Последствия и признаки появления ошибок в программе

Следствием появления ошибки в программе является ее отказ, заключающийся в отклонении от выполнения программой заданных функций. В зависимости от степени серьезности последствий ошибок (отказов) в программе эти отклонения можно разделить следующим образом:

– полное прекращение выполнения функций на длительное или неопределенное время;

– кратковременное нарушение хода вычислительного процесса.

Степень серьезности последствий ошибок в программе может быть оценена соотношением между длительностью восстановительных работ, которые необходимо произвести после отказа в программе, и динамическими характеристиками объектов, использующих результаты работы программных средств. К таким характеристикам объектов относятся, например, инерционность объектов, выступающих в качестве источников и потребителей информации; заданная частота решения задач обработки информации; заданное время реакции вычислительной системы на запросы пользователей и др.

Наиболее типичными симптомами появления ошибок в программе являются:

- преждевременное (аварийное) окончание выполнения программы;
- недопустимое увеличение времени выполнения программы;
- зацикливание ЭВМ на выполнении некоторой последовательности команд одной из программ;

- полная потеря или значительное искажение накопленных данных, необходимых для успешного выполнения решаемых задач;
- нарушение последовательности вызова отдельных программ, в результате чего происходит пропуск необходимых программ либо непредусмотренное обращение к программам;
- искажение отдельных элементов данных (входных, выходных, промежуточных) в результате обработки искаженной исходной информации.

Аварийное завершение прикладных программ, как правило, легко идентифицируется, так как операционные системы обеспечивают возможность выдачи сообщений, содержащих соответствующий аварийный код. Типичными причинами появления кодов аварийного завершения являются ошибки при выполнении макрокоманды; неверное использование методов доступа; нарушение защиты памяти; нехватка ресурсов памяти; неверное использование макрокоманды; возникновение программных прерываний, для которых не указан обработчик, и др.

При появлении подобных ошибок после анализа аварийного кода имеется принципиальная возможность немедленного повторения запуска прикладной программы. Для увеличения эффективности восстановительных процедур необходимо:

- предусмотреть в программах специальные средства диагностики кодов аварийных завершений, в том числе кодов, формируемых самими пользователями;
- ввести в программы контрольные точки;
- обеспечить возможности рестарта программ с контрольных точек.

►Содержание►

Аналитические модели надежности программ

Аналитические модели надежности дают возможность исследовать закономерности проявления ошибок в программах, а также прогнозировать надежность при разработке и эксплуатации. Модели надежности программ строятся на предположении, что проявление ошибки является случайным событием и поэтому имеет вероятностный характер. Такие модели предназначены для оценки показателей надежности программ и программных комплексов в процессе тестирования: числа ошибок, оставшихся не выявленными; времени, необходимого для выявления очередной ошибки в процессе эксплуатации программы; времени, необходимого для выявления всех ошибок с заданной вероятностью и т. д. Модели дают возможность принять обоснованное решение о времени прекращения отладочных работ.

При построении моделей используются следующие характеристики надежности программы.

Функция надежности $P(t)$, определяемая как вероятность того, что ошибки программы не проявятся на интервале времени от **0** до t, т. е. время ее безотказной работы будет больше t.

Функция ненадежности $Q(t)$ – вероятность того, что в течение времени t произойдет отказ программы как результат проявления действия ошибки в программе. Таким образом,

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

Интенсивность отказов $\lambda(t)$ – условная плотность вероятности времени до возникновения отказа программы при условии, что до момента t отказа не было. Когда мы рассматривали потоки отказов и сбоев, мы показали

$$\lambda(t) = -\frac{dP(t)}{dt} / P(t) = \left|\frac{dQ(t)}{dt}\right| / P(t)$$

Средняя наработка на отказ $\overline{T_O}$ – математическое ожидание временного интервала между последовательными отказами.

В настоящее время основными типами применяемых моделей надежности программ являются модели, основанные на предположении о дискретном изменении характеристик надежности программ в моменты устранения ошибок, и модели, основанные на экспоненциальном характере изменения числа ошибок в зависимости от времени тестирования и функционирования программы.

*Модель надежности программ с дискретно-понижающейся частотой (интенсивностью) проявления ошибок***.**

В этой модели предполагается, что интенсивность обнаружения ошибок описывается кусочно-постоянной функцией, пропорциональной числу не устраненных ошибок. Другими словами, предполагается, что интенсивность отказов $\lambda(t)$ постоянна до обнаружения и исправления ошибки, после чего она опять становится константой, но с другим, меньшим значением. При этом предполагается, что между $\lambda(t)$ и числом оставшихся в программе ошибок существует прямая зависимость:

$$\lambda(t) = K(M - i) = \lambda_i,$$

где M – неизвестное первоначальное число ошибок; i – число обнаруженных ошибок, зависящее от времени t, K – некоторая константа (рис.2).

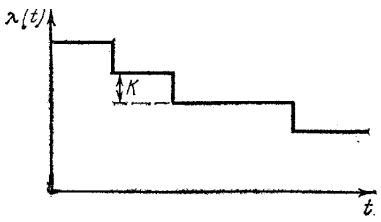

Рис.2 Зависимость интенсивности отказов программы от времени работы (модель с дискретно понижающейся интенсивностью проявления ошибок в программе)

► Содержание ►

Плотность распределения времени обнаружения i-й ошибки t_i задается соотношением

$$f(t_i) = \lambda_i e^{-\lambda_i t_i}.$$

Значение неизвестных параметров K и M может быть оценено на основании последовательности наблюдений интервалов между моментами обнаружения ошибок по методу максимального правдоподобия. При этом для нахождения оценок параметров K и M необходимо решить следующие уравнения:

$$\sum_{i=1}^{m} (\overline{M} - 1)^{-1} = m/(\overline{M} - \theta_m);$$

$$\overline{K} = (m/A)/(\overline{M} - \theta_m),$$

где $\theta_m = B/Am$; $A = \sum_{i=1}^{m} t_i$; $B = \sum_{i=1}^{m} i\,t_i$; $\overline{K}$ и $\overline{M}$ – оценки соответственно K и M, m – число устраненных ошибок в момент оценки надежности программ.

Рассмотренная модель надежности программ является достаточно грубой. На практике часто не соблюдаются условия, на которых она построена. Нередко при устранении ошибки вносятся новые ошибки. Во многих случаях не соблюдается также основное предположение, что при всяком устранении ошибки интенсивность отказов уменьшается на одну и ту же величину K. Не всегда удается определить и устранить причину отказа, и программу часто продолжают использовать, так как при других исходных данных ошибка, вызвавшая отказ, может себя и не проявить.

Модель надежности программ с дискретным увеличением времени наработки на отказ.

Данная модель надежности программ построена на предположении, что устранение ошибки в программе приводит к увеличению времени наработки на отказ на одну и ту же случайную величину.

Предполагается, что время между двумя последовательными отказами $T^{(i)}$ является случайной величиной, которую можно представить в виде суммы двух случайных величин:

$$T^{(i)} = T^{(i-1)} + \Delta T^{(i)},$$

где случайные величины $\Delta T^{(i)}$ независимы и имеют одинаковые математические ожидания $M[\Delta T]$ и средние квадратические отклонения $\sigma_{\Delta T}$.

Из выше приведенной формулы следует, что m-й отказ программы произойдет через время

$$T_m = \sum_{i=0}^{m} \left[T^{(0)} + \sum_{j=1}^{i} \Delta T^{(j)} \right].$$

▼Содержание▼

Предполагается также, что $T^{(0)} << \sum_{j=1}^{i} \Delta T^{(j)}$. Основанием для такого предположения является то, что отказы программы в начальном периоде эксплуатации возникают часто. В этом случае можно считать, что

$$T^{(0)} \approx \Delta T^{(0)}.$$

При этих предположениях средняя наработка между (m—1)-м иm-м отказами программы равна

$$\overline{T}^{(m)} = M(T^{(m)}) = mM[\Delta T],$$

а средняя наработка до возникновения m -го отказа определяется выражением

$$\overline{T}_m = M(T_m) = \frac{m(m+1)}{2} M[\Delta T].$$

Оценки величин $M[\Delta T], \sigma_{\Delta T}$ могут быть получены по данным об отказах программы в течение периода наблюдения t_H следующим образом:

$$\overline{M}[\Delta T] = \frac{1}{m_H} \sum_{i=1}^{m_H} \left[t^{(i)} - t^{(i-1)} \right] = \frac{1}{m_H} \sum_{i=1}^{m_H} \Delta t^{(i)};$$

$$\overline{\sigma}_{\Delta t}^2 = \frac{1}{m_H - 1} \sum_{i=0}^{m_H} (\Delta t^{(i)} - M[\Delta T])^2,$$

где m_H – число отказов программы за период $(0, t_H)$; $t^{(i)}$ – момент возникновения i *-го* отказа программы.

Функция надежности определяется в зависимости от числа возникших отказов

$$P_m(t) = \frac{1}{2} - \Phi\left[\frac{t - mM[\Delta T]}{\sigma_{\Delta T} \sqrt{m}} \right],$$

где ***Φ(x)*** —функция Лапласа.

*Экспоненциальная модель надежности программ***.**

Эта модель основана на предположении об экспоненциальном характере изменения во времени числа ошибок в программе.

В этой модели прогнозируется надежность программы на основе данных, полученных во время тестирования. В модели вводится суммарное время функционирования τ, которое отсчитывается от момента начала тестирования программы (с устранением обнаруженных ошибок) до контрольного момента, когда производится оценка надежности.

Предполагается, что все ошибки в программе независимы и проявляются в случайные моменты времени с постоянной средней интенсивностью в течение всего времени выполнения программы. Это означает, что число ошибок, имеющихся в программе в данный момент, имеет пуассоновское распределение, а

► Содержание ►

временной интервал между двумя отказами распределен по экспоненциальному закону, параметр которого изменяется после исправления ошибки.

Следует отметить, что характер закономерностей в этой модели остается таким же, как и в рассмотренных ранее, а именно интенсивность отказов предполагается пропорциональной числу оставшихся ошибок. Основное отличие данной модели от предыдущих состоит в том, что интенсивность отказов предполагается непрерывной функцией. Это упрощает математическое описание модели, а модель приобретает дополнительную гибкость.

Пусть ***M*** – число ошибок, имеющихся в программе перед фазой тестирования (***M*** рассматривается как некоторая константа); $m(\tau)$ – конечное число исправленных ошибок; $m_0(\tau)$ – число оставшихся ошибок. Тогда

$$m_0(\tau) = M - m(\tau).$$

При принятых предположениях интенсивность отказов пропорциональна $m_0(\tau)$, т.е.

$$\lambda(\tau) = Cm_0(\tau),$$

где ***C*** – коэффициент пропорциональности, учитывающий реальное быстродействие ЭВМ и число команд в программе.

Введем дополнительное предположение, что в процессе корректировки новые ошибки не порождаются, т. е. что интенсивность исправления ошибок dm/dt будет равна интенсивности их обнаружения, т. е.

$$dm/d\tau = \lambda(\tau).$$

Решая совместно два последних уравнения, получаем

$$(dm/d\tau) + Cm = CM.$$

Перед началом работы ЭВМ $(t = 0)$ ни одна ошибка исправлена не была $(\tau = 0)$, поэтому решением этого уравнения является

$$m = M[1 - \exp(-C\tau)].$$

Будем характеризовать надежность программы после тестирования в течение времени τ средним временем наработки на отказ:

$$\overline{T}_0 = 1/\lambda(\tau).$$

Следовательно,

$$\overline{T}_0 = (1/CM)\exp(C\tau).$$

Введем $\overline{T}_0$ – исходное значение среднего времени наработки на отказ перед тестированием. Тогда

$$\overline{T}_0 = 1/CM.$$

► Содержание ►

В результате имеем

$$\overline{T_0} = \overline{T_0'} \exp(\tau / (M \overline{T_0'}))$$

Очевидно, что среднее время наработки на отказ увеличивается по мере выявления и исправления ошибок.

Рассмотрим более общий случай, когда в процессе корректировки могут появляться новые ошибки. Пусть ***B*** – коэффициент уменьшения ошибок, определяемый как отношение интенсивности уменьшения ошибок к интенсивности их проявления, т. е. к интенсивности отказов.

Пусть ***п***—число обнаруженных отказов, а N_0 – число отказов, которые должны произойти, чтобы можно было выявить и устранить***m*** соответствующих ошибок, т. е.

$$n = m / B, \; N_0 = M / B.$$

В этом случае среднее время наработки на отказ $\overline{T_0}$ и число обнаруженных отказов***п*** определяются следующими соотношениями:

$$\overline{T_0} = (1 / BCN_0) \exp(C\tau);$$

$$n = N_0 \left[1 - \exp(-C\tau)\right].$$

Тогда для значения среднего времени наработки на отказ перед тестированием справедливо

$$\overline{T_0'} = 1 / BCN_0.$$

В результате получаем

$$\overline{T_0} = \overline{T_0'} \exp(\tau / BN_0 \overline{T_0'});$$

$$n = N_0 \left[1 - \exp(-\tau / BN_0 \overline{T_0'})\right].$$

Для практического использования представляет интерес число ошибок Δn, которое должно быть обнаружено и исправлено для того, чтобы добиться увеличения среднего времени наработки на отказ от $\overline{T_{01}}$ до $\overline{T_{02}}$. Этот показатель может быть получен из следующих соотношений:

$$n = N_0 (1 - \overline{T_0'} / \overline{T_0})$$

$$\Delta n = n_2 - n_1.$$

Следовательно,

$$\Delta n = N_0 \overline{T_0'} [1 / \overline{T_{01}} - 1 / \overline{T_{02}}].$$

Дополнительное время работы, необходимое для обеспечения увеличения среднего времени наработки на отказ с $\overline{T_{01}}$ до $\overline{T_{02}}$, определяется как

► Содержание ►

$$\Delta\overline{T} = N_0\overline{T}_0 \ln(\overline{T}_{02} / \overline{T}_{01}).$$

Рассмотрим, каким образом можно оценить основные параметры модели. Коэффициент ***B*** может быть определен исходя из статистических данных по числу ошибок, порожденных во время обнаружения других ошибок. Можно считать, что для систем программного обеспечения общего назначения ***B*** изменяется в диапазоне 0,91—0,95.

Для оценки M и M_0 можно рассчитать среднюю интенсивность ошибок (число ошибок на одну команду) в начале различных фаз тестирования. По различным оценкам в начале процесса тестирования для программ, написанных на языке ассемблера, число ошибок на 1000 команд варьируется от 4 до 8. Рассмотренная модель может применяться для определения времени испытаний программ с целью достижения заданного уровня надежности, а также для оценки числа оставшихся в программе ошибок.

Вторым существенным составляющим стохастического функционального тестирования является *проверка правильности результатов вычислений* по генерированным случайным исходным данным.

► Содержание ►

Проверка правильности может осуществляться путем проверки соответствия эталону; принадлежности области; по времени выполнения; сравнения с другими (соседними) значениями; через достижения цели (в замкнутом контуре управления).

Наиболее просто проверка правильности осуществляется через сравнение с эталоном. Эталоном могут являться вычисления, выполняемые по другой, эквивалентной программе или другому алгоритму. Например, если разработан улучшенный по быстродействию вариант программы, то эталоном может быть исходная программа.

Информационные методы повышения

надежности СВТ

Широко распространенным методом повышения надежности СВТ является обеспечение избыточности в составе СВТ, в частности информационной избыточности.

Применение корректирующих кодов является одним из наиболее удобных и гибких методов введения избыточности в ЭВМ. В некоторых условиях, например, при кратковременных сеансах работы, когда ресурсы надежности почти не расходуются, коррекция сбоев, вызываемых различного рода помехами, может иметь решающее значение для обеспечения нормального функционирования. Положительные качества корректирующих кодов следующие:

1. корректирующие коды обеспечивают исправление ошибок без перерывов в работе ЭВМ;

2. способ кодирования и применяемый код выбираются в зависимости от алгоритма функционирования данного вычислительного устройства, что дает возможность согласования корректирующей способности кода со статистическими характеристиками потока ошибок устройства и уменьшения избыточности, требуемой для коррекции ошибок;

3. использование корректирующих кодов позволяет учесть необходимость устранения влияния ошибок в устройстве последовательно на всех этапах проектирования, начиная с алгоритма функционирования.

Избыточность может быть временной и пространственной. Временная избыточность связана с увеличением времени решения задачи (в частном случае процесс решения задачи может быть осуществлен дважды) и вводится программным путем, являясь основой программного способа обнаружения и исправления ошибок.

Пространственная избыточность заключается в удлинении кодов чисел, в которые вводятся дополнительные (контрольные) разряда. Идея обнаружения и исправления ошибок с использованием избыточности состоит в следующем. Все множество N_y выходных слов устройства разбивают на подмножество разрешенных кодовых слов N_{yo}, т.е. таких слов, которые могут появиться в результате правильного выполнения логических и арифметических операций, и подмножество запрещенных кодовых слов $(N_y - N_{yo})$, т.е. таких слов, которые могут появиться только в результате ошибки.

Появившееся на выходе устройства слово y подвергают анализу. Если оно относится к подмножеству разрешенных слов, то оно считается правильным и декодирующее устройство переводит его в соответствующее выходное слово yo_i. Если же слово y оказывается элементом подмножества запрещенных слов, то это свидетельствует о наличии ошибки.

Для исправления обнаруженных ошибок запрещенные кодовые слова разбиваются на группы. При этом каждому разрешенному кодовому слову соответствует одна такая группа. При декодировании обнаруженное на выходе устройства запрещенное кодовое слово заменяется разрешенным словом, в группу которого оно входит. Тем самым ошибка исправляется. Однако работа декодирующего устройства усложняется. Процесс построения корректирующего кода состоит из следующих этапов:

1 этап – выявление наиболее вероятных ошибок для заданного способа функционирования устройства или наиболее опасных ошибок в условиях использования этого устройства;

2 этап – формирование избыточного множества выходных слов, разделение этого множества на подмножества разрешенных и запрещенных кодовых слов и образование декодировочных групп;

3 этап – разработка рационального способа декодирования выходных слов, позволяющего реализовать относительно несложными техническими средствами

► Содержание ►

обнаружение и исправление ошибок;

4 этап – организация множества входных так, чтобы заданное преобразование, выполненное над любым словом этого множества, дало на выходе слово, принадлежащее к подмножеству разрешенных кодовых слов.

При построении ЭВМ, предназначенной для решения определенного класса задач, важным вопросом является рациональный выбор уровня, на котором следует применять корректирующий код.

Известно, что ЭВМ слагается из отдельных устройств, в каждом из которых информация претерпевает определенные изменения. В то же время составляющие части ЭВМ состоят из отдельных блоков (сумматоров, регистров и т.п.), которые в свою очередь набраны из простейших логических элементов (триггеров, схем И, ИЛИ, НЕ и т.п).

Корректирующие коды применяют на любом из этих уровней структуры ЭВМ. Однако результаты получают различные. В ряде случаев корректировать ошибки в работе логических схем значительно труднее, чем например, в работе сумматора в целом. Далее, если контроль правильности и исправления ошибок в работе отдельных блоков и устройств осуществить сложно, то в некоторых случаях прохождение всей задачи легко контролируется применением простейших принципов, например, путем повторения счета.

Введение структурной и информационной избыточности для повышения надежности ЭВМ не дает желаемых результатов, если при ее конструировании будут нерационально решены компоновка и конструктивное исполнение отдельных узлов и блоков.

Для обеспечения высокой надежности ЭВМ необходимо стремиться максимально упростить конструкцию, применять стандартные элементы, обеспечивать возможность проведения профилактики и контроля.

▶ Содержание ▶

<< < Предыдущая 1 2 3 4 5 6 7

Соседние файлы в предмете [НЕСОРТИРОВАННОЕ]

Материал для подготовки (математика 0).doc	11	#	778.24 Кб	11.03.2015
Материаловед.в ответах2.doc	1	#	8.74 Мб	01.03.2025
материаловедение лабороторная №1.docx	56	#	2.97 Мб	12.03.2015
материаловедение лабороторная №2.1.docx	71	#	13.15 Мб	12.03.2015
Материалы к тестам по философии.doc	0	#	230.4 Кб	01.03.2025
Материалы по ЭСВТ.doc	58	#	1.53 Мб	11.03.2015
Матлаб №1.doc	15	#	37.89 Кб	12.03.2015
матмод.doc	2	#	212.48 Кб	29.07.2019

маш и обор Техникум.doc	0	#	1.52 Мб	01.04.2025
маша шп. 1-9.docx	0	#	701.28 Кб	01.05.2025
машиностроение 4 сем.doc	0	#	434.69 Кб	01.07.2025