Dyachenko Evgeny Anatolievich, postgraduate, dyachenko.evgeni@gmail.com, Russia, Tula, Tula state University

УДК 681.5 (519.95)

ОЦЕНКА ЭФФЕКТИВНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РОБОТА С ИСПОЛЬЗОВАНИЕМ СЕТЕЙ ПЕТРИ-МАРКОВА

Е.В. Ларкин, В.В. Котов, Н.А. Котова

Рассмотрены вопросы оценки эффективности разрабатываемого программного обеспечения с использованием сетей Петри-Маркова. Предложена методика формирования соответствующей сети. Описана разработанная программа создания сетей Петри-Маркова. Предложен формат хранения структуры и параметров сети.

Ключевые слова: сеть Петри-Маркова, информационно-управляющая система робота, обработка в реальном времени.

Для современных интеллектуальных робототехнических комплексов характерно наличие развитой многоуровневой информационно-управляющей системы, включающей в себя множество различных сенсоров и один или несколько параллельно функционирующих процессоров, обеспечивающих цифровую обработку данных в реальном масштабе времени [1]. Учитывая, что результаты обработки сенсорной информации непосредственно используются для выработки управляющих воздействий на исполнительные механизмы робота, одним из главных требований, предъявляемых к применяемым алгоритмам, является время их выполнения. Таким образом, одной из важнейших задач, возникающих при проектировании программного обеспечения робота, является контроль и обеспечение требуемых показателей его временной эффективности. Предлагается для её решения использовать модель эффективности, сформированную на основе сети Петри-Маркова (СПМ) [2].

Информационно-управляющую систему робота, включающую внутреннее программное обеспечение, можно представить в виде автомата с конечным числом состояний. В этом случае применение механизма сетей Петри-Маркова позволяет решить две важные задачи в области оценки эффективности программного обеспечения: ответить на вопросы о принципиальной достижимости состояний системы, соответствующих заданным требованиям (в том числе и ошибочных, сбойных состояний), и спрогнозировать моменты переключения в указанные состояния.

Любое программное приложение может быть представлено иерархическим делением (рис. 1). Иерархия программного продукта может быть

отражена следующими уровнями:

0 – уровень программного продукта, элементами которого являются программы информационных подсистем исполнительного, тактического и стратегического уровней, включающие систему ввода-вывода сенсорной информации робота, связи с механотронными элементами, систему обработки данных, и т.п.;

1 – уровень взаимодействующих систем программного продукта, например для системы технического зрения, элементами которой являются подсистема предварительной обработки видеоданных, сегментации и классификации, подсистема распознавания и анализа среды, подсистема моделирования среды и т.п.

2 – уровень взаимодействующих подсистем, включающий программные модули, интерпретирующие подсистемы, которые могут взаимодействовать между собой как в рамках одной системы и подсистемы, так и осуществлять межсистемное взаимодействие;

3 – уровень взаимодействующих модулей на уровне функций, процедур и классов, в зависимости от используемых при разработке программного продукта принципов;

4 – уровень взаимодействующих элементов – конечных управляющих операторов программного кода, в том числе выраженных предикатами первого порядка алгоритмических ветвлений.

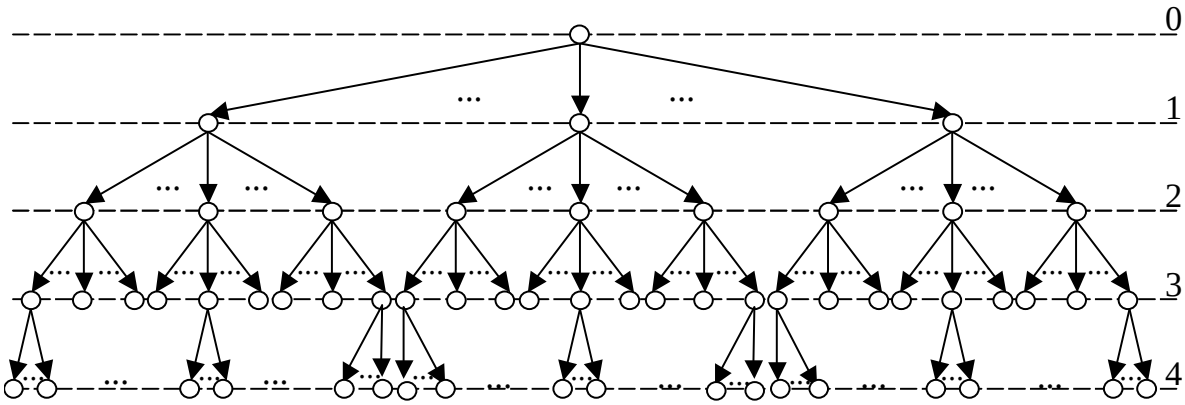

Рис. 1. Иерархическая структура программного обеспечения

Очевидно, что для многих систем возможно и дальнейшее иерархическое деление, но в контексте решаемой задачи такое деление не имеет смысла. Таким образом, объект, представленный на i-м иерархическом уровне как «система», для $(i-1)$-го уровня иерархии представляет собой «элемент». Между элементами данного иерархического уровня существуют определенные структурные связи, что и делает совокупность элементов системой.

Модели систем на каждом иерархическом уровне идентичны и могут быть представлены в виде графа. Часть элементов может принадлежать к ряду систем, что обусловлено принципами многократного использования

программного кода. Таким образом, модели подсистем программного обеспечения на каждом иерархическом уровне идентичны, что позволяет применить к описанию процесса функционирования программного обеспечения единый для всех уровней математический аппарат.

Объекты любого уровня программного продукта могут реализовывать параллелизм вычислений для ускорения работы алгоритма. В связи с этим, выделяют:

последовательные структуры, в которых осуществляются последовательные переходы между объектами рассматриваемого уровня иерархии;

параллельные структуры, в которых моделируется параллелизм работы объектов рассматриваемого уровня иерархии;

параллельно-последовательные структуры, в которых осуществляется моделирование сложного объекта программного обеспечения, переменно использующего параллелизм.

Рекомендуется следующая методика формирования СПМ, моделирующей эффективность в программном обеспечении:

1) Оценка общего уровня эффективности через показатели на разных уровнях иерархии программного продукта с помощью байесовской сети доверия.

2) Декомпозиция уязвимого участка программного кода, требующего детального исследования, на элементы выбранного уровня иерархии (программные модули/функции и классы/операторы). Для каждого элемента должен иметься комплект документации, описывающей его допустимые условия эксплуатации, параметры, характеристики, в том числе и характеристики эффективности.

3) Выделение групп видов сбоев для каждого элемента перечня возможных типов сбоев и возможных состояний.

4) Формирование для каждого элемента элементарной цепи состояний, включающей возникновение сбоев каждого типа, переход элемента из состояния в состояние и т.п.

5) Составление элементарной подсети Петри-Маркова, определяющей структуру модели процесса выполнения алгоритма.

6) Выделение на модели элементарного процесса переходов, требующих взаимодействия с другими элементарными процессами, включая взаимодействия на выбранном уровне иерархии.

7) Определение начального и конечного состояния элементов системы данного уровня – то есть точек входа и выхода в программные модули/функции и классы/совокупности операторов. Выделение на элементарных процессах переходов, предшествующих начальному и следующих за конечным состояниями.

8) Объединение элементарных подсетей в единую сеть через переходы элементарных подсетей, требующих взаимодействия, а также через

начальные и конечные переходы, из указанных переходов формируются непримитивные переходы.

9) Назначение априорных вероятностей при переключении каждого из элементов в сопряженные состояния (например, исходя из опыта эксплуатации аналогичных участков программного кода в аналогичных условиях); определение плотностей распределения времени переключения в сопряженные состояния с использованием требований спецификации на программный продукт. Составление матрицы вероятностей и матрицы плотностей распределения.

10) Определение логики взаимодействия элементарных процессов на непримитивных переходах. Составление матрицы вероятностей.

11) При необходимости (наличие в системе множества начальных состояний) определение вектора вектор, определяющего вероятность начала процесса в одном из переходов из множества начальных.

Моделирование эффективности участка программного кода сводится к построению сети Петри-Маркова по схеме межмодульного взаимодействия и анализу временных и вероятностных характеристик траекторий сбоев.

В случаях, когда основной целью моделирования алгоритма, исполняемого программной системой, является оценка временных затрат на его выполнение, в качестве объектов анализа эффективно использовать логические элементы алгоритма, то есть не формализованные описания шагов выполнения программы, не включенные в иерархию, изображенную на рис. 1. В таких случаях элементарную подсеть Петри-Маркова процесса возможно построить на стадии задания спецификации программного обеспечения и использовать её для верификации корректности реализации алгоритма.

Для реализации предложенного подхода было разработана программа «Редактор сетей Петри-Маркова», обеспечивающая визуальное построение и редактирование сетевых структур в соответствии с теорией сетей Петри-Маркова. Программа имеет стандартизированный, интуитивно понятный интерфейс (рис. 2), предоставляющий пользователю ряд типовых средств, обеспечивающих возможность добавления в сеть новых вершин типа «Позиция» или «Переход», установления связей между вершинами, с контролем типа связываемых вершин, удаления ошибочно внесённых вершин и связей, редактирования свойств каждой из вершин (перечень свойств определяется типом вершины сети). Программа в процессе редактирования обеспечивает контроль корректности структуры создаваемой сети и параметров.

Сформированную структуру сети Петри-Маркова и набор параметров отдельных её вершин можно сохранить в виде файла на диск. В дальнейшем этот файл может быть вновь загружен в программу для дальнейшего редактирования или иного использования. Для хранения СПМ был

разработан специальный формат файла, имеющий следующую структуру.

Строка_1 — общее описание состава сети Петри-Маркова:

<Количество позиций> — целое число; <Количество переходов> — целое число; <Количество узлов дуг> — целое число; <Количество дуг> — целое число; <Признак наличия в файле свойств сети> — 1 – есть, 0 – нет.

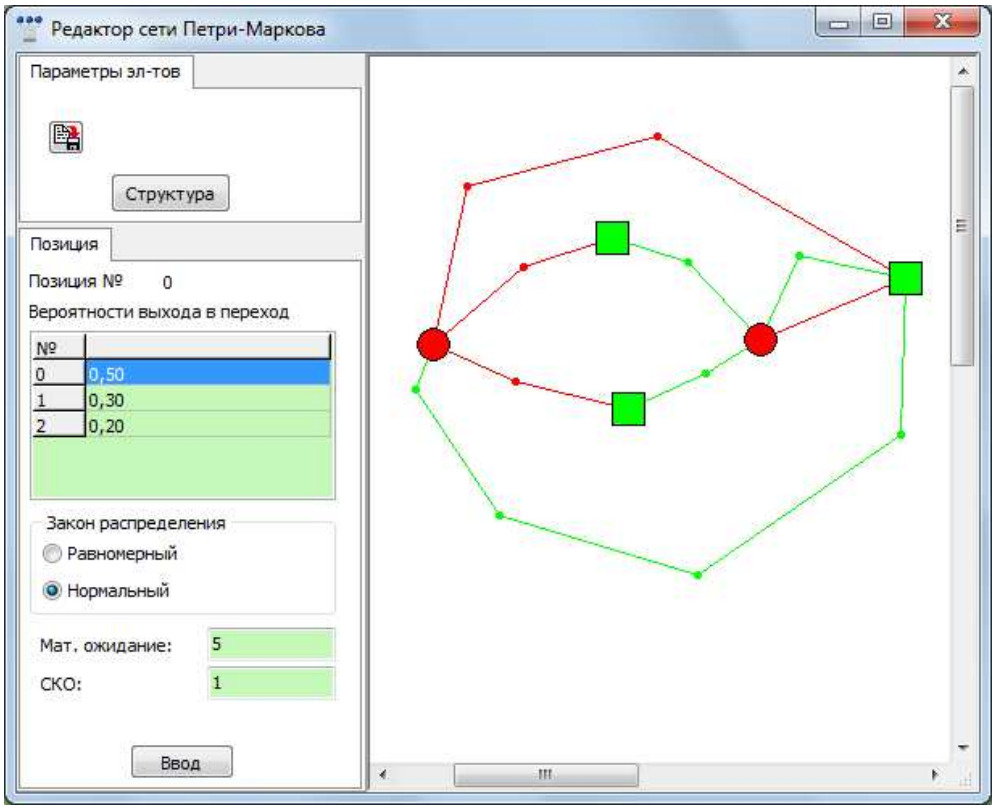

Рис. 2. Внешний вид программы в режиме редактирования параметров

Строка_2 — описание позиций:

<X-координата позиции> — целое число; <Y-координата позиции> — целое число — экранные координаты позиций (количество строк соответствует ранее указанному количеству позиций).

Строка_3 — описание переходов:

<X-координата перехода> — целое число; <Y-координата перехода> — целое число — экранные координаты переходов (количество строк соответствует ранее указанному количеству переходов).

Строка_4 — описание узлов дуг:

<X-координата перехода> — целое число; <Y-координата перехода> — целое число — экранные координаты узлов дуг (количество строк соответствует ранее указанному количеству узлов дуг).

Строка_5 — описание структуры связей дуг сети:

<Тип0> — целое число — определяет тип объекта, который являлся начальной точкой для сложной дуги, состоящей из нескольких отрезков (1 — позиция, 2 — переход); <Индекс1> — целое число — идентификацион-

ный номер объекта, являющегося начальным для текущего отрезка дуги; <Тип1> — целое число — тип объекта, являющегося начальным для текущего отрезка дуги (1 — позиция, 2 — переход, 3 — узел дуги); <Индекс2> — целое число — идентификационный номер объекта, являющегося конечным для текущего отрезка дуги; <Тип2> — целое число — тип объекта, являющегося конечным для текущего отрезка дуги (1 — позиция, 2 — переход, 3 — узел дуги);

Строка_6 — описание свойств сохраняемых позиций:

<Индекс> — целое число — идентификационный номер сохраняемой позиции; <ИсхДуг> — целое число — количество исходящих дуг из данной позиции;

Строка_7 — описание выходов из позиции в тот или иной переход:

<ИндексДуги > — целое число — идентификационный номер дуги, выходящей из текущей позиции в переход; <Вероятность> — вещественное число — вероятность выхода по этой дуге; <Тип закона> — целое число — тип закона распределения времени пребывания фишки в позиции при выборе выхода по этой дуге; <Матожидание> — вещественное число — математическое ожидание закона распределения; <СКО> — вещественное число — среднеквадратическое отклонение закона распределения.

Строка_8 — описание свойств сохраняемых переходов:

<Индекс> — целое число — идентификационный номер сохраняемого перехода; <ИсхДуг> — целое число — количество исходящих дуг из данного перехода; <ВхДуг> — целое число — количество входящих дуг в данный переход

Строка_9 — описание логики срабатывания перехода:

<ИндексДуги> — целое число — идентификационный номер дуги, выходящей из текущего перехода в позицию; <СДНФ> — строка символов — логическая функция в совершенной дизъюнктивной нормальной форме, определяющая условие срабатывания перехода по данному выходу

Строка_10 — описание входящих в переход дуг:

<ИндексДуги> — целое число — индекс входящей в переход дуги.

Предложенный подход позволяет оценить временную эффективность программного обеспечения робототехнического комплекса, и корректность логики взаимодействия параллельных процессов обработки сенсорной информации и формирования управляющих воздействий на элементы исполнительной системы.

Список литературы

1. Ларкин Е.В., Котова Н.А. Проектирование информационных систем роботов с использованием сетей Петри-Маркова: учеб. пособие. Тула: Изд-во ТулГУ, 2008. 158 с.

2. Ларкин Е.В., Котов В.В., Котова Н.А., Соколов В.А. К вопросу о

моделировании отказоустойчивых систем с помощью сетей Петри-Маркова // Фундаментальные исследования. №5. 2007. С. 74-78.

Ларкин Евгений Васильевич, д-р техн. наук, проф., зав. кафедрой, elarkin@mail.ru, Россия, Тула, Тульский государственный университет,

Котов Владислав Викторович, д-р техн. наук, проф., vkotov@list.ru, Россия, Тула, Тульский государственный университет,

Котова Наталья Александровна, канд. техн. наук, доцент, nkotova@inbox.ru, Россия, Тула, Тульский государственный университет

ESTIMATION OF ROBOT SOFTWARE EFFICIENCY USING PETRI-MARKOV NETWORKS

E.V. Larkin, V.V. Kotov, N.A. Kotova

Questions of an estimation of efficiency of the developed software using Petri-Markov networks are considered. The technique of formation of the corresponding network is offered. The developed program for Petri-Markov network creation and editing is described. The format of storage of structure and network parameters is offered.

Key words: Petri-Markov network, information and control system of the robot, real-time signal processing

Larkin Evgeny Vasilievich, doctor of technical science, professor, manager of department, elarkin@mail.ru, Russia, Tula, Tula State University,

Kotov Vladislav Viktorovich, doctor of technical science, professor, vkotov@list.ru, Russia, Tula, Tula State University,

Kotova Natalia Aleksandrovna, candidate of technical science, docent, nkotova@inbox.ru, Russia, Tula, Tula State University