Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**«Пермский государственный национальный исследовательский университет»

Кафедра математического обеспечения вычислительных систем

УДК 004.942

**Разработка инструментальных средств для реализации многомодельного подхода в системе имитационного моделирования TriadNS***Выпускная квалификационная работа*

Работу выполнил студентка  
 группы ПМИ-1  
4 курса механико-математического факультета \_\_\_\_\_\_\_\_\_\_А. А. Семеновых

Научный руководитель:  
к.ф.-м.н., доцент  
\_\_\_\_\_\_\_\_\_\_\_\_Е. Б. Замятина  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Пермь, 2016

# Аннотация

Семеновых А. А.

Разработка инструментальных средств для реализации многомодельного подхода в системе имитационного моделирования TriadNS: Выпускная квалификационная работа. Пермь: ПГНИУ, Механико-математический факультет, Кафедра математического обеспечения вычислительных систем, 2016.

Работа включает 77 страниц пояснительной записи, 45 использованных источников, 7 таблиц, 32 рисунков и 7 приложений.

Ключевые слова: компьютерные сети, имитационное моделирование, сети Петри, многомодельный подход, модель, метамодель, трансформация моделей.

Выпускная работа посвящена разработке средств для применения многомодельного подхода к исследованию компьютерных сетей с использованием методов имитационного моделирования. Многомодельный подход предполагает, что объект исследования может быть представлен различными моделями, в основе которых лежит та или иная математическая схема. В настоящей работе компьютерные сети были представлены в виде сетей Петри, что дает возможность привлечь богатый математический аппарат для изучения их свойств (тупиковые ситуации, параллельная обработка, живость, достижимость).

Во введении формулируется цель, перечисляются задачи, которые необходимо решить для достижения цели, и методы исследования. В первой главе описано использование предметно-ориентированных языков в имитационном моделировании. Во второй главе рассмотрено понятие сетей Петри и их применения в моделировании. В третьей главе излагаются основные понятия, связанные с архитектурой разрабатываемой системы. Наряду с перечисленным, в главе 4 представлены различные алгоритмы преобразования графических моделей компьютерных сетей в сети Петри. Пятая глава посвящена вопросам реализации графического интерфейса, средств анализа сетей Петри и редактора трансформации моделей.

# Оглавление

[Введение 5](#_Toc446021677)

[Глава 1. Модельно-ориентированный подход 7](#_Toc446021678)

[2. 1. Предметно-ориентированные языки 7](#_Toc446021679)

[2. 2. Использование DSL в моделировании 10](#_Toc446021680)

[2. 3. Использование DSL в TriadNS 11](#_Toc446021681)

[2. 4. Выводы 12](#_Toc446021682)

[Глава 2. Сети Петри и их использование для моделирования компьютерных сетей 13](#_Toc446021683)

[2. 1. Основные понятия 13](#_Toc446021684)

[2. 2. Использование сетей Петри для моделирования 15](#_Toc446021685)

[3. 3. Представление компьютерной сети в виде сети Петри 19](#_Toc446021686)

[2. 5. Обзор симуляторов сетей Петри 22](#_Toc446021687)

[2. 6. Выводы 27](#_Toc446021688)

[Глава 3. Система TriadNS 29](#_Toc446021689)

[3. 1. Архитектура системы 29](#_Toc446021690)

[3. 2. Описание модели 30](#_Toc446021691)

[3. 3. Использование онтологий 31](#_Toc446021692)

[3. 4. Основные особенности системы 33](#_Toc446021693)

[3. 5. Доработка системы 34](#_Toc446021694)

[3. 6. Выводы 35](#_Toc446021695)

[Глава 4. Разработка правил трансформации модели компьютерной сети в сеть Петри 36](#_Toc446021696)

[4. 1. Понятие графовых грамматик 36](#_Toc446021697)

[4. 2. Трансформация слоя структур 37](#_Toc446021698)

[4. 3. Трансформация слоя рутин 38](#_Toc446021699)

[4. 4. Трансформация модели на основе понятий активность и событие 39](#_Toc446021700)

[4. 5. Выводы 45](#_Toc446021701)

[Глава 5. Разработка и реализация системы 47](#_Toc446021702)

[5. 1. Реализация графического редактора для работы с сетями Петри 47](#_Toc446021703)

[5. 2. Реализация средств анализа сетей Петри 50](#_Toc446021704)

[5. 3. Реализация трансформатора моделей 52](#_Toc446021705)

[5. 4. Дополнительные доработки системы 54](#_Toc446021706)

[5. 5. Выводы 54](#_Toc446021707)

[Заключение 55](#_Toc446021708)

[Библиографический список 56](#_Toc446021709)

[Приложения 60](#_Toc446021710)

[Приложение A. Требования, предъявляемые к разрабатываемой системе. 60](#_Toc446021711)

[Приложение B. Рутины сущностей модели компьютерной сети 62](#_Toc446021712)

[Приложение C. Блок-схемы рутин 63](#_Toc446021713)

[Приложение D. Применение правил трансформации для преобразования событий и условий системы в сеть Петри 64](#_Toc446021714)

[Приложение E. Иерархия классов онтологии TriadNS 66](#_Toc446021715)

[Приложение F. Онтология предметной области «Система проектирования и моделирования компьютерных сетей TriadNS» 67](#_Toc446021716)

[Приложение G. Шаблоны рутин для элементов сети Петри 75](#_Toc446021717)

# Введение

В работе рассматривается проблема разработки инструментальных средств для реализации многомодельного подхода в имитационном моделировании. Он предполагает, что моделируемый объект может быть представлен различными моделями, в основе некоторых из них лежат те или иные математические схемы. Так, компьютерная сеть может быть представлена в виде систем массового обслуживания (СМО), сетей Петри (СП) и другим образом, а процессы, происходящие в компьютерной сети, или алгоритмы преобразования данных могут быть описаны с использованием нотаций UML и т.д. В этом случае объект исследования рассматривается с различных точек зрения, к исследованию могут быть подключены специалисты в различных областях знаний. В результате такого многоаспектного моделирования обеспечивается создание наиболее адекватных имитационных моделей.

Таким образом, основной целью работы является проектирование и разработка инструментальных средств имитационного моделирования, позволяющих преобразовывать (выполнять трансформации) графические модели компьютерной сети в другие графические модели (сеть Петри, цепи Маркова, СМО).

Объектом настоящего исследования является изучение возможностей многомодельного подхода в имитационном моделировании; предметом исследования – визуальный язык, позволяющий представлять КС в виде сетей Петри.

Компьютерные сети в настоящее время получили большое распространение в различных областях производства, бизнеса, науки, а также в обычной жизни (социальные сети), поэтому тщательное исследование их структуры и их свойств является актуальным. В настоящее время существует большое количество инструментальных средств, выполняющих имитационное моделирование КС [43, 16, 15, 1]. Однако, на сегодняшний день, для того, чтобы применение методов имитационного моделирования было наиболее эффективным, необходимо привлекать к исследованиям специалистов из разных областей знаний.

К примеру, очень важным является тот факт, что при разработке имитационной модели исследователю необходимо оперировать терминами и понятиями, которые являются привычными для него, поэтому иногда целесообразно применять многомодельный подход, который представляет собой не что иное, как исследование объекта с разных сторон, с разных точек зрения [34]. Такая попытка применить многомодельный подход в имитационном исследовании КС является относительно новой.

Предполагается, что построение модели выполняется с помощью графического интерфейса. Для преобразования моделей из одного представления в другое (трансформации) пользователь должен построить соответствующие правила. Трансформации визуальных моделей по определенным правилам выполняются обычно специальными инструментальными средствами – редакторами DSL: MetaEdit+ [21], MetaLanguage [20] и др.

Редакторы выполняют преобразование из одной метамодели в другую на основе горизонтальной трансформации графов. Причем описание правил преобразования моделей должно производиться в терминах исходного и целевого языков моделирования, понятных различным категориям специалистов. Кроме того, достаточно часто появляется необходимость преобразования визуальной модели в некоторое текстовое представление, например, в исходный код на целевом языке программирования, и обратно.

Задачи, которые необходимо решить для выполнения работы:

* рассмотреть теоретические аспекты модельно-ориентированного подхода;
* исследовать трехуровневую структуру в имитационном моделировании (метамодель, модель, данные) на примере работы с системой имитационного моделирования TriadNS;
* разработать визуальный язык для сетей Петри;
* разработать графический редактор для построения и моделирования сетей Петри, предоставляющий удобные средства для работы с моделью;
* разработать средства анализа сетей Петри;
* разработать правила трансформации из модели компьютерной сети в сеть Петри;
* разработать инструментальные средства для трансформации моделей в системе TriadNS.

Для решения поставленных задач могут быть использованы теория сетей Петри, теория графов, теория формальных грамматик, методы построения предметно-ориентированных языков (DSL).

Поскольку модель КС может быть связана с конкретной предметной областью, т.е. описана в терминах одного из предметно-ориентированных языков, следует более подробно рассмотреть вопросы, связанные с предметно-ориентированными языками (DSL) и модельно-ориентированным подходом в разработке программного обеспечения (ПО).

# Модельно-ориентированный подход

Традиционные средства разработки информационных систем (ИС) в настоящее время все больше и больше не удовлетворяют потребности создания новых ИС, применяемые в новых областях, автоматизирующие новые виды деятельности, поэтому наиболее востребованными становятся инструментальные средства, основанные на применении модельно-ориентированного подхода. Такие средства позволяют объединить усилия разработчиков и экспертов в предметной области, снизить трудоемкость создания и сопровождения и сделать ИС более гибкими. Гибкость также достигается за счет того, что при изменении модели не нужно переписывать программный код системы, ее поведение изменяется вместе с моделью.

Модельно-ориентированный подход (МОП) предполагает наличие хотя бы одной модели. В данном случае модель – это абстрактное описание системы, в котором отражены существенные с точки зрения решаемых ее разработчиком задач характеристики, особенности функционирования [38]. Модель создается с использованием определенных языков моделирования, чаще всего визуальных, так как они наглядны и понятны не только программистам, но и экспертам в предметной области, конечным пользователям.

Существуют реализации МОП, которые используют языки моделирования общего назначения для построения моделей, однако они сложны для понимания не только для экспертов в конкретной предметной области, но даже иногда для разработчиков. Именно поэтому в модельно-ориентированной разработке ПО все более широко применяются визуальные предметно-ориентированные языки моделирования (Domain-Specific Modeling Languages – DSML, Domain-Specific Languages – DSL), предназначенные для решения определенного класса задач в конкретной предметной области.

## Предметно-ориентированные языки

Предметно-ориентированный язык (Domain Specific Language, DSL) – язык программирования, созданный для использования в рамках конкретной предметной области. Примерами таких языков могут быть Excel, AutoCad, ProEngineer, Mathematica и др.

DSL просты в использовании, а также понятны пользователям, так как они оперируют терминами предметной области. Это позволяет упростить процесс написания программ, а также программировать «не программистам». Однако DSL обладают существенным недостатком – сложность разработки и, в отличие от языков общего назначения, их нельзя использовать вне конкретной предметной области, поэтому для каждой предметной области, либо задачи, необходимо создавать новый DSL.

DSL отличается от языков общего назначения, так как при их разработке необходимо создавать и собственные полнофункциональные среды для работы с ними, т.е. удобный графический редактор или редактор кода, а также удобный отладчик программ. Для решения этих проблем в последнее время стал активно развиваться новый тип программного обеспечения – языковые инструментарии (language workbenches).

Предметно-ориентированный язык – это язык программирования с ограниченными выразительными возможностями, ориентированный на некую конкретную предметную область. DSL поддерживает минимум возможностей, необходимых для поддержки своей предметной области. Фаулер делит предметно-ориентированные языки на три основные категории: внешние DSL, внутренние DSL и языковые инструментальные средства [45].

Внешний DSL – язык, отделенный от основного языка приложения. Как правило, внешний DSL имеет пользовательский синтаксис, но при этом достаточно широко применяется синтаксис другого языка. Сценарий анализируется кодом принимающего приложения с помощью методов синтаксического анализа текста. Такой DSL трансформируется в приложение с помощью компилятора или интерпретатора. Примерами внешних DSL являются регулярные выражения, SQL, XML-файлы.

Внешние DSL обеспечивают возможность использовать любой синтаксис: можно выразить предметную область в удобной и простой форме. Формат такого DSL будет ограничен лишь созданием транслятора для этого языка. Если язык несложный, то это не составит особого труда, но более сложный язык потребует и больших усилий.

Недостатком можно назвать и тот факт, что внешний DSL никак не связан с основным языком приложения и программная среда ничего не знает о данном DSL. Этот недостаток обеспечивает неудобство при написании кода нового языка, при отладке и так далее.

В случае внешнего DSL имеется четкая граница между предметно-ориентированным языком и языком общего назначения. Языки могут быть ориентированы на предметную область и при этом оставаться языками общего назначения.

Внутренний DSL – это специфический способ использования языка общего назначения. Внутренние DSL еще называют «встроенными» (embedded). Сценарий во внутреннем DSL представляет собой корректный код языка общего назначения, который использует только подмножество возможностей этого языка в определенном стиле, чтобы работать с одним небольшим аспектом всей системы. Результат должен иметь вид исходного текста пользовательского, а не базового языка. Примером этого стиля является Lisp.

Достоинства и недостатки использования внутренних DSL являются зеркальным отражением внешних DSL.

Ограничительные возможности для внутренних DSL связаны прежде всего с тем, что внутренние DSL сами по себе являются языком общего назначения. При формировании выражения DSL ограничения возникают из-за использования небольшого подмножества всех возможностей языка программирования общего назначения. Обычно в этом случае стараются избегать условных конструкций, циклов и переменных.

Внутренние DSL являются наиболее доступной для написания формой DSL: не нужно изучать грамматику и синтаксический анализ, как при освоении внешних DSL, а в отличие от языковых инструментальных средств не нужны специальные инструменты для работы. Используя внутренний DSL, происходит работа в обычной языковой среде.

Однако использование внутренних DSL ограничено базовым языком: так как любое выражение должно быть корректным выражением базового языка, особенности применения внутреннего DSL тесно связаны с возможностями языка общего назначения.

Возможность использовать для DSL все инструменты основного языка программирования является двоякой. С одной стороны, если программист хорошо владеет данным языком, то никаких проблем не возникает. Однако преимущество DSL состоит в том, чтобы программировать, не зная возможностей базового языка. Внутренний DSL может усложнить процесс, если программист недостаточно знаком с возможностями базового языка.

Языковые инструментальные средства – специализированные интегрированные среды разработки для определения и создания DSL. В частности, языковые инструментальные используются не только для определения структуры DSL, но и как пользовательские среды редактирования для людей, которые будут писать сценарий DSL. Получающиеся в результате сценарии тесно объединяют среду редактирования и язык.

Преимущества использования DSL:

1. Главное преимущество в использовании языка DSL это то, что он предоставляет средства для более ясного выражения назначения части системы. Ограниченные выразительные возможности DSL усложняют написание неправильного кода и облегчают поиск сделанных ошибок;
2. Предоставляет ясный и точный язык предметной области, в том числе для общения с заказчиками и пользователями программного обеспечения.

Недостатки:

1. Для создания DSL необходимо разрабатывать полнофункциональные среды, для работы с ними;
2. DSL нельзя применять вне конкретной предметной области, а для решения иных задач нужно создавать новый DSL.

Предметно-ориентированные языки используются в рамках конкретной предметной области, для решения определенного круга задач, в том числе и в моделировании.

## Использование DSL в моделировании

Модель – это абстрактное представление системы, определенное на языке моделирования [2].

Метамоделирование – это процесс полной и точной спецификации предметно-ориентированного языка моделирования (DSML), который используется для определения модели в терминах конкретной предметной области [3]. Метамодель описывает абстрактный синтаксис языка моделирования и определяет язык метамоделирования, наиболее подходящий для описания новых языков моделирования. Метамодель языка метамоделирования называется мета-метамодель. Трехуровневую структуру, представленную моделью, метамоделью и мета-метамоделью, можно представить в виде диаграммы (Рисунок 1). Такой шаблон разработки может применяться несколько раз: модели на уровне n задаются с помощью языка, определенного как метамодель на уровне n+1. Также используют четырехуровневую структуру, в которой внешним (четвертым) уровнем являются сами данные, построенные в соответствии с моделью.

Рисунок 1. Диаграмма метамоделирования

Такой подход используется в различных технологиях разработки, например, в модельно-ориентированном подходе (Model Driven Architecture, MDA).

Инструмент моделирования обеспечивает пути для развития моделей, основанных на языке моделирования. В большинстве случаев, графические инструменты моделирования включают дополнительные инструменты, такие как: преобразование модели, компилятор, интерпретатор, отладчик и др. В этом случае весь набор инструментов называется средой моделирования. Среда метамоделирования – это инструмент для разработки инструмента моделирования, определенного метамоделью. Полученные инструменты могут работать как в среде метамоделирования, так и в качестве отдельных программ. Метамоделирование также называют предметно-ориентированным моделированием (DSM).

Как только модели построены в соответствии с некоторой метамоделью, можно выполнять трансформацию моделей, т. е. переход от некоторой исходной модели в целевую модель на основе некоторых правил.

Иногда для устранения недостатков языков моделирования в процессе разработки DSL создают метаязык, благодаря чему возможно настраивать язык, повторное использование DSL, интеграции нескольких языков в одной системе, а также трансформация моделей с одного языка моделирования на другой [4].

DSL используется и в системе TriadNS для различных целей, поэтому необходимо рассмотреть данный вопрос подробнее.

## Использование DSL в TriadNS

TriadNS разрабатывается на языке общего назначения C# в среде Visual Studio.

В качестве внутреннего предметно-ориентированного языка в TriadNS выступает язык имитационного моделирования Triad. Как и любой внутренний язык, Triad является языком общего назначения, но в данном случае он определяется с помощью инструментальных средств другого языка – C#.

Внешним языком для системы является визуальный, с помощью которого строятся модели в TriadNS. На данный момент в TriadNS возможно построение моделей компьютерной сети, сети Петри, систем массового обслуживания в виде графа. Формальные языки описания данных моделей и будут внешними DSL, которые обрабатываются с помощью инструментальных средств языка C#.

В качестве метамодели для внешних DSL выступают онтологии предметных областей. На основе этих метамоделей происходит построение моделей. В качестве мета-метамодели выступает базовая онтология Triad.Net (Рисунок 2).

Рисунок 2. Структура DSL сетей Петри

Таким образом, была построена трехуровневая структура метамоделирования в системе TriadNS на примере сетей Петри.

## Выводы

В данной главе рассматривается модельно-ориентированный подход, применяемый при создании систем. При разработке инструментальных средств для реализации многомодельного подхода в имитационном моделировании в рамках СИМ TriadNS данный подход широко используется.

В следующей главе рассматриваются основные понятия вспомогательной для исследования компьютерных сетей модели – сети Петри, а также проводится обзор существующих редакторов сетей Петри и анализ их достоинств и недостатков.

# Сети Петри и их использование для моделирования компьютерных сетей

Вычислительные системы достаточно сложны и включают множество взаимодействующих компонент, каждая из которых также может быть сложной. Однако, поведение каждой из компонент можно описать независимо от других, также она имеет некоторое множество состояний. Действиям компонент системы присущи совмещенность и параллелизм, из этого следует, что в исследуемой системе необходим некоторый механизм синхронизации. Сети Петри специально разрабатывались Карлом Петри для моделирования таких сложных систем, имеющим взаимодействующие параллельные компоненты.

Сети Петри – мощный математический аппарат, с помощью которых могут быть промоделированы многие системы, в особенности системы с независимыми компонентами, например, аппаратное и программное обеспечение ЭВМ, физические системы, социальные, биологические системы, бизнес-процессы, использование в медицине и т.д. Они применяются для исследования возникновения различных событий в системе [42].

В данной работе сети Петри рассматривается как дополнительная модель для анализа исследуемого объекта – компьютерных сетей. С помощью этой модели можно анализировать поведение системы, происходящие в ней процессы и выработать замечания по модификации поведения самого объекта, обнаружить потенциальные тупики и исключительные ситуации.

## Основные понятия

Сети Петри применяются исключительно при моделировании и являются инструментом исследования систем. Теория сетей Петри делает возможным моделирование системы математическим представлением ее в виде сети Петри [42].

Возможно несколько подходов к использованию сетей Петри при проектировании систем. В одном из подходов сети Петри рассматриваются как вспомогательный инструмент анализа. Здесь для построения системы используются общепринятые методы проектирования. Затем построенную систему моделируют сетью Петри, и затем выполняют анализ модели. При этом трудности, встречающиеся при анализе, указывают на изъяны в проекте. Для их исправления необходимо модифицировать проект, который затем снова моделируется и анализируется. Цикл повторяется до тех пор, пока проводимый анализ не приведет к успеху.

Другой подход предполагает проведение проектирования системы в терминах сетей Петри. Здесь задача заключается в преобразовании представления сети Петри в реальную систему.

Таким образом, в первом случае необходима разработка методов моделирования систем сетями Петри, а во втором случае должны быть разработаны методы реализации сетей Петри системами. В обоих случаях необходимы методы анализа сетей Петри для определения свойств модели (Рисунок 3).

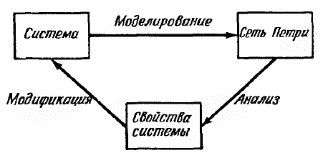


Рисунок 3. Использование сетей Петри для моделирования и анализа систем [42]

Теоретико-множественное определение структуры сети Петри можно представить в виде: , где – конечное множество позиций, – конечное множество переходов. Множество позиций и множество переходов не пересекаются, . является входной функцией – отображением из переходов в комплекты позиций. есть выходная функция – отображение из переходов в комплекты позиций.

Для иллюстрации понятий теории сетей Петри гораздо удобнее использовать графическое представление сетей Петри.

Граф G сети Петри есть двудольный ориентированный мультиграф, , где – множество вершин, а – комплект направленных дуг, , где . Множество V может быть разбито на два непересекающихся подмножеств P и T, таких, что , и для любой направленной дуги , если , тогда либо , либо .

Структура сети Петри представляет собой совокупность позиций и переходов. Вершина-позиция обозначается кружком, вершина-переход обозначается вертикальной чертой. Ориентированные дуги соединяют позиции и переходы.

Маркировка – это присвоение фишек позициям сети Петри, которые используются для определения выполнения сети Петри.

Маркировка μ сети Петри есть функция, отображающая множество позиций P в множество неотрицательных чисел N.

Маркировка может быть определена как вектор .

Маркированная сеть Петри есть совокупность структуры сети Петри и маркировки и может быть записана в виде . На графе сети Петри фишки изображаются в виде точки внутри кружка, который представляет собой позицию сети Петри.

Выполнением сети Петри управляют количество и распределение фишек в сети. Фишки находятся в кружках и управляют выполнением переходов сети. Сеть Петри выполняется посредством запусков переходов. Переход запускается удалением фишек из его входных позиций и образованием новых фишек, помещаемых в его выходные позиции.

Переход может запускаться только в том случае, когда он разрешен. Переход называется разрешенным, если каждая из его входных позиций имеет число фишек по крайней мере равное числу дуг из позиции в переход. Кратные фишки необходимы для кратных входных дуг. Фишки во входной позиции, которые разрешают переход, называются его разрешающими фишками.

Переход запускается удалением всех разрешающих фишек из его входных позиций и последующим помещением в каждую из его выходных позиций по одной фишке для каждой дуги.

Состояние сети Петри определяется ее маркировкой.

## Использование сетей Петри для моделирования

Сети Петри – мощный математический инструмент, предназначенный для моделирования распределенных и параллельных систем. С их помощью могут быть промоделированы многие системы, в особенности системы с независимыми компонентами, например, программное и аппаратное обеспечение ЭВМ, физические системы и др. Сети Петри применяются для моделирования возникновения различных событий в системе. В частности, они могут моделировать поток информации или другие ресурсы системы [42].

В настоящей работе рассматривается исследование компьютерных сетей с помощью сетей Петри.

Простое представление системы с помощью сетей Петри основано на событиях, возникающих в системе, и условиях. В этом случае события представляются в виде переходов, а позиции – в виде условий. При этом входы перехода являются предусловиями, а выходы – постусловиями. Возникновение события запускает переход, а выполнение условия отмечается возникновением фишки в позиции. Запуск перехода удаляет разрешающие фишки, представляющие выполнение предусловий, и создает новые фишки, представляющие выполнение постусловий.

Важной особенностью сетей Петри и систем, которые можно представлять в виде этой модели, является их параллелизм. Параллелизм заключается в том, что два разрешенных невзаимодействующих события могут происходить независимо друг от друга. Таким образом, сети Петри являются удобной моделью для систем, в которых несколько процессов могут выполняться одновременно.

Выполнение сети Петри рассматривается как последовательность дискретных событий. Появление конкретного события является одним из возможных, что приводит к недетерминированности, поэтому с помощью сетей Петри удобно моделировать системы, в которых события происходят асинхронно и независимо друг от друга. В частности, такую ситуацию можно наблюдать при исследовании компьютерных сетей.

Анализ сетей Петри помогает получить информацию о структуре и динамическом поведении моделируемой системы, что может быть полезно при оценке системы и выработке предложений по ее усовершенствованию и изменению [42].

1. Свойства сетей Петри

Возможность моделирования системы с помощью сетей Петри является малополезным без знаний о том, как анализировать полученную модель. Анализ сетей Петри приводит к анализу поведения моделируемой системы и сводится к выполнению некоторых свойств сетей Петри.

Одно из важнейших свойств сети Петри, которая должна моделировать реальное устройство – это безопасность. Позиция сети Петри является безопасной, если число фишек в ней никогда не превышает 1. Сеть Петри называется безопасной, если все позиции сети являются безопасными. При исследовании компьютерных сетей с помощью сетей Петри свойство безопасности может свидетельствовать о возникновении каких-либо событий в исследуемой системе (буфер пуст, сообщение отправлено и т.д.).

Безопасность – это частный случай свойства ограниченности сетей Петри. Безопасность позволяет реализовать позицию триггером, ограниченность – счетчиком. Позиция является k-ограниченной, если количество фишек в ней никогда не превышает числа k. Соответственно, сеть Петри ограничена, если все ее позиции ограничены.

Ограниченность и безопасность характеризуют емкость условий. Так, можно ограничить емкость накопителей. Примером выполнения этого свойства может являться объем буфера, количество сообщений и т.д. в зависимости от исследуемой системы. Возможность неограниченного роста числа меток свидетельствует об опасности неограниченного роста длин очередей (например, переполнение памяти).

Сети Петри можно использовать для моделирования систем распределения ресурсов. В этом случае для сетей Петри важным свойством является сохранение. Сеть Петри является сохраняющей, если число маркеров в ней остается неизменным.

Одна из задач, которая может возникнуть при исследовании различных систем, - это возникновение тупиковых ситуаций. Тупик в сети Петри – это переход, который не может быть запущен. Таким образом, переход называется активным, или живым, если он не является заблокированным. Если какой-либо переход в процессе имитации ни разу не срабатывает, то для исследуемой системы это может свидетельствовать о том, что данная часть системы лишняя или неправильно функционирует.

Также можно выделить следующее понятие; переход называется потенциально живым, если существует такая разметка сети Петри, что данный переход может потенциально сработать.

Чтобы контролировать работу исследуемой системы либо возникновение каких-либо событий, сеть Петри должна удовлетворять свойству достижимости. Достижимость сети Петри заключается в следующем: может ли быть достигнуто определенное состояние сети Петри в процессе ее выполнения. Такая задача может решаться, например, при исследовании пути передачи сообщений между узлами компьютерной сети. Свойство достижимости является наиболее важным, так как все перечисленные выше свойства могут быть сведены к задаче достижимости.

1. Методы анализа сетей Петри

В теории сетей Петри существует несколько методов решения задач анализа модели: дерево достижимости и использование матричных уравнений. Данные методы достаточно просто реализовать программным способом, что является важным при реализации автоматического анализа СП [42], более того, существуют готовые алгоритмы для реализации этих методов.

В данной работе будет рассмотрен только метод построения дерева достижимости, так как этот метод позволяет проверить все рассмотренные выше свойства сетей Петри, тогда как матричные уравнения позволяют проверить свойства сохранения, достижимости. Хотя матричные уравнения достаточно перспективный метод, однако он имеет некоторые трудности [42].

Дерево достижимости представляет множество состояний сетей Петри, т.е. все возможные запуски переходов. Для того чтобы составить дерево достижимости необходимо пронумеровать каждую позицию. Каждая его вершина представляется в следующем виде , где – количество маркеров в позиции с номером, соответствующим номеру индекса.



Корнем дерева представляется состояние модели перед ее выполнением. Если система из состояния переходит в состояние после срабатывания перехода , то называется порождающей маркировкой, а – порожденной. В дереве достижимости вершина, соответствующая состоянию , является родительской для вершины, соответствующей состоянию , а дуга, соединяющая их, подписывается именем перехода , который привел к порожденной маркировке.



Рассмотрим следующий пример: пусть задана сеть Петри (Рисунок 4). Тогда деревом достижимости для такого примера служит дерево, построенное на Рисунок 5 после третьего шага выполнения.

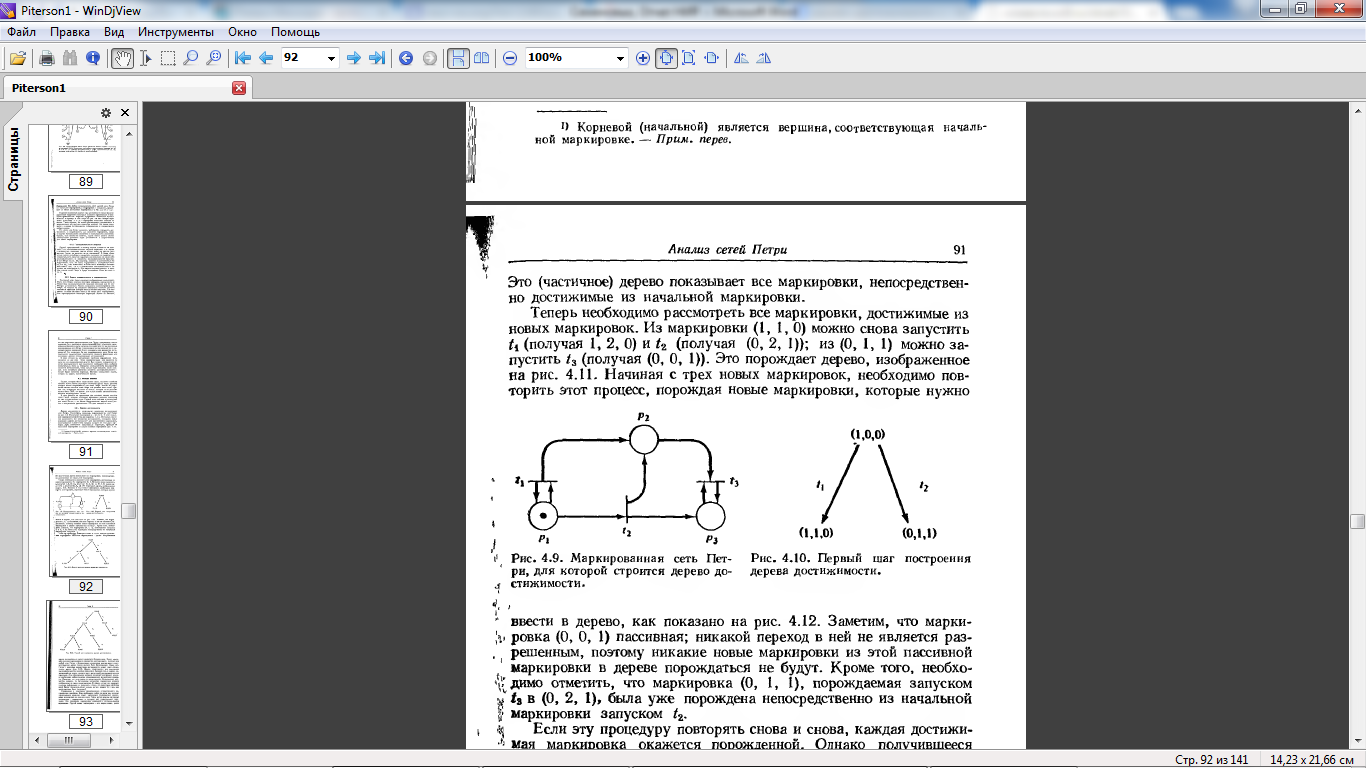


Рисунок 4. Пример сети Петри [42]

В данной сети вершины-позиции имеют такие имена, что удобно их нумеровать, поэтому в дереве достижимости не возникает сомнений, какие вершины сколькими метками обладают.

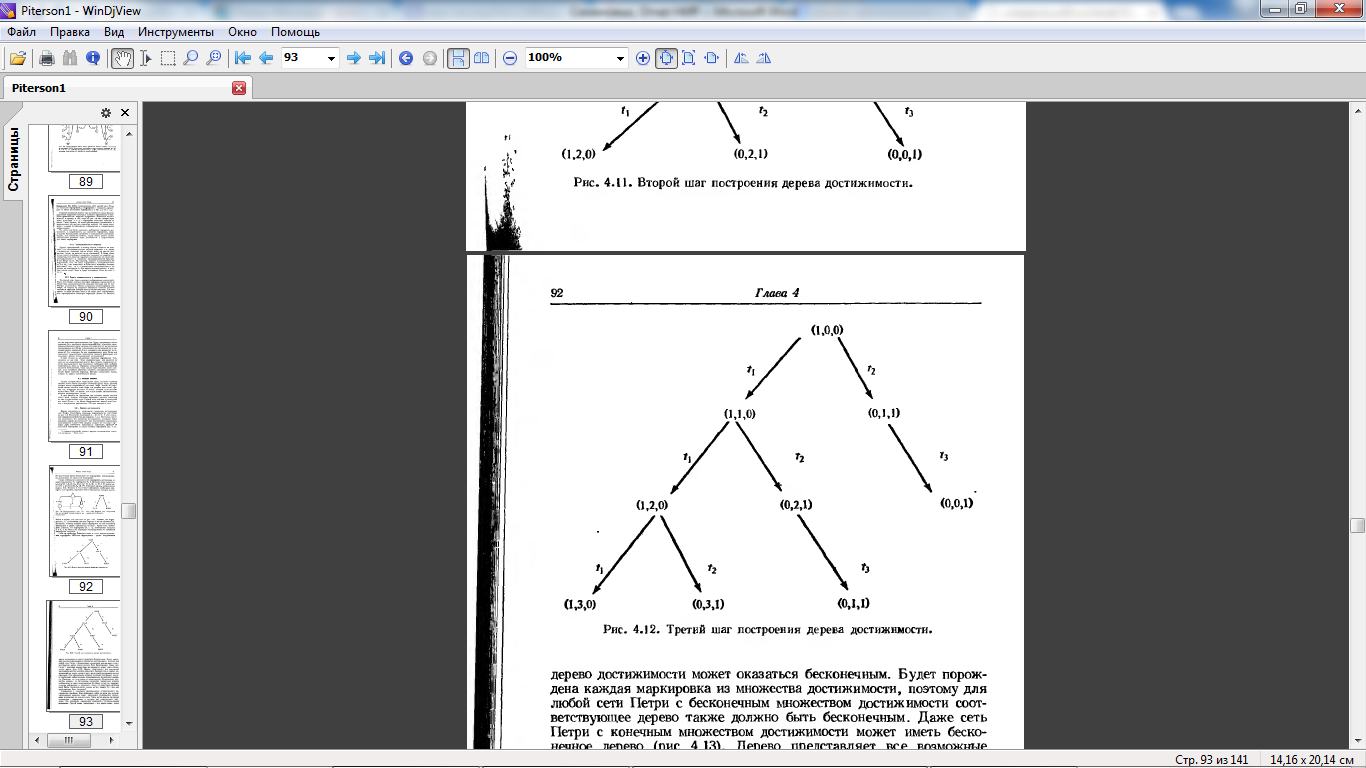


Рисунок 5. Дерево достижимости для примера [42]

Дерево достижимости может оказаться бесконечным. Даже в результате ограниченных сетей Петри может быть построено бесконечное дерево. Для решения такой задачи существуют несколько способов, а также представлен алгоритм в [42].

В [42] также доказано, что все свойства сетей Петри могут быть проверены с помощью построения дерева достижимости, поэтому построение дерева достижимости является не только наглядным методом анализа, но и эффективным.

## Представление компьютерной сети в виде сети Петри

Сети Петри были разработаны и используются в основном для моделирования. С их помощью возможно моделировать достаточно сложные вычислительные системы, аппаратное и программное обеспечение ЭВМ и др.

В данной работе с помощью сетей Петри моделируются компьютерные сети, при этом преобразование компьютерной сети в сеть Петри может быть выполнено несколькими способами в зависимости от детализации функций, выполняемых каждым элементом компьютерной сети.

В простейшем случае каждый элемент компьютерной сети можно представить в виде вершины-позиции сети Петри, а за взаимодействие между элементами компьютерной сети будут отвечать вершины-переходы. Так, например, элемент рабочая станция компьютерной сети можно представить в виде сети Петри, заданной на Рисунок 6.

Рабочая станция

Отправка запроса

. . .

. . .

Рисунок 6. Рабочая станция в виде сети Петри

Если же рассматривать элементы компьютерной сети и функции, которые они обеспечивают, более подробно, то для преобразования компьютерной сети в сеть Петри необходимо использовать определенный алгоритм. В [29] предложена методика перехода от моделей описания последовательности (IDEF0, UML) к сетям Петри. В данном случае сначала строится модель IDEF0, переход от которой ведет к диаграмме последовательностей UML, далее диаграмма последовательностей разбивается на отдельные фрагменты, которые в дальнейшем будет объединены в часть сетей Петри. Полученные части сети Петри соединяются между собой, образуя искомую модель.

Также возможно преобразование диаграммы активностей UML в сеть Петри с помощью различных алгоритмов [18]. При преобразовании процессов в сеть Петри позициями могут быть не только действующие объекты, но и условия, при выполнении которых происходит выполнение некоторых функций. Функции, которые выполняют объекты, преобразуются в переходы. Таким образом, статические элементы исходной модели преобразуются в позиции, а динамические элементы – в переходы.

В статье [27] представлен другой подход к преобразованию модели компьютерной сети в сеть Петри. Сначала определяются основные ролевые функционалы, представленные в виде простых сетей Петри, которые в дальнейшем с помощью операций с сетями Петри преобразуются в необходимые объекты компьютерной сети. Основными ролевыми функционалами в этом случае являются приемник/передатчик трафика, независимый генератор трафика и зависимый генератор трафика. Основные операции, которые выполняются с сетью Петри, - это отождествление позиций, добавление дуги, исключение дуги, переименование позиций, объединение СП, расщепление СП.

Рассмотрим подробнее структуры основных ролевых функционалов.

1. Приемник/передатчик трафика представляет элементарную сеть Петри, на основе которой строятся все остальные функционалы. Он состоит из двух позиций и одного перехода. Трафик поступает в позицию 1 и через переход попадает в позицию 2. С наличием перехода связана задержка в обслуживании. Трафик представляется в виде метки (Рисунок 7).

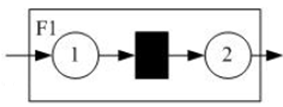


Рисунок 7. Функционал - приемник/передатчик трафика

1. Генератор трафика (независимый вариант): момент поступления новых меток в позицию 2 зависит от временной задержки. При входе в позицию 1 метки после срабатывания перехода в позиции 2 появится метка, а в позиции 1 также появится метка, таким образом генерируются новые сообщения (Рисунок 8).

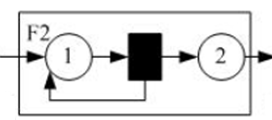


Рисунок 8. Функционал - независимый генератор трафика

1. Генератор трафика (зависимый вариант): генерация нового трафика зависит от некоторого дополнительного условия, т.е. момент поступления новых меток в позицию 2 зависит от временной задержки перехода и от наличия в позиции 3 метки (Рисунок 9).

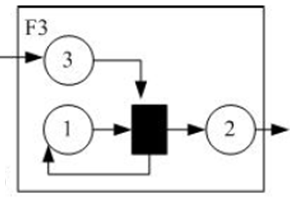


Рисунок 9. Функционал - зависимый генератор трафика

Так, например, рабочая станция с одним сетевым интерфейсом может быть представлена следующим образом (Рисунок 10). Сетевой узел принимает сообщения-запросы, обрабатывает их и отправляет в сеть сообщения-ответы. Подсеть позиция 1, переход 1, переход 2 представляют приемник трафика, который отвечает за прием сообщений; подсеть позиция 3, позиция 4, переход 2, позиция 5 – зависимый генератор трафика, который отвечает за обработку и отправку сообщений в сеть.

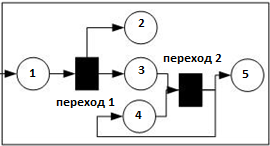


Рисунок 10. Рабочая станция

Маршрутизатор же принимает сообщения, анализирует и, при передаче из одной среды передачи в другую, преобразует сообщение, передает его на один, несколько или все порты (Рисунок 11). В данном случае позиции 1-4 вместе с переходами моделируют переход трафика в сеть с иной средой передачи.

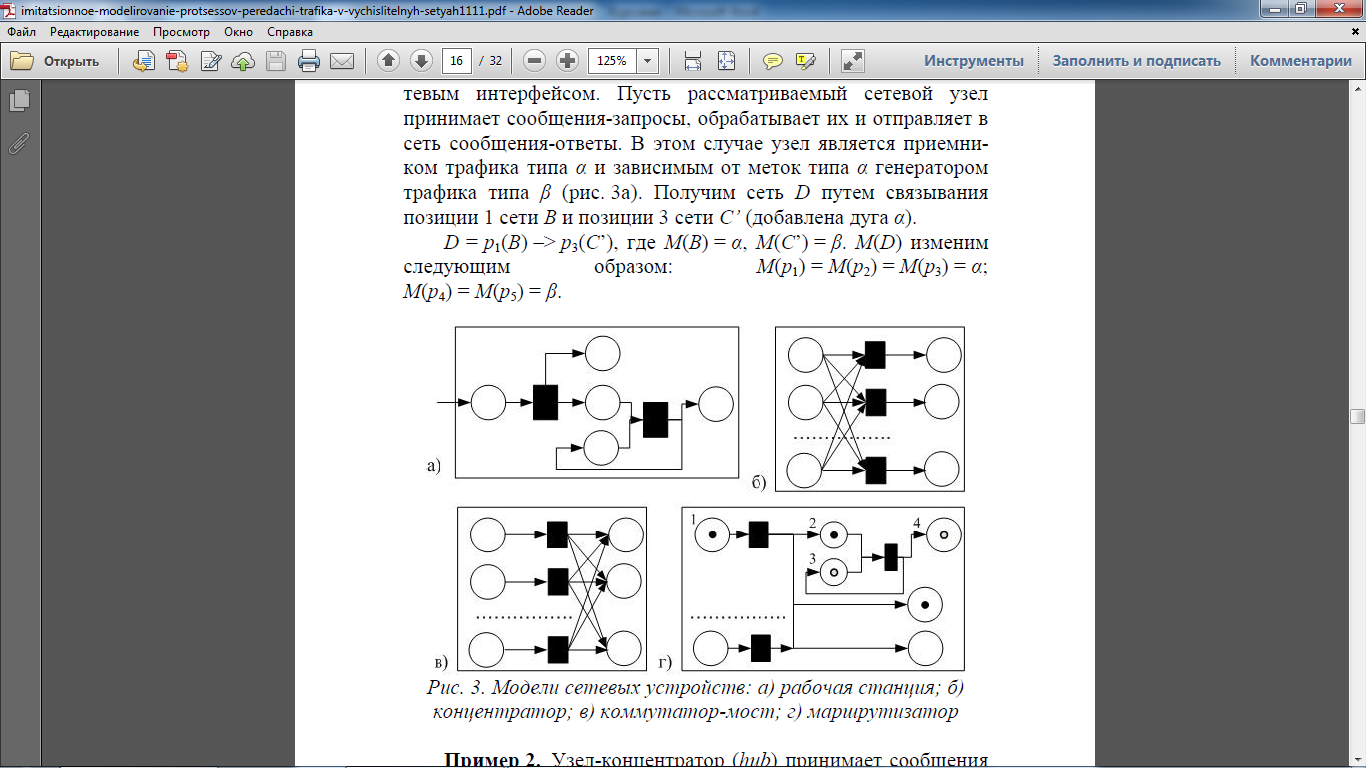


Рисунок 11. Маршрутизатор

Существуют и другие алгоритмы преобразования некоторых процессов в нотации сетей Петри, используя различные алгоритмы и математические средства [30].

## Обзор симуляторов сетей Петри

Теория сетей Петри со времени диссертации доктора Карла Петри в 1962 г. значительно развилась. С этого времени было создано достаточно много редакторов, позволяющих строить различные виды сетей Петри, и симуляторов, производящих моделирование их поведения [11].

В данной главе будут рассмотрены симуляторы, которые сравниваются по следующим выбранным критериям:

1. Удобство построения и представления сетей Петри в графическом редакторе;
2. Визуализация процесса моделирования;
3. Ручной запуск переходов в процессе моделирования;
4. Наглядное отображение результатов анализа и моделирования сетей Петри;
5. Использование различных видов сетей Петри;
6. Стоимость системы;
7. Дополнительные возможности (многоплатформенность и др.)

Главными критериями при анализе симуляторов являются, прежде всего, удобство построения модели, ее графическое представление в системе, а также наглядность моделирования. Дополнительным критерием в сравнении симуляторов в процессе симуляции является возможность ручного управления процессом, так как иногда при исследовании систем необходимо отследить определенные условия. Критерий стоимости системы также учитывается, потому что в работе рассматриваются как платные, так и бесплатные симуляторы. Также не рассматриваются системы имитационного моделирования общего назначения, для которых можно задать любую модель.

Сети Петри достаточно часто используются в качестве математического аппарата либо дополнительной модели для исследования в различных системах, занимающихся моделированием, как платных, так и бесплатных [9, 10]. Существует множество симуляторов сетей Петри, модели в первых из них представлялись в виде текста, а работа с пользователем велась в диалоговом режиме.

В данной работе будут рассмотрены следующие симуляторы: Alpha/Sim, Jsarp, QPNet, Renew и Yasper. Выбор именно этих продуктов обусловлен тем, что в настоящее время они достаточно широко развиваются. При их реализации учитывались особенности симуляторов на тот момент времени, и рассматривать системы-предшественники или схожие системы не имеет смысла. На их основе будет произведен сравнительный анализ по выбранным критериям.

Данный обзор позволит рассмотреть достоинства и недостатки различных симуляторов сетей Петри для расширения возможностей построенного редактора сетей Петри в системе TriadNS, а также его последующей интеграции с редактором для трансформации моделей в системе.

1. **Alpha/Sim**

Alpha/Sim – это средство моделирования дискретных систем общего назначения, основанное на математическом аппарате сетей Петри [13]. С помощью системы можно строить и отлаживать модели в графическом виде, отслеживать результаты в любой точке процесса выполнения моделирования, а также сохранять модели и результаты имитации.

Симулятор поддерживает выполнение множества функций анализа сетей Петри, а также использование сетей Петри различных видов. Более того, пользователь сам может написать функции, необходимые ему при анализе, либо импортировать уже написанный программный код.

Построение модели в Alpha/Sim достаточно удобно и происходит графически (Рисунок 12). Пользователю не нужно писать ни строчки кода, а только перетаскивать пиктограммы в нужное место и задавать параметры.

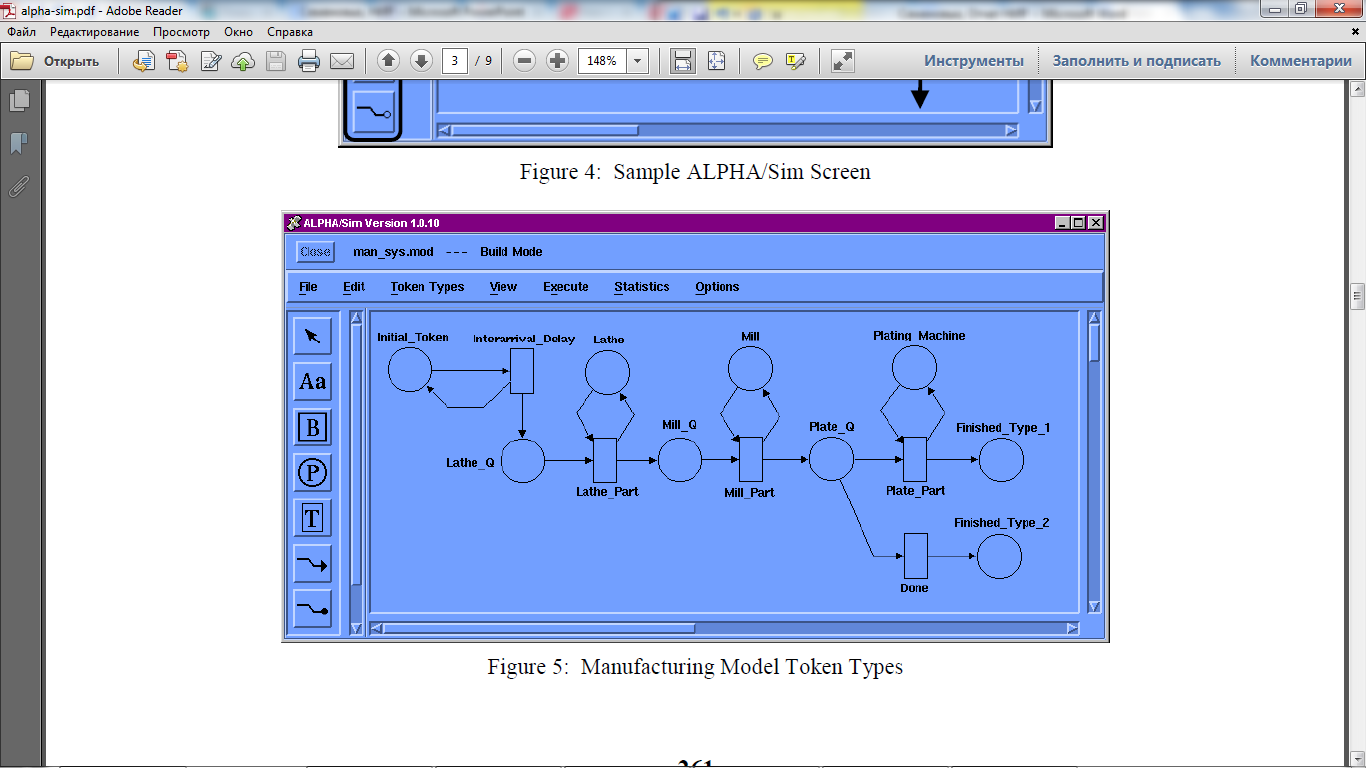


Рисунок 12. Проектирование СП в Alpha/Sim [13]

Перед выполнением симуляции Alpha/Sim автоматически проверяет модель на наличие ошибок, а симуляцию можно остановить на любом шаге выполнения или запустить без остановок. Однако при таких широких возможностях в процессе имитации нельзя запускать переходы вручную.

Результаты моделирования выводятся на экран или сохраняются в файл для дальнейшего анализа. Модель также можно просмотреть в виде дерева для удобной навигации.

Симулятор может быть запущен на персональном компьютере под платформой Windows или на серии рабочих станций (SUN Workstation) под SunOS, Solaris или Motif.

Недостатком рассматриваемого симулятора также может служить его стоимость. Он не распространяется бесплатно, не предоставляет открытый доступ к коду.

1. **Jsarp**

Jsarp – анализатор и симулятор сетей Петри, разработанный в Государственном университете Рио-де-Жанейро. Он обладает достаточно мощными средствами для анализа обыкновенных сетей Петри. Система еще развивается и находится на стадии расширения функционала: планируется добавить моделирование временных, стохастических сетей Петри [8].

Симулятор написан на языке Java, что позволяет использовать его на многих платформах, поддерживающих данную среду, а также с легкостью добавлять различные библиотеки, использовать шаблоны и интегрировать с другими системами. JSarp является бесплатным и предоставляет открытый доступ к программному коду [7].

Проектирование и моделирование сетей Петри в графическом редакторе симулятора происходит достаточно просто (Рисунок 13). Моделирование возможно остановить на любом шаге, а также вручную запустить любой переход. В процессе имитации выводится информация, в которую входит дерево достижимости, данные об элементах сети и т.д. После проведения моделирования пользователь может воспользоваться инструментами для анализа и проверить интересующие свойства сети Петри.

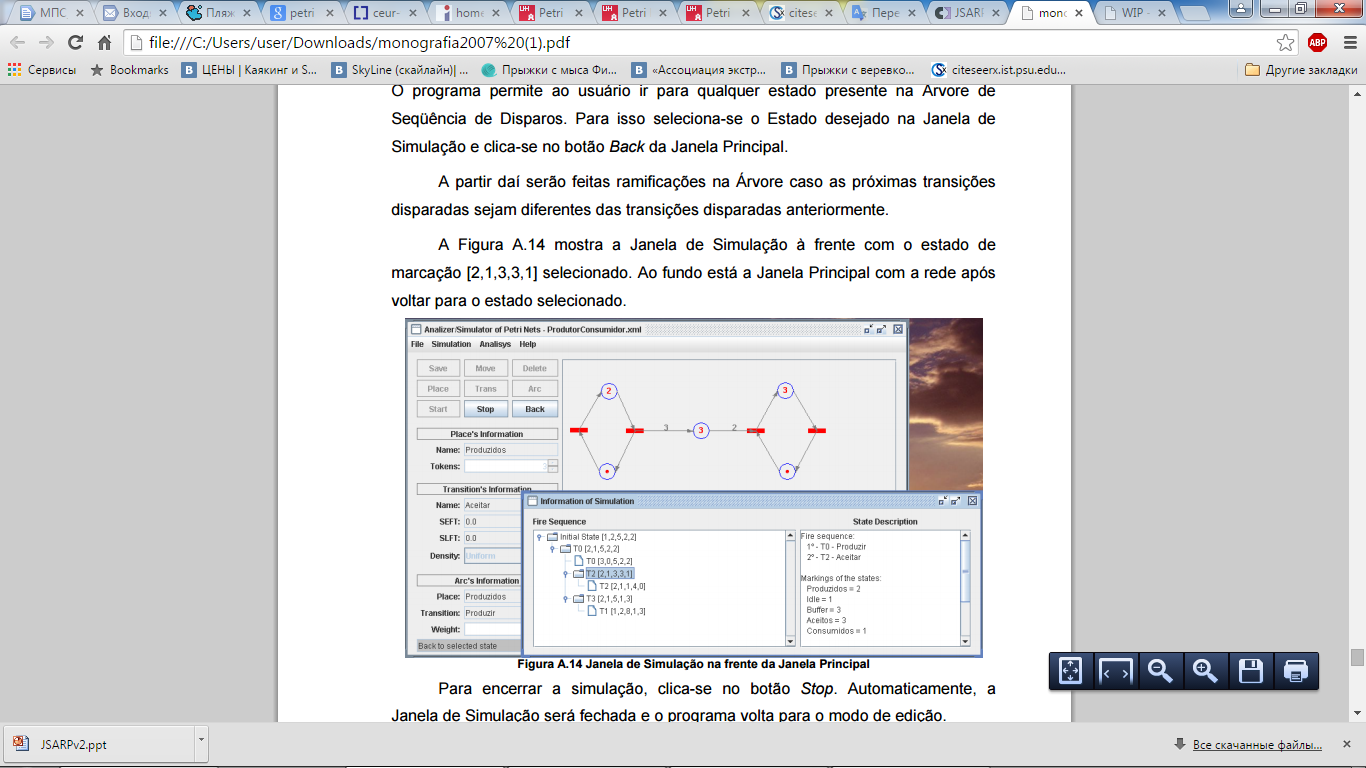


Рисунок 13. Симуляция в JSarp [7]

Построенную модель можно сохранить в файл формата XML для последующей работы с ней.

1. QPNet

QPNet – эмулятор сетей Петри, реализованный в «Московском государственном институте радиотехники, электроники и автоматики». Данная система позволяет строить сети Петри, а также сохранять построенные модели в собственном формате XQP.

Основные особенности эмулятора является его кроссплатформенность, высокая производительность [35]. QPNet разрабатывался с помощью библиотеки QT в среде C++. Данная система распространяется бесплатно и постоянно улучшается, также реализованы функции сбора статистики и задания некоторых распределений для веса дуги и задержки перехода.

Интерфейс системы для построения и моделирования модели достаточно удобен и понятен для пользователя (Рисунок 14), однако, система находится еще на стадии тестирования и расширения функционала. За счет этого вывод результатов, в том числе в процессе симуляции, происходит не достаточно наглядным способом. Система поддерживает моделирование синхронных сетей Петри, а также временных и стохастических СП.

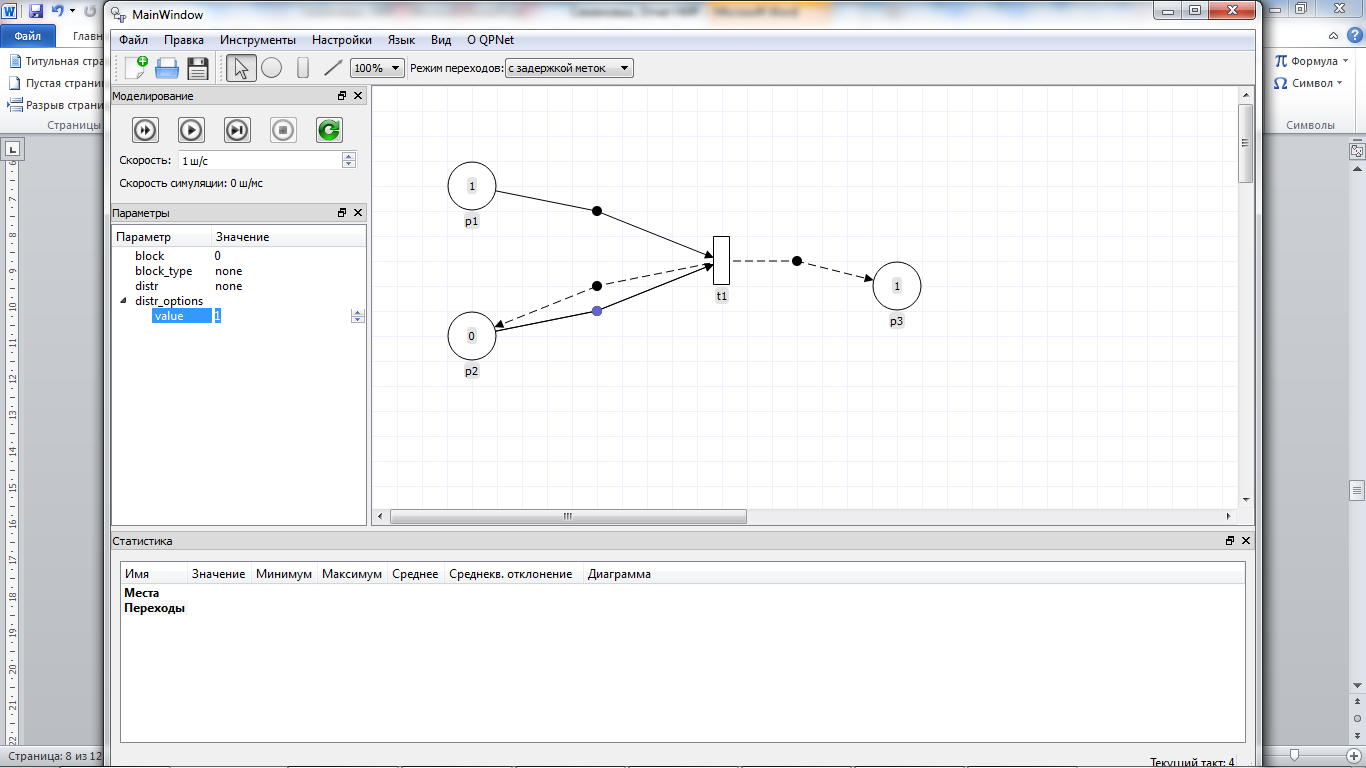


Рисунок 14. Интерфейс системы QPNet

1. Renew

Симулятор Renew предназначен для моделирования сетей Петри высокого уровня, написанный на языке Java, что позволяет использовать его на многих современных операционных системах [17]. Также система предоставляет открытый доступ к программному коду, поэтому она может быть расширяема третьими лицами. Renew – это учебная программа, она постоянно расширяется и улучшается, происходит исправление ошибок.

Как и во многих симуляторах сетей Петри, Renew имеет графический редактор, позволяющий достаточно просто проектировать различные виды сетей Петри (Рисунок 15). Также наряду с обычными инструментами для построения модели можно использовать возможности языка Java.

Моделирование в системе имеет несколько возможностей. По умолчанию имитация происходит автоматически, ее можно остановить на любом шаге, а также отладить работу модели. Более того, Renew поддерживать возможность ручного запуска выбранных переходов. При выборе такой возможности имитация происходит автоматически до тех пор, пока переходы, выбранные для ручного запуска, не станут потенциально активными. После того как пользователь запустит переход, имитация продолжится автоматически.

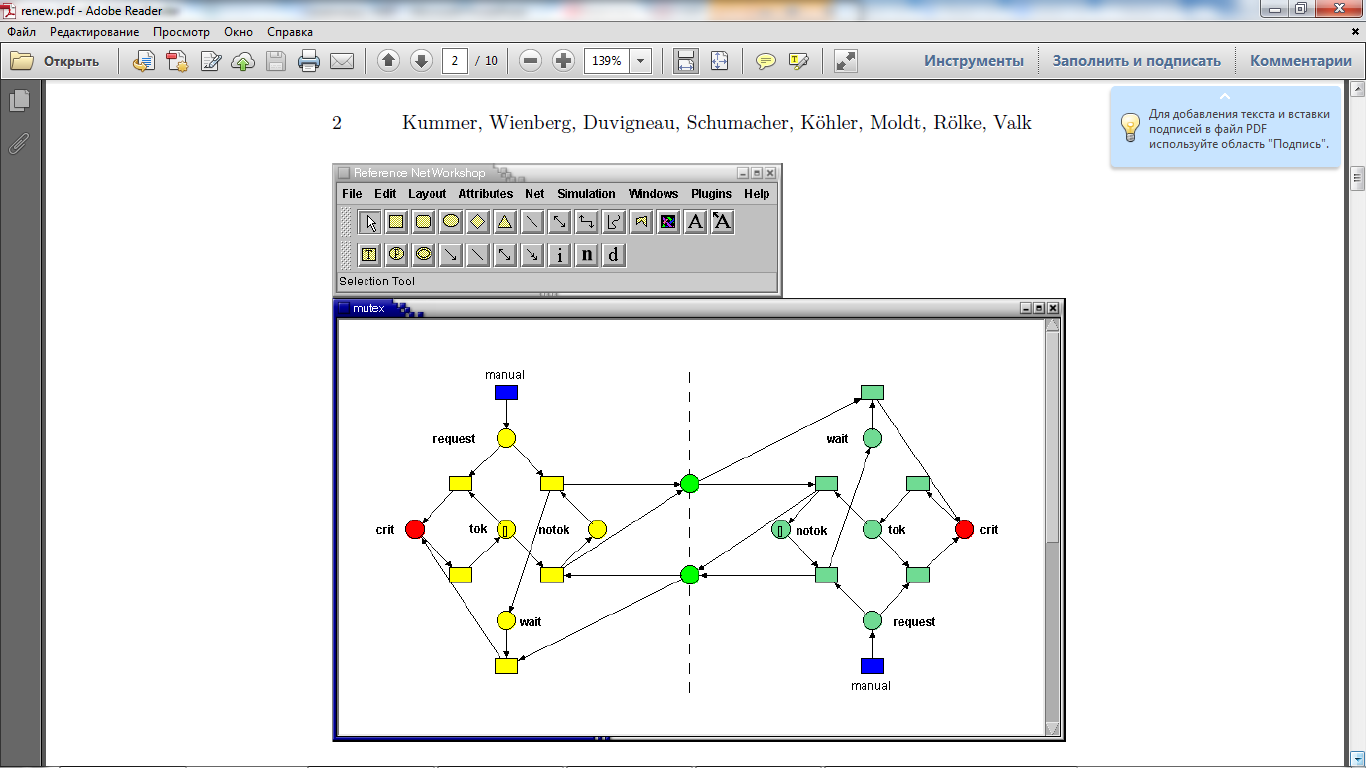


Рисунок 15. Симулятор Renew [17]

Симулятор Renew может предоставлять возможность построения сетей Петри различных видов: обыкновенные, временные, стохастический, раскрашенные и др.

Однако система не лишена недостатков. Одним из недостатков является отсутствие методов и функций анализа, встроенных в симулятор, что является достаточно важным, если с системой работают пользователи, не знающие в полной мере теории сетей Петри. Также использование языка Java при построении моделей ограничивает возможности пользователей, которые не знакомы с этим языком достаточно хорошо.

Существуют симуляторы сетей Петри, включающие все возможности данного симулятора, однако имеют более ясный и понятный интерфейс для конечного пользователя. Примером такого симулятора может быть Yasper [24].

## Выводы

После проведения обзора симуляторов сетей Петри можно заключить, что рассмотренные системы имеют множество достоинств, но также не лишены недостатков. Сравнительная таблица симуляторов приведена в Таблица 1. Также в таблице представлено сравнение симуляторов не только друг с другом, но и с разработанной в TriadNS системой проектирования и моделирования сетей Петри.

Таблица 1. Сравнительная таблица симуляторов сетей Петри

| **Критерии сравнения** | **Alpha/Sim** | **JSarp** | **QPNet** | **Renew** | **TriadNS** |
| --- | --- | --- | --- | --- | --- |
| Удобство построения | + | + | + | + | + |
| Удобство представления | + | + | + | + | + |
| Визуализация процесса моделирования | + | + | + | + | + |
| Ручной запуск переходов | - | + | - | + | + |
| Наглядное отображение результатов анализа | + | + | - | - | + |
| Использование различных видов сетей Петри | + | - | + | + | + |
| Бесплатное распространение | - | + | + | + | + |
| Дополнительные возможности | Кроссплатформенность Пользовательские функции | Кроссплатформенность | Кроссплатформенность | Кроссплатформенность  Возможность использования Java | Возможность задать любое поведение |

Каждый из симуляторов является кроссплатформенным, что является немаловажным и позволяет запускать систему на многих операционных системах. Все рассмотренные симуляторы имеют дружественный и понятный интерфейс и ясное представление сетей Петри в графическом редакторе, визуализация в процессе имитации показывает каждое состояние сети Петри. При этом некоторые симуляторы (JSarp, Renew) предоставляет возможность запуска переходов вручную.

Достаточно важным фактором сравнения симуляторов сетей Петри является наглядное отображение результатов анализа, однако некоторые системы не предоставляют таких инструментов (QPNet, Renew).

Достоинством системы Alpha/Sim по сравнению с другими системами является возможность задания пользовательских функций для анализа модели, однако при этом система не бесплатна. JSarp, по сравнению с другими системами, обладает минимальным набором средств для построения сетей Петри и всеми необходимыми инструментами для анализа, но не поддерживает работу с различными видами сетей Петри, хотя реализация такой возможности имеется в планах разработчиков. Renew обладает наиболее мощными средствами для построения сетей Петри, которые предоставляют возможность использования языка Java.

Опыт разработки рассмотренных симуляторов был использован и при разработке инструментальных средств в TriadNS для реализации графического редактора моделей, представленных в терминах сетей Петри, для разработки соответствующего визуального языка и средств анализа построенных моделей.

Следующая глава будет посвящена системе TriadNS. В ней будет рассмотрена архитектура системы, представление моделей, а также представлены основные достоинства.

# Система TriadNS

Система TriadNS разрабатывается с использованием средств системы имитационного моделирования Triad.Net. Система Triad.Net является развитием системы автоматизированного проектирования Triad, которая была разработана на кафедре математического обеспечения вычислительных систем Пермского государственного университета и предназначалась для проектирования встроенных вычислительных систем.

## Архитектура системы

Система имитационного моделирования TradNS предназначена для проектирования и моделирования компьютерных сетей, а также их исследования. Система в настоящее время широко развивается, в том числе в рамках данной работы необходимо реализовать дополнительный модель для исследования компьютерных сетей с помощью сетей Петри.

СИМ Triad.Net включает следующие компоненты [33]: компилятор, ядро, графический редактор, подсистему отладки, подсистему валидации, подсистему синхронизации распределенных объектов модели, подсистему балансировки (распределенная версия), подсистему организации удаленного доступа, подсистему защиты от внешних и внутренних угроз, подсистему автоматического доопределения модели. Назначение основных компонентов представлено ниже:

* TriadCompile (компилятор языка Triad, переводит описание имитацион­ной модели с языка Triad во внутреннее представление);
* TriadDebugger (отладчик, использует механизм информационных процедур алгоритма исследования, локализует ошибки и вырабатывает рекомендации для их устранения на основании правил из базы данных, для каждого класса ошибок осуществляется поиск по онтологии соответствующего обработчика ошибок);
* TriadCore (ядро системы, включает библиотеки классов основных элементов модели),
* TriadEditor (редактор моделей, предназначен для работы с моделью как в удаленном, так и локальном режимах, локальный режим предполагает работу с системой в том случае, если нет удаленного доступа),
* TriadBuilder (подсистема автоматического доопределения частично описанной модели), база данных, где хранятся экземпляры элементов модели.

## Описание модели

Triad.Net – событийно ориентированная система имитационного моделирования. Основным способом моделирования в системе является описание модели с помощью специального языка Triad. Основная его особенность - это разделение модели на слои.

Описание имитационной модели в Triad состоит из трех слоёв: слоя структур (STR), слоя рутин (ROUT) и слоя сообщений (MES). Таким образом, модель в системе Triad.Net можно определить как M={STR,ROUT,MES} [25].

Слой структур представляет собой совокупность объектов, взаимодействующих друг с другом посредством посылки сообщений. Каждый объект имеет полюса (входные Pin  и выходные Pout), которые служат соответственно для приёма и передачи сообщений. Слой структур можно представить графом , где V– множество вершин, W – множество дуг (ребер), U – множество полюсов. Связи устанавливаются между полюсами различных вершин. В каждой вершине выделяются два подмножества – входных полюсов In(v) и выходных полюсов Out(v). На входные полюса могут поступать сообщения, с выходных передаваться. Некоторые полюса могут быть одновременно входными и выходными. Дуги графа это пары полюсов типа , при этом .

Объекты действуют по определённому алгоритму поведения, который описывают с помощью рутины. Рутина представляет собой множество событий , планирующих друг друга. Выполнение события сопровождается изменением состояния  объекта. Состояние объекта определяется значениями переменных  рутины (, Var – множество переменных рутины). Таким образом, рутину можно представить как [39] , Здесь T – множество моментов времени (линейно упорядоченное абстрактное множество с отношением порядка ≤),  - конечное множество событий, - конечное множество состояний рутины, определяемых значениями переменных рутины . Естественно, множество значений переменных ограничено.  – начальное состояние рутины, InitE – отношение, определяющее начальное состояние рутины:  определяет для каждого события  момент его первоначального наступления. Ch – отношение изменение состояний, является отображением . Это отображение указывает, в какое состояние переходит рутина в результате наступления события. Sch – отображение , это отображение, каждой четвёрке  ставит в соответствие момент времени t’, что означает, что в результате срабатывания события е в момент времени t при состоянии руины q, событие е’ будет запланировано на момент времени t’. Рутины, как и вершины структуры, имеют входные In(R) и выходные Out(R) полюса. Во множестве событий рутины выделено входное событие ein. Все входные полюса рутины обрабатываются входным событием. Обработка выходных полюсов осуществляется остальными событиями рутины. Для передачи сообщения служит специальный оператор out. Совокупность рутин определяет слой рутин ROUT.

Элементарная рутина связывается с вершиной графа. Отображение наложения ставит в соответствие полюсы и события: , где – вершина – множество полюсов, E – множество событий рутины.

Слой сообщений (MES) предназначен для описания сообщений сложной структуры.

Описание модели может проходить поэтапно, начиная с описания структуры. Все слои описывать не обязательно. Для каждого слоя есть свой круг задач, формулируемых и решаемых в случае его задания.

## Использование онтологий

В СИМ Triad.Net онтологии используются для различных целей: построение моделей, проверка моделей на правильность, а также для доопределения частично описанной модели [26], т.е. некоторого устройства, например, маршрутизатора в компьютерной сети. Это бывает необходимо на начальных этапах моделирования, когда исследователю или проектировщику необходимо получить оценки моделируемого объекта или системы, даже если они являются достаточно грубыми. При этом поиск компонента модели (экземпляра рутины) в базе данных осуществляется с помощью онтологий автоматически.

Для оперативного построения имитационной модели в конкретной предметной области используют графический интерфейс [25], добавление нового компонента модели (экземпляра рутины) и настройка на конкретную предметную область выполняется с помощью онтологий.

Онтологии используются и в моделировании, в частности на различных этапах имитационного моделирования, начиная с этапа сбора информации и заканчивая этапом валидации модели.

Для настройки на конкретную предметную область в системе используются онтологии. В Triad.Net разработана базовая онтология, которая является мета-метамоделью для построения метамоделей (Рисунок 16). Ее основу составляют следующие классы:

* TriadEntity – любая логическая сущность языка Triad, имеющая имя. Подклассами TriadEntity являются все классы базовой онтологии;
* Model – имитационная модель;
* ModelElement - составная часть имитационной модели, а также все, чем может быть специализирована вершина структуры имитационной модели.

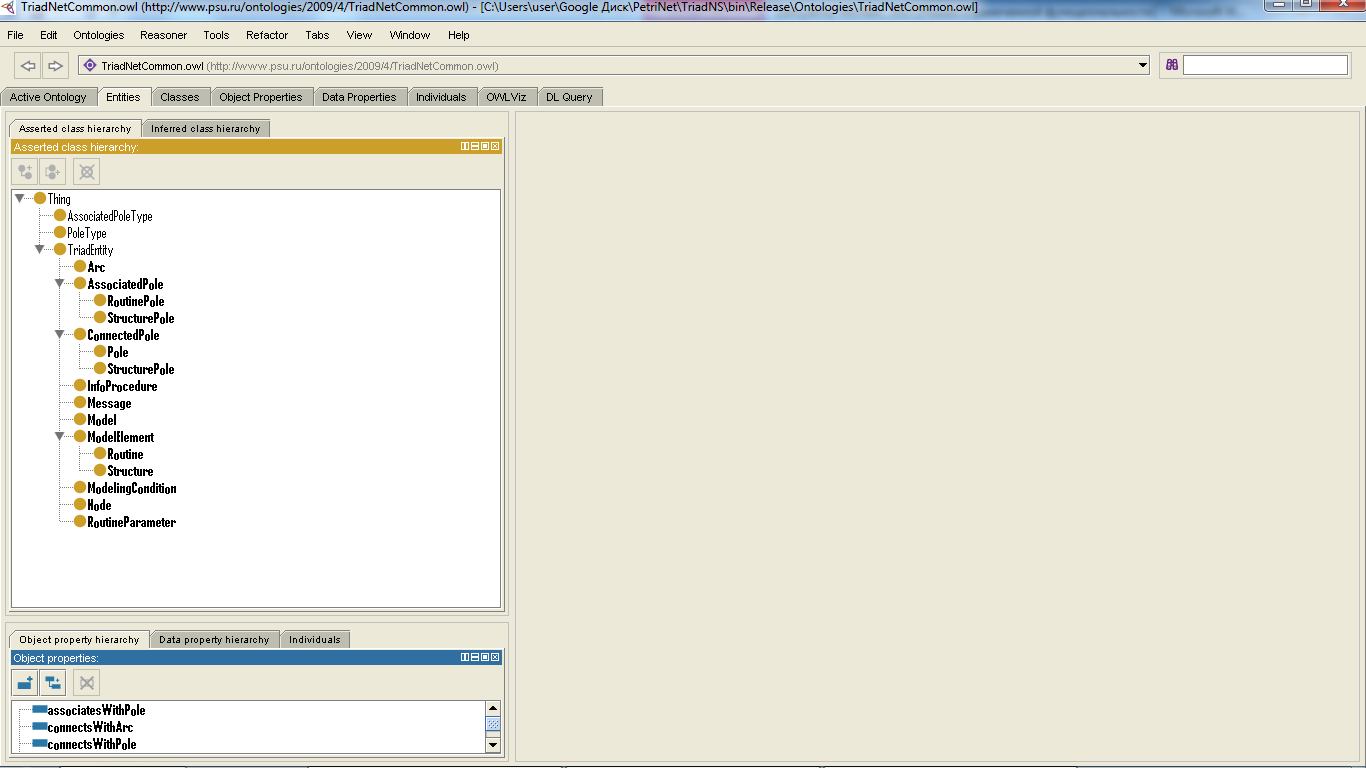


Рисунок 16. Иерархия классов базовой онтологии

Подклассами ModelElement являются классы:

1. Structure – структура имитационной модели.
2. Routine – рутина.

Основными свойствами в базовой онтологии являются следующие:

1. Свойства владения чем-либо: модель имеет структуру, структура имеет вершину, вершина имеет полюс и т.д.
2. Свойства принадлежности к че0му-либо – inverse properties по отношению к соответствующим свойствам владения – структура принадлежит модели, вершина принадлежит структуре, полюс принадлежит вершине и т.д.
3. Свойства, связывающие полюс с присоединенной к нему дугой: connectsWithArc (Pole, Arc), connectsWithPole (Arc, Pole).
4. putsOn (Routine, Node) – свойство, связывающее вершину с наложенной на нее рутиной.
5. modelingToCondition (Model, ModelingCondition) – свойство, связывающее модель с условием моделирования.

Онтология системы TriadNS дополняет базовую онтологию. В зависимости от того, какая предметная область проектируется, используются различные онтологии. Для проектирования компьютерных сетей используется онтология, в которой введены специализированные подклассы основных классов базовой онтологии. Также были добавлены некоторые свойства.

Для построения в системе TriadNS сетей Петри была разработана онтология данной предметной области. К базовой онтологии были добавлены соответствующие подклассы. Загружая данную онтологию, пользователь может строить сети Петри, задавать стандартное поведение для элементов полученной модели и моделировать ее поведение. Имитационная модель представляется графически. Для работы с онтологиями в языке C# используется библиотека OWL API.

В системе имитационного моделирования Triad.Net онтологии также используются для автоматического доопределения частично описанных моделей, так как для проведения моделирования необходимо, чтобы поведение каждой вершины было определено. Все возможные рутины элементов описываются в онтологии. Стандартные рутины имеют параметры, которые пользователь может изменять до запуска моделирования. Стандартным поведением вершин сети Петри выступают вершины-позиции с одной входной и/или одной выходной дугами и вершины-переходы с одной входной и одной выходной дугами.

Для каждого элемента сети можно определить произвольное количество рутин, описав их на языке Triad и добавив в онтологию. При добавлении рутины следует указать необходимость соединения каждого полюса рутины и семантический тип возможной смежной вершины, значения по умолчанию для указанных в рутине параметров. Так, например, для сетей Петри вершины могут быть соединены дугой, если они имеют различный тип, т.е. вершина-позиция может быть соединена только с вершиной-переходом.

## Основные особенности системы

Система Triad.Net является параллельной (распределённой). В ней реализованы распределённые алгоритмы синхронизации функционирующих во времени объектов (консервативный и оптимистический)[26]. Кроме того, для системы Triad.Net характерно следующее:

1. Входной язык описания моделей содержит переменные типа «модель». Над моделями определены операции. Операции определены как для моделей в целом, так и для каждого слоя [39]. Например, в слое структур можно выполнить добавление и удаление вершины, добавить или удалить ребро (дугу), полюсы, найти объединение или пересечение графов. Кроме того, вершине из слоя структур можно приписать (по определённым правилам) ту или иную рутину из слоя рутин, тем самым изменив алгоритм её поведения, и т.д. Таким образом, имитационная модель может быть описана языковыми средствами, а может быть построена в результате исполнения некоторого алгоритма преобразования модели.
2. Модель является иерархической, т.е. каждая вершина в слое структур может быть расшифрована подструктурой.
3. Подсистема анализа модели не ограничивает пользователя строго регламентированным набором собираемых данных.

Что касается выполнения, то преимуществом системы Triad.Net является то, что построенная пользователем модель не интерпретируется, а выполняется. Так достигается значительный прирост в эффективности. То сколько раз прогоняется модель, какая статистика собирается – все это описывается на уровне языка Triad в специальном разделе – условия моделирования. То, какая статистика собирается и как, тоже описывается на уровне языка Triad. Это задается в информационных процедурах модели и условиях моделирования, они реализуют алгоритм исследования модели.

Также преимуществом является и тот факт, что система способна работать с различными моделями и настраиваться на любую предметную область. В этом случае для каждой модели должна быть построена метамодель в виде онтологии, и при работе с новыми моделями система настраивается на предметную область.

## Доработка системы

В рамках данной работы для работы с дополнительными моделями необходимо реализовать графический редактор для работы с моделью, разработать метамодель, а также редактор, позволяющий трансформировать одну графическую модель в другую на основе некоторых правил.

Разработка перечисленных инструментальных средств позволит применить многомодельный подход в имитационном моделировании на примере исследования компьютерных сетей в виде сетей Петри. Это означает, что при исследовании сложных систем можно рассмотреть их с разных точек зрения, с помощью различных моделей, а значит, использовать новые методы анализа.

Так, например, при исследовании систем с помощью сетей Петри можно проанализировать их поведение, получить информацию с целью улучшения поведения исходной системы.

## Выводы

В работе уже были рассмотрены основные понятия модели, с помощью которой будут исследоваться компьютерные сети в разрабатываемой системе TriadNS. В следующей главе будут освящены вопросы разработки правил для трансформации модели компьютерной сети в сеть Петри.

С помощью данных правил пользователь системы сможет переводить модель, представленную в виде графа на некотором визуальном языке в TriadNS, в сети Петри и исследовать ее с помощью средств анализа сетей Петри.

# Разработка правил трансформации модели компьютерной сети в сеть Петри

Для того чтобы реализовать модуль, предназначенный для трансформации одной модели в другую, в системе имитационного моделирования TriadNS, необходимо разобраться, какими способами возможен переход от одной модели к другой в системе, разработать правила трансформации, построить математическую модель и формализовать полученные правила трансформации моделей.

Как было сказано ранее, перевод из одной модели в другую может быть произведен различными способами, в зависимости от целей решаемой задачи. Более того, можно переводить не только статическую структуру графа, но и трансформировать поведение. В данном случае, так как для имитационного эксперимента важно поведение каждого элемента модели, то необходимо трансформировать и поведение.

## Понятие графовых грамматик

Графовой грамматикой называется четверка (*T, N, P, S*), где *T* – множество терминальных символов, *N* – множество нетерминальных символов, *P* – множество правил вида *L → R*, причем *L* – непустая последовательность терминальных и нетерминальных символов, содержащая хотя бы один нетерминальный символ, *R* – любая последовательность терминальных и нетерминальных символов, *S* из *N* – стартовый (начальный) символ [19].

Для того чтобы выполнять преобразования графов, необходимо описать некоторые правила, по которым фрагментам одного графа будут соответствовать конструкции другого графа. В правой части правила будет содержаться целевой граф, а в левой части подграф исходного графа [40].

Пусть даны четыре помеченных графа *G*, *H*, *L*, *R*, причем граф *L* является подграфом графа *G*, а граф *R* является подграфом графа *H*. Применением правила *L* → *R* к исходному графу *G* называется замена в графе *G* подграфа *L* на граф *R*, результатом замены будет граф *H*. Трансформация модели – последовательное применение правил к исходному графу.

Данный подход можно применить и в случае трансформации визуальных моделей. На основе базовых конструкций модели необходимо определить правила, по которым будет происходить трансформация и последовательно применять заданные правила, пока это возможно.

## Трансформация слоя структур

Наиболее простым преобразованием модели компьютерной сети в сеть Петри является трансформация структуры, то есть преобразование одного графа в другой. Правила трансформации описаны в Таблица 2.

Таблица 2. Трансформация структуры модели

|  |  |
| --- | --- |
| C:\Users\user\Documents\GitHub\Tmp\ComputerTab\TriadNS\bin\Debug\img\comp.png | C:\Users\user\Documents\GitHub\Tmp\ComputerTab\TriadNS\bin\Debug\img\node.png |
| C:\Users\user\Documents\GitHub\Tmp\ComputerTab\TriadNS\bin\Debug\img\server.png | C:\Users\user\Documents\GitHub\Tmp\ComputerTab\TriadNS\bin\Debug\img\node.png |
| C:\Users\user\Documents\GitHub\Tmp\ComputerTab\TriadNS\bin\Debug\img\router.png | C:\Users\user\Documents\GitHub\Tmp\ComputerTab\TriadNS\bin\Debug\img\node.png |
|  |  |
|  |  |

Такой способ трансформации является элементарным и в большинстве случаев не несет достаточной информации согласно целям моделирования. С помощью таких правил нельзя четко проследить, что именно происходит внутри каждого элемента модели, как меняется их состояние, какие события происходят (Рисунок 17).

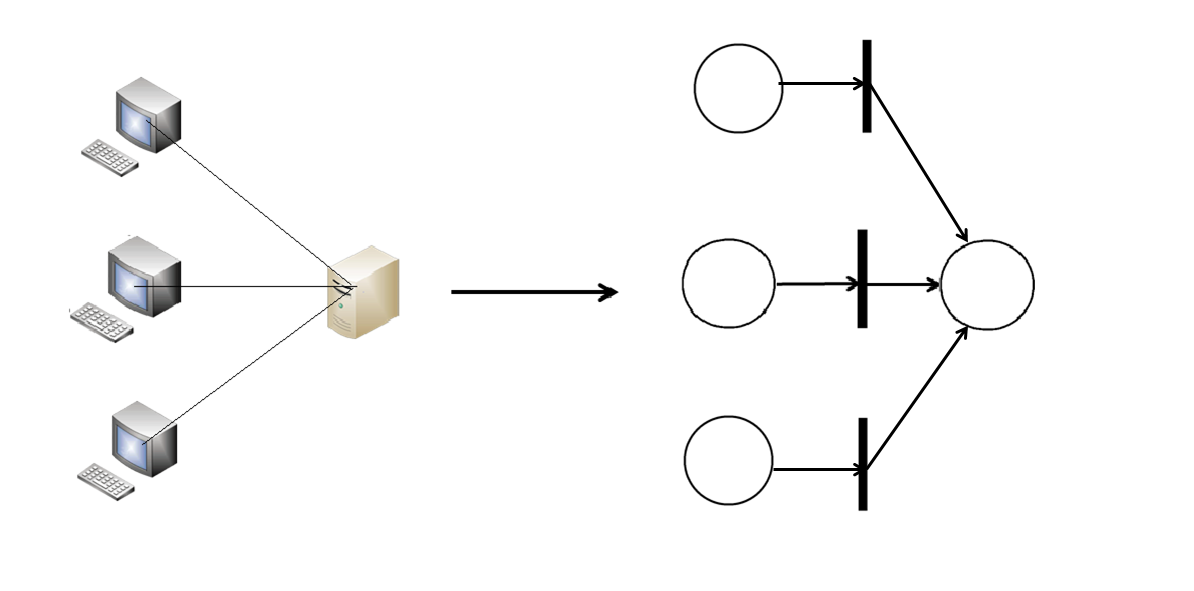


Рисунок 17. Пример преобразования слоя структур

Предложенный способ трансформации может быть полезен, когда необходимо проанализировать поведение системы, если известны только элементы модели и их взаимосвязи, но неизвестно поведение, либо проследить за передачей сообщений от одного узла компьютерной сети к другому узлу.

## Трансформация слоя рутин

Чтобы перевести поведение сущности модели компьютерной сети, необходимо трансформировать алгоритм ее поведения. Любой алгоритм, формализованный на некотором языке (процедурный язык), можно представить в виде блок-схемы, а правила перевода блок-схемы в сеть Петри указаны в [42] и представлены на Рисунок 18.

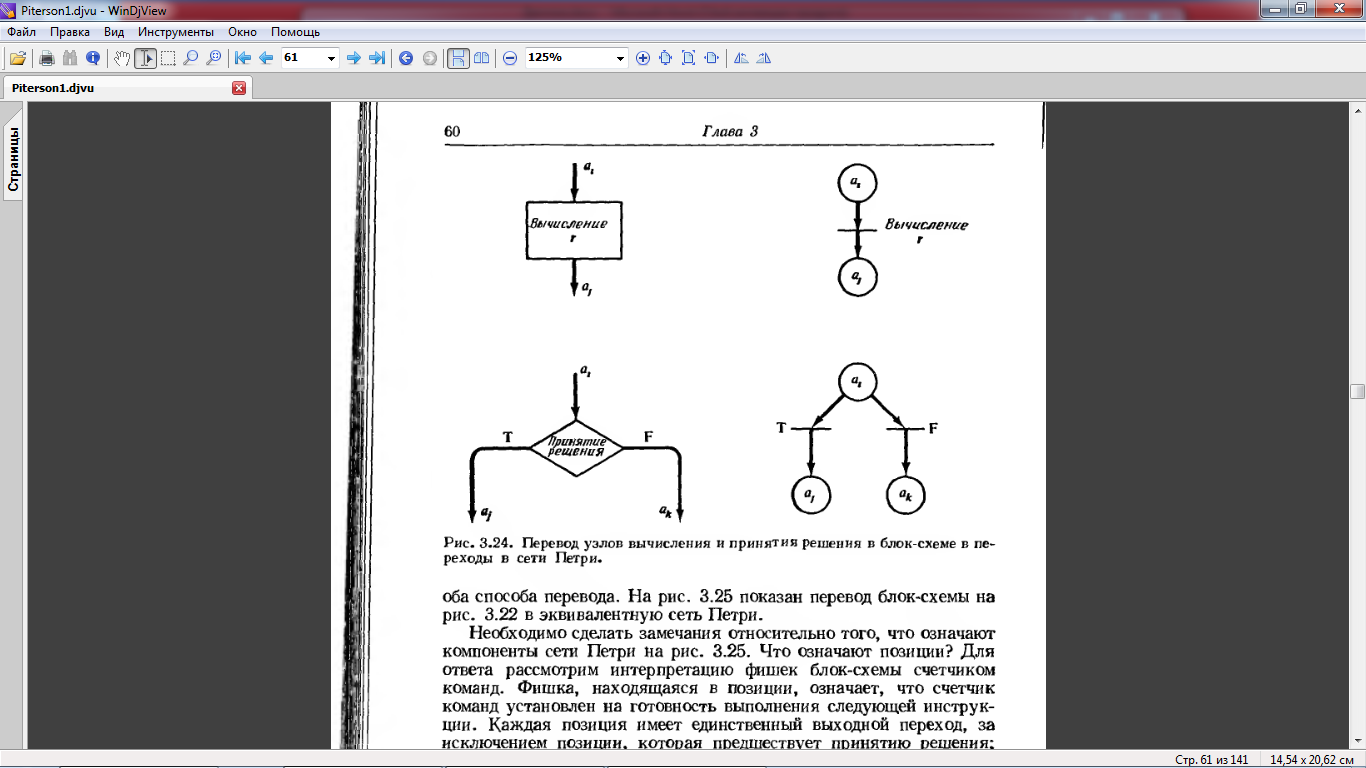


Рисунок 18. Перевод узлов вычисления и принятия решения в блок-схеме в сеть Петри

Для примера такой трансформации опишем поведение элементов системы: сервера и клиента в виде рутины на языке Triad. Рутины приведены в Приложение B, блок-схемы в Приложение C. Для демонстрации данного примера была построена блок-схема для рутины «Клиента» и «Сервера» на языке Triad.

В качестве примера произведена трансформация полученных блок-схем в сеть Петри. Такой пример моделирует работу одного клиента, подключенного к одному серверу (Рисунок 19). Хотя количество элементов мало, но анализ такой сети достаточно трудно выполнить, тем более при увеличении количество элементов компьютерной сети.

Таким образом, рассмотренная трансформация моделей в большинстве случаев является избыточной, так как изменение отдельных переменных, описанных в блоке initial рутины и отвечающих за состояние сущности, чаще всего не влияет на общее поведение системы. Из-за этого представление в виде сети Петри становится запутанным, неинформативным. Такая трансформация может использоваться в тех случаях, когда нам необходимо исследовать поведение системы полностью, вплоть до изменения каждой переменной в рутине, например, при отладке поведения.

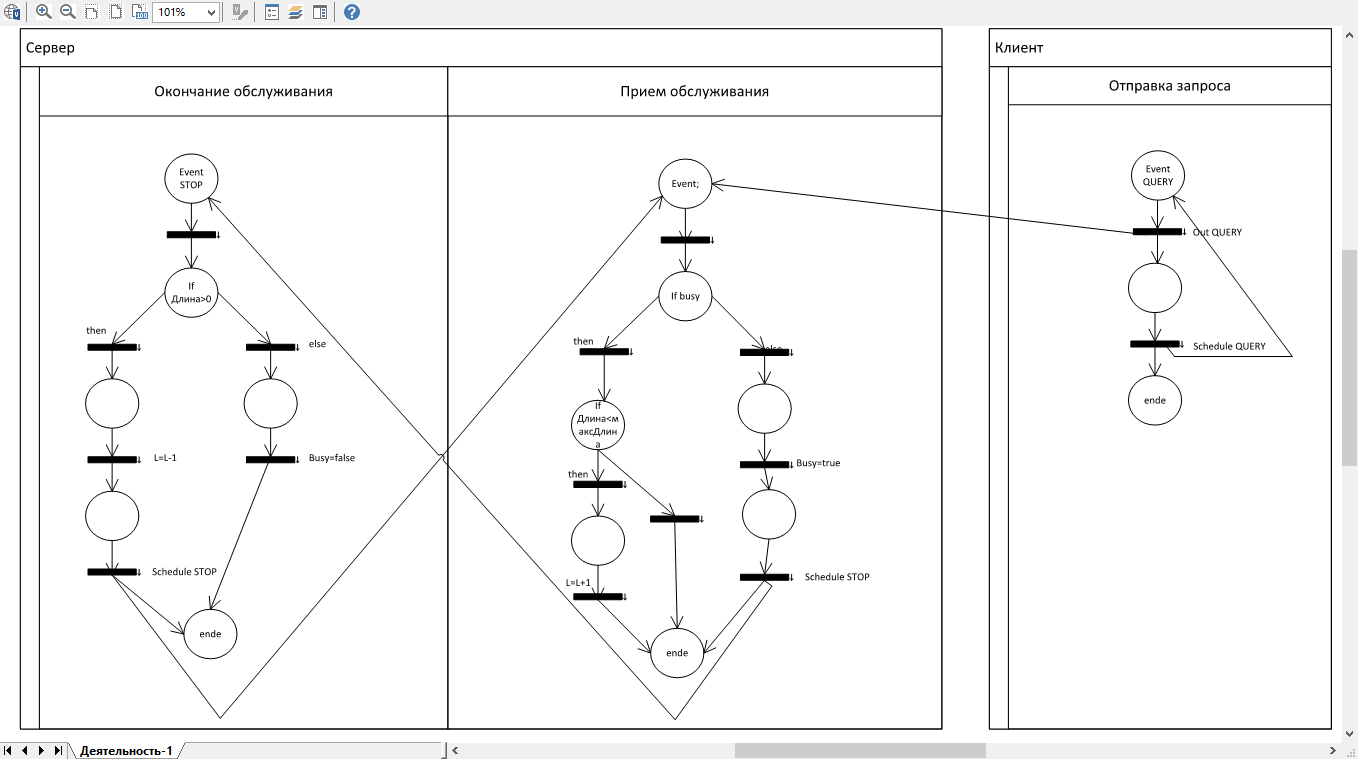


Рисунок 19. Сеть Петри, полученная при трансформации рутины

## Трансформация модели на основе понятий активность и событие

Как уже было показано ранее для перевода модели компьютерной сети в сеть Петри возможны несколько способов, также были приведены примеры трансформации. Более того было установлено, что простое преобразование статической структуры элементов недостаточно для эффективного использования дополнительной модели, а если переводить алгоритм поведения элементов модели полностью, то полученная модель становится слишком избыточной и перестает быть информативной. Таким образом, необходимо выбрать оптимальный способ представления модели в виде сети Петри.

Простое представление системы в виде сетей Петри основано на понятиях условие и событие. Условие определяет возможность возникновения события в системе, т.е. некоторых действий в системе. Оно может быть представлено в виде предиката, принимающего значения истина или ложь. Состояние системы может быть описано как некоторое множество условий системы [14].

Для того чтобы событие произошло необходимо возникновение некоторых условий в системе, которые являются предусловиями, а результатом выполнения действий являются постусловия.

Описанное выше представление системы в виде условий и событий легко моделируется в виде сетей Петри. При этом предусловиями являются позиции с выходными дугами, постусловиями – позиции с входными дугами, а событиями – переходы, а выполнение условия обозначается маркером в позиции [41]. Также необходимо учесть, что событие может выполняться при выполнении нескольких условий и результатом также может быть композиция постусловий. Для таких целей необходимо использовать операции И/ИЛИ в виде сетей Петри.

Операция И аналогична параллельному выполнению действий, а операция ИЛИ аналогична последовательному выполнению действий, при этом если операция И применяется к предусловиям это представляет операцию JOIN, а если к постусловиям – операцию FORK (Рисунок 20).

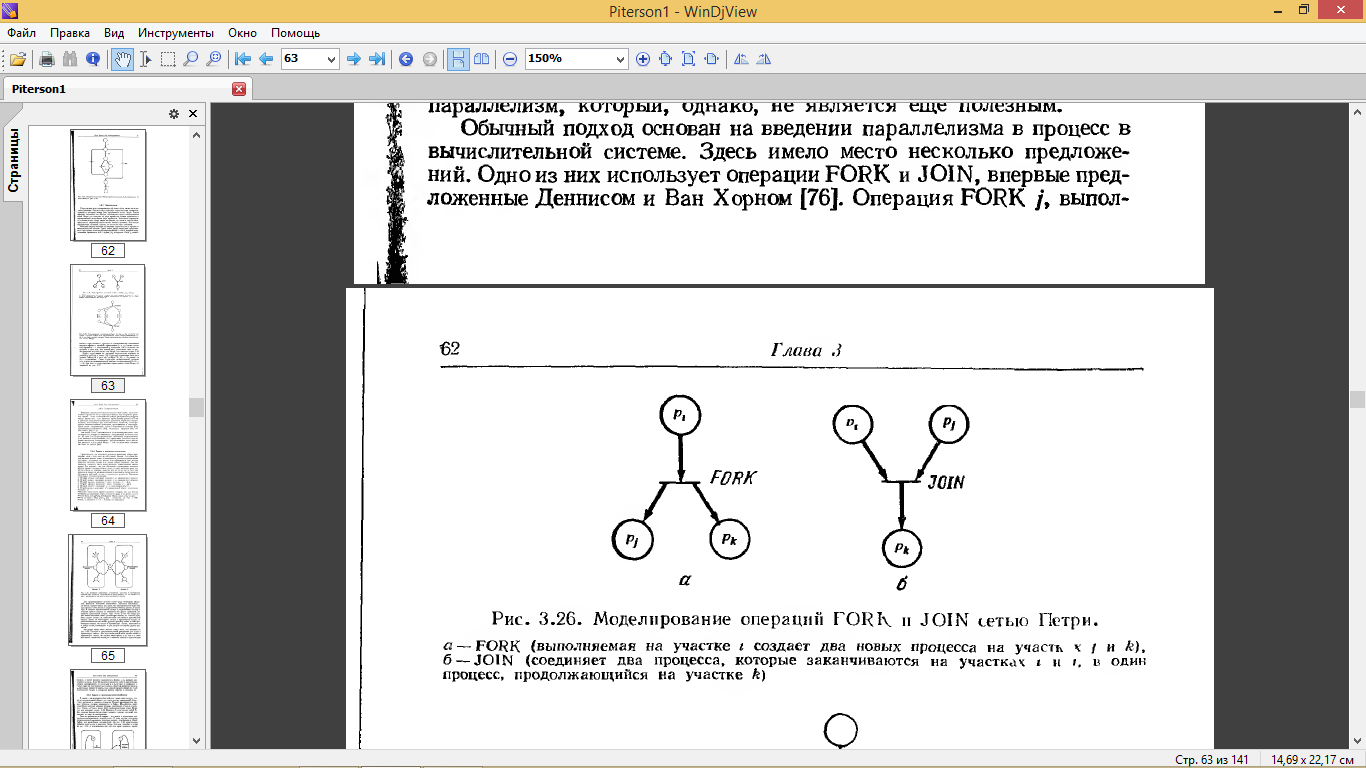


Рисунок 20. Операции FORK и JOIN, представленные сетями Петри [42]

Такое представление сетей Петри является некоторой вспомогательной, концептуальной моделью, с помощью которой можно представить практически любую систему в виде сетей Петри. Для того чтобы унифицированным образом перевести исследуемую модель в сети Петри представим этот перевод в виде графовой трансформации. Для более наглядного перевода введем следующие типы вершин, представленные в Таблица 3.

Таблица 3. Типы вершин в концептуальной модели сетей Петри

|  |  |
| --- | --- |
| ***Тип вершины*** | ***Графическое изображение*** |
| Событие |  |
| Условие |  |
| И |  |
| ИЛИ |  |

С помощью приведенных обозначений введенных вершин удобно строить граф, а также задавать графовые трансформации. Приведем правила трансформации, которые могут перевести описанную концептуальную модель условий и событий в сеть Петри (Таблица 4).

Таблица 4. Правила трансформации модели событий и условий в сеть Петри

|  |  |  |
| --- | --- | --- |
| ***Имя правила*** | ***Левая часть правила*** | ***Правая часть правила*** |
| ИЛИ1 |  |  |
| ИЛИ2 |  |  |
| ИЛИ3 |  |  |
| И1 |  |  |
| И2 |  |  |
| И3 |  |  |
| Событие |  |  |
| Условие |  | C:\Users\user\Documents\GitHub\Tmp\ComputerTab\TriadNS\bin\Debug\img\node.png |

При построении такой концептуальной модели и использовании правил также нужно учитывать ограничения:

1. Кратность связи между вершинами типа событие и операций (И или ИЛИ) 1:1. Это означает, что несколько событий не могут быть соединены с одной и той же операцией;
2. Если вершина типа операция имеет несколько входящих дуг, то она имеет только одну выходящую дугу;
3. Если вершина типа операция имеет несколько выходных дуг, то входящая у нее только одна;
4. Вершины типа операция имеют по крайней мере одну выходящую и одну входящую дугу;
5. Если из события исходит дуга в вершину типа операция, то выходящая дуга из этой вершины ведет либо в вершину типа условие;
6. Если из вершины типа операция исходит дуга в событие, то входные дуги исходят из вершин типа условие или операция.

По приведенным правилам в качестве примера рассмотрим перевод рутины в системе TriadNS, написанной на языке Triad. Для этого рассмотрим, что в рутине является условиями, событиями, в каких состояниях может быть система и определим связи между этими понятиями.

Концепция рутины в системе TriadNS определяется на основе элементарных понятий – активность и событие, причем активность является неделимым действием и приводит к изменению состояния устройства, а иногда к появлению выходного сообщения. Хотя изменение состояния может происходить постепенно, мы всегда относим это изменение к окончанию активности. Событие – это мгновенное изменение в процессе работы устройства, например, начало или конец активности, приход или выдача сообщений.

Как было описано выше, отображение наложения ставит в соответствие полюсы и события. Это значит, что при поступлении сообщения на некоторый полюс, происходит событие «приема сообщений», которое может инициировать целую цепочку последующих событий. Если выходное событие связано с полюсом вершины, то во время возникновения этого события, на полюсе появится выходное сообщение.

Таким образом, отправка сообщения через каждый полюс устройства должна считаться отдельным событием, а прием сообщения с любого полюса объединяется в одно событие приема сообщений и вызывает единственное событие, которое не планируется с помощью конструкции schedule – событие приема сообщений event;.

Важным является отображение событий, происходящих в системе, а также условий их возникновения. Исходя из формулировки соответствия элементов сети Петри условиям и событиям: события моделируются переходами, а условия – позициями, - рутина, основанная на концепции элементарных понятий активности и событий, легко моделируется с помощью сетей Петри.

Используя такой подход, необходимо учесть, что все-таки некоторые события могут изменять состояние устройства, и это изменение может повлиять на поведение и его необходимо учесть. В этом случае необходимо определить такие состояния, которые наряду с другими условиями (прием сообщений или планирование события) будут управлять переходами в сети Петри.

События, происходящие в рутине, указаны явно. Условиями их возникновения являются отправка сообщения (out), отправка времени возникновения события в календарь событий (schedule), определенное состояние система или их композиция. Исходя из данных рассуждений, необходимо определить события, происходящие в системе, а также условия их возникновения. В Таблица 5 представлены события в исследуемой системе.

Таблица 5. События в исследуемой системе

| ***Формализация на языке Triad*** | ***Событие*** |
| --- | --- |
| Event QUERY; … ende | Отправка запроса |
| Event;  …  Ende | Прием сообщения |
| Event STOP;  …  ende | Окончание обслуживания |

Далее в Таблица 6 перечисленные условия, возникающие в системе, а также события, для которых данные условия являются входами, т.е. предусловиями, и выходами, т.е. постусловиями. Из данной таблицы можно заключить в каких отношениях находятся условия, т.е. в каких случаях нужно применить операцию И или ИЛИ.

Таблица 6. Условия в исследуемой системе

| ***Формализация на языке Triad*** | ***Условие*** | ***Предусловие для события*** | ***Постусловие для события*** |
| --- | --- | --- | --- |
| Schedule QUERY | Планирование отправки сообщения | Отправка запроса | Отправка запроса |
| Out “Query” | Выдача сообщения | Отправка сообщения через полюс | Отправка запроса |
| Busy | Состояние сервера | Прием сообщения | Окончание обслуживания |
| Schedule STOP | Планирование окончания обслуживания | Окончание обслуживания | Прием сообщения |

Событие приема сообщения возникает в том случае, если хотя бы одно сообщение поступило на полюс устройства. Соответственно необходимо применить операцию ИЛИ ко всем отправленным сообщениям.

При таком описании несложно представить граф концептуальной модели (Рисунок 21), а также выполнить трансформацию в сеть Петри с применением описанных выше правил трансформации. Указанная трансформация приведена в Приложение D.

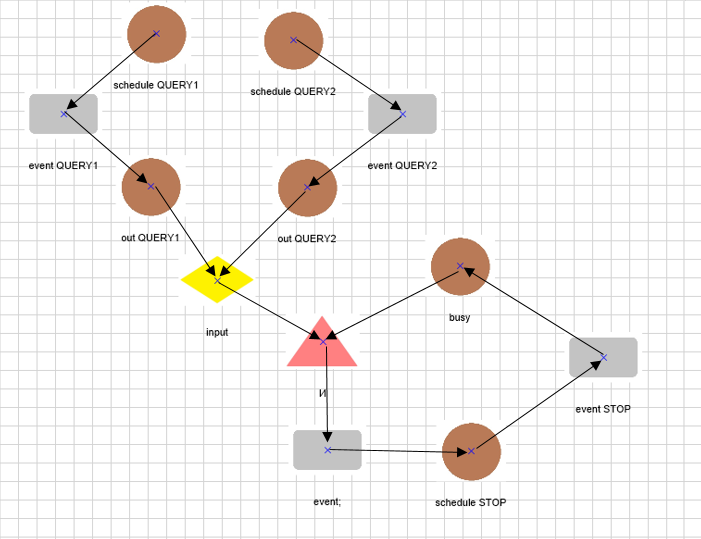


Рисунок 21. Концептуальная модель сети Петри, описанной по рутинам элементов сети

При использовании правил, описанных выше, и рассмотренных условиях и событиях в исследуемой системе, получаем модель, представленную на Рисунок 22. При проведении анализа полученной сети Петри можно установить, что в полученной сети не выполняются свойства безопасности, ограниченности, сохранение. Из невыполнимости свойства ограниченность можно установить, что если налагается ограничение на количество ресурсов системы, то система с такой нагрузкой может не справиться.

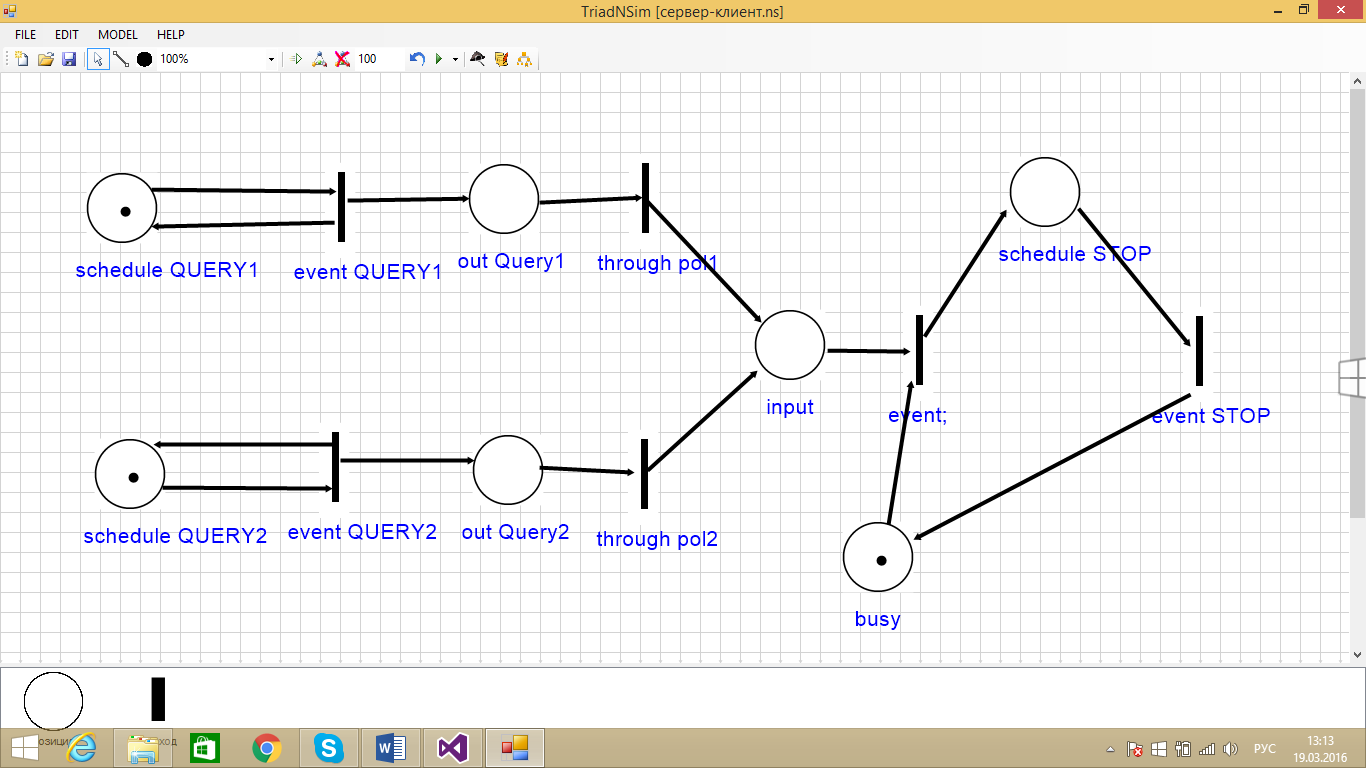


Рисунок 22. Полученная сеть Петри

Изменение поведения исследуемой системы можно изменить, если ввести, например, условие отправки запроса клиентом, и, если сообщение уже отправлено и сервер занят, то заявка вернется обратно. В таком случае выполняется условие безопасности и ограниченности, что позволяет преодолеть вышеуказанный недостаток (Рисунок 23).

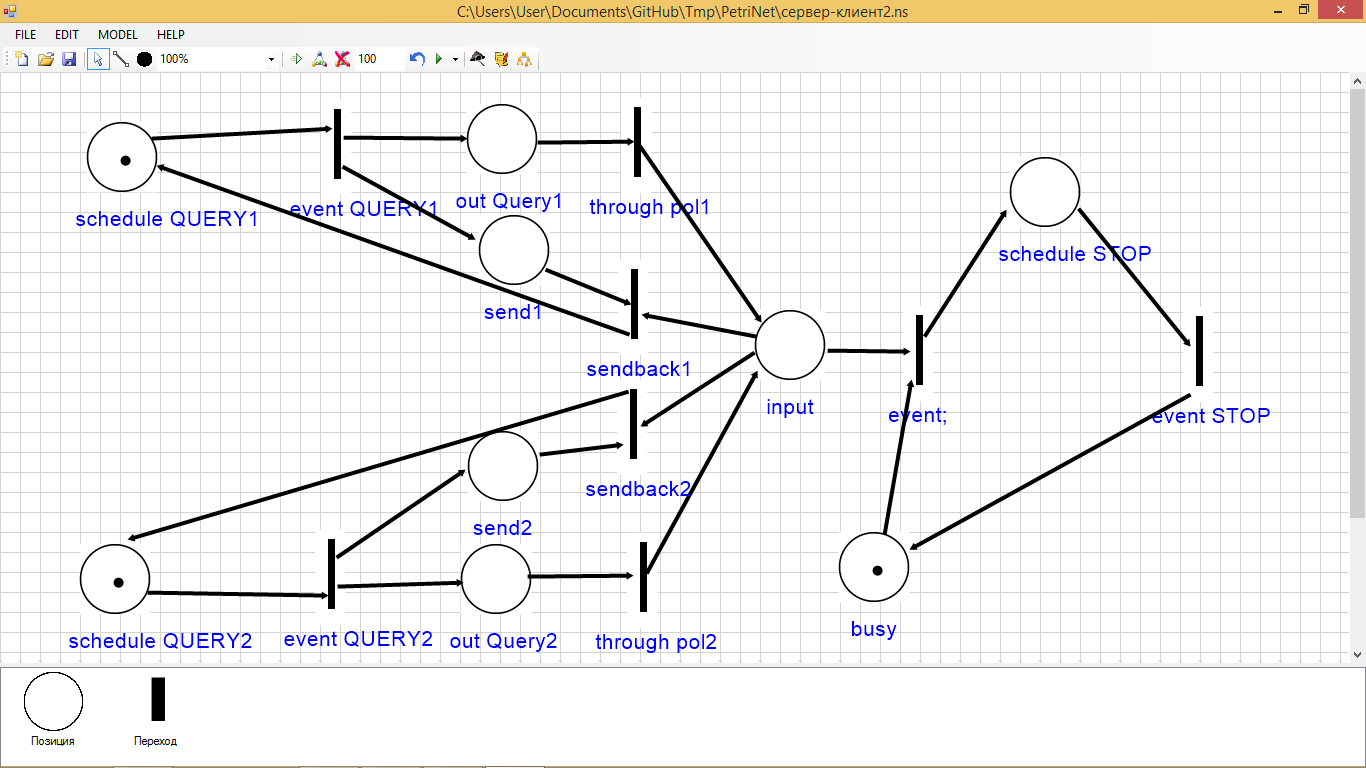


Рисунок 23. Сеть Петри после анализа

После перевода исследуемой модели в сеть Петри можно наложить любую рутину и, таким образом, использовать практически любой вид сетей Петри (цветные, стохастические, временные и т.д.), а также проводить имитационное моделирование, анализировать полученную модель и пр.

Также нужно учесть, что в блоке initial может быть вызов события. В этом случае данный вызов определяет начальную маркировку полученной сети Петри.

## Выводы

Данная глава была посвящена разработке правил для трансформации компьютерных сетей в сети Петри, с помощью которых пользователи системы имитационного моделирования TriadNS могут переводить одну модель в другую и применять новые средства анализа, а также выработки предложений по улучшению исследуемой системы.

Было установлено, что простое преобразование структуры модели компьютерной сети в сеть Петри не дает результатов, позволяющих эффективно использовать данную модель. Если же производить трансформацию на основе преобразования алгоритма поведения элементов модели, то такое представления является избыточным, и такую модель становится сложно анализировать.

Оптимальным вариантом трансформации модели компьютерной сети, и многих других моделей, в сеть Петри является представление системы в виде условий и событий. С помощью указанных трансформаций систему удобно анализировать и исследовать возникновение интересующих событий и ее нахождение в определенном состоянии.

Если взять в основу рассмотренные правила трансформации из модели событий и условий в сеть Петри, то для трансформации компьютерной сети в сеть Петри можно определить следующие правила (Таблица 7).

Таблица 7. Правила трансформации из модели компьютерной сети в сеть Петри

|  |  |  |
| --- | --- | --- |
| ***Имя правила*** | ***Левая часть*** | ***Правая часть*** |
| Клиент-Сервер |  |  |
| Клиент |  |  |
| Сервер |  |  |

В следующей главе будут представлены результаты разработки системы TriadNS, включающие графический редактор для работы с сетями Петри, средства анализа сетей Петри, а также модуля трансформации одной визуальной модели в другую на основе предложенных правил.

# Разработка и реализация системы

В предыдущих главах были рассмотрены теоретические аспекты, необходимые для исследования компьютерных сетей с помощью других моделей, а именно с помощью сетей Петри. Для возможности анализа компьютерных сетей с помощью сетей Петри была спроектирована и разработана подсистема на основе TriadNS, в которой можно строить сети Петри и моделировать их поведение.

## Реализация графического редактора для работы с сетями Петри

В результате работы был спроектирован и реализован модуль для работы с сетями Петри в системе TriadNS, в который входит графический редактор и средства анализа сетей Петри, основанный на методе построения дерева достижимости.

Диаграммы UML, построенные в ходе проектирования модуля приведены в приложении.

Для того чтобы система настраивалась на новую модель, необходимо построить онтологию предметной области, которая является для нее метамоделью и представляется в виде онтологии (Рисунок 24).

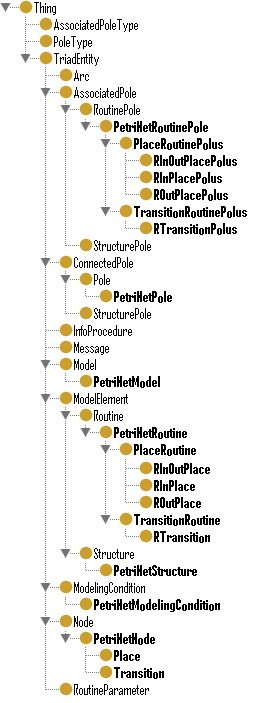


Рисунок 24. Иерархия классов онтологии сетей Петри

Перед тем как создавать систему построения сетей Петри с помощью графического редактора, уже существующего в системе, необходимо построить онтологию предметной области. Используя базовую онтологию в качестве метамодели и добавив соответствующие подклассы, была построена онтология сетей Петри с помощью инструментальных средств Protégé 4.0.2. Эта онтология встроена в онтологию TriadNS для возможности работы пользователя с системой, моделируя исследуемый несколькими моделями. Полная иерархия классов онтологии TriadNS приведена в Приложение D.

После того как онтология предметной области была построена в графический редактор системы TriadNS были добавлены соответствующие элементы, а в программу – классы. В системе были созданы инструменты для построения сетей Петри: пиктограммы вершин сети Петри, элементы управления для добавления дуг и меток. Также система поддерживает построение сетей Петри с кратными дугами и кратными метками.

1. Основные элементы системы

Элементы сети: вершины и позиции – загружаются из онтологии. Все подклассы класса PetriNetNode, который содержит два подкласса: «Позиция» и «Переход». Они отображаются в виде пиктограмм на панели, из которых пользователь может их перетащить на рабочую область в нужное место.

После перемещения элементов на рабочую область они имеют определенный семантический тип, но поведение вершин изначально не определено. Поведение этих элементов можно задать автоматически либо вручную (Рисунок 25). В качестве стандартного поведения вершин выступает следующие варианты: вершины с одной входной и/или одной выходной дугами (для позиций) и вершины с одной входной и одной выходной дугами (для перехода), которые можно выбрать в контекстном меню при выборе рутин. Туда попадают все экземпляры рутин данного семантического типа. Каждая из рутин имеет некоторое количество параметров, которые пользователь может изменять после наложения рутины на элемент. Если у некоторых элементов поведение не задано, то перед началом моделирования система предложит доопределить их поведение автоматически.

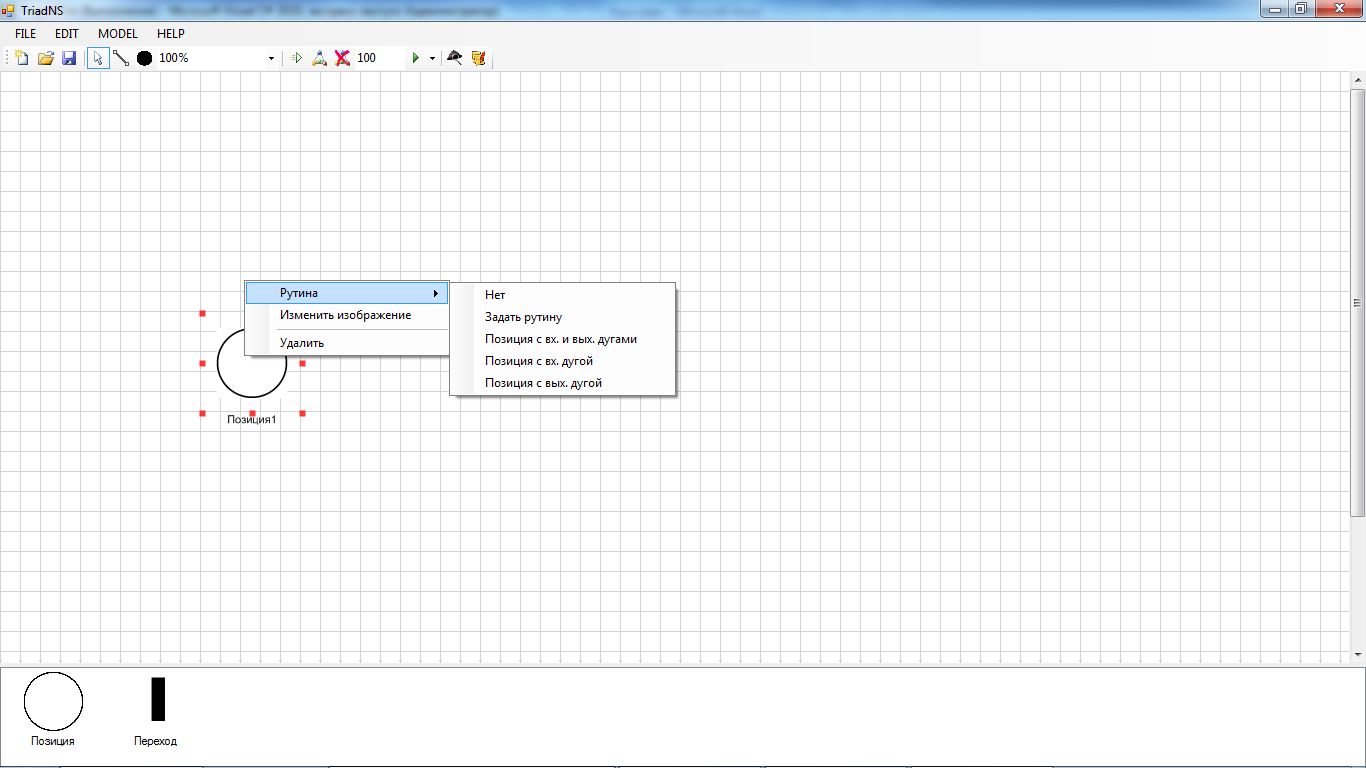


Рисунок 25. Определение поведения вершин

Далее необходимо для каждой дуги и вершины определить полюса, через которые вершины будут обмениваться сообщениями (Рисунок 26), также в диалоговом окне можно определить кратность дуги. Пользователь может задавать кратность дуг и простым перетаскиванием дуги.

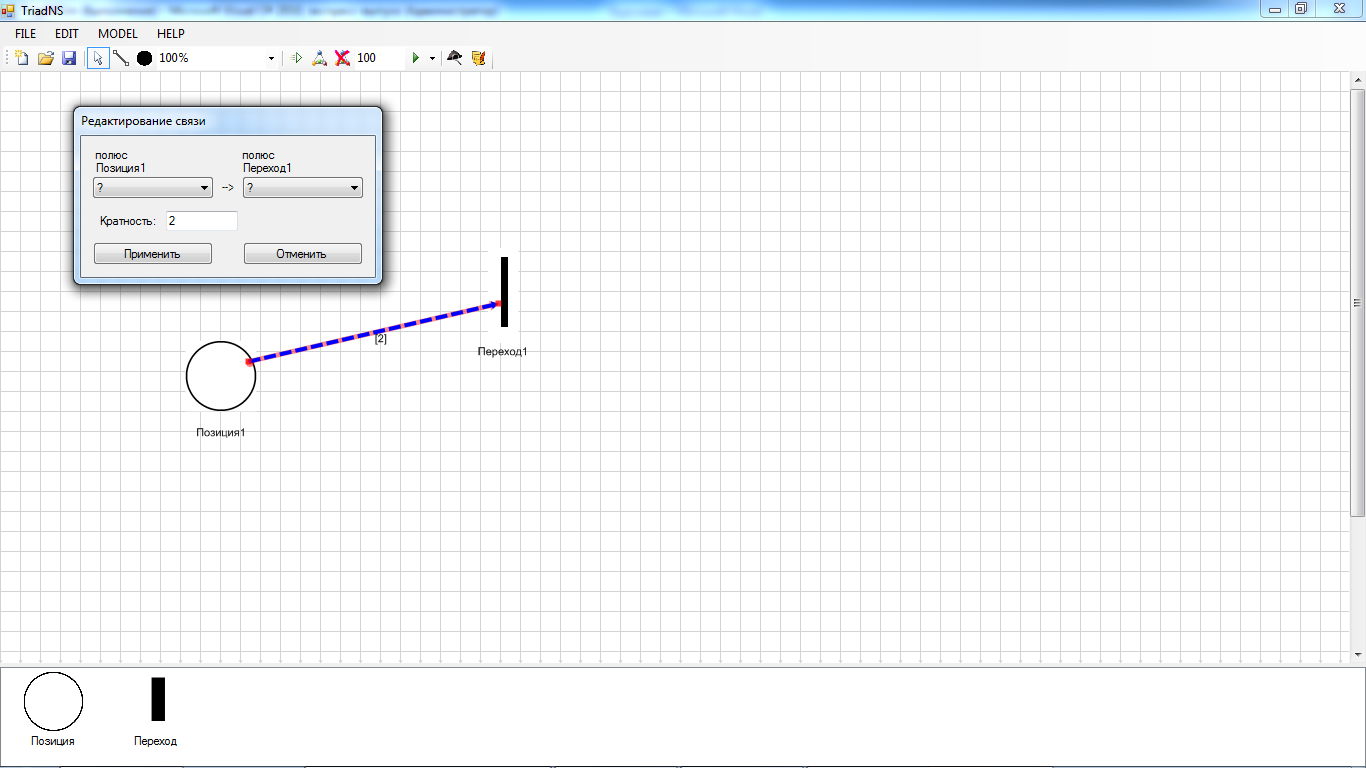


Рисунок 26. Редактирование связи

Для каждой вершины, в зависимости от количества соединенных с ней дуг, должны существовать разные рутины. Для удобства пользователя был разработан шаблон для задания рутины, который будет определять стандартное поведение вершины. Шаблон загружается в текстовый редактор для создания рутин в зависимости от того, сколько фишек имеет вершина и сколько дуг с ней соединено. Таким образом, пользователю не нужно с нуля описывать поведение вершины, а только принять рутину, описанную шаблоном. Шаблоны для вершин представлены в Приложение G.

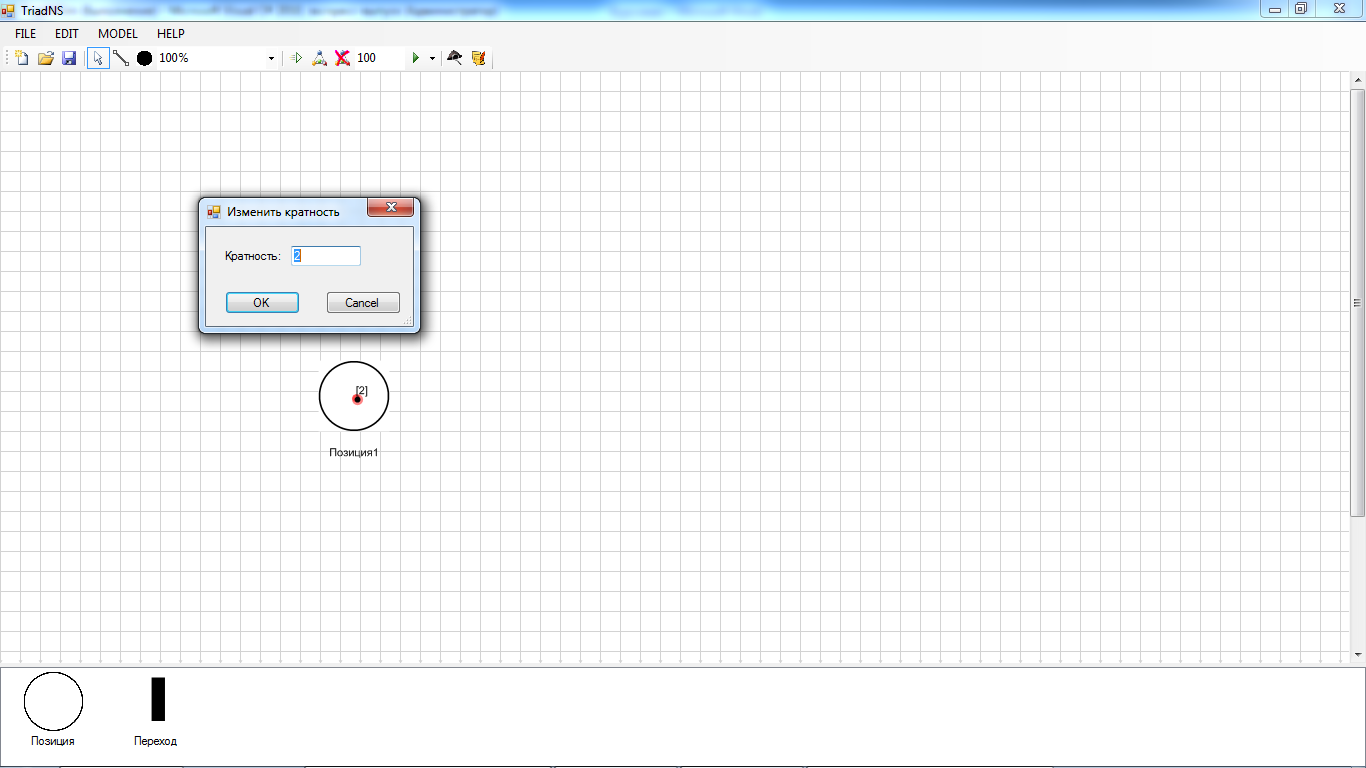


Рисунок 27. Изменение кратности маркеров

Для того чтобы пользователю не нужно было задавать количество маркеров для каждой вершины перед процессом имитации, их число автоматически синхронизируется с параметром InitialNumMarks, который есть в рутине каждой вершины. Таким образом, пользователь либо задает количество маркеров в диалоговом окне изменения значений параметров (Рисунок 27), либо изменяет количество маркеров – графических элементов.

В процессе имитации работы сети Петри происходит ее визуализация. Состояние сети Петри перерисовывается, если изменяется количество маркеров у какой-либо вершины. После имитации в отдельном окне выводится отчет о выполнении с печатью тех сообщений, которые были отправлены при выполнении рутин.

1. Пример построения сети Петри на основе задачи о писателе и читателе

Для демонстрации работы системы была спроектирована сеть Петри, показывающая работу задачи «Писателей и читателей» (Рисунок 28). При проектировании данной сети позициями являются статические объекты, используемые в задаче («Писатель», «Читатель», «Буфер»), и условия выполнения ими функций («Запись\_произведена», «Запись\_считана»). Переходами являются действия, выполняемые объектами: «Произвести\_запись», «Поместить\_в\_буфер», «Удалить\_из\_буфера», «Использовать\_запись». Поведение элементов же было автоматически определено.

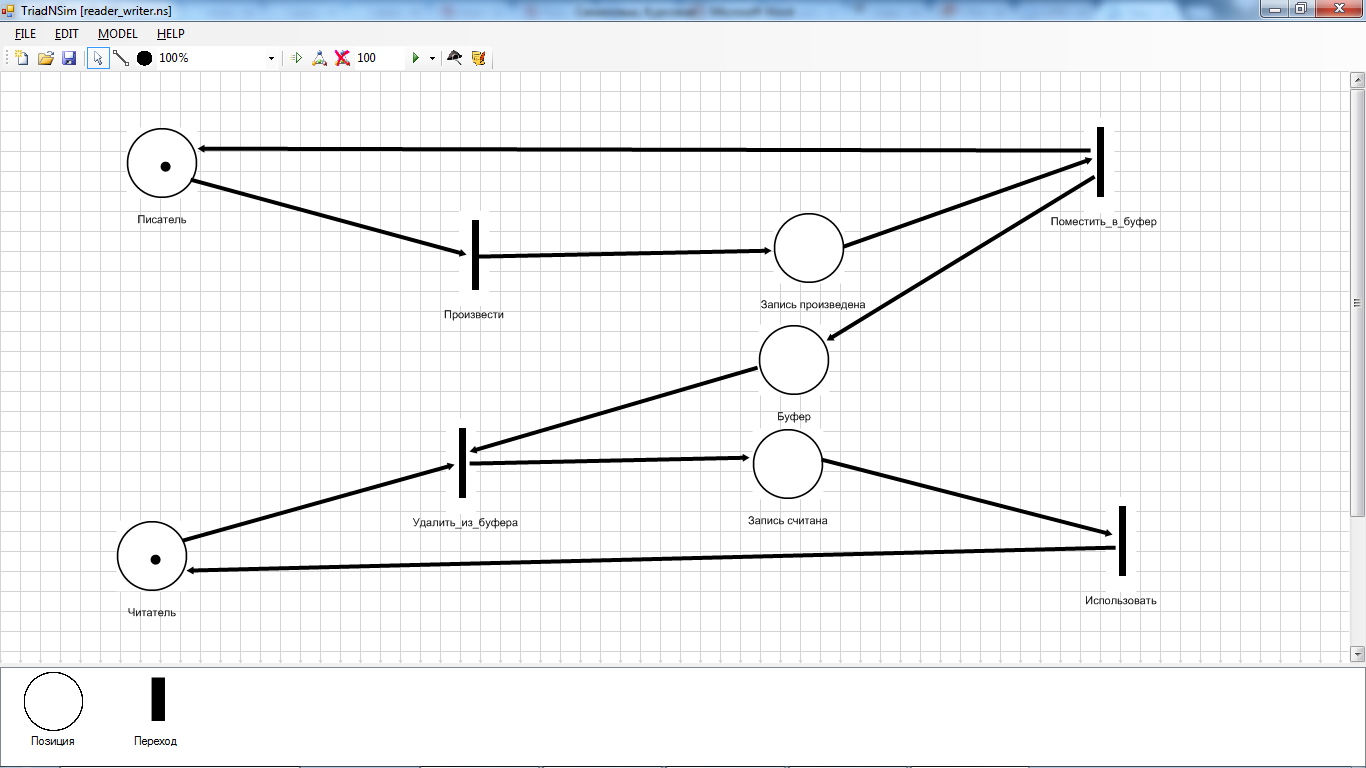


Рисунок 28. Сеть Петри задачи о писателе и читателе

## Реализация средств анализа сетей Петри

Как было описано ранее, сети Петри имеют особые средства анализа, которые сводятся к проверке выполнимости свойств модели. Было установлено, что наиболее наглядным и эффективным средством анализа сетей Петри является дерево достижимости (так как этот метод охватывает проверку практически всех свойств). Также была освящена проблема построения конечного дерева достижимости.

В системе TriadNS помимо графического редактора для работы с сетями Петри была реализована форма построения дерева достижимости по построенной сети Петри (Рисунок 29). С помощью данного сервиса осуществляется построение конечного дерева достижимости по следующему алгоритму.

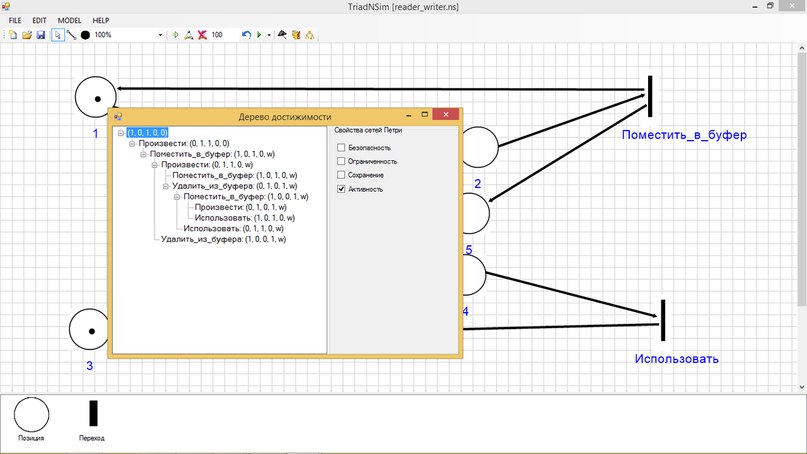


Рисунок 29. Дерево достижимости в TriadNS

Пусть *x* – граничная вершина, которую необходимо обработать. Граничной вершиной называется вершина, еще не обработанная алгоритмом

1. Если в дереве имеется другая вершина *y*, не являющаяся граничной, и с ней связана та же маркировка *μ[x] = μ[y]*, то вершина *x* – дублирующая.
2. Если для некоторой маркировки *μ[x]* ни один из переходов не может сработать, то x – терминальная вершина.
3. Для всякого другого перехода, разрешенной в маркировке вершины *x* *μ[x]*, создать новую вершину *z*. Маркировка добавленной вершины определяется для каждой позиции pi следующим образом:
   1. Если *μ[x]i=ω*, то *μ[z]i=ω*.
   2. Если на пути от корневой вершины к x существует вершина *y* c *μ[y]<δ(μ[x],tj)* и *μ[y]i<δ(μ[x],tj)*, то *μ[z]i=ω*.
   3. В противном случае *μ[z]i=δ(μ[x],tj)i*.

Дуга помеченная *tj* направлена от вершины *x* к вершине *z*. Вершина *x* переопределяется как внутренняя, вершина *z* становится граничной.

Алгоритм выполняется до тех пор, пока все вершины не будут терминальными, дублирующими или внутренними, т.е. пока не кончатся граничные вершины.

Благодаря дереву достижимости в системе можно проверять выполнение свойств активности, безопасности, ограниченности, сохранения, а также аналитически проследить возможные достижимые и интересующие состояния системы. С помощью дерева достижимости можно анализировать поведение исследуемой системы и вносить изменения в ее поведение, если это требуется.

## Реализация трансформатора моделей

Для того, чтобы выполнить трансформацию моделей, необходимо выбрать исходную и целевую модели, а также задать правила трансформации (Рисунок 30). Правила задаются в графическом виде, то есть необходимо создать левую и правую части правил, дать имя правилу.

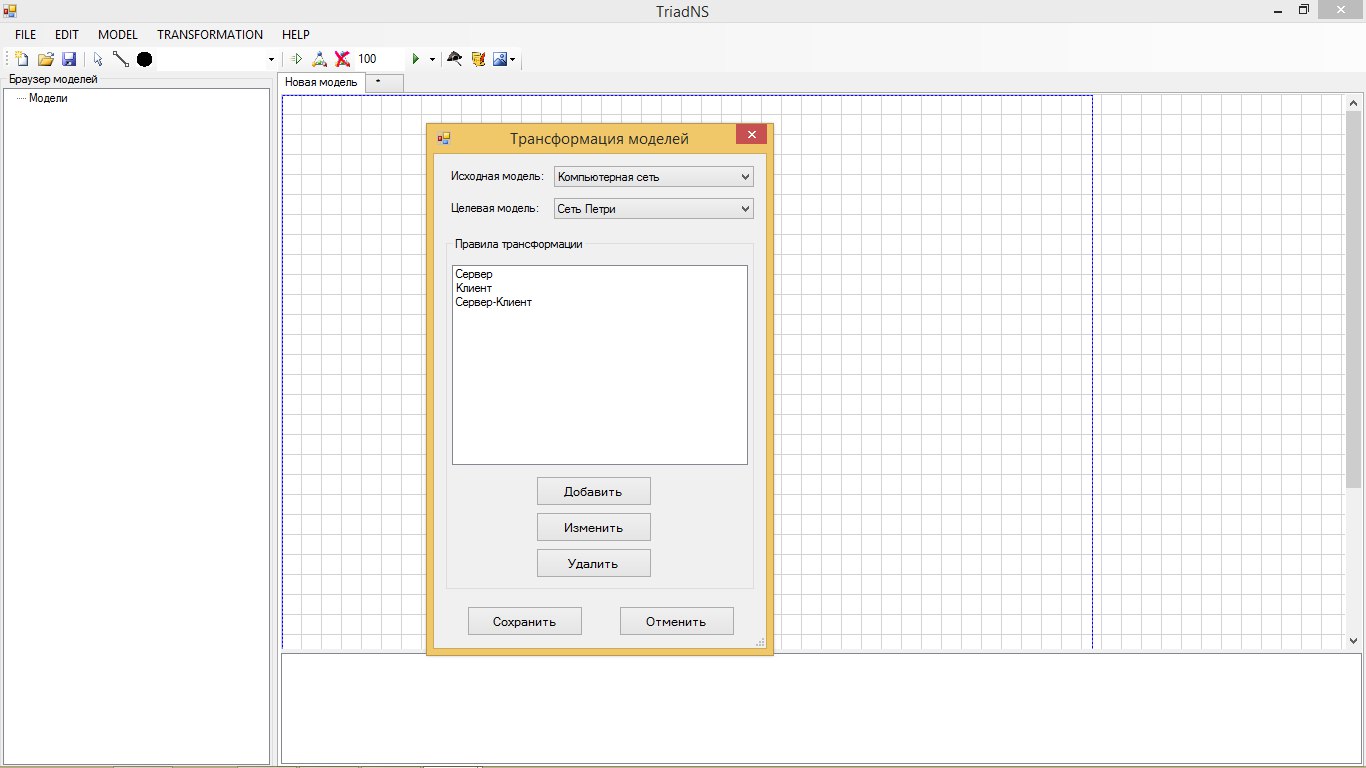


Рисунок 30. Форма для определения трансформации моделей

Также на пользователь сам ответственен за расположение правил трансформации в верном порядке. После построения целевой модели выполняется ее проверка, определяется, все ли нетерминальные символы были выведены, все ли вершины и связи полученной модели соответствуют построены по ее метамодели.

Чтобы задать правила трансформации, пользователь перетаскивает нужные ему элементы на рабочую область и протягивает необходимые связи, так же, как и при работе с графическим редактором (Рисунок 31).

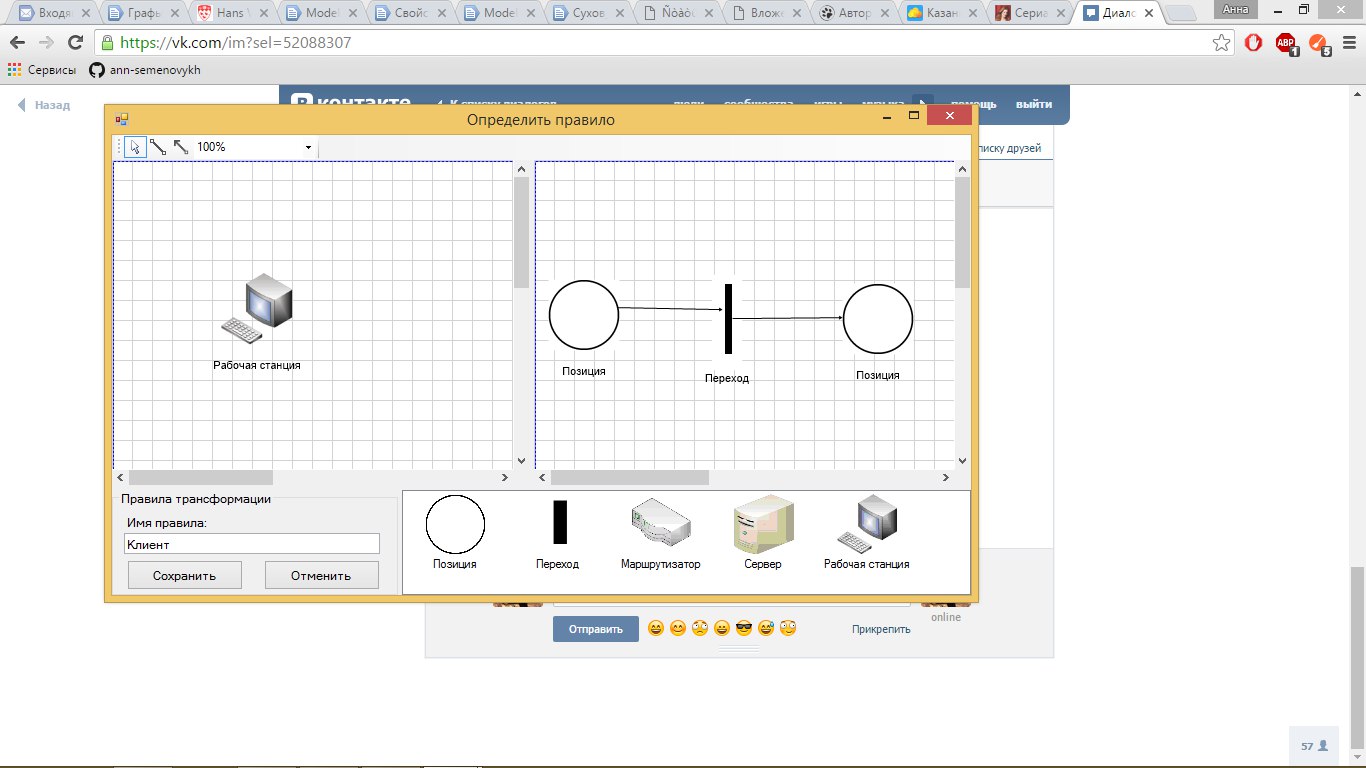


Рисунок 31. Форма определения правил трансформации

После того, как выбраны исходная и целевая модели, а также заданы все правила трансформации и расположены в необходимом порядке, заданную графовую трансформацию необходимо сохранить в файл. Это позволяет использовать одну и ту же трансформацию для различных моделей, а также задавать разные трансформации для одной и той же модели.

Для проведения трансформации пользователю необходимо выбрать модель или фрагмент модели, который необходимо перевести в другую модель (Рисунок 32), а также выбрать правила, заданные заранее. Далее система пытается выполнить правила трансформации, пока это возможно. Как только вывод прекратился, система выводит полученную модель для проверки пользователю, а также выводит полученные ошибки, например, если не были выведены все нетерминальные символы или получены вершины или связи, недоступные для целевой модели.

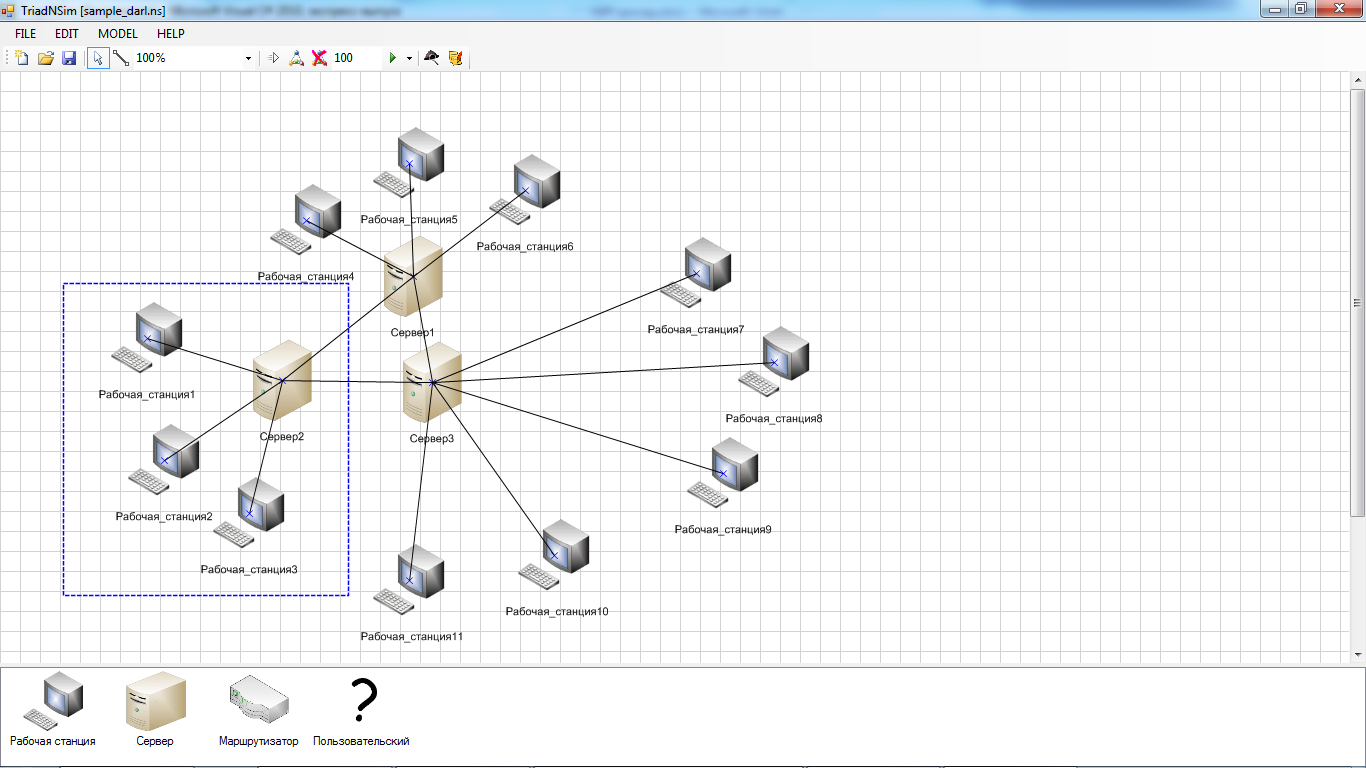


Рисунок 32. Выбор фрагмента графа для трансформации

## Дополнительные доработки системы

Также был доработан интерфейс системы. Для работы с несколькими моделями и их быстрой навигации был реализован браузер моделей. В правой части экрана располагается окно графического редактора моделей с отдельными вкладками для каждой модели. В левой части – дерево, отображающее модели, с которыми работает пользователь в текущее время, а также их сущности.

При добавлении новой вкладки из онтологии TriadNS загружаются доступные модели, которые могут быть построены в системе.

Также была реализована возможность добавления так называемых динамических объектов в модель – это такие объекты, от расположения которых зависит состояние моделируемой системы, такие объекты перемещаются в процессе моделирования (например, метки, или маркеры, в сетях Петри, заявки в СМО). Была реализована возможность проведения визуализации модели в процессе моделирования. Именно изменение расположения таких динамических объектов отображается при моделировании.

Для того чтобы пользователь мог вручную моделировать поведение исследуемой системы, была реализована возможность ручного управления процессом имитации. С помощью такой функциональности, пользователь может запускать переходы вручную, для исследования определенных событий, происходящих в системе.

## Выводы

В результате работы был спроектирован и реализован модуль для работы с сетями Петри в системе имитационного моделирования TriadNS, а также реализован модуль, с помощью которого можно задать правила трансформации из одной модели в другую и выполнить перевод на основе графовых грамматик.

Благодаря добавлению такой функциональности в систему в ней стало возможным применить многомодельный подход при исследовании компьютерных сетей и других сложных систем. Это позволяет анализировать объект с различных сторон и применять различные методы анализа, что ведет к построению наиболее точных и адекватных имитационных моделей, помогает избежать ошибок при построении сложных систем.

# Заключение

В результате выполненной работы были спроектированы и разработаны инструментальные средства работы с сетями Петри в системе имитационного моделирования TriadNS, а именно: графический редактор для построения сетей Петри, средства анализа сетей Петри в виде дерева достижимости. Также был реализован модуль для трансформации графических моделей на основе правил, заданных пользователем.

Таким образом, была достигнута основная цель работы: проектирование и разработка инструментальных средств имитационного моделирования, позволяющих преобразовывать (выполнять трансформации) графические модели компьютерной сети в сеть Петри.

Также были выполнены задачи, необходимые для достижения цели работы:

* рассмотрены теоретические аспекты модельно-ориентированного подхода и исследована трехуровневая структура в имитационном моделировании (метамодель, модель, данные) на примере работы с системой имитационного моделирования TriadNS. Таким образом, метамоделью в TriadNS является онтология предметной области, моделью является некоторый математический аппарат либо визуальное представление онтологии (модель компьютерной сети или сеть Петри), а данными являются конкретные построенные модели;
* был сделан обзор существующих симуляторов сетей Петри, активно развивающихся в настоящее время, также проанализированы их достоинства и недостатки (Таблица 1);
* разработан визуальный язык сетей Петри для работы с моделью в TriadNS;
* разработан графический редактор для построения и моделирования сетей Петри, предоставляющий удобные средства для работы с моделью, а также средства анализа сетей Петри на основе построения дерева достижимости;
* разработаны правила трансформации модели компьютерной сети в сеть Петри, построенные на основе концепции событий и условий, возникающих в системе при определении стандартных рутин для элементов сети;
* разработаны инструментальные средства для реализации трансформации моделей в системе TriadNS на основе графовых грамматик.

Таким образом, в системе реализована возможность проведения имитационного моделирования с помощью нескольких моделей. Это позволяет исследовать объект с различных сторон, привлекать специалистов различных предметных областей, строить наиболее точные и приближенные к реальности имитационные модели.

# Библиографический список

1. Batarseh O., McGinnis Leon F. System Modeling in SysML and System Analysis in Arena // Proceedings of the 2012 Winter Simulation Conference, 2011, pp. 2924-2935
2. Küçükkeçeci Çetinkaya D. Model Driven Development of Simulation Models: Defining and Transforming Conceptual Models into Simulation Models by Using Metamodels and Model Transformations. – TU Delft, Delft University of Technology, 2013
3. Cetinkaya D., Verbraeck A. Metamodeling and model transformations in modeling and simulation // Proceedings of the 2011 Winter Simulation Conference, 2011, p. 3048-3058
4. Czarnecki K., Helsen S. Classification of Model Transformation Approaches // OOPSLA’03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, 2003, pp. 1-17
5. De Lara J., Vangheluwe H. Computer aided multi-paradigm modelling to process petri-nets and statecharts // International Conference on Graph Transformations (ICGT), Lecture Notes in Computer Science, 2002, vol. 2505, pp. 239-253
6. De Lara J., Vangheluwe H., J. Mosterman P. Modelling and analysis of traffic networks based on graph transformation // Formal Methods for Automation and Safety in Railway and Automotive Systems, page 11. IEEE Computer Society Press, 2004
7. G. de Oliveira Lino F. Analisador e Simulador de Redes de Petri. - Rio de Janeiro: 2007. – 70 p.
8. G. de Oliveira Lino F., Sztajnberg A. Analisador e Simulador de Redes de Petri // Anais do XXVII Congresso da SBC, 2007, pp. 898-903
9. GPC mbH - Poses++ [Сайт] URL: <http://www.gpc.de/e_poses.html> (дата обращения 14.06.2016)
10. Home – Horus software GmbH [Сайт] URL: <http://www.horus.biz/ru/> (дата обращения 14.06.2016)
11. Jie T. W., Ariff bin Ameedeen M. A Survey of Petri Net Tools // ARPN Journal of Engineering and Applied Sciences, No. 8, Vol. 9, 2014, pp. 1209-1214
12. Kharitonov D., Golenkov E., Tarasov G. A Method of Sample Models of Program Construction in Terms of Petri Nets / Kharitonov D., Golenkov E., Tarasov G., Leontyev D. // Modeling and Analysis of Information Systems, 2015, Vol.22, No.4, pp.563–577
13. Moore K. E., Chiang J. C. Alpha/Sim Simulation Software Tutorial // Proceedings of the 2000 Winter Simulation Conference, 2000, pp. 259-267
14. Murata T. Petri Nets: Properties, Analysis and Applications // Proceedings of the IEEE, Vol. 77, No. 4, 1989, p. 541-580
15. NetSim Cisco Network Simulator & Router Simulator. [Сайт] URL: <http://www.boson.com/netsim-cisco-network-simulator> (дата обращения 14.06.2016)
16. OMNeT++ Community Site. [Сайт] URL: http://www.omnetpp.org (дата обращения 14.06.2016)
17. Renew – User Guide / Olaf Kummer, Frank Wienberg, Michael Duvigneau, Lawrence Cabac, 2015
18. Spiteri Staines A. A Triple Graph Grammar Mapping of UML 2 Activities into Petri Nets // International Journal of Computers Issue1, 2010, Vol.4, p. 27-35
19. Sukhov A. O., Lyadova L. N. Horizontal Transformations of Visual Models in MetaLanguage System, in: Proceedings of the 7th Spring/Summer Young Researchers’ Colloquium on Software Engineering, SYRCoSE 2013 / Отв. ред.: A. Kamkin.; Ed. by A. Petrenko, A. Terekhov. Kazan : -, 2013. P. 31-40
20. Sukhov A. O., Lyadova L. N. MetaLanguage: a Tool for Creating Visual Domain-Specific Modeling Languages. pp. 42-53. DOI: 10.15514/SYRCOSE-2012-6-5
21. Tolvanen J., Rossi M. MetaEdit+: defining and using domainspecific modeling languages and code generators. Proceeding OOPSLA '03 Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, 2003, pp. 92-93
22. Vangheluwe H., De Lara J. Automatic generation of model-to-model transformations from rule-based specifications of operational semantics // Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling., Computer Science and Information Systems Reports TR-38, 2007, pp.139 - 148
23. Vangheluwe H., De Lara J., J. Mosterman P. An introduction to multi-paradigm modelling and simulation // Proceedings of the AIS'2002 Conference, 2002, pp.9 – 20
24. Yasper Process Modeling Software [Сайт] URL: <http://www.yasper.org/> (дата обращения 14.06.2016)
25. Zamyatina E. B., Mikov A. I., Mikheev R. A. Linguistic and Program Tools For Debugging and Testing Of Simulation Models Of Computer Networks // International Journal "Information Models and Analyses". 2013. Vol. 2. No. 1. P. 70-80
26. Zamyatina E. B., Mikov A. I., Mikheev R. A. Оntological Approach to a Construction of the Simulation System for the Specific Domain // Information Models and Analyses. 2015. Vol. 4. No. 1. P. 41-53
27. Гудов А. М., Семехина М. В. Имитационное моделирование процессов передачи трафика в вычислительных сетях // Управление большими системами, вып.31. – с. 130-161
28. Дмитриев В.Н., Тушнов А.С., Сергеева Е.В. Имитационное моделирование системы мониторинга многозвенной сети передачи данных // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. 2013. №2 С.86-91
29. Дорофеев Р. С. Модели структурного описания для оценки их качества: Автореф. дис. на соискание ученой степени канд. тех. наук: 05.13.01 – Иркутск, 2014. – 16 c.
30. Доррер М. Г. Алгоритм преобразования моделей бизнес-процессов в безопасные сети Петри на основе матричных формул [Электронный ресурс] // Электронный научный журнал «ИССЛЕДОВАНО В РОССИИ» URL: <http://zhurnal.ape.relarn.ru/articles/2008/104.pdf> (дата обращения 14.06.2016)
31. Заболотная А. Метод трансляций SDL-спецификаций с помощью модифицированных сетей Петри высокого уровня // International Journal "Information Models and Analyses". 2012. Vol. 1. No. 1. P. 390-397
32. Зайцев Д.А., Шмелёва Т.Р. Моделирование коммутируемой локальной сети раскрашенными сетями Петри // Зв'язок, № 2(46), 2004, с. 56-60
33. Замятина Е. Б., Михеев Р. А. Опыт разработки графического пользовательского интерфейса, настраиваемого на предметную область // Open Semantic Technologies for Intelligent Systems, 2012. – 9 c.
34. Замятина Е. Б., Сухов А. О., Ходырева В. А. Опыт применения многомодельного подхода к моделированию бизнес-процессов // Технологии разработки информационных систем: сборник статей международной научно-практической конференции: Изд-во ЮФУ, Таганрог, 2015
35. Коба В., Кочетков Н. Быстродействующий эмулятор сетей Петри QPNet // III Общероссийская студенческая электронная научная конференция  
    "СТУДЕНЧЕСКИЙ НАУЧНЫЙ ФОРУМ 2011", 2011
36. Крывый С.Л., Чугаенко А.В. Формальные методы анализа дискретных систем с использованием языка спецификаций // Кибернетика и системный анализ. — 2009. — № 4. — С. 31-48.
37. Ломазова И. А. Вложенные сети Петри: моделирование и анализ распределенных систем с объектной структурой. М. : Научный мир, 2004, - 208с
38. Лядова Л.Н. Метамоделирование как основа средств разработки профессионально-ориентированных информационных систем / Математика программных систем: межвуз. сб. науч. ст. Пермь: Изд-во Перм. гос. нац. исслед. ун-та, 2012. – Вып. 9. – С. 20-32.
39. Миков А. И. Автоматизация синтеза микропроцессорных управляющих систем. – Иркутск: Изд-во Иркут. ун-та, 1987. – 288 с.
40. Миков А.И. Графы и грамматики / А.И. Миков. – Краснодар: Кубанский государственный университет, 2014. – 161 с.
41. Палагин А. В., Алишов Н. И., Полиновский В. В., Иваськив Ю. Л. Метод синтеза услуг в задачах компьютерной телефонии // ММС. 2004. №3 С.89-101
42. Питерсон Дж. Теория сетей Петри и моделирование систем: Пер. с англ. – М.: Мир, 1984. – 264 с.
43. Подкорытов Д.И. Агентно-ориентированная среда моделирования сетевых систем AGNES // Ползуновский вестник, 2012. № 2/1, с. 94-99
44. Стеценко И. В. Теоретические основы Петри-объектного моделирования систем // ММС. 2011. №4 С.136-148
45. Фаулер М. Предметно-ориентированные языки программирования.: пер. с англ. – М.: ООО «И. Д. Вильямс», 2011. – 576с.

# Приложения

## Требования, предъявляемые к разрабатываемой системе.

1. Наименование системы. Анализ и моделирование компьютерных сетей с использованием нотаций сетей Петри.
2. Цели и назначение системы. Система предназначена для реализации многомодельного подхода к имитационному моделированию компьютерных сетей. Для компьютерных сетей (далее КС) можно построить несколько математических моделей: сеть Петри, системы массового обслуживания и т.д. Система TriadNS позволяет описать КС в нотации сетей Петри и выполнить моделирование, используя теорию сетей Петри. Кроме того, систему можно использовать в учебных целях для демонстрации сетей Петри и их использования при изучении тупиковых ситуаций.

Моделирование системы с использованием математической теории сетей Петри позволит: повысить эффективность анализа компьютерных сетей, выявить «опасные» участки компьютерных сетей, в том числе тупиковые состояния.

1. Требования к системе в целом. Система должна представлять собой расширение системы проектирования компьютерных сетей TriadNS, построенной на базе системы имитационного моделирования Triad.Net .
2. Источником данных для системы должны быть. система проектирования и моделирования компьютерных сетей TriadNS, онтологии.
3. Функциональные требования. Система должна обеспечивать возможность выполнения перечисленных ниже функций:
4. Построение сетей Петри;
5. Имитационное моделирование сети Петри;
6. Редактирование сети Петри;
7. Визуализация модели;
8. Визуализация процесса моделирования;
9. Ручной запуск переходов в процессе моделирования;
10. Трансформация модели компьютерной сети в сеть Петри;
11. Автоматическая трансформация модели компьютерной сети в сеть Петри на основе заданных правил.
12. Требования к надежности системы. Уровень надежности достигается за счет программно-аппаратных средств, организации бесперебойного питания технических средств. При возникновении ошибок в работе системы должны выводиться сообщения о некорректной работе и причине ошибки.
13. Требования к внешнему оформлению системы. Должна быть выполнена реализация графического редактора для построения сети Петри с возможностью изменения и настройки графических элементов редактора, возможностью изменения масштаба построенной модели.

Требования к интерфейсу пользователя. интерфейс должен обеспечивать удобную работу конечного пользователя, знающего предметную область.

1. Требования к программному обеспечению. Система должна разрабатываться и эксплуатироваться на основе уже существующей системы проектирования компьютерных сетей TriadNS. Разработка проекта ведется с использованием языка программирования высокого уровня C#, в среде Visual Studio 2010, построение пользовательских дополнений к системе может быть выполнено на языке Triad. Имитационное моделирование полученной модели проводится с помощью системы имитационного моделирования Triad.Net. Основным языком взаимодействия пользователя и системы является русский или английский язык: взаимодействие пользователя с программой осуществляется на русском языке, все отчеты выводятся на русском языке, графический интерфейс с пользователем должен быть на русском или английском языке.
2. Дополнительные требования. Требования к масштабируемости: подсистема может расширяться в дальнейшем с помощью написания дополнительных программ/подпрограмм. Требования к переносимости: система должна переноситься на другие как продукт Microsoft. Требования к открытости: подсистема является программным продуктом с закрытым доступом к коду.

## Рутины сущностей модели компьютерной сети

Рутина сервера

routine SERVER [integer max; real T](input receive; output send)

initial

boolean busy;

integer length;

busy=false;

length=0;

endi

event;

if busy then

if length<max then

length=length+1;

endif

else

schedule STOP in T;

busy=true;

endif

ende

event STOP;

if length>0 then

length=length-1;

schedule STOP in T;

else

busy=false;

endif

ende

endrout

Рутина Клиента

routine Client [real T] (Input pol)

initial

schedule QUERY in 0;

endi

event QUERY;

out “Query” through pol;

schedule QUERY in T;

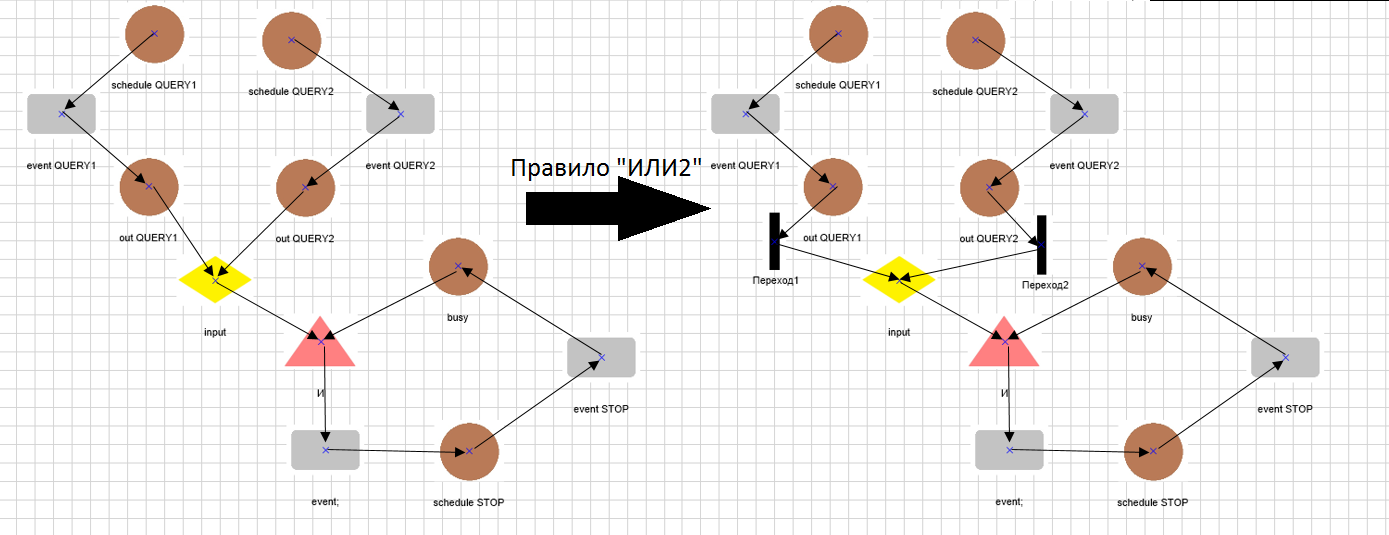
ende

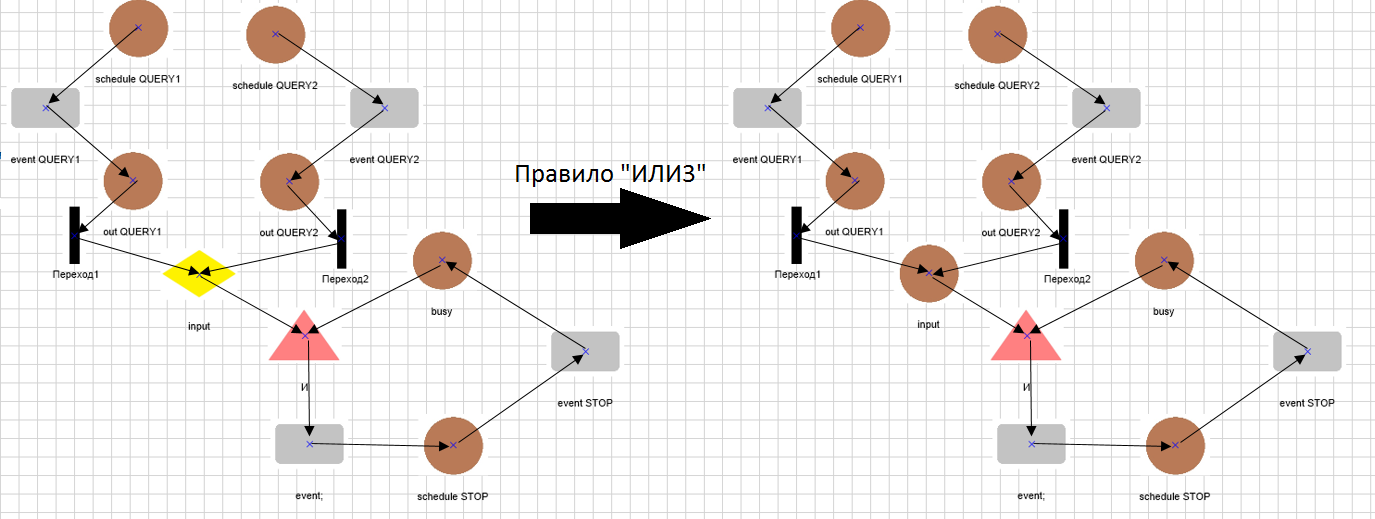
endrout

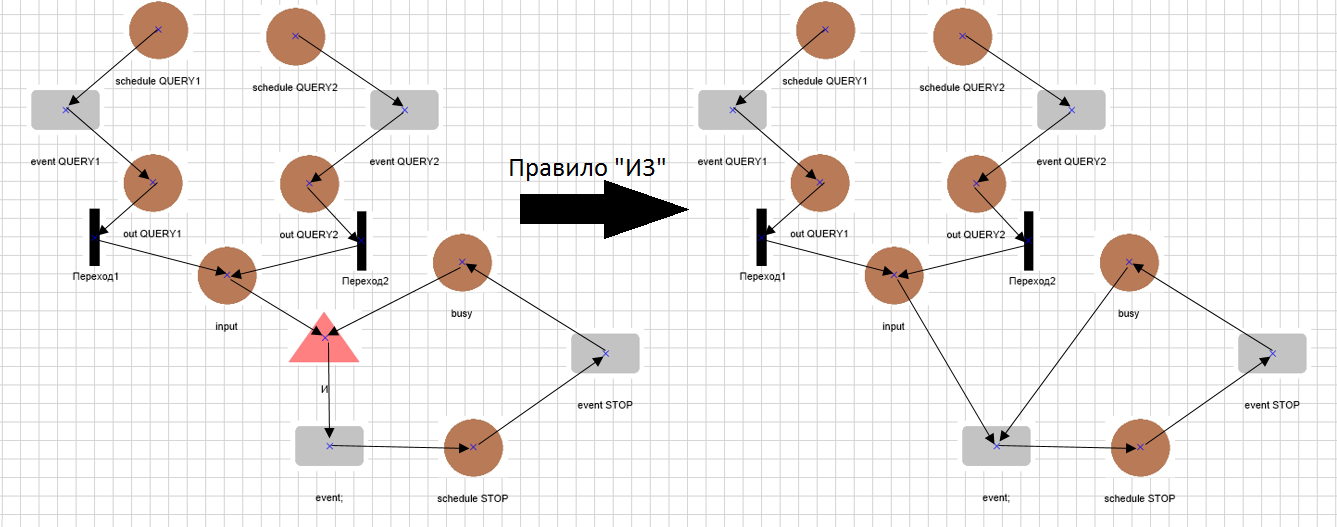
## Блок-схемы рутин

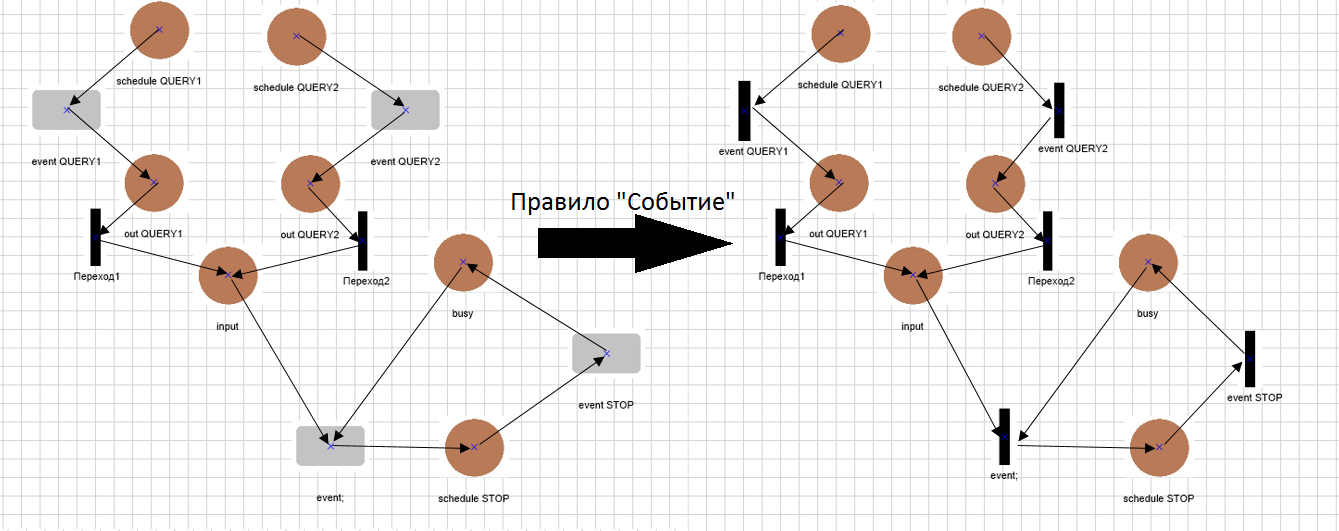


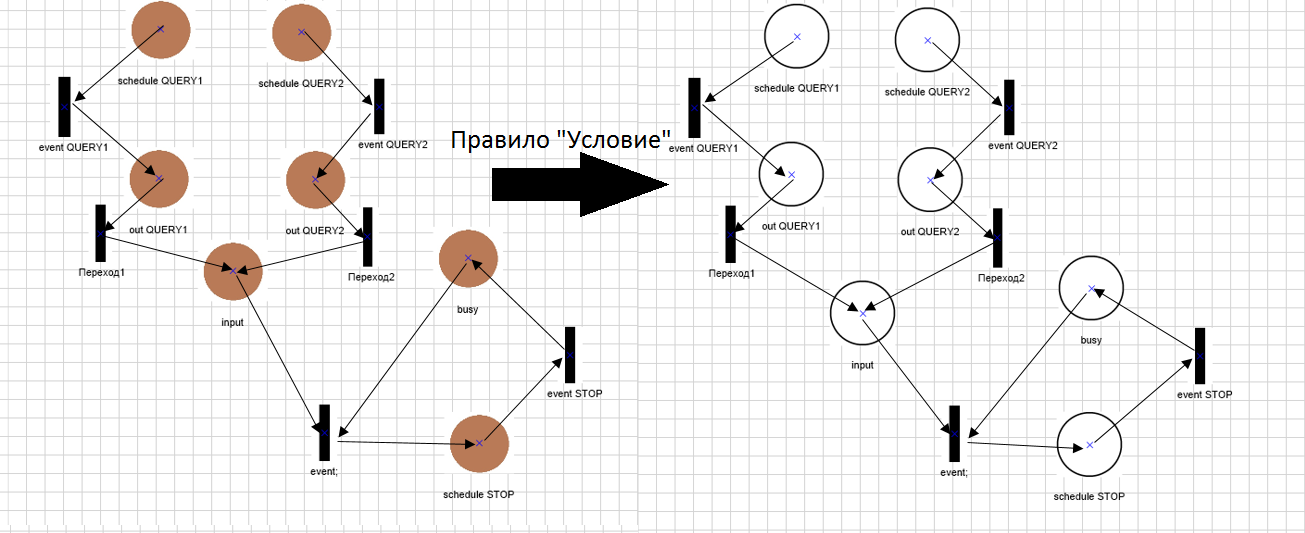
# Применение правил трансформации для преобразования событий и условий системы в сеть Петри



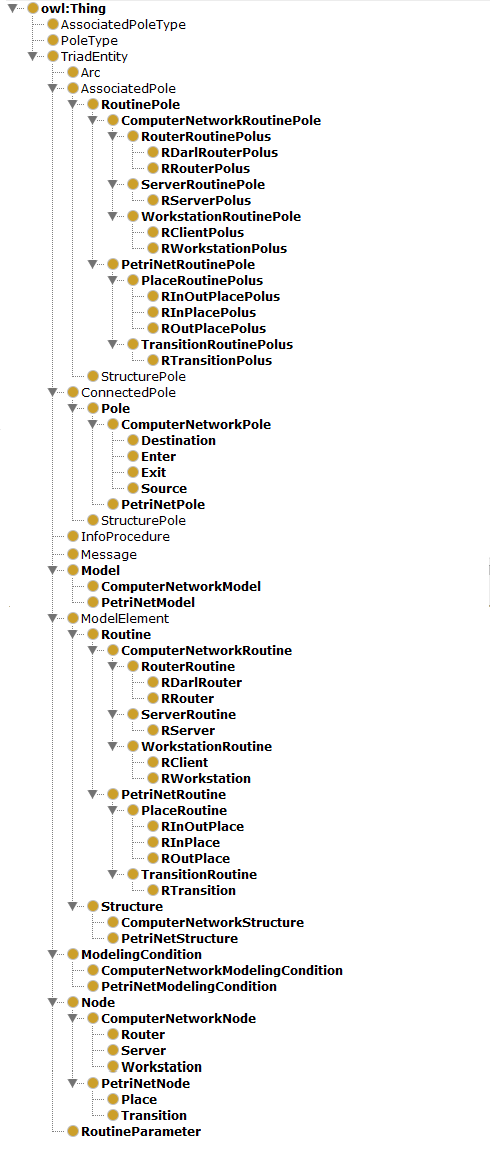








# Иерархия классов онтологии TriadNS



## **Онтология предметной области «Система проектирования и моделирования компьютерных сетей TriadNS»**

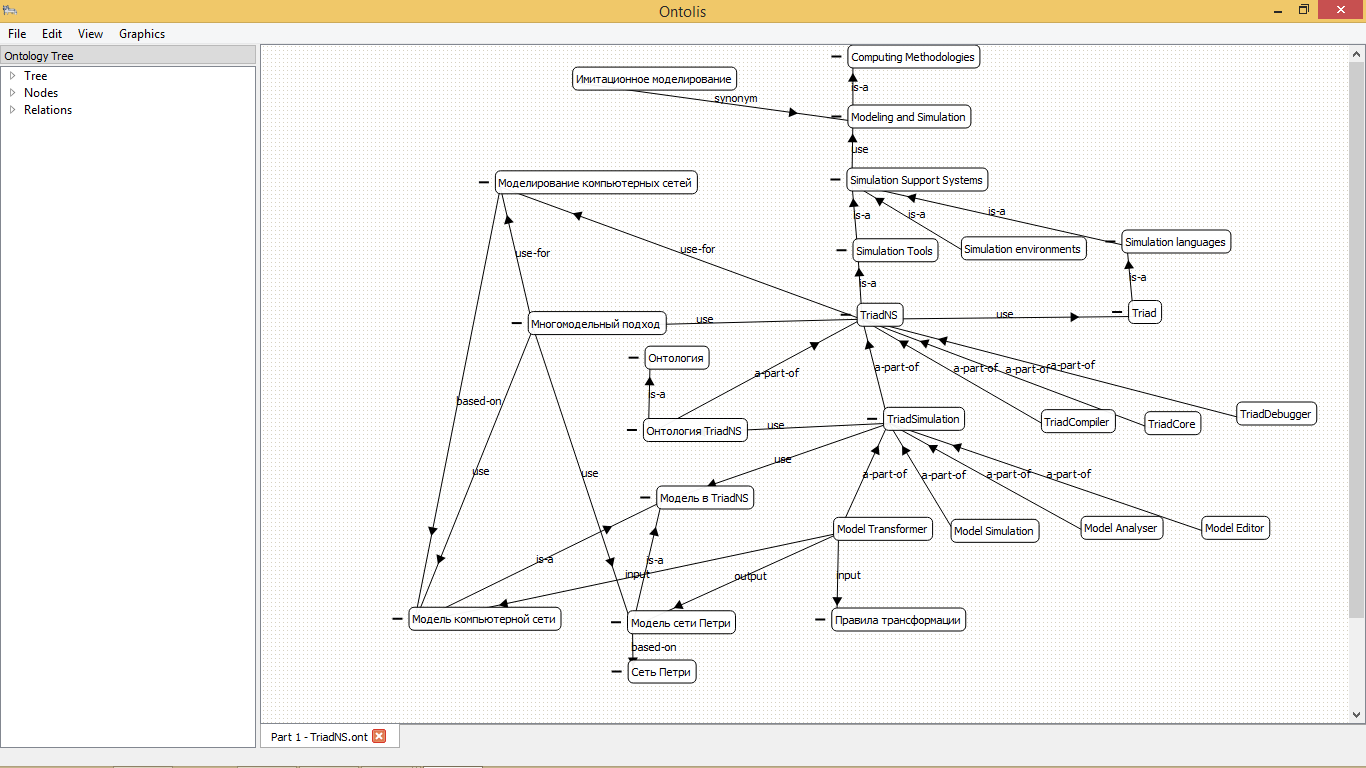
Цель: Описание используемых методов в системе имитационного моделирования TriadNS, модулей системы

Идентификация онтологии

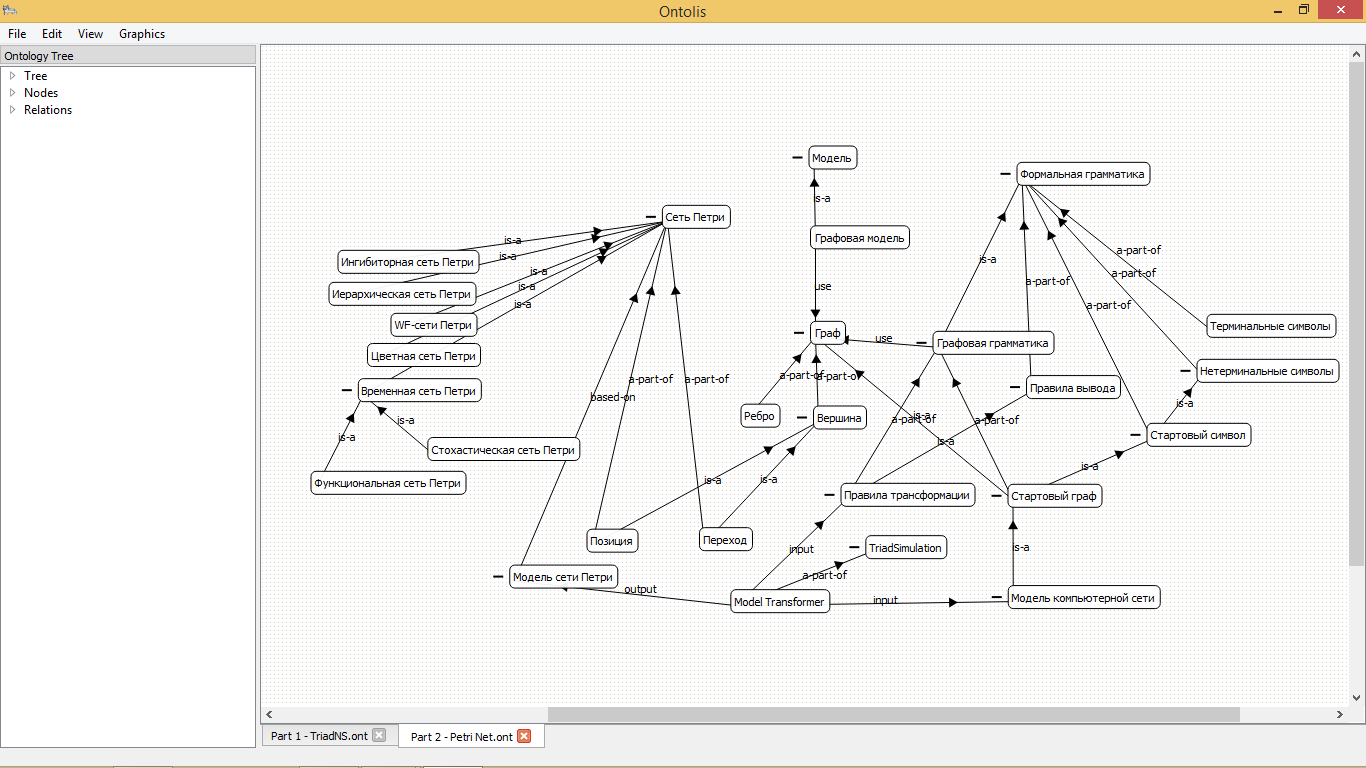
1. Computing Methodologies – компьютерный анализ и обработка проблем в той или иной области
2. Modeling and Simulation – процесс создания модели исследуемого объекта и процесс анализа поведения объекта путем подмены самого объекта его моделью
3. Simulation Environment – комплекс программных средств, предназначенный для имитационного моделирования
4. Simulation Language – язык программирования, предназначенный для имитационного моделирования
5. Simulation Support System – системы, способные производить исследование объектов с помощью имитационного моделирования
6. Simulation Tools – программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ имитационного моделирования
7. Model Analyzer – анализатор модели, входящий в состав модуля TriadSimulation системы TriadNS
8. Model Editor – редактор модели модуля TriadSimulation, позволяющий строить графическую модель исследуемого объекта на основе его метамодели, описанной в онтологии системы
9. Model Simulator – симулятор модели в модуле TriadSimulation, позволяющий моделировать объект и производить визуализацию в процессе симуляции
10. Model Transformer – редактор трансформации модели в модуле TriadSimulation, позволяющий производить перевод одной модели в другую на основе правил графовой грамматики
11. Triad – язык имитационного моделирования, используемый в системе TriadNS
12. TriadCompiler – модуль системы TriadNS, производящий компиляцию программы на Triad
13. TriadCore – модуль системы TriadNS, позволяющий работать с основными структурами языка Triad
14. TriadDebugger – модуль системы TriadNS, производящий отладку программы на Triad
15. TriadNS – система, предназначенная для проектирования, моделирования и анализа компьютерных сетей, использующая язык Triad
16. TriadSimulation – модуль системы TriadNS, производящий проектирование, моделирование и анализ исследуемого объекта, состоит из редактора модели, анализатора модели, симулятора модели и редактора трансформации моделей
17. Базовая онтология TriadNS (TriadNS Base Ontology) – онтология основных структур системы
18. Модель в TriadNS (TriadNS Model) – графическая имитационная модель в системе TriadNS
19. Модель компьютерной сети (Computer Network Model) – графическая имитационная модель в системе TriadNS, входные данные для Model Transformer
20. Модель сети Петри (Petri Net Model) – графическая имитационная модель в системе TriadNS, выходная модель Model Transformer
21. Онтология (Ontology) – точная спецификация концептуализации
22. Онтология TriadNS (TriadNS Ontology) – онтология, являющаяся метамоделью для системы TriadNS, и входящая в состав системы, включает в себя базовую онтологию системы и онтологию предметной области (конкретной модели)
23. Онтология компьютерных сетей (Computer Network Ontology) – предметная онтология с описанием модели компьютерной сети в системе TriadNS
24. Онтология предметной области (Domain ontology) – онтология конкретной модели
25. Онтология сетей Петри (Petri Net Ontology) – предметная онтология с описанием сетей Петри в системе TriadNS
26. Дискретно-событийное моделирование (Descrete-Event simulation) – метод моделирования, когда поведение системы представляется в виде хронологической последовательности действий
27. Имитационная модель (Simulation model) – логико-математическое описание объекта, которое может быть использовано для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта
28. Имитационное моделирование (Simulation) – моделирование, при котором изучаемая система заменяется моделью, описывающей реальную систему, с которой проводятся эксперименты с целью получения информации об исходной системе
29. Метод (Method) – систематизированная совокупность шагов, действий, которые нацелены на решение определённой задачи или достижение определённой цели
30. Многомодельный подход в моделировании (Multi-model approach) – подход к моделированию на основе методов, когда объект исследуется с помощью различных моделей
31. Моделирование компьютерных сетей (Computer Network Simulation) – дискретно-событийное моделирование, исследующее компьютерные сети с помощью модели КС
32. Моделирование сетей Петри (Petri Net Simulation) – дискретно-событийное моделирование, исследующие сети Петри с помощью модели СП
33. WF-сеть (WF-net) – сеть Петри, в которой существуют только одна исходная (не имеет входящих дуг) и только одна конечная (не имеет выходящих дуг) вершины, а все остальные находятся по пути из исходной в конечную
34. Вершина (Vertex) – часть графа, представляющая некоторый объект
35. Временная сеть Петри (Timed Petri Net) – сеть Петри, в которой переходам соответствуют веса, определяющими время срабатывания
36. Граф (Graph) – математический объект, состоящий из множества вершин и дуг между ними
37. Графовая модель (Graph Model) – модель, представленная в виде графа
38. Ребро (Link) – часть графа, представляющая связь между объектами
39. Иерархическая сеть Петри (Hierarchical Petri Net) – сеть Петри, в которой вершиной может быть некоторая другая сеть Петри
40. Ингибиторная сеть Петри (Inhibitory Petri Net) – сеть Петри, в которой возможны ингибиторные дуги, запрещающие срабатывание перехода, если во входной позиции, связанной с переходом ингибиторной дугой, находится метка
41. Метамодель (Meta-model) – модель модели
42. Модель (Model) – абстрактное представление исследуемого объекта, отображающее основные и значимые свойства
43. Переход (Transition) – часть сети Петри, представляющая некоторое событие
44. Позиция (Place) – часть сети Петри, представляющая некоторый объект
45. Сеть Петри (Petri Net) – математический аппарат, состоящий из множества позиций и переходов
46. Стохастическая сеть Петри (Stochastic Petri Net) – сеть Петри, время срабатывания перехода в которой является случайным числом
47. Функциональная сеть Петри (Functional Petri Net) – сеть Петри, в которой время срабатывания перехода является функцией от некоторого аргумента
48. Цветная сеть Петри (Colored Petri Net) – сеть Петри, в которой вершины-позиции могут быть раскрашены (помечены)
49. Графовая грамматика (Graph grammar) – обобщение формальной грамматики, когда элементами алфавита являются графы
50. Нетерминальные символы (Nonterminal symbols) – объект, обозначающий какую-либо сущность языка и не имеющий конкретного символьного значения
51. Правила вывода (Production rules) – набор правил вида: «левая часть» -> «правая часть», где: «левая часть» — непустая последовательность терминалов и нетерминалов, содержащая хотя бы один нетерминал; «правая часть» — любая последовательность терминалов и нетерминалов
52. Правила трансформации (Transformation rules) – правила вывода графовой грамматики
53. Стартовый граф (Start graph) – стартовый символ графовой грамматики
54. Стартовый символ (Start symbol) – нетерминальный символ, с которого начинается выполнение правил вывода
55. Терминальные символы (Terminal symbols) – объект формального языка, имеющий в нём конкретное неизменяемое значение и являющийся элементом построения слов данного языка
56. Формальная грамматика (Formal grammar) – способ описания формального языка, состоящий из множества терминальных, нетерминальных символов, стартового символа и множества правил
57. Анализ потока информации (трафика) в системе (Data-flow Analysis of System) – задача исследования поведения динамических объектов в системе, задача сбора и обработки информации о данных, находящихся в системе, и их перемещения
58. Анализ сложных систем (Analysis of Complicated Systems) – задача исследования характеристик системы
59. Дополнительные исследования системы (Additional System Research) – задача исследования системы с помощью дополнительных методов [[1]](#footnote-1)
60. Задача (Task) – проблемная ситуация с явно заданной целью, которую необходимо достичь
61. Нахождение тупиков системе (Deadlocks Search) – задача поиска блокировок в системе
62. Прогнозирование поведения систем (System Behavior Prediction) – задача предсказания поведения системы с течением времени при заданных условиях
63. Проектирование систем (System Engineering) – задача создания, представления системы по выявленным требованиям, целям и задачам

Концептуализация выполнена в программе ОНТОЛИС и представлена на стр. 72-74.

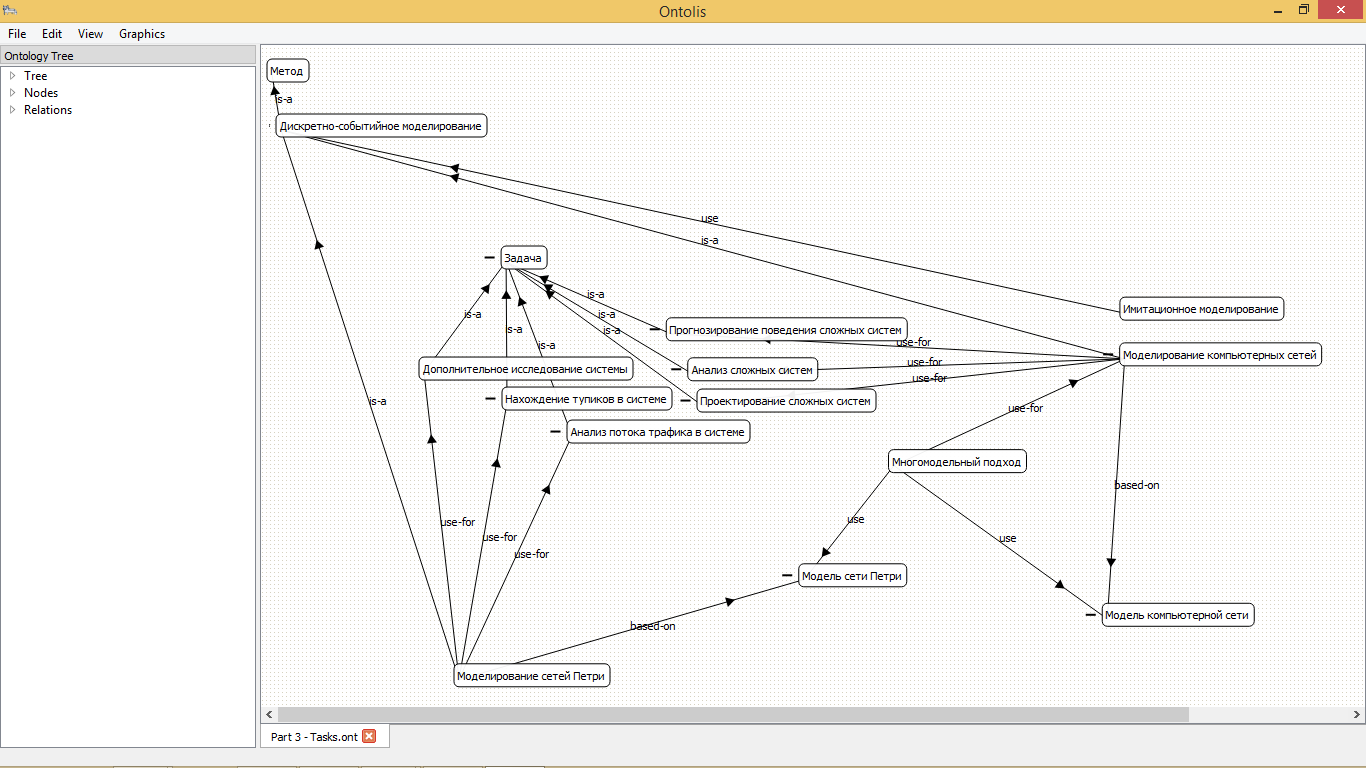
Фрагмент 1. Модули системы TriadNS и возможности ее использования



Фрагмент 2. Трансформация модели компьютерной сети в сеть Петри



Фрагмент 3. Задачи, которые решаются при использовании многомодельного подхода



## **Шаблоны рутин для элементов сети Петри**

Шаблон рутины для позиции

routine <Имя рутины>[integer InitialNumMarks;real T](InOut Place\_Tr[<кол-во вых.дуг>];Input Tr\_Place[<кол-во вх. дуг>])

initial

integer inarc,outarc;

inarc:=<кол-во вх.дуг>;

outarc:=<кол-во вых.дуг>;

integer curpol;

curpol:=1;

schedule TryQuery in 0;

array[2] of integer msg;

Print "Инициализация вершины";

endi

event TryQuery;

if InitialNumMarks>0 then

msg[0]:=1;

msg[1]:=curpol;

out IntArrayToStr(msg) through Place\_Tr[curpol];

Print "Отправка запроса через полюс "+ToStr(curpol);

schedule Wait in T;

endif;

ende

event Wait;

if curpol<>0 then

curpol:=curpol+1;

schedule TryQuery in 0;

endif;

if curpol=outarc then

curpol:=1;

schedule TryQuery in 0;

endif;

end;

event;

msg:=StrToIntArray(message);

if msg[0]=2 then

Print "Ответ получен через полюс "+ ToStr(msg[1]);

InitialNumMarks:=InitialNumMarks-1;

msg[0]:=3;

out IntArrayToStr(msg) through Place\_Tr[msg[1]];

Print "Отправляем фишку через полюс"+ ToStr(msg[1]);

endif;

if msg[0]=4 then

Print "Фишка получена через полюс "+ToStr(msg[1]);

InitialNumMarks:=InitialNumMarks+1;

schedule Show in 0;

schedule Wait in T;

endif;

ende

event Show;

Print "Количество фишек "+ToStr(InitialNumMarks);

ende

endrout

Шаблон рутины для перехода

routine <Имя рутины>[real T](InOut Place\_Tr[<кол-во вх.дуг>],Output Tr\_Place[<кол-во вых. дуг>])

initial

integer count;

count:=0;

integer answcount;

answcount:=0;

integer inarc,outarc;

inarc:=<кол-во вых.дуг>;

outarc:=<кол-во вх.дуг>;

array[<кол-во вх.дуг>] of boolean pol;

array[<кол-во вх.дуг>] of boolean answ;

for i:=0 to inarc-1 do

pol[i]:=false;

answ[i]:=false;

endf;

array[2] of integer msg;

Print "Инициализация вершины";

endi

event;

msg:=StrToIntArray(message);

if msg[0]=1 then

Print "Получен запрос из полюса "+ToStr(msg[1]);

if (pol[msg[1]-1]=false) then

pol[msg[1]-1]:=true;

count:=count+1;

schedule Verify in 0;

endif;

else

if msg[0]=3 then

Print "Фишка получена через полюс "+ToStr(msg[1]);

if (answ[msg[1]-1]=false) then

answ[msg[1]-1]:=true;

answcount:=answcount+1;

schedule Send in T;

endif;

endif;

endif;

ende

event Send;

if (answcount=inarc) tnen

msg[0]:=4;

for i:=0 to outarc-1 do

out IntArrayToStr(msg) through Tr\_Place[i];

Print "Передача фишки через полюс "+ToStr(msg[1]);

endf;

count:=0;

for i:=0 to inarc-1 do

pol[i]:=false;

answ[i]:=false;

endf;

answcount:=0;

endif;

ende

event Verify;

if (count=inarc) then

Print "Получен запрос от всех вершин";

msg[0]:=2;

for i:=0 to inarc-1 do

msg[1]:=i+1;

out IntArrayToStr(msg) through Place\_Tr[i];

Print "Отправляем ответ через полюс "+ToStr(msg[1]);

endf;

endif;

ende

endrout

1. Например, исследование распределенных или параллельных систем с помощью метода анализа сетей Петри – дерево достижимости сетей Петри [↑](#footnote-ref-1)