УДК 519.876.5

СИСТЕМНО-ОБЪЕКТНОЕ ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ТРАНСПОРТНЫХ И ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

SYSTEM-OBJECT SIMULATION OF TRANSPORT AND TECHNOLOGICAL PROCESSES

С.И. Маторин, А.Г. Жихарев, Н.О. Зайцева
S.I. Matorin, A. G. Zhikharev, N.O. Zaitseva

Белгородский государственный национальный исследовательский университет, Россия, 308015, Белгород, ул. Победы, 85
Belgorod State National Research University, 85 Pobeda St, Belgorod, 308015, Russia

e-mail: matorin@bsu.edu.ru, zhikharev@bsu.edu.ru, zaitseva_n_o@bsu.edu.ru

Аннотация. В статье рассматриваются возможности системно-объектного подхода «Узел-Функция-Объект» в области имитационного моделирования транспортных и технологических процессов за счет использования формального аппарата исчисления объектов и языка моделирования производственных процессов «Chi».

Resume. The possibility of system-object approach "Unit-Function Objects" in the field of simulation of transport and technological processes through the use of the formal apparatus of the objects calculus and modeling language productions «Chi».

Ключевые слова: имитационное моделирование, транспортный поток, технологический процесс, исчисление объектов, язык моделирования производственных процессов, системно-объектный подход «Узел-Функция-Объект»

Keywords: simulation, transport stream, technological process, object calculation, modeling language production processes, system-object approach " Unit-Function Objects"

Введение

Исторически в имитационном моделировании сложились три основные парадигмы:

– *Системная динамика* (Дж. Форрестер 50-е годы прошлого века) — парадигма моделирования, в рамках которой для исследуемой системы строятся диаграммы причинных связей и глобальных влияний одних параметров на другие во времени, а затем на основе этих диаграмм создается алгоритмическая модель, которая и проигрывается на компьютере.

– *Дискретно-событийное моделирование* (Дж. Гордон 60-е годы прошлого века) — парадигма моделирования, в рамках которой структура моделируемой системы адекватно отображается в модели, а процессы ее функционирования проигрываются (имитируются) на построенной модели. Предлагается так же абстрагироваться от непрерывной природы событий и рассматривать только основные события моделируемой системы, такие, как: «ожидание», «обработка заказа», «движение с грузом», «разгрузка» и другие. Дискретно-событийное моделирование наиболее развито и имеет огромную сферу приложений — от логистики и систем массового обслуживания до транспортных и производственных систем.

– *Агентное моделирование* (90-е - 2000-е годы) - парадигма моделирования, в рамках которой исследуются децентрализованные системы, динамика функционирования которых определяется не глобальными правилами и законами (как в других парадигмах моделирования), а наоборот, когда эти глобальные правила и законы являются результатом индивидуальной активности членов группы.

При этом одно из определений имитационного моделирования подчеркивает, что это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, с которой проводятся эксперименты с целью получения информации об этой системе. В рамках данного определения наиболее адекватной представляется дискретно-событийная парадигма моделирования. В рамках же этой парадигмы наиболее эффективным будет метод моделирования, который позволяет наиболее полно описывать структуру, процессы функционирования и другие свойства моделируемого объекта или системы. В качестве такого метода авторы предлагают рассматривать метод системно-объектного моделирования, в основе которого лежит оригинальный системный (системно-объектный) подход «Узел-Функция-Объект» (**УФО-подход:** http://ru.wikipedia.org/ wiki/Узел-Функция-Объект) Развитием и формализацией УФО-подхода является метод и алгоритм системного анализа, именуемый для краткости **УФО-анализом**. В целях автоматизации применения УФО–анализа спроектирован и

реализован CASE-инструментарий **UFO-toolkit** (свидетельство Роспатента №2006612046, http://www.ufo-toolkit.ru/) [1, 2].

Существование большого множества программных систем имитационного моделирования, основанных на самых разных теориях и подходах, свидетельствует о применимости и востребованности имитационных моделей. Однако кроме того, это свидетельствует о существовании нерешенных до сих пор проблем в области имитационного моделирования, что выражается, в частности, в сложности оценки адекватности описания моделируемой системы и интерпретации результатов. Отдельные недостатки имитационного моделирования описаны, например, в [3]:

1. Разработка хорошей имитационной модели часто обходится дорого и требует больших затрат времени.

2. Иногда кажется, что имитационная модель отражает реальное положение вещей, хотя в действительности это не так. Если не учитывать этого, то некоторые свойственные имитации особенности могут привести к неверному решению.

3. Имитационная модель в принципе неточна, и нет возможности измерить степень этой неточности. Это затруднение может быть преодолено лишь частично путем анализа чувствительности модели к изменению определенных параметров.

4. На имитационной модели можно получить ответ только после очередного имитационного эксперимента и возможности прогнозирования имитационного моделирования значительно меньше, чем аналитического моделирования.

Сказанное выше позволяет говорить об актуальности исследований в данной области, причем, в первую очередь, с точки зрения решения задачи преобразования компьютерных визуальных (графоаналитических) системно-объектных моделей в терминах «Узел-Функция-Объект» (**УФО-моделей**) в модели имитационные.

Учитывая, что имитационное моделирование — это частный случай математического моделирования объектов, для которых по различным причинам не разработаны аналитические модели, либо не разработаны методы решения таких моделей, необходимо для создания имитационной модели на основе УФО-модели осуществить формализацию последней, т.е. ее трансформацию в математическое (алгебраическое) описание.

Рассмотрим два варианта такой трансформации.

Имитационное моделирование путем формализации УФО-моделей средствами исчисления объектов

Для формального описания элементов «Узел-Функция-Объект» системно-объектных моделей (**УФО-элементов**) может быть использован алгебраический аппарат *исчисления объектов* Абади-Кардели [4], разработанный для формализации объектно-ориентированного программирования. При этом УФО-элементу ставится в соответствие специальный класс объектов исчисления Абади-Кардели, который содержит специально выделенные поля и методы (**узловой объект**), как показано в приведенном ниже выражении.

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

Кроме того, самими связям/потокам ставится в соответствие другой специальный класс абстрактных объектов, который обладает только набором полей, содержащих основные характеристики объекта (**потоковый объект**) , как показано в приведенном ниже выражении.

$a_i = [l_j = b_j]$, где:

– a_i – потоковый объект;

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

Если для хранения и обработки знаний о каких-либо процессах представлять их в виде УФО-элементов, то, с учетом формального их описания средствами исчисления объектов, манипулирование этими знаниями, в частности имитацию динамики процессов, можно обеспечить путем организации цепочки вызовов методов узловых объектов со стороны соответствующих потоковых объектов. Цепочка организуется на уровне декомпозиции УФО-элемента. В соответствии с прави-

лами исчисления объектов Абади-Кардели вызов узлового объекта можно записать следующим образом:

$$G.l_n\, l!_j\{l?_i \mid\rightarrow G\}.$$

Подобный вызов метода узлового объекта будет иметь место в том случае, если на вход узлового объекта поступает поток, наименование объектов которого (потоковых) совпадает со значением поля узлового объекта, которое содержит значение входных потоковых объектов. Старт процедуры имитационного моделирования осуществляется путем инициализации некоторого контекстного потокового объекта, после чего значение контекстного потокового объекта попадает в соответствующее поле интерфейсного узлового объекта, после чего вызывается метод этого узлового объекта, который выполнив некоторые действия, вызывает метод следующего узлового объекта и так пока не достигается конец модели. Формально с учетом описания УФО-элементов средствами исчисления объектов упомянутая процедура имитации может быть представлена следующим образом:

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

Вызов метода узлового объекта осуществляется за счет «движка» модели, который использует скриптовый язык, включающий в свой состав следующие операторы:

– оператор создания переменной;

– оператор присваивания значения переменной;

– операторы условия и цикла (в классической форме организации);

– математические операторы сложения, вычитания, умножения и деления;

– логические операторы.

Таким образом, процедура системно-объектного имитационного моделирования при условии формализации УФО-подхода с помощью алгебраического аппарата исчисления объектов, будет организована в соответствии с рисунком 1. Как видно из рисунка 1, организация имитации функционирования системы, представляемой в виде УФО-элемента, имеет, как и сам УФО-элемент, три уровня представления. Первый уровень – визуальное представление системы (то, что видит пользователь на экране монитора). На данном уровне модель системы строится из отдельных узлов системы, связанных между собою некоторыми потоковыми объектами X, Y, Z, L. Второй уровень - описание функционирования системы с помощью операторов скриптового языка, причем здесь имеют место предопределенные методы:

– getLink(X.x1, X.x2,...,Y.y1, Y.y2,...) – позволяет получить значения характеристик входных потоковых объектов;

– setLink(Z.z1, Z.z2,...,L.l1, L.l2,...) – позволяет задать значения характеристикам выходных потоковых объектов;

– GetObjectParam(P1,P2,...) – позволяет получить значения характеристик узлового объекта, что соответствует третьему уровню описания системы – объектный уровень.

Таким образом, существует возможность описать функцию системы либо с помощью дальнейшей декомпозиции ее на подсистемы более низкого уровня, либо, если необходимый уровень декомпозиции достигнут, с помощью конструкций некоторого скриптового языка. Фактически получим организованную имитационную модель системы, состоящую из отдельных исполняемых блоков программы

Исходя из вышесказанного, описать функционирование процесса возможно с помощью скриптового языка. Авторами данной стать был разработан язык УФО-скрипт, который представляет собою язык описания функциональных узлов. Рассмотрим основные конструкции данного языка. Синтаксически язык УФО-скрипт схож с языком программирования Pascal.

Для объявления переменных, используемых в рамках описания функционирования конкретного подпроцесса используется следующее выражение:

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

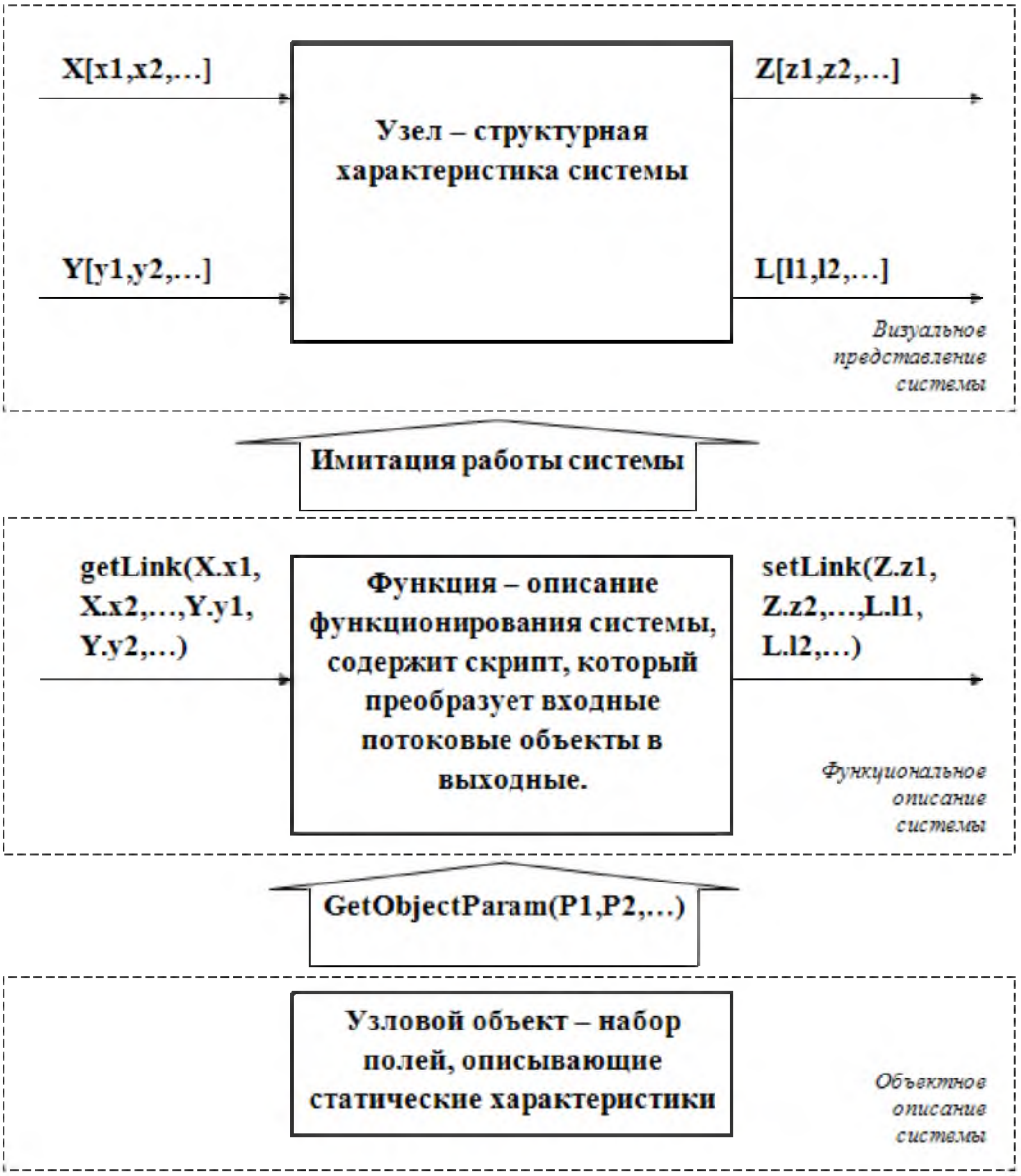

Рис. 1. Структура имитации функционирования системы, представляемой с помощью УФО-элементов и исчисления объектов

Fig. 1. The structure of the simulation of the system, provided by a NFO-elements and theory of objects

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** – представляет собою дробные числа;
– **string** – строковый тип данных;
– **boolean** – логически тип данных

Каждый оператор языка УФО-скрипт заканчивается знаком «;». Фрагменты программного кода заключаются в программные скобки, как показано на примере ниже:

```
begin
…
end
```

Для обработки разветвляющихся алгоритмов исполнения функциональных узлов, УФО-скрипт содержит оператор условия, синтаксис которого показан ниже:

```
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 выходя из цикла, иначе программа зациклится.

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

Информационные потоки входа обозначаются следующим образом:

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.

Как было отмечено выше, узловой объект может содержать некоторые показатели или характеристики узлового объекта. Фактически, это константы для программы имитирующей работу функционального узла. Для получения значений характеристик узлового объекта и использования их в программе, реализована процедура **getobjpropi('имя свойства узлового объекта').** Еще одной предопределенной процедурой, которая имеет важное значение в процессе разработки имитационных моделей процессов является процедура **delay(time).** Эта процедура представляет собою временную задержку, которая имитирует, например, работу некоторого механизма или агрегата. Параметром данной процедуры является время задержки, которое будет измеряться в условных единицах, зависящих от масштаба времени имитационной модели. Рассмотрим небольшой пример описания функции узла с входящим потоком **l.x** и выходящим потоком **l.y**, как показано на рисунке ниже:

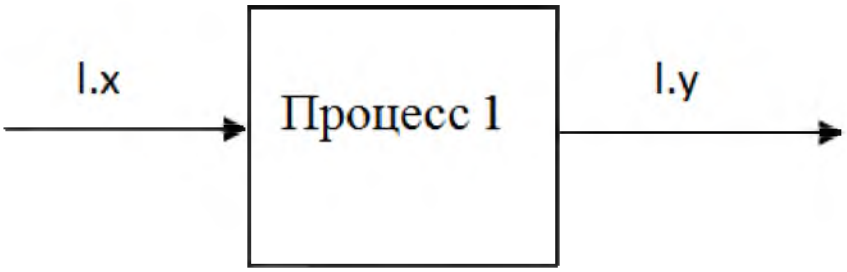

Рис. 2. Пример абстрактного процесса
Fig. 2. Example of an abstract process

```
var
 i,a,n,k,r:integer;
begin
 k:=getobjpropi('k1'); // получение значения свойства узлового объекта k1
 r:=getobjpropi('k2'); // получение значения свойства узлового объекта k2
 a:=0;
 delay(1); // задержка времени в 1 секунду
 while getlinki('l.x')>=k do
 begin
  setlinki('l.x',getlinki('l.x')-k);
  a:=1;
  delay(1);
  setlinki('l.y',getlinki('l.y')+r);
  delay(1);
 end;
 a:=0;
 delay(3);
end.
```

Далее рассмотрим описание уже не абстрактного примера, а описание функционирования хлебопечи. Которая позволяет выпекать 20 буханок хлеба одновременно и выпекает их в течении 10 минут. Возьмем за условие, что модель реализуется в минутном масштабе времени. Хлебопечь имеет два ключевых показателя работы, которые в модели отражаются как характеристики узлового объекта, назовем их: count=20 – максимальная загрузка хлебопечи; work_time=10 – время выпечки одной партии в минутах. Входящим потоком является конвеер, по которому поступают сырые буханки, назовем его потоком input со свойством потокового объекта count представляющим количество сырых буханок. Выходящий поток так же представляет собою конвеер, но уже с готовыми буханками, которые дальше поступают на упаковку, обозначим его output со свойством count. Примем так же за условие, что хлебопечь начинает работать, только в слвучае полной заполненности, т.е. когда input.count=20. Описание работы хлебопечи с помощью УФО-скрипта будет иметь следующий вид:

```
var
begin
if (getlinki('input.count') = getobjpropi('count')) then
 begin
    setlinki('input.count',0);
   delay(getobjpropi('work_time'));
   setlinki('output.count',20);
end;
end.
```

В приведенном выше фрагменте описания работы хлебопечи, сперва, проверяется условие наполненности печи, если печь заполнена, тогда обнуляется значение входного потока и запускается временная задержка, которая имитирует выпечку одной партии в течении 10 минут, после чего выходящий поток пополняется 20 готовыми буханками хлеба.

Представленный язык описания функциональных узлов является универсальным и позволяет описывать не только работу технологических процессов, но так же и других, например транс-

портных. Рассмотрим небольшой пример имитации движения автомобилей через перекресток, как показано на рисунке 3.

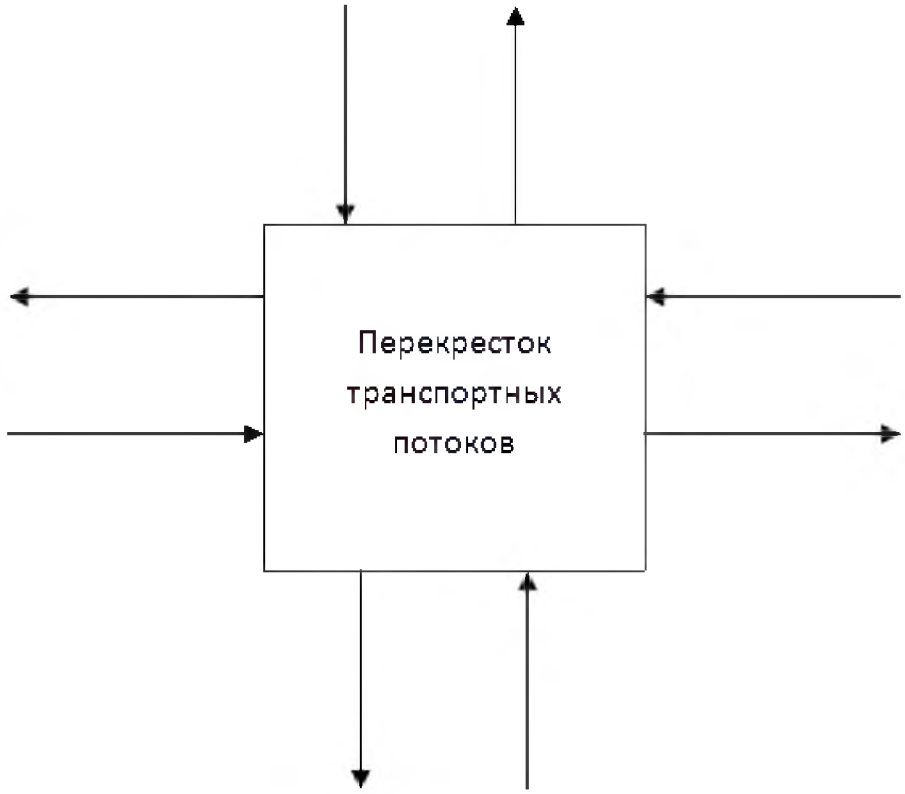

Рис. 3. Схема направлений транспортных потоков на перекрестке с двухсторонним движением
Fig. 3. Driving directions of traffic at the intersection of a two-way

Обозначив входящие и исходящие транспортные потоки именами, например input_left – входящий поток справа относительно перекрестка, так же необходимо добавить потокам свойства, где будет храниться количество автомобилей в транспортном потоке в текущий момент времени. Для учета характеристик перекрестка, таких как, например: пропускная способность, сигнал светофора и т.п. необходимо создать соответствующие свойства узлового объекта. После чего с помощью языка УФО-скрипт, можно описать процесс пересечения перекрестка автомобилями. Для этого необходимо уменьшать количество автомобилей во входящем потоке и увеличивать количество автомобилей в исходящем потоке. Это реализуется с помощью функций **setlinki** и **getlinki**.

```
var
 i,a,n,k,r,e:integer;
begin
 k:=getobjpropi('k1');
 r:=getobjpropi('k2');
 e:=getobjpropi('k3');
 a:=0;
 while getlinki('l.y')<k do
  delay(2);
 while getlinki('l.y')>=k do
 begin
  setlinki('l.y',getlinki('l.y')-k);
  a:=1;
  delay(3);
  setlinki('l.z',getlinki('l.z')+r);
  setlinki('l.e',getlinki('l.e')+e);
  delay(3);
 end;
 a:=0;
```

delay(3);
end.

Так же имеется возможность смоделировать различные алгоритмы включения и отключения сигналов светофора. Поэтому мы можем говорить об универсальности рассматриваемого подхода к имитации процессов.

Имитационное моделирование путем преобразования УФО-моделей в конструкции языка «Chi» (χ)

Для формального описания элементов «Узел-Функция-Объект» системно-объектных моделей (**УФО-элементов**) может быть использован алгебраический аппарат, полученный путем интеграции *теории паттернов* Гренандера [5] и *исчисления процессов* Милнера (в варианте CCS - *Calculus of communication systems*) [6]. В результате получено алгебраическое представление системы $\mathbf{s_i}$ как УФО-элемента [7]:

$\mathbf{s_i} = \mathbf{<(L?_i, L!_i), (P_i, P^o_i, L\tau_i), (n_i, \alpha_i, \beta?_i, \beta!_i)>}$, где

– $\mathbf{(L?_i, L!_i)}$ – «Узел: **U**» УФО-элемента, $\mathbf{L?_i \subset L}$ – множество входных связей, $\mathbf{L!_i \subset L}$ – множество выходных связей;

– $\mathbf{(P_i, P^o_i, L\tau_i)}$ – «Функция: **F**» УФО-элемента, где $\mathbf{P_i}$ – множество подпроцессов процесса, соответствующего «Функции», которые реализуются УФО-элементами нижнего яруса иерархии; $\mathbf{P^o_i \subset P_i}$ – множество интерфейсных подпроцессов (входных $\mathbf{P?_i}$ и выходных $\mathbf{P!_i}$, причем $\mathbf{P^o_i = P?_i \cup P!_i}$; в число входных связей $\mathbf{P?_i}$ входит $\mathbf{L?_i}$, в число выходных связей $\mathbf{P!_i}$ входит $\mathbf{L!_i}$); $\mathbf{L\tau_i}$ – множество внутренних связей/переходов в $\mathbf{P_i}$, осуществляемых путем передачи, ввода и вывода элементов глубинного яруса связанных подпроцессов;

– $\mathbf{(n_i, \alpha_i, \beta?_i, \beta!_i)}$ – «Объект: **O**» УФО-элемента, где $\mathbf{n_i}$ – имя «Объекта» ($\mathbf{n_i \in N}$); $\mathbf{\alpha_i}$ – множество признаков «Объекта» $\mathbf{n_i}$; $\mathbf{\beta?_i}$ – множество показателей $\mathbf{L?_i}$; $\mathbf{\beta!_i}$ – множество показателей $\mathbf{L!_i}$.

Для функции УФО-элемента, кроме того, с учетом *исчисления процессов* Милнера (в варианте *пи-исчисления*) [8], в работе [9] получено следующее выражение:

$\mathbf{F = <c(x).P, \tau_p, t(y).P>}$, где

– $\mathbf{c(x)}$ - входной префикс, получение данных $\boldsymbol{x}$ из канала $\boldsymbol{c}$ процессом P;

– $\mathbf{\tau_p}$ - внутренние действия процесса **P**, соответствующего функции **F**;

– $\mathbf{t(y)}$ - выходной префикс, передача данных $\boldsymbol{y}$ по каналу $\boldsymbol{t}$ процессом **P**.

Модифицируя алгебраическое представление системы $\mathbf{s_i}$ как УФО-элемента с учетом последнего выражения для функции этого элемента средствами пи-исчисления, получаем следующее выражение:

$\mathbf{s_i = <(L?_i, L!_i), ((L?_i(\beta?_i).P_i), \tau_p, (L!_i(\beta!_i).P_i)), (n_i, \alpha_i, \beta?_i, \beta!_i)>}$,

или по правилам пи-исчисления:

$\mathbf{s_i = <(L?_i, L!_i), (L?_i(\beta?_i).\tau_p.L!_i(\beta!_i)), (n_i, \alpha_i, \beta?_i, \beta!_i)>}$, где

– $\mathbf{L?_i(\beta?_i)}$ - входной префикс;

– $\mathbf{L!_i(\beta!_i)}$ - выходной префикс процесса **P**.

Алгебраическое представление УФО-элементов (с учетом описания функции элемента средствами пи-исчисления) может быть преобразовано в описание на языке моделирования производственных процессов «CHI» (χ). Возможность такого преобразования обусловлена тем, что пи-исчисление использовано для определения формальной семантики языка χ, хотя базовая версия этого языка появилась более двадцати лет назад (в 1982 году), т.е. раньше исчисления процессов [10, 11]. Один из вариантов методики данного преобразования представлен в таблице.

Язык χ расширяет пи-исчисление процессов понятиями времени, данных и типов данных, а также генератором случайных чисел. Время в языке χ задается следующим образом [10, 12]:

1. $\mathbf{skip = \emptyset : true} >> \tau$ (если процесс происходит мгновенно, т.е. время равно нулю).

Например: skip = ∅ : true >>'результат'.

2. $\mathbf{time = T : true} >> \tau$ (если процесс ограничен по времени).

Например: time = 5 : true >> 'ответ'.

Следовательно, при описании графоаналитической УФО-модели на языке χ появляется возможность "внедрить" в нее временную характеристику и преобразовать, таким образом, визуальную модель процесса в модель имитационную. Естественно, подобная трансформация модели из одного формального (алгебраического) описания в другое (на формальном языке программирования) должна иметь четкую алгоритмическую базу.

Для проведения имитации на языке χ создан специальный "движок", который представляет собой программную среду, реализующую одновременное выполнение нескольких процессов

[13]. Суть его работы состоит в следующем: в фазе генерации кода, компилятор переводит модель на язык программирования высокого уровня. Этот код сравнивается с оригинальной моделью. Каждая конструкция языка χ отображается на соответствующую конструкцию языка программирования. Общая функциональность процесса определяется с помощью абстрактного класса. В сгенерированном коде для каждого процесса χ-модели ставится в соответствие специфический процесс абстрактного класса. Каждому типу данных χ-модели ставится в соответствие тип данных языка программирования. Каналы реализованы в виде экземпляров класса. Классы, которые представляют собой каналы передачи данных, генерируются из шаблона. Этот шаблон в качестве параметра имеет тип данных канала и, следовательно, может быть сформирован класс канала с различными типами данных. Экземпляры классов формируются таким же образом.

Таблица
Table

Методика преобразования УФО-модели в конструкции языка χ.
Methods of converting NFO-model design language

Наименование элементов УФО-модели	Алгебраическое описание УФО- элементов	Описание УФО- элементов на языке χ
Узел (перекресток входяших и выходящих связей)	**$(L?_i, L!_i)$**	**⟨chan {$L?_i$, $L!_i$}** ⟩ где chan { $L?_i$, $L!_i$} – множество входных и выходных каналов
Функция (процесс преобразования входа в выход)	**$(L?_i(\beta?_i).\tau_p.L!_i(\beta!_i))$**	**⟨disc\cont\alg $\beta?_i$, $\beta!_i$, chan {$L?_i$, $L!_i$}, j,** | τ_p ⟩ где disc – если показатели $\beta?_i$, $\beta!_i$ дискретные, cont – если показатели $\beta?_i$, $\beta!_i$ непрерывные, alg – если показатели $\beta?_i$, $\beta!_i$ алгебраические, j – ограничение допустимых значений переменных τ_p – процесс преобразования входных данных в выходные
Объект (структура, реализуюшая функцию и занимаюшая данный узел)	**$(n_i, \alpha_i, \beta?_i, \beta!_i)$**	**⟨object n_i = [disc\cont\alg $\beta?_i$, $\beta!_i$, chan {$L?_i$, $L!_i$}, j, α_i** | τ_p **]** ⟩

В результате создается исполняемый файл с расширением *.exe, при запуске которого на экран выводится модель. В файле с расширением *.log хранятся результаты моделирования. Этот файл содержит время работы каждого элемента модели и имена переменных, задействованных в том или ином процессе; он может быть использован для построения диаграммы.

Также генерируется файл трассировки, с помощью которого можно восстановить процесс моделирования при сбое. Он хранит в себе информацию о состоянии работы системы, событиях на определенный момент времени. Данный файл не является обязательным.

Выводы

Представленные в данной работе результаты позволяют говорить о возможности и целесообразности создания средств имитационного моделирования, использующих в своей основе подход «Узел-Функция-Объект».

Описанный способ системно-объектного моделирования знаний о транспортных и производственно-технологических процессах позволяет в удобном визуальном виде автоматически получать цепочки процессов для различных конкретных ситуаций. Используя такую модель, пользователь не только сможет обеспечить автоматизированную поддержку принятия решений по управлению процессами, но так же накапливать и хранить опыт в виде удачных решений и использовать его в дальнейшем. Процесс имитационного моделирования на основе системно-объектной модели знаний и ее формализации средствами исчисления объектов состоит из следующих этапов:

Создание иерархии связей, т.е. потоковых объектов системно-объектной модели, у которых определяются, важные для данной предметной области, параметры.

Создание визуальной УФО-модели обработки потоковых объектов, на которой отображаются все узлы, ветвления и т.п.

Описание узловых объекты с их параметрами и методами алгебраическими средствами исчисления объектов. При необходимости метод узлового объекта декомпозируется на подпроцессы нижнего уровня.

Использование полученной модели путем инициализации начальных значений модели и запуске механизма логического вывода, в результате которого формируется модель поведения системы для текущего конкретного случая.

Представленный способ описания средствами языка χ формализованных графоаналитических УФО-моделей обеспечивает возможность формального решения задачи имитации функционирования производственных, технологических и транспортных процессов на их системно-объектных визуальных моделях. Процесс имитационного моделирования на основе формализованных графоаналитических УФО-моделей с использованием языка χ состоит из следующих этапов:

1. Построение графоаналитической УФО-модели динамической системы.
2. Формальное описание УФО-модели в терминах пи-исчисления.
3. Преобразование алгебраического описания визуальной модели в конструкции языка χ.
4. Проведение имитации моделируемого процесса с помощью "движка" языка χ.

В настоящее время разрабатывается новая версия программного инструментария UFO-toolkit, в которой будут автоматизированы все описанные возможности системно-объектного моделирования.

Работа выполнена при поддержке грантов РФФИ 13-07-00096 а, 13-07-12000 офи_м, 14-47-08003.

Список литературы
References

1. Маторин С.И., Попов А.С., Маторин В.С. Моделирование организационных систем в свете нового подхода «Узел-Функция-Объект» // Научно-техническая информация. Сер.2. 2005. N1.

Matorin S.I., Popov A.S., Matorin V.S. Modelirovanie organizacionnyh sistem v svete novogo podhoda «Uzel-Funkcija-Ob'ekt» // Nauchno-tehnicheskaja informacija. Ser.2. 2005. N1.

2. Маторин С.И., Зимовец О.А., Жихарев А.Г. О развитии технологии графоаналитического моделирования бизнеса с использованием системного подхода «Узел-Функция-Объект» // Научно-техническая информация. Сер. 2. 2007. №11.

Matorin S.I., Zimovec O.A., Zhiharev A.G. O razvitii tehnologii grafoanaliticheskogo modelirovanija biznesa s ispol'zovaniem sistemnogo podhoda «Uzel-Funkcija-Ob'ekt» // Nauchno-tehnicheskaja informacija. Ser. 2. 2007. №11.

3. Имитационное моделирование [Электронный ресурс] / URL: www.e-sab.narod.ru/Student/msu/4-imitation.pdf

Imitacionnoe modelirovanie [Jelektronnyj resurs] / URL: www.e-sab.narod.ru/Student/msu/4-imitation.pdf

4. Abadi Martin and Luca Cardelli A Theory of Objects. Springer-Verlag, 1996.

Abadi Martin and Luca Cardelli A Theory of Objects. Springer-Verlag, 1996.

5. Гренандер У. Лекции по теории образов. 1. Синтез образов. // Пер. с англ. М.: Мир, 1979.

Grenander U. Lekcii po teorii obrazov. 1. Sintez obrazov // Per. s angl. M.: Mir, 1979.

6. Milner R., Parrow J., Walker D.A. Calculus of Mobile Processes – Part I. LFCS Report 89 – 85. University of Edinburgh, 1989.

Milner R., Parrow J., Walker D.A. Calculus of Mobile Processes – Part I. LFCS Report 89 – 85. University of Edinburgh, 1989.

7. Zimovets O., Matorin S. Integration of graphic analytical models "Unit-Function-Object" formalize means // Scientific and Technical Information Processing. - Allerton Press, Inc. 2013. – V.40. – N6.

Zimovets O., Matorin S. Integration of graphic analytical models "Unit-Function-Object" formalize means // Scientific and Technical Information Processing. - Allerton Press, Inc. 2013. – V.40. – N6.

8. Milner R. The poliadic π-calculus: a Tutorial, University of Edinburgh, 1991.

Milner R. The poliadic π-calculus: a Tutorial, University of Edinburgh, 1991.

9. Михелев М.В., Маторин С.И. Формализация УФО элементов с помощью алгебраического аппарата ПИ-исчисления. // Научные ведомости БелГУ. Сер. Информатика. 2010. №19(90). Выпуск 16/1.

Mihelev M.V., Matorin S.I. Formalizacija UFO jelementov s pomoshh'ju algebraicheskogo ap-parata PI-ischislenija. // Nauchnye vedomosti BelGU. Ser. Informatika. 2010. №19(90). Vypusk 16/1.

10. Сынтульский С.С. Язык программирования сверхбыстрых гибридных вычислительных устройств. [Электронный ресурс] // URL: http:// www.math.spbu.ru/user/gran/sb2/ssyntulsky.pdf

Syntul'skij S.S. Jazyk programmirovanija sverhbystryh gibridnyh vychislitel'nyh ustrojstv. [Jelektronnyj resurs] // URL: http:// www.math.spbu.ru/user/gran/sb2/ssyntulsky.pdf

11. Rooda J. E. Vervoort J. Learning χ 0.8. Eindhoven. 2003.

Rooda J. E. Vervoort J. Learning χ 0.8. Eindhoven. 2003.

12. van Beek D.A., Man K.L., Reniers M.A., Rooda J.E., Schiffelers R.R.H. Syntax and consistent equation semantics of hybrid Chi. [Электронный ресурс] // URL: http://ac.els-cdn.com/S1567832605000895/1-s2.0-S1567832605000895-main.pdf?_tid=26231c90-c739-11e4-aa5c-00000aacb35f&acdnat=1426001079_b450fab5616606d5bfad09e7ecac25c6

van Beek D.A., Man K.L., Reniers M.A., Rooda J.E., Schiffelers R.R.H. Syntax and consistent equation semantics of hybrid Chi. [Электронный ресурс] // URL: http://ac.els-cdn.com/S1567832605000895/1-s2.0-S1567832605000895-main.pdf?_tid=26231c90-c739-11e4-aa5c-00000aacb35f&acdnat=1426001079_b450fab5616606d5bfad09e7ecac25c6

13. Fabian Georgina. A language and simulator for hybrid systems / by Georgina Fabian. Eindhoven: Technische Universteit Eindhoven, 1999.

Fabian Georgina. A language and simulator for hybrid systems / by Georgina Fabian. Eindhoven: Technische Universteit Eindhoven, 1999.