Тырва А.В., Хомоненко А.Д.

МЕТОД ПЛАНИРОВАНИЯ ТЕСТИРОВАНИЯ СЛОЖНЫХ ПРОГРАММНЫХ КОМПЛЕКСОВ НА ЭТАПАХ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ

Введение

Планирование и распределение ресурсов программных проектов является важной управленческой задачей. Ее решение необходимо для повышения эффективности процесса разработки программного обеспечения. Задача планирования работ по проведению тестирования и отладки является особенно актуальной при разработке сложных программных комплексов (СПК), к которым предъявляются высокие требования надежности. В частности, к таким СПК относятся многие комплексы автоматизированных систем на железнодорожном транспорте (АСУ процессом перевозок, технологическими процессами и т.д.).

При разработке СПК важно определить набор модулей, потенциально приводящих к отказу, чтобы выделить для них наибольшие ресурсы: квалифицированный персонал, большее время на тестирование и т.д. Чем раньше это будет сделано, тем больший будет достигнут эффект. Так как, с одной стороны, исправление обнаруженных уже после выпуска программного средства ошибок обходится в десятки и сотни раз дороже, чем на этапе проектирования и реализации. С другой стороны, статистика показывает, что наибольшее число ошибок локализовано в относительно небольшом количестве отдельных модулей: в среднем 80% ошибок приходится лишь на 20% модулей.

1. Общее описание предлагаемого метода планирования тестирования

В статье предлагается метод планирования работ по тестированию СПК, развивающий подход [Смагин2004], в котором СПК представляется в виде совокупности m программных модулей. Отдельные модули подвергаются автономной отладке и тестированию для оценивания и

повышения характеристик их надежности (вероятности безошибочного функционирования). Далее на основе результатов тестирования (число зафиксированных отказов, интенсивность отказов каждого модуля) и предположения об экспоненциальном характере распределения времени между отказами по модели Джелинского-Моранды находятся вероятности безотказной работы каждого модуля. Полученные оценки служат для определения потенциально приводящих к отказу модулей, последовательности отладки (в порядке увеличения вероятности безотказной работы) и момента окончания отладки.

Применение названного подхода возможно на поздних этапах цикла разработки программных средств (ПС), когда готов программный код, началась автономная отладка и собран достаточный объем статистики по отказам модулей СПК для использования моделей надежности. Используемая стратегия не является наилучшей в смысле минимизации трудозатрат при тестировании и отладке СПК. Так как модуль, у которого вероятность безошибочного функционирования является наименьшей, не обязательно оказывает наибольшее влияние на снижение вероятности безошибочной работы СПК в целом.

Предлагаемый метод планирования тестирования СПК снижает трудозатраты на тестирование и отладку, и обладает следующими особенностями:

- использует метрики сложности в качестве исходных данных для анализа и планирования;
- обеспечивает учет вклада каждого модуля в надежность СПК в целом на основе прогноза времени и вероятности его исполнения;
- может применяться на всех этапах цикла разработки СПК;
- использует экспертные оценок о характере распределения времени выполнения каждого модуля;

- основан на использовании стохастического графа СПК (в виде матрицы вероятностей вызовов отдельных модулей).

Применение предлагаемого метода планирования испытаний схематично проиллюстрировано на рис. 1. На нем приведены два проекта по разработке ПС. В проекте 1 завершены этапы проектирования (собраны проектные метрики сложности из UML-диаграмм), кодирования и тестирования (см. раздел 2). По статистике ошибок, собранной при тестировании, модули СПК разделяются на два класса – содержащие и не содержащие ошибки. Затем строится модель прогноза надежности – зависимость вероятности проявления ошибки в модуле от значений проектных метрик сложности (см. раздел 3). В проекте 2 выполнены только работы этапа проектирования и собраны проектные метрики сложности из UML-диаграмм. По построенной модели прогнозируются потенциально ненадежные модули. Использование экспертных заключений о характере распределения времени исполнения отдельных модулей и стохастического графа СПК позволяет выполнить оценку степени влияния каждого модуля на надежность СПК в целом и, следовательно, оптимизировать распределение ресурсов тестирования, построить план тестирования (см. раздел 4).

Сравнение результатов использования предлагаемой стратегии с известными аналогами при планировании тестирования планирования проведения тестирования СПК показывает, что достигается повышение эффективности тестирования и рост надежности программного продукта (см. раздел 5).

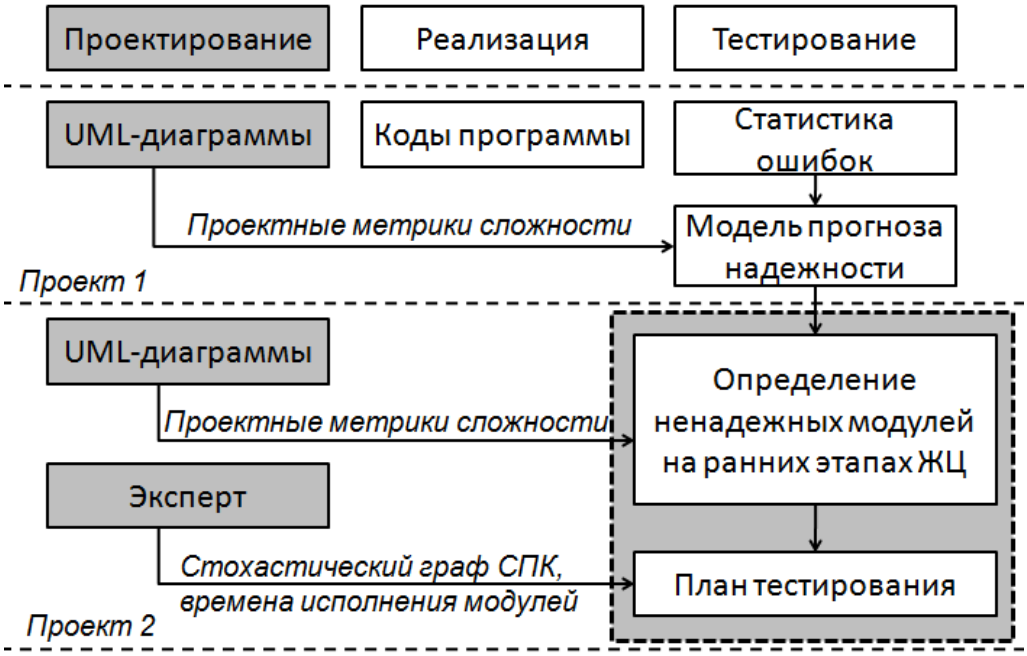

Рисунок 1 – Использование стратегии планирования тестирования СПК

2. Жизненный цикл

В процессе разработки СПК проходит несколько этапов. Вне зависимости от модели жизненного цикла (водопадная, спиральная, экстремальное программирование и др.) обычно выделяются следующие этапы: анализ требований, проектирование, реализация, тестирование и отладка, внедрение, эксплуатация и сопровождение. На каждом из этапов разрабатываются артефакты, которые могут служить источником измерений для последующего моделирования и планирования проекта. Например, на этапе проектирования разрабатывается архитектура СПК и подсистем, структур данных и т.д. При использовании широко распространенного унифицированного языка моделирования UML (Unified Modeling Language) на этапе проектирования создается набор диаграмм: диаграмма классов, состояний, вариантов использования и др. Эти диаграммы могут быть использованы для сбора измерений и прогноза надежности компонентов системы (см. следующий раздел).

На этапе реализации разрабатываются алгоритмы обработки данных, и создается программный код. Таким образом, по сравнению с

проектированием, становится доступной дополнительная информация для анализа, связанная с качеством и сложностью кода, структурой алгоритма и т.д. в виде соответствующих метрик сложности (см. следующий раздел).

На последующих этапах жизненного цикла собирается статистика об отказах СПК (в процессе отладки и эксплуатации), и для прогноза надежности становится возможным применение различных моделей надежности: Джелинского-Моранды, Мусы, Гоэл-Окумото и др.

3. Метрики сложности

Предлагаемая стратегия планирования тестирования может применяться на различных этапах жизненного цикла – от проектирования до внедрения. Наибольший эффект достигается при проектировании, что является главной темой статьи. Применение метода на других этапах жизненного цикла отличается используемыми средствами моделирования надежности программных модулей и набором метрик сложности, и кратко описано в конце текущего раздела.

В литературе большое внимание уделяется вопросам разработки, проверки и применения метрик сложности программных средств, доступных на этапе проектирования (обзор подходов можно найти в [El-Emam2001, Genero2005]). Для решения описываемых в данной статье задач предлагается использовать набор объектно-ориентированных метрик, предложенных в [Briand1997, Chidamber1994]. Их выбор обоснован следующими соображениями:

- метрики основаны на объектно-ориентированном подходе и применимы (с небольшими изменениями) для различных языков программирования (C++, C#, Java и др.);
- метрики теоретически и математически обоснованы [Briand1997, Chidamber1994], хорошо известны и практически применимы [Genero2005];

- исследована и подтверждена возможность применения для раннего обнаружения модулей, потенциально приводящих к отказу [Basili1996, Briand1997, El Emam2001, El Emam2002, Genero2005];
- сбор значений метрик сложности не требует больших трудозатрат и может быть автоматизирован [Genero2005], в том числе при сборе из проектных документов, разработанных с использованием унифицированного языка моделирования UML – фактически промышленного стандарта проектирования.

На этапе проектирования СПК целесообразно использовать следующие метрики сложности [Basili1996, Briand1997, Chidamber1994, El-Emam2001]:

- количество классов, использующих данный класс (в качестве атрибута, параметров методов, локальных переменных и др.) – coupling between object classes (CBO);
- количество методов и функций, которые используются для обработки поступающих событий – response for a class (RFC);
- длина наибольшего пути иерархии наследования – depth of inheritance tree (DIT);
- количество классов-наследников – number of children (NOC);
- взвешенное количество методов класса – weighted methods per class (WMC);
- разность между количеством пар методов класса, использующих общие объекты, и количеством пар методов, не использующих общие объекты – lack of cohesion in methods (LCOM);
- количество атрибутов классов – total number of attributes (NAI);
- набор метрик, определяющих взаимодействие между классами: количество классов с параметрами методов типа данного класса – others class-method export coupling (OCMEC), количество классов-

друзей с атрибутами типа данного класса – friends class-attribute export coupling (FCAEC) и другие [Briand1997, El-Emam2001].

На основе полученных метрик решается задача классификации модулей с целью выявления потенциально вызывающих ошибку. Построение такой модели осуществляется с использованием средств математической статистики или искусственного интеллекта. Например, вероятность наличия ошибок в модуле π можно оценить с использованием логистической регрессии и выразить в виде [Basili1996, El-Emam2001]:

$$\pi_i = \frac{1}{1 + e^{-(\beta_0 + \sum \beta_j x_j)}} \tag{1}$$

где x_i – значение метрики сложности;

β_i – коэффициент.

Параметры модели β_i находятся методом максимального правдоподобия [Hosmer2000]. Например, для статистики о n модулях функция правдоподобия $L(\boldsymbol{\beta})$ имеет вид:

$$L(\overline{\beta}) = \prod_{i=1}^{n} \pi(\overline{x_i})^{y_i} [1 - \pi(\overline{x_i})]^{1-y_i}, \tag{2}$$

где yi = 1, если в i-м модуле была обнаружена ошибка, yi = 0 иначе;

$\overline{x_i}$ – вектор значений метрик сложности для i-го модуля;

$\overline{\beta}$ – вектор параметров модели.

Значения параметров модели определяет вектор $\boldsymbol{\beta}$, при котором функция правдоподобия максимальна. Для нахождения вектора численными методами решается система уравнений [Hosmer2000]:

$$\begin{cases} \sum_{i=1}^{n} [y_i - \pi(x_i)] = 0 \\ \sum_{j} x_j [y_i - \pi(x_i)] = 0 \end{cases} \tag{3}$$

В [El-Emam2001] приводится пример и результат такого моделирования, для чего авторами было проанализировано готовое коммерческое приложение, программные модули классифицированы как ненадежные (при тестировании обнаружены ошибки) и надежные, определены проектные метрики сложности и построена регрессионная модель:

$$\pi = \frac{1}{1 + e^{-(-3.97 + 0.464\,NAI + 1.47\,OCMEC + 1.06\,DIT)}} \qquad (4)$$

Модели, аналогичные описанной выше, могут быть построены и для других наборов метрик сложности, в том числе на поздних этапах жизненного цикла. Например, на этапе реализации для анализа программного кода могут быть использованы следующие метрики: LOC (Lines of code), метрики Холстеда и МакКейба, функциональные точки и др. (обзор метрик сложности и примеры их использования для моделирования надежности можно найти в [Khoshgoftaar1992]). Как отмечалось выше, после начала отладки, когда появляются данные о распределении отказов, становится возможным учитывать в прогнозе также оценки моделей надежности.

4. Математическая модель планирования тестирования

Следуя общепринятому подходу, планирование тестирования СПК основывается на результатах двоичной классификации модулей СПК на класс потенциально приводящих к отказу и надежных.

Новизной предлагаемого подхода является то, что описанная в [Смагин2004] стратегия расширена на более ранние этапы жизненного цикла, а исходными данными являются программные метрики сложности. Кроме того, при планировании испытаний учитывается вклад каждого модуля в надежность СПК в целом на основе прогноза времени и вероятности его исполнения, получаемого из экспертной оценки

стохастического графа СПК и характера распределения времени исполнения модулей.

Таким образом, для расчета вероятности π_i' наличия ошибки в i-м модуле при функционировании в СПК предлагается использовать оценки его надежности π_i, вероятности q_i выполнения модуля при работе в СПК и среднего времени выполнения t_i:

$$\pi_i' = K \cdot \pi_i \cdot q_i, \tag{5}$$

Для расчета времени выполнения модулей, СПК представляется в виде сети массового обслуживания, узлами которой являются модули СПК, а ветвями – вызовы отдельных модулей с различными вероятностями. Затем вычисляются первые моменты времени прохождения заявки между модулями за бесконечное число шагов. Для этого строится матрица $y(s)$ [Смагин1989, Смагин2004]:

$$y(s) = \begin{vmatrix} P_{00} & P_{01} & P_{02} & . & . & . & P_{0,L+1} \\ P_{10}y_1(s) & P_{11}y_1(s) & P_{12}y_1(s) & . & . & . & P_{1,L+1}y_1(s) \\ . & . & . & & . & & . \\ . & . & . & & . & & . \\ . & . & . & & . & & . \\ P_{L0}y_L(s) & P_{L1}y_L(s) & P_{L2}y_L(s) & . & . & . & P_{L,L+1}y_L(s) \\ P_{L+1,0} & P_{L+1,1} & P_{L+1,2} & . & . & . & P_{L+1,L+1} \end{vmatrix} \tag{6}$$

где P_{ij} – вероятность вызова j-го модуля после исполнения i-го;

$y_i(s)$ – преобразование Лапласа-Стилтьеса (ПЛС) распределения времени исполнения i-го модуля.

Для определения времени выполнения СПК необходимо рассмотреть матрицу, элементами которой являются ПЛС функции распределения перехода из одного модуля в другой за бесконечное число шагов $V(s)$:

$$V(s)=I+y(s)+y^2(s)+\ldots=I(I-y(s))^{-1} \tag{7}$$

где I – единичная матрица.

$$v_{ij}(s)=A_{ji}(s)/D(s) \tag{8}$$

где $A_{ji}(s)$ – алгебраическое дополнение элемента (i,j) матрицы $v(s)$;

$D(s)$ – определитель матрицы $v(s)$.

Среднее время выполнения СПК (при s=0):

$$T = -\frac{dv_{0,L+1}(s)}{ds}, \tag{9}$$

Среднее время выполнения i-го модуля:

$$T_i = q_i T \tag{10}$$

С целью универсализации алгоритма предлагается аппроксимировать конкретные виды распределений (экспоненциального, Вейбулла, гамма и других) гиперэкспоненциальным. В этом случае плотность распределения представляется суммой конечного числа экспонент. С практической точки зрения для обеспечения приемлемой точности достаточно использовать две экспоненциальные фазы.

Вероятность выполнения j-ого модуля q_j определяется путем решения системы линейных уравнений:

$$q_j \frac{1}{b_j} = \sum_{i=1}^{L} q_i \frac{P_{i,j}}{b_i} \tag{13}$$

где b_i – среднее время выполнения i-ого модуля;

$P_{i,j}$ – вероятность перехода из i-го модуля в j-ый.

Распределение ресурсов тестирования производится на основе значений полученных оценок надежности модулей (вероятности наличия ошибок π_i'). Например, если известно общее время модульного тестирования СПК τ, то расчет времени тестирования i-го модуля τ_i производится по следующей формуле:

$$\tau_i = \tau \cdot \pi_i' \tag{14}$$

Модель, описанная в предыдущем разделе, запрограммирована с использованием математического пакета MATLAB R2007B,

инструментального средства символьных вычислений Symbolic Math Toolbox.

5. Оценка эффективности предлагаемой стратегии планирования проведения тестирования СПК

Для оценки эффективности предлагаемого подхода произведен расчет прогнозируемой вероятности безотказной работы СПК после проведения тестирования в течение времени, рассчитанного по двум моделям: предлагаемой и описанной в [Смагин2004]. Поскольку модель [Смагин2004] не предназначена для применения на этапе проектирования, для сравнения использовалась модифицированная модель. В ней при распределении времени тестирования между модулями учитываются полученные по метрикам сложности оценки надежности π_i и не учитываются вероятности исполнения P_j, что соответствует процедуре тестирования критичных модулей, описанной в [Смагин2004]. Для расчета вероятности безотказной работы i-го модуля R_i за среднее время функционирования t_i использована модель Мусы [Муса1980]:

$$R_i = e^{-t_i / T_i}, \tag{15}$$

Здесь T_i – средняя наработка на отказ после проведения тестовых испытаний.

$$T_i = T_i' e^{\frac{C\tau_i}{M_i T_i'}}, \tag{16}$$

где T_i' – средняя наработка на отказ в начале испытаний;

C – коэффициент сжатия тестов;

τ_i – время испытаний;

M_i – общее число дефектов, которые могут произойти за время испытаний τ_i.

При расчетах были использованы следующие предположения:

- средняя наработка на отказ в начале испытаний T_i' обратно пропорциональна оценке вероятности ошибки в модуле, полученной по значениям метрик сложности;
- зависимость общего количества ошибок в модуле M_i от оценки вероятности ошибки в модуле имеет вид сигмоидной функции;
- время тестирования модуля τ_i пропорциональны взвешенной оценке наличия ошибки в модуле, полученной по двум анализируемым моделям.

Вероятность безотказной работы СПК может быть вычислена с использованием формул (6)-(9), если заменить преобразование Лапласа-Стилтьеса $y_i\ (s)$ вероятностью безотказной работы i-го модуля в течение среднего времени функционирования $p_i(t_i)$.

В ходе анализа исследуется зависимость эффективности применения предложенного метода планирования от общего времени тестирования, выраженного в количестве прогонов программы при тестировании.

Под эффективностью понимается отношение (R'-R) / R, где R' – прогнозная вероятность безотказной работы СПК после тестирования в течение времени, рассчитанного с использованием предлагаемой стратегии, R' – с использованием стратегии, описанной в [Смагин2004].

Зависимость эффективности использования предложенной модели от времени тестирования приведена на рис. 2. Анализ графика показывает, что предложенная модель эффективнее аналогов. Наибольший эффект достигается при относительно небольшом времени тестирования и составляет около 5%.

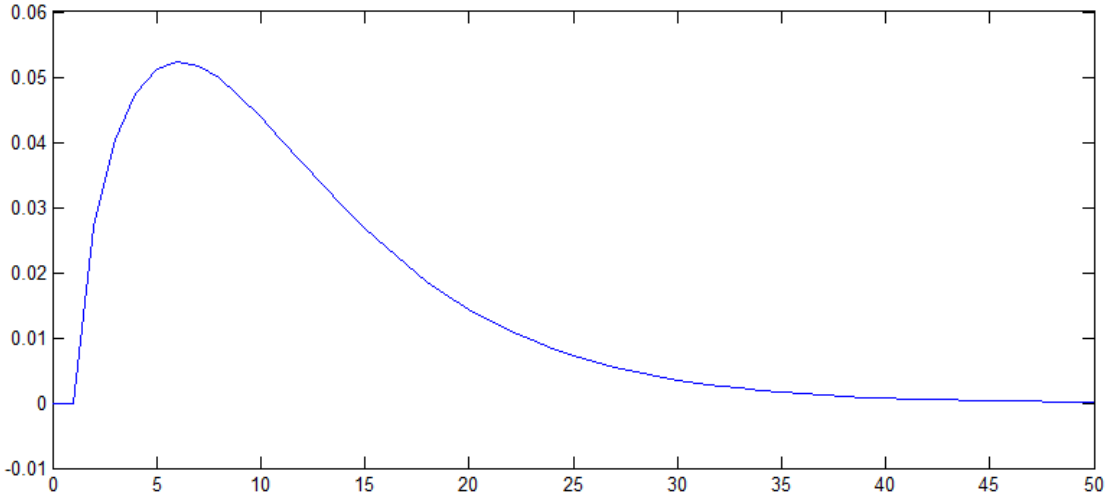

Рисунок 2 - Зависимость эффективности использования модели от времени тестирования

Выводы

В статье приведено описание метода планирования работ по тестированию СПК с использованием метрик сложности модулей СПК. Разработанная модель позволяет проводить классификацию модулей с целью раннего обнаружения потенциально вызывающих отказ. Одним из преимуществ разработанного подхода является то, что он может быть применен на различных этапах цикла разработки СПК, включая этап проектирования.

Литература

1. *Муса Дж. Д.* Измерение и обеспечение надежности программных средств – ТИИЭР, т. 68, №9, 1980.
2. *Смагин В.А.* Техническая синергетика. Вероятностные модели сложных систем ─ СПб. ─ 2004.
3. *Смагин В.А., Бубнов В.П., Филимонихин Г.В.* Расчет вероятностно-временных характеристик пребывания задач в сетевой модели массового обслуживания. // Изв. ВУЗов. Приборостроение. ─ Т. XXXII ─ № 2. ─ 1989.

4. *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.
5. *Briand L., Devanbu, P., Melo, W.* An Investigation into Coupling Measures for C++ - Proceedings of the 1997 (19th) International Conference on Software Engineering, 17-23 May 1997.
6. *Chidamber S.R., Kemerer C.F.* A metrics suite for object oriented design - Software Engineering, IEEE Transactions on Volume 20, Issue 6, Jun 1994.
7. *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, 1 February 2001
8. *Genero M., Piattini M., Caleron C.* A survey of metrics for UML class diagrams - Journal of Object Technology, Vol. 4, 2005.
9. *Hosmer D. W., Lemeshow S.* Applied Logistic Regression, 2nd ed. New York: Wiley, 2000.
10. *Khoshgoftaar T. M., Munson J. C., Bhattacharya B. B., Richardson G. D.* Predictive Modeling Techniques of Software Quality from Software Measures - IEEE TRANSACTIONS ON SOFTWARE ENtilNtERING. VOL IH, NO 11, NOVEMBER 1992.