УДК 004.94

В.В. ЛИТВИНОВ[*], А.А. ЗАДОРОЖНИЙ[*], И.В. БОГДАН[*]

ЯЗЫК БЛОЧНОГО ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ НА БАЗЕ МОДИФИЦИРОВАННЫХ ДИАГРАММ ДЕЯТЕЛЬНОСТИ UML

[*]Черниговский национальный технологический университет, г. Чернигов, Украина

Анотація. *У роботі розглядається нова мова блочного імітаційного моделювання, яка дозволяє включати в загальну схему моделювання блоки, отримані шляхом індуктивного моделювання як у вигляді функцій, так і у вигляді автоматних перетворювачів. Запропонована мова моделювання дозволяє об'єднувати в єдину схему моделювання індуктивний і дедуктивний підходи до створення імітаційних моделей. Дворівнева архітектура запропонованої мови дозволяє відділити рівень управління від рівня перетворення даних, чого не можна зробити при застосуванні агрегатного підходу.*
Ключові слова: *блочне імітаційне моделювання, блочна мова моделювання, мережа Петрі, агрегат.*

Аннотация. *В статье рассматривается новый язык блочного имитационного моделирования, который позволяет включать в общую схему моделирования блоки, полученные путем индуктивного моделирования как в виде функций, так и в виде автоматных преобразователей. Предложенный язык моделирования позволяет объединить в единой модели индуктивный и дедуктивный подходы к созданию имитационных моделей. Двухуровневая архитектура предложенного языка позволяет отделить уровень управления от уровня преобразования данные, чего нельзя получить при использовании агрегатного подхода.*
Ключевые слова: *блочное имитационное моделирование, блочный язык моделирования, сеть Петри, агрегат.*

Abstract. *New language of block simulation modeling which allows including blocks obtained with a help of using inductive simulation both functions and automata transducers into the general scheme of the simulation are discussed in the paper. The proposed simulation language allows combining in the single model inductive and deductive approaches to creating simulation models. Two fold architecture of the proposed language allows separating the management level from data transformation level and it cannot be obtained with using aggregate approach.*
Keywords: *block simulation modeling, block simulation language, Petri network, aggregate.*

1. Введение

Современные языки моделирования направлены на имитацию больших разветвленных систем, в которых процессы могут протекать как последовательно, так и параллельно, данные могут быть как дискретными, так и непрерывными, а процессы, протекающие в моделируемых системах, могут быть слабо изученными, количество данных, полученных при проведении экспериментов с реальной системой, может быть большим. Создание моделей систем в таких условиях является сложной задачей, требующей решения большого количества неопределенностей, одной из которых является неопределенность входных данных. Применение индуктивного подхода при создании блоков имитационной модели позволяет устранить неопределенность входных данных. Но не существует языков моделирования, которые бы позволяли включать в схему моделирования блоки, полученные с использованием индуктивного подхода.

Целью статьи является описание элементов и структурной схемы нового имитационного блочного языка моделирования, в котором для графического представления используются модифицированные диаграммы деятельности UML.

ISSN 1028-9763. Математичні машини і системи, 2017, № 4

2. Диаграмма деятельности UML как удобный способ графического представления блочной имитационной модели

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на этих диаграммах также присутствуют обозначения состояний и переходов. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние выполняется только при завершении этой операции.

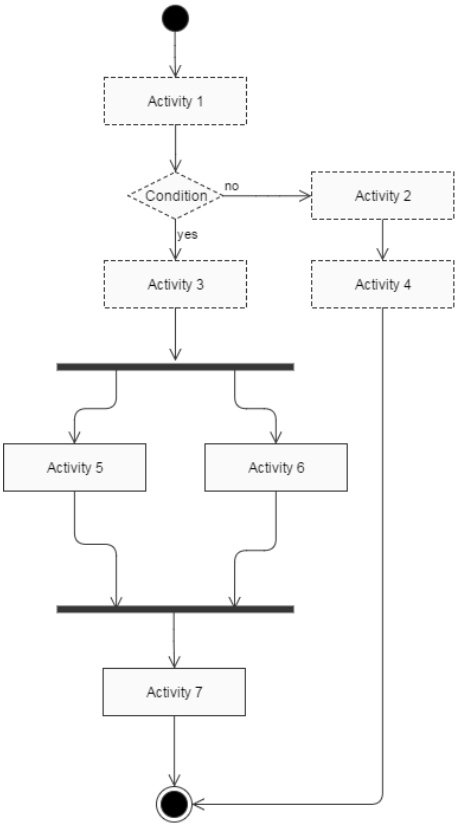

Рис. 1. Пример диаграммы деятельности

Таким образом, диаграммы деятельности можно считать частным случаем диаграмм состояний. Они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних деятельностей и действий. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения.

В контексте языка UML деятельность (activity) представляет собой совокупность отдельных вычислений, выполняемых автоматом, приводящих к некоторому результату или действию (action). На диаграмме деятельности отображаются логика и последовательность переходов от одной деятельности к другой, а внимание аналитика фокусируется на результатах. Результат деятельности может привести к изменению состояния системы или возвращению некоторого значения. Пример диаграммы состояний приведен на рис. 1.

В большинстве случаев диаграмма деятельности состоит из начального и конечного элементов, активностей, элементов разветвления-соединения потоков и блоков принятия решений, соединенных между собой направленными стрелками, которые показывают направление протекания процесса. В примере, представленном на рис. 1, при переходе от элемента Activity 1 к элементу Condition происходит принятие решения в зависимости от условия заданного в элементе Condition. Если условие выполняется, то управление передается элементу Activity 1, а если условие не выполняется, то управление передается элементу Activity 2. При переходе от элемента Activity 3 к элементам Activity 5 и Activity 6 происходит разветвление потока на два потока, а при переходе от элементов Activity 5 и Activity 6 к элементу Activity 7 происходит соединение потоков.

Поскольку диаграммы активности UML предназначены для представления последовательного выполнения процесса, в который входят некоторые базовые операции, и в диаграмме активности присутствуют элементы принятия решений и разветвления-соединения потоков, то с помощью таких диаграмм удобно представлять блочную имитационную модель. В данной работе графические нотации диаграммы UML были использованы в качестве основы блочного языка имитационного моделирования, поскольку с их помощью удобно представлять сложные разветвленные процессы, которые могут протекать как последовательно, так и параллельно, что является основным требованием современного языка блочного имитационного моделирования.

3. Представление общей схемы моделирования

Под общей схемой моделирования имеются в виду правила функционирования схемы, каким образом будут представлены блоки и связи между ними, каким образом будет происходить активация блоков, а также, каким образом будет происходить процесс усложнения блоков. Таким образом, общая схема моделирования определяет механизмы представления и усложнения блоков, механизмы соединения блоков между собой и механизмы активации блоков.

Современный язык имитационного моделирования должен предоставлять возможность создавать сложные имитационные модели с разветвленной сетевой структурой [1]. Если говорить о блочном языке имитационного моделирования, то он, как правило, включает в себя два аспекта: функциональные блоки и управляющие блоки. Функциональные блоки предназначены для осуществления различных функциональных преобразований. Функциональные блоки могут быть представлены преобразованиями в виде формул либо в виде автоматов. Таким образом, современный язык имитационного моделирования является двухуровневым (рис. 2):

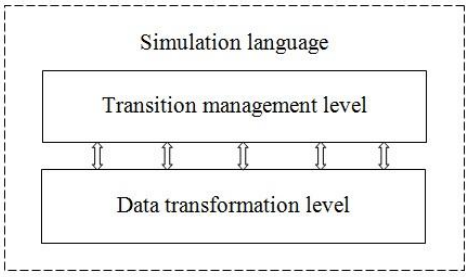

Рис. 2. Двухуровневая структура современного языка имитационного моделирования

Существует много разных подходов к представлению общей схемы моделирования, каждый из которых имеет свои достоинства и недостатки. Одним из способов представления общей схемы моделирования является сеть Петри [2]. Достоинство сетей Петри – это хорошо представленный управляющий уровень языка в виде маркеров и переходов, которые срабатывают при поступлении необходимого количества маркеров. Сетью Петри [2] называется совокупность множеств $\{P,T,I,O\}$, где P – конечное множество, элементы которого называются позициями, T – конечное множество, элементы которого называются переходами, $P \cap T = \varnothing$, I – множество входных функций, $I: T \to P$, O – множество выходных функций, $O: T \to P$. Сети Петри используются главным образом для моделирования параллельных процессов и обладают многими свойствами блок-схем и конечных автоматов. Сети Петри были разработаны и используются для моделирования параллельных и асинхронных систем. При моделировании в сетях Петри позиции символизируют какое-либо состояние системы, а переход символизируют какие-то действия, происходящие в системе. Система, находясь в каком-то состоянии, может порождать определенные действия, и наоборот, выполнение какого-то действия переводит систему из одного состояния в другое.

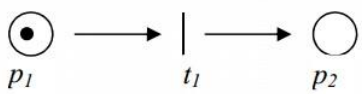

Рис. 3. Сеть Петри до срабатывания перехода t_1

Рассмотрим сеть Петри, представленную на рис. 3. В этой сети Петри $P = \{p_1; p_2\}$, $T = \{t_1\}$, $I(t_1) = \{p_1\}$, $O(t_1) = \{p_2\}$, $\mu = (1;0)$. Срабатывание каждого перехода меняет маркировку сети. Новая маркировка $\mu' = (0;1)$.

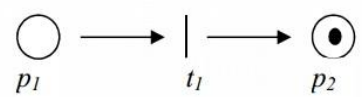

Рис. 4. Сеть Петри после срабатывания перехода t_1

Действия в сети отображаются срабатываниями переходов. Срабатывание перехода t означает удаление по одной метке из каждой позиции p_i, если существует дуга из p_i в t, и добавление метки в каждую позицию p_j, если имеется дуга из t в p_j. Например, срабатывание t_1 из предыдущего примера приводит к сети Петри, представленной на рис. 4.

Таким образом, сети Петри предоставляют удобный механизм перехода между состояниями системы с использованием специального элемента в виде перехода и механизма меток, но механизма по произведению функциональных преобразований и передаче данных не предоставляют.

Вторым способом представления общей схемы моделирования является механизм агрегатов. Агрегаты [3] позволяют представить на едином языке детерминированные и стохастические объекты, которые функционируют как непрерывно, так и дискретно. Понятие агрегата определяется на основании единого подхода к формализации процесса функционирования системы:

– состояние системы в данный момент времени определяется предыдущими состояниями и входными сигналами, поступившими в данный момент времени и ранее;

– выходной сигнал в данный момент времени определяется входными сигналами и состояниями системы, которые относятся к данному состоянию и предшествующим состояниям.

С позиций моделирования агрегат выступает как универсальный преобразователь информации: за конечный интервал времени он воспринимает конечное число входных сигналов и выдает конечное число выходных сигналов. Одним из видов входных сигналов являются управляющие сигналы.

Агрегат имеет входные контакты, на которые в моменты времени t_j поступают входные сигналы. Входной сигнал x является элементом некоторого множества $X : x \in X$. Входной сигнал является вектором, размерность которого равна числу входных контактов. Входной сигнал может быть представлен конечным набором элементарных сигналов $x_1(t),...x_n(t), x_i \in X_i, i = 1, n$, одновременно возникающих на входе агрегата.

На другие особые контакты системы поступают управляющие сигналы в моменты времени τ_i. Управляющий сигнал g является элементом множества $G : g \in G$. За конечный интервал времени в агрегат поступает конечное число входных и управляющих сигналов. Совокупность входных сигналов, расположенных в порядке их поступления, называется входным сообщением соответственно управляющих сигналов – управляющим сообщением.

Выходной сигнал агрегата y является элементом некоторого множества Y и определяется по состояниям агрегата $z(t)$ при помощи оператора G. За конечный интервал времени оператор выдает конечное число выходных сигналов. В общем случае оператор является случайным оператором. Совокупность выходных сигналов, упорядоченная относительно времени выдачи, называется выходным сообщением.

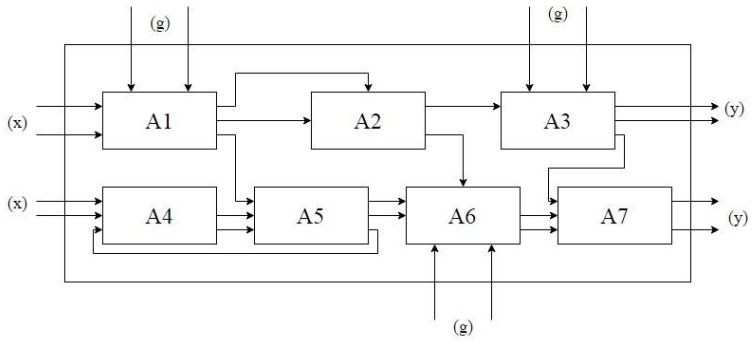

Рис. 5. Пример агрегатной схемы

В каждый момент времени $t \in (0, T)$, в который функционирует система, агрегат находится в одном из возможных состояний. В общем случае множество T может быть непрерывным, дискретным или дискретно-непрерывным. Пример возможной схемы агрегатной системы приведен на рис. 5.

Таким образом, агрегаты хорошо подходят для произведения последовательных и параллельных функциональных преобразований. Связи между агрегатами представлены с использованием оператора сопряжения, что является неудобным, а механизм управления переходами между агрегатами отсутствует.

Третьим способом представления общей схемы моделирования являются механизмы, предложенные в архитектуре WRIGHT. Достоинством WRIGHT по сравнению с агрегатами, является то, что в ней четко определено понятие канала данных (Connector), в отличие от агрегатов, в которых понятие канала данных отсутствует. Механизм управления переходами в архитектуре WRIGHT отсутствует. Как язык архитектурного описания язык WRIGHT построен на базе трех архитектурных абстракций: компонентов, разъемов и конфигураций. Язык WRIGHT предоставляет нотацию для каждого из этих элементов, формализуя понятие компонента как вычислителя и соединителя, как паттерна взаимодействия.

Компонент (component) представляет локальное независимое вычисление. В WRIGHT определение компонента состоит из двух важных частей: интерфейса и вычислителя. Интерфейс состоит из набора портов, которые предназначены для взаимодействия компонента с другими компонентами.

Соединитель (connector) представляет взаимодействие между набором компонентов. Например, соединитель Pipe представляет последовательный поток данных между фильтрами. Соединитель определяет требования, которым должны соответствовать компонент и информация, которая определяет, что компонент может ожидать от внешней системы. WRIGHT определяет соединитель как набор элементов Role и элемент Glue. Каждый элемент Role определяет поведение каждого участника взаимодействия.

Таким образом, для совмещения в общей схеме моделирования механизма функциональных преобразований, механизма передачи данных и механизма управления переходами необходимо применить комбинированный подход, который бы использовал механизм функциональных преобразований, описанный в агрегатном подходе, механизм передачи данных, предложенный в архитектуре WRIGHT, и механизм управления, который применяется в сетях Петри.

4. Элементы языка моделирования уровня преобразования данных

Уровень преобразования данных блочного языка моделирования предназначен для генерации, отображения и преобразования данных. Генерация и отображение данных происходят с использованием элементов «Generator activity» и «Display activity», а преобразование данных происходит с применением элементов «Functional transformation activity» и «Automata transformation activity».

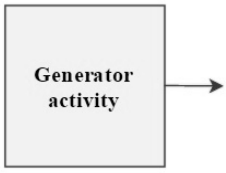

Рис. 6. Блок «Generator activity»

Генерация данных в предложенном языке производится с использованием блока «Generator activity», внешний вид которого представлен на рис. 6. Данный блок имеет всего один выход и должен быть соединен при помощи канала данных с блоком уровня управления «State activity». «Generator activity» может производить генерацию как случайных, так и постоянных данных.

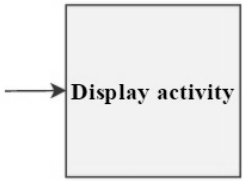

Рис. 7. Блок «Display activity»

Блок «Display activity» предназначен для сбора и отображения результатов моделирования. Он имеет один вход, который должен быть соединен с элементом «Transition activity» уровня управления с использованием канала данных. Внешний вид блока «Display activity» представлен на рис. 7.

Также одной из задач блока «Display activity» является восстановление функциональной зависимости с использованием метода группового учета аргументов. Таким образом, в новой технологии блочного имитационного моделирования автоматизированных систем индуктивный подход используется не только на этапе построения блоков имитационной модели, но и на этапе обработки экспериментов.

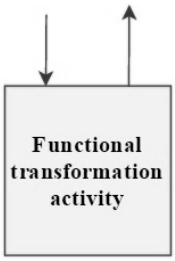

Рис. 8. Блок «Functional transformation activity»

Блок «Functional transformation activity» предназначен для произведения функциональных преобразований с данными, представленными в виде маркера данных. Внешний вид блока представлен на рис. 8. Данный блок имеет один вход и один выход, и данные входы и выходы должны быть соединены каналами данных с блоком «Transition activity». При поступлении маркера данных на вход «Functional transformation activity» производится приведение данных маркера к виду, который подходит для представленной в блоке функции. После завершения функционального преобразования результат преобразования упаковывается в маркер и передается через канал данных в блок «Transition activity».

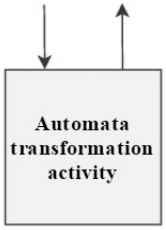

Рис. 9. Блок «Automata transformation activity»

Функции блока «Automata transformation activity» схожи с блоком «Functional transformation activity». Его внешний вид представлен на рис. 9. Данный блок содержит один вход и один выход. Когда на его вход поступает маркер данных, «Automata transformation activity» переходит в активное состояние. Данные извлекаются из маркера данных, после чего они преобразовываются в вид, пригодный для подачи на вход автомата. Автомат преобразовывает данные, поданные на его вход. Данные, которые возвратил автомат, упаковываются в маркер данных, и маркер подается на выход элемента «Automata transformation activity». Вход и выход элемента «Automata transformation activity» должны быть соединены с элементом «Transition activity» уровня управления с использованием канала данных.

Блоки «Functional transformation activity» и «Automata transformation activity» в предложенном языке создаются с использованием индуктивных подходов, поскольку зачастую с большой натяжкой можно считать, что известна функция, которая соответствует законам, протекающим в блоке моделируемой системы, а при таких условиях наилучшим образом подходят индуктивные подходы.

Рис. 10. Канал данных

Внешний вид канала данных представлен на рис. 10. Основной задачей канала данных является связь между собой блоков уровня управления и блоков уровня данных для определения пути следования маркера данных. Может происходить разветвление и соединение нескольких каналов данных в один канал с использованием элемента разветвления-слияния канала данных с уровня управления. Данный элемент можно отнести не только к уровню данных, но и к уровню управления, поскольку он используется для соединения элементов с обоих уровней.

5. Элементы языка моделирования уровня управления

Уровень управления блочного языка предназначен для управления уровнем преобразования данных на уровне преобразования данных, а также управлением модельным временем и изменением состояний модели. Данный уровень использует принцип функционирования сетей Петри и состоит их блоков «State activity», «Transiting activity», блока разветвления-слияния канала данных и блока принятия решений. Канал данных относится как к уровню управления, так и к уровню преобразования данных.

Элемент «State activity» предназначен для накапливания маркеров данных для срабатывания перехода, представленного в виде элемента «Transiting activity». Внешний вид элемента «State activity» представлен на рис. 11. Принцип его функционирования схож с принципом работы позиций в сетях Петри. Отличием является то, что вместо обычных

маркеров, которые в сетях Петри несут только признак присутствия или отсутствия метки, в элементе «State activity» накапливаются метки данных, в которых находится результат преобразования, переданный в позицию от предыдущего элемента «Transiting activity».

Элемент «Transition activity» предназначен для управления переходами между состояниями системы. Его внешний вид представлен на рис. 12. Задачей элемента «Transition activity» является управление переходами между состояниями системы. Срабатывание перехода активируется при поступлении необходимого количества маркеров данных на входы данного элемента. При поступлении маркера выполнение элемента «Transition activity» приостанавливается, а маркер данных передается по каналу данных элементу «Functional transformation activity» либо «Automata transformation activity». Происходит преобразование данных маркера. Элемент «Transition activity» ожидает возвращения маркера с элемента-преобразователя, а при возвращении маркера происходит срабатывание перехода. Существует возможность задания постоянной либо случайной задержки после срабатывания перехода «Transition activity». Данный подход используется во временных сетях Петри. Элемент «Fork-Join» канала данных предназначен для клонирования маркеров данных для их последующей передачи в позиции, стоящие непосредственно после него, либо для сведения нескольких маркеров данных в один маркер данных по определенным правилам. Внешний вид элемента «Fork-Join» представлен на рис. 13. Входы элемента «Fork-Join» всегда соединяются с выходом элемента «Transition activity», а выходы элемента «Fork-Join» соединяются с входами элемента «State activity» с использованием канала данных.

Элемент принятия решения предназначен для разветвления канала данных на два канала данных и выполнения одного из них в зависимости от условия, заданного в этом блоке. Внешний вид элемента принятия решения представлен на рис. 14.

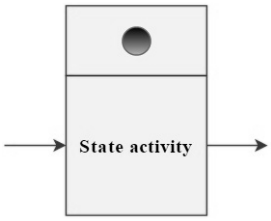

Рис. 11. Блок «State activity»

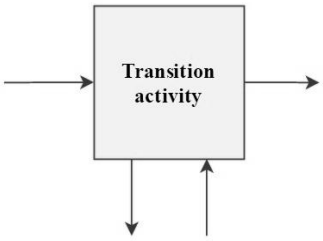

Рис. 12. Блок «Transition activity»

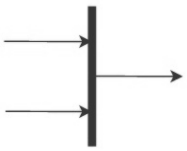

Рис. 13. Блок Fork-Join

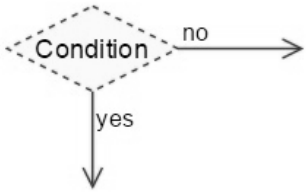

Рис. 14. Блок принятия решения

6. Общая схема блочной имитационной модели

На рис. 15 представлена блочная модель, описанная на языке, созданном на основании комбинации сетей Петри, агрегатов и WRIGHT архитектуры. В начальный момент модельного времени элемент «Generator activity 1» производит генерацию маркера данных, в который упаковывается вектор данных, полученный случайным образом. Маркер данных мгновенно попадает в элемент «State activity 1», присоединенный к элементу «Generator activity 1» каналом данных. Если количество маркеров данных в элементе State activity достаточно, то происходит активация перехода «Transition activity 1». При активации перехода «Transition activity 1» происходит передача маркера данных элементу «Functional transformation activity 1», а выполнение перехода приостанавливается до момента возврата ему маркера данных. При поступлении маркера на вход элемента «Functional transformation activity 1» происходит извлечение данных с маркера, выполняются преобразования данных, необходимые для выполнения функционального преобразования, а затем выполняется само функциональное преобразование. Полученные данные упаковываются в маркер данных и передаются по каналу данных элементу «Transition activity 1». При получении маркера данных элементом «Transition activity 1» происходит срабатывание перехода, система ме-

няет свое состояние и происходит продвижение модельного времени. Маркер данных передается от элемента «Transition activity 1» на элемент разветвления-слияния канала данных. Происходит дублирование маркера данных, и вместо одного маркера в системе при параллельном выполнении двух веток находятся одновременно 2 маркера данных. Один из маркеров попадает в элемент «State activity 2», а второй – в элемент «State activity 2». Происходит параллельное выполнение двух веток. Поскольку маркеров данных достаточно как в элементе «State activity 2», так и в элементе «State activity 3», то происходит активация переходов «Transition activity 2» и «Transition activity 3». Маркеры данных мгновенно передаются элементам «Automata transformation activity 1» и «Automata transformation activity 2». Принцип работы «Automata transformation activity 1» и «Automata transformation activity 2» схож с принципом работы элемента «Functional transformation activity 1». После преобразования данных и упаковки данных в маркеры данных они передаются по каналам данных назад в элементы «Transition activity 2» и «Transition activity 3», а затем происходит срабатывание этих переходов. Слияние данных в единый маркер данных происходит в элементе разветвления-слияния канала данных, а затем маркер данных попадает в элемент «State activity 4», поскольку в этом элементе маркеров данных достаточно, чтобы сработал переход «Transition activity 4», тогда маркер данных попадает на элемент «Display activity 1». Этот элемент либо отображает результат на экране, либо производит протоколирование полученных данных.

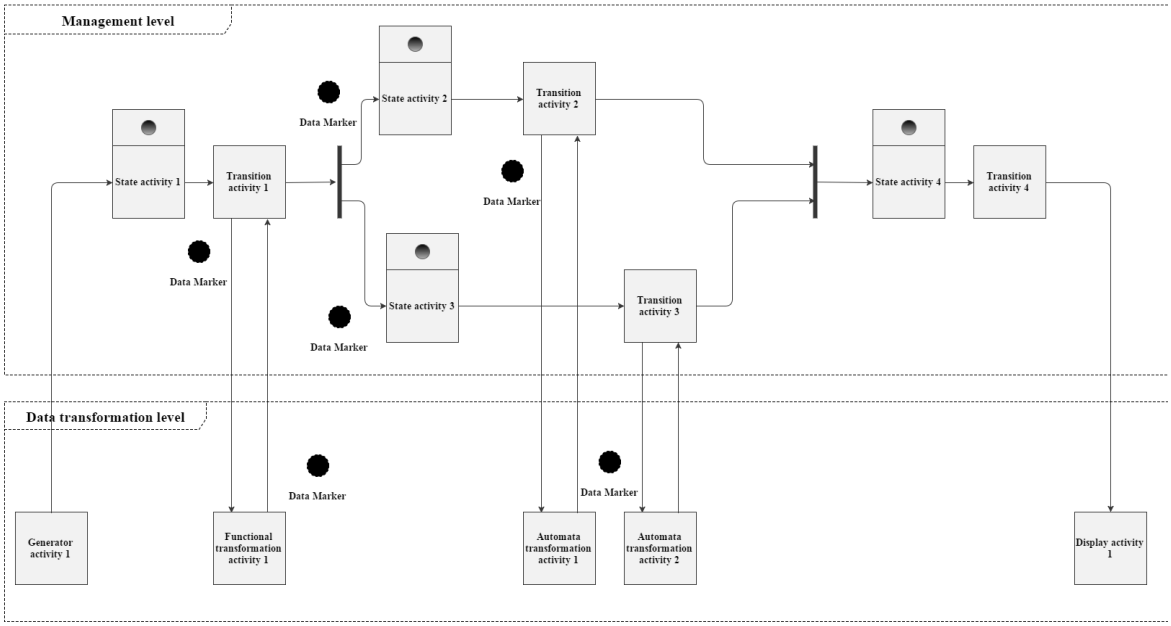

Рис. 15. Двухуровневая структура предложенного языка моделирования

7. Выводы

Современный язык моделирования должен соответствовать ряду требований, чтобы обеспечивать возможность моделирования сложных разветвленных систем, в которых процессы могут протекать как последовательно, так и параллельно, данные могут быть дискретными и непрерывными. В ходе анализа были рассмотрены три возможных способа представления общей схемы моделирования: сети Петри, агрегаты и архитектура WRIGHT. Ни один из предложенных способов не дает возможности организовать схему моделирования, которая включает в себя функциональные и автоматные преобразования. Таким образом, был предложен комбинированный подход, который использует в качестве основы механизм переходов, использованный в сетях Петри, агрегаты как вычислительные элементы и понятие канала данных из архитектуры WRIGHT. На базе комбинированного подхода был

предложен новый язык моделирования, элементы которого разделены на 2 уровня: уровень управления и уровень преобразования данных. Предложенный язык имитационного моделирования предполагает четкое разделение уровня управления и уровня преобразования, в отличие от существующего агрегатного подхода, а также предоставляет возможность создания имитационных моделей, в которых возможно исследование не только временных характеристик, но и функциональных преобразований, которые протекают в системе. В автоматизированных системах необходимо моделировать не только временные характеристики, но и функциональные преобразования, которые протекают в такой системе.

СПИСОК ЛИТЕРАТУРЫ

1. Бусленко Н.П. Лекции по теории сложных систем / Бусленко Н.П., Калашников В.В., Коваленко И.Н. – М.: Советское радио, 1973. – 440 с.
2. Киндлер Е. Языки моделирования / Киндлер Е. – М.: Энергоатомиздат, 1985. – 287 с.
3. Полляк Ю.Г. Вероятностное моделирование на электронных вычислительных машинах / Полляк Ю.Г. – М.: Советское радио, 1971. – 400 с.

Стаття надійшла до редакції 20.11.2017