Федеральное агентство по образованию
Государственное образовательное учреждение высшего
профессионального образования
«Самарский государственный аэрокосмический университет
имени академика С.П. Королева» (СГАУ)

Оценка качества программных средств

Методические указания
к практическим занятиям по курсу
«Стандартизация и сертификация научно – технической продукции»

САМАРА 2007

Составитель: В.А. Колпаков

УДК 681.3

Оценка качества программных средств: Метод. указания к практическим занятиям по курсу «Стандартизация и сертификация научно – технической продукции» / СГАУ; Сост. *В.А. Колпаков*. Самара, 2007. 19с.

В методических указаниях рассматриваются основы тестирования программных средств, являющегося одним из ключевых этапов жизненного цикла продукции. Изучаются методы по оценке показателей качества программного средства и его надежности. Описывается методика оценки технико – экономических показателей разработки программных средств.

Работа предназначена для студентов специальности 010200 «Прикладная математика и информатика» и является циклом работ по курсу «Стандартизация и сертификация научно – технической продукции». Выполнена на кафедре технической кибернетики.

Работа выполнена при поддержке Министерства образования и науки РФ, Администрации Самарской области и Американского фонда гражданских исследований и развития (CRDF Project SA-014-02) в рамках российско-американской программы “Фундаментальные исследования и высшее образование” (BRHE), а также за счет средств инновационной образовательной программы СГАУ «Развитие центра компетенции и подготовка специалистов мирового уровня в области аэрокосмических и геоинформационных технологий»

Печатается по решению редакционно-издательского совета Самарского государственного аэрокосмического университета академика С.П. Королева

Рецензент: зав. кав. геоинформатики, д.т.н., профессор Сергеев В.В.

СОДЕРЖАНИЕ

Практические занятия

Практические занятия предназначены для закрепления студентами лекционного материала, а также для приобретения ими практических навыков в создании качественного программного обеспечения (ПО). Контроль знаний студентов на практических занятиях осуществляется путем проведения аудиторных самостоятельных работ, а также опроса студентов.

Практическая работа и методические указания по её выполнению

Целью выполнения практической работы является закрепление теоретических знаний и практических навыков в умении оценивать программные средства (ПС).

Результатом выполнения практической работы должно быть оцененное программное средство, реализованное в среде PASCAL, C++ и т.д..

1 Задание на практическую работу

Практическая работа предполагает наличие программного средства, которое должно отвечать следующим параметрам: ориентировано на пользователя, практически не знакомого с персональной техникой; должно иметь эффективный пользовательский интерфейс и средства защиты от несанкционированного доступа. Разработанное ПС должно быть протестировано. Тестовые наборы данных и результаты, полученные при проведении машинного эксперимента, должны быть описаны.

Практическая работа содержит исследование проблем, возникающих при оценке разработанного программного средства. Студенты оценивают качественные показатели ПС, количественные показатели надежности ПС, технико-экономические показатели разработки ПС.

2 Методические рекомендации по тестированию программного средства

Тестирование является одним из этапов жизненного цикла ПС, направленного на повышение качественных характеристик.

Программы, как объекты тестирования, имеют ряд особенностей, которые отличают процесс их тестирования от общепринятого, применяемого при разработке аппаратуры и других технических изделий. Особенностями тестирования программных средств являются:

- отсутствие эталона (программы), которому должна соответствовать тестируемая программа;
- сложность программ и принципиальная невозможность исчерпывающего тестирования;
- практическая невозможность создания единой методики тестирования (формализация процесса тестирования) в силу большого разнообразия программных средств по их сложности, функциональному назначению, области использования и т.д. Применительно к программному средству тестирование - процесс многократного выполнения программ с целью обнаружения ошибок.

Цель тестирования - выявление как можно большего количества ошибок. Тестовый прогон считается удачным, если он позволяет выявить ошибки; эффективным, если имеет высокую вероятность обнаружения большего числа ошибок.

Существуют следующие методы тестирования программ:

- *статический;* наиболее формализованный, базируется на правилах структурного построения программ и обработки данных. Проверка степени выполнения этих правил проводится без изменения объектного кода программы путем формального анализа текста программы на языке программирования. Операторы и операнды текста программы анализируются в символьном виде, поэтому этот метод тестирования иногда называют *символическим* тестированием;
- *детерминированный;* наиболее трудоемкий и детализированный метод тестирования, требует многократного выполнения программы с использованием определенных, специальным образом подобранных тестовых наборов данных. При детерминированном тестировании контролируются каждая комбинация исходных данных и соответствующие результаты, а также каждое утверждение в спецификации тестируемой программы. Детерминированное тестирование в силу трудоемкости, возможно, применять для отдельных модулей в процессе сборки программы или для небольших и несложных программных комплексов;

При тестировании программного средства невозможно перебрать все комбинации исходных данных и проконтролировать результаты функционирования на каждой из них, поэтому для комплексного тестирования программного средства применяется *стохастическое* тестирование. Оно предполагает использование в качестве исходных данных множества случайных величин с соответствующими распределениями. Стохастическое тестирова-

ние применяется в основном для обнаружения ошибок;

Программное средство, предназначенное для работы в системах реального времени, должны проходить *тестирование в реальном масштабе времени.*

Каждый из рассмотренных методов тестирования не исключает последовательного применения другого метода, требование к повышению качества программного средства предполагает необходимость подвергать его различным методам тестирования и (или) их сочетаниям в зависимости от сложности области применения.

Для тестирования программного средства практической работой рекомендуется детерминированное и стохастическое тестирование, и их сочетание.

Детерминированное тестирование или тестирование на определенных входных данных, основывается на двух подходах: структурное тестирование и функциональное тестирование. Структурное тестирование, или тестирование программ как «белого ящика» (стратегия тестирования, управляемого логикой программы), предполагает детальное изучение текста (логики) программы и построение таких входных данных, которые позволили бы при многократном выполнении программы обеспечить выполнение максимально возможного количества маршрутов, логических ветвлений, циклов и так далее. Функциональное тестирование, или тестирование программ, как «Черного ящика» (тестирование по «входу - выходу»), полностью абстрагируется от логики программы, предполагается, что логика программы неизвестна, а тестовые наборы подбираются на основании анализа функциональных входных спецификаций.

При построении наборов данных по принципу «белого ящика» руководствуются следующими критериями:

- покрытие операторов; предполагает выбор такого тестового набора данных, который вызывает выполнение каждого оператора в программе хотя бы один раз. Очень слабый критерий.
- покрытие узлов ветвления; предполагает разработку такого количества тестов, чтобы в каждом узле ветвления был обеспечен переход по веткам «истина» и «ложь» хотя бы один раз.
- покрытие условий; если узел ветвления содержит более одного условия, тогда нужно разработать число тестов, достаточное для того, чтобы возможные результаты каждого условия выполнялись, по крайней мере, один раз, каждой точке входа в программу должно быть передано управление при

вызове, по крайней мере, один раз.

- комбинаторное покрытие условий; используется для выполнения ошибок в логических выражениях. Требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении и все точки входа выполнялись, по крайней мере, один раз.

К стратегии «черного ящика» относятся методы:

- эквивалентного разбиения. Осуществляется в два этапа - выделение классов эквивалентности, построение тестов. Классом эквивалентности называется множество входных значений, каждое из которых имеет одинаковую вероятность обнаружения конкретного типа ошибки.
- анализ граничных значений. Предполагает исследование ситуаций, возникающих на границах и вблизи границ эквивалентных разбиений.
- метод функциональных диаграмм. Заключается в преобразовании входной спецификации программы в функциональную диаграмму (диаграмму причинно-следственных связей) с помощью простейших булевских отношений, построение таблицы решений (методом обратной трассировки), которая является основой для написания эффективных тестовых наборов данных.

1. В спецификации программы выделяются причины и следствия. Причина - отдельное входное условие или класс эквивалентности входных условий. Следствие - выходное условие или результат преобразования системы. Каждой причине и следствию приписывается уникальный номер.
2. Анализируется семантическое содержание спецификации, которая преобразуется в булевский граф, связывающий причины и следствия. Каждая вершина может находиться в состоянии «истина» или «ложь».
3. Диаграмма снабжается примечаниями, задающими ограничения и описывающими комбинации причин и (или) следствий, которые являются невозможными из-за синтаксических или внешних ограничений.
4. По полученной функциональной диаграмме строится таблица решений. Для этого поочередно для каждого следствия, значение которого условно устанавливается в «истину», прослеживается обратный путь (по диаграмме) ко всем причинам, связанным с этим следствием, и фиксируется их состояние. Каждый столбец таблицы решений соответствует тесту.
5. Столбцы решений преобразуют в тесты.
6. В практической работе выполнить тестирование программного средства, предварительно выбрав и обосновав выбранные методы тестирования.

3 Методические рекомендации по оценке программного средства

В практической работе оцениваются качественные показатели ПС, показатели надежности ПС, технико-экономические показатели ПС.

3.1 Оценка качественных показателей программного средства

Методика оценки качественных показателей ПС основана на составлении метрики ПС.

В практической работе необходимо выполнить следующее:

1. Выбрать показатели качества (не менее 10) и сформулировать их сущность. Каждый показатель должен быть существенным, т. е. должны быть ясны потенциальные выгоды его использования. Показатели представить в виде таблицы (таблица 1).

Таблица 1

Показатели качества	Сущность показателя	Экспертная оценка (вес) w_i	Оценка, установленная экспериментом r_i

2. Установить веса показателей $w_i \left(\sum_i w_i = 1 \right)$.

3. Для каждого показателя установить конкретную численную оценку r_i от 0 до 1, исходя из следующего:

0- свойство в ПС присутствует, но качество его неприемлемо;

0.5 - 1 - свойство в ПС присутствует и обладает приемлемым качеством;

1- свойство в ПС присутствует и обладает очень высоким качеством.

Возможно присвоение промежуточных значений в соответствии с мнением оценивающего лица относительно полезности того или иного свойства ПС.

4. Определить качество ПС как иерархическую взвешенную сумму весов отдельных показателей. $Качество\ показателя = w_i \cdot r_i$.

5. ***Ошибка! Закладка не определена.*** Определить среднее значение оценки качества ПС по формуле (1).

$$ПК = \frac{\sum_i w_i \cdot r_i}{Общее\ количество\ показателей} \qquad (1)$$

6. Представить выходные данные:

- перечень всех показателей с оценкой 0 с указанием причин такой оценки;

- гистограмму, показывающую распределение показателей по интервалам оценок;

- какие дефекты ПС обнаружены в результате анализа показателей качества.

3.2 Оценка надежности программного средства

Одной из важнейших характеристик качества программного средства является надежность.

Надежность - свойство программного средства сохранять работоспособность в течение определенного периода времени, в определенных условиях эксплуатации с учетом последствий для пользователя каждого отказа.

Работоспособным называется такое состояние программного средства, при котором оно способно выполнять заданные функции с параметрами, установленными требованиями технического задания. С переходом в неработоспособное состояние связано событие отказа.

Причиной отказа программного средства является невозможность его полной проверки в процессе тестирования и испытаний. При эксплуатации программного средства в реальных условиях может возникнуть такая комбинация входных данных, которая вызовет отказ, следовательно, работоспособность программного средства зависит от входных данных, и чем меньше эта зависимость, тем выше уровень надежности.

Для оценки надежности используются три группы показателей: качественные, порядковые и количественные.

К основным количественным показателям надежности программного средства относятся:

- Вероятность безотказной работы $P(t_3)$ - это вероятность того, что в пределах заданной наработки отказ системы не возникает. Наработка - продолжительность или объем работ.

- $P(t_3) = P(t \geq t_3)$, где t - случайное время работы ПС до отказа, t_3 - заданная наработка.
- Вероятность отказа - вероятность того, что в пределах заданной наработки отказ системы возникает. Этот показатель, обратный предыдущему:

$Q(t_3) = 1 - P(t_3)$.

- Интенсивность отказов системы $\lambda(t)$ - это условная плотность вероятности возникновения отказа программного средства в определенный момент времени при условии, что до этого времени отказ не возник.

$$\lambda(t) = \frac{f(t)}{P(t)}, \qquad (2)$$

где $f(t)$ – плотность вероятности отказа в момент времени t.

$$f(t) = \frac{dQ(t)}{dt} = \frac{d}{dt}\left[1 - P(t)\right] = -\frac{d}{dt}P(t). \qquad (3)$$

Существует следующая связь между $\lambda(t)$ и $P(t)$: В частном случае $\lambda = const$.

$$P(t) = \exp\left(-\int \lambda(t)\, dt\right)$$

$$P(t) = \exp(-\lambda_t). \qquad (4)$$

Если в процессе тестирования фиксируется число отказов за определенный интервал времени, то $\lambda(t)$ - число отказов в единицу времени.

- Средняя наработка до отказа T_i - математическое ожидание времени работы программного средства до очередного отказа

$$T_i = \int t f(t) dt, \qquad (5)$$

где t - время работы программного средства от $(K-1)$ до K отказа.

$$T_i = (t_1 + t_2 + \ldots + t_n)/n, \qquad (6)$$

где t_i - время работы программного средства между отказами, n - количество отказов.

- Среднее время восстановления T_B - математическое ожидание времени восстановления t_{Bi} - времени, затраченного на восстановление и локализацию отказа – $t_{O.Л.i}$, времени устранения отказа - $t_{У.O.i}$, времени пропускной проверки работоспособности - $t_{П.П.i}$.

$$t_{Bi} = t_{O.Л.i} + t_{У.O.i} + t_{П.П.i}. \qquad (7)$$

Для этого показателя термин «время» означает время, затраченное программистом на перечисленные виды работ.

- Коэффициент готовности K_2 - вероятность того, что программное средство ожидается в работоспособном состоянии в произвольный момент времени его использования по назначению:

$$K_2 = \frac{T_i}{T_i + T_B}. \qquad (8)$$

Причиной отказа программного средства являются ошибки, которые могут быть вызваны: внутренним свойством программного средства, реакцией программного средства на изменение внешней среды функционирования. Это значит, что при самом тщательном тестировании, если предположить, что удалось избавиться от всех внутренних ошибок, нельзя с полной уверенностью утверждать, что в процессе эксплуатации программного средства не возникнет отказ.

Основным средством определения количественных показателей надежности являются модели надежности, под которыми понимают математическую модель, построенную для оценки зависимости надежности от заранее известных или оцененных в ходе создания программного средства параметров. В связи с этим определение надежности показателей принято рассматривать в единстве трех процессов - предсказание, измерение, оценивание.

Предсказание - это определение количественных показателей надежности исходя из характеристик будущего программного средства.

Измерение - это определение количественных показателей надежности, основанное на анализе данных об интервалах между отказами, полученных при выполнении программ в условиях тестовых испытаний.

Оценивание - это определение количественных показателей надежности, основанное на данных об интервалах между отказами, полученными при испытании программного средства в реальных условиях функционирования.

Все модели надежности можно классифицировать по тому, какой из перечисленных процессов они поддерживают (предсказывающие, прогнозные, оценивающие, измеряющие). Нужно отметить, что модели надежности, которые в качестве исходной информации используют данные об интервалах между отказами, можно отнести и к измеряющим, и к оценивающим в равной степени. Некоторые модели, основанные на информации, полученной в ходе тестирования программного средства, дают возможность делать прогнозы поведения программного средства в процессе эксплуатации.

Рассмотрим аналитические и эмпирические модели надежности.

Аналитические модели дают возможность рассчитать количественные показатели надежности, основываясь на данных о поведении программы в

процессе тестирования (измеряющие и оценивающие модели). Эмпирические модели базируются на анализе структурных особенностей программ. Они рассматривают зависимость показателей надежности от числа межмодульных связей, количества циклов в модулях, отношения количества прямолинейных участков к количеству точек ветвления и тому подобное. Нужно отметить, что часто эмпирические модели не дают конечных результатов показателей надежности.

Аналитическое моделирование надежности программного средства включает четыре шага:

- определение предложений, связанных с процедурой тестирования программного средства;
- разработка или выбор аналитической модели, базирующейся на предположениях о процедуре тестирования;
- выбор параметров моделей с использованием полученных данных;
- применение модели - расчет количественных показателей надежности по модели.

Аналитические модели представлены двумя группами: динамические и статические модели. В динамических моделях надежности программного средства поведение программы (появление отказов) рассматривается во времени. В статических моделях по явление отказов не связывают со временем, а учитывают только зависимость количества ошибок от числа тестовых прогонов (по области ошибок) или зависимость количества ошибок от характеристики входных данных (по области данных). Для использования динамических моделей необходимо иметь данные о появлении отказов во времени. Статические модели принципиально отличаются от динамических тем, что в них не учитывается время появления ошибок в процессе тестирования и не используется никаких предположений о поведении функции риска $\lambda(t)$. Эти модели строятся на твердом статистическом фундаменте.

Модель Коркорэна

Применение модели предполагает знание следующих ее показателей:

- модель содержит изменяющуюся вероятность отказов для различных источников ошибок и соответственно разную вероятность их исправления;
- в модели используются такие параметры, как результат только N испытаний, в которых наблюдается N_i ошибок i-го типа;

- выявление в ходе N испытаний ошибки i-го типа появляются с вероятностью a_i.

Показатель уровня надежности R вычисляют по следующей формуле:

$$R = \frac{N_0}{N} + \sum_{i=1}^{k} \frac{Y_i \cdot (N_i - 1)}{N}, \qquad (9)$$

где N_0 - число безотказных (или безуспешных) испытаний, выполненных в серии из N испытаний;

k - известное число типов ошибок;

Y_i - вероятность появления ошибок; при $N_i > 0, \quad Y_i = a_i$, при $N_i = 0, \quad Y_i = 0$.

Модель Шумана

Модель Шумана относится к динамическим моделям дискретного времени, данные для которой собираются в процессе тестирования программного обеспечения в течение фиксированных или случайных интервалов времени. Модель Шумана предполагает, что тестирование проводится в несколько этапов. Каждый этап представляет собой выполнение программы на полном комплексе разработанных тестовых данных. Выявленные ошибки регистрируются, но не исправляются. В конце этапа рассчитываются количественные показатели надежности, исправляются найденные ошибки, корректируются тестовые наборы и проводится следующий этап тестирования. В модели Шумана предполагается, что число ошибок в программе постоянно и в процессе корректировки новые ошибки не вносятся. Скорость обнаружения ошибок пропорциональна числу оставшихся ошибок.

Предполагается, что до начала тестирования имеется E_t ошибок. В течение времени тестирования τ обнаруживается ε_c ошибок в расчете на одну команду в машинном языке. Таким образом, удельное число ошибок на одну машинную команду, оставшихся в системе после τ времени тестирования, равно:

$$\varepsilon_r = E_t / I_t \cdot \varepsilon_c(\tau), \qquad (10)$$

где I_t - общее число машинных команд, которое предполагается постоянным в рамках этапа тестирования.

Предполагается, что значение функции частоты отказов $Z(t)$ пропорционально числу ошибок, оставшихся в программе после израсходованного на тестирование времени τ.

$Z(t) = C \cdot \varepsilon_r(\tau)$

где C - некоторая постоянная;

t - время работы программы без отказов.

Тогда, если время работы программы без отказа t отсчитывается от точки $t=0$, а τ остается фиксированным, функция надежности, или вероятность безотказной работы на интервале от *0* до t, равна:

$$R(t,\tau)=\exp\{C[E_t/I_t-\varepsilon_c(\tau)]\cdot t\}, \qquad (11)$$

$$t_{cp}=\frac{1}{C\left[E_t/I_t-\varepsilon_c(\tau)\right]}. \qquad (12)$$

Нам необходимо найти начальное значение ошибок E_t и коэффициент пропорциональности - C. В процессе тестирования собирается информация о времени и количестве ошибок на каждом прогоне, т.е. общее время тестирования τ складывается из времени каждого прогона

$$\tau=\tau_1+\tau_2+\tau_3+...+\tau_n. \qquad (13)$$

Предполагая, что интенсивность появления ошибок постоянна и равна λ, можно вычислить ее как число ошибок в единицу времени

$$\lambda=\frac{\sum_{i=1}^{k}A_i}{\tau}, \qquad (14)$$

где A_i - количество ошибок на i-м прогоне.

Имея данные для двух различных моментов тестирования τ_a и τ_b, которые выбираются произвольно с учетом требования, чтобы $\varepsilon_c(\tau_b)>\varepsilon_c(\tau_a)$, можно сопоставить уравнения (12) и (15) при τ_a и τ_b.

$$t_{cp}=\frac{\tau}{\sum_{i=1}^{k}A_i}, \qquad (15)$$

$$\frac{1}{\lambda(\tau_a)}=\frac{1}{C[E_t/I_t-\varepsilon_c(\tau_a)]}, \qquad (16)$$

$$\frac{1}{\lambda(\tau_b)}=\frac{1}{C[E_t/I_t-\varepsilon_c(\tau_b)]}. \qquad (17)$$

Тогда

$$E_t=\frac{I_t[\lambda(\tau_b)/\lambda(\tau_a)\cdot\varepsilon_c(\tau_a)-\varepsilon_c(\tau_b)]}{(\lambda(\tau_b)/\lambda(\tau_a))-1}, \qquad (18)$$

$$C=\frac{\lambda(\tau_a)}{[E_t-\varepsilon_c(\tau_a)]}. \qquad (19)$$

Получив неизвестные E_t и C, можно рассчитать надежность программы по формуле (11).

Проведем расчеты применительно к учебной программе.

Например, в программе имеется $I_t = 4381$ оператор. В процессе последовательных тестовых прогонов были получены следующие данные:

№ прогона	1	2A	3	4	5	6	7	8B	9	10
Кол-во ошибок	1	2	1	1	1	1	1	2	1	1
Время (м)	5	8	2	1	5	1	1	2	5	5

Выберем две точки, исходя из требования, чтобы число ошибок, найденных на интервале $A \div B$, было больше, чем на интервале $0 \div A$ За точку A возьмем 2 прогон, а за точку B - 8 прогон. Тогда ошибки, найденные на этапах тестирования на интервалах $0 \div A$ и $A \div B$, будут равны соответственно:

$$\varepsilon_c(\tau_A) = 3/4381 = 0{,}0007$$

$$\varepsilon_c(\tau_B) = 7/4381 = 0{,}0015.$$

Время тестирования на интервалах равно: $\tau_A = 13, \quad \tau_B = 12$.

Рассчитаем интенсивности появления ошибок на двух интервалах:

$$\lambda(\tau_A) = 3/13 = 0{,}23,$$

$$\lambda(\tau_B) = 7/12 = 0{,}58.$$

Тогда число имеющихся до начала тестирования ошибок равно:

$$E_t = \frac{4381(0{,}58/0{,}23 \cdot 0{,}0007 - 0{,}0015)}{0{,}58/0{,}23 - 1} = 0{,}763 \approx 1 \; \textit{ошибка}.$$

$$C = \frac{0{,}23}{0{,}763 - 0{,}0007} = 0{,}301.$$

Рассчитаем вероятность безотказной работы в течение времени t при $\tau = 35$ *мин*. Возьмем $t = 60$ *мин*

$$R(60{,}35) = \exp\{0{,}301 \cdot [0{,}0002 - 0{,}0027] \cdot 60\} \approx 0{,}955,$$

$$R(t{,}35) = \exp\{0{,}301 \cdot [0{,}763/4381 - \varepsilon_c(35)] \cdot t\}.$$

Таким образом, надежность безотказной работы достаточно велика и вероятность сбоев и возникновения ошибок небольшая.

3.3 Оценка технико-экономических показателей разработки программных средств

В практической работе необходимо определить следующие показатели:

- объём программного средства (в операторах языка или строках текста) - O;
- длительность разработки (по фактическому времени) - $Д$;

$Д = Д_1 - Д_2$,

где $Д_1$ - дата начала разработки технического задания на ПС,

$Д_2$ - дата сдачи ПС;

- число программных и информационных модулей в ПС - P;
- количество фактически затраченного времени на разработку ПС - M;
- трудоемкость разработки ПС (по фактически затраченному времени по стадиям разработки) - T;
- абсолютное снижение трудовых затрат - TA

$TA = T_0 - T_1$,

где T_0 - трудовые затраты на решение транспортной задачи по базовому варианту (вручную, на данных, представленных преподавателем);

T_1 - трудовые затраты на решение транспортной задачи по предлагаемому варианту;

- коэффициент относительного снижения трудовых затрат - $КТ$

$КТ = TA/T_0 \cdot 100$;

- индекс снижения трудовых затрат или повышение производительности труда - $УТ$

$УТ = T_0/T_1$;

- абсолютное снижение стоимостных затрат - $СА$

$СА = C_0 - C_1$,

где C_0 - стоимостные затраты на решение транспортной задачи по базовому варианту,

C_1 - стоимостные затраты на решение транспортной задачи по предлагаемому варианту;

- коэффициент относительного снижения стоимостных затрат - $КС$

$КС = СА/C_0 \cdot 100$,

- индекс снижения стоимости затрат - $УС$

$УС = C_0/C_1$,

- срок окупаемости ПС - $К$

$К = КТ/СА$,

где $КТ$ - затраты на разработку и внедрение программного средства.

4 Методические рекомендации по оформлению практического занятия

Разработанное программное обеспечение должно быть оформлено стандартным образом, и содержать разделы в соответствии с приведенным ниже перечнем.

1. Титульный лист.

2. Содержание.

3. Задание на выполнение лабораторной работы.

4. Описание программы

4.1. Общие сведения
4.2. Логическая структура программного комплекса
4.3. Входные данные
4.4. Выходные данные
4.5. Вызов и загрузка

5. Тестирование программы

5.1. Программа проведения тестирования
5.2. Набор тестовых данных
5.3. Анализ результатов тестирования

6. Оценка разработанной программы

6.1. Выбор и обоснование методов оценки
6.2. Расчет показателей качества
6.3. Расчет показателей надежности
6.4. Оценка технико-экономических показателей

7. Список литературы.

8. Приложения

8.1. Распечатки экранов

8.2. Текст программы

8.3. Результаты тестирования и выполнения программы

8.4. Заставка.

Каждый раздел должен содержать только ту информацию, которая требуется. При возникновении дублирования допускается ссылка по данному вопросу на другой раздел отчета.

Раздел «Входные данные» содержит описание входных данных, форму их подготовки для ввода. Раздел не должен содержать конкретных данных выполнения программы.

Раздел «Выходные данные» должен иллюстрировать форму выдачи результатов работы программы. Выходными данными являются, кроме результатов решения задачи, все сообщения программы.

При оценке количественных показателей надежности обязательно:

□ Задаться требуемой наработкой на отказ;

□ Рассчитать текущую величину наработки на отказ и, если она меньше заданной, определить дополнительное возможное число отказов и время тестирования, необходимое, чтобы достичь требуемого уровня;

□ Определить возможное количество оставшихся в программе ошибок;

□ Определить уровень надежности разработанной программы;

Заставка к программе должна содержать информацию об авторе программы, компьютере и системе программирования, преподавателе и дате выполнения.

При оформлении практической работы необходимо учитывать:

□ Названия параграфов должны соответствовать их наименованию, указанному в содержании;

□ Подчеркивание наименований параграфов и др. не допускаются;

□ Все страницы работы должны быть пронумерованы последовательно арабскими цифрами и соответствовать проставленным страницам в содержании;

□ Практическая работа должна быть жестко скреплена:

□ Не допускается наличие элементов оформления в карандаше.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Определение жизненного цикла продукции.
2. Состав жизненного цикла продукции.
3. Какова цель тестирования программного средства?
4. Каким образом осуществляется детерминированное тестирование?
5. Каким образом осуществляется стохастическое тестирование?
6. В чем заключается стратегия «черного ящика»?
7. Каким образом осуществляется оценка надежности программных средств по модели Коркорэна?
8. Каким образом осуществляется оценка надежности программных средств по модели Шумана?
9. Каким образом осуществляется оценка технико – экономических показателей разработки программных средств?
10. Каким образом осуществляется оценка показателей качества программных средств?
11. Для чего и каким образом осуществляется сертификация научно - технической продукции?

Список рекомендуемой литературы

1. С.А. Орлов. Технологии разработки программного обеспечения: Учебник. – СПб: Питер, 2002. – 464 с.
2. Липаев В.В. Управление разработкой программных средств: Методы, стандарты, технология. – М.: Финансы и статистика, 1993.
3. Липаев В.В. Тестирование программ. – М.: Радио и связь, 1986.
4. Липаев В.В., Позин Б.А., Штрик А.А. Технология сборочного программирования. – М.: Радио и связь, 1992.
5. Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством. – М.: Изд-во стандартов, 1990.
6. Лифиц И.М. Стандартизация, метрология и сертификация. М.: Юрайт-Издат, 2004. 335 с.
7. Александровская Л.Н., Аронов И.З., Смирнов В.В., Шолом А.М. Сертификация сложных технических систем. М.: Логос, 2001. 312 с.