УДК 681.3

С.Н. Звигляныч, Н.П. Изюмский

Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков

ТЕСТОВЫЙ МОНИТОР ДЛЯ УПРАВЛЯЮЩИХ ПРОГРАММ

Предложен метод построения тестового монитора для проведения тестирования управляющих программ, функционирующих в контуре управления сложных технических систем. Вводятся показатели, позволяющие провести сравнительную оценку применяемых тестов.

Ключевые слова: *программное средство, надежность программного обеспечения, тестирование программ.*

Введение

Постановка проблемы. Под отказом программного средства (ПС) будем понимать такое состояние вычислительного процесса, при котором система прекращает выполнение своих функций. При этом надежность ПС выступает как его свойство работать без отказов в течение определенного периода времени, и выражается через вероятность этого события [1]. Следует отметить, что надежность не является внутренним свойством программы, она во многом связана с тем, как программа используется [1].

Проблема надежности ПС имеет два аспекта: во-первых, обеспечение надежности как таковой при его разработке; во-вторых, проведение оценки (измерений) надежности при разработке, отладке и функционировании ПС.

Как правило, большинство монографий, посвященных вопросам надежности ПС, затрагивают в основном первый аспект данной проблемы. Но, прежде чем улучшить какое-либо свойство ПС, необходимо обеспечить возможность измерения характеристик, описывающих это свойство.

Поэтому рассмотрение методов измерения характеристик ПС является на сегодняшний день актуальной задачей.

Анализ литературы. В большинстве монографий, как отмечалось выше, в основном уделяют внимание технологиям проектирования надежного ПС [1, 2].

Вопросы же измерения некоторых параметров ПС, позволяющих количественно оценить его надежность, при решении задач остаются в стороне.

Целью статьи является обоснование метода построения тестового монитора, позволяющего количественно оценить ряд параметров разрабатываемого ПС.

Основной материал

Будем рассматривать ПС, которые представляются программами, работающими в контуре управления сложных технических систем.

В самом общем виде алгоритм таких программ (рис. 1) представим как последовательное выполнение команд блока ввода исходных данных (ИД), поступающих от ряда датчиков (органов контроля), команд блока реализации алгоритма управления, команд блока формирования и выдачи на исполнительные элементы вектора управляющих воздействий.

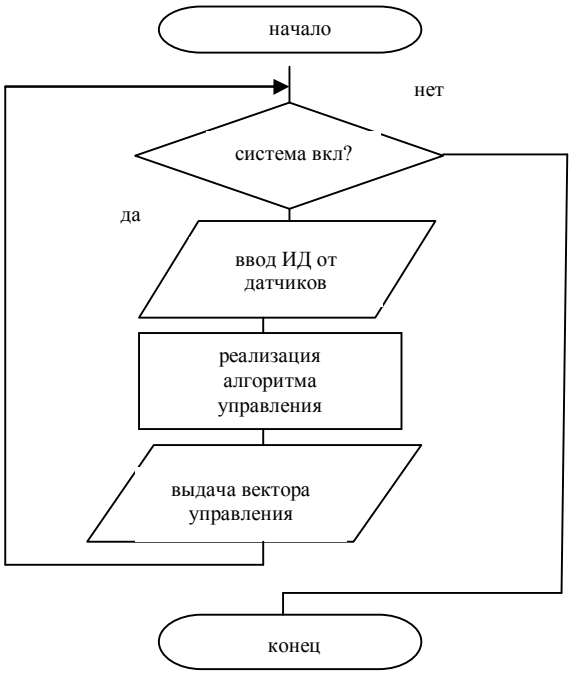

Рис. 1. Блок-схема обобщенного алгоритма

Этап тестирования программы является одним из важнейших этапов ее жизненного цикла.

Тест представляется набором входных данных, значения которых соответствуют значениям контролируемых параметров при протекании реальных процессов функционирования рассматриваемых технических систем.

Зададимся вопросом, как различить тесты между собой, или какие произвести измерения, чтобы количественно оценить тест?

В любой программе рассматриваемого типа можно выделить линейные участки (ЛУ), где выполнение программы происходит от команды к команде последовательно без переходов.

Для ассемблерных программ границами таких участков будут служить метки.

Для программ, разработанных в системах программирования, реализующих языки высокого уровня (для примера С++), такие ЛУ будут соответствовать организованным циклам (параметрическим “for …”, с верхней проверкой “while …”, с нижней проверкой “do … while”), блокам, организованными операторами условий “if … else”, операторами множественного доступа “switch”.

Представим программу в виде графа, где вершины ЛУ, а дуги – переходы, отражающие алгоритм выполнения.

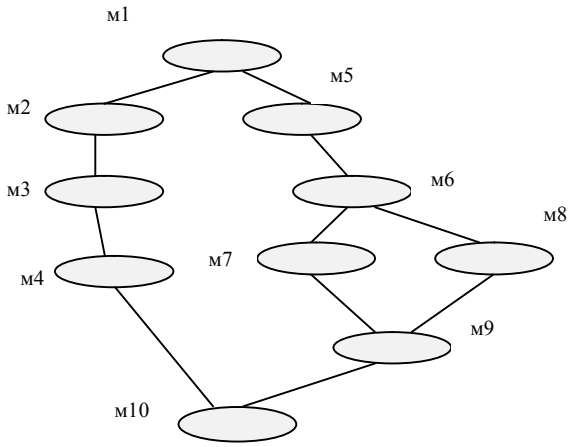

Рис. 2. Граф ассемблерной программы

ЛУ различаются между собой количеством операторов (команд) и необходимым временем для их выполнения.

Выполнение тех или иных ЛУ в ходе реализации программы зависит от конкретного вектора исходных данных.

Для анализа процесса реализации программы желательно получить последовательность выполняемых ЛУ. Назовем такую последовательность трассой выполнения программы.

Получить такую трассу можно, выполнив предварительный шаг перед трансляцией программы, то есть, реализовать своего рода препроцессорную обработку программы.

Суть такой обработки заключается во вставке перед каждым ЛУ операторов (команд), выводящих информацию о начале выполнения данного ЛУ в создаваемый для этих целей специальный служебный файл. Также, используя службу времени операционной системы, в запись этого файла можно внести и время начала выполнения ЛУ.

После исполнения такой “преобразованной” программы, анализ созданного служебного файла позволяет построить трассу выполнения, то есть, проследить последовательность исполнения ЛУ и затраченное на это время.

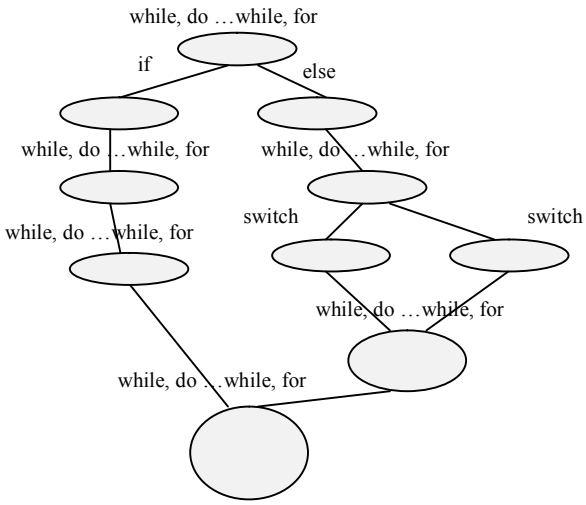

Рис. 3. Граф программы на языке С++

Назовем такую программу, выполняющую препроцессорную обработку, тестовым монитором.

Рассмотрим несколько вариантов вектора исходных данных.

Каждому варианту соответствует своя трасса выполнения. Трассы выполнения отличаются числом пройденных ЛУ и затраченным на это временем.

Пусть N – общее число ЛУ в программе, а L_i – число ЛУ трассы выполнения i-го теста. Введем в рассмотрение коэффициент полноты теста и определим его как

$$kp_i = \frac{L_i}{N}. \qquad (1)$$

Тест, имеющий больший коэффициент полноты, обеспечивает проверку большего число ЛУ. Поэтому вполне естественно потребовать использовать для тестирования такие варианты вектора исходных данных, которые обеспечивали бы максимальное значение kp.

Очень часто знание времени выполнения того или иного участка программы позволяет найти в ней “узкие” места и оптимизировать алгоритм.

Если в служебном файле есть отметка о начале выполнения каждого j-го ЛУ трассы, можно ввести коэффициент нагрузки как

$$kn_j = \frac{t_j}{T_o}, \qquad (2)$$

где t_j - время выполнения j-го ЛУ;

T_o – общее время выполнения теста.

Данный коэффициент как раз и может позволить выявить наиболее критичные по времени выполнения ЛУ тестируемой программы.

Для управляющих программ в качестве ИД выступают значения измеряемых параметров. Как пра-

вило, это случайные величины с нормальным законом распределения.

Если пренебречь вероятностями менее 0.01, можно считать практически достоверным, что случайная величина, подчиненная нормальному закону, отклонится от центра рассеивания m не более чем на четыре вероятных отклонений E [3].

Допустим, что управление производится по двум параметрам. Тогда область достоверных ИД представляется заштрихованной областью на рис. 4. Соответственно при n параметрах данная область будет представляться неким n-мерным пространством.

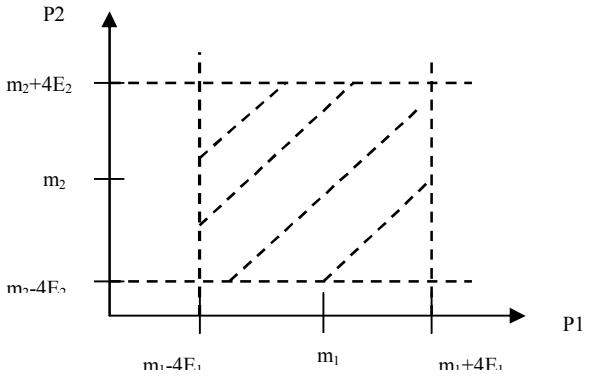

Рис. 4. Достоверная область ИД

При тестировании необходимо, прежде всего, использовать наиболее вероятные значения ИД, то есть, ИД для тестов должны быть из данного n-мерного пространства.

Пусть имеются результаты нескольких тестов. Тогда появляется возможность подсчитать число ЛУ, имеющих не менее одного прохода по результатам всех тестов. Обозначим их как LTST.

Зная общее число ЛУ в программе N, введем суммарный коэффициент полноты тестирования как

$$KPS = \frac{L_{TST}}{N}. \qquad (3)$$

Используя данный коэффициент, определим требование к процессу тестирования, введя критерий полноты тестирования, а именно, для обеспечения требуемой надежности разрабатываемого ПС общее количество проводимых тестов должно обеспечить значение KPS не ниже требуемого.

$$KPS \geq KPS_{тр}. \qquad (4)$$

Требуемое значение коэффициента полноты тестирования в большей степени определяется опытом персонала, проводящего тестирование, и местом разрабатываемого ПС в контуре управления.

Выводы

Отметим, что при тестировании проведение трассировки само по себе не дает однозначного ответа относительно оптимального варианта алгоритма разрабатываемого ПС, его надежности. Но полученные количественные оценки процесса выполнения тестируемой программы позволяют, во-первых, определить тенденции, направленные на улучшение самого алгоритма как такового, а, во-вторых, обосновано очертить ту область исходных данных, которая обеспечивает требуемую надежность разрабатываемого ПС.

Список литературы

1. Майерс Г. Надежность программного обеспечения: Пер. с англ. под ред. В.Ш. Кауфмана / Г. Майерс. – М.: Мир, 1980.

2. Зелковиц М. Принципы разработки программного обеспечения: пер. с англ. / М. Зелковиц, А. Шоу, Дж. Гэннон. – М.: Мир, 1982.

3. Вентцель Е.С. Теория вероятностей / Е.С. Вентцель. – М.: Государственное издательство физико-математической литературы, 1958. – 463 с.

Поступила в редколлегию 1.03.2012

Рецензент: д-р техн. наук, проф. И.И. Обод, Национальный технический университет «ХПИ», Харьков.

ТЕСТОВИЙ МОНІТОР ДЛЯ УПРАВЛЯЮЧИХ ПРОГРАМ

С.М. Звиглянич, М.П Ізюмський

Запропонований метод побудови тестового монітора для проведення тестування управляючих програм, що функціонують в контурі управління складних технічних систем. Вводяться показники, що дозволяють провести порівняльну оцінку вживаних тестів.

Ключові слова: *програмний засіб, надійність програмного забезпечення, тестування програм.*

THE TEST MONITOR FOR CONTROL PROGRAMS

S.N. Zviglyanich, N.P. Izyumskiy

The article considers method for development test monitor for checking control programs in the complex technical systems control loop and parameters for comparative assessment appropriated tests.

Keywords: *software, software reliability, program testing.*