А.Г. Жихарев, С.И. Маторин, Н.О. Зайцева

Системно-объектный инструментарий для имитационного моделирования технологических процессов и транспортных потоков[1]

Аннотация. В работе рассматривается новый метод имитационного моделирования с применением подхода «Узел-Функция-Объект», системно-объектного метода представления знаний, математической теории объектов и языка описания функциональных узлов «УФО-скрипт». Приводятся тестовые примеры имитационного моделирования технологического процесса и транспортных потоков, выполненные с помощью новой версии программного инструментария системно-объектного моделирования.

Ключевые слова: системный подход «Узел-Функция-Объект», системно-объектный метод представления знаний, исчисление объектов, язык «УФО-скрипт», технологический процесс, транспортный поток.

Введение

Успехи имитационного моделирования давно привели к его признанию как перспективного научного направления и широкому использованию его инструментальных средств для решения различных практических задач. Однако естественные в любой развивающейся сфере науки и техники проблемы стимулируют продолжение исследований и разработок, в том числе, и в области имитационного моделирования.

В настоящее время не существует общепризнанного определения понятия *имитационное моделирование*. Поэтому для конкретизации предлагаемых в данной статье результатов необходимо уточнить авторское понимание сути данного научно-практического направления. Оно в целом соответствует определениям, изложенным в интервью Президента Национального общества имитационного моделирования, директора СПИИРАН Р.М. Юсупова [1]. Наиболее адекватным направлению деятельности авторов является упомянутое в данном интервью, а также в статье [2], понимание имитационного моделирования как «метода исследования, при котором изучаемая система заменяется *моделью, с достаточной точностью описывающей реальную систему* (курсив наш). С ней проводятся эксперименты с целью получения информации об этой системе» [1, с.11]. Следуя этому определению, исследования и разработки авторов находятся в рамках дискретно-событийного подхода к имитационному моделированию.

Необходимость замены изучаемой системы моделью, с достаточной точностью ее описывающей, естественным образом вынуждает использовать системный подход для создания такой модели. Авторы для моделирования систем применяют оригинальный системно-объектный подход «Узел-Функция-Объект» (*УФО-подход*) [3]. Данные подход позволяет целостно описывать любую систему как элемент «Узел-

[1] Работа выполнена при финансовой поддержке РФФИ (проекты № 13-07-00096, №13-07-12000, 14-47-08003).

Функция-Объект» (*УФО-элемент*) и при этом одновременно с трех точек зрения:

– как *структурного элемента* надсистемы в виде перекрестка связей с другими системами — *узла*;

– как *динамического элемента*, выполняющего определенную роль с точки зрения поддержания надсистемы путем балансирования данного узла — *функции*;

– как *субстанциального элемента*, реализующего данную функцию в виде некоторого материального образования, обладающего конструктивными, эксплуатационными и т.д. характеристиками — *объекта*.

УФО-элементы, собранные в различные конфигурации, образуют диаграммы взаимодействия элементов, которые позволяют визуализировать функциональность элементов системы более высоких уровней. Таким образом, разрабатываемая система представляется в виде иерархии УФО-элементов. Данное представление позволяет учесть различные аспекты рассмотрения этой системы (структурные, функциональные, субстанциальные) в одной системно-объектной модели – *УФО-модели*.

В целях автоматизации применения УФО–подхода был спроектирован и реализован CASE-инструментарий «*UFO-toolkit*» (*http://www.ufo-toolkit.ru/*). Иерархия УФО-элементов и их конфигураций, которую поддерживает UFO-toolkit, основана на классификации связей (потоков), пересечения которых и образуют узлы. Моделирование любой системы начинается со специализации базовой классификации связей под конкретную предметную область. Абстрактный класс «Связь (*L*)» в базовой классификации связей делится на подклассы «Материальная связь (*M*)» и «Информационная связь (*I*)». Класс материальных связей делится на подклассы «Вещественная связь (*S*)» и «Энергетическая связь (*E*)», класс информационных связей – на подклассы «Связь по данным (*D*)» и «Управляющая связь (*C*)».

Рассмотрим результаты проектирования новой версии CASE-инструментария UFO-toolkit, предназначенной не только для спецификации организационно-деловых и производственно-технологических процессов, но и для имитации их функционирования на основе системно-объектных УФО-моделей.

1. Концептуальные и формальные основы

На основе УФО-подхода авторами разработан формализованный «Системно-Объектный Метод Представления Знаний» (*СОМПЗ*) как инструмент создания универсальных моделей знаний о системах произвольной природы [4-6]. Формальный аппарат СОМПЗ позволяет описывать УФО-элементы и связи/потоки в терминах *исчисления объектов* Абади-Кардели [7]. При этом УФО-элемент представляется в виде специализированного абстрактного объекта (*узлового объекта*) данного исчисления:

$$G = [l?_i = a?_i,\ l!_j = a!_j;\ l_n = F(l?_i)l!_j;\ l_m = b_m],$$

где G - узловой объект;

$l?_i$ - поле узлового объекта (может представлять собой набор или множество), которое содержит значение входных потоковых объектов $a?_i$ и, соответственно, имеет такой же тип данных;

$l!_j$ - поле узлового объекта (может представлять собой набор или множество), которое содержит значения выходных потоковых объектов $a!_j$ и имеет такой же тип данных;

l_n - метод узлового объекта (может представлять собой набор или множество), преобразующий входные потоковые объекты узла в выходные;

l_m - поле узлового объекта (может представлять собой набор или множество), которое содержит основные характеристики данного объект (b_m).

Таким образом, узел УФО-элемента описывается наборами входных ($a?_i$) и выходных ($a!_j$) потоков объекта G, функция УФО-элемента – методом ($F(l?_i)l!_j$) объекта G, а объект УФО-элемента – в виде набора констант (b_m), характеризующих объект G.

Связь/поток также представляется в виде специализированного абстрактного объекта (*потокового объекта*) упомянутого выше исчисления:

$$a_i = [l_j = b_j],$$

где a_i – потоковый объект;

$l_j = b_j$ – поля потокового объекта с некоторыми значениями b_j.

Так как вычисление объектов представляет собой последовательность вызовов и переопределения методов, для чего определены правила

редукции, то в СОМПЗ формальные преобразования выполняются в соответствии с *правилом вызова метода узлового объекта* следующего вида: $G.l_n\ l!_j\{l?_i \mapsto G\}$.

В терминах УФО-подхода имитационная модель системы, в первую очередь, представляет собой конфигурацию УФО-элементов (УФО-модель), «с достаточной точностью описывающую эту систему». Математически, в терминах формализма СОМПЗ, имитационная УФО-модель представляет собой комбинацию узловых объектов и потоковых объектов, взаимодействие которых определяется упомянутым выше правилом вызова метода:

$$a_i = [l_m = b_m]: a_i = a?_i = l?_i \rightarrow G_k.l_n \rightarrow l!_j\{l?_i \rightarrow G_k\} \rightarrow a_{i+1} = [l_{m+1} = b_{m+1}]: a_{i+1} = a?_{i+1} = l?_{i+1} \rightarrow G_{k+1}.l_{n+1} \rightarrow l!_{j+1}\{l?_{i+1} \rightarrow G_{k+1}\} \rightarrow a_{i+2} = [l_{m+2} = b_{m+2}]: a_{i+2} = a?_{i+2} = l?_{i+2} \rightarrow G_{k+2}.l_{n+2} \rightarrow l!_{j+2}\{l?_{i+2} \rightarrow G_{k+2}\} \rightarrow a_{i+3} = [l_{m+3} = b_{m+3}]: \ldots$$

Таким образом, функционирование имитационной модели определяется методами узловых объектов, которые описывают, как входные потоковые объекты трансформируются в выходные потоковые объекты.

2. Лингвистическое обеспечение

В настоящее время авторами разрабатывается новая версия программного инструментария системно-объектного моделирования, в котором реализована возможность описания методов узловых объектов с помощью языка описания функциональных узлов *УФО-скрипт*, синтаксически подобного языку программирования Pascal.

Рассмотрим особенности предлагаемого языка подробнее. Для объявления переменных, используемых в рамках описания функционирования конкретного процесса, используется следующее выражение:

var <имя переменной 1>,<имя переменной 2>, <…>: тип данных.

В приведенном примере имена переменных следуют после ключевого слова «*var*», далее через запятую перечисляются имена переменных, после двоеточия указывается тип данных, хранящихся в переменных. УФО-скрипт позволяет работать со следующими типами данных: *integer* – целочисленный тип данных; *real* – дробные числа; *string* – строковый тип данных; *boolean* – логически тип данных

Каждый оператор языка УФО-скрипт заканчивается знаком «;». Фрагменты программного кода заключаются в программные скобки: *begin* … *end*.

Для обработки разветвляющихся алгоритмов исполнения функций УФО-элементов (методов узловых объектов) УФО-скрипт содержит оператор условия, синтаксис которого имеет вид:

if (условие)
then
begin
//набор операторов № 1
end;
else
begin
//набор операторов № 1
end;

Оператор условия языка УФО-скрипт работает, как и в других языках программирования: если условие принимает истинное значение – выполняется фрагмент программы №1, иначе, если условие принимает ложное значение, выполняется фрагмент программы № 2.

Для реализации циклических алгоритмов язык УФО-скрипт содержит два вида организации циклов. Первый - цикл со счетчиком, синтаксис которого показан ниже:

for i:=<начальное значение счетчика> *to* <конечное значение счетчика> *do*
begin
<*оператор* 1>;
<*оператор* 2>;
…
<*оператор* n>;
end;

Кроме приведенного выше примера, так же имеется возможность работать с циклом с условием:

while <условие выполнение цикла> *do*
begin
<*оператор* 1>;
<*оператор* 2>;
…
<*оператор* n>;
end;

Приведенный выше цикл будет выполняться пока истинно условие выполнения цикла, поэтому в теле цикла необходимо предусмотреть условия выхода из него.

Кроме стандартных конструкций, УФО-скрипт имеет ряд предопределенных процедур и функций, предназначенных для имитации работы функциональных узлов. Для получения значений входных потоков узла, которому принадлежит функция, описываемая с помощью УФО-скрипта, были разработаны предопределенные функции:

getlinki(‘имя входящего потокового объекта’) – входной поток типа integer;

getlinkr(‘имя входящего потокового объекта’) – входной поток типа real;

getlinkb(‘имя входящего потокового объекта’) – входной поток типа boolean;

getlinks(‘имя входящего потокового объекта’) – входной поток типа string.

После выполнения необходимых действий, необходимо задать значения выходных потоковых объектов, для этого были разработаны следующие процедуры:

setlinki(‘имя потокового объекта’, значение) – выходящий поток типа integer;

setlinkr(‘имя выходящего потокового объекта’, значение) – выходящий поток типа real;

setlinkb(‘имя выходящего потокового объекта’, значение) – выходящий поток типа boolean;

setlinks(‘имя выходящего потокового объекта’, значение) – выходящий поток типа string.

Приведенные конструкции языка описания функциональных узлов позволяют в полной мере смоделировать функционирование процесса.

3. Процедура системно-объектного имитационного моделирования

Рассмотрим процедуру создания имитационной модели системы с применением новой версии программного инструментария. Согласно УФО-подхода, перед разработкой модели системы, определяется иерархия потоковых объектов, которые в дальнейшем будут использоваться в модели. При этом необходимо указать свойства потоковых объектов, которые будут исследоваться в процессе имитации функционирования моделируемой системы. После описания иерархии потоковых объектов, создается модель системы, представляющая собой набор УФО-элементов, которые соединяются потоковыми объектами, т.е. создается спецификация моделируемого процесса. После чего, для каждого УФО-элемента определяется его функция, т.е. описывается, как его входящие потоки преобразуются в выходящие из него.

Интерфейс нового программного инструмента на данном этапе разработки показан на Рис. 1, где его рабочая область состоит из трех окон. Верхнее левое окно показывает иерархию потоковых объектов моделируемой системы. Нижнее левое окно показывает иерархию узловых объектов, расположенных на диаграмме в правом окне. После создания иерархии потоковых объектов имеется возможность создания узлов и соединяющих их потоков с помощью

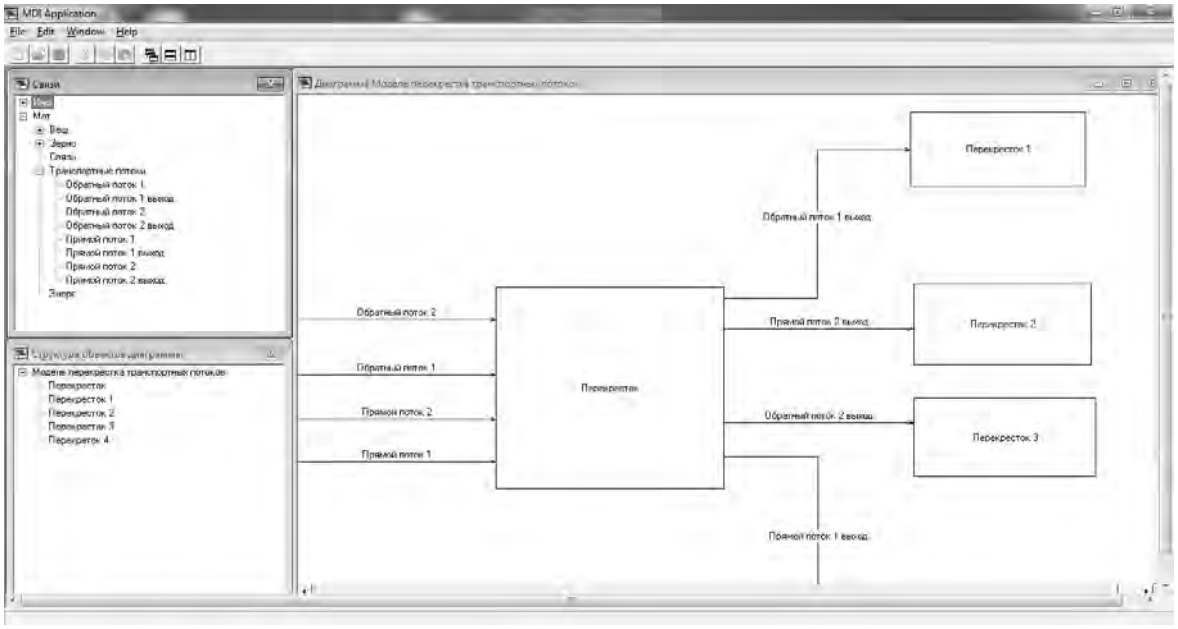

Рис. 1. Интерфейс новой версии системно-объектного инструментария

редактора УФО-диаграмм, а также задания функции каждому узлу. После задания функции УФО-элемента, ее можно описать с помощью УФО-диаграммы более низкого уровня, если имеет место сложная процедура преобразования входных потоковых объектов в выходные. Иначе имеется возможность описать работу функции с помощью языка УФО-скрипт. Функция УФО-элемента описывается скриптом в том случае, если достигнут необходимый уровень декомпозиции (зависит от целей моделирования) и если она представляет собой набор достаточно простых команд.

Для использования созданной имитационной модели системы необходимо задать начальные параметры модели, которые определяются входными потоковыми объектами контекстной модели системы. После задания начальных параметров, модель запускается на исполнение. При запуске имитации, построенная модель транслируется в программу, которая выполняется по следующей схеме: исполняется функция того узла, потоковые объекты которого имеют конкретные значения своих параметров.

Для имитации функционирования процесса в программном пакете имеется отдельный модуль, который позволяет запустить модель, настроить параметры модели, приостановить работу модели, визуализировать результаты имитационного моделирования.

4. Тестовые примеры

Рассмотрим в качестве примера имитационного моделирования технологического процесса сборку станка. Допустим, что она осуществляется путем сборки корпуса станка и его механизмов, которые, в свою очередь, производятся из специальных заготовок. Причем, заранее известно, что производство корпуса станка занимает 2 единицы времени, производство механизма станка - 6 единиц, а сборка станка - 1 единицу. Для построения имитационной модели создадим потоковые объекты, которые будут участвовать в описанном процессе: заготовка для корпуса - заготовка для механизма - корпус танка - механизм станка - станок.

Все перечисленные потоковые объекты будут иметь одинаковую структуру – содержать свойство «Количество», так как в данном примере нас интересуют количественные характеристика моделируемого процесса. Далее с помощью редактора УФО-диаграмм создаем узлы, соответствующие производству механизма станка, производству корпуса станка и сборке станка. Внешний вид УФО-модели процесса производства станков показан на Рис.2.

На следующем этапе необходимо создать функции узлов, которые отражают функционирование подпроцессов, и описать их работу с помощью языка УФО-скрипт. Скрипт-функции

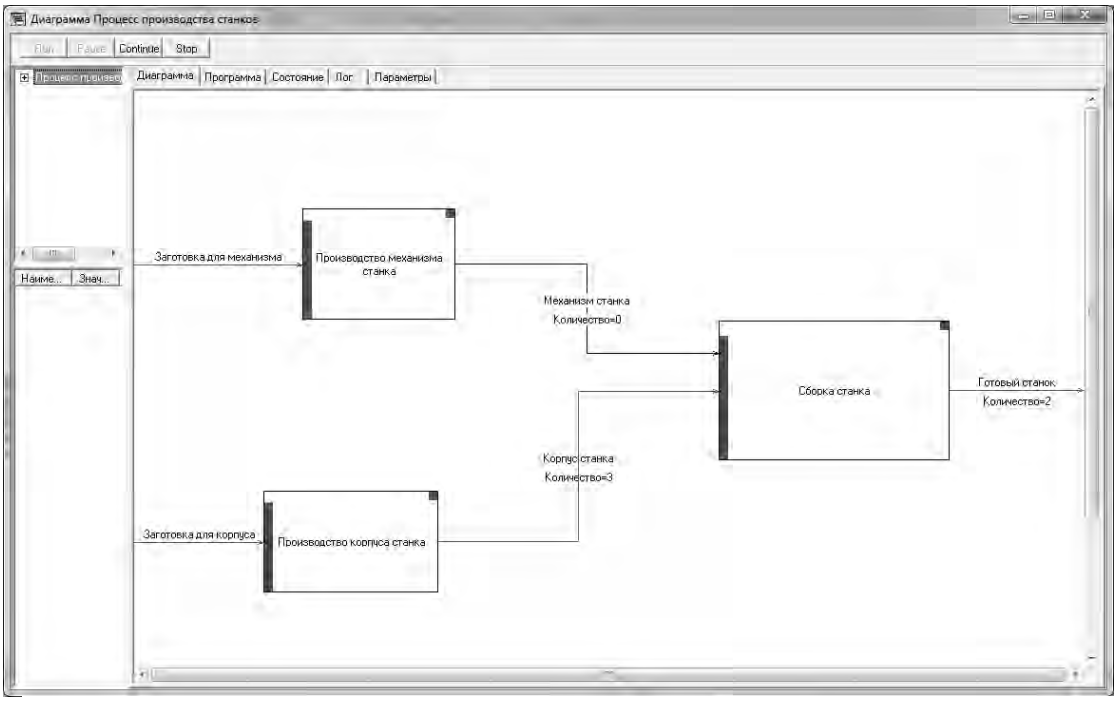

Рис. 2. УФО-модель процесса производства станков

производства механизмов станка представлен ниже:

```
begin
 while getlinki('Заготовка для механизма.Количество')>0 do
  begin
    SetObjPropB('#active',true);
    setlinki('Заготовка для механизма. Количество', getlinki('Заготовка для механизма. Количество')-1);
    delay(6);
    setlinki('Механизм станка.Количество', getlinki('Механизм станка.Количество')+1);
  end;
  SetObjPropB('#active',false);
end.
```

Логика работы скрипта состоит в следующем. В первую очередь проверяется входной потоковый объект на наличие заготовок для производства механизмов. Доступ к значениям свойств входных интерфейсных потоковых объектов текущего узла осуществляется с помощью предопределенной функции *getlinki*, параметром которой является определенное свойство потокового объекта. После проверки наличия заготовок для производства механизмов, свойство *active* текущего узла устанавливается в значение *true*, что говорит об активации текущего узла.

Далее с помощью предопределенной функции *setlinki* значение свойства «*Заготовка для механизма. Количество*» уменьшается на единицу и запускается временная задержка с помощью функции *delay*. Параметром функции является время задержки в единицах времени. После организации временной задержки, значение свойства «*Механизм станка. Количество*» увеличивается на единицу. Это повторяется до тех пор, пока на входе рассматриваемого процесса производства механизмов станков имеются заготовки для их производства. Таким же образом организована функция производства корпусов станка, единственное отличие - во времени задержки.

Скрипт описания функции сборки станков:

```
begin
while 1>0 do
begin
 while (getlinki('Механизм станка. Количество')>0) and (getlinki('Корпус станка. Количество')>0) do
  begin
    SetObjPropB('#active',true);
    setlinki('Механизм станка.Количество', getlinki('Механизм станка.Количество')-1);
    setlinki('Корпус станка.Количество', getlinki('Корпус станка.Количество')-1);
    delay(1);
    setlinki('Готовый станок.Количество', getlinki('Готовый станок.Количество')+1);
  end;
end;
  SetObjPropB('#active',false);
end.
```

Логика работы представленного скрипта такая же, как и в случае с производством механизмов, отличие состоит лишь в том, что в данном случае проверяются и изменяются два потоковых объекта. То есть, сборка станка осуществляется в том случае, если имеется в наличии хотя бы один корпус станка и его механизм.

После описания функций узлов модели, можно приступить к имитации моделируемого процесса. Для этого построенная модель транслируется в исполняемый код, открывающийся в отдельном окне имитации, в котором необходимо задать основные параметры имитации. Программа предлагает следующие параметры имитации:

- масштаб времени формата [t1:t2] – где t1 – единица модельного времени, t2 – единица реального времени;
- время интегрирования Δt – определяет промежуток времени, относительно которого оценивается загрузка потокового и узлового объекта;
- загрузка узлового и потокового объектов – задается в процентах и определяет цветовые индикаторы загрузки.

В случае, когда нарушается баланс между подпроцессами, т.е. один производит больше чем может обработать следующий, программа выделяет потоковый объект «Корпус станка» красным цветом, что говорит о нарушении баланса между узлами. В соответствии с условием, производство механизмов станка занимает 6 секунд, а производство корпусов станка занимает лишь 2 секунды. Таким образом, из модели можно увидеть, что количество корпусов станка растет, это говорит о том, что процесс работает неэффективно. Решение подобной проблемы может заключаться, как например, в добавлении еще одного звена сборки механизмов станка, или в увеличении скорости сборки механизмов станков.

Увеличив модельное время производства механизмов станков, имитация показывает, что подпроцессы практически сбалансированы, но имеется незначительное увеличение количества корпусов станков, чему свидетельствует показатель «количество» потокового объекта «корпус станка».

Реализованная модель наглядно показывает несбалансированность между подпроцессами, что уже является ценной информацией, так как в случаях, когда технологический процесс состоит из множества подпроцессов, трудно адекватно оценивать его работу «в ручную» и устранять «узкие места».

Кроме моделирования технологических процессов, инструментарий позволяет так же имитировать, например, транспортные потоки. Рассмотрим для примера модель движения транспорта на перекрестке. В соответствии с алгоритмом моделирования в первую очередь необходимо создать иерархию потоковых объектов. Для данной предметной области потоковыми объектами будут являться транспортные потоки, имеющие направление движения и протяженность. Ключевой же характеристикой таких потоков является количество автомобилей в потоке.

Перечень представленных на Рис. 1 (левое верхнее окно) потоковых объектов соответствует транспортным потокам перекрестка двух автомобильных дорог с двумя полосами для движения. Обозначения транспортных потоков имеют следующую логику: прямые потоки 1 и 2 представляют собою левые полосы движения автодороги – входящие потоки в перекресток, прямые потоки 1 и 2 – левые полосы движения автодороги, исходящие из перекрестка. Обратные потоки представляют собой правые полосы движения автодорог. Для удобства на диаграмме поместим узел «Перекресток», у которого с одной стороны обозначим все входящие транспортные потоки, а с другой стороны – исходящие. Также на диаграмме размещены еще четыре узла, представляющие так же транспортные перекрестки. Далее рассмотрим описание функции перекрестка потоков. Функция представляет собою распределение автомобилей из входных транспортных потоков в выходные, в зависимости от времени горения зеленого сигнала светофора. С помощью УФО-скрипта подобное распределение автомобилей описывается следующим образом:

```
while (tg1<60) and (g1) do
  begin
    tg1:=tg1+1;
    delay(1);
    tcar1:=tcar1+1;
    if (tcar1=3) and (getlinki('Прямой поток 1.
Количество')>0)  then
      begin
        setlinki('Прямой поток 1.Количество',
getlinki('Прямой поток 1.Количество')-1);
        setlinki('Прямой поток 1 выход. Ко-
личество', getlinki('Прямой поток 1 выход. Ко-
личество')+1);
        setlinki('Обратный поток 1. Количе-
ство', getlinki('Обратный поток 1.Количество')-1);
        setlinki('Обратный поток 1 выход.
Количество', getlinki('Обратный поток 1 выход.
Количество')+1);
        tcar1:=0;
      end;
  end;
```

В приведенном выше скрипте, переменная tg1 представляет собою счетчик горения зеленого сигнала светофора. Обработка происходит в цикле - одна итерация которого занимает одну секунду. Распределение автомобилей по транспортным потокам осуществляется исходя из того, что среднее время проезда автомобилем перекрестка занимает 3 секунды. Подобным же образом описаны функции остальных узлов диаграммы. Далее запустив модель на исполнение, можно отследить пробки на дороге. Пробка визуально представляет собою выделенный цветом потоковый объект – транспортный поток. Если поток выделен красным цветом – происходит интенсивное увеличение некоторой его характеристики (в нашем случае увеличивается количество автомобилей в потоке). Если поток выделен зеленым цветом – наблюдается сбалансированность между перекрестками, если синим – происходит уменьшение некоторой характеристики потока.

Как видно из Рис. 3, в данный момент времени в потоке с именем «Прямой поток 2 выход» имеет место пробка. Устранить которую можно, например, увеличив соответствующее время горения зеленого сигнала светофора на перекрестке № 2. Таким образом, на модели наглядно видно, как себя поведут транспортные потоки при различных условиях. С помощью скрипта можно учесть также элемент случайности, например, когда поток имеет два направ-

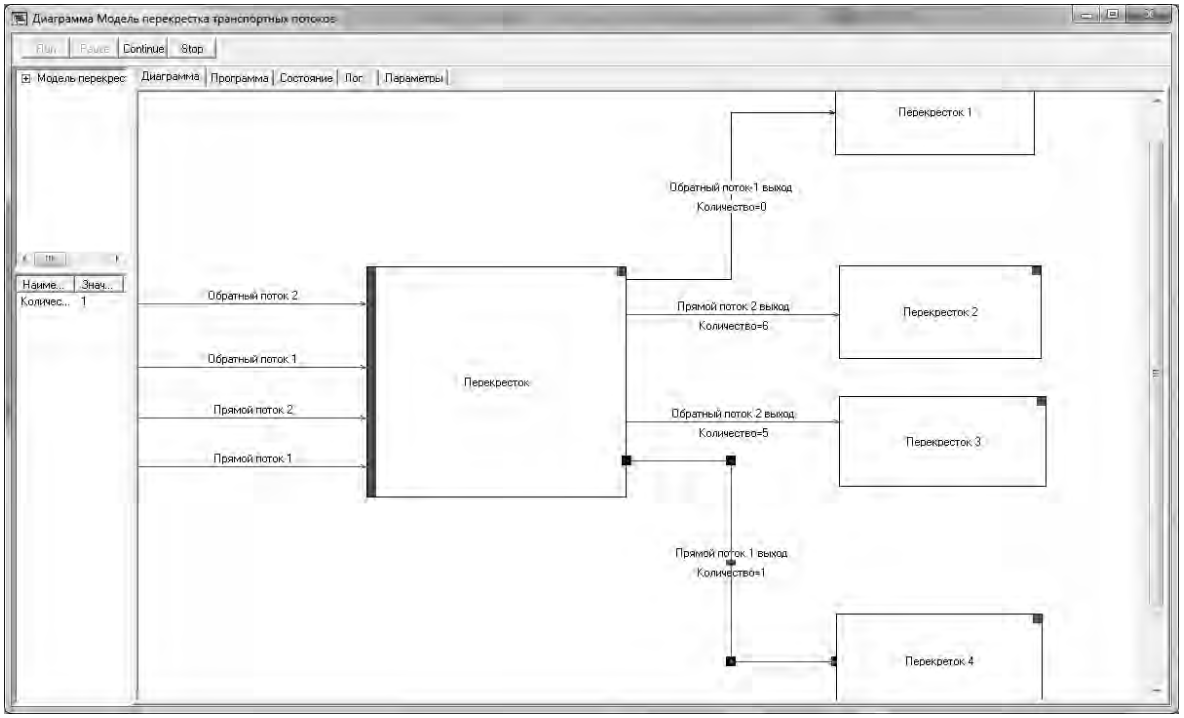

Рис. 3. УФО-модель транспортных потоков

ления движения и многие другие характеристики дорожной ситуации.

Заключение

Описанные выше концептуальные, формальные и лингвистические основы системно-объектного имитационного моделирования показывают, что системный подход «Узел-Функция-Объект» и системно-объектный метод представления знаний позволяют, с одной стороны, создавать графоаналитические модели, с достаточной точностью описывающие реальные системы, а с другой стороны, трансформировать эти модели в алгебраические описания. Получаемые таким образом математические модели могут имитировать функционирование любых систем с необходимой степенью подробности.

Представленные возможности нового программного инструментария для имитационного моделирования свидетельствуют об универсальности предложенного метода и программного средства. Эти возможности обусловлены использованием в ходе моделирования системного УФО-подхода и системно-объектного метода представления знаний. Универсальность достигается также за счет описания функциональных узлов с помощью УФО-скрипта, что, в некоторой степени, увеличивает трудоемкость моделирования, но, при этом, обеспечивает гибкость подхода и возможность моделировать процессы любой природы с любой точностью.

Ближайшей задачей разработчиков в плане развития и совершенствования предлагаемого инструментария является создание на языке УФО-скрипт библиотек часто используемых модельных элементов для различных предметных областей.

Литература

1. Национальное общество имитационного моделирования России – начало пути / Интервью Р.М. Юсупова, члена-корреспондента РАН, директора СПИИРАН редакции журнала CAD/CAM/CAE Observer // CAD/CAM/CAE Observer. – 2012. - №2 (70).- С.10-18.
2. Имитационное моделирование [Электронный ресурс] // URL: (https://ru.wikipedia.org/wiki/Имитационное_ моделирование.
3. Узел-Функция-Объект [Электронный ресурс] // URL: http://ru.wikipedia.org/wiki/Узел-Функция-Объект.
4. Жихарев А.Г., Маторин С.И. Метод формализации организационных знаний // Искусственный интеллект и принятие решений. – 2011. - №2. – С. 52-63.
5. Жихарев А.Г., Маторин С.И., Маматов Е.М., Смородина Н.Н. О системно-объектном методе представления организационных знаний // Научные ведомости

Белгородского государственного университета. Сер. История. Политология. Экономика. Информатика. – 2013. – № 8 (151), Вып. 26/1. С. 137-146.
6. Маторин С.И., Жихарев А.Г., Зайцева Н.О., Четвериков С.Н. Системный подход «Узел-Функция-Объект» при моделировании транспортных потоков // Труды ИСА РАН. - 2014. - №1.- Том 64.- С. 9-18.
7. Abadi Martin, Luca Cardelli. A Theory of Objects. - Springer-Verlag. - 1996. – 397p.

Жихарев Александр Геннадиевич. Старший преподаватель кафедры информационных систем управления Белгородского государственного национального исследовательского университета. Кандидат технических наук.
E-mail: zhikharev@bsu.edu.ru

Маторин Сергей Игоревич. Профессор кафедры информационных систем и технологий Белгородского университета кооперации, экономики и права. Доктор технических наук. E-mail: matorin@softconnect.ru

Зайцева Наталья Олеговна. Старший преподаватель кафедры прикладной информатики и информационных технологий Белгородского государственного национального исследовательского университета. E-mail: zaitseva_n_o@bsu.edu.ru