АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ В МИКРОЭЛЕКТРОНИКЕ.
ТЕОРИЯ, МЕТОДЫ, АЛГОРИТМЫ
(Вычислительные системы)
1978 год Выпуск 77

УДК 519.682.6

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ОСНОВНЫХ ЯЗЫКОВ ДИСКРЕТНОГО МОДЕЛИРОВАНИЯ

Н.И.Дубовская

Необходимость исследования сложных реальных процессов, имеющих место в различных системах - экономических, социальных, технических и других - вызвала развитие нового направления анализа таких систем - моделирования.

Метод моделирования, представляющий собой аппарат анализа систем многофакторных, динамических и стохастических, является весьма перспективным при решении ряда задач в области научно-технического прогнозирования, связанных с изучением свойств и поведения сложных объектов в различных условиях в течение продолжительных периодов времени. Подобные задачи возникают при разработке прогнозов и программ развития отрасли, выборе стратегии управления динамическими системами, проектировании и размещении предприятий, оценке перспективности технических изделий и др.

При создании моделей исследуемых объектов, реализуемых на цифровых вычислительных машинах, как правило, применяется дискретный подход, при котором непрерывные явления реальной физической системы представляются последовательностью дискретных изменений - событий.

Эффективность моделирования в значительной степени определяется выбором соответствующего языка описания и реализации модели. Язык моделирования должен служить удобным инструментом в процессе разработки моделей и отвечать следующим требованиям [1,2].

1. Концептуальная целостность: терминология языка и его структура должны способствовать пониманию и описанию с единой точки зрения различных систем, изучаемых моделированием.

2. Язык должен дать средства для разбиения сложных и больших по объему систем на легко обозримые компоненты, каждая из которых могла бы быть независимо описана отдельной программой.

3. Наличие средств представления упорядоченных во времени событий (введение понятия системного времени, средств моделирования параллельно протекающих процессов).

4. Возможность работы с такими сложными структурами данных, как деревья, списки и наборы.

5. Наличие средств имитации случайных факторов и воздействий - датчика случайных чисел и процедур генерирования случайных величин с различными законами распределений.

6. Средства обработки результатов экспериментов - вычисления статистических характеристик, накопления и вывода гистограмм и др.

7. Возможность формального описания вычислительных процессов (наличие общеалгоритмических средств).

К настоящему времени имеется достаточно большое число языков дискретного моделирования. При этом большинство из них повторяют в том или ином виде черты наиболее известных и признанных языков дискретного моделирования, сравнительному анализу которых и посвящена данная статья. Такими языками являются: GPSS [3], СИМСКРИПТ [4], CSL [5], SOL [6], СИМУЛА [7]. Весьма интересным и перспективным языком является СИМУЛА-67 [8].

Основой перечисленных языков, как и других языков моделирования, является управляющая программа, определяющая порядок выполнения отдельных функциональных блоков модели. Эти блоки в различных языках моделирования носят названия событий, работ или процессов. Общую структуру имитационной программы можно представить в виде схемы, показанной на рис. I.

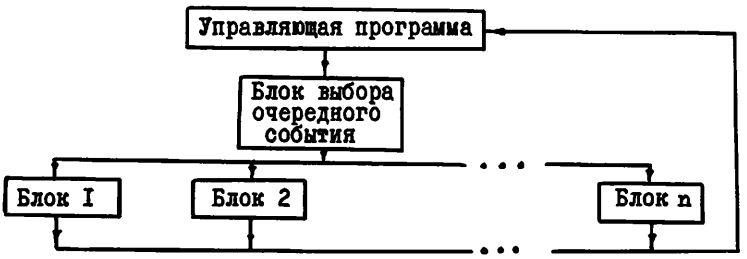

Рис. I

В каждом языке моделирования предусмотрена своя структура управляющей программы, соответствующая содержанию функциональных блоков модели,и в этом заключается принципиальное различие языков при организации динамического имитационного процесса.

Существуют три подхода к описанию динамики системы:

I) поисковый метод выбора очередного события,

2) использование схемы расписания событий,

3) описание системы на языке взаимодействия параллельных процессов.

Поисковый метод выбора очередного события применяется в языке CSL . Имитационная программа строится таким образом, чтобы каждый ее блок соответствовал некоторой работе. Работа описывается программой, состоящей из условия и тела (контрольного и функционального блоков). В контрольном блоке перечислены все условия, при соблюдении которых может быть выполнена программа функционального блока, описывающая необходимые изменения элементов системы. При этом изменения в данных привязываются к так называемому локальному времени, фиксирующему конец данной работы,или, иными словами, момент завершения перехода между состояниями ее элементов. Смена же состояний (событие) в любой динамической модели дискретных событий происходит мгновенно.

Управляющая программа последовательно просматривает программы всех работ. Если некоторая работа может быть выполнена (соблюдаются все условия контрольного блока), то управление передается ее функциональному блоку. При невыполнении хотя бы одного условия, обращения к функциональному блоку не происходит. Управляющая программа продолжает последовательно анализировать все работы,не изменяя системного времени. Реализовав все переключения данного момента, управляющая программа переводит системное время на ближайшее локальное время. Остальные локальные часы будут показывать время,большее или в крайнем случае равное выбранному. После этого управляющая программа вновь начинает циклически анализировать возможности различных переключений.

Из изложенного видно, что поисковый метод выбора очередного события, применяемый в CSL- языке работ, является не эффективным с точки зрения времени функционирования модели. Тем не менее применение языка может быть целесообразным для описания систем с большим числом различных ресурсов, так как позволяет задавать в лаконичной форме.

В языке СИМСКРИПТ реализован второй подход - использование схемы расписания событий. Это язык другого класса, чем GSL . СИМСКРИПТ - язык событий. Модель системы строится таким образом, что каждый ее блок соответствует некоторому событию. Каждый вид события перечисляется в "списке событий", на основании которого автоматически генерируется главная программа-календарь (управляющая программа). Эта программа следит за ходом системного времени и вызывает различные программы событий. В каждой программе события содержатся операторы, определяющие смену состояний элементов модели, и операторы, планирующие другие события в некоторые моменты системного времени, в том числе данное событие. События могут наступать в любой требуемый момент системного времени. Когда выполнение определенной программы события заканчивается, системное время немедленно приравнивается к моменту времени следующего наиболее близкого события.

Таким образом, при использовании схемы расписания событий удается избежать перебора, но логическая схема модели получается достаточно дробленной, так как для каждого вида событий должна быть написана отдельная программа.

Третий подход к организации динамической имитационной модели - описание системы на языке взаимодействия параллельных процессов. К языкам процессов относятся языки GPSS, SOL и СИМУЛА. Этот подход представляет собой шаг к объединению вычислительной эффективности языка событий и компактности описания модели на языке работ.

Каждый блок теперь соответствует процессу. Процесс характеризуется некоторой структурой данных и правилом действий. Каждый процесс может быть активным на некоторых этапах своего существования в системе и пассивным на других этапах. У процесса может быть несколько точек связи (точек реактивации) с другими процессами и несколько активных фаз. Динамика системы описывается последовательностью мгновенных событий, причем каждое событие является активной фазой некоторого процесса. Это отличает язык процессов от языков работ и событий. Данная работа или событие может взаимодействовать с другими работами или событиями только после того, как выполнены все относящиеся к ней операции. Программа работы или события начинается с одного и того же выполняемого оператора, для процесса же существуют точки реактивации, которые определяют те места в программе процесса, с которых надо ее продолжить после

выполнения прерывающих команд. Такими прерывающими командами являются специальные операторы управления типа ЗАДЕРЖКА, ЖДАТЬ и ЖДАТЬ ПОКА.

В языках моделирования для управления событиями существуют средства двух типов - императивного и интеррогативного управлений. Операторы первого типа определяют, какое событие должно произойти и когда, например:

"вызвать событие А в момент времени Т".

Операторы интеррогативного управления требуют выяснения сформулированных условий:

"вызвать событие А при выполнении условия Р".

Операторы ЗАДЕРЖКА,ЖДАТЬ являются операторами императивного, а ЖДАТЬ ПОКА - интеррогативного управления.

Ниже приведена классификация рассмотренных языков по принятой схеме описания динамики системы и типам управления ("I" указывает на соответствие языка данной схеме и типу управления).

Язык	Классификация				
	Язык работ	Язык событий	Язык процессов	Императивное управление	Интеррогативное управление
CSL	I	0	0	0	I
СИМСКРИПТ	0	I	0	I	0
GPSS	0	0	I	I	I
SOL	0	0	I	I	I
СИМУЛА	0	0	I	I	0

Языки GPSS и SOL интересны тем, что в них реализованы операторы как императивного, так и интеррогативного управления. Средством формализации динамики системы в этих языках служит элемент потока-транзакт, функциональный цикл которого включает перемещение от одного стационарного блока модели к другому. В обоих языках предварительно уже заложено определение таких стационарных устройств, как "средство обслуживания" и "хранилище". GPSS и SOL удобны для описания динамики функционирования систем массового обслуживания. В языке SOL, помимо понятия транзакта (процесса),имеется также понятие класса процессов. Как GPSS , так и SOL представляют собой языковые системы интерпретирующего типа.

В нашей стране и за рубежом большую популярность получил язык СИМУЛА. Этот язык основан на АЛГОЛе и содержит последний в качестве своего подмножества. Основное расширение АЛГОЛа заключается во введении средств моделирования параллельно протекающих процессов, средств обработки списков и набора библиотечных процедур случайного выбора и анализа данных. Программа, написанная на языке СИМУЛА, компилируется в рабочую программу.

Приведем кратко основные понятия языка СИМУЛА и средства оперирования с ними. Итак, система с дискретными событиями рассматривается как совокупность процессов, последовательность действий которых полностью описывает её функционирование. Количество процессов может быть постоянным или переменным, они поступают в систему и покидают ее в результате действий, выполняемых внутри самой системы. Процессы с аналогичной структурой данных и одинаковой схемой поведения объединяются в классы, называемые деятельностями.

Описание класса процессов имеет следующий вид:

activity P (A,B,...,E); спецификации A,B,...,E;
begin объявление атрибутов; правило действий end .

Здесь P - имя деятельности; A,B,...,E - формальные параметры, значения которых определяются для каждого отдельного процесса во время его генерирования. Формальные параметры носят названия экзогенных признаков. Величины, локализованные в самом внешнем блоке тела деятельности, называются эндогенными признаками.

Тело правила действий может быть пустым, и тогда соответствующий класс описывает некоторую структуру данных, т.е. является пассивным носителем информации.

Для возможности хранения и обработки групп процессов в языке имеется понятие набора - set . Набор может состоять из любой смеси процессов различных классов и динамически изменяться в ходе работы системы. Фактическое содержание набора составляют ссылки на процессы (название процессов), а не сами процессы. Отдельное название процесса называется элементом.

В языке вводится также понятие механизма присоединения, обеспечивающего взаимодействие между процессами. Некоторый процесс во время своей активной фазы может "присоединять" другие процессы, получая доступ к их признакам. Присоединяющий оператор имеет следующий вид:

inspect X when A_1 do S_1
...
when A_k do S_k
...
when A_n do S_n
otherwise S,

где $A_1,\ldots,A_n$ – идентификаторы классов процессов; $S_1,\ldots,S_n$, S – операторы. Если X – процесс, принадлежащий классу A_k, то выполняется оператор S_k, а остальные игнорируются. Конструкция when A_k do S_k называется присоединением. Таких конструкций может быть любое число, но не менее одной. Конструкция otherwise может отсутствовать. Каждый из операторов S_k называется присоединяющим блоком. Он считается непосредственно вложенным в самый внешний блок деятельности A_k, в силу чего экзогенные и эндогенные признаки присоединенного процесса становятся доступными по своим локальным наименованиям. Блок, содержащий присоединяющий оператор, действует в качестве внешнего блока.

Понятия элемента и набора вместе с механизмом присоединения делают язык СИМУЛА удобным для обработки списков.

Рассмотрим имеющиеся в языке средства для управления событиями. К ним прежде всего относится ряд планирующих операторов, порождающих уведомления о событиях. Во время активной фазы процесс может "запланировать" себе или другому процессу последующую активную фазу – событие. Например, оператор activate X at T планирует очередную активную фазу процесса X на момент T системного времени. При этом порождается "уведомление о событии", которое включается в управляющий список. Уведомление о событии содержит ссылку на время системы, когда оно должно произойти, и ссылку на процесс, который должен стать при этом активным. Все уведомления о событиях упорядочены в управляющем списке по возрастающим значениям системного времени. Уведомление о событии, находящееся в начале списка, указывает на текущий процесс, активный в данный момент. Когда текущее событие заканчивается, уведомление о нем исключается из списка и новым текущим событием становится очередная активная фаза процесса, на который ссылается следующее по порядку уведомление.

Планирующий оператор может иметь вид activate X delay t, т.е. запланировать очередное событие для процесса X через интервал времени t относительно текущего системного времени.

Оператор hold(t) отменяет текущее событие, планируя для того же процесса новую активную фазу через интервал t.

Возможно планирование с указанием приоритета: activate X at T prior или activate X delay t prior. В этом случае уведомление о новом событии помещается в управляющий список перед другими уведомлениями с такой же временной ссылкой.

Имеются операторы вида

activate X before Y

activate X after Y .

Уведомление о событии, порождаемое для процесса X , включается в управляющий список непосредственно до или после уведомления, ссылающегося на процесс Y . Оно получает ту же временную ссылку, что и последнее.

Непосредственное планирование осуществляется с помощью оператора activate X . Событие планируется перед текущим событием, так что указанная активная фаза исполняется немедленно. Текущая активная фаза заканчивается, но уведомление о ней остается в управляющем списке в качестве очередного события.

Операторы, имеющиеся в языке СИМУЛА для планирования событий и позволяющие осуществлять моделирование параллельно протекающих процессов, являются операторами императивного управления. Специфика некоторых задач такова, что может быть сложным или неудобным соотносить события с определенным моментом системного времени. Необходимость задания системного времени может потребовать большого количества перекрестных ссылок между компонентами системы и проверки выполнения требуемых условий в заданные моменты времени. Средства интеррогативного управления обеспечивают "непрерывность" проверки выполнения требуемых условий, т.е. всякий раз, когда происходит изменение состояния системы. Расширение средств управления событиями операторами интеррогативного управления превратило бы такой язык, как СИМУЛА, удовлетворяющий всем вышеперечисленным требованиям к языкам моделирования, в более мощный аппарат для описания динамики функционирования сложных систем. При этом, в зависимости от конкретной задачи, можно обращаться к тем или иным средствам управления, повышая тем самым вычислительную эффективность системы в целом.

При решении различных задач может оказаться недостаточно тех понятий и операторов, которые имеются в данном языке. Необходимо, чтобы языки были способны к расширению, т.е. все новое могло бы

включаться в язык самими пользователями. В этом отношении интерес представляет подход, изложенный в [8]: "Потребность в специализированных языках лишь кажущимся образом противоречит желанию иметь универсальный язык программирования вместо множества различных языков, существующих в настоящее время. Задача состоит в том,чтобы разработать такой универсальный язык программирования, который служил бы базой для построения специализированных языков". В качестве такой базы авторами [8] предложен универсальный язык программирования СИМУЛА-67. Этот язык в качестве своего подмножества содержит, как и СИМУЛА, практически все средства АЛГОЛа-60. Центральным понятием языка СИМУЛА-67 является понятие класса. Декларация класса определяет структуру данных и правило действий всех объектов, принадлежащих этому классу. Классы могут использоваться в качестве префиксов (приставок) к другим классам, которые становятся в этом случае подклассами первых. При этом все понятия и операции, определенные в префиксе, становятся доступными для объектов класса с данным префиксом.

Основная идея возможности применения универсального языка СИМУЛА-67 в различных областях заключается в том, что специализация языка может быть осуществлена путем определения нового класса, содержащего требуемые для данной области понятия. Такой класс используется затем в качестве префикса к программе, составленной для данной области.

Классы, используемые в качестве префиксов, могут быть разработаны заранее и входить в язык в качестве системных классов. Одним из таких системных классов, имеющимся в языке СИМУЛА-67,является класс SIMULATION . В декларации этого класса:

```
class SIMULATION;
      begin  ... end ;
```

определены средства, аналогичные введенным в языке СИМУЛА,для решения задач моделирования. Пользователь, желающий написать программу моделирования, начинает ее словами:

```
SIMULATION begin  ...
```

Пользователь может также сам создавать новые классы соответственно содержанию конкретных задач. При этом может быть задана различная иерархия классов. Например, декларация классов:

```
      class A ...  ;
A class B ...  ;
B class C ...  ;
```

A class D ... ;
B class E ... ;

определяет следующую иерархию классов и структуру объектов, принадлежащих каждому классу (рис.2):

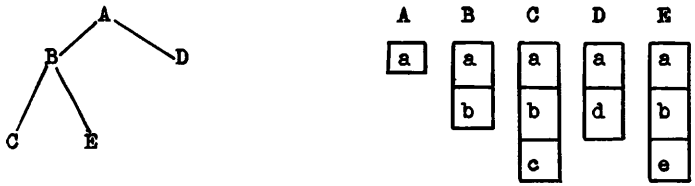

Рис. 2

Малые буквы на рис.2 означают атрибуты объекта, принадлежащего классу, обозначенному соответствующей заглавной буквой. Каждый из классов может иметь последовательность префиксов произвольной глубины. Введение классов позволяет не только описывать различные иерархические структуры данных, но и легко осуществлять специализацию языка. При этом необходимым условием является наличие в трансляторе средств включения в общую программу отдельно скомпилированных классов.

В заключение приведем таблицу, в которой отражен ряд отечественных разработок языков дискретного моделирования:

Таблица

Название языка	Прототип языка	ЭВМ	Город, организация
СЛЭНГ	SOL	М-220	Киев, ИК
АЛСИМ	СЛЭНГ	БЭСМ-6	Киев, ИК
НЕДИС	Расширенный средствами моделирования непрерывных систем язык СЛЭНГ	БЭСМ-6	Киев, ИК
СКИФ	СЛЭНГ	АСВТ	Москва, ИНЭУМ
СИМУЛА	SIMULA	БЭСМ-6	Москва, ИПМ
ДИС	CSL	БЭСМ-6	Новосибирск, ВЦ СО АН СССР
АРГОН	В ограниченном варианте SIMULA	БЭСМ-6	Новосибирск, НФ ИТМиВТ
МОДЕЛЬ-6	GPSS	БЭСМ-6	Новосибирск, НФ ИТМиВТ

Литература

1. Языки программирования. Под ред. Ф.Хенни, М., "Мир", 1972.

2. НЕЙЛОР Т. Машинные имитационные эксперименты с моделями экономических систем. М., "Мир", 1975.

3. Общецелевая система моделирования GPSS/360. Пер. с англ. Под ред. О.В.Голованова. М., НИИТЭХИМ, 1974.

4. МАРКОВИЦ Г., ХАУСПЕР Б., КАРР Г. СИМСКРИПТ . Алгоритмический язык для моделирования. М., "Сов.радио", 1966.

5. BUXTON J.N., LASKI J.G. Control and Simulation Language.- "Computer J.", 1962, v.5, N 3, p.194-199.

6. KNUTH P.E., McNELEY J.L. SOL - a symbolic language for general purpose systems simulation.- "IEEE Trans. on Elect.Comput.", 1964, v.EC-13, N 4, p.401-408.

7. ДАЛ У.-И., НИГАРД К. СИМУЛА - язык для программирования и описания систем с дискретными событиями. -В кн.: Алгоритмы и алгоритмические языки. Вып. 2. М., ВЦ АН СССР, 1967.

8. ДАЛ У.-И., МЮРХАУГ Б., НЮГОРД К. СИМУЛА-67. Универсальный язык программирования. М., "Мир", 1969.

Поступила в ред.-изд.отд.
11 августа 1978 года