# Модель обработки данных вычислительных экспериментов

**Д.В. Леонтьев, Г.В. Тарасов, Д.И. Харитонов**

*Институт автоматики и процессов управления ДВО РАН*

**Аннотация.** Рассматривается подход к построению моделей обработки данных вычислительных экспериментов. Для построения моделей используются сети Петри. Модель вычислительного эксперимента состоит из моделей вычислительного и управляющего процессов. Построение этих моделей происходит раздельно. Построение модели вычислительного процесса происходит в два этапа. Сначала пользователь формирует дерево событий вычислительного эксперимента. Далее происходит автоматическое построение модели вычислительного процесса из дерева событий. При построении модели используются шаблонные конструкции, которые склеиваются между собой. Каждому элементу из дерева событий соответствует своя шаблонная конструкция. Модель управляющего процесса строится на основе шаблонов реакции. Представлены три шаблона реакции: реакция на предыдущее событие, реакция на каждое N-ое событие, реакция на следующее событие. Шаблон реакции настраивается на событие, на которое он должен реагировать. Важной чертой подхода является отсутствие необходимости в перепрограммировании исходной вычислительной задачи. Подход позволяет строить модели обработки данных пользователям с минимальной подготовкой. В работе приводятся примеры построения моделей.

**Ключевые слова:** суперкомпьютерные вычисления, обработка больших данных, многопроцессорные вычислительные системы, визуализация научных данных, сети Петри

# The data processing model of computational experiments

**D.V. Leontev, G.V. Tarasov, D.I. Kharitonov**

*Institute of automation and control processes FEB RAS*

**Abstract.** An approach to construct a data processing models of a computational experiments is considered. To make models the Petri nets are used. The model of computational experiment consists of computational and control processes models. The models are built separately. The computational process model is built in a two stages. On the first stage the user generates the event tree of computational experiment. On the second stage the computational process model is automatically built from the event tree. For the building of the model a template constructions are used, which are composed together. Each element in the event tree has its own template constructions. The model of the control process is built from a reaction patterns. The following three reaction patterns are developed: reaction on a previous event, reaction on each N-th event, reaction on a next event. The reaction pattern is configured on the triggered event. The main feature of the developed approach is that there is no need to reprogram an original computational task. The approach allows users with minimal skills to make the data processing models. The examples of constructing of data processing models are presented.

**Keywords:** high performance computing, big data processing, multiprocessor computing systems, scientific data visualization, Petri nets

Для решения сложных численных задач за приемлемое время необходимо использовать вычислительные кластера. Работа с такими кластерами происходит удаленно. Вычислительный эксперимент обычно представляет собой некоторую последовательность рутинных действий, выполняемых либо вручную, либо с помощью скриптов, поставляемых вместе с используемым вычислительным пакетом или разработанных пользователем самостоятельно. В общем виде процесс проведения эксперимента можно разбить на подготовку входных данных, выполнение эксперимента, визуализацию результатов и их оценку.

Входные данные для вычислительного эксперимента записываются в текстовом или бинарном виде. Однако процесс их формирования для каждого типа вычислительного эксперимента может серьезно отличаться. Поэтому для разных экспериментов требуются разные функциональные возможности редакторов. Например, для формирования входных данных для пакетов Gamess, FireFly, Gaussian часто требуются такая возможность как вставка и удаление столбцов данных, что не поддерживается традиционными редакторами. Другим примером может послужить обработка карт высокого пространственного разрешения, для работы с которыми может потребоваться объем оперативной

памяти, превышающий возможности настольного компьютера. Также результаты одного вычислительного эксперимента могут быть входными данными другого эксперимента, при этом может возникнуть необходимость преобразования формата представления данных. Исходя из этого, можно сделать вывод, что на подготовку входных данных, включая передачу через сеть Интернет, преобразование форматов и другие необходимые операции, затрачивается значительное количество времени и трудозатрат со стороны пользователя.

Выполнение вычислительного эксперимента на кластере включает в себя некоторую последовательность операций: подготовка среды выполнения вычислительного эксперимента, постановка задачи в систему очередей, получение результатов и другие действия, специфичные для каждого вычислительного кластера. Администраторам вычислительных кластеров приходится разрабатывать целый спектр программ, утилит и скриптов для упрощения работы пользователя. Однако, не всегда пользователь может правильно использовать весь предоставленный спектр средств, что в свою очередь приводит к лишним затратам сил и времени пользователя на решение уже решенных проблем.

Наиболее сложным этапом вычислительного эксперимента, является качественная визуализация и интерпретация полученных результатов. В результате проведения экспериментов генерируются значительные объемы данных, которые превосходят человеческие аналитические возможности. Поэтому для оценки результатов экспериментов и их презентации широкому кругу лиц используется так называемая визуализация научных данных. Следует учитывать, что сама визуализация данных может требовать значительных вычислительных ресурсов и дискового пространства, поэтому наиболее выгодным вариантом становится визуализация данных непосредственно на кластере [1, 2]. Такой вариант позволяет избавиться от загрузки результатов «неудачных» экспериментов на компьютер пользователя, что существенно экономит время. Дополнительным плюсом при визуализации данных является разгрузка сетевых соединений и как следствие более быструю доставку результатов до конечного потребителя. Так в процессе визуализации данные, занимающие гигабайты дискового пространства могут быть преобразованы к читаемому пользователем виду, занимающему в сотни, а то и в тысячи раз меньше места. Существуют различные варианты представления данных. Например, таблицы со статистическими расчетами; 2х-мерные и 3х-мерные графики, гистограммы и диаграммы; деревья и графы; проекция данных на поверхность визуализации в виде раскраски или нормалей, отображение данных на картах, анимация изменения данных и аудиоанимация данных и т. д.

Существует и разрабатывается большое количество программных средств, предназначенных для визуализации данных. Значительная часть этих средств выпускается под лицензиями, допускающими свободное распространение. В частности, можно отметить средства для визуализации

массивов данных: ParaView, gnuplot, Visualization Toolkit, SALOME, Mayavi [3]. Есть средства для визуализации информации на картах GMT (General Mapping Tools), QGIS. Для визуализации молекул и химических комплексов разработаны Ascalaph Designer, CCP4mg, OpenStructure. Есть множество средств для визуализации деревьев и графов, например Gephi, Visual Graph, Graphviz, Tulip. На практике, при применении средств визуализации приходится решать такие типичные задачи как: преобразование данных между форматами разных программных систем, вычисление опорных данных для визуализации, таких как размеры сетки, максимальные и минимальные значения, и т. д., а также распараллеливание процесса визуализации.

Каждый из описанных выше этапов выполнения вычислительного эксперимента может быть автоматизирован или упрощен для пользователя. Однако для решения этой задачи необходимо иметь полное представление о ходе вычислительного эксперимента и о происходящих в его рамках процессах и событиях. Понимание хода вычислительного эксперимента позволяет осознать, какие компоненты (программы, скрипты, инструменты и т.д.) требуются для проведения эксперимента. Наиболее наглядным и удобным подходом для представления этих процессов являются модели. Для построения моделей авторы предлагают использовать математический аппарат сетей Петри, который представляется наиболее проработанным и интересным для описания динамических процессов. В данной работе авторы рассматривают подход к построению моделей обработки данных вычислительных экспериментов.

## 1. Модель вычислительного процесса

Модель вычислительного процесса описывает все взаимодействия процесса с «внешним» окружением в рамках вычислительного эксперимента. При описании модели используются понятия, имеющие аналогии в программирование, такие как варианты (ветвления), циклы, стадии (последовательность команд), события (команды).

Модель представляет собой сеть Петри, состоящую из множества мест, множества переходов, входной функция инцидентности переходов (мультимножества мест необходимых для возбуждения переходов) и выходной функции инцидентности переходов. В рамках данной статьи модели в сетях Петри строятся при помощи шаблонных конструкций. В общем виде процесс построения заключается в «склеивании» шаблонных конструкций в сеть с помощью операции композиции по местам. У каждого шаблона есть входные и выходные точки доступа. У каждой точки доступа есть количество мест, которое необходимо для «склеивания» с другим шаблоном. Это количество мест называется мощностью точки доступа. Выходные точки доступа могут «склеиваться» только с входными точками доступа, при этом их мощности должны совпадать.

Построение модели вычислительного процесса можно разделить на два этапа. На первом этапе пользователь формирует дерево событий, происходящих в рамках вычислительного эксперимента, на которые необходимо реагировать. На втором этапе на основе дерева событий происходит автоматическое построение модели вычислительного процесса в терминах сетей Петри из шаблонных конструкций. Шаблонные конструкции и примеры построения моделей приведены далее.

Рассмотрим формирование пользователем дерева событий, с использованием шаблонных элементов, представленных на рис. 1. В рамках данной работы определены следующие допустимые шаблонные элементы:

- цикл – все «дочерние» элементы 1-го уровня вложенности представляют собой последовательность событий, происходящих в цикле;
- стадия (шаг) – все «дочерние» элементы 1-го уровня вложенности представляют собой последовательность событий, происходящих друг за другом;
- варианты – все «дочерние» элементы 1-го уровня вложенности представляют собой возможные варианты событий, которые могут произойти;
- событие – действие вычислительного процесса с «внешним» окружением. Допустимые события представлены в списке ниже по тексту. Событие может содержать параметры, такие как название файла, способ записи данных (слои данных сохраняются в одном файле или в разных) и т.д. У данного элемента не может быть «дочерних» элементов;
- невидимое событие – действие вычислительного процесса с «внешним» окружением, которое не входит в список допустимых событий. У данного элемента не может быть «дочерних» элементов.

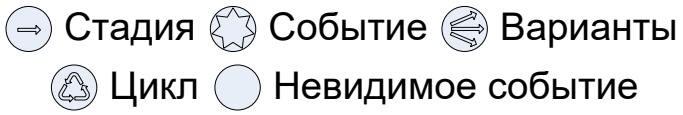


Рис. 1. Шаблоны элементов дерева событий

Каждый шаблонный элемент содержит три параметра:

- тип элемента – обозначает тип шаблонной конструкции, которая будет использована при генерации модели (допустимые шаблонные элементы показаны далее на рис. 3);
- значение элемента – этот параметр задает тип события (этот параметр является значимым только для элемента «Событие», для остальных элементов он может использоваться как комментарий);
- ссылка на элемент «родитель» – используется для построения связей между элементами дерева.

В рамках данной работы определены следующие допустимые события:

- запись в файл;
- запись в лог;
- запись в базу данных;
- запрос данных;
- Fail;
- взаимодействие с сокетами;
- запуск программы.

Пример построенного дерева событий представлен на рис. 2.


Рис. 1. Пример дерева событий

Следующим шагом происходит автоматическая генерация модели (сети Петри) из дерева событий. Построение сети Петри происходит с использованием шаблонных конструкций (рис. 3), которые соответствуют шаблонным элементам. При изображении шаблонов используются следующие обозначения. Сеть Петри, описывающая структуру шаблона, помещается в прямоугольник. Для изображения сетей Петри используется обычная графическая нотация в виде двудольного ориентированного графа, где места изображаются окружностями, а переходы – прямоугольниками. Места и переходы соединяются дугами. На границах прямоугольника размещаются символические изображения простых точек доступа по местам в виде окружности. Все входные точки доступа по местам размещаются сверху прямоугольника, а выходные – снизу.

«Стартовый шаблон» - единственный в описываемом наборе шаблон, в котором отсутствует входная точка доступа. С данного шаблона начинается

построение любой модели. Первое место, в котором находится токен, является стартовой точкой модели. У шаблона одна выходная точка доступа.

Шаблон «Цикл» предназначен для моделирования циклов. Первая точка доступа выходного интерфейса этого шаблона используется для моделирования тела цикла. Вторая точка доступа – для продолжения модели после цикла.

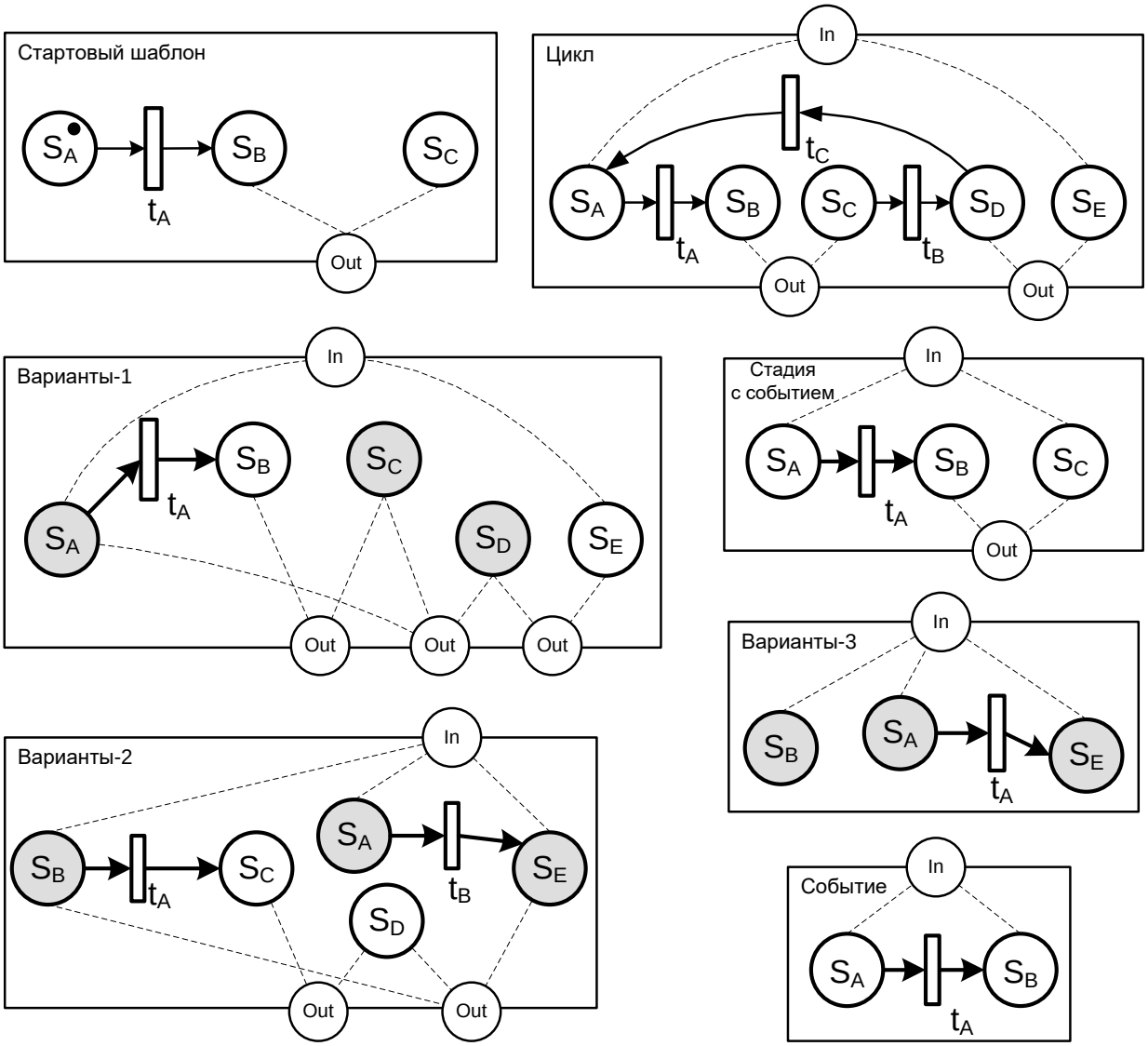


Рис. 3. Шаблонные конструкции

Шаблоны «Варианты-1,2,3» моделируют части конструкции варианты исполнения: основной шаблон «Варианты-1» – начало и конец конструкции, «Варианты-2» - альтернативная ветвь исполнения, «Варианты-3» - заглушка. Основной шаблон «Варианты-1» имеет одну точку доступа из пары мест во входном интерфейсе и три точки доступа в выходном интерфейсе, предназначенные для продолжения конструкции варианты, построения тела первого варианта выполнения и продолжения модели. Точка доступа для продолжения конструкции переключатель имеет три места, а две других точки

доступа – по два. Шаблон «Варианты-3» имеет единственную входную точку доступа из трёх мест и ни одной выходной, поэтому он является завершающим для конструкции варианты, так как после «склейки» конструкции переключатель с шаблоном «Варианты-3» добавление новых вариантов будет невозможно. Шаблон «Варианты-2» предназначен для добавления нового варианта исполнения. С помощью единственной точки доступа из трёх мест во входном интерфейсе, эти шаблоны могут быть «склеены» только с шаблонами из набора конструкции варианты. Выходной интерфейс состоит из двух точек доступа: первая точка из трёх мест предназначена для развития конструкции варианты, вторая точка доступа из двух мест – для построения тела этого случая. Таким образом, для построения конструкции варианты исполнения необходимо использовать основной шаблон, затем «склеить» его по порядку с необходимым количеством вариантов исполнения и закончить завершающей конструкцией.

Шаблон «Стадия с событием» моделирует событие с возможностью «склейки» с другими шаблонами.

Шаблон «Событие» моделирует событие и имеет только входной интерфейс из пары мест, поэтому после «склейки» с точкой доступа входного интерфейса в результирующей сети будет на одну выходную точку доступа меньше.

Например, для дерева событий изображенного на рис. 2 модель в терминах сетей Петри будет выглядеть, как изображено на рис. 4. Более подробно узнать о процессе построения моделей в терминах сетей Петри можно в статье [4].

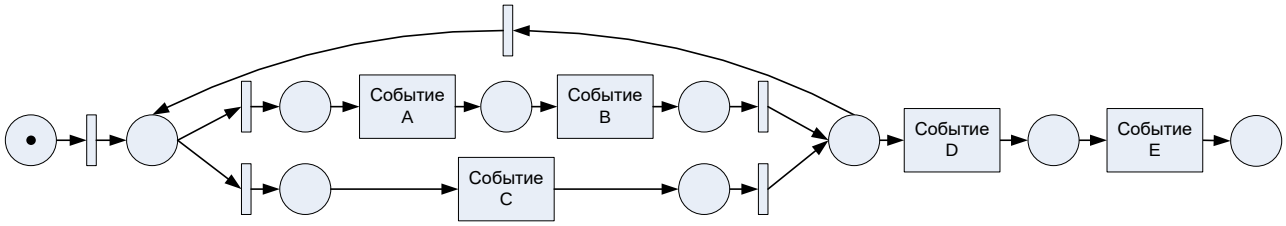


Рис. 4. Результат преобразования дерева событий в модель в терминах сетей Петри

**2. Модель управляющего процесса.**

Модель управляющего процесса описывает работу процесса, отвечающего за отслеживание событий, генерируемых вычислительным процессом и реагирование на них. После построения модели вычислительного процесса, каждому событию сопоставляется некоторый шаблон реакции. Шаблон реакции позволяет установить зависимость происходящих событий и реакции на них. В рамках данной работы используются три шаблона реакций, которые изображены на рис. 5. Шаблон реакции также представляют собой сеть Петри. В стартовом месте находится токен, с которого начинается выполнение сети. В прямоугольниках записываются события, а в кружках состояния. У

событий могут быть условия срабатывания, которые указываются в прямоугольнике под событием. На дугах указываются переменные, которые должны быть переданы.

Отслеживание событий вычислительного процесса является малозатратной деятельностью и практически не влияет на балансировку загрузки узлов, в то время как реакция на события может быть длительным и ресурсоёмким процессом. В рамках данной статьи обработка данных в ходе реакции рассматривается как самостоятельный процесс, запуск которого может быть настроен параллельно основному через систему очередей на вычислительном кластере. Вопросы тонкой настройки событий реакции, для получения подходящей картины визуализации или необходимых статистических данных, не влияют на моделирование процесса обработки данных, и могут быть выполнены в отдельных файлах настройки реакции на события.

Шаблон «Реакция на предыдущее событие» - данный шаблон предназначен для реагирования на событие, которое предшествовало текущему событию. Шаблон «Реакция на каждое N-ое событие» - данный шаблон предназначен для реагирования на каждое N-ое событие. Шаблон «Реакция на следующее событие» - данный шаблон предназначен для реагирования на событие, которое следует сразу за текущим событием. Работа данных реакций будет рассмотрена в примерах ниже.

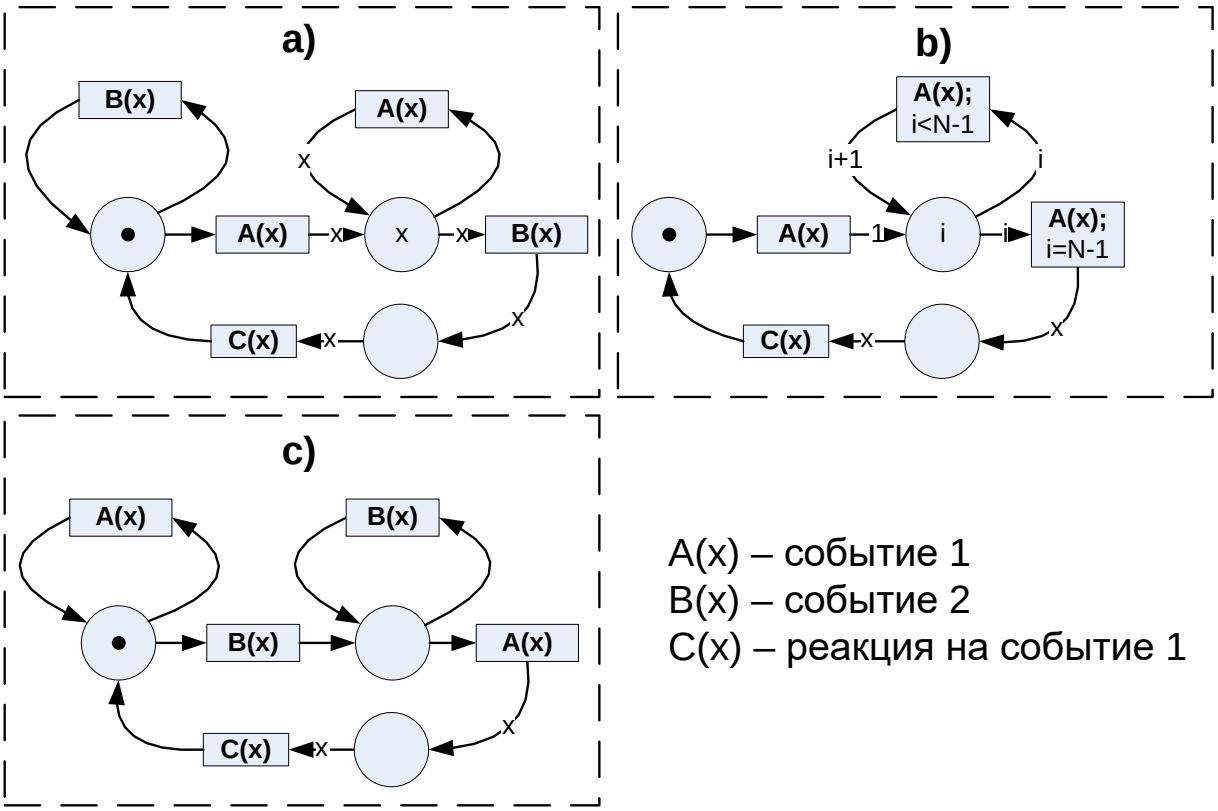


Рис. 5. Шаблоны реакций: a) реакция на предыдущее событие , b) реакция на каждое N-е событие, c) реакция на следующее событие.

### 3. Примеры построения моделей вычислительных экспериментов

Рассмотрим несколько реальных вычислительных экспериментов и построим для них соответствующие модели в терминах сетей Петри.

**Пример №1**. Моделирование волн цунами. Вычислительный эксперимент носит итерационный характер. Итерации происходят по расчётному времени. Итерации отличаются друг от друга по значению этого времени. Раз в заданный интервал времени происходит запись расчётных массивов в файл с уникальным именем (например, eta0001, eta0002 и т.д.). Раз в заданный интервал модельного времени происходит запись усреднённых величин в лог. После записи усреднённых величин необходимо визуализировать расчётные массивы (данные берутся из файла, записанного до записи лога). По окончанию расчета сохраняются массивы данных в файлы, и записывается информация в лог. После этого происходит визуализация всех записанных массивов и склейка в видеоролик.

На рис. 6 представлено дерево событий для примера №1. Далее это дерево событий преобразовывается в модель вычислительного эксперимента (рис. 7). Следующим шагом происходит построение модели управляющего процесса (рис. 8). Для этого из шаблонов реакций выбираются необходимые реакции. В данном примере используется шаблон «реакция на предыдущее событие». На рис. 8(a) приведена модель реакции на событие запись усредненных значений в лог, происходящая в вычислительном эксперименте в цикле. Т.е. в процессе вычислительного эксперимента происходит получение события W(x). Параметры данного события сохраняются и передаются далее. Получение события W(x) происходит до тех пор, пока не будет получено событие WL(x), после которого будет выполнена процедура визуализации последнего массива данных Z(x), полученного в событие W(x). А на рис. 8(b) представлена реакция на событие финальной записи в лог. Отличие данной реакции от предыдущей заключается в отслеживаемом событии WLF(x) (финальная запись в лог) и реакции на него M(x) (визуализация всех массивов данных и генерация видео).

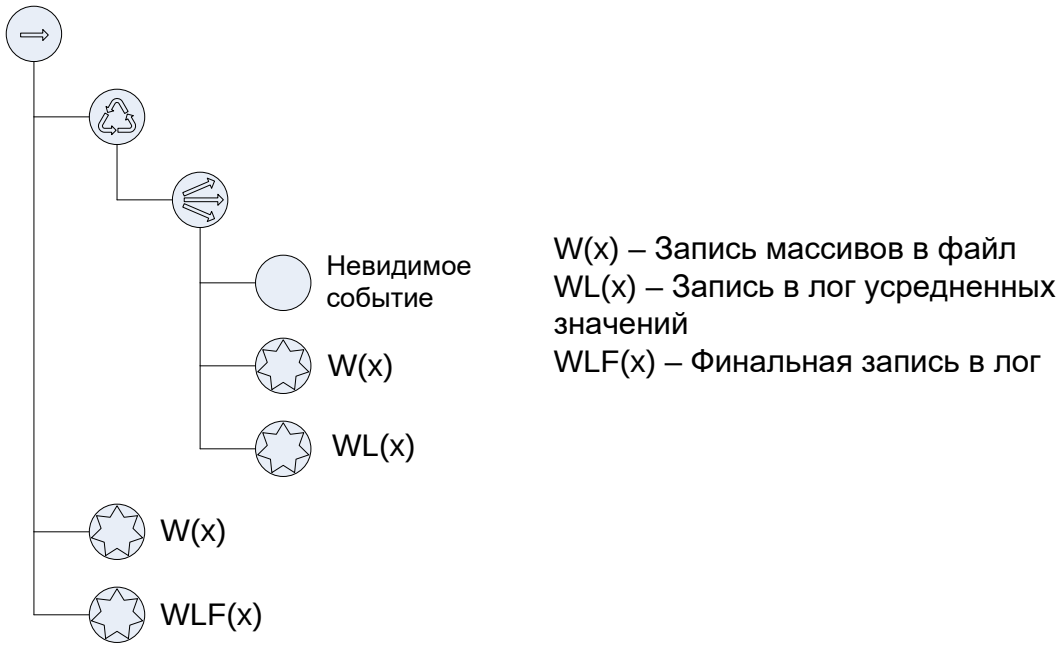


Рис. 6. Дерево событий для примера №1

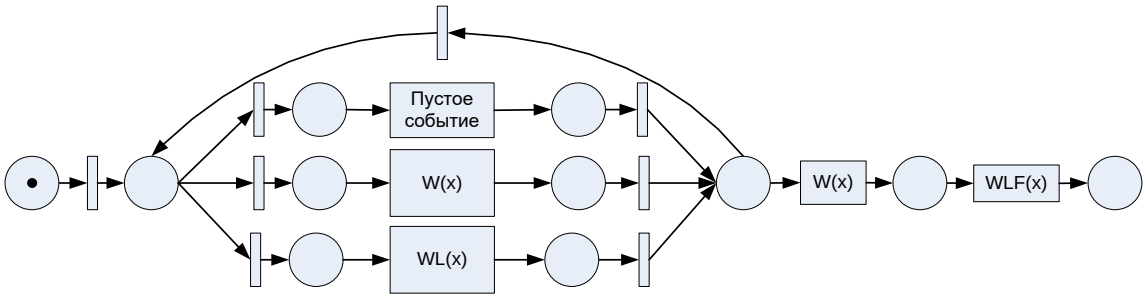


Рис. 7. Модель вычислительного процесса для примера №1

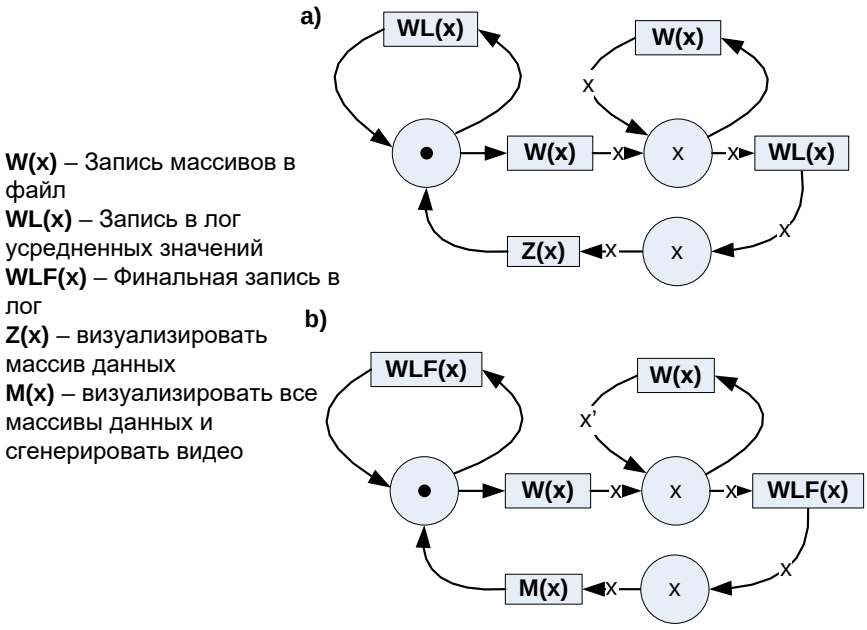


Рис. 8. Модель управляющего процесса для примера №1: a) реакция на событие запись усредненных значений в лог, b) – реакция на событие финальная запись в лог.

**Пример №2.** Вычислительный эксперимент носит итерационный характер. Все итерации записываются в один файл. Раз в N-е количество записей необходимо найти связи, имеющие максимальные показатели энергии. Найденные связи необходимо отобразить на изображении.

На рис. 9(a) представлено дерево событий для примера №2. Далее это дерево событий преобразовывается в модель вычислительного эксперимента (рис. 9(b)). Следующим шагом происходит построение модели управляющего процесса (рис. 10). В данном примере используется шаблон «Реакция на каждое N-ое событие». Т.е. в процессе вычислительного эксперимента происходит подсчет полученных событий W(x) и при достижении необходимого количества N происходит обработка данных и отображение связей с максимальными показателями энергии на картинке Z(x).

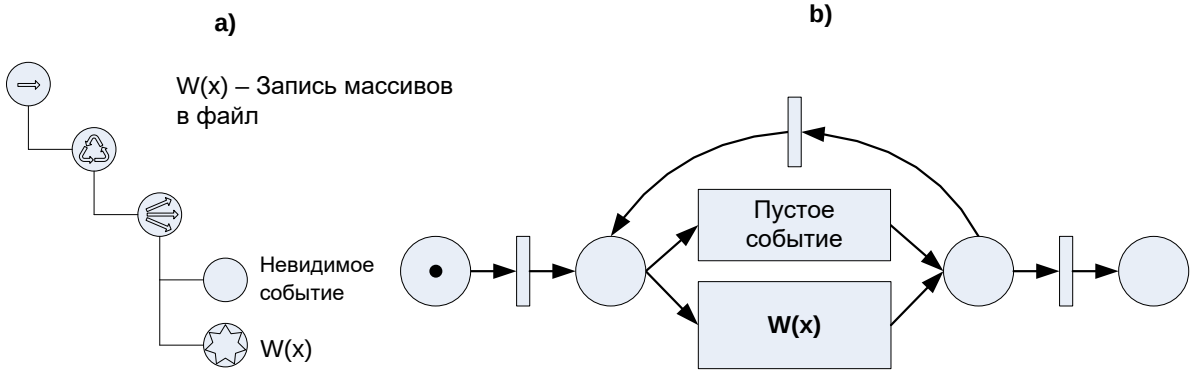


Рис. 9. a) Дерево событий для примера №2. b) Модель вычислительного процесса для примера №2.

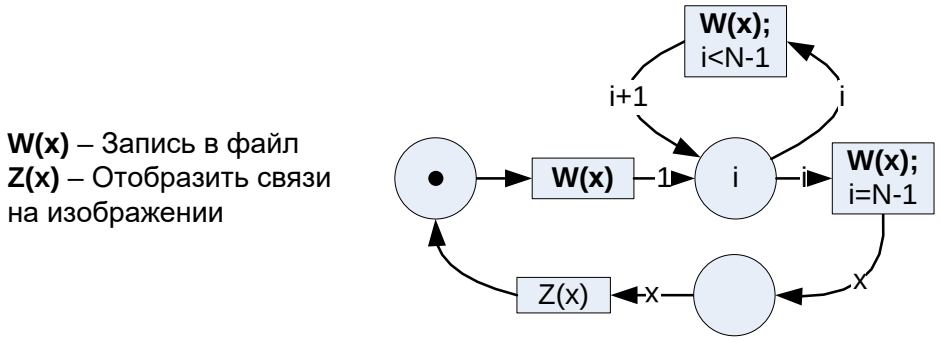


Рис. 10. Модель управляющего процесса для примера №2

## 4. Заключение

В представленной работе рассмотрен механизм построения моделей в терминах сетей Петри, моделирующих процесс обработки данных при проведении вычислительного эксперимента. Этот механизм является основой нового, не имеющего аналогов, подхода в организации обработки данных вычислительных экспериментов. Построение моделей вычислительного и управляющего процессов происходит раздельно. Взаимодействие между ними происходит благодаря использованию композиционального подхода к сетям Петри. Важной чертой подхода является отсутствие необходимости в перепрограммировании исходной вычислительной задачи и возможность построения моделей обработки данных пользователем с минимальной подготовкой. В дальнейшем данный подход будет реализован в рамках системы управления прохождения заданиями WBS [5].

Работа выполнена в рамках проекта ДВО РАН «Модели и методы автоматизации разработки гибридных интерфейсов и визуализации результатов web-приложений, реализованных на гетерогенных вычислительных архитектурах», проект 18-5-078.

## Литература

1. Джосан О.В. О визуализации научных данных для высокопроизводительных параллельных приложений // тезисы конференции ПАВТ 2009, Россия, Нижний Новгород, 2009.
2. Корж О.В. Автоматическое построение передаточных функций для систем визуализации распределенных данных на суперкомпьютерах // Параллельные вычислительные технологии (ПАВТ 2012): труды международной научной конференции (Новосибирск, 26 – 30 марта 2012). — Издательский центр ЮУрГУ Челябинск, 2012. — С. 726–726.
3. Ramachandran, P. and Varoquaux, G., `Mayavi: 3D Visualization of Scientific Data` IEEE Computing in Science & Engineering, 13 (2), pp. 40-51 (2011).
4. Leontyev D. V., Kharitonov D. I., Tarasov G. V. Imperative programs behavior simulation in terms of compositional Petri nets // International Journal of Computer Networks & Communications (IJCNC) vol. 10, №1 January 2018.

5. Одякова Д.С., Тарасов Г.В., Харитонов Д.И. Система WBS как расширенный инструмент управления вычислительной средой // Суперкомпьютерный форум «Суперкомпьютерные технологии в образовании, науке и промышленности». 26 - 28 ноября 2012 г., Нижний Новгород, Нижегородский государственный университет им. Н.И.Лобачевского.

**References**

1. Dzhosan O.V. O vizualizatsii nauchnykh dannykh dlia vysokoproizvoditelnykh parallelnykh prilozhenii // tezisy konferentsii PAVT 2009, Russia, Nizhny Novgorod, 2009.
2. Korzh O.V. Avtomaticheskoe postroenie peredatochnykh funktsii dlia sistem vizualizatsii raspredelennykh dannykh na superkompiuterakh // Parallelnye vychislitelnye tekhnologii (PAVT 2012): trudy mezhdunarodnoi nauchnoi konferentsii (Novosibirsk, 26 – 30 march 2012.). — Izdatelskii tsentr SUSU Chelyabinsk, 2012. — P. 726–726.
3. Ramachandran, P. and Varoquaux, G., `Mayavi: 3D Visualization of Scientific Data` IEEE Computing in Science & Engineering, 13 (2), P. 40-51 (2011).
4. Leontyev D. V., Kharitonov D. I., Tarasov G. V. Imperative programs behavior simulation in terms of compositional Petri nets // International Journal of Computer Networks & Communications (IJCNC) vol. 10, №1 January 2018.
5. Odiakova D.S., Tarasov G.V., Kharitonov D.I. Sistema WBS kak rasshirennyi instrument upravleniia vychislitelnoi sredoi // Superkompiuternyi forum «Superkompiuternye tekhnologii v obrazovanii, nauke i promyshlennosti». 26 - 28 november 2012, Nizhny Novgorod, Nizhegorodskii gosudarstvennyi universitet im. N.I.Lobachevskogo.