Занятие № 11
Имитационное моделирование

1.1 Процедура имитационного моделирования.

Определение метода имитационного моделирования. Метод ИМ заключается в создании логико-аналитической (математической модели системы и внешних воздействий), имитации функционирования системы, т.е. в определении временных изменений состояния системы под влиянием внешних воздействий и в поучении выборок значений выходных параметров, по которым определяются их основные вероятностные характеристики. Данное определение справедливо для стохастических систем.

При исследовании детерминированных систем отпадает необходимость изучения выборок значений выходных параметров.

Модель системы со структурным принципом управления представляет собой совокупность моделей элементов и их функциональные взаимосвязи. Модель элемента (агрегата, обслуживающего прибора) - это, в первую очередь, набор правил (алгоритмов) поведения устройства по отношению к выходным воздействиям (заявкам) и правил изменений состояний элемента. Элемент отображает функциональное устройство на том или ином уровне детализации. В простейшем случае устройство может находится в работоспособном состоянии или в состоянии отказа. В работоспособном состоянии устройство может быть занято, например, выполнение операции по обслуживанию заявки или быть свободным. К правилам поведения устройства относятся правила выборки заявок из очереди; реакция устройства на поступление заявки, когда устройство занято или к нему имеется очередь заявок; реакция устройства на возникновение отказа в процессе обслуживания заявки и некоторые другие.

Имитационное моделирование (ИМ) — это метод исследования, который основан на том, что анализируемая динамическая система заменяется имитатором и с ним производятся эксперименты для получения об изучаемой системе. Роль имитатора зачастую выполняет программа ЭВМ.

Основная идея метода ИМ состоит в следующем. Пусть необходимо определить функцию распределения случайной величины **y**. Допустим, что искомая величина **y** может быть представлена в виде зависимости: $\mathbf{y=f(\alpha,\beta,....,\omega)}$ где $\alpha,\beta,....,\omega$ случайные величины с известными функциями распределения.

Для решения задач такого вида применяется следующий алгоритм:

1) по каждой из величин $\alpha,\beta,....,\omega$ производится случайное испытание, в результате каждого определяется некоторое конкретное значение случайной величины $\alpha_i,\beta_i,....,\omega_i$;
2) используя найденные величины, определяется одно частное значение $\mathbf{y_i}$ по выше приведённой зависимости;
3) предыдущие операции повторяются N раз, в результате чего определяется N значений случайной величины **y**;

4) на основании N значений величины находится её эмпирическая функция распределения.

1.2 Имитация функционирования системы.

Предположим, исследуется вычислительная система (ВС), состоящая из процессора 1 с основной памятью, устройство вода перфокарт 4, АЦПУ 2 и дисплея 3 (рис. 4.1.).

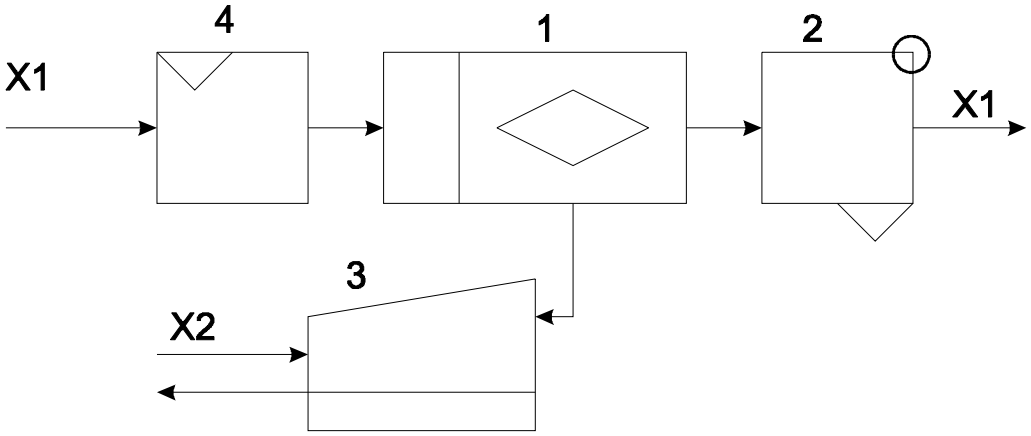

Рис. 4.1. Упрощённая схема моделируемой системы.

Через устройство 4 поступает поток заданий X1. Процессор обрабатывает задания и результаты выдаёт на АЦПУ 2. Одновременно с этим ВС используется, например, как информационно-справочная система. Оператор-пользователь, работающий за дисплеем, посылает в систему запросы X2, которые обрабатываются процессором и ответы выводятся на экран дисплея. Процессор работает в 2-х программном режиме: в одном разделе обрабатываются задания X1, в другом, с более высоким относительным приоритетом запросы X2. Представим данную ВС в упрощённом варианте в виде стохастической сети из 4-х СМО. Потоки заданий и запросы будем называть потоками заявок. Считаем потоки X1 и X2 независимыми. Известны ф.р. периодов следования заявок τ_1 и τ_2 и длительность обслуживания T_{1K} , T_{2K} заявок в к-ом устройстве. Требуется определить времена загрузки каждого устройства и времена реакции по каждому из потоков.

Вначале определяется момент поступления в систему 1-ой заявки потока X1 по результатам случайного испытания в соответствии с ф.р. периода следования заявок.

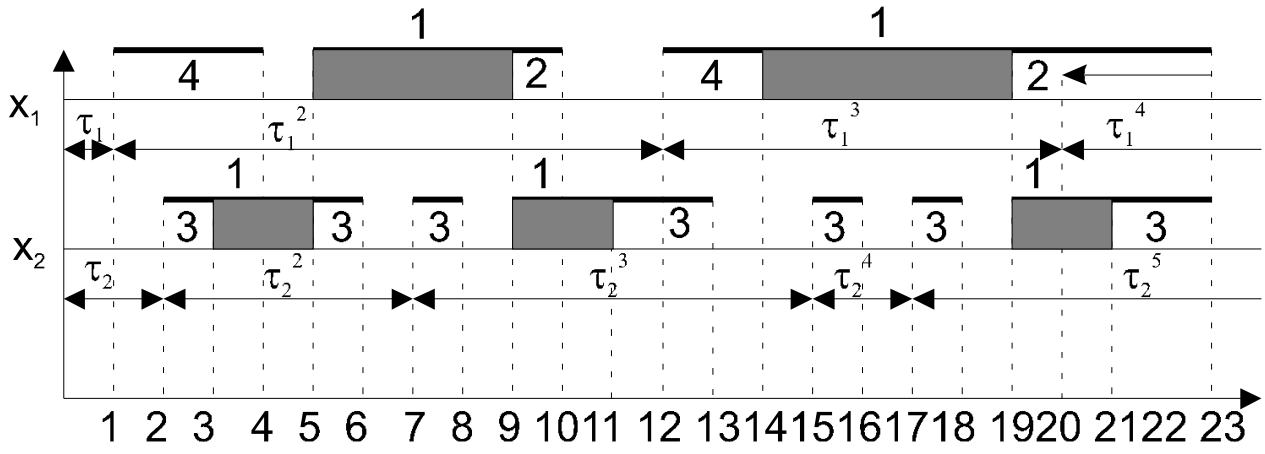

Рис. 4.2. Временная диаграмма функционирования ВС.

На рис. 2 это момент времени $t_1=0+\tau_1^1$ (здесь и далее верхний индекс обозначает порядковый номер заявки данного потока). То же самое делается для потока X2. На рис.2 момент поступления 1-ой заявки потока X2 $t_2=0+\tau_2^1$. Затем находится минимальное время, т.е. наиболее раннее событие. В примере это время t_1. Для 1-ой заявки потока X1определяется время обслуживания устройством ввода перфокарт T^1_{14} методом случайного испытания и отмечается момент окончания обслуживания $t_4=t_1+ T^1_{14}$. На рис. показан переход устройства 4 в состояние "занято". Одновременно определяется момент поступления следующей заявки потока X1: $t_{12}=t_1+\tau_1^2$. Следующее минимальное время это момент поступления заявки потока X2 - t_2. Для этой заявки находится время обслуживания на дисплее T^1_{23} и отслеживается время окончания обслуживания $t_3=t_2+ T^1_{23}$. Определяется момент поступления второй заявки потока X2: $t_7=t_2+\tau_2^2$. Снова выбирается минимальное время — это t_3. В этот момент заявка потока X2 начинает обрабатываться процессором. По результату случайного испытания определяется время её обслуживания T^1_{21} и отмечается момент $t_5=t_3+ T^1_{21}$ окончания обслуживания. Следующее минимальное время t_4 - момент завершения обслуживания заявки потока X1 устройством 4. С этого момента заявка может начать обрабатываться процессором, но он занят обслуживанием потока X2. Тогда заявка потока X1 переходит в состояние ожидания, становиться в очередь. В следующий момент времени t_5 освобождается процессор. С этого момента процессор начинает обрабатывать заявку потока X1, а заявка потока X2 переходит на обслуживание дисплеем, т.е. ответ на запрос пользователя передаётся из основной памяти в буферный накопитель дисплея. Далее определяются соответствующие времена обслуживания: T^1_{11} и T^1_{23} и отмечаются моменты времени $t_9=t_5+ T^1_{11}$ и $t_6=t_5+ T^1_{23}$. В момент t_6 полностью завершается обработка первой заявки потока X2. По разности времени t_6 и t_2 вычисляется время реакции по этой заявке $u^1_2= t_6- t_2$. Следующий минимальный момент t_7 - это наступление 2-ой заявки потока X2. Определяет время поступления очередной заявки этого потока $t_{15}= t_7+\tau_2^3$. Затем вычисляется время

обслуживания 2-ой заявки на дисплее T^2_{23} и отмечается момент $t_8=t_7+ T^2_{23}$, после чего заявка становится в очередь, т.к. процессор занят. Эта заявка поступит на обслуживание в процессор только после его освобождения в момент t_9 . В этот момент заявка потока X1 начинает обслуживаться в АЦПУ. Определяются времена обслуживания T^2_{21} и T^1_{12} по результатам случайных испытаний и отмечаются моменты окончания обслуживания $t_{11}= t_9+T^2_{23}$ и $t_{10}= t_9+T^1_{12}$. В момент времени t_{10} завершается полное обслуживание 1-ой заявки потока X1. Разность между этим моментом и моментом времени t_1 даёт 1-ое значение времени реакции по потоку X1 $u^1_1= t_{10}- t_1$.

Указанные процедуры выполняются до истечения времени моделирования. В результате получается некоторое количество (выборка) случайных значений времени реакции (u_1) и (u_2) по 1-ому и 2-ому потокам. По этим значениям могут быть определены эмпирические функции распределения и вычислены количественные вероятностные характеристики времени реакции. В процессе моделирования можно суммировать продолжительности занятости каждого устройства обслуживанием всех потоков. Например, на рис. 2 занятость процессора 1 выделена заштрихованными ступеньками. Если результаты суммирования разделить на время моделирования, то получатся коэффициенты загрузки устройств.

Можно определить время ожидания заявок в очереди, обслуженных системой, среднюю и максимальную длину очереди заявок к каждому устройству, требуемая ёмкость памяти и др.

Имитация даёт возможность учесть надёжностные характеристики ВС. В частности, если известны времена наработки на отказ и восстановления всех входящих в систему устройств, то определяются моменты возникновения отказов устройств в период моделирования и моменты восстановления. Если устройство отказало, то возможны решения:

- снятие заявки без возврата;
- помещение заявки в очередь и дообслуживание после восстановления;
- поступление на повторное обслуживание из очереди;

1.3 Моделирование систем и языки программирования.

Большое значение при реализации модели на ЭВМ имеет вопрос правильного выбора языка программирования.

Язык программирования должен отражать внутреннюю структуру понятий при описании широкого круга понятий. Высокий уровень языка моделирования значительно упрощает программирование моделей.

Основными моментами при выборе ЯМ является:

- проблемная ориентация;
- возможности сбора, обработки, вывода результатов;
- быстродействие;
- простота отладки;
- доступность восприятия.

Этими свойствами обладают процедурные языки высокого уровня. Для моделирования могут быть использованы языки Имитационного моделирования (ЯИМ) и общего назначения (ЯОМ).
Более удобными являются ЯИМ. Они обеспечивают:

- удобство программирования модели системы;
- проблемная ориентация.

Недостатки ЯИМ:

- неэффективность рабочих программ;
- сложность отладки;
- недостаток документации.

Основные функции языка программирования:

- управление процессами (согласование системного и машинного времени);
- управление ресурсами (выбор и распределение ограниченных средств описываемой системы).

Как специализированные языки, ЯИМ обладают некоторыми программными свойствами и понятиями, которые не встречаются в ЯОН. К ним относятся:
Совмещение. Параллельно протекающие в реальных системах S процессы представляются с помощью последовательно работающей ЭВМ. ЯИМ позволяют обойти эту трудность путём введения понятий системного времени.
Размер. ЯИМ используют динамическое распределение памяти (компоненты модели системы М появляются в ОЗУ и исчезают в зависимости от текущего состояния. Эффективность моделирования достигается так же использованием блочных конструкций: блоков, подблоков и т.д.
Изменения. ЯИМ предусматривают обработку списков, отражающих изменения состояний процесса функционирования моделируемой системы на системном уровне.
Взаимосвязь. Для отражения большого количества между компонентами модели в статике и динамике ЯИМ включаем системно организованные логические возможности и реализации теории множеств.
Стохастичность. ЯИМ используют специальные программные генерации последовательностей случайных чисел, программы преобразования в соответствующие законы распределения.
Анализ. ЯИМ предусматривают системные способы статистической обработки и анализа результатов моделирования.
Наиболее известными языками моделирования являются SIMULA, SIMSCRIPT, GPSS, SOL, CSL.
Для языков, используемых в задачах моделирования, можно составить классификацию следующего вида. (см. рис. 9.1.)

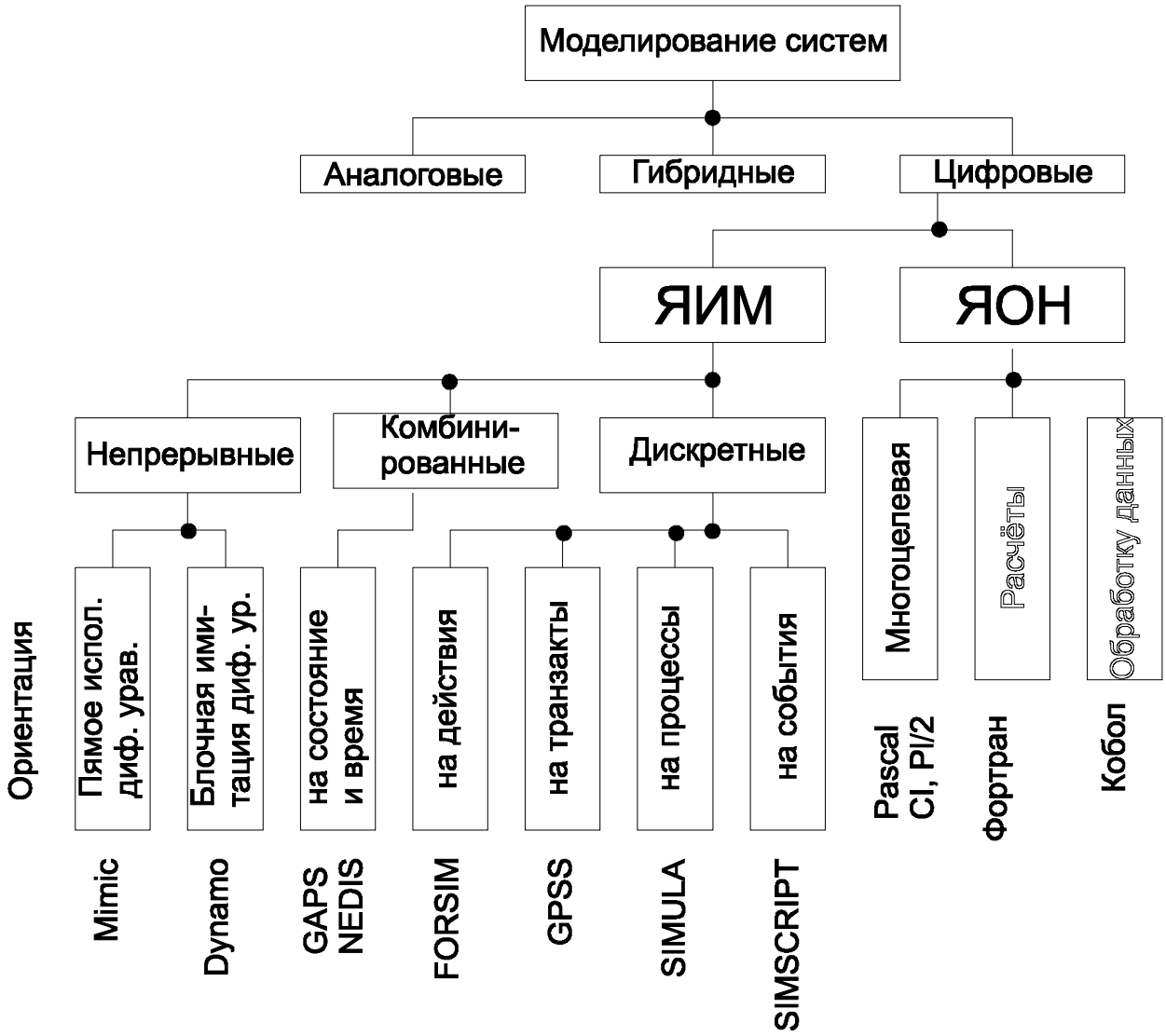

Рис. 9.1. Классификация языков моделирования.

Язык DYNAMO используется для решения разностных уравнений.

Представление системы S в виде типовой схемы, в которой участвуют как дискретные, так и непрерывные величины, называются комбинированными. Предполагается, что в системе могут наступать события двух видов: 1) события, от состоянии Z_i; 2) события, зависящие от времени t. При использовании языка GAPS на пользователь возлагается работа по составлению на яз. FORTRAN подпрограмм, в которых описываются условия наступления событий, законы изменения непрерывной величины, правил перехода из одного состояния в другое. SIMSCRIPT - язык событий, созданный на базе языка FORNRAN. Каждая модель M_j состоит из элементов, с которыми происходят события, представляющие собой последовательность формул, изменяющих состояние моделируемой системы с течением времени. Работа со списками, определяемые пользователем, последовательность событий в системном времени, работа с множествами. FORSIT - пакет ПП на языке FORNRAN позволяет оперировать только фиксированными массивами данных, описывающих объекты моделируемой системы. Удобен для описания систем с большим числом разнообразных

ресурсов. Полное описание динамики модели можно получить с помощью ПП.

SIMULA - расширение языка ALGOL. Блочное представление моделируемой системы. Функционирование процесса разбивается на этапы, происходящие в системном времени. Главная роль в языке SIMULA отводится понятию параллельного оперирования с процессами в системном времени, универсальной обработки списков с процессами в роли компонент.

GPSS- интегрирующая языковая система, применяющаяся для описания пространственного движения объектов. Такие динамические объекты в языке GPSS называются транзактами и представляют собой элементы потока. Транзакты "создаются" и "уничтожаются". Функцию каждого из них можно представить как движение через модель М с поочерёдным воздействием на её блоки. Функциональный аппарат языка образуют блоки, описывающие логику модели, сообщая транзактам, куда двигаться и что делать дальше. Данные для ЭВМ подготавливаются в виде пакета управляющих и определяющих карт, которым составляется по схеме модели, набранной из стандартных символов. Созданная программа GPSS, работая в режиме интерпретации, генерирует и передаёт транзакты из блока в блок. Каждый переход транзакта приписывается к определенному моменту системного времени.

При моделировании предпочтение отдают языку, который более знаком, универсален. Вместе с увеличением числа команд возрастают трудности использования ЯИМ. Получены экспертные оценки ЯИМ по степени их эффективности.

Баллы	Возможности	Простота применения	Предпочтение пользователя
5	SIMULA	GPSS	SIMSCRIPT
4	SIMSCRIPT	SIMSCRIPT	GPSS
3	GPSS	SIMULA	SIMULA

Суммарный бал:

SIMULA -11
SIMSCRIPT -13
GPSS -12

Если предпочтение отдаётся блочной конструкции модели при наличии минимального опыта в моделировании, то следует выбрать язык GPSS, но при этом следует помнить, что он негибок, требует большого объёма памяти и затрат машинного времени для счёта.