УДК 519.6 URL: https://ptsj.bmstu.ru/catalog/icec/auto/994.html

РАЗРАБОТКА МЕТОДОВ ВЕРИФИКАЦИИ ЭЛЕМЕНТОВ ДИСКРЕТНО-СОБЫТИЙНОЙ МОДЕЛИ

А.С. Горохова a.s.gorokhova@yandex.ru
Л.А. Завадская zavadskayalus@gmail.com
М.А. Новичкова marie.nov2104@gmail.com

МГТУ им. Н.Э. Баумана, Москва, Россия

Рассмотрено моделирование процессов в дискретно-событийной системе (ДСС) для управления группой мобильных роботов, методы верификации модели объекта управления и спецификации ДСС. Поскольку при построении модели могут возникать ошибки, дальнейшее моделирование на некорректно построенной модели может приводить к дополнительному объему работы, излишней трате вычислительных ресурсов и иметь недопустимый результат. Предложена возможность повысить качество системы за счет применения валидации элементов ДСС с использованием программных инструментов анализа сети Петри и корректности спецификации. Представлена двухэтапная проверка ДСС: на уровне построения модели объекта управления в целом и на уровне конкретной спецификации поведения системы. На первом этапе моделируемую систему представляют в виде сети Петри и с помощью построения и анализа дерева достижимости, ограниченного до конечного размера, проверяют сеть на соответствие свойствам безопасности, живости, достижимости и связности. На втором этапе модель поведения системы проверяют на корректность переходов между событиями и контролируют связность цепочки событий. Предложена практическая реализация системы верификации, выполненная на языке программирования C++. Результаты могут служить основой для разработки системы, позволяющей автоматизировать процесс проектирования систем группового управления роботами.

Ключевые слова: дискретно-событийная система, конечные автоматы, группа роботов, сети Петри, дерево достижимости, спецификация поведения, логическая структура последовательности строк, верификация

Введение. Группы мобильных роботов находят практическое применение при решении широкого спектра задач. Преимущества группового применения роботов очевидны: больший радиус действия, достигаемый в результате рассредоточения роботов по рабочей зоне, расширенный набор выполняемых функций, достигаемый благодаря установке на каждого робота индивидуальных исполнительных устройств. Поэтому сложнейшие задачи специальной робототехники — участие в боевых и спасательных операциях, сборка сложных конструкций в космосе и масштабное исследование поверхности планеты — могут эффективно решаться роботами только при их групповом управ-

 ISSN 2541-8009

лении [1]. Однако при этом возникают новые проблемы группового управления и коммуникации, связанные с организацией группового взаимодействия роботов [2, 3]. При решении задачи управления группой роботов верхнеуровневый процесс функционирования может быть представлен последовательностью изменений его состояний, связанных с наступлением определенных событий в некоторые моменты времени. Это позволяет рассматривать группу мобильных роботов как *дискретно-событийную систему* (ДСС).

Поведение ДСС рассматривается с самых общих позиций как некоторый генератор последовательностей событий из конечного множества *событий E*. Событие $e \in E$ — это абстракция для множества фактов наблюдения «жизни» дискретно-событийных систем. События мгновенны, их появление спонтанно, происходят они в непредсказуемые моменты времени, поэтому все, что можно наблюдать, — **их последовательности, которые представлены строками**. $L(G)$ — язык, моделирующий поведение ДСС, всевозможные состояния и события, где G — объект управления, генератор для $L(G)$. Отличительная особенность ДСС — это разделение в исходных данных модели объекта G и требований к его поведению, задающихся *спецификацией K*, и затем постановка и решение задачи синтеза S — *супервизора* (управляющей компоненты ДСС), обеспечивающего поведение G в соответствии с K [4]. Как правило, G и K определяются конечными автоматами, языками или сетями Петри [5]. Супервизор S должен обеспечивать переход системы в целевое состояние и не допускать ее блокировки, т. е. супервизор должен быть неблокирующим.

Первый этап верификации элементов ДСС — анализ соответствующей объекту управления сети Петри. Второй — анализ корректности последовательности строк, задающей требуемое поведение объекта управления. Верификация на втором этапе основана на алгоритме $R(S)$, предложенном А.А. Амбарцумяном [4].

Модели дискретно-событийных систем. На рассматриваемом уровне абстракции события условны и неделимы, однако «внутри себя» могут содержать последовательности команд для их выполнения (запуск механизма, подъем манипулятора и пр.).

Событийную модель можно представить как конечный автомат. Для того чтобы задать конечный автомат, необходимо знать множество событий e, множество состояний q и их связь. Задаются также начальное состояние q_0 и конечное. Конечных состояний в автомате может быть несколько.

Конечный автомат удобно визуализировать в виде ориентированного графа, в котором могут присутствовать петли, а веса на ребрах обозначают допустимое событие, связывающее две вершины-состояния. Иные связки со-

стояний не допускаются. Такой способ визуализации дает возможность понять, какие состояния могут быть связаны и через какие события.

Для описания пути в конечном автомате используется спецификация — некоторое множество связей событий и состояний. Спецификация может задаваться с помощью строки событий. Каждый элемент такой строки — событие, связывающее последовательные состояния. Если очередное событие в строке не может связать текущее состояние с каким-нибудь из допустимых для него по спецификации, причем это состояние не конечное, строку считают неверной. Если строка не приводит к конечному состоянию или первое событие в ней не является начальным, согласно ДСС, ее также считают неверной. Для описания поведения ДСС необходимо проверить корректность на уровне задания требований к ней, а также проанализировать корректность требуемого поведения. Затем, получив набор строк событий, их последовательности, задающей сценарий в ДСС — логическую структуру последовательности строк (ЛСПС), необходимо удостовериться в их корректности с точки зрения ограничений, заданных внутри ДСС.

Логическая структура последовательности строк — представление графа переходов конечного автомата в виде строк символов состояний и событий этого графа. Каждая строка ЛСПС соответствует простому пути графа переходов и состоит из имен состояний и событий, взвешивающих ребра, в порядке их следования в этом пути. Также строка может кодировать циклы и ветвления.

Порядок следования таких путей в графическом описании строки задают стрелками с индексами. Стрелка вверх — с индексом n означает исходящее ребро (в n). Исходящее ребро обязательно завершает простой путь (служит переходом на новую строку событий). Стрелка вниз с индексом n означает входящее ребро (из n). Входящее ребро располагают перед началом строки состояний (перед состоянием, в которое входит помеченное раннее стрелкой исходящее ребро). Если после некоторого состояния идет ветвление, ветвление заканчивает строку, а соответствующие события, через которые будут выполнены переход из некоторого предшествующего состояния, заключают в фигурные скобки {...}.

Таким образом, следующая ЛСПС описывает схему на рис. 1:

$$\downarrow^1 q_1 e_{3-1} q_2 e_{3-2} q_3 e_{3-3} \downarrow^4 q_4 \left\{ e_{3-8} \uparrow^2, e_{3-4} \uparrow^3 \right\};$$

$$\downarrow^2 q_{10} e_{3-2} q_{11} e_{3-9} \uparrow^1;$$

$$\downarrow^3 q_5 e_{3-5} q_6 e_{3-6} q_7 e_{3-7} q_8 e_{3-5} q_9 e_{3-3} \uparrow^4 .$$

 ISSN 2541-8009

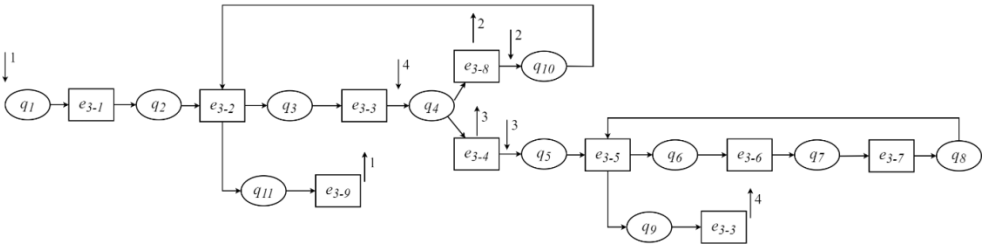

Рис. 1. Пример заданной спецификации

Каждое событие будет ассоциировано с номером строки, которой оно принадлежит, и номером выходящей из нее строки.

Самым распространенным способом представления ДСС, наиболее наглядным для визуального представления, является сеть Петри [7].

Анализ сетей Петри. Рассмотрим основные компоненты сетей Петри. Формальным образом сеть Петри определяется как четверка [9, 10] — (S, T, W, μ_0), где S — конечное множество позиций; T — конечное множество переходов; W: $(S \times T) \cup (T \times S) \to Z^+$ — мультимножество дуг; μ_0: $S \to Z^+$ — начальная маркировка сети. На рис. 2 представлена сеть Петри, для которой корректна спецификация с рис. 1. Маркировка μ — присвоение фишек позициям сети Петри. Фишки, как и позиции и переходы, — базовые понятия сети Петри — используются для определения выполнения сети Петри и присваиваются позициям. Количество и положение фишек может изменяться при выполнении сети Петри.

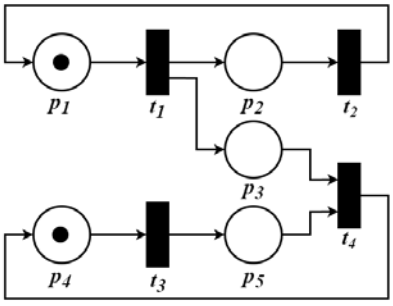

Рис. 2. Пример сети Петри

Выделяют следующие свойства сетей Петри.

1. Достижимость маркировки для сети — существует такая последовательность переходов, при запуске которой можно получить такую маркировку сети из начальной маркировки μ_0.

2. Живость — сеть не имеет тупиковых состояний, все ее переходы были «запущены» при функционировании моделируемого объекта.

3. Безопасность — все позиции сети Петри безопасны (в любой достижимой маркировке позиции не более одной метки).

4. Ограниченность — любая позиция сети Петри ограничена [8] (существует такое k, что в любой достижимой маркировке в данной позиции будет не более k меток).

5. Связность сети Петри — ориентированный граф, соответствующий сети Петри, является связным.

Существует два метода анализа сетей Петри: с использованием матричных уравнений и с использованием дерева достижимости. Для анализа указанных свойств сети Петри оптимально использовать дерево достижимости, поскольку для его хранения требуется меньше памяти, что особенно важно для моделей большой размерности. Дерево достижимости представляет собой множество достижимости сети Петри. Построение дерева осуществляется при определении множества значений позиций при последовательном срабатывании переходов. Для построения дерева необходимо ограничить его до конечного размера. Ограничение происходит тогда, когда все вершины являются *дублирующими*, *терминальными* и *внутренними* [5].

Существуют такие последовательности запусков переходов, для которых конечная маркировка последовательности μ' совпадает с начальной маркировкой последовательности μ, за исключением того, что имеет некоторые дополнительные метки в отдельных позициях. Запуская переходы в таких последовательностях некоторое количество раз можно получить бесконечное число меток. Чтобы ограничить дерево достижимости представим бесконечное число меток, получаемых из циклов такого типа символом ω. При наличии в дереве достижимости вершин, чьи маркировки содержат ω (бесконечное число меток), сеть не является ограниченной.

Для анализа на связность сети применяют методы обхода графа. В частности, использован алгоритм обхода в ширину. Если при обходе графа были посещены все вершины, то сеть, которой этот граф соответствует, является связной, иначе — не является. Подробный алгоритм построения дерева достижимости приведен в [3].

Алгоритмы верификации элементов ДСС. Рассмотренные методы верификации (анализ свойств сети Петри и верификация спецификации) применимы к компьютерной модели ДСС. Пример ее реализации приведен в [8]. В данной работе для создания дискретно-событийной модели использовались классы события Event, состояния State и компонентного автомата Automation библиотеки simple_des_model ([8]).

 ISSN 2541-8009

С помощью библиотеки simple_des_model были реализованы следующие методы-функции, предназначенные для компьютерных моделей.

1. Алгоритм построения конечного дерева достижимости.

Каждая вершина i дерева связывается с маркировкой $\mu[i]$ в которой число меток в позиции может быть либо неотрицательным целым, либо равным ω. Каждая вершина классифицируется или как граничная вершина, или как внутренняя. Граничными являются вершины, которые еще не обработаны алгоритмом; алгоритм превратит их в терминальные, дублирующие или внутренние вершины. Алгоритм начинает с определения начальной маркировки корнем дерева, т. е. граничной вершиной. До тех пор пока имеются граничные вершины, они обрабатываются алгоритмом.

Пусть x — граничная вершина, которую необходимо обработать. Если в дереве имеется другая вершина y, не являющаяся граничной и с ней связано та же маркировка, то вершина x — дублирующая. Если для маркировки $\mu[x]$ не один из переходов не разрешен, то вершина x — терминальная. Для всякого перехода $t_j \in T$, разрешенного в $\mu[x]$, создать новую вершину z дерева достижимости. Маркировка $\mu[z]$, связанная с этой вершиной определяется следующим образом: если $\mu[x]_i = \omega$, то $\mu[x]_j = \omega$. Если на пути от корневой вершины к x, существует вершина y, маркировка которой $\mu[y]$ меньше маркировки $\mu[z]$, то в позициях, в которых $\mu[x]_i > \mu[x]_j$, $\mu[x]_i = \omega$. В противном случае число меток в каждой позиции определяется правилами запуска перехода. Когда все вершины дерева являются терминальными, дублирующими или внутренними, алгоритм останавливается.

2. Реализация метода анализа сети на связность.

Метод bfs(des::Automation& model) получает в качестве параметра сеть Петри (model) в виде ссылки на объект класса Automation. Получаем множество всех вершин графа, объединяя множества позиций и переходов сети. Из множества выбираем первую вершину, от которой будем вести обход графа. Выбранную вершину записываем в очередь и во множество пройденных вершин. Далее, пока очередь не пустая, выбираем и удаляем первую вершину в очереди, каждую смежную с удаленной вершину, если ее еще нет в множестве посещенных вершин, записываем в очередь и во множество пройденных вершин. После окончания цикла сравниваем множества посещенных и пройденных вершин. Если множества равны, то сеть связная, возвращаем единицу. Иначе сеть не связная, возвращаем ноль.

3. Реализация метода обработки отдельной вершины дерева.

Метод analyse_node(Node* start, des::Automation& model) получает в качестве параметров указатель на анализируемую вершину и сеть Петри (model)

в виде ссылки на объект класса Automation. Декларируем словарь next, в котором будет храниться получаемая маркировка сети. В словарь ready_events записываем все возможные события для данного узла модели. Если словарь ready_events пуст, увеличиваем счетчик term терминальных вершин дерева на единицу. Иначе для каждого элемента в этом словаре выполняем следующие действия. Записываем в словарь next маркировку, получаемую при запуске этого элемента. В множество отработанных событий done_events записываем текущий элемент. Создаем следующий узел next_node как объект класса Node, которому в качестве родителя передаем узел start, а в качестве маркировки — словарь next. Проверяем, содержится ли уже вершина с параметрами узла next_node в контейнере закрытых вершин и не совпадает ли она с анализируемой вершине. Если такая вершина еще не встречалась, то в контейнер открытых вершин open добавляем следующую вершину с параметрами next_node; добавляем в конец дерева следующую вершину с параметрами next_node. Иначе, если параметры следующего узла равны параметрам начальной вершины дерева, увеличиваем счетчик dubl_start вершин, которые дублируют начальную, на единицу.

4. Реализация метода полного анализа.

Метод run_analyse(des::Automation& model) получает в качестве параметра сеть Петри (model) в виде ссылки на объект класса Automation. Декларируем счетчик *i*, значение которого равно нулю; словарь analysis_result, который будет возвращать данная функция анализа сети Петри, и в котором содержатся пары «ключ – значение», соответствующие характеристикам сети Петри, а именно: alive — 0, coherent — 0, safe — 0, reachable — 0, по умолчанию сеть неживая, недостижимая, несвязная, неограниченная; словарь mark, в котором будет храниться маркировка в виде «ключ – значение»; множество всех переходов сети Петри all_events. Получаем начальную маркировку сети: для всех состояний сети Петри, записываем в mark количество меток активности. Создаем начальную вершину дерева start как объект класса Node с маркировкой mark, записываем ее в корень дерева tree. Заносим маркировку начальной вершины в множество открытых вершин open. Далее, пока размер контейнера открытых вершин не равен нулю, выполняем следующие действия. Анализируем *i*-ю вершину дерева в функции analyse_node, добавляем *i*-ю вершину дерева в контейнер close (закрытых вершин дерева), из контейнера открытых вершин удаляем *i*-ю вершину дерева, увеличиваем значение счетчика *i* на единицу. После выхода из цикла, если терминальных вершин не найдено и были обработаны все переходы, в словаре analysis_result устанавливаем единичное значение ключа alive (т. е. сеть Петри — живая). Если счетчик dubl_start не нулевой, т. е. существует хотя бы одна вершина, которая

 ISSN 2541-8009

дублирует начальную вершину, в словаре analysis_result устанавливаем единичное значение ключа reachable (т. е. сеть Петри — достижимая). Проверяем сеть Петри на ограниченность: пусть она ограниченная, тогда в словаре analysis_result устанавливаем единичное значение ключа safe, далее проходим по всем маркировкам из множества close (пройденные вершины), если в позициях этих маркировок встречается бесконечное количество меток, т. с. является неограниченной, в словаре analysis_result устанавливаем значение ключа safe на ноль. Далее проверяем сеть Петри на связность с помощью функции bfs. Очищаем дерево tree и контейнеры open (открытых вершин), close (закрытых вершин), done_events (отработанных переходов), обнуляем счетчики term (терминальных вершин), dubl_start (дублирующих начальную вершину). Возвращаем словарь analysis_result, в котором теперь записаны актуальные свойства данной сети Петри.

5. Алгоритм верификации спецификации.

Правильная обработка спецификации подразумевает выявление всех путей (строк событий) в спецификации с учетом ветвлений и циклов.

Каждый найденный путь должен быть проверен на корректность переходов. Поскольку путь — это строка событий, для данной задачи используется адаптированная версия алгоритма $R(s)$, предложенного А.А. Амбарцумяном. Его кодирует функция *checkEventsRowCorrectness*. Для упрощения обработки спецификаций, содержащих ветвления, данная функция обрабатывает единственную строку событий, один из полных путей от начальной до конечной точки. Проверяется корректность переходов из события в событие, также может быть выведена информация о начальном и конечном состояниях строки для проверки корректности работы алгоритма.

6. Нахождение самих путей и проверка их корректности.

Осуществляется с помощью рекурсивной функции сборки прототипов полных путей в спецификации — списков входящих в них имен событий — *checkPathsInSpecification*, а также функции проверки связи событий в найденных списках имен событий *checkEventsRowCorrectness*.

7. Составление полного пути от начального до конечного состояния.

Функция *checkPathsInSpecification* собирает полный путь по следующему принципу: начало всегда в строке с индексом 1, следующая строка в пути указана как выходящая (output) из первой, следующая выходит из нее и т. д. Все события строк до ветвителя последовательно фиксируются в списке имен событий. При ветвлении строки ее заключительное событие будет иметь соответствующий флаг (маркер). Если строка заканчивается ветвлением, то каждый ветвитель помещается в список и с помощью рекурсивного вызова обрабатываются варианты строк событий для каждого из них. По достиже-

нии строки, выходящая из которой имеет индекс 1 (начальная), фиксируется конец пути и получившийся список передается далее, в функцию проверки индексного списка *checkEventsRowConnection*.

Примеры работы реализованных алгоритмов верификации. Рассмотрим несколько примеров.

1. ***Неограниченная и связная сеть Петри*** изображена на рис. 2. Дерево достижимости для нее приведено на рис. 3.

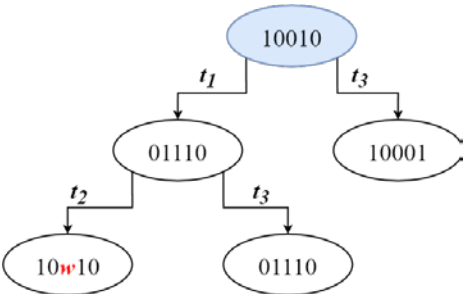

Рис. 3. Дерево достижимости для сети Петри (см. рис. 2)

Этап 1. При построении дерева достижимости получается маркировка, содержащая символ, который обозначает бесконечное число меток. Это говорит о том, что существует последовательность запусков переходов, при которой метки накапливаются в некоторой позиции. Следовательно, сеть не соответствует свойству ограниченности.

Поскольку сеть не пройдет проверку этапа 1, верификация любой спецификации, составленной для соответствующей сети ДСС, не будет иметь смысла, проверка этапа 2 не будет запущена.

2. ***Достижимая, ограниченная, связная сеть Петри*** изображена на рис. 4.

Этап 1. Анализ дерева достижимости (рис. 5) демонстрирует, что сеть Петри соответствует свойствам достижимости — из любой маркировки можно вернуться в начальную, ограниченности — нет маркировок с неограниченным числом меток, живости — все переходы могут быть запущены и нет терминальных вершин.

Этап 2. Рассмотрим процесс верификации нескольких спецификаций, составленных для прошедшей проверку ДСС, представленной на рис. 4.

1. Корректная спецификация с ветвлением для сети Петри на рис. 4

$\downarrow^1 p_1 \left\{t_1 \uparrow^2, t_3 \uparrow^3\right\}$; $\downarrow^2 p_2 t_2 p_3 t_3 p_4 t_4 p_1$; $\downarrow^3 p_4 t_4 p_1$.

При работе алгоритма будут найдены следующие пути событий:

 ISSN 2541-8009

1) t_1, t_2, t_3, t_4;

2) t_3, t_4.

Действительно, переход в лидирующие события в обоих случаях может происходить из начального состояния, и конечные события ведут в начальное состояние. Кроме того, согласно ДСС, для которой построена спецификация, существуют связи для любых двух соседних событий через некоторые состояния. Таким образом, спецификация пройдет проверку.

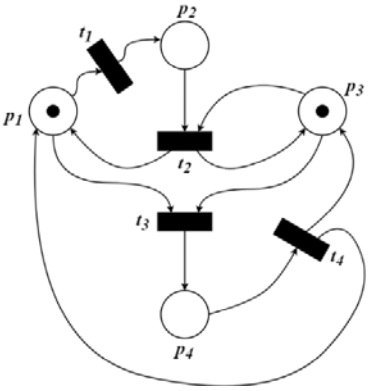

Рис. 4. Достижимая, ограниченная, связная сеть Петри

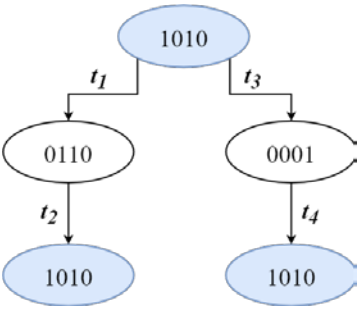

Рис. 5. Дерево достижимости для сети Петри, представленной на рис. 4

2. Корректная спецификация с циклом для сети Петри на рис. 4

$\downarrow^1 p_1 t_3 \uparrow^2$; $\downarrow^2 p_4 \left\{ t_4 \uparrow^3, t_4 \uparrow^4 \right\}$; $\downarrow^3 p_3 t_3 \uparrow^2$; $\downarrow^4 p_1$.

В данной спецификации может произойти зацикливание строк 2 и 3.

Алгоритм преобразует спецификацию в строки событий следующим образом:

1) t_3, t_4, t_3, t_4;

2) t_3, t_4.

3. Спецификация с ошибкой в порядке событий для сети Петри на рис. 4

$\downarrow^1 p_1 \left\{ t_1 \uparrow^2, t_3 \uparrow^3 \right\}$; $\downarrow^2 p_2 t_4 p_3 t_3 p_4 t_4 p_1$; $\downarrow^3 p_4 t_4 p_1$.

Данная спецификация будет развернута в строки событий:

1) t_1, t_4, t_3, t_4;

2) t_3, t_4.

Первая строка событий некорректна, поскольку не существует связи между событиями t_1, t_4. Спецификация не пройдет проверку.

Заключение. Проанализирован дискретно-событийный подход к описанию системы управления мобильными роботами. Рассмотрены известные модели дискретно-событийных систем: конечные автоматы, сети Петри, а также способ представления спецификации поведения. Предложена двухэтапная проверка модели ДСС: на уровне построения самой модели в целом и на уровне конкретной спецификации поведения системы. Реализован алгоритм анализ модели с учетом свойств сети Петри (живость, ограниченность, достижимость и связность) на языке C++. Кроме того, реализован алгоритм проверки заданной спецификации на корректность требуемого поведения. Представлены примеры, демонстрирующие работоспособность разработанного программного обеспечения.

В дальнейшем работа может служить основой для разработки системы, позволяющей автоматизировать процесс проектирования систем группового управления роботами. Также результаты могут быть использованы в промышленных системах управления группами роботов.

Литература

[1] Амбарцумян А.А., Потехин И.П. Групповое управление в дискретно-событийных системах. *Проблемы управления*, 2012, № 5, с. 46–53.

[2] Козов А.В., Волосатова Т.М., Тачков А.А. Направления автоматизации проектирования систем управления группами мобильных роботов. *Фундаментально-прикладные проблемы безопасности, живучести, надежности, устойчивости и эффективности систем. III Междунар. науч.-практ. конф., посв. 110-летию со дня рождения академика Н.А. Пилюгина*: *сб. тр*. Елец, Елецкий гос. ун-т им. И.А. Бунина, 2019, с. 335–339.

[3] Васильев И.А., Половко С.А., Смирнова Е.Ю. Организация группового управления мобильными роботами для задач специальной робототехники. *Информатика, телекоммуникации и управление*, 2013, № 1 (164), с. 119–123.

[4] Амбарцумян А.А. Супервизорное управление структурированными динамическими дискретно-событийными системами. *Автомат. и телемех*., 2009, № 8, с. 156–176.

[5] Амбарцумян А.А. Сетецентрическое управление на сетях Петри в структурированной дискретно-событийной системе. *УБС*, 2010, т. 30 (1), с. 506–535.

[6] Веретельникова Е.Л. *Теоретическая информатика. Теория сетей Петри и моделирование систем: учебное пособие*. Новосибирск, Изд-во НГТУ, 2018, 82 с.

[7] Котов В.Е. *Сети Петри*. Москва, Наука. Главная редакция физико-математической литературы, 1984, 160 с.

 ISSN 2541-8009

[8] Козов А.В. Реализация компьютерной модели дискретно-событийной системы группового управления мобильными роботами. *Экстремальная робототехника*, 2022, т. 1, № 1, с. 139–146.

[9] Амбарцумян А.А., Аристов В.В. Сети Петри как аппарат моделирования и синтеза супервизорного управления дискретно-событийных систем. *Динамика систем, механизмов и машин*, 2012, № 1, с. 217–220.

Поступила в редакцию 23.04.2024

Горохова Александра Сергеевна — студентка кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Россия.

Завадская Людмила Андреевна — студентка кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Россия.

Новичкова Мария Алексеевна — студентка кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Россия.

Научный руководитель — Козов Алексей Владимирович, ст. преподаватель, кафедра «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Россия. E-mail: alexey.kozov@bmstu.ru

Ссылку на эту статью просим оформлять следующим образом:
Горохова А.С., Завадская Л.А., Новичкова М.А. Разработка методов верификации элементов дискретно-событийной модели. *Политехнический молодежный журнал*, 2024, № 04 (93). URL: https://ptsj.bmstu.ru/catalog/icec/auto/994.html

DEVELOPING METHODS FOR VERIFYING THE DISCRETE-EVENT MODEL ELEMENTS

A.S. Gorokhova a.s.gorokhova@yandex.ru
L.A. Zavadskaya zavadskayala@student.bmstu.ru
M.A. Novichkova marie.nov2104@gmail.com

Bauman Moscow State Technical University, Moscow, Russia

The paper considers simulation processes in a discrete-event system (DES) to control a group of mobile robots, control object model verification methods, and the DES specification. Since errors could appear in the model construction, further simulation using an incorrectly constructed model could result in additional workload, excessive computation waste and lead to an unacceptable result. The paper proposes an opportunity to improve system quality by applying the DES elements validation, using the Petri net software tools analysis and specification correctness. It presents the DES two-stage verification at the level of constructing the control object model in general and at the level of concrete specification of the system behavior. At the first stage, the simulated system is represented as a Petri net. Through construction and analysis of the reachability tree constrained to a finite size, the network is being checked to meet the security, liveliness, reachability and connectivity properties. At the second step, the system behavior model is checked for correctness in transitions between the events and coherence of the event chain. The paper proposes practical implementation of the verification system realized in the C++ programming language. The results could serve as a basis in developing a system that allows automating the design process for the robot group control systems.

Keywords: discrete-event system, finite automata, robot group, Petri nets, reachability tree, behavior specification, logical string sequence structure, verification

Received 23.04.2024

Gorokhova A.S. — Student, Department of Robotics and Complex Automation Systems, Bauman Moscow State Technical University, Moscow, Russia.

Zavadskaya L.A. — Student, Department of Robotics and Complex Automation Systems, Bauman Moscow State Technical University, Moscow, Russia.

Novichkova M.A. — Student, Department of Robotics and Complex Automation Systems, Bauman Moscow State Technical University, Moscow, Russia.

Academic Advisor — Kozov A.V., Senior Lecturer, Department of Robotics and Complex Automation Systems, Bauman Moscow State Technical University, Moscow, Russia. E-mail: alexey.kozov@bmstu.ru

Please cite this article in English as:
Gorokhova A.S., Zavadskaya L.A., Novichkova M.A. Developing Methods for Verifying the Discrete-Event Model Elements. *Politekhnicheskiy molodezhnyy zhurnal*, 2024, no. 04 (93). (In Russ.). URL: https://ptsj.bmstu.ru/catalog/icec/auto/994.html

 ISSN 2541-8009