Вышеприведенные гипотезы требуют обратить особое внимание в развитии модельных исследований на описание энергетических аспектов эволюции. Представляется, что именно описание судьбы энергетических потоков, обеспечивающих далекое от равновесного стационарного состояния структуры современной биосферы может дать шанс на существенное продвижение в становлении общесистемного подхода исследований устойчивого ее развития. В работах [8–10] сделаны попытки получения такого рода формализаций биосферных процессов с акцентом на рассмотрение энергетической их постановки. Привлечение представленной выше теории возникновения и эволюции жизни, модельной теории эволюции [1, 2], определение селективной ценности как меры эффективности использования энергии для сохранительных работ позволило заложить основы модельного аппарата исследований. Далеко не все проблемы использования этого аппарата в исследованиях устойчивого развития получили решение. Но уже то, что такой общий подход сформулирован, внушает определенный (конечно же, субъективный) оптимизм.

Литература

1. *Эйген М.* Самоорганизация материи и эволюция биологических макромолекул. М.: Мир, 1973.
2. *Волькенштейн М. В.* Общая биофизика. М.: Наука, 1978. 590 с.
3. *Романовский Ю. М.*, *Степанова Н. В.*, *Чернавский Д. С.* Математическое моделирование в биофизике. М.: Наука, 1975. 343 с.
4. *Рубин А. Б.*, *Пытьева Н. Ф.*, *Ризниченко Т. В.* Кинетика биологических процессов. М.: Изд. МГУ, 1977.
5. Физическая энциклопедия. М.: Советская энциклопедия, 988. Т. 1. С. 294–297.
6. *Шноль С. Э.* Общая биология. **34**, 331. 1972.
7. *Ростопшин Ю. А.*, *Климовицкий В. Я.* Возможные приложения биоэнергетики в моделировании физиологической терморегуляции. Биофизика. 1979. Т. XXIV. Вып. 5. С. 885–891.
8. *Ростопшин Ю. А.* Моделирование структурно-функционального состояния биологических систем // Моделирование процессов экологического развития. М.: ВНИИСИ, 1981. Вып. 2.
9. *Ростопшин Ю. А.* Методологические основы моделирования природной среды // Природа моделей и модели природы. М.: Мысль, 1996. С. 82–118.
10. *Ростопшин Ю. А.* Системно-методологические основы исследований устойчивого развития // Системные исследования. Методологические проблемы. Ежегодник 2003–2005. М.: КомКнига URSS, 2006. С. 104–131.
11. Stokes *Kenneth M.* Man and Biosphere: toward a co evolutionary political economy. N. Y.: M. E. Sharpe, 1992. 325 pp.

Разработка инструментальной системы распределенного имитационного моделирования*

Ю. И. Бродский, Ю. Н. Павловский

Вычислительный центр им. А. А. Дородницына РАН, Москва

Проектирование инструментальной системы распределенного имитационного моделирования неизбежно ставит перед разработчиком ряд вопросов, ответы на которые необходимо найти до начала практической реализации инструментальной системы. К таким вопросам относятся выбор класса моделей и их компонент, на которые будет ориентирована проектируемая инструментальная система моделирования; выбор архитектуры инструментальной системы; способы взаимодействия распределенных частей модели; концепция моделирования, предлагаемая исследователю инструментальной системой моделирования и, наконец, набор средств, предоставляемых проектируемой системой, для воплощения этой концепции. Наметить ответы на большинство из подобных вопросов и призвана предлагаемая ниже статья.

1. Компонента — основная составляющая распределенной модели

Основными понятиями данной работы будут понятия комплекса и компоненты. Это связанные между собой понятия, способные переходить друг в друга. Так, комплекс, внутренняя структура которого сложна и образована множеством компонент, в то же время извне может восприниматься как единая компонента. В то же время, у любой из компонент подробное исследование может выявить внутреннюю составную структуру, образующую комплекс.

* Работа выполнена при поддержке РФФИ, грант № 07–07–00071-а.

Попробуем в ближайших двух разделах определить, что же такое компонента модели. Основной абстракцией, которой мы будем оперировать далее, будет динамическая система

$$\vec{x}_{i+1} = \vec{F}(\vec{x}_i, \vec{a}_i, \Delta t), \tag{1}$$

где вектор $\vec{x}_i$ — набор внутренних или фазовых характеристик системы, вектор $\vec{a}_i$ — набор ее внешних характеристик или параметров, Δt — интервал модельного времени между двумя смежными шагами динамического процесса, i и $(i+1)$, $\vec{F}$ — некий алгоритм, детерминированный или стохастический, позволяющий для любого i по значениям $\vec{x}_i$, $\vec{a}_i$ и Δt однозначно определять реализацию фазовых характеристик $\vec{x}_{i+1}$ на следующем шаге динамического процесса. Существование такого алгоритма означает замкнутость системы, т. е. определенную ее независимость от внешнего мира, а именно, задавая $\vec{x}_0$ и $\vec{a}_i$, можно рассчитать внутренние характеристики системы в любой момент модельного времени. Здесь $\vec{x}_0$ — начальное состояние системы, вектор же внешних характеристик $\vec{a}_i$ на самом деле осуществляет связь нашей системы с внешним миром. Динамическую систему вида (1), будем называть моделью, в тех случаях, когда нужно подчеркнуть ее вхождение в качестве компоненты в более крупную модель, — подмоделью или компонентой, а в тех случаях, когда нужно подчеркнуть ее собственную многокомпонентную внутреннюю структуру — метамоделью или комплексом.

2. Реакция компоненты на события

Известно, что реальные сложные системы, являющиеся предметом моделирования, в зависимости от определенных сочетаний своих характеристик и характеристик окружающего мира, способны порой достаточно резко менять свое поведение. Для отражения этого их свойства введем формализованное определение события. Будем считать, что события могут происходить в самом начале интервала модельного времени Δt, при переходе системы от i-го к $(i+1)$-му состоянию. С моделью может быть связано несколько событий, событие k, $k = 1, \ldots, K$ состоит в том, что некая скалярная функция $\Phi_i(\vec{x}, \vec{a})$ обращается в ноль на наборе характеристик модели $\vec{x}_i$, $\vec{a}_i$ в начале очередного интервала моделирования: $\Phi_k(\vec{x}_i, \vec{a}_i) = 0$. Далее, можно считать, что имеется некоторое количество способов поведения модели, задающееся различными алгоритмами $\vec{F}_m(\vec{x}, \vec{a}, \Delta t)$, $m = 1, \ldots, M$, и однозначное отображение множества 2^K всевозможных реализаций всех связанных с моделью событий, во множество M различных алгоритмов ее поведения, которое и задает реакцию модели на произошедшее событие.

Заметим, что стартовав с системно-динамического описания компоненты модели, мы пришли к двойственному, объектно-ориентированному описанию: компонента, определенная выше, задает класс объектов со свойствами $\vec{x}_i$, $\vec{a}_i$, методами $\vec{F}_m(\vec{x}, \vec{a}, \Delta t)$, $m = 1, \ldots, M$, переключения между методами задаются событиями $\Phi_k(\vec{x}_i, \vec{a}_i) = 0$, $k = 1, \ldots, K$.

3. Анализ метамодели (комплекса)

Предположим, что мы исследуем модель некоторого явления, которая описывается динамической системой (1). В работе [2] было показано, что если алгоритм динамического процесса задается функционально, то при некоторых не слишком обременительных условиях, накладываемых на функцию $\vec{F}$, систему (1) с любой наперед заданной точностью можно заменить несколькими динамическими системами, просто разбивая вектор $\vec{x}_i$ на несколько подвекторов, и синхронизуя подсистемы между собою в точках событий. Далее, попытаемся проиллюстрировать еще один источник возникновения нескольких динамических систем, при углубленном изучении модели, первоначально описывавшейся одной системой.

Мы предполагаем замкнутость нашей модели, т. е. некоторую ее изолированность от внешнего мира, а именно, что зная векторы $\vec{x}_0$ и $\vec{a}$ мы можем рассчитать траекторию нашей системы на любом шаге. При этом о внешнем мире нам вроде бы ничего и не надо знать. На самом же деле, все в этом мире связано, и связь внешнего мира с нашей моделью осуществляется именно через внешние переменные $\vec{a}$ модели, а также через вид функции $\vec{F}$, который как «вещь в себе» никогда нам не известен, а всегда придуман исследователем и является синтезом его знаний о данном явлении и о его связях с внешним миром, на настоящем этапе исследования. Обычно функция $\vec{F}$ придумывается исследователем как некоторая комбинация «элементарных» (т. е. хорошо ему известных, ставших стандартными **элементами** его языка) функций, при этом компоненты вектора $\vec{a}$ — есть различные коэффициенты в этой комбинации. Или же функция $\vec{F}$ может реализовываться некоторым придуманным исследователем компьютерным алгоритмом, умеющим превращать векторы $\vec{x}_i$ и $\vec{a}$ в вектор $\vec{x}_{i+1}$. На начальных этапах моделирования компоненты $\vec{a}$ можно считать постоянными, или же меняющимися во времени по некоторому эмпирическому закону $\vec{a}(t)$. Однако, при более глубоком изучении явления, такой подход может перестать устраивать исследователя, могут, например, возникнуть некоторые соображения о природе закона изменения $\vec{a}(t)$, и как следствие — может возникнуть желание описать изменения всех компонент вектора $\vec{a}$ или же какой-либо их части также динамической системой $\vec{a}_{i+1} = \vec{\Psi}(\vec{a}_i, \vec{b}_i, \Delta t)$. В этой динамической системе $\vec{a}$ — уже

внутренние переменные, а внешними являются компоненты вектора $\vec{b}$. Если еще при этом окажется, что вектор $\vec{b}$, это на самом деле, по-другому названный вектор $\vec{x}$, а над моделью $\vec{a}_{i+1} = \vec{\Psi}(\vec{a}_i, \vec{b}_i, \Delta t)$ уже давно и успешно работает некий другой исследователь, то можно объединить усилия этих исследователей и построить к их общей пользе объединяющую две модели метамодель: $\vec{x}_{i+1} = \vec{F}(\vec{x}_i, \vec{a}_i, \Delta t)$, $\vec{a}_{i+1} = \vec{\Psi}(\vec{a}_i, \vec{b}_i, \Delta t)$.

Конечно, в реальной жизни такое полное и быстрое замыкание двух моделей получить трудно. Скорее всего, будут задействовано большее количество подмоделей и меньшее количество параметров.

4. Синтез метамодели (комплекса)

Предположим, наша модель описывается несколькими динамическими системами, как это было описано в предыдущем пункте. Возникает вопрос, как построить из них работающую метамодель.

Во-первых, необходим информационный синтез подмоделей, т. е. необходимо указать для каждой подмодели, какие ее внутренние переменные являются внешними переменными каких-либо, и каких именно подмоделей. И наоборот, какие из ее внешних переменных на самом деле явно моделируются какой-то (и какой именно) подмоделью. Будем считать, что мы располагаем N компонентами A_k, описываемыми динамическими системами $\vec{x}^k_{i+1} = \vec{F}^k(\vec{x}^k_i, \vec{a}^k_i, \Delta t)$, $k = 1, \ldots, N$. На практике вполне может оказаться, что результаты работы некоторых компонент (т. е. их внутренние переменные) определяют внешние переменные некоторых других компонент не непосредственно, а через определенные функциональные или алгоритмические зависимости, подобно тому, как это бывает в задаче наблюдения: мы не умеем непосредственно измерять интересующие нас величины, но умеем измерять некоторые другие, функционально связанные с первыми, и на основе измерений и зависимостей вычислять их. Однако, мы имеем право считать, что все подобные зависимости представлены в универсальной форме (1), т. е. в форме компонент, и, таким образом входят в число компонент нашего комплекса. Далее, для каждой из $N(N-1)$ пар (A_i, A_j), $i \neq j$, можно построить матрицу коммутации $Q_{i,j}$ размера $n_i \times m_j$, где n_i — размерность вектора $\vec{a}^i$ компоненты A_i, а m_j — размерность вектора $\vec{x}^j$ компоненты A_j. На пересечении p-й строки $(1 \leqslant p \leqslant n_i)$ и q-го столбца $(1 \leqslant q \leqslant m_j)$ этой матрицы стоит 1, если внешняя переменная a^i_p компоненты A_i явно моделируется внутренней переменной x^j_q компоненты A_j; и 0 — в противном случае. Задание матриц коммутации $Q_{i,j}$, $1 \leqslant i,\ j \leqslant N$; $i \neq j$ полностью решает вопрос информационного синтеза комплекса из компонент. Здесь следует

заметить, что при фиксированном i, и при $1 \leqslant j \leqslant N$; $i \neq j$, наличие единицы в фиксированной строке более чем в одной из матриц $Q_{i,j}$, также как и наличие в одной строке более одной единицы, свидетельствовало бы о моделировании одной величины несколькими способами, и, следовательно, ставило бы вопрос о согласованности таких моделей. Стало быть, подобная ситуация должна вызывать если и не сообщение о фатальной ошибке при коммутации компонент комплекса, то по крайней мере, очень серьезное предупреждение с призывом тщательно проверить согласованность компонент. С другой стороны, наличие нескольких единиц в столбце матрицы $Q_{i,j}$, а также, при фиксированном j, и при $1 \leqslant i \leqslant N$; $i \neq j$, наличие единицы в фиксированном столбце более чем одной из матриц $Q_{i,j}$, говорит о том, что внутренняя переменная одной из компонент используется в качестве внешней переменной более чем одной компоненты, что не противоречит нашей концепции, и должно быть разрешено. Следует также заметить, что поскольку, как следует из сказанного выше, единица — достаточно «редкий гость» в матрицах коммутации, то при описании комплекса на формальном языке, будет проще специальным оператором языка описывать только существующие коммутации, вместо выписывания всех матриц коммутации, которые должны автоматически строиться и проверяться на этапе компиляции описания комплекса.

Во-вторых, возникает вопрос о синхронизации во времени работы подмоделей. Предлагается следующий механизм синхронизации модельного времени компонент комплекса.

- Из соображений выбора масштаба и точности моделирования выбирается некий стандартный интервал моделирования Δt.
- На интервале моделирования Δt аппроксимируются значения векторов $\vec{x}(t), \vec{a}(t)$. Это может быть линейная аппроксимация по двум точкам $(\vec{x}_i, \vec{a}_i)$ и $(\vec{x}_{i+1}, \vec{a}_{i+1})$ на концах этого интервала, или аппроксимация по уже существующим точкам моделирования от начала динамического процесса, или же вычисление одной или нескольких промежуточных точек на интервале Δt (т. е., фактически, счет с меньшим шагом моделирования), и затем, аппроксимация одним из известных методов, по этим точкам. Выбор того или иного способа аппроксимации зависит от масштаба моделирования и желаемой точности модели.
- Далее, для каждого события каждой подмодели ищется решение t уравнения $\Phi_k(\vec{x}(t), \vec{a}(t)) = 0$, $k = 1, \ldots, K$ на интервале Δt. Если таких решений нет, в качестве следующего интервала моделирования снова выбирается стандартный интервал моделирования Δt, если же решения есть, то наименьшее из них будет правым концом

следующего интервала моделирования. Таким образом, интервал моделирования может уменьшаться, с тем, чтобы ни одна из подмоделей не «проскочила» очередное событие, которое может существенно повлиять на ее поведение.

- Необходимо избегать «дурной бесконечности» событий, руководствуясь принципом достаточности конечного числа событий на любом конечном интервале времени, сформулированном в [2], т. е. точек накопления событий на конечном интервале времени не должно быть.

В заключение следует отметить, что комплекс, изнутри 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. Методами комплекса объявляется объединение всех методов его компонент.
3. Событиями комплекса объявляется объединение всех событий его компонент.
4. Внешними переменными комплекса объявляется объединение всех внешних переменных его компонент, из которого исключаются все те переменные a_p^i, для которых p-я строка одной из матриц коммутации $Q_{i,j}$, $1 \leqslant i,\ j \leqslant N$; $i \neq j$ содержит единицу.

Мы видим, что тогда как изнутри комплекс имеет сложный многокомпонентный состав, извне он, в соответствии с данным в начале работы определением, вполне может быть воспринят как единая компонента. Единственное отличие от данного выше определения компоненты — возможность параллельного (т. е. одновременного) выполнения нескольких методов. Это отличие можно снять, например, разрешив компоненте одновременно выполнять несколько потоков (процессов), состоящих в чередовании некоторого числа выполняемых последовательно методов, как это было в MISS [1]. Так или иначе, определяющим остается то, что задавая для комплекса, рассматриваемого как единая компонента, начальные данные и значения внешних переменных, можно однозначно определить его внутренние переменные на любом шаге моделирования.

Таким образом, любая модель может быть представлена с одной стороны агрегировано, как единая компонента, а с другой стороны подробно, причем с произвольной степенью подробности, как комплекс компонент, в котором каждая из компонент тоже в свою очередь допускает представление в виде комплекса, тем самым, позволяя реализовывать концепцию мультимоделирования, т. е. построения семейства моделей разной степени подробности, для изучения одного и того же явления.

5. Общая архитектура инструментальной системы распределенного имитационного моделирования

Система распределенного имитационного моделирования представляется пиринговой сетью средней степени централизации, наподобие популярной сегодня в определенных кругах пользователей Интернета файлообменной сети e-Donkey.

- Несколько десятков серверов глобальной сети имитационного моделирования должны выполнять следующие функции обслуживания нескольких тысяч рабочих станций:
- Регистрация клиентских рабочих станций и предоставляемых ими в сеть разделяемых ресурсов, т. е. моделей, компонент моделей, данных, алгоритмов и т. д.
- Хранение и постоянное поддержание баз данных модельных ресурсов в сети, т. е. какими рабочими станциями предоставлены в сеть те или иные модели, компоненты моделей и прочие ресурсы. Какие из рабочих станций реально присутствует в сети, кто из них и когда был доступен в последний раз, возможно, расписаний присутствия рабочих станций в сети.
- Поиск ресурсов в сети, т. е. уже существующих моделей, компонент моделей, необходимых данных и алгоритмов (примерно так, как осуществляется поиск различных файлов в современных пиринговых файлообменных сетях).
- Осуществление функций посредников стандартных запросов рабочих станций.
- Осуществление некоторых организационных функций, например, возможности нескольким рабочим станциям договориться о проведении совместной сессии имитационных экспериментов определенной продолжительности, с началом в определенное время, или же спланировать время проведения такой сессии в соответствии с их расписаниями присутствия в сети.

Клиентское программное обеспечение рабочих станций такой сети вполне могло бы базироваться на описанных выше принципах анализа и синтеза модели. А именно, основу программного обеспечения рабочей станции составляет «система программирования» на некотором непроцедурном языке описания моделей и их компонент. Эта система программирования должна включать редактор (в том числе желательно и с графическим интерфейсом) описания моделей и их компонент. При описании компонент модели той или иной конструкции, должна быть

возможность ссылки на найденные уже существующие в сети компоненты, подходящие на роль компонент описываемой конструкции (наподобие $e2k$:-ссылок сети e-Donkey). Далее, система программирования клиентской части должна включать средства отладки описаний отдельных компонент и всей модели в целом. Далее, в клиентской части должен присутствовать сборщик модели, создающий базу характеристик компонент модели, проверяя при этом ее целостность.

Блок клиентской части инструментальной системы, ответственный за организацию имитационных экспериментов, должен с помощью существующих интеграционных технологий вызывать как локальные, так и удаленные компоненты модели по их известным адресам, передавая им необходимые характеристики из базы данных модели и принимая от них обратно в базу обработанные характеристики. Служебные методы, предоставляемые системой, также должны быть рассчитаны как на локальное, так и на удаленное использование. На этапе имитационного эксперимента рабочая станция, проводящая эксперимент, работает напрямую, минуя сервера, с рабочими станциями, содержащими используемые в модели компоненты. Как и в современных пиринговых сетях, серверы здесь нужны лишь на этапе поиска партнеров.

Уже отлаженные, работающие модели и их компоненты могут быть объявлены доступными для всеобщего использования, и пополнить фонд доступных в сети моделей и их компонент. При этом автоматически становятся доступными также и описания модели и ее компонент на языке описаний, по которым можно судить о ее адекватности тем или иным способам применения в других моделях. Таким образом, каждая рабочая станция может предоставлять в общее пользование, как свой вычислительный потенциал, так и существующие наработки в области моделирования.

Предложенная концепция могла бы применяться и более широко — как универсальное средство совместного использования в сети разнородных и разноплатформенных информационных и алгоритмических ресурсов, что могло бы существенно расширить круг ее потенциального применения. Тем самым у инструментальных систем имитационного моделирования появился бы шанс превратиться из экзотических программных продуктов предназначенных для узкого круга специалистов в области имитации, в массовое и универсальное средство объединения ресурсов Интернета для совместного выполнения широкого круга задач.

Предложенная концепция построения глобальной пиринговой сети имитационного моделирования может быть реализована на базе технологии IARnet, разработанной в Институте системного анализа РАН [3].

6. Как описываются комплексы и компоненты

Для описания комплексов и компонент необходим специализированный непроцедурный язык описаний классов комплексов и компонент. Одним из первых языков подобного рода был предложенный в [1] язык описания спецификаций MISS, затем в разное время обретали и теряли популярность такие языки, как IDL, UML, различные языки описания спецификаций на основе XML, наконец, Slice. В настоящее время в этой области наблюдается определенный разброд и шатание. Поэтому в данной работе мы не остановимся на каком-либо конкретном из перечисленных выше языков, тем более, что их возможности а стало быть, и сложность их реализации, существенно превосходят функциональные потребности данного проекта, а сконцентрируем внимание именно на этих функциональных потребностях.

Начнем с описания компоненты. Ее описание естественно разбить на несколько секций.

1. Секции описаний внутренних и внешних переменных достаточно традиционны, их можно сделать C-подобными или же Pascal-подобными.
2. Секция описания методов вычисляющих динамические процессы — просто перечисляет имена соответствующих методов.
3. Секция описания методов вычисляющих внешние переменные компоненты, заданные временными рядами — также перечисляет имена соответствующих методов.
4. Секция описания событий. В самом общем случае, проверка наступления события — это вызов соответствующего метода, возвращающего значение функции $\Phi_i(\vec{x}, \vec{a})$. Кроме этого в языке нужно предусмотреть возможность описания упрощенных событий, вида равенств или неравенств двух выражений от аргументов $\vec{x}$ и $\vec{a}$.
5. Секция описания переключения методов в зависимости от сочетаний наступивших событий. Здесь можно позаимствовать, например, синтаксические конструкции из описания переключений элементов приборов в языке спецификаций MISS [1].

Описание комплекса состоит из двух секций.

1. Секция описания состава комплекса. Здесь перечисляется, сколько экземпляров компонент какого класса входит в комплекс.
2. Секция коммутации. Здесь указывается, какие из внешних переменных, каких компонент, явно моделируются внутренними переменными других компонент. В качестве основы синтаксиса можно предложить оператор коммутации языка спецификаций MISS [1].

По описанию комплекса, согласно сформулированным выше правилам объединения компонент, может быть автоматически сгенерировано описание комплекса как компоненты.

7. Язык спецификаций

В данном разделе подробно разобран специализированный непроцедурный язык, на котором должны составляться спецификации классов компонент модели. Отметим, что хотя аккуратнее употреблять термин «спецификация класса компонент», мы ради краткости будем пользоваться словосочетанием «спецификация компоненты». Лингвистические формулы записываются ниже в стандартной нотации, согласно которой служебные слова выделяются жирным шрифтом, разделители — двойными кавычками, а необязательные включения квадратными или фигурными скобками, причем фигурные скобки указывают на возможность повторения. Разбор языка начнем с двух нетривиальных конструкций, применяемых в спецификациях всех типов. Это — определение структуры данных и оператор коммутации.

7.1. Определение типов записей фазовых переменных. Каталогизированные типы

Собирая модель, инструментальная система автоматически генерирует ее базу данных, руководствуясь при этом содержимым специальных параграфов спецификаций компонент. Речь идет о параграфах определений типов фазовых переменных и констант. Их синтаксис различается только открывающей ключевой конструкцией, а в остальном сводится к правилам определения структур данных. Формально определение структуры данных есть набор определений списков полей, т. е. — набор списков идентификаторов, завершающихся указаниями типов поименованных величин. При этом допускаются простые типы, структурные типы и тип массива. К простым относятся встроенные и каталогизированные типы, фиксируемые своими идентификаторами. Структурные типы — это типы обычных записей, ключей виртуальных записей и ключей виртуальных списков. Их указания выглядят как выделенные служебными словами наборы определений списков полей (здесь — рекурсия!). Наконец, диапазоны индексов допускаются лишь числовые и тип элемента массива должен быть простым или структурным типом.

В стандартной нотации определение структуры данных выглядит следующим образом:

```
ОпрСтруктДанных: ::= ОпрГруппыПолей { ОпрГруппыПолей }.
```



```
ОпрГруппыПолей ::= Имя { "." Имя } ":" УказаниеТипа .

УказаниеТипа ::= ИмяПростогоТипа ";" |
                 УказаниеСтруктурногоТипа |
                 УказаниеТипаМассива |
                 УказаниеТипаПеречисления.

ИмяПростогоТипа ::= ИмяВстроенногоТипа |
                    ИмяКаталогизированногоТипа.

ИмяВстроенногоТипа ::= BOOLEAN | CHAR | BYTE | CARDINAL |
                       BITSET | INTEGER | LONGINT | REAL |
                       INDEFINITE | TEXT | NAME | PICTURE |
                       BACKGRND.

ИмяКаталогизированногоТипа  ::= Имя

УказаниеСтруктурногоТипа ::= ИмяСтруктурногоТипа
                              ОпрСтруктДанных {END ";"}.

ИмяСтруктурногоТипа ::= RECORD | LIST OF | BLOCKS OF |
                         ADDRESS OF.

УказаниеТипаМассива ::= ARRAY Диапазон { "," Диапазон }
                        OF ТипЭлемента.

Диапазон :: = " [" ЦелоеЧисло ".." ЦелоеЧисло " ]".

ТипЭлемента ::= ИмяПростогоТипа ";" |
                УказаниеСтруктурногоТипа.

УказаниеТипаПеречисления :: = " (" Имя { "," Имя } ");''.
```

В этих формулах термин `Имя` означает любую последовательность символов без пробелов длиной не более 20, не содержащую использованных в контексте разделителей.

Понятие каталогизированного типа и специфичные встроенные типы `INDEFINITE`, `TEXT`, `NAME`, `PICTURE`, `BACKGRND`, `LIST OF`, `BLOCKS OF`, `ADDRESS OF` будут разобраны ниже. Прочие встроенные типы определяются таблицей соответствий (табл. 1).

Чтобы была возможность использовать в разных спецификациях одни и те же структурные типы данных, введен аппарат каталогизированных

Таблица 1

Тип в инструментальной системе	Тип в МОДУЛЕ-2	Тип в языке С	Длина в байтах
BOOLEAN	BOOLEAN	Char	1
CHAR	CHAR	Char	1
BYTE	BYTE	Char	1
перечисление	перечисление		1
BITSET	BITSET	Unsigned	2
CARDINAL	CARDINAL	Unsigned	2
INTEGER	INTEGER	int	2
LONGINT	LONGINT	long	4
REAL	LONGREAL	double	8

типов: в библиотеке системы, наряду со спецификациями групп, объектов и приборов, можно заводить в специально выделенной секции спецификации структурных типов данных, тем самым каталогизируя их. Синтаксис спецификации каталогизированного типа таков:

```
ОписательКаталогТипа. :: = TYPE Имя ";" ОпрСтруктДанных
                           {END ";"}.
```

Смысл употребленных в правой части терминов тот же, что в предыдущей серии определений, причем Имя — и есть то имя, под которым данный тип будет зарегистрирован в библиотеке, и под которым его можно упоминать в других спецификациях. Отметим, что приведенное определение рекурсивно, т. е. допускает использование в спецификации каталогизированного типа имен иных типов того же сорта. Циклов в этой рекурсии быть не должно, и система не допустит их, отказываясь компилировать спецификацию, если еще не заведены или не откомпилированы спецификации всех упоминаемых в ней каталогизированных типов.

Примеры спецификаций каталогизированных типов:

```
TYPE VECTOR;
       X, Y, Z : REAL;
     END;
```



```
TYPE MATRIX;
       Rows : ARRAY [1..3] OF VECTOR;
     END;
```

С каталогизацией непосредственно связан встроенный тип INDEFINITE. Он означает, что соответствующие переменные могут быть записями любого из каталогизированных типов. Это — принятый в системе способ поддержки варьируемых структур данных.

Именами TEXT и NAME помечаются динамические переменные текстовой природы. Первое относится к текстам произвольной длины, хранимым в специальном формате. Основное их назначение служить развернутыми комментариями к спецификациям.

В номенклатуре встроенных типов есть два графических. Это — PICTURE и BACKGRND. Оба определяют ключи неких картинок, формально являющихся динамическими записями особой структуры. Первый тип относится к маленьким картинкам-пиктограммам. Второй — тип ключа полномерной картинки. Переменные обоих типов предназначаются для графической анимализации моделей. Использование их в программах имитации в качестве фактических параметров соответствующих инструментальных процедур позволяет создавать «мультфильмы» эволюции воспроизводимых процессов. Инициализация этих переменных, включающая и создание ассоциируемых с ними изображений, обычно должна осуществляется через соответствующие графические редакторы, но может выполняться и программно.

Следует упомянуть и три еще не разобранных динамических типа. Типы BLOCKS OF и LIST OF относятся к ключам блокированных и неблокированных списков из записей одной длины, а тип ADDRESS OF — к ключам одиночных динамических записей. Структура записей всегда фиксируется текстом, следующим за именем типа.

При инициализации все динамические структуры размещаются в копии базы данных, создаваемой системой на время сеанса в виртуальной памяти. По завершении сеанса эту копию можно сохранить, и тогда сохранятся все заведенные данные; тем самым будет обеспечена возможность их использования в последующих сеансах. Пример определения структуры данных:

```
numb : CARDINAL; vect : VECTOR; cards : LIST OF
  card : RECORD
  name : (knave, queen, king);
  pict : PICTURE;
END;
color : (spades, leaves, diamonds, harts);
trump : BOOLEAN; END;
```

7.2. Операторы коммутации

Согласно принятой концепции, часть параметров компонент комплекса может явно моделироваться в других его компонентах, т. е. являться частью фазовых переменных этих компонент. Комплексу также разрешено иметь собственные параметры и фазовые переменные. Для описания возможных связей параметров с фазовыми переменными используются операторы коммутации. Реально стыковка параметров и фазовых переменных, а также проверка возможности такой стыковки осуществляется во время сборки модели.

Общий вид оператора коммутации таков:

```
ОператорКоммутации :: = {ОпределительДиапазона}АдресПараметра
                         "=" АдресФазовойПеременной ";".
```

Один оператор определяет один фрагмент связи, причем четко фиксируется направление связи — от контакта, указанного за равенством, к контакту, указанному перед ним. Синтаксис адресов контактов в операторах коммутации определяется формулами вида:

```
АдресПараметра :: = {АдресКомпоненты.}ИмяПараметра
                     {ИндексПараметра}

АдресКомпоненты :: = ИмяКомпоненты " (" ИндексКомпоненты ")"

АдресФазовойПеременной :: = {АдресКомпоненты"." }
                              ИмяФазовойПеременной
                              {ИндексФазовойПеременной}
```

Если адрес контакта сводится к его индексу, то это значит, что контакт принадлежит внешнему разъему описываемой компоненты модели. ИмяПодгруппы, ИмяОбъекта и ИмяПрибора — суть идентификаторы соответствующих классов компонент. Служебное слово INT помечает внутренние разъемы приборов. Термины ИндексПараметра, ИндексФазовойПеременной формально расшифровываются одинаково:

```
ИндексПараметра, ИндексФазовойПеременной ::=
                                "(" ПростоеЦелоеВыражение ")".

ПростоеЦелоеВыражение ::= ЦелоеБезЗнака |
           [ Целое Знак ] "i*" ЦелоеБезЗнака |
           [ Целое Знак ] ЦелоеБезЗнака "*i" |
           [ Знак ] "i*" ЦелоеБезЗнака Знак ЦелоеБезЗнака |
           [ Знак ] ЦелоеБезЗнака "*i" Знак ЦелоеБезЗнака.

Знак ::= "+" | "-".
```


Во второй формуле все варианты, кроме первого, относятся к случаю, когда записи оператора коммутации предшествует

```
ОпределительДиапазона  ::= "i=" Целое ".." Целое ":".
```

Это — конструкция, позволяющая одной записью оператора коммутации определить целую группу фрагментов каналов связи, а каких именно — ясно из контекста. Осталось только подчеркнуть, что указываемые в квадратных скобках номера подгрупп и объектов или приборов являются порядковыми во внутренней индексации той группы или объекта, в чей описатель войдет оператор. Примеры операторов коммутации:

```
i = 1..10 :
ОбъектПервогоТипа (i).МассивПараметров(2*i-1) =
     ОбъектВторогоТипа (10-i).МассивФазовыхПеременных(2*i);
A1=Спутник.X; Объект(0).x=x;
```

7.3. Синтаксис описателей комплексов

Двух предыдущих подразделов достаточно, чтобы без подробных дополнительных разъяснений определить принятые правила формирования спецификаций комплексов и компонент. Формальный синтаксис спецификации комплексов таков:

```
СпецификацияГруппы ::= COMPLEX ИмяКомплекса  ";"
        ПараграфКомпонент [ПараграфФазовыхПеременных ]
        [ПараграфКонстант ][ПараграфПараметров]
        [ПараграфКоммутации ].

ПараграфКомпонент ::=  COMPONENTS
                       ПереченьСоставляющих;
                       { ПереченьСоставляющих }{END";"}

ПараграфФазовыхПеременных ::= PHASE
                       ОпрСтруктДанных {END ";"}

ПараграфПараметров ::= PARAMETERS
                       ОпрСтруктДанных {END ";"}

ПараграфКонстант ::= CONST
                       ОпрСтруктДанных {END ";"}

ПараграфКоммутации ::= COMMUTATION
          ГруппаКоммутации {ГруппаКоммутации.} {END ";"}.
```

Во всех параграфах, кроме последнего, в комплексе END'ы необязательны. Заголовок следующего параграфа автоматически означает конец предыдущего.

Смысл использованного здесь термина ОпрСтруктДанных разобран в предыдущем разделе, а несложные формулы определений других новых терминов выглядят следующим образом:

```
ПеречеңьСоставляющих ::= ИмяКомпоненты
    "(" ЧислоЭкземпляров ")"
    "," ИмяКомпоненты "(" ЧислоЭкземпляров ")"  ";".

ГруппаКоммутации ::= [ ОпределительДиапазона ]
                     ОператорКоммутации.
```

В этих формулах ИмяКомплекса и ИмяКомпоненты суть имена классов компонент соответствующего типа, а формально — идентификаторы длиной не более 20 символов; ЧислоЭкземпляров — положительное целое, равное числу составляющих указанного перед скобкой класса; ОпределительДиапазона, ОператорыКоммутации — термины из предыдущего раздела.

7.4. Синтаксис описателей компонент

Формула спецификации компоненты такова:

```
ОписательКомпоненты :: = COMPONENT ИмяКомпоненты ";"
                         [ ПараграфФазовыхПеременных ]
                         [ ПараграфПараметров ]
                         [ ПараграфКонстант ]
                         [ ПараграфМетодов ]
                         [ ПараграфСобытий ]
                         [ ПараграфПереключателей ].
```

Как и в случае комплексов, END обязателен лишь в самом последнем параграфе. В параграфе методов спецификации компоненты одним или несколькими потоками перечисляются все выполняемые в данном потоке методы, начиная с корневого, с которого по умолчанию будет начинаться модельная жизнь каждого потока. Имена методов должны быть уникальными в пределах спецификации: нельзя одинаково назвать два разнотипных метода одной компоненты, но использование одного имени в спецификациях разных компонент не возбраняется. Формально синтаксис параграфа методов определяется так:

```
ПараграфМетодов :: = ПотокМетодов |
                     ПараграфМетодов ПотокМетодов {END ";"}.
```



```
ПотокМетодов ::= ЭлементСписка  {"," ЭлементСписка}  ";".

ЭлементСписка ::= ИмяМетода | ИмяМетода "(" FAST ")" |
                  ИмяЭлемента"(" CONV ")"
```

Здесь ИмяЭлемента — идентификатор длиной не более 20 символов, а ключевые слова FAST и CONV помечают сосредоточенные и условно распределенные методы.

Параграф событий определяет набор событий, связанных с данной компонентой. Определяется идентификатор события и условие его наступления.

```
ПараграфСобытий ::= ОписаниеСобытия  {";" ОписаниеСобытия}
                    ";" {END ";"}.

ОписаниеСобытия ::= ИмяСобытия ":" ЛогическоеВыражение ";".

ЛогическоеВыражение ::=
              ЛогическийОперанд | "^" ЛогическоеВыражение |
              ЛогическоеВыражение "&" ЛогическоеВыражение |
              ЛогическоеВыражение "|" ЛогическоеВыражение.

ЛогическийОперанд ::= ЛогическаяКонстанта |
              АрифметическоеВыражение ЗнакСравнения
                    АрифметическоеВыражение .

ЗнакСравнения :: = > | < | # | >= | <=.

АрифметическоеВыражение :: = АрифметическийОперанд |
    "(" АрифметическоеВыражение ")" |
    АрифметическоеВыражение ЗнакАрифмОперации
        АрифметическоеВыражение .

ЗнакАрифмОперации :: = + | -| * | ^ .

АрифметическийОперанд :: = Число | МетодФункция |
                            Параметр | ФазоваяПеременная .
```

Например:

```
Хватит : (Пешеход(0).x-Пешеход(1).y)\ 2<0,01;
```

Параграф переключателей определяет автоматную функцию потока. В нем для каждого элемента каждого потока должны быть перечислены

все разрешенные переходы, т. е. названы все элементы, на которые прибор может переключаться по завершении данного, и указаны события, в зависимости от которых реализуется то или иное переключение. Формат параграфа таков:

```
ПараграфПереключателей ::= SWITCHES АвтоматнаяФункция
                           {END ";"}

АвтоматнаяФункция  ::= ПереключенияЭлемента
                       {ПереключенияЭлемента} .

ПереключенияЭлемента :: = ИмяЭлемента ":" СписокПереходов.

СписокПереходов ::= {ИмяЭлемента "," ЗаписьСобытия "."}
                    ИмяЭлемента ";".
```

Понятно, что перед двоеточием во второй формуле стоит имя того элемента, переходы с которого определяет следующая за двоеточием конструкция. В последней перечисляются возможные переключения. Они упорядочиваются по старшинству событий, причем завершается перечень именем элемента, при котором не стоит никакого события. Это значит, что если не реализуется ни одно из вошедших в СписокПереходов событий, то произойдет переключение на элемент с указанным именем.

Чтобы полностью описать синтаксис параграфа автоматной функции, осталось уточнить формат, в котором представляется ЗаписьСобытия. Это — дизъюнктивная нормальная форма от булевых переменных, фиксирующих приход сигналов на внутренний входной разъем прибора. Принятие ею значения TRUE есть признак реализации события. В качестве имен булевых индикаторов сигналов используются номера соответствующих контактов внутреннего входного разъема. Знаком дизъюнкции служит "+", конъюнкции — "*" а отрицания — "^". Итак,

```
ЗаписьСобытия  ::= ФрагментСобытия  "|" ФрагментСобытия .

ФрагментСобытия ::= КонъюктивныйФрагмент | Терм.

КонъюктивныйФрагмент  ::= "(" Терм  {"&" Терм}  ")".

Терм ::= [ "^" ] ИмяСобытия [ ".." ИмяСобытия ].

ИмяСобытия ::= Идентификатор | ЦелоеБезЗнака.
```

Пример записи события:

```
C1 + \ C2 + C4..C6 + ( C7..C9 & C11 &  C13 ).
```


Смысл конструкций вида n1, . . . ,n2, допускаемых четвертой формулой, зависит от контекста: если такая конструкция стоит в конъюнктивном фрагменте, то ее следует понимать как конъюнкцию всех событий с номерами от n1 до n2 включительно, иначе — как их дизъюнкцию. Символ отрицания перед такой конструкцией не меняет ее конъюнктивного или дизъюнктивного смысла и должен трактоваться как эквивалент отрицания каждого из соответствующих сигналов. Так, используя знак равенства как символ эквивалентности, можно написать, что

```
+ \ 2..4 = +\ 2 +\ 3 +\ 4   и   * \ 2..4 = *\ 2 *\ 3 *\ 4.
```

В качестве примера опишем следующую модель: два пешехода идут навстречу друг другу с постоянными скоростями. Между ними летает муха также с постоянной скоростью, превосходящей скорость любого из пешеходов. Долетая до одного из пешеходов, муха тут же разворачивается навстречу другому.

```
COMPLEX ПешеходыИмуха;
COMPONENTS Пешеход(2), Муха(1);
COMMUTATION Муха(0).x1=Пешеход(0).x;
            Муха(0).x2=Пешеход(1).x;
            END
COMPONENT Пешеход; PHASE x, v : REAL;
METHODS Move; END;
COMPONENT Муха; PHASE x, v : REAL;
PARAMETERS x1, x2 : REAL;
METHODS MoveLeft, MoveRight, HALT; /* самый левый метод
        считается корневым в потоке*/
EVENTS ДолетДоЛевого : x=x1;
       ДолетДоПравого : x=x2;
       Хватит : x2-x1<0,01;
SWITCHES Хватит : HALT;
         ДолетДоЛевого : MoveRight;
         ДолетДоПравого : MoveLeft;
END;
```

Приведем также описание комплекса ПешеходыИмуха как компоненты. Как уже говорилось выше, такое описание должно генерироваться автоматически по описаниям комплекса и его компонент:

```
COMPONENT ПешеходыИмуха; PHASE x0, v0 : REAL;
                         x1, v1 : REAL; x2, v2 : REAL;
METHODS Move; /* методы разных параллельных потоков
                 разделяются точкой с запятой */
```

```
MoveLeft, MoveRight, HALT; /* методы одного
                              потока разделяются запятой */
EVENTS ДолетДоЛевого : x0=x1;
       ДолетДоПравого : x0=x2;
       Хватит : x2-x1<0,01;
SWITCHES Хватит : HALT;
         ДолетДоЛевого : MoveRight;
         ДолетДоПравого : MoveLeft;
END;
```

8. Как строится распределенная имитационная модель

Отправной точкой для построения модели является описание комплекса на языке описания спецификаций. Компиляция этого описания подразумевает нахождение описаний всех входящих в комплекс компонент, построение баз данных для хранения характеристик компонент, осуществление коммутации между компонентами. При этом также компилируются все компоненты, которые на самом деле являются комплексами. На этапе компиляции обнаруживаются различные ошибки в описаниях, например, отсутствие описаний тех или иных компонент, несоответствие типов при коммутации компонент комплекса. Результатом успешной компиляции будет являться созданная незаполненная база данных комплекса и полное описание комплекса как компоненты.

Созданная база данных может быть заполнена начальными данными модели вручную, как это было в MISS [1], или же могут быть предложены какие-либо средства автоматизации этого процесса, например, операторы инициализации переменных в языке описания спецификаций.

База данных комплекса всегда создается на локальной рабочей станции. Распределению в сети подлежат отдельные компоненты комплекса, а так как все их характеристики хранятся в базе данных на локальной рабочей станции, фактически распределенными могут быть методы, как динамических процессов, так и вычисляющие внешние переменные (в том числе и статические внешние данные модели). Распределенными могут быть и компоненты комплекса, которые сами являются комплексами. В последнем случае, на удаленной рабочей станции не просто вызывается метод, а запускается процесс моделирования, все данные которого хранятся в базе данных локальной машины. Связь между распределенными в сети рабочими станциями осуществляется средствами выбранной интеграционной технологии.

9. Организация вычислительного процесса

Для понимания хода вычислительного процесса, полезно представление комплекса, как единой компоненты.

1. В цикле вызываются текущие методы всех протекающих параллельно потоков (процессов). В результате чего вычисляются фазовые характеристики модели следующего шага.
2. Выбранным способом аппроксимации приближаются значение фазовых и внешних переменных внутри шага модели. Проверяется наступление событий.
3. Если событий нет — переход к следующему шагу моделирования стандартной продолжительности.
4. Если события есть — правым концом интервала моделирования становится время ближайшего события. Фазовым и внешним переменным следующего шага моделирования назначаются их аппроксимированные значения, соответствующие этому моменту времени. Переход к следующему шагу моделирования.

10. Работа с моделью во время имитационного эксперимента

Предполагается возможность в любой момент приостановить имитационный эксперимент, после чего исследователь может работать с характеристиками модели средствами применяемой СУБД. После работы с базой данных возможно возобновление приостановленного имитационного эксперимента.

Литература

1. *Бродский Ю. И.*, *Лебедев В. Ю.* Инструментальная система имитации MISS. М.: ВЦ АН СССР, 1991. 180 с.
2. *Павловский Ю. Н.*, *Белотелов Н. В.*, *Бродский Ю. И.* Имитационное моделирование. М.: Академия, 2008. 236 с.
3. *Афанасьев А. П.*, *Волошинов В. В.*, *Рогов С. В.*, *Сухорослов О. В.* Развитие концепции распределенных вычислительных сред // Проблемы вычислений в распределенной среде: организация вычислений в глобальных сетях. Труды ИСА РАН. М.: РОХОС, 2004. С. 6–105.