ИНСТРУМЕНТАЛЬНАЯ СИСТЕМА РАСПРЕДЕЛЕННОГО ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Ю.И. Бродский (ВЦ РАН)
yury_brodsky@mail.ru

Цель исследования – создание инструментальных средств, для распределенного имитационного моделирования сложных организационно-технических систем. Имеются в виду такие сложные системы, про которые хорошо известно, из каких компонент они состоят, что эти компоненты умеют делать, по каким правилам взаимодействуют между собой. Проблемой моделирования, притом весьма непростой, является воспроизведение поведения и оценка возможностей такой системы в целом. Для ее решения исходные представления о сложной системе формализуются. Ключевые понятия этой формализации — понятия компонент, образующих комплекс, и комплекса, состоящего из компонент, но на более высоком уровне абстракции воспринимающегося как единая компонента. Разработанная концепция распределенного имитационного моделирования реализована в виде инструментальной системы, которая, во-первых, помогает построить синтез сложной модели, и во-вторых, позволяет создать в Интернете пиринговую сеть моделирования, в которой каждый ее участник может предоставлять во всеобщий доступ (опубликовать) разработанные им методы, а также может использовать в разрабатываемых им моделях подходящие методы, опубликованные другими участниками.

Проблемы и решения в распределенном имитационном моделировании

Анализ приведенных примеров реализаций инструментальных средств моделирования показывает, что авторы этих разработок в разное время, и по всей видимости независимо друг от друга, приходили к весьма сходным решениям относительно способов организации вычислительного процесса имитации.

Назовем основные из этих решений:

1. Объектная декомпозиция модели.
2. Моделирование с переменным шагом времени. Декомпозиция модельного времени на события и промежутки между ними.
3. Планирование или прогнозирование событий.
4. Декомпозиция активности модели на параллельные процессы-потоки активностей, каждый из которых есть череда сменяющихся элементарных алгоритмов.
5. Определение правил перехода между элементарными алгоритмами, как функции состояния и характеристик модели.
6. Взаимодействие компонент модели через механизм сигналов/сообщений.
7. Наличие специального непроцедурного языка для описания состава модели, связей между ее компонентами и некоторых правил их функционирования.
8. Интеграция инструментальной системы с одним или несколькими языками программирования высокого уровня.

Возникает вопрос, действительно ли это набор наилучших решений из всех возможных в данной области? Или же среди них есть случайно попавшие в список, без которых вполне можно было бы обойтись?

Кроме этого, нерешенным остается ряд проблем, связанных с моделированием сложных систем. Перечислим некоторые из них:

1. Все приведенные в качестве примеров инструментальные системы моделирования исходят из предположения, что моделирование интересующей разработчика предметной области в принципе возможно. Однако крайняя степень такого оптимизма была бы в философском смысле лапласовским детерминизмом, и поэтому вряд ли может быть оправдана, Возникает вопрос: а все-таки когда возможно моделирование, в том смысле как его понимают, например, большинство разработчиков упомянутых систем?
2. Все разработчики систем моделирования соглашаются, что сложные модели приходится моделировать с переменным шагом времени, уменьшая шаг моделирования, в случае наступления на этом шаге некоторого события. Возникает вопрос: а не можем ли мы в результате все уменьшающегося шага моделирования получить сходящуюся

последовательность, наподобие знаменитой апории Зенона про Ахиллеса и черепаху? И вообще, какой способ управления временем выбрать, – их много хороших и разных?

3. Большинство разработчиков описанных выше систем соглашаются, что активность сложных моделей естественно декомпозировать на ряд параллельно выполняющихся процессов. Это было бы очень выгодно с точки зрения повышения эффективности вычислительного процесса – параллельность в настоящее время магистральный и далекий до исчерпания путь повышения эффективности вычислений. Как тогда быть с взаимодействием параллельно выполняющихся компонент? Разрешать ли всем желающим доступ ко всем характеристикам и организовывать очередность доступа к ним? Или лучше подписки? Или сигналы/сообщения? Или все вместе? Или что-то еще?
4. Большинство специалистов в области моделирования согласны с тем, что процесс имитационных вычислений должен быть детерминированным (что, конечно, отнюдь не исключает моделирование этими детерминированными вычислениями в том числе и полностью стохастических процессов). Детерминированность процесса вычислений может, например, означать что создаются списки на выполнение каких-то действий с какими-то объектами. При этом объекты попадают в эти списки в случайном (в смысле отсутствия у разработчика модели явных намерений на это счет) порядке, например, становятся первыми в списке из-за того, что когда-то были первыми описаны или созданы. Возникает вопрос, как организовать имитационные вычисления так, чтобы одни объекты моделирования не имели систематических преимуществ перед другими из-за случайно полученного места в том или ином списке обработки?

Предлагаемая концепция моделирования

Часто, при построении весьма непростого синтеза многокомпонентных систем, разработчики не ограничивают себя в средствах его реализации какой-либо специальной дисциплиной их применения, по-видимому, полагая, что для решения столь сложной задачи избыточных средств не бывает. В результате, приходится решать задачу взаимодействия нескольких параллельно протекающих процессов в самом общем виде, откуда возникают весьма непростые механизмы синхронизации времени. Отсюда же возникает забота о дисциплине доступа к фазовым переменным модели, и связанные с ней проблемы зацикливания, при ожидании доступа к фазе.

Мы будем исходить из того, что реализуемая модель удовлетворяет некоторому набору основных гипотез (без выполнения которых имитационное моделирование сложных систем представляется нам невозможным). Приняв эти гипотезы, мы окажемся в принципиально иной ситуации – взаимодействовать между собой будут не произвольные параллельные процессы, а такие, которые являются составляющими модели, удовлетворяющей этим гипотезам. Казалось бы, разница невелика – что не может быть моделью сложной системы? Тем не менее, используя свойства модели, заявленные в основных гипотезах, удается построить достаточно простой синтез модели сложной системы из компонент, при этом решая за счет определенной дисциплины проектирования модели такую проблему, как, например, доступ к фазовым переменным.

Отправной точкой моделирования обычно является гипотеза о замкнутости – предположение о том, что если в момент t известны значения внутренних и внешних переменных модели, то найдется такое $\Delta t > 0$, что по крайней мере на отрезке $[t, t + \Delta t]$ мы сможем детерминировано и однозначно прогнозировать внутренние переменные. Если гипотеза о замкнутости не выполняется, построение модели остается под вопросом. Часто на практике, предположив, что в каждой точке отрезка гипотеза о замкнутости выполнена, пытаются распространить ее на весь отрезок. Оказывается, это возможно не всегда.

Определение 1.

Будем называть модель замкнутой в точке $t \in [0, T)$, если найдется число $\Delta t > 0$, $t + \Delta t \in (0, T]$, которое будем называть отрезком прогноза модели для точки t, такое что для любого $\tau \in (t, t + \Delta t]$:

1. Известен алгоритм, позволяющий детерминировано, однозначно и с устраивающей разработчика модели точностью вычислять по значениям внутренних $\vec{x}(t)$ и внешних $\vec{a}(t)$

переменных в момент модельного времени t, значения внутренних переменных $\vec{x}(\tau)$ в момент модельного времени τ.

2. Упомянутый алгоритм обладает следующим свойством аддитивности: если выполняется предыдущий пункт и $\tau \in (t, t+\Delta t)$, то алгоритм позволяет получить прогноз $\vec{x}(\theta)$, исходя из начальных значений $\vec{x}(\tau)$ и $\vec{a}(\tau)$, по крайней мере на полуинтервале $\theta \in (\tau, t+\Delta t]$, и этот прогноз с устраивающей разработчика точностью совпадает на отрезке $[\tau, t+\Delta t]$, с прогнозом, полученным на всем отрезке $[t, t+\Delta t]$, исходя из начальных значений характеристик $\vec{x}(t)$ и $\vec{a}(t)$.

Под упомянутым в определении алгоритмом будем понимать отображение $\vec{x} = F(\vec{x}, \vec{a}, \Delta t)$, желательно сюръективное (что соответствовало бы управляемости модели внешними характеристиками), из $P \times Q \times [0,T]$ в P. Здесь $P \subset R_n$ – замкнутое ограниченное множество допустимых значений внутренних переменных модели, а $Q \subset R_m$ – замкнутое ограниченное множество допустимых значений ее внешних переменных. Вообще говоря, отображение F не предполагается непрерывным, так как некоторые характеристики модели могут быть дискретными.

«Устраивающей разработчика точностью» можно назвать топологию на P, которую разработчик согласится считать таковой, т. е. оценивать близость в P, в смысле этой топологии. Топология «устраивающей разработчика точности» вполне может оказаться слабее топологии задаваемой нормой R_n, и притом настолько, что в смысле этой топологии отображение F уже может оказаться непрерывным относительно Δt. Если эта непрерывность кроме того будет равномерной и равностепенной относительно $\vec{x}, \vec{a} \in P \times Q$, то алгоритм, задаваемый отображением F, удовлетворит условиям определения 1.

Определение 2.

Будем называть модель локально замкнутой на отрезке $[0,T]$, если она замкнута в каждой точке $t \in [0,T)$.

Определение 3.

Будем называть модель прогнозируемой или лапласовской на отрезке $[0,T]$, если существует конечное разбиение этого отрезка точками $0 = t_0 < t_1 <, \ldots, < t_n = T$, такое что модель замкнута в каждой из точек t_{i-1}, $i = 1, \ldots, n$, и каждый из полуинтервалов $(t_{i-1}, t_i]$, $i = 1, \ldots, n$, принадлежит отрезку прогноза для своего левого конца.

Определение 4.

Будем называть модель прогнозируемой в точке $t \in (0,T]$, если найдется точка $\tau_t \in [0,t)$, которую будем называть базовой для точки t, такая что:

1. Модель замкнута в точке τ_t.
2. Точка t принадлежит отрезку прогноза модели в точке τ_t: $t \in (\tau_t, \tau_t + \Delta\tau_t]$.

Определение 5.

Будем называть модель локально прогнозируемой на отрезке $[0,T]$, если она прогнозируема в каждой точке $t \in (0,T]$.

Теорема (О лапласовской модели).

Для прогнозируемости модели на отрезке модельного времени $[0,T]$, необходима и достаточна ее локальная замкнутость, и локальная прогнозируемость на этом отрезке.

Приведем пример локально замкнутой, но не лапласовской модели.

Рассмотрим задачу, встречающуюся в курсе математики начальной школы. Два пешехода идут навстречу друг другу с постоянной скоростью. Между ними летает муха с постоянной по абсолютной величине скоростью, большей, чем скорость любого из пешеходов. Как только муха долетает до одного из пешеходов, она тут же разворачивается и летит к другому. В школьной задаче обычно спрашивается, какое расстояние пролетит муха за время от начала движения, до момента встречи пешеходов.

Данная модель замкнута в любой точке любого разумного отрезка времени, но не прогнозируема в момент встречи пешеходов. В результате невозможно построить лапласовскую модель для этой задачи на любом отрезке времени, включающем с окрестностью момент встречи пешеходов.

Рассматриваются вопросы декомпозиции модели. Показывается, что система описываемая обыкновенными дифференциальными уравнениями с липшицевой правой частью допускает сколь угодно точную декомпозицию на два подъобекта, и является лапласовской моделью.

Далее предлагается концепция построения инструментальной системы моделирования. Дается формализованное описание элементарной модели сложной системы – **компонента**.

1. Характеристики. Компонента, как и объект, имеет характеристики. Как и в случае динамических систем, эти характеристики мы будем разбивать на внутренние и внешние. Внутренние характеристики – это то, что компонента моделирует, внешние – это то, что она знает о внешнем мире. Обычно мы будем предполагать локальную прогнозируемость компоненты.

2. Процессы. Функциональность компоненты удобно структурировать следующим образом: Считается, что компонента реализует один или несколько параллельно выполняющихся процессов. Процесс состоит в чередовании элементов – алгоритмически элементарных методов.

3. Элементы. Элементарные, алгоритмически однородные методы, реализующие функциональности компоненты (т. е. то, что компонента умеет делать, получение на основании значений некоторых внутренних и внешних характеристик компоненты, новых значений некоторых ее внутренних характеристик).

По отношению к модельному времени некоторые элементы выполняются мгновенно, это сосредоточенные или быстрые элементы. Быстрыми элементами можно моделировать дискретные характеристики системы.

Выполнение других элементов занимает определенное время. Если при этом элемент для любого промежутка времени $\Delta\tau$, не превосходящего стандартный шаг моделирования Δt выдает некий осмысленный результат, такой элемент называется распределенным или медленным. Распределенные элементы – естественное средство вычисления непрерывных характеристик модели.

Может оказаться и так, что выполнение элемента занимает определенное модельное время, но результат его действия наступает лишь в конце, после полного выполнения элемента, т. е. никаких промежуточных результатов за время меньшее полного времени выполнения нет. Такие элементы называются условно-распределенными. Вообще говоря, условно-распределенные элементы можно не рассматривать как отдельный класс, а моделировать парой: пустой распределенный элемент, за которым идет сосредоточенный, выдающий результат.

Частью предлагаемой концепции является жесткая дисциплина работы методов с фазовыми переменными модели: каждый метод имеет право изменять только «свои» переменные. Эта дисциплина основана на принятии предположения о детерминированности и однозначности имитационных вычислений. В рамках предлагаемой концепции, конфликт доступа возникающий, когда методы A и B вычисляют одну и ту же характеристику $x = x_A$ и $x = x_B$, может быть разрешен, например, введением метода C, который получая на входе в качестве параметров x_A и x_B, вычисляет на их основании искомую характеристику x, устраняя тем самым не только конфликт доступа к ней, но и очевидно, имевшую место неоднозначность вычисления упомянутой характеристики. Принятая дисциплина доступа к характеристикам позволяет вызывать параллельно те методы, которые в модельном времени выполняются одновременно.

Наконец, последнее, что следует заметить относительно элементов. Как и всякий метод, каждый элемент имеет входные и возвращаемые параметры. Концепция моделирования предполагает возможность распределенного вычисления элементов, т. е. элемент, вообще говоря, может быть найден разработчиком компоненты на просторах Интернета, поэтому его параметры таковы, какими их сделал его автор, и скорее всего, никак не согласованы с характеристиками компоненты, которые придумал ее разработчик. Поэтому, при описании компоненты обязательно должно быть уделено внимание описанию коммутаций входных параметров элементов с внутренними и внешними характеристиками компоненты и выходных параметров элементов – с ее внутренними характеристиками.

4. События. Содержательно, события – это то, что нельзя пропустить при моделировании динамики системы – точки синхронизации различных ее функциональностей, представляемых

процессами. Точки, когда получены такие значения характеристик модели и внешних переменных, на которые обязаны отреагировать некоторые процессы компоненты.

Формально событие – функция значений внутренних и внешних переменных в начале шага моделирования. С точки зрения организации имитационных вычислений, событие – это метод, входными параметрами которого является подмножество внутренних и внешних характеристик компоненты, а выходной параметр один – прогнозируемое время до наступления этого события. Если это прогнозируемое время равно нулю – значит, событие уже наступило. События управляют чередованием элементов в процессе.

Для каждой упорядоченной пары элементов процесса $\{A, B\}$, если между ними возможен переход, то ему обязан соответствовать метод-событие $E_{\{A,B\}}$, прогнозирующий время этого перехода. Возможны также события вида $E_{\{A,A\}}$, прерывающее выполнение элемента A, например, если его еще не нужно заканчивать, но он вычислил характеристики компоненты, которые могут повлечь смену элементов других процессов. Процесс перехода должен быть однозначным. Одновременное наступление событий $E_{\{A,B\}}$ и $E_{\{A,C\}}$ говорит лишь о том, что разработчик модели при ее проектировании упустил из своего рассмотрения этот случай, которому, быть может, должен соответствовать переход $\{A, D\}$.

5. Выполнение компоненты. Компонента – элементарная, но, тем не менее, полноправная модель сложной системы. Поэтому ее можно запустить на выполнение имитационного эксперимента. Конечно, если имеются начальные данные и способ нахождения внешних характеристик компоненты в моменты событий. Алгоритм функционирования компоненты после ее запуска на счет описан в виде четырех правил:

1. Вычисляются события связанные с текущими элементами процессов. Если есть наступившие события, проверяется нет ли переходов к быстрым (сосредоточенным) элементам, если они есть – выполняются быстрые элементы (они становятся текущими), затем возврат к началу п.1; если нет переходов к быстрым элементам – совершаются переходы к новым медленным (распределенным) элементам, затем возврат к началу п.1.
2. Если нет наступивших событий – из всех прогнозов событий выбирается ближайший $\Delta\tau$.
3. Если стандартный шаг моделирования не превосходит прогнозируемого времени до ближайшего события, $\Delta t \le \Delta\tau$ – моделируем текущие распределенные элементы со стандартным шагом Δt. В противном случае – моделируем их с шагом времени до ближайшего спрогнозированного события $\Delta\tau$.
4. Возвращаемся к началу п.1.

Компоненты могут объединяться в **комплекс**, при этом (необязательно) может оказаться, что некоторые компоненты явно моделируют внешние переменные некоторых других компонент. Для того, чтобы полностью описать комплекс, достаточно указать:

1. Какие компоненты и в каком количестве экземпляров в него входят.
2. Коммутацию компонент внутри комплекса, если она имеет место, т. е., какие внутренние переменные каких компонент являются какими внешними переменными и каких именно компонент комплекса.

Комплекс состоящий из многих компонент, вовне может проявляться в качестве единой компоненты. Введем следующую операцию объединения компонент комплекса:

1. Внутренними переменными комплекса объявляется объединение внутренних переменных всех его компонент.
2. Процессами комплекса объявляется объединение всех процессов его компонент.
3. Методами комплекса объявляется объединение всех методов его компонент.
4. Событиями комплекса объявляется объединение всех событий его компонент.
5. Внешними переменными комплекса объявляется объединение всех внешних переменных его компонент, из которого исключаются все те переменные, которые моделируются явно какими-либо компонентами.

Операция объединения превращает комплекс в компоненту, причем объединенный комплекс, рассматриваемый как компонента, наследует от своих компонент такое важное свойство, как лапласовость (прогнозируемость на отрезке).

Этот факт позволяет строить модель как фрактальную конструкцию, сложность которой (и соответственно подробность моделирования) ограничивается лишь желанием разработчика.

Основные предложения концепции

Прогнозируемость и замкнутость модели

На уровне гуманитарного понимания, под возможностью построения имитационной модели некоторого явления на отрезке времени $[0,T]$, по-видимому, обычно понимается нечто близкое к данному выше определению 3 прогнозируемости модели на отрезке времени.

Отметим, что утверждение о возможности моделирования в этом смысле любого явления, вполне в духе детерминизма Лапласа, и поэтому наряду с признанием таких проявлений спонтанности как чудо, акт творчества, проявление свободной воли или просто радиоактивный распад, является скорее предметом религиозно-философских убеждений, нежели научным фактом.

Оказывается, что для детерминизма моделирования в смысле определения 3, недостаточно лишь классической замкнутости модели в смысле определения 1, – возможности распространить имеющееся в момент t знание о состоянии системы и окружающего мира на знание о состоянии системы на небольшом последующем интервале времени $[t,t+\Delta t]$. Кроме этого, еще необходима обусловленность каждого текущего состояния системы некоторой его предысторией в смысле определения 4 прогнозируемости модели в точке. Без такой обусловленности, эволюция модели разбивается непрогнозируемыми моментами на ряд эпизодов, не вполне связанных между собою причинно-следственной связью.

Между прочим, у Лапласа имеется следующее широко цитируемое высказывание: «Современные события имеют с событиями предшествующими связь, основанную на очевидном принципе, что никакой предмет не может начать быть без причины, которая его произвела...», которое, по-видимому (Лаплас обусловливает современные события их предысторией), свидетельствует о понимании им важности именно прогнозируемости в смысле определения 4, для возможности построения детерминированной модели изучаемого явления.

Требование замкнутости модели в начальной точке и прогнозируемости ее в остальных точках отрезка моделирования гарантирует существование прогноза поведения модели на отрезке в смысле определения, однако, вовсе не гарантируют, что любые усилия разработчика в области построения модели завершатся именно таким прогнозом. Например, вспомним знаменитую апорию Зенона Элейского про Ахиллеса и черепаху. По существу (в отличие, например, от «пешеходов и мухи»), – это и замкнутая и прогнозируемая во всех точках модель. Однако ее разработчик Зенон так выбрал системные события (моменты, когда Ахиллес добегает до того места, где в начале шага находилась черепаха), что у них оказалась точка накопления – момент, когда Ахиллес поравняется с черепахой. С точки зрения развиваемой здесь теории, так получилось из-за того, что выбранные Зеноном точки синхронизации произвольны и неудачны в том смысле, что в предметной области модели за ними не стоят какие-либо реальные события, из-за которых стоило бы прерывать моделирование с постоянным шагом. Однако, выбери Зенон иные события – и знаменитой апории не получилось бы. Таким образом, успех разработчика в построении модели будет зависеть от владения им искусством моделирования, так как процесс такого построения в настоящее время не формализован. Наука же моделирования посредством приведенной выше теоремы может лишь пообещать разработчику, что его усилия не бессмысленны, так как в принципе такое построение возможно.

Однозначность и параллельные вычисления

Попытка со стороны двух или более параллельно выполняющихся в модельном времени элементов изменить одну и ту же характеристику модели, всегда означает неоднозначность имитационных вычислений указанной характеристики. Поэтому последовательное проведение в жизнь требования однозначности имитационных вычислений на стадии проектирования модели должно обеспечить безопасный параллельный вызов одновременно в модельном времени выполняющихся элементов.

Еще на стадии проектирования модели все ее характеристики должны быть распределены между ее элементами, которые могут выполняться одновременно в модельном времени так, чтобы ни в коем случае никакой элемент не имел права изменять «чужие» характеристики. Каждый элемент имеет право лишь на изменение «своих», выделенных именно ему, характеристик модели. Дисциплина изменения каждым элементом только своих характеристик должна быть столь же жесткой, как например, требование обходиться без оператора goto в структурном программировании. Точно также как и в предыдущем пункте, можно сказать, что осуществление этой дисциплины в значительной мере относится к искусству моделирования. В значительной, а

не полной, – потому что например, такой формальный способ разрешения неоднозначности, как введение дополнительных характеристик и дополнительных элементов, выбирающих единственную характеристику из серии неоднозначных – работает всегда, не заменяя при этом, конечно, искусство моделирования.

Существование формального способа разрешения неоднозначностей должно убеждать разработчика в том, что требование жесткой дисциплины доступа к характеристикам модели вполне осуществимо. Примерно такую же роль в обосновании жестких требований структурного программирования играла известная теорема Дейкстры, доказывающая принципиальную возможность обходиться в программах без оператора безусловного перехода.

Награда за исполнение жесткой дисциплины доступа к характеристикам модели – возможность параллельного запуска на счет элементов, выполняющихся в модельном времени одновременно.

Декомпозиция модели и организация вычислительного процесса имитации

Организация вычислительного процесса имитационного эксперимента как простейшей, однокомпонентной, так и более сложной модели, также в конце концов приведенной к однокомпонентному виду, происходит согласно пяти правилам, приведенным выше.

Так как эти правила одинаковы для любой модели сложной системы, вполне естественно возложить упомянутую организацию вычислительного процесса имитации на специальное программное обеспечение для подобных задач – на инструментальную систему распределенного имитационного моделирования сложных систем.

Для реализации модели на компьютере, от разработчика потребуется лишь

1. Описать как устроена модель, а именно, для каждой компоненты описать какие процессы в нее входят, из каких элементов они состоят, какие элементы могут переключаться на какие, какие события вызывают эти переключения, наконец, указать коммутацию входящих и выходящих параметров элементов с характеристиками модели. На этапе трансляции таких описаний может быть первичный контроль за однозначностью вычислительного процесса. Затем можно строить иерархию комплексов из имеющихся компонент, рассматривая каждый комплекс как новую компоненту. Все упомянутые описания можно делать на специальном непроцедурном языке описаний компонент и комплексов, а можно, как это модно делать в последние годы, придумать для этого специальный графический интерфейс.
2. Написать на обычном языке программирования (например, C++, C#, java) все методы-элементы и методы-события. При этом, сосредоточенные элементы – это достаточно обычные методы, получающие входящие параметры и возвращающие выходящие. Распределенные же, кроме этого, еще всегда получают продолжительность текущего шага моделирования Δt, и должны уметь выдавать свой результат в соответствии с этой продолжительностью.

Инструментальная система распределенного моделирования

Предлагаемая инструментальная система во-первых, помогает построить синтез сложной многокомпонентной модели, что само по себе обычно является нетривиальной задачей, и, во-вторых, позволяет создать в Интернете пиринговую сеть моделирования, в которой с одной стороны, каждый участник сети может предоставлять во всеобщий доступ (опубликовать) разработанные им методы, а с другой стороны, может использовать в разрабатываемых им моделях подходящие методы, разработанные другими участниками и опубликованные ими в Интернете. При этом от публикуемых методов требуется лишь, чтобы они по своему текущему состоянию (набору внутренних характеристик), набору внешних характеристик (параметров) и интервалу времени моделирования, могли вычислять внутренние переменные в конце этого интервала. Как правило, так или иначе именно этим и занимается большинство компьютерных программ. Публикация метода, помимо описания его назначения и интерфейса на естественном языке, предполагает в дальнейшем возможность его удаленных вызовов. В основе инструментальной системы лежит описанная в предыдущей главе концепция анализа и синтеза сложных систем, ориентированная на параллельное выполнение методов, одновременно протекающих в модельном времени.

Основой сети распределенного моделирования является программное обеспечение пиринговой рабочей станции, оно состоит из клиентской и серверной частей. Клиентская часть

ответственна за создание моделей. Она содержит средства, позволяющие по описаниям на специальном формальном непроцедурном языке ЯОКК (языке описания комплексов и компонент) методов, событий, компонент и комплексов строить информационную структуру модели, в первую очередь ее базу данных, а также средства поддержки выполнения модели и наблюдения за результатами моделирования. При этом, методы, обеспечивающие функциональность модели могут быть как локальными, так и удаленными. Серверная часть пирингового клиента предоставляет в первую очередь сервис удаленных вызовов опубликованных на этом хосте методов, а также сервис просмотра каталога и описаний этих методов.

Архитектура программного обеспечения рабочей станции такова, что вся модель от начала и до конца всегда собирается на одной рабочей станции. При этом, вообще говоря, ни один из методов модели, будь то элемент или же событие, может не быть реализован на этой рабочей станции. Любые методы модели могут быть «чужими» и выполняться удаленно на других компьютерах сети распределенного моделирования. В этом контексте слова о том, что вся модель находится на рабочей станции, означают, что на ней по исходным описаниям создается база данных модели, которая затем заполняется начальными данными, а впоследствии содержит все характеристики модели, как внутренние так и внешние, на всех отработавших шагах моделирования. Распределено вызываются только методы, которым в момент вызова передаются их входные параметры, а после вызова принимаются их возвращаемые параметры.

Для формального описания компонент и комплексов модели, служит специальный непроцедурный язык описания комплексов и компонент (ЯОКК). На этом языке специальными языковыми конструкциями описываются типы данных; элементы (в том числе адрес элемента в сети, типы входящих и возвращаемых параметров); события; компоненты (в описании компоненты присутствует описание ее характеристик, перечень процессов, элементы составляющие процессы, правила переключения элементов в зависимости от событий, и, наконец, коммутация между характеристиками модели и входными и возвращаемыми параметрами элементов); комплексы (здесь описывается состав комплекса из компонент и коммутация некоторых внутренних переменных одних компонент с определенными внешними переменными других). Программное обеспечение рабочей станции содержит средства редактирования, компиляции и отладки конструкций ЯОКК. Так, компиляция компоненты в случае ее успешного окончания создает базу данных модели с незаполненной таблицей начальных характеристик. На рабочей станции имеются средства для работы с этой базой, как для начальной идентификации модели (заполнения сгенерированной компилятором базы внешними и начальными переменными), так и для работы с характеристиками модели во время приостановки имитационного эксперимента. Описание комплекса компилируется в описание же на языке ЯОКК компоненты в соответствии с указанными выше правилами объединения, после чего полученное описание компилируется уже как описание компоненты. Откомпилированную компоненту с заполненной начальными данными и внешними переменными базой данных можно запустить на выполнение. Этим занимается монитор времени выполнения рабочей станции. Выполняемые одновременно в модельном времени элемент при этом по возможности выполняются параллельно.

В качестве примера построения распределенной модели сложной системы на рабочей станции пиринговой сети имитационного моделирования, приводится полная реализация упоминавшейся выше простейшей нелапласовой модели «Пешеходы и муха».

Основные выводы и результаты

- Положено начало формализации моделирования сложных многокомпонентных систем. Очерчен класс систем, допускающих моделирование – это лапласовские системы – системы одновременно локально замкнутые и локально прогнозируемые. Показано на примере, что одной только локальной замкнутости недостаточно для построения модели. В класс лапласовских систем, например, попадают системы, описываемые обыкновенными дифференциальными уравнениями с липшицевой правой частью.
- Разработана концепция построения инструментальных средств распределенного имитационного моделирования сложных систем. Проведенная формализация позволила отказаться в этой концепции от ряда достаточно распространенных, но как оказалось, необязательных средств организации имитационных вычислений (таких, например, как сигналы/сообщения, или сложные системы управления модельным временем).

- Разработана архитектура пиринговой сети имитационного моделирования в Интернете, отличающаяся от известных, и позволяющая разработчику полностью контролировать модель, даже если все ее составляющие – удаленные.
- Реализован макет программного обеспечения рабочей станции пиринговой сети имитационного моделирования, на котором проверены основные моменты разработанной концепции моделирования.

Литература

1. Бродский Ю.И. РАСПРЕДЕЛЕННОЕ ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ: эвристика, теория, инструментальные средства – Saarbrucken: LAP LAMBERT Academic Publishing, 2011, 152 с.
2. Бродский Ю.И. Распределенное имитационное моделирование сложных систем М.: ВЦ РАН, 2010, 156 с.