслуживания запросов выходной информации;

i – номер запроса выходной информации, i = 1, …, n ;

n – количество запросов выходной информации.

Качество ПО

4 Определение T среднего времени выполнения запросов выходной информации по формуле

$$\bar{T} = T / n \qquad (6.4)$$

где $\bar{T}$ – суммарное время выполнения запросов выходной информации

n – количество запросов выходной информации.

Качество ПО

5 Проверка критерия своевременности представления запрашиваемой выходной информации по формуле

$$T \leq t_{зад} , \qquad (6.5)$$

где T – среднее время выполнения запросов выходной информации;

$t_{зад}$ – заданное время по условию задачи.

6 Вывод результата решения задачи.

Качество ПО

Модель обслуживания запросов информации с относительными приоритетами

Запрос, который начал выполняться, не прерывается при поступлении запроса с более высоким приоритетом. Приоритеты с меньшим значением имеют больший вес. Приоритет с номером 1 считается наивысшим.

При решении задачи определяется показатель качества – среднее время обработки запроса. Поступление запросов последовательное, по одному запросу каждого типа.

Алгоритм решения задачи обслуживания запросов выходной информации с относительными приоритетами:

Качество ПО

1 Построение диаграммы поступления и обслуживания запросов выходной информации с относительными приоритетами.

2 Определение первоочерёдности обслуживания запросов выходной информации в соответствии с моментом поступления и относительным приоритетом запроса в очереди. Обслуживание текущего запроса выходной информации не останавливается, если в процессе выполнения запроса в очередь поступил новый запрос с более высоким приоритетом. Запрос с более высоким приоритетом будет исполняться после завершения обслуживания текущего запроса.

Качество ПО

3 Определение T_i моментов времени окончания обслуживания запросов выходной информации с относительными приоритетами. Переменная i – номер запроса выходной информации, i = 1, … , n. Величина n – количество запросов выходной информации.

4 Определение суммарного времени T выполнения запросов выходной информации по формуле 6.3.

5 Определение $\overline{T}$ среднего времени выполнения запросов выходной информации по формуле 6.4.

Качество ПО

6 Проверка критерия своевременности представления запрашиваемой выходной информации по формуле 6.5.

7 Вывод результата решения задачи.

Модель обслуживания запросов информации с абсолютными приоритетами

Модель обработки запросов выходных данных, в которой при поступлении нового запроса с более высоким приоритетом выполнение текущего запроса приостанавливается и начинается обслуживание вновь поступившего запроса.

Качество ПО

Запрос, выполнение которого временно прекращено, возвращается в очередь, и его обслуживание будет продолжено, когда приоритет запроса окажется наивысшим среди всех запросов информации, находящихся в данный момент в очереди.

При решении задачи определяется показатель качества – среднее время обработки запроса.

Алгоритм решения задачи обслуживания запросов выходной информации с абсолютными приоритетами:

1 Построение диаграммы поступления и обслуживания запросов выходной информации с абсолютными приоритетами.

Качество ПО

2 Определение первоочерёдности обслуживания запросов выходной информации в соответствии с моментом поступления и абсолютным приоритетом запроса в очереди. Обслуживание текущего запроса выходной информации приостанавливается, если в процессе выполнения запроса в очередь поступил новый запрос с более высоким приоритетом.

3 Определение времени распределения обработки запросов выходной информации.

4 Определение T_i моментов времени окончания обслуживания запросов выходной информации с абсолютными приоритетами. Переменная i – номер запроса выходной информации, i = 1, …, n. Величина n – количество запросов выходной информации.

Качество ПО

5 Определение суммарного времени T выполнения запросов выходной информации по формуле 6.3.

6 Определение $\overline{T}$ среднего времени выполнения запросов выходной информации по формуле 6.4.

7 Проверка критерия своевременности представления запрашиваемой выходной информации по формуле 6.5.

8 Вывод результата решения задачи.

Модели оценки сложности программных модулей

Модели оценки сложности

Качество программных средств во многом зависит от их сложности. Например, чем сложнее программа, тем ниже ее надежность и сопровождаемость. Поэтому при оценке качества программ обычно оценивается и их сложность.

Метрики лексического анализа программ

Лексический анализ текста программы – процесс определения длины реализации и объема программы на основе анализа текста программы посредством подсчёта количества операндов, операторов и числа их вхождений в текст программы.

Модели оценки сложности

Для оценки характеристик программ на основе лексического анализа применяются метрики Холстеда, Джилба и Чепина.

Метрики Холстеда

При проведении лексического анализа текста программы определяются длина реализации и объём программы.

Основные метрические характеристики программы:

- η_1 – число простых (уникальных) операторов в программе;

Модели оценки сложности

- η_2 – число простых (уникальных) операндов в программе;
- N_1 – общее число всех операторов в программе;
- N_2 – общее число всех операндов в программе;
- f_{1j} – число появлений в программе j -го оператора, j = 1, …, η_1;
- f_{2j} – число появлений в программе j -го операнда, j = 1, …, η_2 ;

На основе метрических характеристик программы определяются словарь, длина реализации программы и длина программы.

Модели оценки сложности

Количество слов в словаре определяется по формуле

$$\eta = \eta_1 + \eta_2 , \qquad (2.1)$$

где η_1 – число простых (уникальных) операторов в программе;

η_2 – число простых (уникальных) операндов в программе.

Длина реализации программы определяется по формуле

$$N = N_1 + N_2 , \qquad (2.2)$$

где N_1 – общее число всех операторов в программе;

N_2 – общее число всех операндов в программе.

Модели оценки сложности

Длина программы определяется по формуле

$$\overline{N} = (\eta_1 * \log_2 \eta_1) + (\eta_2 * \log_2 \eta_2) , \qquad (2.3)$$

где η_1 – число простых (уникальных) операторов в программе;

η_2 – число простых (уникальных) операндов в программе.

Метрики длины программы и длины реализации программы используются для выявления несовершенств программирования:

Модели оценки сложности

- наличие последовательности дополняющих друг друга операторов к одному и тому же операнду;

- наличие неоднозначных операндов;
- наличие синонимичных операндов;
- наличие общих подвыражений;
- ненужное присваивание;

- наличие выражений, которые не представлены в свёрнутом виде как произведение множителей.

Модели оценки сложности

Объем программы определяется по формуле

$$V = N * \log_2 \eta = \eta * \log_{2^2} \eta \,, \qquad (2.4)$$

где N – длина реализации программы;

η – количество слов в словаре.

Соотношение Холстеда имеет вид:

$$N = \eta_1 * \log_2 \eta + \eta_2 * \log_2 \eta \approx$$

$$\eta_1 * \log_2 \eta_1 + \eta_2 * \log_2 \eta_2 = N_1 + N_2 \,, \quad (2.5)$$

Модели оценки сложности

где η_1 – число простых (уникальных) операторов в программе;

η_2 – число простых (уникальных) операндов в программе;

η – количество слов в словаре;

N_1 – общее число всех операторов в программе;

N_2 – общее число всех операндов в программе.

Взаимная связь между длиной реализации программы и размером словаря определяется по формуле

$$N \approx \eta * \log_2 \eta \,, \qquad (2.6)$$

где η – количество слов в словаре.

Модели оценки сложности

Взаимная связь между величиной словаря и объёмом программы определяется по формуле

$$V = \eta * \log_2^2 \eta \,, \qquad (2.7)$$

где η – количество слов в словаре.

Метрики Джилба

В качестве меры логической сложности Джилб предложил число логических «двоичных принятий решений». Логическая сложность является определяющим фактором для оценки стоимости программы на начальных этапах проектирования.

Модели оценки сложности

Логическую сложность программы Джилб определяет как насыщенность программы условными операторами if-then-else и операторами цикла.

Характеристики программного средства:

- CL – абсолютная сложность программы, характеризуемая количеством операторов условий;
- cl – относительная сложность программы, определяющая насыщенность программы операторами условия;
- количество операторов цикла;
- количество операторов условия;
- число модулей или подсистем;
- отношение числа связей;

Модели оценки сложности

- отношение числа ненормальных выходов из множества операторов к общему числу операторов.

Программную надёжность Джилб предлагает рассчитывать как единицу минус отношение числа логических сбоев к общему числу запусков.

Точность определяется как отношение количества правильных данных ко всей совокупности данных.

Прецизионность определяется как мера того, насколько часто появляются ошибки, вызванные одинаковыми причинами.

Модели оценки сложности

Прецизионность рассчитывается как отношение числа фактических ошибок на входе к общему числу наблюдённых ошибок, причинами которых явились эти ошибки на входе.

Метрики Чепина

Метод Чепина состоит в оценке информационной прочности программного модуля на основе результатов анализа характера использования переменных, входящих в состав списка ввода и вывода.

Множество переменных, составляющих список ввода-вывода, разбивается на четыре группы:

Модели оценки сложности

- P – вводимые переменные для расчётов и для обеспечения вывода;
- M – модифицируемые, создаваемые внутри программы, переменные;
- C – переменные, участвующие в управлении работой программного модуля;
- T – неиспользуемые в программе переменные.

Выражение для определения метрики Чепина имеет вид

$$Q = a_1 * P + a_2 * M + a_3 * C + a_4 * T , \qquad (2.8)$$

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

Модели оценки сложности

Весовые коэффициенты применяются для отражения влияния на сложность программы функциональной группы переменных.

С учётом весовых коэффициентов расчётное выражение метрики Чепина имеет вид

$$Q = 1 * P + 2 * M + 3 * C + 0,5 * T , \qquad (2.9)$$

где P – вводимые переменные;

M – модифицируемые переменные;

C – переменные управления;

T – неиспользуемые переменные.

Модели оценки сложности

Метрики структурной сложности программ

Структурная сложность программ определяется:

- количеством взаимодействующих компонентов;
- числом связей между компонентами;
- сложностью взаимодействия между компонентами.

Поведение выполняющейся программы и связь между входными и выходными данными определяется набором маршрутов управляющего потокового графа. Управляющий потоковый граф – множество вершин и дуг графа управления программой. Сложность программных модулей связана с количеством и сложностью маршрутов выполнения.

Модели оценки сложности

Метрика структурной сложности программ используется для оценки трудоёмкости тестирования и сопровождения модуля, оценки надёжности функционирования.

Маршруты выполнения программы подразделяются на два вида:

- вычислительные маршруты;
- маршруты принятия логических решений.

Вычислительные маршруты – это маршруты арифметической обработки данных, предназначенные для преобразования величин, являющихся результатами измерения характеристик.

Модели оценки сложности

Сложность вычислительных маршрутов определяется по формуле

$$S_1 = \sum_{i=1}^{m} \sum_{j=1}^{l_i} \nu_j , \qquad (2.10)$$

где m – количество маршрутов выполнения программы;

l_i – число данных, обрабатываемых в i -м маршруте;

ν_j – число значений обрабатываемых данных j -го типа, $2 \le \nu_j \le 5$.

Модели оценки сложности

Маршруты принятия логических решений – это маршруты управляющего потокового графа программы, отражающие логику выполнения программы, которая может изменять последовательность выполнения команд, переводить управление на удалённые участки программы или завершать выполнение.

Сложность маршрутов принятия логических решений определяется по формуле

$$S_2 = \sum_{i=1}^{m} p_i , \qquad (2.11)$$

где p_i – число ветвлений или число проверяемых условий в i -м маршруте.

Модели оценки сложности

Общая сложность программы определяется по формуле

$$S = c * \sum_{i=1}^{m} \left(\sum_{j=1}^{l_i} \nu_i + p_i \right), \qquad (2.12)$$

где c – коэффициент пропорциональности, корректирующий взаимное использование метрик сложности вычислительных маршрутов и маршрутов принятия логических решений.

Модели оценки сложности

Критерии формирования маршрутов

Поток управления – последовательность выполнения модулей и операторов программы. Граф потока управления (управляющий граф) – ориентированный граф, моделирующий поток управления программой.

Оценка структурной сложности программы определяется по критериям:

- критерий 1;
- критерий 2;
- критерий 3.

Модели оценки сложности

Структурная сложность по первому критерию вычисляется по формуле 2.11.

Критерий 1 предполагает, что граф потока управления программой должен быть проверен по минимальному набору маршрутов, проходящих через каждый оператор ветвления по каждой дуге.

Прохождение по маршруту выполняется один раз, повторная проверка дуг не проводится и считается избыточной.

В процессе проверки гарантируется выполнение всех передач управления между операторами программы и каждого оператора не менее одного раза.

Модели оценки сложности

Критерий 2 основан на анализе базовых маршрутов в программе, которые формируются и оцениваются на основе цикломатического числа графа потока управления программой.

Цикломатическое число графа потока управления программой определяется по формуле

$$Z = m - n + 2 * p, \qquad (2.13)$$

где m – общее число дуг в графе;

p – число связных компонентов графа.

Модели оценки сложности

Число связных компонентов графа p равно количеству дуг, необходимых для превращения исходного графа в максимально связный граф – граф потока управления, у которого любая вершина доступна из любой другой вершины.

Второй критерий, выбранный по цикломатическому числу, требует однократной проверки или тестирования каждого линейно независимого цикла и каждого линейно независимого ациклического участка.

При использовании второго критерия количество проверяемых маршрутов равно цикломатическому числу.

Модели оценки сложности

Для автоматического анализа графов по второму критерию используются матрицы смежности и достижимости графов, содержащие информацию о структуре проверяемой программы.

Третий критерий формирования маршрутов для тестирования программ основан на формировании полного состава базовых структур графа потока управления программой и заключается в анализе каждого из реальных ациклических маршрутов исходного графа программы и каждого цикла.

Каждый компонент структуры программы должен быть проанализирован хотя бы один раз.

Модели оценки сложности

Если при прохождении ациклического маршрута исходного графа потока управления достижимы несколько элементарных циклов, то при тестировании должны исполняться все достижимые циклы.

Метрика Маккейба

Метрика Маккейба характеризует цикломатическое число графа потока управления программой и определяется по формуле

$$M = m - n + 2, \qquad (2.14)$$

где m – количество рёбер графа;

n – число вершин графа.

Модели оценки сложности

Цикломатическое число Маккейба – величина M, рассчитанная по формуле 2.14.

Цикломатическая сложность программы – структурная мера сложности программы, применяемая для измерения качества программного обеспечения и основанная на методах статического анализа кода программных средств.

При определении цикломатической сложности используется граф потока управления программой: узлы графа соответствуют неделимым группам команд программы и ориентированным рёбрам, каждый из которых соединяет два узла и соответствует двум командам.

Модели оценки сложности

Цикломатическая сложность может быть применена для отдельных функций, модулей, методов или классов в пределах анализируемого программного средства. Эта стратегия тестирования называется основным маршрутом тестирования Маккейба. Тестирование представляет собой проверку линейного независимого маршрута графа программы. Количество тестов должно быть равно цикломатической сложности.

Теоретической основой определения цикломатического числа Маккейба является теория графов.

Оценка качества сложной программной системы

Оценка качества

Функциональные указатели и метрики свойств

Оценка качества процедурно-ориентированных программных средств производится на основе функциональных указателей (FP – functions points) и указателей свойств (FP – features points).

Функциональные указатели характеризуют функциональную сложность программы. Количество функциональных указателей определяется количеством ссылок на внешний ввод, внешний вывод, внешний запрос, автономный локальный файл, глобальный файл.

Количество функциональных указателей определяется по формуле:

Оценка качества

$$FP = F * (0{,}65 + 0{,}01 * \sum_{i=1}^{14} k_i) , \qquad (2.15)$$

где k_i – коэффициент регулировки сложности, зависящий от ответов на вопросы, связанные с особенностями программы, i = 1, …, 14 .

F – общее количество функциональных указателей.

Общее количество функциональных указателей определяется по формуле

$$F = \sum_{j=1}^{5} f_j, \qquad (2.16)$$

где f_j – оценочные элементы, j = 1, …, 5.

Оценка качества

Оценочные элементы подразделяются на виды:

- f_1 – количество внешних вводов (вводов данных пользователем);
- f_2 – количество внешних выводов (отчёты, экраны, распечатки, сообщения);
- f_3 – количество внешних запросов (диалоговых вводов-выводов);

- f_4 – количество локальных внутренних логических файлов (группа логически связанных данных, которая размещена внутри приложения и обслуживается через внешние вводы, базы данных);

Оценка качества

- f_5 – количество внешних интерфейсных файлов, разделяемых с другими программами глобальных файлов (группа логически связанных данных, которая размещается и поддерживается внутри другого приложения, внутренний логический файл в другом приложении).

  Значения коэффициентов регулировки сложности k_i зависят от ответов на вопросы, касающиеся влияния определенных факторов на выполнение функций программного обеспечения:

1 Какое влияние имеет наличие средств передачи данных?

2 Какое влияние имеет распределенная обработка данных?

Оценка качества

3 Какое влияние имеет распространенность используемой аппаратной платформы?

4 Какое влияние имеет критичность к требованиям производительности и ограничению времени ответа?

5 Какое влияние имеет частота транзакций?

6 Какое влияние имеет ввод данных в режиме реального времени?

7 Какое влияние имеет эффективность работы конечного пользователя?

8 Какое влияние имеет оперативное обновление локальных файлов в режиме реального времени?

9 Какое влияние имеет скорость обработки данных?

Оценка качества

10 Какое влияние имеют количество и категории пользователей?

11 Какое влияние имеет легкость инсталляции?

12 Какое влияние имеет легкость эксплуатации?

13 Какое влияние имеет разнообразие условий применения?

14 Какое влияние имеет простота внесения изменений?

Ответы на каждый вопрос должны быть даны в соответствии со следующими категориями ответов, касающихся влияния конкретных факторов:

Оценка качества

- влияние случайное;
- влияние небольшое, эпизодическое;
- влияние среднее;
- влияние важное, значительное;
- влияние основное, существенное.

Значения весовым коэффициентам важности назначают следующим образом:

- $k_i = 1$, если на i -й вопрос выбран ответ «влияние случайное»;
- $k_i = 2$, если на i -й вопрос выбран ответ «влияние небольшое»;

Оценка качества

- k_i = 3, если на i -й вопрос выбран ответ «влияние среднее»;
- k_i = 4, если на i -й вопрос выбран ответ «влияние важное»;
- k_i = 5, если на i -й вопрос выбран ответ «влияние основное».

Оценка качества программных средств определяется по формуле:

$$DQ = \frac{\text{Количество ошибок}}{\sum_{j=1}^{5} f_j * (0.65 + 0.01 * \sum_{i=1}^{14} k_i)}, \qquad (2.17)$$

Оценка качества

где f_j – оценочные элементы, j = 1, …, 5;

k_i – коэффициент регулировки сложности, зависящий от ответов на вопросы, связанные с особенностями программы, i = 1, …, 14.

После вычисления количества функциональных указателей FP на его основе формируются косвенные метрики:

- производительность;
- качество;
- удельная стоимость;
- документированность.

Оценка качества

Производительность определяется как отношение количества функциональных указателей к затратам. Единицы измерения производительности – FP/чел.-мес.

Качество рассчитывается как отношение количества ошибок к количеству функциональных указателей. Единицы измерения качества – единицы/FP.

Удельная стоимость определяется как отношение стоимости к количеству функциональных указателей. Единицы измерения удельной стоимости – тыс. руб./FP.

Оценка качества

Документированность рассчитывается как отношение количества страниц документа к количеству функциональных указателей. Единицы измерения документированности – количество страниц/FP.

Функциональные указатели применяются в коммерческих информационных системах.

Оценка качества программных средств с высокой алгоритмической сложностью производится посредством метрики свойств (features points).

Они применяются в системном и инженерном программном обеспечении, встроенном программном обеспечении.

Оценка качества

Для вычисления указателя свойств используется характеристика – количество алгоритмов. Для формирования указателя свойств составляется таблица. Расчёт указателя свойств производится по формуле расчёта функциональных указателей (2.15).

Связность программных модулей

Программный модуль – часть программы, оформленная как самостоятельный программный продукт, который может быть применён для использования в описаниях проектируемого процесса.

Оценка качества

Программные модули позволяют снизить сложность программы и исключить повторение ранее проделанных действий при программировании.

При использовании модульной структуры программы необходимо обеспечить связывание программных модулей.

Для оценки качества программного средства используют метрики связности модулей: уровень качества программного продукта прямо пропорционален силе связности (прочности) программных модулей. Для обеспечения надёжности программы разработка качественного программного средства должна максимизировать связность программных модулей.

Оценка качества

Связность – мера прочности соединения функциональных и информационных объектов в пределах одного программного модуля.

Типы связности программных модулей:

1 Связность по совпадению (сила связности 0) – отсутствие внутренних связей в программных модулях.

2 Логическая связность (сила связности 1) – связность в программных модулях, содержащих подпрограммы для различных вариантов обращения к модулям.

3 Временная связность (сила связности 3) – связность программных модулей, составные части которых не связаны между собой, но должны быть выполнены в течение определённого периода времени.

Оценка качества

4 Процедурная связность (сила связности 5) – связность программных модулей, в которых имеет место порядковая связь составляющих их частей, реализующих процедуру обработки данных.

5 Коммуникационная связность (сила связности 7) – связность программных модулей, составляющие которых связаны по данным (используют одну и ту же структуру данных).

6 Информационная связность (сила связности 9) – связность программных модулей, когда выходные данные одной части модуля используются как входные данные другой части модуля..

Оценка качества

7 Функциональная связность (сила связности 10) – связность, при которой модуль как единое целое реализует единственную функцию, так как программный модуль содержит набор объектов, предназначенных для выполнения одной и только одной задачи.

Связность типов 1, 2, 3, 4 свидетельствует о недостатках проектирования архитектуры программы с точки зрения тестирования и последующего сопровождения.

Программные модули с высокой функциональной связностью создают хорошие предпосылки для применения прогрессивной технологии повторного использования в процессе будущей разработки.

Оценка качества

Для определения типа связности программных модулей применяется методика:

1 Если модуль реализует единственную прикладную функцию, то тип связности – функциональный. Далее анализ можно прекратить (конец процедуры). В противном случае следует перейти к пункту 2.

2 Если действия внутри модуля связаны, то нужно перейти к пункту 3. Если же действия внутри модуля ничем не связаны, то следует перейти к пункту 5 для дополнительного анализа.

3 Если действия внутри модуля связаны по данным, то перейти к пункту 4, а если действия внутри модуля связаны потоком управления – перейти к пункту 6.

Оценка качества

4 Если порядок действий внутри модуля важен, то тип связности информационный, в противном случае тип связности – коммуникационный. Анализ прекращается, и осуществляется переход к концу процедуры.

5 Если действия внутри модуля принадлежат к одной категории, то уровень связности – логический. Если действия внутри модуля не принадлежат к одной категории, то модуль обладает типом связности по совпадению. Далее следует переход к концу процедуры.

6 Если порядок действия внутри программного модуля важен, то тип связности – процедурный. В противном случае тип связности – временной. Анализ прекращается.

Оценка качества

Сцепление программных модулей

Сцепление – мера межмодульной связи, которую для повышения качества программных средств необходимо уменьшать.

Для оценки качества программного средства используют метрики сцепления модулей: уровень качества программного продукта обратно пропорционален силе сцепления программных модулей.

Для обеспечения надёжности программы разработка качественного программного средства должна минимизировать сцепление программных модулей.

Оценка качества

Для измерения сцепления используют целочисленную шкалу степени сцепления в интервале значений от 0 до 9. Для каждой степени сцепления определен тип сцепления, который может быть сопоставлен модулю в процессе проектирования.

Типы сцепления программных модулей:

1 Сцепление по данным (сила сцепления 1) – сцепление, при котором модуль является вызываемым, входные параметры представляются простыми элементами данных.

2 Сцепление по образцу (сила сцепления 3) – сцепление, при котором модуль является вызываемым, а в качестве параметров используются агрегатные типы данных.

Оценка качества

3 Сцепление по управлению (сила сцепления 4) – сцепление, при котором модуль является вызывающим и передаёт вызываемому модулю список управляющих параметров, явно влияющих на его работу.

4 Сцепление по внешним ссылкам (сила сцепления 5) – сцепление, при котором модуль имеет адресные ссылки (указатели) на глобальный элемент данных, на который ссылается другой программный модуль.

5 Сцепление по общей области (сила сцепления 7) – сцепление, при котором модуль разделяет (совместно использует) с другим модулем одну и ту же глобальную структуру данных.

Оценка качества

6 Сцепление по содержанию (сила сцепления 9) – сцепление, при котором модуль напрямую ссылается на содержимое другого модуля.

Хорошо спроектированная система характеризуется слабым сцеплением программных модулей.

Методы достижения слабого сцепления программных модулей:

- удаление необязательных связей;
- снижение количества необходимых связей;
- упрощение необходимых связей.

Способы снижения сцепления программных модулей:

Оценка качества

1 Создание минимальных по количеству параметров межмодульных связей.

2 Создание прямых межмодульных связей.

3 Создание локализованных связей.

4 Создание явных связей.

5 Создание гибких связей для облегчения модификаций.

Объектно-ориентированные метрики

Для объектно-ориентированных метрик целесообразно представление абстракций в терминологии измерения класса.