электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 519.876.5

Метод формализации программного обеспечения иерархическими сетями Петри

Пащенкова А.В., *студентка,*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Рудаков И.В., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
irudakov@bmstu.ru

Введение

Сложные программные системы характеризуются большим разнообразием взаимосвязей элементов, обработкой больших массивов информации, элементов конкуренции при использовании ресурсов ЭВМ. Проектирование систем обработки данных связано с синтезом оптимального состава модулей программного обеспечения на этапе технического проектирования программной системы. Структура программных модулей определяется обычно без учета альтернативных вариантов обработки, возможности параллельной реализации отдельных процедур, ветвей алгоритма и программных моделей. Возникает необходимость в разработке моделей системы для изучения взаимодействия ее элементов.

Одним из известных методов исследования процесса функционирования сложных систем является их формализация в виде сетей Петри. Данный математический аппарат позволяет формировать адекватные модели сложных систем и разрабатывать оптимальные алгоритмы решения задач.

Постановка задачи

Разработка программного обеспечения (ПО) – сложный многоэтапный процесс, включающий в себя этапы анализа, непосредственного написания, тестирования и внедрения. Проектирование программного обеспечения, как и любых других сложных систем, выполняется поэтапно с использованием блочно-иерархического подхода, который основан на разбиении сложной задачи большой размерности на последовательно и (или) параллельно решаемые группы задач малой размерности. Такой подход позволяет

разбивать исследуемый объект на компоненты требуемой степени детализации и проверять работу каждой из компонент посредством моделирования.

Моделирование работы отдельных частей алгоритма позволяет учесть особенности процесса функционирования разрабатываемого программного обеспечения. Модель при этом анализируется с поведенческой точки зрения, рассматривается в виде последовательности дискретных событий.

Анализ блок-схемы алгоритма программного обеспечения, формализованного с помощью сетей Петри, позволяет получить информацию о наличие взаимоблокировок, невыполнимых операций, зацикливаний.

Формализация блок-схем алгоритмов в терминах сети Петри

Сети Петри используются для формального моделирования программного обеспечения. Программа представляет два различных аспекта процесса: вычисление и управление. Вычисление связано с текущими арифметическими и логическими операциями, вводом и выводом, обычными манипуляциями над участками памяти и их содержимым. Управление же связано не со значениями или выполняемыми вычислениями, а только с порядком их выполнения.

Сети Петри удачно представляют структуру управления программ. Стандартный способ представления структуры управления программ – это блок-схема. Блок-схема представляет поток управления в программе. Блок-схема во многом подобна сети Петри: блок-схема представима в виде узлов двух типов (принятия решения, показанные ромбами, и вычисления, показанные прямоугольниками) и дуг между ними. Удобный способ выполнения блок-схемы – введение фишки, которая представляет текущую инструкцию. По мере выполнения инструкций фишка передвигается по блок-схеме.

Перевод блок-схемы в эквивалентную сеть Петри заменяет узлы блок-схемы на переходы сети Петри, а дуги блок-схемы – на позиции сети Петри. Каждая дуга блок-схемы соответствует точно одной позиции в сети Петри. Узлы блок-схемы представляются по-разному в зависимости от типа узла: вычисления или принятия решения.

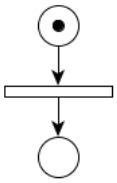
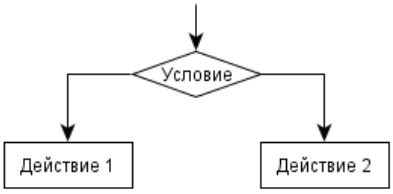
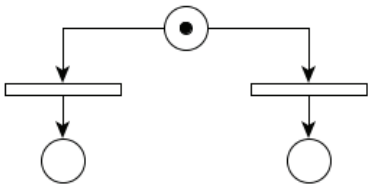
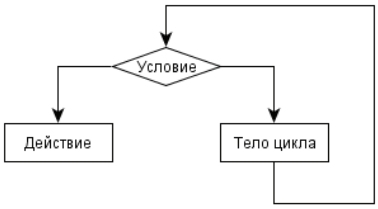
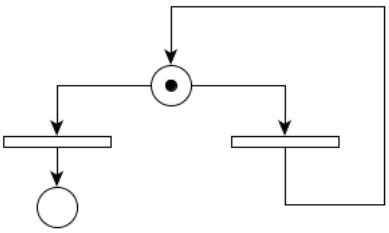
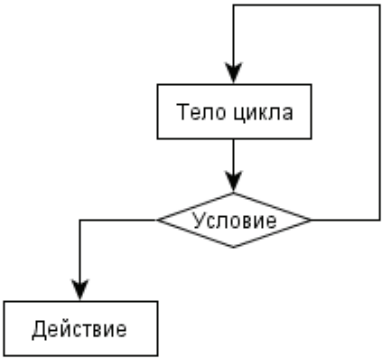
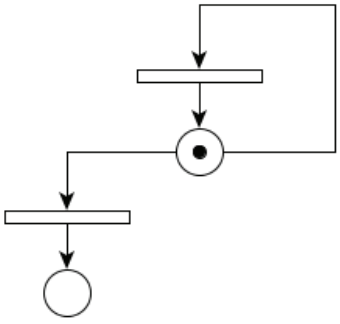
Фишка, находящаяся в позиции, означает, что счетчик команд установлен на готовность выполнения следующей инструкции. Каждая позиция имеет единственный выходной переход, за исключением позиции имеют по два выходных перехода, соответствующих истинному и ложному значению предиката.

Для интерпретации сети Петри необходимо интерпретировать каждый переход. Следует также отметить, что переходы для вычислений имеют один вход и один выход.

В приведенной ниже таблице (Таблица 1) в терминах сетей Петри представлены базовые управляющие структуры программного алгоритма. Иерархические сети Петри позволяют анализировать различные участки алгоритма с различной степенью детализации, что дает возможность на этапе формализации использовать уже существующие модели, интегрируя их с разрабатываемой сетью, а также локализовать обнаруженные при анализе ошибки путем уточнения проблемных участков [1].

Таблица 1

Базовые управляющие структуры алгоритма

Название управляющей структуры	Базовая управляющая структура	Сеть Петри
Следование		
Ветвление		
Цикл с предусловием		
Цикл с постусловием		

Методы анализа сети Петри

Существуют различные методы анализа сетей Петри. Одним из методов является имитационное моделирование. При запуске работы сети, сеть начинает свое функционирование, при котором одно событие сменяет другое. Последовательность событий представляет собой моделируемый процесс. Моделируя работу алгоритма таким образом, определяется, какие действия происходят, какие состояния предшествуют этим действиям. Как уже упоминалось ранее, одним из преимуществ сетей Петри является то, что процесс моделирования можно наблюдать интерактивно.

Другим методов анализа сетей Петри является дерево достижимости, представляющее собой множество достижимости сети. Корневая вершина представляет первоначальную маркировку. Из каждой вершины исходят дуги, соответствующие разрешенным переходам. Всякий путь в дереве, начинающийся в корне, соответствует допустимой последовательности переходов. Сеть Петри может иметь бесконечное дерево достижимости. Для получения дерева, которое можно считать полезным инструментом анализа, необходимо найти средства ограничения его до конечного размера.

Особенностью алгоритма построения конечного дерева достижимости является специальная классификация маркировок. Каждая вершина дерева классифицируется как граничная, терминальная, дублирующая или внутренняя вершина.

Граничными являются вершины, которые еще не обработаны алгоритмом. После обработки граничные вершины становятся либо терминальными, либо дублирующими, либо внутренними.

Маркировки, в которых нет разрешенных переходов, являются *терминальными* вершинами дерева достижимости. Другой класс маркировок – это маркировки, ранее встречавшиеся в дереве. Такие дублирующие маркировки называются *дублирующими* вершинами. Никакие последующие маркировки рассматривать не нужно, так как все они будут порождены из места первого появления дублирующей маркировки в дереве.

Для сведения дерева достижимости к конечному представлению используется еще одно средство. Для позиций, которые увеличивают число фишек некоторой последовательностью запусков переходов, можно создать произвольно большое число фишек, просто повторяя данную последовательность столько раз, сколько это нужно. Бесконечное число маркировок, получающихся из циклов такого типа, представляется с помощью специального символа *w*, который обозначает «бесконечность». Таким образом, в маркировке число фишек может быть либо неотрицательным целым, либо *w*.

Первый шаг алгоритма определяет начальную маркировку корнем дерева, то есть граничной вершиной. Последующие шаги направлены на обработку граничных вершин. До тех пор, пока имеются граничные вершины, они обрабатываются алгоритмом.

Пусть x – граничная вершина, которую необходимо обработать.

1. Если в дереве имеется другая вершина y, не являющаяся граничной, и с ней связана та же маркировка, $\mu[x] = \mu[y]$, то вершина x — дублирующая.

2. Если для маркировки $\mu[x]$ ни один из переходов не разрешен (то есть $\delta(\mu[x], t_j])$ не определено для всех $t_j \in T$), то x — терминальная вершина.

3. Для всякого перехода $t_j \in T$, разрешенного в $\mu[x]$ (т. е. $\delta(\mu[x], t_j)$ определено), создать новую вершину z дерева достижимости. Маркировка $\mu[z]$, связанная с этой вершиной, определяется для каждой позиции p_i следующим образом:

– Если $\mu[x]_i = w$, то $\mu[z]_i = w$.

– Если на пути от корневой вершины к x существует вершина y с $\mu[y] < \delta(\mu[x], t_j)$ и $\mu[y]_i < \delta(\mu[x], t_j)_i$, то $\mu[z]_i = w$.

– В противном случае $\mu[z]_i = \delta(\mu[x], t_j)_i$.

Дуга, помеченная t_j, направлена от вершины x к вершине z. Вершина x переопределяется как внутренняя, вершина z становится граничной. Когда все вершины дерева — терминальные, дублирующие или внутренние, алгоритм останавливается. На рис. 1 показана блок-схема алгоритма построения дерева достижимости.

Особенностью данного метода анализа алгоритма программного обеспечения, формализованного иерархической сетью Петри, является необходимость различать успешное завершение работы подсети от тупиковой ситуации. Для это предлагается вводить дополнительно конечное состояние сети [1].

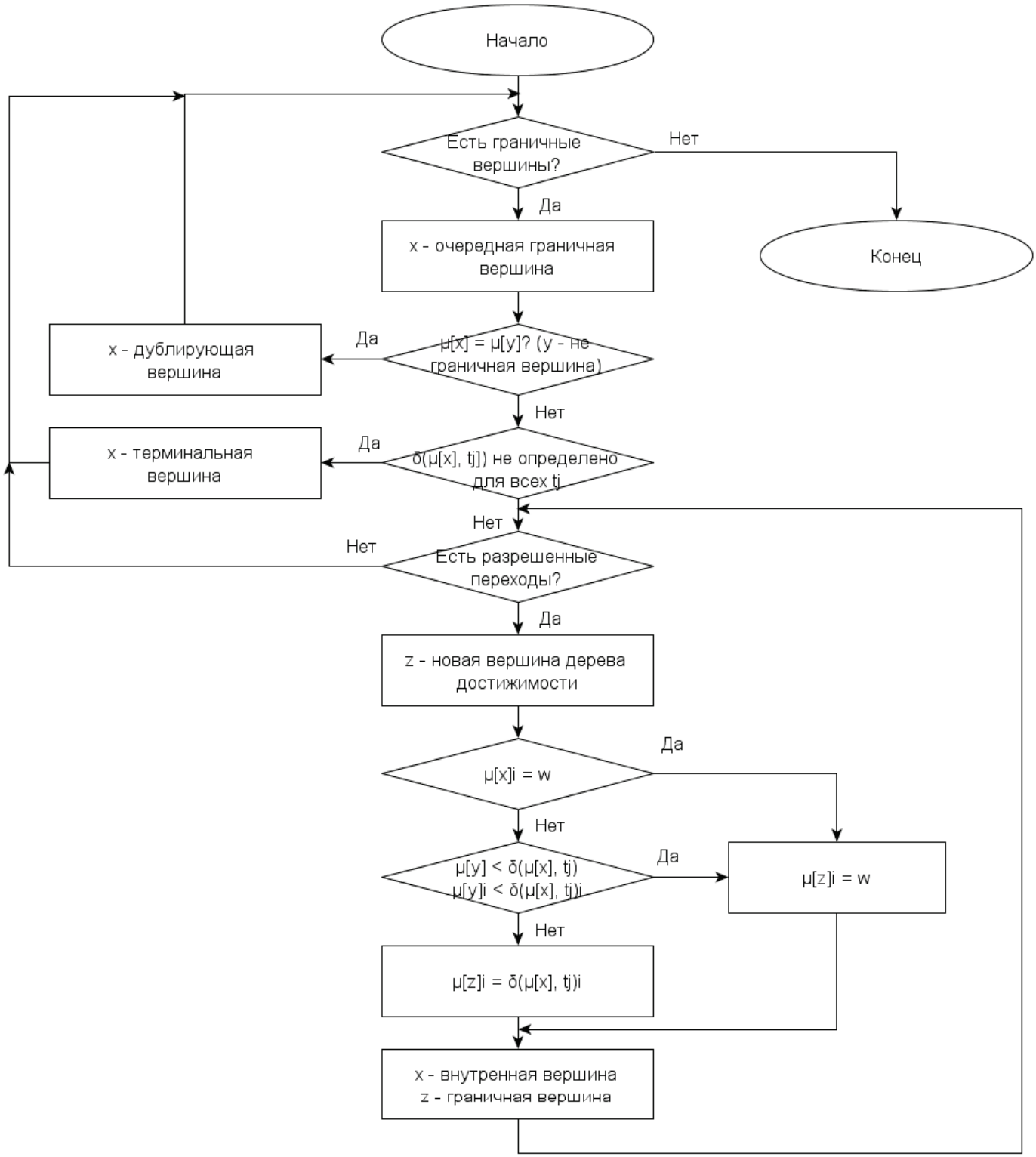


Рис. 1. Алгоритм построения дерева достижимости

Другой подход к анализу сетей Петри основан на матричном представлении сетей Петри и решении матричных уравнений. Матричный подход основывается на представлении сети двумя матрицами D^- и D^+, представляющими входную и выходную функции сети. Каждая матрица имеет m строк (по одной на переход) и n столбцов (по одному на позицию). D^- описывает входы в переходы, D^+ – выходы из переходов. Матрица $D=D^+-D^-$ – составная матрица изменений.

Матричная теория является инструментом для решения проблемы достижимости. Пусть маркировка M' достижима из M_0, тогда существует последовательность (возможно пустая) запусков переходов G, которая приводит из M_0 к M'. Это означает, что $f(G)$ является неотрицательным целым решением следующего матричного уравнения для x:

$$M' = M_0 + x \cdot D \quad (1)$$

Следовательно, если M' достижима из M_0, тогда уравнение (1) имеет решение в неотрицательных целых. Если уравнение не имеет решения, тогда M' недостижима из M_0.

Недостатки матричных методов:

1.Матрица D теряет информацию о ситуациях, когда переходы имеют входы и выходы из одной позиции (петли).

2.Отсутствие информации о последовательности в векторе запуска. Хотя и известно число переходов, порядок их запуска неизвестен.

3.Решение уравнения (1) является необходимым для достижимости, но недостаточным [1].

Матричный метод анализа алгоритма программного обеспечения, формализованного иерархической сетью Петри, позволяет найти недостижимые участки, то есть те части алгоритма, которые никогда не выполняются.

Метод формализации блок-схем алгоритмов иерархическими сетями Петри был реализован программно и протестирован на ряде известных алгоритмов. Тем самым метод показал свою работоспособность. Установлено, что время анализа модели алгоритма зависит непосредственно от размера построенного дерева достижимости сети. Чем больше вершин содержит дерево достижимости, тем больше время анализа. Предложенный метод позволяет получить информацию о наличие взаимоблокировок, невыполнимых операций, циклов и зацикливаний, что повышает надежность разрабатываемого программного обеспечения.

Формализация UML-диаграмм в терминах сети Петри

Существенным недостатком метода формализации блок-схем алгоритмов является отсутствие его практического применения при разработке больших программ. Процесс

проектирования сложных программных систем неразрывно связан с UML (Unified Modeling Language – унифицированный язык моделирования) – стандартом, позволяющим разработчикам формулировать мысли и общаться между собой. Однако UML – всего лишь система обозначений, основанная на диаграммах, она не может защитить от ошибок проектирования. Скрупулезный анализ и хорошо спланированное тестирование позволяют исключить львиную долю ошибок, тем не менее в достаточно сложных системах ошибки могут быть выявлены уже в процессе эксплуатации. Стоимость исправления таких ошибок может быть весьма существенной. Моделирование проектируемой системы сетью Петри благодаря развитым инструментам позволяет разработчикам эффективно обнаруживать и исправлять ошибки на ранних стадиях.

Для преобразования диаграмм в сеть Петри необходимо выполнить пять шагов [3].

Шаг 1. С использованием результатов этапа анализа создается диаграмма классов (class diagram).

Шаг 2. Для каждого класса системы создается диаграмма состояний, отражающая последовательности состояний, в которые попадает класс при его создании, удалении и вызове его методов (statechart diagram).

Шаг 3. Взаимодействие классов фиксируется на диаграмме последовательности (sequence diagram).

Шаг 4. Каждая диаграмма состояний с помощью набора правил, преобразуются в страницу раскрашенной иерархической сети Петри.

Шаг 5. Все полученные страницы иерархической сети Петри связываются в единую сеть на основании диаграммы последовательности.

На основании проведения анализа полученной сети Петри делается вывод о правильности функционирования модели, а в случае обнаружения ошибок и неточностей производится корректировка сети Петри и UML-диаграмм [2].

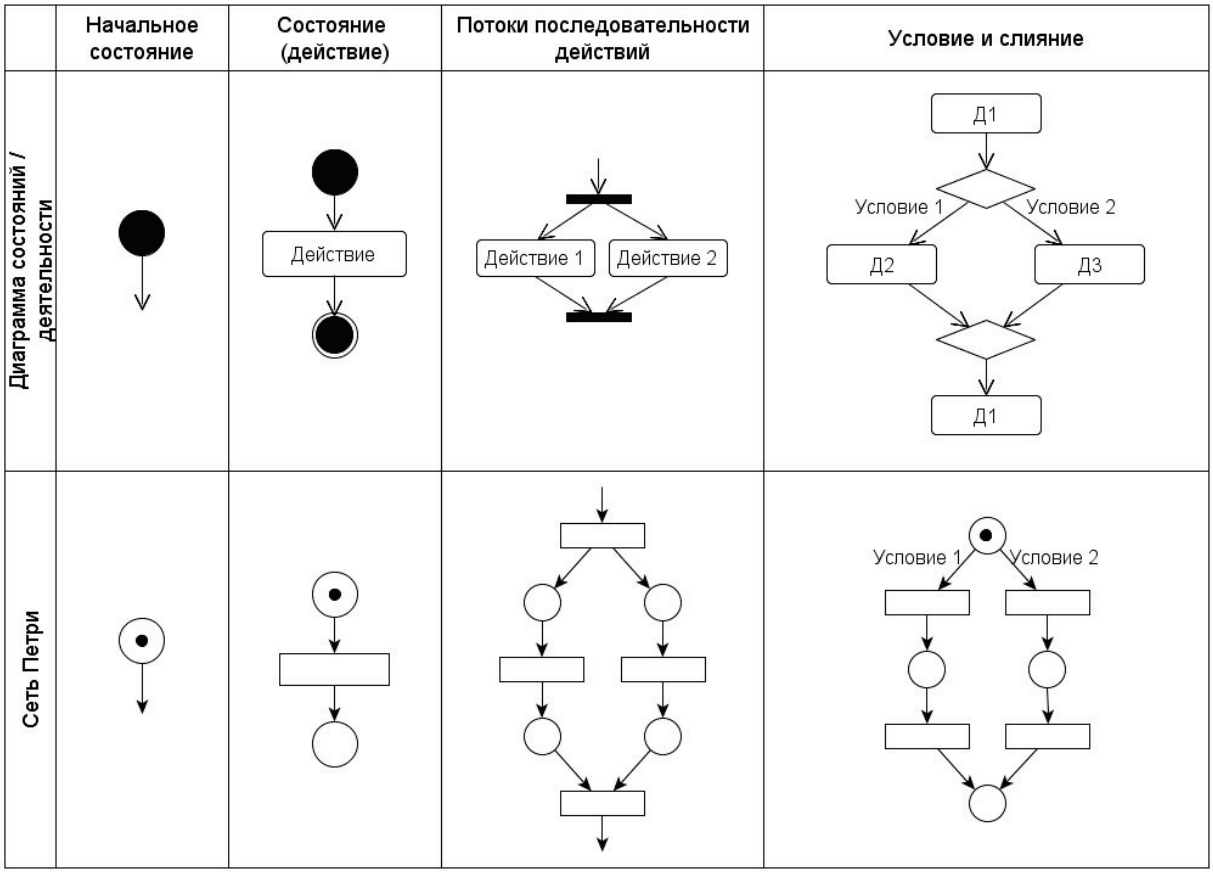


Рис. 2. Формализация диаграммы состояний в терминах сети Петри

Сам по себе UML описывает лишь представление программного комплекса с помощью диаграмм. Однако в настоящее время существует стандарт XMI (стандарт OMG для обмена метаданными с помощью языка XML), который описывает формат данных, предназначенный для хранения UML-описаний. XMI полезен для передачи модели от одного шага к следующему, как например, от проектирования к процессу кодирования, или для передачи от одного инструмента разработки к другому. На Рисунке 3 показана функциональная диаграмма метода формализации программного обеспечения иерархическими сетями Петри.

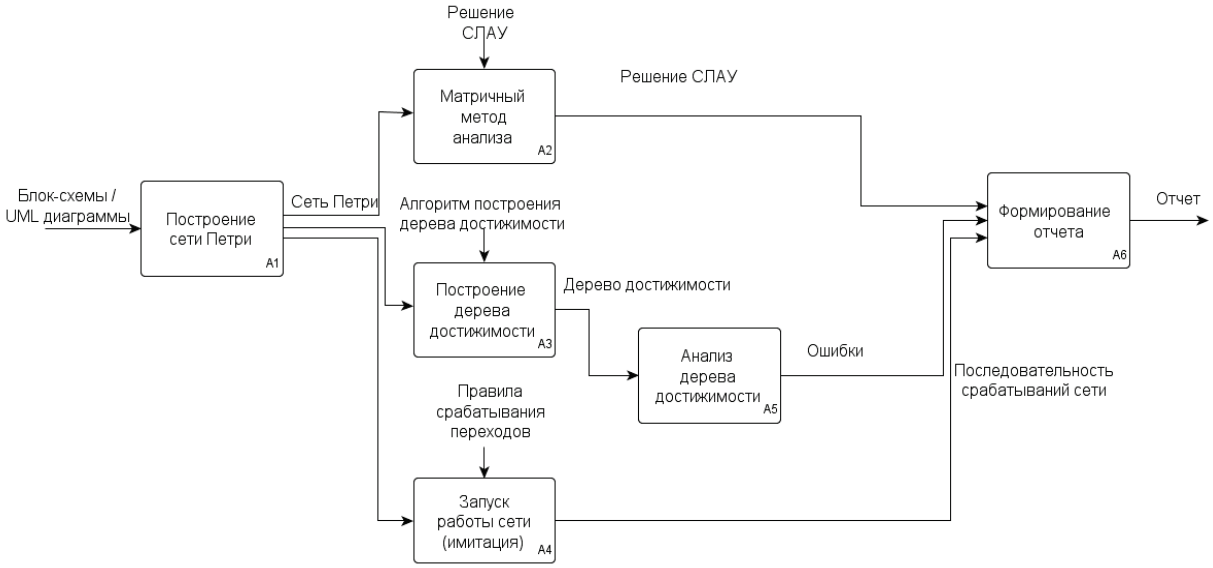


Рис. 3. Функциональная диаграмма метода

Заключение

В статье изложен метод формализации программного обеспечения (блок-схем, UML-диаграмм) иерархическими сетями Петри. Показано, как преобразовать блок-схемы и UML-диаграммы в сеть Петри, приведены способы анализа сети Петри, разработано программное обеспечение, реализующее метод формализации блок-схем алгоритмов иерархическими сетями Петри и позволяющее обнаружить ошибки функционирования.

Список литературы

1. Пащенкова А.В., Рудаков И.В. Программный комплекс верификации алгоритмов программного обеспечения с помощью иерархических сетей Петри // Электронный журнал «Вестник МГТУ им. Н.Э. Баумана: электронное научно-техническое издание». 2012.
2. Воевода А.А., Прытков Д.В. Применение сетей Петри на этапе объектно-ориентированного проектирования. – Сборник научных трудов НГТУ. – 2010. – № 2(60).
3. Коротиков С.В. Применение сетей Петри в разработке программного обеспечения центров дистанционного контроля и управления: дис. канд. техн. Наук. – Новосибирск: НГТУ, 2007.