**Тема 5. РАЗРАБОТКА ИМИТАЦИОННОЙ МОДЕЛИ**

**5.1. Управление модельным временем**

Приступая к изучению механизмов управления модельным временем, уместно поговорить о роли времени в имитационном моделировании. Ранее было отмечено, что имитационное моделирование представляет собой наблюдение за поведением системы в течение некоторого промежутка времени. Конечно, далеко не во всех статистических испытаниях фактор времени играет ведущую роль, а в некоторых и вообще может не рассматриваться. Но значительно больше таких за­дач, в которых оценка эффективности моделируемой системы напрямую связана с временными характеристиками ее функционирования. К ним относятся задачи по оценке производительности, некоторые задачи по оценке на­дежности, качества распределения ресурсов, а также все задачи, связанные с иссле­дованием эффективности процессов обслуживания. Характерной особенностью большинства практических задач является то, что скорость протекания рассматри­ваемых в них процессов значительно ниже скорости реализации модельного экспе­римента. Например, если моделируется работа авторемонтной мастерской в тече­ние недели, вряд ли кому-то придет в голову воспроизводить этот процесс в модели в таком же масштабе времени. А в ряде задач требуется именно реализация реального масштаба времени.

При разработке практически любой имитационной модели и пла­нировании проведения модельных экспериментов необходимо соотносить между собой три представления времени:

• реальное время, в котором происходит функционирование имитируемой системы;

• модельное (или, как его еще называют, системное) время, в масштабе которо­го организуется работа модели;

• машинное время, отражающее затраты времени ЭВМ на проведение ими­тации.

С помощью механизма модельного времени решаются следующие задачи:

1) отображается переход моделируемой системы из одного состояния в другое;

2) производится синхронизация работы компонент модели;

3) изменяется масштаб времени «жизни» (функционирования) исследуемой системы;

4) производится управление ходом модельного эксперимента.

5) моделируется квазипараллельная реализация событий в модели;

Приставка «квази» в данном случае отражает последовательный характер об­работки событий (процессов) в ИМ, которые в реальной системе возникают (про­текают) одновременно.

Необходимость решения последней задачи связана с тем, что в распоряжении исследователя находится, как правило, однопроцессорная вычислительная систе­ма, а модель может содержать значительно большее число одновременно работаю­щих подсистем. Поэтому действительно параллельная (одновременная) реализа­ция всех компонент модели невозможна. Даже если используется так называемая распределенная модель, реализуемая на нескольких узлах вычислительной сети, совсем необязательно число узлов будет совпадать с числом одновременно рабо­тающих компонент модели. Следует отметить, что реали­зация квазипараллельной работы компонент модели является достаточно сложной технической задачей. Некоторые возможные методы ее решения рассматриваются в следующем разделе.

Ранее были названы два метода реализации механизма модельного времени — с постоянным шагом и по особым состояниям.

Выбор метода реализации механизма модельного времени зависит от назначе­ния модели, ее сложности, характера исследуемых процессов, требуемой точности результатов и т. д.

При использовании метода ***постоянного шага*** отсчет системного времени ве­дется через фиксированные, выбранные исследователем интервалы времени. Со­бытия в модели считаются наступившими в момент окончания этого интервала. Погрешность в измерении временных характеристик системы в этом случае зави­сит от величины шага моделирования Δt.

Метод постоянного шага предпочтительнее, если:

• события появляются регулярно, их распределение во времени достаточно рав­номерно;

• число событий велико и моменты их появления близки;

• невозможно заранее определить моменты появления событий.

Данный метод управления модельным временем достаточно просто реализовать в том случае, когда условия появления событий всех типов в модели можно пред­ставить как функцию времени.

Пусть, например, событие состоит в том, что летящий самолет пересекает неко­торый воздушный рубеж, расстояние до которого равно *R.* Если самолет движется по прямой с постоянной скоростью *V,* то можно вычислять путь, пройденный само­летом, с интервалом времени Δ*t: S*=S+V·Δt. Соответственно событие считается наступившим, если выполняется условие S > *R,* а момент времени наступления со­бытия принимается равным *п • Δt,* где *п* — номер шага моделирования, на котором условие стало истинным.

Выбор величины шага моделирования является нелегким и очень важным делом. Универсальной методики решения этой проблемы не су­ществует, но во многих случаях можно использовать один из следующих под­ходов:

• принимать величину шага равной средней интенсивности возникновения со­бытий различных типов;

• выбирать величину Δt равной среднему интервалу между наиболее частыми (или наиболее важными) событиями.



• принимать величину шага равной средней интенсивности возникновения со­бытий различных типов;

• выбирать величину Δt равной среднему интервалу между наиболее частыми (или наиболее важными) событиями.

При моделировании ***по особым состояниям*** системное время каждый раз из­меняется на величину, строго соответствующую интервалу времени до момента наступления очередного события. В этом случае события обрабатываются в поряд­ке их наступления, а одновременно наступившими считаются только те, которые являются одновременными в действительности.

Метод моделирования по особым состояниям сложнее в реализации, так как для него требуется разработка специальной процедуры планирования событий (так называемого календаря событий).

Моделирование по особым состояниям целесообразно использовать, если:

• события распределяются во времени неравномерно или интервалы между ними велики;

• предъявляются повышенные требования к точности определения взаимного положения событий во времени;

• необходимо реализовать квазипараллельную обработку одновременных событий.

Дополнительное достоинство метода заключается в том, что он позволяет эко­номить машинное время, особенно при моделировании систем периодического дей­ствия, в которых события длительное время могут не наступать.

Таким образом:

* Выбор механизма изменения модельного времени определяет и технологию реализации имитационной модели.
* На выбор метода моделирования влияет целый ряд факторов, однако определяющим является тип моделирующей системы: для дискретных систем, события в которых распределены во времени неравномерно, более удобным является изменение модельного времени по особым состояниям.

Если в модели должны быть представлены компоненты реальной системы, работа которых измеряется в разных единицах времени, то они должны быть предварительно приведены к единому масштабу.

**5.2. Моделирование параллельных процессов**

Практически любая более или менее сложная система имеет в своем составе компоненты, работающие одновременно, или, как принято говорить на языке тех­ники, параллельно. Параллельно работающие подсистемы могут вза­имодействовать самым различным образом, либо вообще работать независимо друг от друга. Способ взаимодействия подсистем определяет вид параллельных процес­сов, протекающих в системе. В свою очередь, вид моделируемых процессов влияет на выбор метода их имитации.

**5.2.1.Виды параллельных процессов в сложных системах**

***Асинхронный параллельный процесс*** *—* такой процесс, состояние которого не зависит от состояния другого параллельного процесса (ПП).

Пример асинхронных ПП из области вычислительной техники: выполнение вычислений процессором и вы­вод информации на печать.

***Синхронный ПП*** *—* такой процесс, состояние которого зависит от состояния взаимодействующих с ним ПП.

Пример синхронного ПП: работа торговой организации и доставка товара со склада (нет товара — нет торговли).

Один и тот же процесс может быть синхронным по отношению к одному из ак­тивных ПП и асинхронным по отношению к другому. Так, при работе вычисли­тельной сети по технологии «клиент-сервер» каждый из узлов сети синхронизиру­ет свою работу с работой сервера, но не зависит от работы других узлов.

***Подчиненный ПП*** *—* создается и управляется другим процессом (более высоко­го уровня). Весьма характерным примером таких процессов является ведение боевых дей­ствий подчиненными подразделениями.

***Независимый ПП—*** не является подчиненным ни для одного из процессов. Скажем, после запуска неуправляемой зенитной ракеты ее полет можно рас­сматривать как независимый процесс, одновременно с которым самолет ведет бое­вые действия другими средствами.

Способ организации параллельных процессов в системе зависит от физической сущности этой системы.

Остановимся несколько подробнее на особенностях реализации параллельных процессов в вычислительных системах (ВС). Это обусловлено следующей причи­ной.

Разработка и использование любой ИМ предполагает ее программную реализа­цию и исследование с применением ВС. Поэтому для реализации моделей, имити­рующих параллельные процессы, в некоторых случаях применимы механизмы, используемые при выполнении параллельных вычислений.

Вместе с тем, реализация параллельных процессов в ВС имеет свои особенности:

• на уровне задач вычислительные процессы могут быть истинно параллельны­ми только в многопроцессорных ВС или вычислительных сетях;

• многие ПП используют одни и те же ресурсы, поэтому даже асинхронные ПП в пределах одной ВС вынуждены согласовывать свои действия при обращении к общим ресурсам;

• в ВС дополнительно используется еще два вида ПП: родительский и дочер­ний ПП; особенность их состоит в том, что процесс-родитель не может быть завер­шен, пока не завершатся все его дочерние процессы.

В силу перечисленных особенностей для организации взаимодействия парал­лельных процессов в ВС используются три основных подхода:

• на основе «взаимного исключения»;

• на основе синхронизации посредством сигналов;

• на основе обмена информацией (сообщениями).

***«Взаимное исключение»*** предполагает запрет доступа к общим ресурсам (об­щим данным) для всех ПП, кроме одного, на время его работы с этими ресурсами (данными).

***Синхронизация*** подразумевает обмен сигналами между двумя или более про­цессами по установленному протоколу. Такой «сигнал» рассматривается как некоторое событие, вызывающее у получившего его процесса соответствующие действия.

Часто возникает необходимость передавать от одного ПП другому более под­робную информацию, чем просто «сигнал-событие». В этом случае процессы со­гласуют свою работу на основе обмена сообщениями.

Перечисленные механизмы реализуются в ВС на двух уровнях — системном и прикладном.

Механизм взаимодействия между ПП на системном уровне определяется еще на этапе разработки ВС и реализуется в основном средствами операционной систе­мы (частично — с использованием аппаратных средств).

На прикладном уровне взаимодействие между ПП реализуется программистом средствами языка, на котором разрабатывается программное обеспечение.

Наибольшими возможностями в этом отношении обладают так называемые язы­ки реального времени (ЯРВ) и языки моделирования.

Языки реального времени — это языки, предназначенные для создания про­граммного обеспечения, работающего в реальном масштабе времени, например для разработки различных автоматизированных систем управления (предприя­тием, воздушным движением и т. д.). К ним, в частности, относятся: язык *Ада,* язык *Модула* и практически единственный отечественный язык реального време­ни — *Эль-76* (использовавшийся в многопроцессорных вычислительных комп­лексах семейства «Эльбрус»).

**5.2.2. Методы описания параллельных процессов в системах и языках моделирования**

Языки моделирования по сравнению с языками реального времени требуют от разработчика значительно менее высокого уровня подготовки в области програм­мирования, что обусловлено двумя обстоятельствами:

• во-первых, средства моделирования изначально ориентированы на квазипа­раллельную обработку параллельных процессов;

• во-вторых, механизмы реализации ПП относятся, как правило, к внутренней орга­низации системы (языка) моделирования и их работа скрыта от программиста.

В практике имитационного моделирования одинаково широко используются как процессно-ориентированные языки (системы) моделирования, например *SIMULA,* так и языки, ориентированные на обработку транзактов (например, язык *GPSS).* В тех и других используются аналогичные методы реализации квазипарал­лелизма, основанные на ведении списков событий. В процессно-ориентированных системах используются списки событий следования, а в транзактных системах — списки событий изменения состояний.

Современные языки и системы моделирования, ориентированные на использо­вание в среде многозадачных операционных систем типа Windows, частично используют их механизмы управления процессами, что делает их применение еще более эффективным. В пакете MATLAB также имеется собственный язык модели­рования, и к нему в полной мере можно отнести сказанное выше. Тем не менее во многих случаях оказывается полезным знание общего механизма реализации ПП в языках моделирования.

Рассмотрим его применительно к моделированию на основе тракзактов.

В этом случае под событием понимается любое перемещение транзакта по сис­теме, а также изменение его состояния (обслуживается, заблокирован и т. д.).

Событие, связанное с данным транзактом, может храниться в одном из следую­щих списков.

***Список текущих событий.*** В этом списке находятся события, время наступле­ния которых меньше или равно текущему модельному времени. События с «мень­шим» временем связаны с перемещением тех транзактов, которые должны были начать двигаться, но были заблокированы.

***Список будущих событий.*** Этот список содержит события, время наступле­ния которых больше текущего модельного времени, то есть события, которые дол­жны произойти в будущем (условия наступления которых уже определены — на­пример, известно, что транзакт будет обслуживаться некоторым устройством 10 единиц времени).

***Список прерываний.*** Данный список содержит события, связанные с возобнов­лением обработки прерванных транзактов. События из этого списка выбираются в том случае, если сняты условия прерывания.

В списке текущих событий транзакты расположены в порядке убывания при­оритета соответствующих событий; при равных приоритетах — в порядке поступ­ления в список.

Каждое событие (транзакт) в списке текущих событий может находиться либо в активном состоянии, либо в состоянии задержки. Если событие активно, то соот­ветствующий транзакт может быть продвинут по системе; если продвижение не­возможно (например, из-за занятости устройства), то событие (и транзакт) пере­водится в состояние задержки.

Как только завершается обработка (продвижение) очередного активного транзакта, просматривается список задержанных транзактов, и ряд из них пе­реводится в активное состояние. Процедура повторяется до тех пор, пока в спис­ке текущих событий не будут обработаны все активные события. После этого просматривается список будущих событий. Модельному времени присваивает­ся значение, равное времени наступления ближайшего из этих событий. Данное событие заносится в список текущих событий. Затем просматриваются осталь­ные события списка. Те из них, время которых равно текущему модельному вре­мени, также переписываются в список текущих событий. Просмотр заканчива­ется, когда в списке остаются события, времена которых больше текущего модельного времени.

В качестве иллюстрации к изложенному рассмотрим небольшой пример.

*►Пусть в систему поступают транзакты трех типов, каждый из которых обслу­живается отдельным устройством. Известны законы поступления транзактов в систему и длительность их обслуживания. Таким образом, в системе существуют три параллельных независимых процесса (PI, P2, РЗ).*

*Временная диаграмма работы системы при обслуживании одного транзакта каж­дого типа показана на рис.2.7.*

*На рисунке события, относящиеся к процессу Р1, обозначены как С1i, относя­щиеся к Р2 и к РЗ — соответственно как С2i. и СЗi. Моменты времени tвх и tвых соот­ветствуют началу и окончанию обслуживания транзакта.*



*Для каждого процесса строится своя цепь событий, однако списки событий являются общими для всей модели. Формирование списков начинается с за­полнения списка будущих событий. Как было отмечено выше, в этот список помещаются события, время наступления которых превышает текущее значе­ние модельного времени. Очевидно, что на момент заполнения списка время наступления прогнозируемых событий должно быть известно. На первом шаге tм=0, и в список будущих событий заносятся события С11, С21, С31. Затем событие с наименьшим временем наступления — С11 — переносится в список те­кущих событий; если одновременных с ним событий нет, то оно обрабатывается и исключается из списка текущих событий. После этого вновь корректирует­ся список будущих событий и т.д., пока не истечет заданный интервал моде­лирования.*

*Динамика изменения списков текущих и будущих событий для рассмотренного примера отражена в приведенной ниже таблице.*

|  |  |  |
| --- | --- | --- |
| ***t*** | ***Список текущих событий*** | ***Список будущих событий*** |
| ***0*** | *0* | *C11, C21,C31* |
| ***t11*** | *C11* | *C21, C31, C12* |
| ***t21*** | *C21* | *C31, C12, C22* |
| ***t31*** | *C31* | *C12, C22, C32* |
| ***t12*** | *C12* | *C22, C32, C13* |
| ***t22*** | *C22* | *C32, C13, C23* |
| ***t32*** | *C32* | *C13, C23, C33* |
| ***t13*** | *C13* | *C23, C33* |
| ***t23*** | *C23, C33* |  |

*◄*

Многие авторы книг по имитационному моделированию считают, что знание механизма ведения списков событий просто необходимо разра­ботчику модели; умение проследить в динамике цепь происходящих в модели собы­тий, во-первых, повышает уверенность создателя модели в том, что она работает пра­вильно и, во вторых, существенно облегчает процесс отладки и модификации модели.

**5.2.3. Применение сетевых моделей для описания параллельных процессов**

Этапу программной реализации модели (т. е. ее опи­санию на одном из языков программирования) должен предшествовать так на­зываемый этап алгоритмизации. Другими словами, прежде чем превратить имитационную модель в работающую программу, ее создатель должен воспользоваться каким-то менее формальным и более наглядным сред­ством описания логики работы будущей программы. Это требова­ние не является обязательным, т.к. при наличии достаточного опыта программа не очень сложной модели может быть написана сразу. Однако при моделировании более сложных систем даже опытные разработчики бывают вынуждены немного «притормозить» на этапе алгоритмизации. Для описания логики работы модели могут быть использова­ны различные средства: либо русский язык (устный или письменный), либо тра­диционные схемы алгоритмов, либо какие-то другие «подручные» средства. Первые два варианта являются, как правило, наи­более знакомыми и наиболее часто используемыми. Однако такие схемы совершенно не приспо­соблены для описания параллельных процессов.

Одним из наиболее элегантных и весьма распространенных средств описания параллель­ных процессов — описание ***сетями Петри.*** Рассмотрим те основные сведения, кото­рые необходимы с точки зрения реализации технологии имитационного моделирования параллельных процессов.

Одно из основных достоинств аппарата сетей Петри заключается в том, что они могут быть представлены как в графической форме (что обеспечивает наглядность), так и в аналитической (что позволяет автоматизировать процесс их анализа).

При графической интерпретации сеть Петри представляет собой граф осо­бого вида, состоящий из вершин двух типов — *позиций* и *переходов,* соединен­ных ориентированными дугами, причем каждая дуга может связывать лишь раз­нотипные вершины (позицию с переходом или переход с позицией). Вершины-позиции обозначаются кружками, вершины-переходы — черточками. С содержательной точки зрения, переходы соответствуют событиям, присущим исследуемой системе, а позиции — условиям их возникновения. Таким образом, совокупность переходов, позиций и дуг позволяет описать причинно-следствен­ные связи, присущие системе, но в статике. Чтобы сеть Петри «ожила», вводят еще один вид объектов сети — так называемые *фишки,* или метки позиций. Пе­реход считается активным (событие может произойти), если в каждой его вход­ной позиции есть хотя бы одна фишка. Расположение фишек в позициях сети называется ***разметкой сети*** (пример перемещения фишек по сети приведен на рис.5.6).



В аналитической форме сеть Петри может быть представлена следующим образом:

*P=(B,D,I,0,M),*

где ***В =***{bi} — конечное непустое множество позиций;

***D***= ***{di}*** — конечное непустое множество переходов;

***I : B*х*D ->*** *0,1* — входная функция (прямая функция инцидентности), которая для каждого перехода задает множество его входных позиций;

***О : D*x*B* ->** 0,1 — выходная функция (обратная функция инцидентности), кото­рая для каждого перехода задает множество его входных позиций;

***М*** *—* функция разметки сети, ***М : В ->***0, 1, 2,... — ставит каждой позиции сети в соответствие неотрицательное целое число.

С учетом введенных обозначений необходимое условие срабатывания перехода *dj* может быть записано следующим образом:

***∀bi ∈ I(di) {M(bi)≥1}***

(*для всех входных позиций разметка должна быть >1*).

Срабатывание перехода *dj* изменяет разметку сети *М(В)* на разметку *М’(В)* по следующему правилу:

***M’(B)=M(B)-I(dj)+O(dj),***

то есть переход *dj* изымает по одной метке из каждой своей входной позиции и добав­ляет по одной метке в каждую из выходных позиций. Смену разметки обозначают так:

***dj***

***Mo⎪⎯M’***

Входная и выходная функции сети Петри (***I*** и ***О***) позволяют описать любую сеть с помощью двух матриц размера *т х п* (матриц входных и выходных позиций), имеющих следующую структуру:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | d1 | d2 | ... | dj | ... | dn |
| b1 | 0 | 1 | ... | 0 | ... | 0 |
| b2 | 1 | 1 | ... | 0 | ... | 1 |
| ... | ... | ... | ... | ... | ... | ... |
| bj | 0 | 1 | ... | 0 | ... | 1 |
| ... | ... | ... | ... | ... | ... | ... |
| bm | 1 | 0 | ... | 1 | ... | 0 |

Основные направления анализа сети Петри следующие:

1. *Проблема достижимости:* в сети Петри с начальной разметкой М0 требуется определить, достижима ли принципиально некоторая разметка *М'* из M0.

С точки зрения исследования моделируемой системы, эта проблема интер­претируется как проблема достижимости (реализуемости) некоторого состоя­ния системы.

2. *Свойство живости.* Под живостью перехода понимают возможность его сра­батывания в данной сети при начальной разметке М0. Анализ модели на свойство живости позволяет выявить невозможные состо­яния в моделируемой системе (например, неисполняемые ветви в программе).

3. *Безопасность сети.* Безопасной является такая сеть Петри, в которой ни при каких условиях не может появиться более одной метки в каждой из позиций. Для исследуемой системы это означает возможность функционирования ее в стационарном режиме. На основе анализа данного свойства могут быть определе­ны требования к буферным накопителям в системе.

Итак, достоинства сетей Петри заключаются в том, что они:

1) позволяют моделировать ПП всех возможных типов с учетом вероятных кон­фликтов между ними;

2) обладают наглядностью и обеспечивают возможность автоматизирован­ного анализа;

3) позволяют переходить от одного уровня детализации описания системы к другому (за счет раскрытия/закрытия переходов).

Вместе с тем, сети Петри имеют ряд недостатков, ограничивающих их возмож­ности. Основной из них — время срабатывания перехода считается равным 0, что не позволяет исследовать с помощью сетей Петри временные характеристики мо­делируемых систем.

В результате развития аппарата сетей Петри был разработан ряд расширений. Одно из наиболее мощных — так называемые .Е-сети (evaluation — «вычисления», «оценка») — «оценочные сети».

В отличие от сетей Петри, в E-сетях:

1) имеются несколько типов вершин-позиций: простые позиции, позиции-оче­реди, разрешающие позиции;

2) фишки (метки) могут снабжаться набором признаков (атрибутов);

3) с каждым переходом может быть связана ненулевая задержка и функция пре­образования атрибутов фишек;

4) введены дополнительные виды вершин-переходов.

5) в любую позицию может входить не более одной дуги и выходить также не более одной.

В связи с этим любой переход может быть описан тройкой параметров:

***dj=(S,t(dj),ρ(dj)),***

где S — тип перехода,

*t(dj), —* функция задержки,

*ρ(dj)* — функция преобразования атрибутов.

Особенности E-сетей существенно расширяют их возможности для моделирования дискретных систем вообще и параллельных процессов в част­ности. Технология моделирования систем в виде Е-сетей может быть реализована с помощью инструмента SIMULINK, вхо­дящего в состав па