Бубнов В.П., Тырва А.В., Хомоненко А.Д.

ОБОСНОВАНИЕ СТРАТЕГИИ ОТЛАДКИ ПРОГРАММ НА ОСНОВЕ НЕСТАЦИОНАРНОЙ МОДЕЛИ НАДЕЖНОСТИ

Рассматривается модель роста надежности программ при отладке, описываемая с помощью нестационарной марковской системы обслуживания. Приводятся граф переходов между состояниями системы обслуживания, система дифференциальных уравнений, примеры расчета вероятностных характеристик. Предлагаются варианты использования модели для прогнозирования времени отладки программ с целью обоснования наилучшей стратегии. Коротко затрагиваются вопросы задания начальных условий и исходных данных как параметров модели.

Введение

Модели надежности программного обеспечения получили достаточно широкую проработку в научной литературе. При этом наибольшую известность получили модели Джелинского-Моранды, Мусы и Литлвуда. Систематическое описание этих и ряда других моделей надежности программ приводится в [1]. Авторами [2] описываются нетрадиционные методы оценки надежности информационных систем. В книге [3] рассматриваются модели поведения программ, в которых учитываются интервалы работоспособного и неисправного состояний программы.

Сама по себе оценка характеристик надежности программ представляет несомненный интерес. Однако следующий важный шаг, конечно, связан с выработкой практических рекомендаций по организации отладки программ, позволяющих улучшить характеристики процесса отладки. Здесь можно назвать работы по планированию испытаний сложных программных комплексов, например [4, 5]. В них делается существенное предположение о том, что характеристики надежности отдельных программных модулей известны.

Мы рассмотрим вопрос обоснования стратегии отладки (отдельного модуля) программы на основе использования модели роста надежности программы, аналогичной вероятностным моделям надежности программ, таким как, экспоненциальная модель Джелинского-Моранды, геометрическая модель, модели Мусы и других экспоненциальных моделей (см. [1]). Отметим, модели роста надежности программ часто используют вероятностные подходы и модели систем массового обслуживания с конечным и бесконечным числом каналов обслуживания и сетей массового обслуживания Джексона. К примеру, в работе [6] предлагается модель, основанная на марковской модели систем массового обслуживания, с решением задачи появления и устранения ошибок в программе как марковского процесса гибели и размножения с непрерывным временем и нахождением его характеристик. Недостатком названного решения является использование предположения о стационарности рассматриваемых процессов.

В основу предлагаемого решения положим модель нестационарной системы обслуживания, предложенную в [7]. В предлагаемой модели используется допущение об экспоненциальном распределении длительностей интервалов между обнаружением и исправлением последовательных ошибок. Однако, в отличие от названных выше моделей, не накладывается ограничений на вид зависимости интенсивностей обнаружения и устранения ошибок от номера ошибки (количества еще не исправленных ошибок в программе).

При соответствующей вероятностной интерпретации (аналогичной модели Джелинского-Моранды и подобным) модель позволяет описывать и рассчитывать характеристики надежности программы, а также обосновывать стратегию отладки программы. Модель в большей степени подходит для получения прогнозных оценок надежности отдельных

программных модулей, число ошибок в которых может колебаться от единиц до двух-трех десятков.

Важным вопросом при моделировании надежности программного обеспечения является выбор исходных данных. Распространенным подходом [1] является использование собранной в процессе разработки уже завершенных проектов статистики ошибок (интенсивности обнаружения ошибок при отладке, временные интервалы между моментами обнаружения ошибок). Однако, для учета особенностей конкретного проекта целесообразно также использовать его характеристики, влияющие на надежность – например, объектно-ориентированные метрики [8, 9, 10] (длина наибольшего пути иерархии наследования, взвешенное количество методов класса и др.). Метод планирования тестирования программ с использованием объектно-ориентированных метрик сложности предложен в [5].

Марковская нестационарная модель процесса отладки

Отладка программы представляет собой процесс, состоящий из двух этапов: тестирование (поиск ошибок) и устранение обнаруженных ошибок. Эти два типа деятельности, как правило, разделены и выполняются различными исполнителями (командами). Для упрощения будем считать, что при устранении ошибки новые ошибки не вносятся. Кроме того, считаем, что интенсивности обнаружения и исправления ошибок зависят от номера обнаруживаемой и устраняемой ошибки.

Предположение о зависимости интенсивности обнаружения и исправления ошибок от номера ошибки широко используется во многих вероятностных моделях роста надежности программного обеспечения. Такое предположение основано на том, что по мере обнаружения и устранения ошибок в программе число ошибок в программе остается все меньше и, соответственно интенсивность обнаружения ошибок уменьшается.

Предположение о зависимости интенсивности исправления ошибок от номера устраняемой ошибки в данном случае не является ограничением предлагаемой модели роста надежности программы, а расширяет ее возможности. В частности, вполне можно предположить, что при отладке конкретной программы, по мере приобретения опыта по устранению выявленных ошибок интенсивность исправления ошибок может увеличиваться. Кроме того, так как предлагаемая модель не накладывает ограничений на вид такой зависимости, можно рассматривать ее как обобщение разработанных ранее моделей.

Введем следующие обозначения:

N – общее число ошибок в программе;

λ_k – интенсивность обнаружения k-й ошибки при отладке программы, $k = \overline{1, N}$;

μ_k – интенсивность исправления k-й ошибки при отладке программы, $k = \overline{1, N}$. Вероятностная марковская модель процесса отладки программы описывается ниже.

На вход системы последовательно поступает N запросов на устранение обнаруженных ошибок. Распределения длительности интервалов между моментами обнаружения ошибок описываются экспоненциальными законами с интенсивностями $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$, зависящими от номера ошибки. Распределения времени исправления ошибок — экспоненциальные с интенсивностями $\{\mu_1, \mu_2, \ldots, \mu_N\}$, также зависящими от номера ошибки.

Представим такую систему цепью Маркова с дискретным множеством состояний и непрерывным временем. Состояния системы в каждый момент времени будем характеризовать парой (i, j), где i - число обнаруженных, но еще не исправленных ошибок $(i = \overline{0, N})$, а j – число уже исправленных ошибок $(j = \overline{0, N - i})$. Вероятности пребывания системы в этих

состояниях обозначим через $P_{i,j}(t)$. В таком представлении система будет иметь конечное число состояний. Обозначим ее как систему обслуживания типа M(i)/M(j)/N.

Диаграмма переходов между состояниями системы M(i)/M(j)/N [7], описывающей процесс отладки программ, приведена на рис. 1.

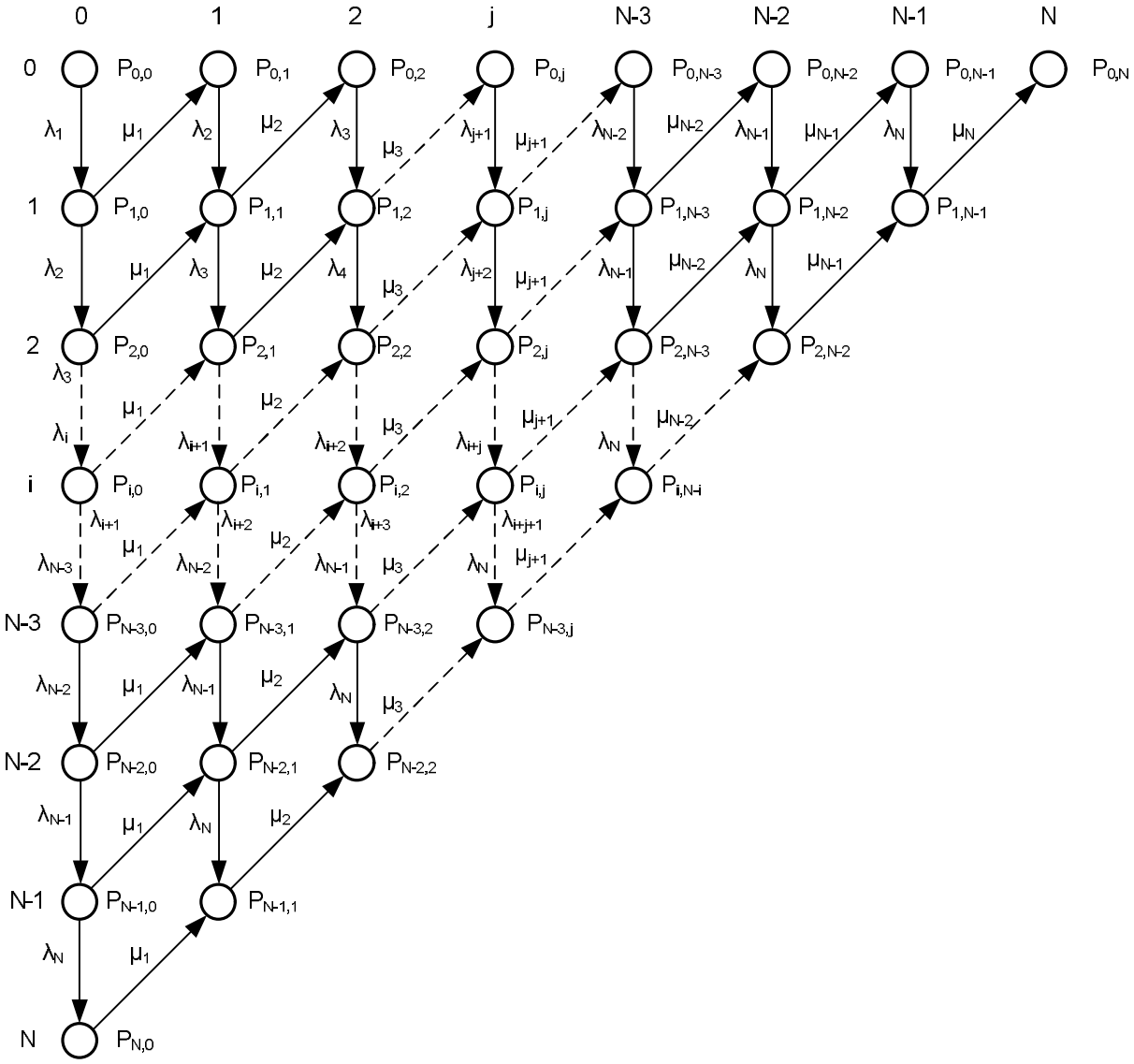

Рисунок 1. Диаграмма переходов между состояниями системы M(i)/M(j)/N

На представленной диаграмме переход из состояния (*i, j*) в состояние (*i+1, j*) означает, что при тестировании была обнаружена (*i+j+1*)-я ошибка. Переход из состояния (*i, j*) в состояние (*i-1, j+1*) означает, что была исправлена (*j+1*)-я ошибка. Общее число состояний *k* вычисляется по формуле:

$$k = (N+1)(N+2)/2. \quad (1)$$

Для определения вероятностей нахождения в состояниях (*i, j*) необходимо решить систему из *k* дифференциальных уравнений:

$$\frac{dP_{i,j}(t)}{dt} = \delta(i)\left(P_{i-1,j}(t)\lambda_{i+j} - P_{i,j}(t)\mu_{j+1}\right) + \\ +\delta(j)P_{i+1,j-1}(t)\mu_j - \delta(N-i-j)P_{i,j}(t)\lambda_{i+j+1}, \quad (2)$$

где $i = \overline{0,N}, j = \overline{0,N-i}$;

$$\delta(k) = \begin{cases} 1, \textit{если } k > 0; \\ 0, \textit{если } k \leq 0. \end{cases}$$

Для каждого момента времени *t* должно соблюдаться условие нормировки вида $\sum_{i=0}^{N}\sum_{j=0}^{N-i} P_{i,j}(t) = 1$.

Задав начальные условия к системе в виде

$$P_{i,j}(0) = \begin{cases} 0, \textit{если } i+j \neq 0; \\ 1, \textit{если } i+j = 0, \end{cases}$$

можно найти численное решение соответствующей задачи Коши для произвольного значения *t*.

Эта модель описывает процесс поиска и устранения ошибок, при котором обнаруженные ошибки устраняются последовательно по мере их обнаружения, причем поиск и устранение ошибок выполняются одновременно (рис. 1). Можно регламентировать и другие стратегии отладки (подробнее см. следующий раздел), которые представимы в виде графа, включающего в себя подмножество вершин графа, приведенного на рис. 1. Соответствующая система дифференциальных уравнений также будет определяться используемой стратегией отладки.

Выбор конкретной стратегии можно сделать по результатам моделирования с использованием значений показателей надежности и некоторых критериев (например, требуется выбрать стратегию с меньшим временем отладки до исправления (обнаружения) *k* ошибок с заданной вероятностью *p* и т. д.)

Варианты стратегии отладки

С точки зрения распределения времени между этапами тестирования и исправления ошибок можно рассматривать разные стратегии отладки программ (диаграммы переходов показаны на рис. 2):

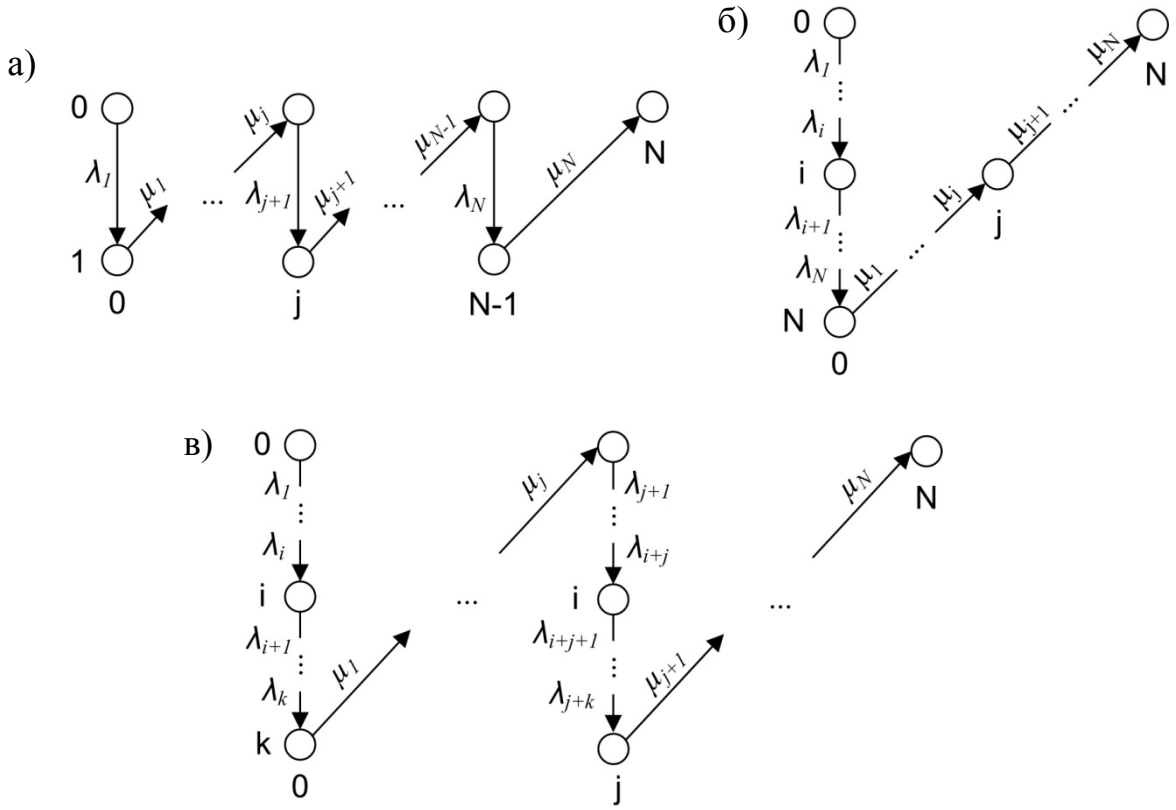

Рисунок 2. Диаграммы переходов для стратегий отладки:

а – стратегия 1, б – стратегия 2, в – стратегия 3

1. Этап тестирования продолжается, пока не будет обнаружена одна ошибка. После обнаружения ошибки тестирование приостанавливается, ошибка исправляется. После исправления ошибки вновь начинается тестирование.

Диаграмма переходов, представляющая данную стратегию отладки, представлена на рис. 2, а.

Количество состояний системы:

$$k = 2N + 1\,. \qquad (3)$$

Система дифференциальных уравнений (2) примет следующий вид:

$$\frac{dP_{i,j}(t)}{dt} = -\delta(N-j-iN)P_{i,j}(t)\lambda_{j+1} + \\ +\delta(i)\big(P_{i-1,j}(t)\lambda_{j+1} - P_{i,j}(t)\mu_{j+1}\big) + \\ +\delta(j(1-i))P_{i+1,j-1}(t)\mu_j\,, \tag{4}$$

где $i = \overline{0,1}, j = \overline{0,N-i}$.

2. Этап тестирования продолжается, пока не будут обнаружены все ошибки. Затем обнаруженные ошибки исправляются. Диаграмма, представляющая данную стратегию отладки, представлена на рис. 2, б.

Количество состояний системы рассчитывается по формуле (3).

Система дифференциальных уравнений (2) примет следующий вид:

$$\frac{dP_{i,j}(t)}{dt} = -\delta(N-i-jN)P_{i,j}(t)\lambda_{i+1} + \delta\big((1-j)i\big)P_{i-1,j}(t)\lambda_i - \\ -\delta\big((i+j+1-N)i\big)P_{i,j}(t)\mu_{j+1} + \\ +\delta\big((i+j+1-N)j\big)P_{i+1,j-1}(t)\mu_j\,, \tag{5}$$

где при $j = 0, i = \overline{0,N}$; при $j = \overline{1,N}, i = N-j$.

3. Этап тестирования продолжается, пока не будут обнаружены k ошибок. После обнаружения k ошибок тестирование приостанавливается, ошибки полностью исправляются. Затем тестирование продолжается. Диаграмма, представляющая данную стратегию отладки, представлена на рис. 2, в.

Количество состояний системы рассчитывается по формуле (3).

Система дифференциальных уравнений (2) примет следующий вид:

$$\frac{dP_{i,j}(t)}{dt} = \delta(1-j\ mod\ k)\times \\ \times\big(\delta(i)P_{i-1,j}(t)\lambda_{i+j} - \delta(k-i)\delta(N-i-j)P_{i,j}(t)\lambda_{i+j+1}\big) - \\ -\delta(i)\big(\delta(1-(i+j)\ mod\ k) + \delta(1-(i+j)\ mod\ N)\big)\times \\ \times P_{i,j}(t)\mu_{j+1} + \delta(i+j)\delta(N\ mod\ k - i)\times \\ \times(\delta(1-(i+j)\ mod\ k)\delta(k-i) + \\ +\delta(1-(i+j)\ mod\ N)\delta(N\ mod\ k - i))P_{i,j}(t)\mu_j\,. \tag{6}$$

Здесь: i mod j – остаток от деления i / j;

$i = \overline{0,k}$ при $j \bmod k = 0 \,\&\, j < N - N \bmod k$;

$i = k - j \bmod k$ при $j \bmod k \neq 0 \,\&\, j < N - N \bmod k$;

$i = \overline{0, N \bmod k}$ при $j = N - N \bmod k$;

$i = N \bmod K - j \bmod k$ при $N - N \bmod k < j \leq N$.

Модель, приведенную в предыдущем разделе и описываемую диаграммой на рис. 1 и системой дифференциальных уравнений (2), будем рассматривать в качестве стратегии 0.

Результаты прикладных расчетов и обоснование стратегии отладки

Применение описанных моделей позволяет получить динамические значения ряда величин (вероятности пребывания системы в состоянии (*i, j*), функцию распределения времени обнаружения и устранения ошибок, математическое ожидания случайных величин количество обнаруженных и исправленных ошибок). В качестве исходных данных для расчетов используются: число ошибок *N*, интенсивности $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ и $\{\mu_1, \mu_2, \ldots, \mu_N\}$ – обнаружения и исправления ошибок, соответственно.

При практическом использовании модели надежности M(i)/M(j)/N для расчетов в качестве исходных данных могут быть использованы статистические данные по интервалам времени между обнаружением и исправлением ошибок. Общее число ошибок *N* при этом неизвестно, однако может быть спрогнозировано, исходя из эмпирических соображений на основе опыта разработки похожих проектов, с использованием существующих прогнозных моделей (Джелинского-Моранды и подобных).

Наилучшим подходом представляется использовать для этого значения объектно-ориентированных метрик сложности, тем самым в большей степени будут учтены особенности конкретного проекта. Модель на основе логистической регрессии, описанная в работах [8, 9, 10], позволяет найти вероятности ошибки в классах программы. С

использованием этих вероятностей может быть дан прогноз общего количества ошибок программы, например, в виде количества классов, вероятность ошибки которых превышает некоторое пороговое значение. Могут быть предложены и более сложные алгоритмы определения количества ошибок в программе на основе значений объектно-ориентированных метрик сложности (например, с использованием нечетких множеств и продукционных систем), однако подробное рассмотрение данного вопроса выходит за тему статьи.

Представляет интерес сравнение рассмотренных выше стратегий отладки. В качестве исходных данных для представленных далее расчетов использована статистика по обнаружению и устранению ошибок в процессе разработки программного продукта учебный курс «Управление качеством при разработке программного обеспечения на основе современных стандартов и моделей». Проект выполнен в 2008 году в Петербургском университете путей сообщения. Учебный курс представляет собой web-приложение, при разработке которого использованы языки программирования javascript, actionscript. Данные по обнаруженным и устраненным ошибкам приведены в табл. 1. В ходе отладки были обнаружены и устранены ошибки пользовательского интерфейса, взаимодействия учебного курса с системой дистанционного обучения, верстки и др.

Таблица 1 **Исходные данные о программных ошибках**

Время тестирования (час)	**Число обнаруженных ошибок**	**Время устранения (час)**	**Интенсивность обнаружения ($час^{-1}$)**	**Интенсивность устранения ($час^{-1}$)**
4	25	10	6.25	2.5
4	19	4	4.75	4.75
2	3	2	1.5	1.5
2	1	1	0.5	1
1	1	1	1	1

Для расчетов использованы следующие значения интенсивностей обнаружения ошибок: λ_1=…=λ_{25}=6.25, λ_{26}=…=λ_{44}=4.75, λ_{45}=…=λ_{47}=1.5, λ_{48}=0.5, λ_{49}=1 и интенсивностей устранения ошибок μ_1=…=μ_{25}=2.5, μ_{26}=…=μ_{44}=4.75, μ_{45}=…=μ_{47}=1.5, μ_{48}=1, μ_{49}=1.

Полученные в результате решения систем дифференциальных уравнений (2, 4-6) вероятности $P_{i,j}$ пребывания систем в состоянии (i, j) могут быть использованы для получения значений показателей надежности.

Вероятность $R_j(t)$ того, что в процессе отладки *найдено* ровно *j* ошибок, может быть вычислена по следующей формуле:

$$R_j(t) = \sum_{i=0}^{j} P_{j-i,i}\,(t)\,. \tag{7}$$

Вероятность $P_j(t)$ того, что в процессе отладки *устранено* ровно *j* ошибок, может быть вычислена по следующей формуле:

$$P_j(t) = \sum_{i=0}^{N-j} P_{i,j}\ (t). \tag{8}$$

Среднее число (математическое ожидание) $N_R(t)$ обнаруженных ошибок к моменту времени t:

$$N_R(t) = \sum_{i=1}^{N} iR_i\,(t). \tag{9}$$

Среднее число (математическое ожидание) $N_P(t)$ исправленных ошибок к моменту времени t:

$$N_P(t) = \sum_{i=1}^{N} iP_i\,(t). \tag{10}$$

Соответственно, среднее число $N_{RL}(t)$ ошибок, не обнаруженных к моменту времени t, и среднее число $N_{PL}(t)$ ошибок, не исправленных к моменту времени t, определяются по формулам:

$$N_{RL}(t) = N - N_R(t). \tag{11}$$

$$N_{PL}(t) = N - N_P(t). \quad (12)$$

На рис. 3 приведены графики среднего числа обнаруженных ошибок и среднего числа еще не исправленных ошибок для предлагаемой модели надежности – системы M(i)/M(j)/N. Графики среднего числа обнаруженных ошибок для стратегий 0 и 2 совпадают.

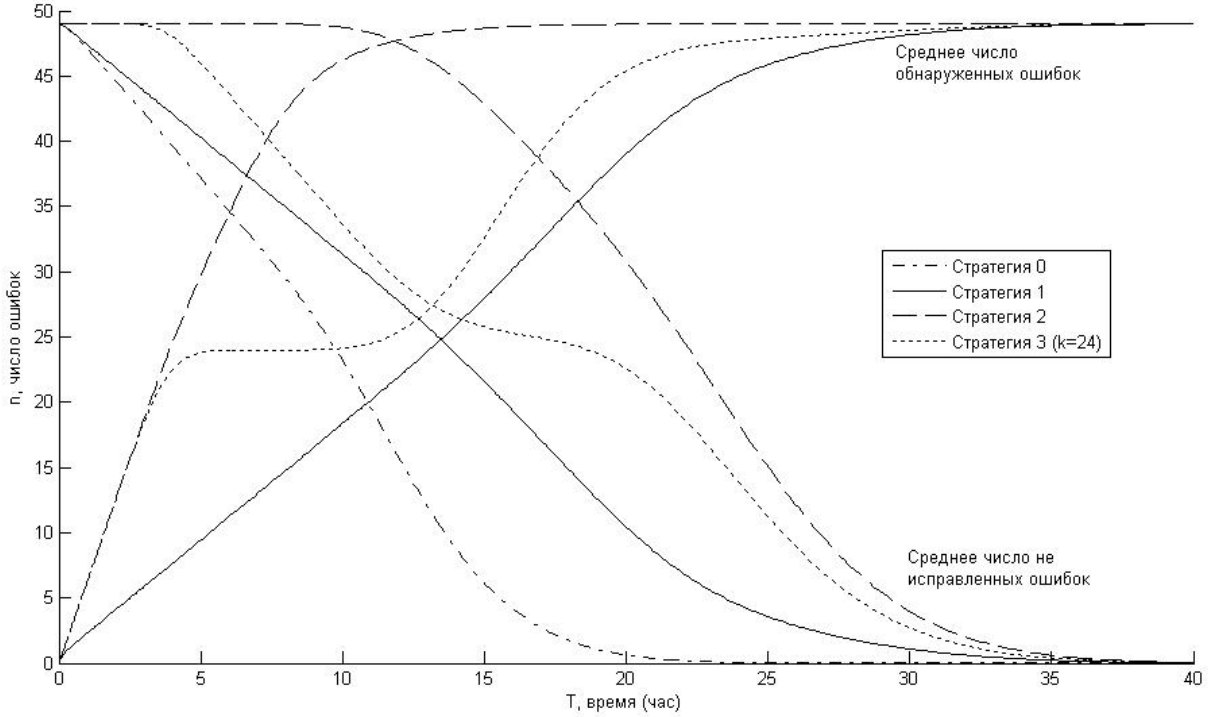

Рисунок 3. Графики среднего числа обнаруженных ошибок и среднего числа не исправленных ошибок

Приведенные графики могут быть использованы для расчета числа остающихся в программе ошибок к моменту времени τ с момента начала отладки. Для этого в качестве исходных данных должна использоваться достаточно полная статистика, собранная в процессе разработки большого числа проектов. При этом наличие для ряда программных проектов статистики по длительности интервалов обнаружения и устранения ошибок позволяет для аналогичных программных проектов использовать предположение о схожем характере динамики изменения интенсивностей выявления и устранения ошибок. В частности, можно говорить об одинаковом распределении отношений интенсивностей обнаружения и

устранения ошибок в случае, если программные проекты являются сопоставимыми по объему кода, квалификации разработчиков, используемым технологиям и инструментальным средствам и т.д.

Пример такого расчета для системы, *аналогичной описанной выше*, приведен в табл. 2.

Таблица 2

Расчет числа ошибок, не исправленных при отладке

Время τ (час)	**Среднее число не исправленных ошибок**			
	Стратегия 0	**Стратегия 1**	**Стратегия 2**	**Стратегия 3 (k=10)**
10	23.1	31.2	48.7	33.5
15	6.0	21.4	42.8	25.7
20	0.6	10.4	30.6	22.5
25	0.0	3.5	14.8	11.1
30	0.0	1.1	3.9	2.6
35	0.0	0.2	0.6	0.4

Предложенная модель позволяет найти функции $F_i(t)$ распределения времени устранения с помощью соотношения вида:

$$F_i(t) = \sum_{l=i}^{N} P_l(t), \ i = \overline{0, N} . \quad (13)$$

Это дает возможность оценить время отладки $T_{N_{\text{тр}},P_{\text{тр}}}$, которое требуется для исправления $N_{\text{тр}}$ ошибок с заданной вероятностью $P_{\text{тр}}$ как:

$$T_{N_{\text{тр}},P_{\text{тр}}} = t: F_{N_{\text{тр}}}(t) \geq P_{\text{тр}} . \quad (14)$$

Время $T_{N,P_{\text{тр}}}$ исправления всех программных ошибок не зависит от выбора стратегии отладки. Пример оценки времени $T_{N,P_{\text{тр}}}$ для системы M(i)/M(j)/N, описанной выше, приведен в табл. 3.

Таблица 3

Расчет времени отладки по вероятности отсутствия ошибок

Вероятность $P_{\text{тр}}$ отсутствия ошибок	**Требуемое время отладки $\tau_{\text{тр}}$ (час)**
0.8	34.3
0.9	36.3
0.95	38.1
0.99	40.4

Пример оценки требуемого времени отладки $T_{N_{\text{тр}},P_{\text{тр}}}$ при $P_{\text{тр}} = 0.95$ для разного числа ошибок $N_{\text{тр}}$, выполненный с помощью предлагаемой модели надежности M(i)/M(j)/N, приведен в табл. 4.

Таблица 4

Расчет времени отладки по допустимому числу ошибок

Количество оставшихся ошибок (N-$N_{\text{тр}}$)	**Время отладки $\tau_{\text{тр}}$ (час)**			
	Стратегия 0	**Стратегия 1**	**Стратегия 2**	**Стратегия 3 (k=10)**
5	18.1	26.4	33.3	32.1
3	19.7	29.6	34.9	33.6
1	22.2	35.6	36.9	35.6

Приведенные результаты дают наглядную численную картину того, как соотносятся между собой рассмотренные стратегии отладки ПС по приведенным характеристикам (см. табл. 2-4).

Заключение

В статье предложена нестационарная модель оценивания роста надежности программ на этапе отладки. По сравнению с разработанными ранее экспоненциальными вероятностными моделями надежности снимается предположение о виде зависимости интенсивности обнаружения ошибок от текущего номера обнаруженной ошибки.

Предложенная модель позволяет для различных стратегий отладки рассчитать:

- численные значения среднего числа обнаруженных ошибок и среднего числа не исправленных ошибок;
- функцию распределения времени обнаружения и устранения заданного числа ошибок;
- время отладки по заданной вероятности отсутствия ошибок или по допустимому числу ошибок.

Список литературы

1. Handbook of Software Reliability Engineering, McGraw Hill, 1996.
2. Уткин Л.В., Шубинский И.Б. Нетрадиционные методы оценки надежности информационных систем. – СПб.: Любавич, 2000.
3. Половко А.М., Гуров С.В. Основы теории надежности. СПб.: БХВ-Петербург, 2006.
4. Смагин В.А. Техническая синергетика. Вероятностные модели сложных систем ─ СПб. ─ 2004.
5. Тырва А.В., Хомоненко А.Д. Метод планирования тестирования сложных программных комплексов на этапах проектирования и разработки. Научно-технический вестник СПбГПУ / Информатика. Телекоммуникации. Управление. 2009. № 4(82). – С. 125-131.
6. Huang C.-Y., Huang W.-C. Software Reliability Analysis and Measurement Using Finite and Infinite Server Queuing Models. IEEE Transactions On Reliability, V. 57, NO. 1, march 2008. – 192-203 pp.
7. Бубнов В.П., Сафонов В.И. Разработка динамических моделей нестационарных систем обслуживания. – СПб.: Издательство “Лань”, 1999.
8. El-Emam K., Melo W., Machado J.C. The prediction of faulty classes using object-oriented design metrics - Journal of Systems and Software, Volume 56, Number 1, 2001. – pp. 63-75.
9. Chidamber S.R., Kemerer C.F. A metrics suite for object oriented design. Software Engineering, IEEE Transactions on Volume 20, Issue 6, Jun 1994. – pp. 476-493.
10. Basili V.R., Briand L.C., Melo W.L. A validation of object-oriented design metrics as quality indicators - IEEE Transactions on Software Engineering, Volume 22, Issue 10, Oct 1996. – pp. 451-761.