УДК 004.054; 004.582

А.С. ПРИГОЖЕВ

Одесский национальный политехнический университет, Украина

ИСПОЛЬЗОВАНИЕ РЕСУРСНЫХ СЕТЕЙ ДЛЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Основной задачей тестирования является установление соответствия программы и её спецификации. Этот процесс представляется достаточно сложным и требующим автоматизации. Основными проблемами при этом является большая разветвлённость процесса решения задачи, большое количество тестовых наборов, а также отсутствие формальных критериев, позволяющих однозначно оценить соответствие программы и спецификации. В представленной работе предлагается использование для тестирования программного обеспечения новой потоковой модели – гибридных ресурсных сетей. Использование данного механизма позволяет существенно упростить и автоматизировать процесс тестирования программного обеспечения, сведя его к задаче поиска подобия между изображениями. Использование предлагаемого подхода в автоматизированных системах тестирования позволяет формализовать критерии, по которым возможно установить соответствие между программной и спецификацией.

Ключевые слова: *тестирование программ, поддержка разработчика, автоматизация тестирования, ресурсные сети.*

Введение

Тестирование программного обеспечения (ПО) является достаточно сложной и трудоёмкой задачей. Это обуславливается большим количеством данных, обрабатываемых программой, а также сложностью самих задач, решаемых с помощью ПО. Поэтому, тестирование никогда не может быть выполнено в полном объёме и гарантировать полное соответствие специфицируемым требованиям. Актуальной задачей в таких условиях становится разработка автоматизированных систем тестирования ПО.

Актуальной задачей при проведении тестирования программного обеспечения является верификация ПО, т.е. проверка ПО на соответствие некоторой эталонной спецификации [1].

Основная цель процесса верификации – доказательство того, что результат разработки соответствует предъявленным к нему требованиям. Обычно процесс верификации проводится сверху вниз, начиная от общих требований, заданных в техническом задании и/или спецификации на всю информационную систему до детальных требований на программные модули и их взаимодействие. В состав задач процесса входит последовательная проверка того, что в программной системе:

1. Общие требования к информационной системе, предназначенные для программной реализации, корректно переработаны в спецификацию требований высокого уровня к комплексу программ, удовлетворяющих исходным системным требованиям;

2. требования высокого уровня правильно переработаны в архитектуру ПО и в спецификации требований к функциональным компонентам низкого уровня, которые удовлетворяют требованиям высокого уровня;

3. спецификации требований к функциональным компонентам ПО, расположенным между компонентами высокого и низкого уровня, удовлетворяют требованиям более высокого уровня;

4. архитектура ПО и требования к компонентам низкого уровня корректно переработаны в удовлетворяющие им исходные тексты программных и информационных модулей;

5. исходные тексты программ и соответствующий им исполняемый код не содержат ошибок

Для решения перечисленных задач в настоящее время разработано достаточно большое количество разнообразных автоматизированных сред тестирования. Рассмотрим подробнее достоинства и недостатки таких систем.

1. Существующие средства автоматизации тестирования и верификации

На сегодняшний день существует большое количество разнообразных подходов и программ для тестирования программного обеспечения. Рассмот-

рим наиболее известные подходы. Одним из наиболее распространённых подходов к автоматизации процесса тестирования и верификации являются модель Крипке [2], темпоральные логики, а также модели, основанные на автоматах[3]. Данные подходы оперируют с таким понятием, как состояние программы, которое понимается как набор всевозможных значений обрабатываемых данных. При достаточно большом числе данных и размерности этих данных возникает проблема «экспоненциального взрыва размерности». Именно данная проблема послужила основной причиной того, что на рынке ПО для верификации и тестирования присутствуют в основном узкоспециализированные решения, ориентированные, как правило, только на тестирование отдельных типов ПО (без их верификации), в частности web-приложений [4-6].

Такая распространённость средств автоматизированного тестирования веб-приложений связана с несколькими причинами: наличием достаточного развитых методов анализа сетевых протоколов, возможностью описания функционирования приложений на основе автоматных моделей и т.п.

Не менее актуальным является развитие средств тестирования ПО, не связанного напрямую с Web, и написанных на алгоритмических языках программирования, таких как C++, C#, Java и т.п. Сейчас существует ограниченное количество средств, связанных с тестированием программ на алгоритмических языках программирования. Это связано прежде всего с тем, что автоматные модели таких программ имеют достаточно большую (часто бесконечную) размерность, т.е. возникает проблема экспоненциального роста состояний.

Поэтому, актуальной является задача дальнейшего развития средств автоматизированного тестирования и верификации, за счёт использования в них универсальных моделей, ориентированных на описание любого процесса взаимодействия между двумя объектами, независимо от языки .

Перспективным направлением построения таких универсальных языконезависимых моделей на основе интеграции известных транспортных сетей (т.н. моделей Форда-Фалкерсона) и новой потоковой модели – однородной ресурсной сети [7, 8]. Рассмотрим построение такой интегрированной модели, а также её функционирование на примере объектно-ориентированного ПО.

2. Гибридная ресурсная сеть

Под гибридной ресурсной сетью будем в дальнейшем понимать многослойную сеть в которой каждый слой, в зависимости от спецификации модели, ведёт себя либо как однородная ресурсная сеть, либо как обычная транспортная сеть [9, 10] с вершинами-истоками и вершинами-стоками.

Для слоёв, связанных с однородными ресурсными сетями, будем заранее предполагать, что пропускная способность всех рёбер есть некоторая функция f. Определим вид данной функции.

Состоянием ресурсной сети в момент времени t называется вектор вида:

$$V(t) = (x_1(t), x_2(t), ..., x_i(t), ..., x_n(t)). \quad (1)$$

В формуле (1) $x_i(t)$ - количество ресурса в i-й вершине ресурсной сети в момент времени t, n - количество вершин в данной сети.

Рассмотрим функцию вида:

$$X(t) : V(t) \to V(t+1). \quad (2)$$

В формуле (1) $V(t)$ - вектор состояний вершин ресурсной сети в некоторый текущий момент времени t, $V(t+1)$ - состояние ресурсной сети в момент времени $t+1$.

Данную функцию будем считать далее случайной функцией, зависящей от времени t. Это позволит говорить о наличии множества некоторых реализаций данной функции.

Сопоставим случайной $X(t)$ функции предикат, который определим следующим образом:

$$F : V(t) \times V(t+1) \to \{0;1\}. \quad (3)$$

Данный предикат будет истинен тогда и только тогда, когда для некоторых двух моментов времени t и $t+1$ имеет место утверждение: «пара $(V(t), V(t+1)) \in X(t)$»

Определим понятие ориентированного графа, соответствующего данной ресурсной сети следующим образом:

Ориентированный граф $G =< V, R >$ соответствующий данной ресурсной сети - это ресурсная сеть без указания пропускных способностей.

Введём в рассмотрение двуместный предикат $L : V \times V \to \{0;1\}$, определяемый следующим образом: $L(x_i, x_j)$ истинен тогда, и только тогда, когда $(x_i, x_j) \in R$.

Введём в рассмотрение двуместный однородный предикат H на множестве вершин ресурсной сети V и зависящий от времени t, определяемый следующим образом:

$$H : V \times V \to \{0;1\}. \quad (3)$$

Данный предикат истинен для некоторых двух вершин $x_i, x_j \in V$ тогда и только тогда, когда в некоторый момент времени t выполняется:

$$L(x_i, x_j) \,\&\, F((x_1(t), x_2(t), ..., x_i(t), ..x_j(t), ...x_n(t)), (x_1(t+1), x_2(t+1), ..., x_i(t)-1, ..x_j(t)+1, ...x_n(t))) = 1. \quad (4)$$

Сопоставим указанный в формуле (3) предикат рёбрам, соединяющим указанные две вершины и значение данного предиката в момент времени t будем считать величиной пропускной способности ребра в некоторый момент времени t.

Для слоёв, являющихся транспортными сетями, также введём понятие состояния сети в некоторый момент времени t, согласно формуле (1). В этом случае компонентами этого вектора будут являться вершины соответствующей транспортной сети.

Пропускные способности транспортной сети будут величинами из множества $\{0;1\}$. Эти величины могут быть как константами, так и предикатами, задаваемыми в предметной области разрабатываемого программного продукта.

Для гибридной ресурсной сети введём понятие яркости[8] некоторого слоя как количества ресурса, находящегося в данный момент времени на данном слое. Соответственно, состоянием слоя будем называть состояние соответствующей ему сети, определяемое формулой (1). Состоянием сети в целом будем называть вектор, элементами которого будут являться вектора состояний слоёв. Поскольку эти вектора зависят от времени, то последовательность данных векторов будет представлять собой двумерный сигнал. К такого рода сигналу применимы известные алгоритмы обработки сигналов, позволяющих установить подобие между сигналами, создаваемыми моделью, основанной на спецификации и моделью, основанной на программном коде. Это позволяет в значительной степени упростить решение проблемы тестирования и верификации программ.

3. Моделирование программного кода с использованием гибридных ресурсных сетей

Элементарной единицей программного кода в процедурных и объектных языках является процедура (или метод объекта). Любая процедура в программе может быть представлена в виде некоторого графа, по аналогии с [11]. Рассмотрим описание с помощью такого графа процедуры сортировки методом пузырька:

```
void bubblesort(int *a, int n)
{
  int i, j,t;
 for (i = n - 1; i > 0; i--)
  {
    for (j = 0; j < i; j++)
    {
      if (a[j] > a[j + 1])
      {
      t = a[j];
     a[j] = a[j + 1];
      a[j + 1] = t;
     }}}
```

Соответствующий данной процедуре граф (рис. 1), при тестировании программного обеспечения, можно рассматривать как транспортную сеть, где начальная вершина – это стартовая точка процедуры, конечная – точка окончания процедуры, а ресурсом являются тестовые пакеты, содержащие все используемые переменные и их характеристики. Формат тестового пакета совпадает с описанием одного шага алгоритма, приведенного в [12].

Такая транспортная сеть, построенная для одной процедуры образует один слой гибридной ресурсной сети.

При взаимных вызовах нескольких процедур возможны два основных варианта модели: построение гибридной ресурсной сети для вызовов независимых процедур и построение модели для рекурсивного вызова процедур.

При построении модели для взаимного вызова двух несвязанных процедур для каждой процедуры строится отдельная транспортная сеть. Каждая из построенных сетей представляет собой отдельный слой модели. Такой слой модели будем называть слоем реализации процедуры. Далее вершина, соответствующая вызову процедуры и вершина-исток транспортной сети, моделирующей вызываемую процедуру, объединяются в единую транспортную сеть, образующую отдельный слой, именуемый в дальнейшем слой вызова, где вершиной истоком служит первая вершина, а стоком – вторая.

Однако такой подход не применим в случаях, когда речь идёт о рекурсивной процедуре, поскольку в этом случае такое построение приводит к появлению бесконечного количества одинаковых слоёв реализаций и вызовов процедур. Чтобы этого избежать и одновременно промоделировать рекурсивный вызов процедуры, необходимо вершину, в которой осуществляется рекурсивный вызов соединить направленным ребром с вершиной-истоком того же слоя

4. Моделирование интерфейса пользователя с использованием гибридных ресурсных сетей

Современные программные системы используют в качестве средства взаимодействия с пользователем графический интерфейс. Достоинства данного интерфейса в лёгкости освоения пользователем и объектной ориентированности. Это позволяет ис-

пользовать его как основу для построения модели пользователя системы.

Поскольку единственным источником сведений о работе пользователя с интерфейсом является последовательность событий, возникающих в интерфейсе, а также вероятность последовательного наступления двух событий (её можно вычислить, зная журнал работы пользователя, подробнее о построении такого журнала см. [13]), то целесообразным представляется использование для моделирования работы пользователя с графическим интерфейсом использование модифицированного в п. 3 аппарата ресурсных сетей.

Любое приложение с графическим интерфейсом содержит большое количество окон. Каждое окно представляет собой набор компонентов управления, между которыми, с некоторой вероятностью перехода, переключается пользователь.

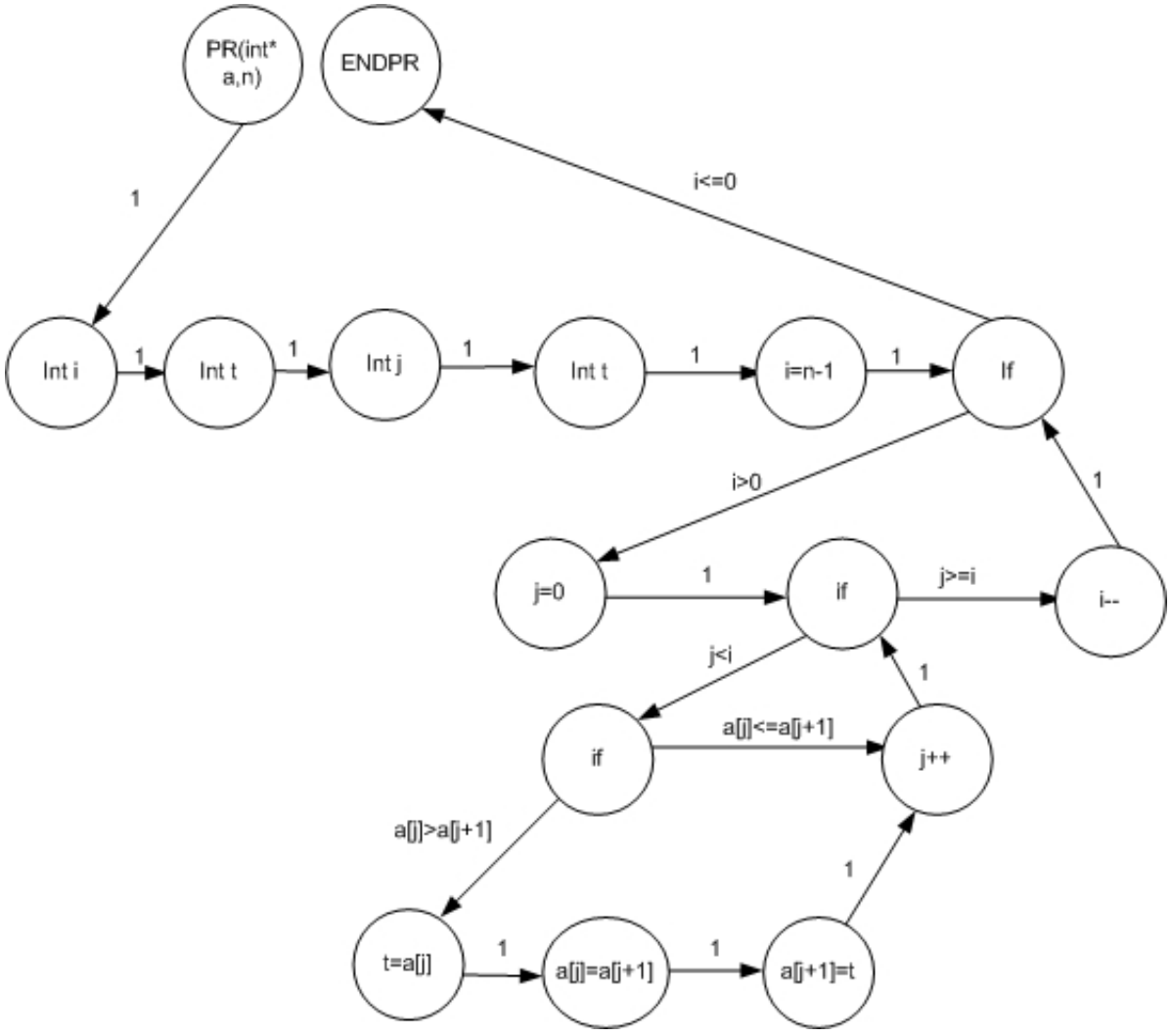

Рис. 1. Транспортная сеть для процедуры

Поэтому, каждое окно пользователя целесообразно представлять в виде ресурсной сети, образующей отдельный слой. Вершинами данной модифицированной ресурсной сети будут являться элементы управления окна, а дуги будут указывать направления переходов.

Каждой дуге такого графа будет сопоставлен предикат H. Как следует из формул (1)-(3), данный предикат принимает в случайном порядке значения $\{0;1\}$. Вычисление значения данного предиката основано на алгоритме, подобном арифметическому сжатию [14].

Рассмотренный нами слой будет отличаться от слоя реализации, рассмотренного в предыдущем пункте, только тем, что вместо транспортной сети на данном слое будет функционировать модифицированная ресурсная сеть.

Поэтому в дальнейшем такой слой будем называть слоем реализации окна графического интерфейса.

В большинстве реализаций графического интерфейса элемент управления также является специализированным окном, «элементами управления» в котором являются события этого компонента, об-

работка которых осуществляется при помощи процедур обработки событий.

Учитывая вышесказанное, для моделирования элемента управления также удобно использовать слой реализации окна графического интерфейса. С соответствующим элементом управления данный слой связан слоем вызова, вершиной-истоком в котором является вершина, соответствующая компоненту, а вершиной-стоком – вершина, соответствующая событию активизации компонента управления на форме.

Если некоторому событию сопоставлена процедура обработки, то модель данной процедуры связывается с событием также с использованием слоя вызова. В слое вызова вершиной-истоком служит вершина, соответствующая данному событию, а вершиной-стоком – вершина-исток слоя реализации процедуры обработки события.

Выводы

В данной статье предложена новая потоковая модель – гибридные ресурсные сети, являющиеся интеграцией известных подходов, основанных на транспортных и ресурсных сетях.

Для данной модели введено понятие состояния модели, как вектора состояний вершин. Последовательность состояний таких векторов позволяет получить двумерный сигнал – изображение, как для программы, так и для её спецификации. Анализ данного изображения известными методами анализа изображений позволяет установить степень подобия между поведением программы и спецификации. Использование гибридных ресурсных сетей в системах автоматизации тестирования позволит снизить размерность анализируемых данных и применять принцип модульности тестирования в системах автоматизированного тестирования.

Литература

1. Основы инженерии качества программных систем [Текст]: 2-е изд., перераб. и доп. / Ф.И. Андон, Г.И. Коваль, Т.М, Коротун, Е.М. Лаврищева, В.Ю. Суслов. – К.: Академпериодика, 2007. – 672 с.

2. Clarke, E. Model checking [Text] / E. Clarke, O. Grumberg, D. Peled // The MIT Press. – 2000. – 142 p.

3. Вельдер, С.Э. Введение в верификацию автоматных программ на основе метода Model checking [Текст] / С.Э. Вельдер, А.А. Шалыто. – СПб: СПБГУ ИТМО, 2005 . – 52 с.

4. Watir: automated-testing.info [Електронний ресурс]. – Режим доступу: http://www.automated-testing.info/tools/watir [Назва з контейнера].

5. Canoo WebTest: automated-testing.info[Електронний ресурс]. – Режим доступу: http://www.automated - testing.info/tools/canoo-webtest [Назва з контенйера].

6. Selenium: automated-testing.info [Електронний ресурс]. – Режим доступу: http://www.automated-testing.info/tools/sele-nium [Назва з контейнера].

7. Кузнецов, О.П. Однородные ресурсные сети I. Полные графы [Текст] / О.П. Кузнецов // Автомат. и телемех. – 2009. – № 11. – С. 136 – 147.

8. Жилякова, Л.Ю. Рекурсивный поиск в динамиче-ской ассоциативной ресурсной сети [Текст] // Труды конференции OSTIS-2011. – С. 155 – 169.

9. Форд, Л.Р. Потоки в сетях [Текст] / Л.Р. Форд, Д. Фалкерсон. – М.: Мир, 1966. – 320 с.

10. Ahuja, R.K. Network Flows: Theory, Algorithms and Applications [Text] / R.K. Ahuja, T.L. Magnati, J.B. Orlin. – Prentice Hall, New Jersey, 1993. – 224 p.

11. Бейзер, Б. Тестирование черного ящика Технологии функционального тестирования программно-го обеспечения и систем [Текст] / Б. Бейзер. – СПб.: Питер, 2004 – 318 с

12. Пригожев, А.С. Визуализация данных для анализа программного обеспечения с использованием экспертной системы [Текст] / А.С. Пригожев // Искусственный интеллект. – 2009. – № 3. – С. 347 – 351.

13. Пригожев, А.С. Реализация решателя и базы знаний экспертной системы для планирования дей-ствий пользователя при разработке программ [Текст] / А.С. Пригожев // Радиоэлектроника и информатика. – 2005. – № 4. – С. 110 – 116.

14. Методы сжатия данных. Устройство архиваторов, изображений и видео [Текст] / Д. Ватолин, А. Ратушияк, М. Смирнов, В. Юркин. – М.: ДИАЛОГ-МИФИ, 2003. – 384 с.

Поступила в редакцию 7.02.2012

Рецензент: д-р техн. наук, проф., проф. кафедры А.В. Дрозд, Одесский национальный политехнический университет, Одесса, Украина.

ВИКОРИСТАННЯ РЕСУРСНИХ МЕРЕЖ ДЛЯ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

О.С. Пригожев

Основним завданням тестування є встановлення відповідності програми та її специфікацій. Цей процес є досить складним і потребуючим автоматизації. Основними проблемами при цьому є велика розгалуженість процесу рішення завдання, велика кількість тестових наборів, а також відсутність формальних критеріїв, що дозволяють однозначно оцінити відповідність програми її специфікації. У представленій роботі пропонується використання для тестування програмного забезпечення нової потокової моделі - гібридних ресурсних мереж. Використання даного механізму дозволяє істотно спростити та автоматизувати процес тестування програмного забезпечення, звівши його до завдання пошуку подоби між зображеннями. Використання пропонованого підходу в автоматизованих системах тестування дозволяє формалізувати критерії, по яких можливо встановити відповідність між програмою та специфікацією.

Ключові слова: тестування програм, підтримка розробника, автоматизація тестування, ресурсні мережі.

RESOURCE NETWORKS USE FOR TESTING SOFTWARE

O.S. Prygozhev

The main objective of testing is to establish a compliance of program and its specification. This process is quite complex and requires automation. The main problems are large process branching of solving the problem, large number of test kits, and the absence of formal criteria for simultaneous assessment of program and its specification compliance. In the present paper, we propose to use hybrid resource networks for software testing of the new threading model. The use of this mechanism allows simplifying and automation of the software testing process, reducing it to the problem of finding the similarity between images. Using the proposed approach for automated test systems allows formalizing the criteria by which it is possible to establish a correspondence between program and specification.

Keywords: software testing, user support, automation testing, resource networks.

Пригожев Александр Сергеевич – канд. техн. наук, доцент кафедры системного программного обеспечения Одесского национального политехнического университета, Одесса, Украина.