3. Facey P., Gaines B. Real-time system design under an emulator embeded in a high-level language. Datafair 73 Conference Papers Vol.2, London

УДК 681.3.06 : 51

И.А.Будячевский, М.А.Кораблин, С.В.Смирнов

МОДЕЛИРОВАНИЕ ДИСКРЕТНЫХ СИСТЕМ В РАМКАХ ПЛ/I (ДОС ЕС)

I. Практическая реализация метода имитационного моделирования систем с помощью ЭВМ включает в себя ряд необходимых этапов [1,2]. Здесь рассматриваются вопросы, связанные с составлением программы для ЭВМ применительно к моделям дискретных систем.

Моделируемую систему будем называть дискретной, если происходящие в ней процессы можно представить в модели некоторой последовательностью дискретных изменений ("событий"). Подобным образом можно описать весьма широкий класс реально функционирующих сложных систем (АСУ производством, экспериментальными исследованиями и т.п.).

В отличие от широко применяемых универсальных языков программирования, трансляторы с которых включены в заводское математическое обеспечение современных ЭВМ, трансляторы со специализированных языков моделирования имеются лишь в отдельных организациях, и получение подобного математического обеспечения с соответствующей документацией весьма проблематично.

Описываемый ниже набор подпрограмм, формализующих и воспроизводящих динамические свойства моделируемой системы, наряду со стандартными возможностями дисковой операционной системы ЕС ЭВМ и имеющегося в ней подмножества языка ПЛ/I [3] позволит пользователю достаточно быстро и при оперировании единой системой понятий составлять сложные машинные имитационные программы. Следует отметить, что средства универсального языка программирования ПЛ/I оказываются весьма удобными для описания таких сложных объектов как системы управления экспериментом, включающие в себя ЭВМ.

II. Имитация динамики поведения моделируемой системы осуществляется в соответствии с имитационной концепцией языков событий

см., например, СИМСКРИПТ [4]. При этом имитационная программа составляется из идентифицированных подмоделей исходной системы - программ событий, определяющих совокупность операций по изменению состояния системы (или ее части), происходящего мгновенно в системном времени (так будем называть представление в модели реального времени). Последовательность событий, каждому из которых поставлен в соответствие некоторый момент системного времени, отражает функционирование моделируемой системы во времени.

Выбор данной схемы имитации был обусловлен (I) простой ее реализации в рамках ПЛ/I ДОС ЕС, (2) возможностью строгого контроля прохождения программы [I], что является особенно важным ввиду практического отсутствия в разработанной системе моделирования методов нахождения ошибок, которые могут быть допущены пользователем при описании динамических связей модели, (3) хорошей приспособленностью языков событий к исследованию задач массового обслуживания, представляющих для пользователя значительный интерес.

В дополнение к этому имеется возможность выполнять программу события всякий раз, когда соблюдено определенное наперед заданное условие (схема "работ"). Тем самым определено средство описания систем с большим числом разнообразных ресурсов, позволяющее записывать условия их доступности в компактной форме [I], с.4I8 .

Представление реального времени в модели и поддержание порядка выполнения отдельных программ событий возложено на подпрограмму *WSTART*. Каждый раз при передаче управления подпрограмме *WSTART* она вызывает программу некоторого события, руководствуясь либо (I) составленным ранее расписанием - упорядоченным списком событий, каждый элемент которого содержит в частности номер события и запланированное время его появления, - либо (2) результатами проверки соблюдения некоторого условия. В первом случае из списка исключается событие с наименьшим временем, системное время устанавливается равным времени появления этого события и вызывается соответствующая этому событию программа.

Второй случай соответствует так называемому интеррогативному управлению [5] .

Если во время отработки текущего события возникают новые события, то они /точнее номер соответствующей программы/ заносятся в список вместе с планируемым временем их появления в будущем. Такое планирование событий осуществляется с помощью вызова подпрограмм

императивного управления [5], перечень которых с необходимыми комментариями приводится ниже.

Прежде отметим, что текущее значение системного времени при соблюдении определенных условий доступно во всех программах под именем *WSTIME*.

WINITC(TS) - инициализирует подпрограммы императивного управления и *WSTART*, подготавливая возможность имитации поведения системы на отрезке $[\Phi, TS]$.

CREAS(E,S) - "вызвать событие E в момент времени S". Здесь и далее параметр E /как и E1, E2 / указывает номер планируемого события и может принимать значения 1,2,...,50, за исключением тех из них, которые не использованы для нумерации событий в данной имитационной программе. Параметр S определяет момент появления события в системном времени. События, вносимые в список с одинаковым временем, упорядочиваются по принципу "первый пришел - первый вышел". Этот порядок можно изменить, воспользовавшись подпрограммой *CREASP(E,S)*, которая отличается от рассматриваемой лишь тем, что событие E размещается в списке перед событиями с тем же временем S, запланированными с помощью *CREAS(E,S)*, но тем не менее, после событий с временем S, внесенных в список подпрограммой *CREASP(E,S)* ранее.

CREAT(E,T) и *CREATP(E,T)* работают аквивалентно соответственно *CREAS(E,WSTIME+T)* и *CREASP(E,WSTIME+T)*.

Замечание 1. Если $S>TS$ или $WSTIME+T>TS$, то вызов рассмотренных подпрограмм соответствует выполнению пустого оператора. Попытка запланировать событие в "прошлом" приводит к внесению его в список с временем появления равным *WSTIME*.

Вместо задания системного времени для определения места события E1 в списке планируемых событий можно указать его положение относительно другого события E2 в этом списке. Событию E1 будет приписано то же системное время, что и событию E2. Эта операция осуществляется подпрограммами *CREABF (E1, E2)* и *CREAAF (E1, E2)*, с помощью которых планируется вызов события E1 соответственно "перед" и "вслед за" событием E2.

Вызов подпрограммы *CANCEL (E2)* приводит к исключению события E2 из списка.

Замечание 2. Параметр E2 трех последних подпрограмм опреде-

лает номер события, имеющего наименьшее время из всех событий с тем же номером, хранящихся в списке. Если в списке не окажется событий с номером E2, то вызов этих подпрограмм соответствует выполнению пустого оператора.

Замечание 3. Фактические параметры рассмотренных подпрограмм задаются по правилам, принятым в ПЛ/I ДОС ЕС. При этом E /равно E1 и E2/ и TS /равно S и T / должны иметь атрибуты соответственно DECIMAL FIXED (5,0) и DECIMAL FLOAT (6).

Интеррогативное управление организуется с помощью пронумерованных булевских программ - функций, реализующих проверку некоторых условий. Конкретное содержание этих программ определяется пользователем. Их число в имитационной программе не должно превышать двадцати и для их идентификации допустимы имена CN1, CN2,..., CN20.

Интеррогативное управление может вводиться и отменяться пользователем в ходе имитации для любых используемых в имитационной программе программ событий. После вызова подпрограммы ITRR(E,#C) программа события с номером E будет вызываться всякий раз, когда значение программы - функции CN#C ,где #C - номер этой программы, равно 'I'B /фактический параметр #C задается аналогично параметру E - см. Замечание 3/. Вызов подпрограммы ABITRR(#C) приводит к отмене интеррогативного управления, осуществляемого с помощью функции CN#C /вне зависимости от того, каков номер управляемого события/.

Подпрограмма WSTART каждый раз после очередного пересчета системного времени последовательно /от меньших номеров к большим/ проверяет значения программ - функций, участвующих на текущий момент в организации интеррогативного управления /"активных" программ/, и вызывает "разрешенные" события. После вычисления последней "активной" программы-функции цикл проверки повторяется вновь и т.д., пока хотя бы одна "активная" программа равна 'I'B. Системное время остается фиксированным.

Замечание 4. Каждая булевская подпрограмма-функция может управлять только одним событием. Таким образом, одновременно интеррогативно управляемы не более двадцати событий. Последовательное в ходе имитации выполнение операторов CALL ITRR(E1,#C);... CALL ITRR(E2,#C); автоматически приводит к отмене интеррогативного управления функций CN#C для события E1 и, разумеется, к введению такого управления для события E2.

Ш. Содержание программы имитации, т.е. логическая последовательность действий, которые должны быть выполнены вычислительной машиной целиком определяется пользователем. Методология имитационных экспериментов в настоящее время хорошо разработана /см.например, [I,2,6] /. В [I,2] подробно разобраны примеры описания систем в терминах дискретных событий. Остановимся лишь на организации в ДОС ЕС задания на выполнение имитационной программы при использовании описанных подпрограмм.

Программы событий записываются на языке ПЛ/I в виде процедур без параметров и могут иметь имена EV1, EV2, ..., EV5Ф . В общем случае каждая из этих программ должна транслироваться отдельно, поэтому данные, оперирование с которыми возможно при выполнении нескольких программ событий, должны быть описаны в них с атрибутом EXTERNAL.

Замечание 5. Запрещается использование знака @ в качестве начальной буквы идентификаторов данных с атрибутом EXTERNAL Включение в любую программу оператора DECLARE @STIME EXTERNAL делает доступным в ней значение системного времени.

В общем случае отдельно транслируются программы-функции, необходимые для организации интеррогативного управления.

Основной исходный модуль на ПЛ/I /главная процедура/ должен содержать оператор CALL @INITC (TS) , где TS определено. После выполнения этого оператора возможен доступ к подпрограммам управляющей программы. Для начала процесса имитации необходимо выполнить оператор CALL @START; .

Задание на выполнение имитационной программы на ПЛ/I обязательно включает в трансляцию всех перечисленных исходных модулей и составляется по обычным правилам ДОС ЕС.

Л И Т Е Р А Т У Р А

I. Нейлор Т. Машинные имитационные эксперименты с моделями экономических систем. М., "Мир", I975, с.502.

2. Вагнер Г. Основы исследования операций. Т.3, М., "Мир", I973, с.502.

3. Универсальный язык программирования PL /I. М., "Мир", I968, с.352.

4. Марковиц Г., Хаусснер Б., Карр Г. СИМСКРИПТ. Алгоритмический язык программирования, М., "Сов.радио", I966, с.I52.

5. Дал У.-И. Языки для моделирования систем с дискретными событиями. - В кн. :Языки программирования. М., "Мир", 1972, 344-403 с.

6. Бусленко Н.П. Моделирование сложных систем. М., "Наука", 1968, 356 с.

УДК 681.3

В.В.Пшеничников

РАЗРАБОТКА ВХОДНОГО ЯЗЫКА ПРИ МОДЕЛИРОВАНИИ СИСТЕМЫ СБОРА И ОБРАБОТКИ ИНФОРМАЦИИ

Математическое обеспечение системы сбора и обработки информации включает в себя достаточно большое количество процедур. Пользователь должен в зависимости от алгоритма функционирования системы организовать выполнение этих процедур в заданной последовательности. Кроме того, на него возлагается обязанность обеспечить увязку параметров различных процедур для их правильной работы. Это значительно усложняет работу экспериментатора. Та же проблема возникает и при моделировании сложных автоматизированных систем управления.

Обычно при разработке систем узкого назначения считалось нецелесообразным создание специализированного языка, так как временные затраты на разработку компилятора с него значительны. В большей степени это объясняется тем, что разработку компилятора приходилось вести на уровне машинных команд или, в лучшем случае, на каком-либо автокоде (язык Ассемблера).

В последнее время получает широкое распространение язык ПЛ/I. Обладая значительной универсальностью, ПЛ/I позволяет писать программы как для обработки данных (в нашем случае это набор процедур математического обеспечения системы), так и собственно интерпретатор входного языка, преобразующий программу на входном языке в заданную последовательность обращений к процедурам.

Для опытного программиста создание такого интерпретатора не составит особого труда, так как в ПЛ/I есть хорошие средства для работы с символьной информацией.

В данной работе рассматривается входной язык для работы с моделью автоматизированной системы динамических испытаний дат-