Саратовский госуниверситет им. Н.Г. Чернышевского

кафедра вычислительного
эксперимента в механике

Маркушин А.Г.

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

С ПРИМЕНЕНИЕМ ЯЗЫКА “СИМУЛА – 67”

Саратов – 2010

Оглавление

ГЛАВА 1. Основные понятия имитационного моделирования

1 Модели 4

2 Структура имитационной модели 6

3 Построение имитационной модели 8

4 Методологические подходы в имитационном моделировании 11
4.1 Дискретное моделирование 12
4.2 Событийный подход 13
4.3 Подход сканирования активностей 13
4.4 Процессно - ориентированный подход 13
4.5 Непрерывное имитационное моделирование 14
4.6 Комбинированные дискретно - непрерывные модели 14

5 Языки имитационного моделирования 14

ГЛАВА 2 Моделирование на языке Симула - 67 33

1 Понятие объекта в языке Симула-67 35
1.1 Объекты и классы объектов 35
1.2 Создание объектов и их наименование 36
1.3 Иерархическое описание классов объектов 38

2 Средства доступа к атрибутам объектов 40
2.1 Дистанционные идентификаторы 40
2.2 Присоединяющие операторы. 40
2.3 Виртуальные атрибуты 40

3 Ввод - вывод в Симула-67 41

4 Средства имитационного моделирования 41

5 0 средствах для сбора статистики и стохастического моделирования 48

6 Программа моделирования работы магазина 50
6.1 Постановка задачи 50
6.2 Описание программы моделирования 51

Список литературы 52

ГЛАВА 1

Основные понятия имитационного моделирования

Проблемы, с которыми человеку приходится сталкиваться в различных сферах своей деятельности, постоянно усложняются, это определяет необходимость совершенствования уже имеющихся и разработки новых методов их решения. С появлением электронных вычислительных машин одним из наиболее эффективным средством анализа и решения сложных проблем стало имитационное моделирование.

Само по себе моделирование имеет давнюю историю - формирование понятия моделирования и разработка моделей играли жизненно важную роль в духовной деятельности человечества с тех пор, как оно стало стремиться к пониманию и изменению окружающей среды. Люди всегда использовали концепцию модели, пытаясь представить и выразить с ее помощью абстрактные идеи и реальные объекты. Моделирование охватывает широкий диапазон актов человеческого общения от наскальной живописи и сооружения идолов до составления систем сложных математических уравнений, описывающих полет ракеты в космическом пространстве. По существу прогресс и история науки и техники нашли свое наиболее точное выражение в развитии способности человека создавать модели естественных явлений, понятий и объектов.

Имитировать, согласно словарю, значит "вообразить, постичь суть явления, не прибегая к экспериментам на реальном объекте". По существу, каждая модель или представление вещи есть форма имитации. Имитационное моделирование является широким, но недостаточно четко определенным понятием. Кратко его можно сформулировать так: имитационное моделирование есть процесс конструирования модели реальной системы и постановки экспериментов на этой модели с целью либо понять поведение системы, либо оценить различные стратегии, которые обеспечивали бы нормальное функционирование данной системы. Таким образом, процесс имитационного моделирования понимается как процесс, включающий и конструирование модели, и применение модели для изучения некоторой проблемы. Идея имитационного моделирования дает возможность пользователю экспериментировать с системами в тех случаях, когда делать это на реальном объекте практически невозможно или нецелесообразно. Но нельзя ограничивать понятие имитационного моделирования лишь экспериментами, проводимыми с помощью машинных моделей, много полезных видов имитационного моделирования может быть осуществлено и осуществляется всего лишь при помощи листа бумаги и пера. Поэтому имитационное моделирование является экспериментальной и прикладной методологией, имеющей целью:
описать поведение систем; построить теории и гипотезы, которые могут объяснить наблюдаемое поведение; использовать эти теории для предсказания будущего поведения системы.

В отличие от большинства технических методов, имитационное моделирование применимо в любой отрасли науки. Как известно, оно получило свой первоначальный толчок в ходе реализации авиа - космических программ, но краткий обзор литературы по имитационному моделированию показывает, сколь обширна сфера его применения. Так, например, написаны книги по применению имитационного моделирования в коммерческой деятельности [18], экономике [17], маркетинге [19], в системе образования [21], политике [25], обществоведении [20], науке о поведении [23], международных отношениях [24], на транспорте [22], в кадровой политике, в области соблюдения законности, в исследовании проблем городов и глобальных систем, а также во многих других областях, это бесчисленное множество книг, технических статей, отчетов в различных сферах человеческой деятельности свидетельствует о росте использования и распространения влияния имитационного моделирования почти на все стороны нашей жизни.

Модели

Модель является описанием объекта, системы или понятия в некоторой форме некоторыми средствами. Модель служит обычно средством, помогающим в объяснении, понимании или совершенствовании системы. Модель какого - либо субъекта может быть или точной копией этого объекта, или отображать некоторые характерные свойства объекта в абстрактной форме.

Идея представления некоторого объекта, системы или понятия при помощи модели носит настолько общий характер, что перечислить все функции модели очень трудно. Но выделяют пять ставших привычными случаев применения моделей в качестве: средства осмысления действительности, средства общения, средства обучения и тренировки, инструмента прогнозирования, средства постановки экспериментов.

Модели как средства осмысления действительности могут помочь нам упорядочить наши нечеткие или противоречивые понятия. Правильно построенная модель вынуждает нас организовать наши замыслы, оценить и проверить их обоснованность. Например, представление работ по проектированию сложных систем побуждает продумать какие шаги и в какой последовательности необходимо предпринять. Такая модель помогает выявить какие-то взаимосвязи, соотношения и т.д.

Полезность модели как средства общения очевидна. Все языки, в основе которых лежит слово, в той или иной мере оказываются не точными, когда дело доходит до сложных понятий и описаний. Правильно построенные модели могут помочь нам устранить эти неточности, предоставляя в наше распоряжение более действенные, более успешные способы общения. Преимущество модели перед словесными описаниями - в сжатости и точности представления заданной ситуации. Модель делает более понятной общую структуру исследуемого объекта и вскрывает важные причинно-следственные связи.

Модели применялись и применяются как превосходные средства обучения лиц, которые должны уметь справляться с всевозможными случайностями до возникновения реальной критической ситуации. Например, все знакомы с такими применениями моделей как натурные макеты или модели космических кораблей, используемые для тренировки космонавтов, тренажеры для обучения водителей автомашин и деловые игры для обучения персонала фирм и предприятий.

Одно из наиболее важных применений моделей является прогнозирование поведения моделируемых объектов. Строить какой-либо объект для опредения его специфических характеристик экономически не целесообразно, однако они могут быть предсказаны средствами моделирования. Так, при полете космического корабля "Апполон-13" имитационное моделирование было применено для анализа чрезвычайных мер до того, как были даны команды на их осуществление; эти меры дали возможность космонавтам возвратиться на Землю невредимыми после взрыва баллона с кислородом.

Применение моделей также позволяет проводить эксперименты в ситуациях, где экспериментирование на реальных объектах было бы практически невозможным или экономически не целесообразным. Экспериментирование с системой обычно состоит в измении ее некоторых параметров; при этом, поддерживая все остальные параметры неизменными, наблюдают результаты эксперимента. Когда ставить эксперимент на реальной системе слишком дорого или невозможно, зачастую может быть построена модель, на которой необходимые эксперименты могут быть проведены с относительной легкостью и недорого.

Таким образом, модель может служить для достижения одной из двух целей - описательной, если модель служит для объяснения или лучшего понимания объекта, или предписывающей, когда модель позволяет предсказать поведение объекта.

Модели вообще подразделяют на два основных класса: вещественные (иконические) и символические. Символические делятся на словесно-описательные и математические модели.

Словесно-описательные модели позволяют достаточно полно описать процесс, ситуацию или объект, но их невозможно использовать для анализа формализованным путем, поэтому словеснно-описательные модели преобразуют в математические, если это возможно.

Математическая модель - это совокупность математических объектов (чисел, переменных, векторов, матриц, множеств и т.д.) и отношений между ними, которые отображают некоторые свойства моделируемого объекта. Например, большой класс математических моделей составляют системы уравнений.

Формирование математических моделей называют моделированием объекта. Но в последнее время к понятию моделирования стали относить еще и оперирование моделью с целью получения полезной информации об исследуемом объекте.

В зависимости от характера отображаемых свойств моделируемого объекта математические модели делятся на функциональные и структурные. Функциональная модель отображает процессы функционирования объекта: они чаще всего имеют форму систем уравнений.

Нередки случаи использования моделей, отображающие только структурные свойства объектов, например, геометрические. Структурные модели могут иметь форму графов, матриц, векторов и отображать взаимное расположение элементов в пространстве, наличие непосредственных связей между ними и т.д.

По способам получения функциональные модели делят на теоретические и формальные. Первые из них строятся на основе физических закономерностей. Последние получают на основе проявления свойств моделируемого объекта во внешней среде.

К сожалению, не всегда возможно создать математическую модель. При исследовании большинства промышленных систем можно предусмотреть, чтобы модель подчинялась техническими экономическим законам. При этом могут быть представлены в той или иной математической форме существенные связи в системе. В отличии от этого решение проблем защиты от загрязнения воздушной среды, предотвращения преступлений, здравоохранения и роста городов связано с неясными и противоречивыми целями. Следовательно, определение модели должно включать в себя как количественные, так и качественные характеристики.

Структура имитационной модели

Имитационной моделью называется логико-математическое описание системы. Благодаря машинным экспериментам с построенной имитационной моделью какой-либо реальной системы можно сделать выводы о поведении этой системы:

без ее физического построения, если это проектируемая система; без вмешательства в ее функционирование, если это действующая система; экспериментирование с которой или слишком дорого, или небезопасно; без ее разрушения, если цель эксперимента состоит в определении пределов воздействия на систему.

Таким образом, имитационные модели могут использоваться для проектирования, анализа и оценки функционирования систем.

В имитационном моделировании предполагается, что систему можно описать в терминах, понятных вычислительной системе.

Прежде чем начать разработку моделей, необходимо понять, что собой представляет структурные элементы, из которых она состоит. Хотя математическая или физическая структура модели может быть очень сложной, основы ее построения просты. В самом общем виде структуру моделей мы можем представить математически в виде:

$$E=f\ (x[i],y[j]),$$

где E результат действия системы; x[i] - переменные и параметры, которыми мы можем управлять; y[j] - переменные и параметры, которыми мы управлять не можем; f - функциональная зависимость между x[i] и y[j]. которая определяет величину E.

Каждая модель представляет собой некоторую композицию таких составляющих как :

- компоненты
- переменные,
- параметры,
- функциональные зависимости,
- ограничения,
- целевые функции.

Компоненты представляют собой части, которые при соответствующем объединении образуют систему. Можно считать компонентами элементы системы или ее подсистемы. Например, модель города можно составить из таких компонентов как система образования, система здравоохранения, транспортная система и т.д.

Параметры - величины, которые можно выбирать произвольно, в отличие от переменных, которые могут принимать значения, определенные видом данной функции, т.е. параметры, после того как они установлены, являются постоянными величинами, не подлежащими изменению.

В модели системы различаются переменные двух видов - экзогенные и эндогенные. Экзогенные переменные называются также входными. Это значит, что они порождаются вне системы или являются результатом воздействия внешних причин. Эндогенными переменными называются переменные, возникающие в системе или в результате воздействия внутренних причин. Можно также назвать эндогенные переменные переменными состояния (когда они характеризуют состояние или условия, имеющие место в системе) либо выходными переменными (когда речь идет о выходах системы).

Функциональные зависимости описывают поведение переменных и параметров в пределах компонента или выражают соотношение между компонентами системы. Эти соотношения являются либо детерминистскими, либо стохастическими. Детерминистские соотношения - это тождества и определения, которые устанавливают зависимость между определяющими переменными или параметрами тогда, когда процесс на выходе системы однозначно определяется заданной информацией на входе. Стохастические соотношения при заданной входной информации дают на выходе неопределенный результат. Оба типа соотношений выражаются в форме математического уравнения, которое устанавливает зависимость между эндогенными и экзогенными переменными.

Ограничения представляют собой устанавливаемые пределы изменения значений переменных или ограничивающие условия распределения тех или иных средств. Ограничения могут вводиться разработчиком или самой

системой. Таким ограничением можно считать, например, максимальный и минимальный уровень занятости рабочих.

Целевая функция или функция критерия - это точное отображение целей или задач системы. Есть два типа целей: сохранение и приобретение. Цели сохранения связаны с сохранением или поддержанием каких-либо ресурсов или состояний. Цели приобретения связаны с приобретением новых ресурсов или достижением определенных состояний, к которым стремится организация или руководитель. Функция критерия обычно является составной частью модели и весь процесс манипулирования с моделью направлен на оптимизацию или удовлетворение заданного критерия.

В процессе экспериментальной работы обычно используется двух-этапное исследование модели с целью выбора наилучшего варианта ее функционирования. На первом этапе производится изменение атрибутов одной или нескольких компонент системы. На втором этапе происходит изменение операционных правил, выполняется повторение первого этапа с новыми операционными правилами. На основе проведенных расчетов производится выбор наилучшего варианта.

Для оценки возможности практического использования результатов имитации необходимо указать все предпосылки, которые приняты в начале работы. Эти предпосылки могут быть различными:
предположение о несущественности ряда факторов в модели, что позволяет пренебречь ими и упростить модель;
предположение о законе распределения некоторой случайной величины;
предположение о линейности или специальной форме некоторых отношений; предположение о возможности представления некоторой непрерывной величины с помощью дискретной.

Разработка исходной математической модели основана на этих предпосылках. Как уже упоминалось, реальная система и ее динамика могут быть описаны в терминах компонент, переменных, параметров и взаимосвязей между ними. На основе этого представления строится математическая модель развития реальной системы, которая должна отражать все основные черты и особенности изучаемой системы.

В этой модели находят свое выражение элементы всех указанных типов. Дается количественное описание с учетом их характеристик. Переменные выражаются и описываются как элементы функциональных соотношений. Параметры определяются как постоянные величины, значения которых могут быть изменены лишь по желанию исследователя. Взаимосвязи между элементами системы выражаются с помощью алгебраических и логических формул.

Все эти выражения вместе и образуют математическую модель исследуемой системы как упрощенное представление реальной системы, сохраняющее ее основные черты.

Построение имитационной модели

Одним из главных элементов, необходимых для эффективного решения сложных задач, является построение и использование модели.

Разработка моделей - процесс сложный и во многом близок к искусству. Этот процесс можно упростить, если: известны физические законы, описывающие функционирование системы; может быть разработано графическое представление системы; можно управлять входами и выходами системы.

Разработка имитационной системы состоит из нескольких основных этапов. Прежде всего, формулируется главная цель, или цели, предполагаемого эксперимента. Эта цель может быть выражена, например, в виде требования проверить некоторые предположения, или в виде требования изучить поведение реальной системы в определенных условиях. Таким образом, следует строить модель, ориентированную на решение конкретных вопросов.

После того как цель эксперимента поставлена, необходимо заняться постановкой задачи. Эйнштейн сказал, что правильная постановка задачи даже более важна, чем ее решение. Для того, чтобы найти приемлемое или оптимальное решение задачи, необходимо сначала знать, в чем она состоит. Важной частью постановки задачи является определение характеристик системы, подлежащей изучению. Поэтому надо определить цели и ограничения, которые необходимо учитывать в процессе построения формальной модели. Определив цели исследования и границы системы, далее реальную систему сводят к логической блок - схеме или к статической модели. Описание системы для целей имитационного моделирования состоит из двух частей: статического и динамического представлений. На этапе формирования статического представления должны быть решены следующие вопросы: какие компоненты системы будут включены в модель, какие элементы будут исключены или будут считаться частью окружающей среды и какие структурные взаимосвязи будут установлены между ними. На втором этапе описания рассматриваются изменения. Здесь выясняется: какие изменения состояния возможны в системе и окружающих условиях и какова последовательность этих изменений.

Первой задачей является приобретение нужных сведений о системе, подлежащей моделированию. По мере того как в процессе разработки знания о системе углубляются, в модель вносятся необходимые изменения. Поэтому наиболее эффективный подход состоит в том, чтобы начинать разработку с общей количественной трактовки и затем постепенно детализировать ее по мере расширения наших знаний о моделируемой системе.

Каждое исследование охватывает и сбор данных, под которым обычно понимают получение каких-то численных характеристик. Наиболее очевидными источниками данных являются измерения и наблюдения, но также важны и некоторые другие средства получения информации, такие, как документы, интервью и личное участие в работе. К документам относятся руководства, чертежи, инструкции, отчеты, спецификации, корреспонденция и т.д. В большинстве организаций имеется обычно обширное

делопроизводство и ведется документация. Важно найти и соответствующим образом интерпретировать документы.

Исходный материал исследования представляется в виде выборочной совокупности, поэтому необходимо решить ряд проблем, связанных с выборочными обследованиями, такими как объем выборки, доверительные интервалы и т.д. При разработке плана имитационного исследования большую роль играют вопросы оценки параметров исследуемой системы, выполняемой с помощью анализа и обработки данных о поведении системы. При этом собранные данные используются для:

- определения типа распределения случайной величины с помощью критерия возможности совпадения выборочных данных с фиксированным известным распределением;
- определения таких параметров распределения, как среднее значение, дисперсия, пределы изменения случайной величины;
- определения отношений между компонентами системы с помощью корреляционного и регрессионного анализа.

Очень часто на практике исследователь сталкивается со следующей специфической чертой моделирования: каждый машинный прогон имитационного моделирования дает результаты, которые действительны только для определенных значений параметров и переменных. Даже после осуществления большого числа имитационных прогонов обычно довольно трудно сделать вывод о том, как функционирует имитируемая система. В процессе построения имитационной модели исследователь узнает особенности функционирования лишь отдельных ее компонентов. Однако, проникновение в поведение всей системы возможно только при выполнении многих машинных прогонов имитационной программы. Эти прогоны дают массу выходных данных. Поэтому выходные данные должны быть статистически обработаны и описаны путем использования таких величин, как среднее значение, среднеквадратическое отклонение, коэффициенты корреляции и т.д.

Таким образом, перед исследователем всегда стоит проблема выбора метода сбора информации. Эта проблема получила название стратегического планирования.

Наряду с понятием стратегического планирования существует понятие тактического планирования, которое связано с определением способов проведения имитационных прогонов, намеченных планом эксперимента. Тактическое планирование, прежде всего, связано с решением задач двух типов: определение начальных условий в той мере, в какой они влияют на достижение установившегося режима, при одновременном сокращении необходимых размеров выборки. Основными вопросами решающимися в рамках тактического планирования являются:

- продолжительность имитационного прогона;
- задание начальных условий;

- требуемое число повторений каждого прогона с различными последовательностями чисел;
- оценка точности получаемых результатов (доверительные интервалы);
- уменьшение дисперсии выходов.

Имитационный прогон обычно заканчивается в случае, когда происходит некоторое критическое событие. Все возможные критические события можно разделить на два типа:

- критические события по состоянию, т.е. имитационный прогон заканчивается, когда в модели фиксируется критическое состояние;
- критическое по времени событие, т.е. когда имитационное время достигает заранее заданной величины.

Примером критического по состоянию события является, например, поломки оборудования, после которых техническая система полностью выходит из строя; превышение максимальной длины возможной очереди в различных системах массового обслуживания и т.д. Критические модели по времени события используются в тех случаях, когда экспериментатор изучает с помощью модели функционирование системы на некотором отрезке времени. В ряде же случаев интервал моделирования определяется в ходе имитационного эксперимента.

Время достижения модельно установившегося состояния находится в прямой зависимости от задаваемых начальных условий. Теоретически начальные условия должны быть выбраны в соответствии с совместной вероятностью их наступления, характерной для установившегося состояния. Однако на практике это не осуществимо. Часто используется способ задания нулевых начальных условий, которые соответствуют крайне редким состояниям системы. Например, имитация процесса производства начинается с состояния системы, когда никакая продукция не находится в производстве и все станки в исправном состоянии. Каждый прогон таких имитационных программ начинается с событий, вызванных искусственно введенными начальными условиями и имеющими низкую вероятность появления в дальнейшем.

После завершения этапов разработки и планирования осуществляется прогон модели с целью получения желаемой информации. Этот этап называется экспериментированием. В отличии от экспериментирования с реальными системами пользователь может изменять по своему желанию любой параметр и суслить о поведении модели по наблюдаемым результатам.

Следующий этап - анализ полученных результатов при подготовке выводов или проверка гипотез о функционировании реальной системы. Здесь используются методы математической статистики.

Последний этап - это реализация полученных решений и документирование имитационной модели и результатов ее использования. Ни один из имитационных проектов не должен считаться успешно завершенным до тех пор* пока, результаты не станут использоваться в

процессе принятия решений. Успех реализации напрямую зависит от того, насколько правильно разработчик модели выполнил все предыдущие этапы имитационного исследования.

Методологические подходы в имитационном моделировании

В случае, когда разработчик имитационной модели использует какой-либо язык имитационного моделирования, методологический подход описания модели неявно задается этим языком. Но если разработчик использует универсальный язык программирования, то он сам должен разработать такой подход.

Модели систем подразделяются на дискретные и непрерывно изменяющиеся. Одну и ту же систему можно представить как дискретно, так и непрерывно изменяющейся моделью.

При дискретной имитации зависимые переменные меняются дискретно в определенные моменты имитационного времени, называемые моментами совершения событий, хотя само время может быть непрерывным.

При непрерывной имитации зависимые переменные изменяются непрерывно, хотя само время может изменяться и дискретно.

При комбинированной имитации зависимые переменные могут изменяться дискретно, непрерывно или непрерывно с дискретными скачками. Время может меняться как непрерывно, так и дискретно.

Дискретное моделирование

Элементы дискретных систем, такие как люди, оборудование, заказы и т.д. включающиеся в имитационную модель, называются компонентами. Эти компоненты различаются своими типами, описываемыми различными характеристиками - атрибутами. Компоненты могут иметь одну или несколько общих характеристик, что позволяет объединить их в группы.

Целью дискретного моделирования является воспроизведение взаимодействий, в которых участвуют компоненты, и изучение поведения исследуемой системы. Для этого выделяются состояния системы и описываются действия, которые переводят ее из одного состояния в другое.

При дискретной имитации состояние системы может меняться только в моменты совершения событий. В большинстве языков моделирования дискретной имитации используется механизм продвижения времени, основанный на поиске следующего ближайшего события.

Функционирование дискретной имитационной модели можно задать следующим образом:

- определяя изменения состояния системы, происходящие в моменты совершения событий:
- описывая действия, в которых принимают участие элементы системы, или процесс, в котором они задействованы.

Событие происходит в тот момент времени, когда принимается решение о начале или окончании действия. Процесс - это ориентированная во времени последовательность событий, которая может состоять из нескольких действий. Эти понятия или представления лежат в основе трех методологических подходов к построению дискретных имитационных моделей, называемых обычно событийным подходом, подходом сканирования активностей (ориентирован на описание действий) и пронесено - ориентированным подходом.

Событийный подход

При событийном подходе система моделируется посредством идентификации изменений, происходящих в ней в моменты совершения событий. Имитация работы системы осуществляется путем выполнения упорядоченной во времени последовательности логически взаимосвязанных событий. При этом имитационное время продвигается от одного события к другому.

Для работы модели необходимо воспроизвести хронологию событий и причины вызывающие их появление в соответствующие моменты времени. Все это обеспечивает календарь событий, который первоначально содержит только отметку о первом событии. В ходе имитации системы возникновение других событий должно быть запланировано в календаре событий в соответствии с логикой функционирования системы.

Такие языки моделирования как GASP [7], [14], SIMSCRIPT [7], [16], [8] обладают встроенными средствами для ведения календаря событий.

Подход сканирования активностей

При использовании этого подхода разработчик описывает действия, в которых принимают участие элементы системы и задает условия, определяющие начало и конец этих действий. События, которые начинают или завершают действие, не планируются разработчиком модели, а имитируются по условиям, определенным для данного действия. Условия начала или окончания действия проверяются после очередного продвижения имитационного времени. Если заданные условия удовлетворяются, происходит соответствующее действие. Для того чтобы было выполнено каждое действие в модели, сканирование условий производится для всего множества действии при каждом продвижении имитационного времени.

Это подход обеспечивает простую схему моделирования для решения многих проблем, но все же менее эффективен в сравнении с событийным подходом, т.к. необходимо все время выполнять сканирование условия и поэтому этот подход имеет ограниченное применение в дискретном имитации.

Процессно - ориентированный подход

Имитационная модель нередко включает в себя одну или несколько однородных компонент, возникающих в модели по определенной схеме. Имитационные языки, включающие операторы для моделирования процесса прохождения элементов (компонент) через систему, обычно называются процессно-ориентированными. Процессно-ориентированный подход сочетает в себе черты событийного подхода и подхода сканирования активностей. Простота этого подхода состоит в том, что логика событий заложена в самом имитационном языке. Операторы языка определяют последовательность событий, которые автоматически выполняются системой, по мере того как элементы продвигаются через систему-модель. Однако этот подход является менее гибкими чем событийный.

К наиболее ярким представителям процессно-ориентированным языкам моделирования относится GPSS [4], [5], [6] , СИМУЛА-67 [7], [12], [13], Q-GERT [3].

Непрерывное имитационное моделирование

В непрерывном имитационном моделировании состояние системы представляется с помощью непрерывно изменяющихся переменных - переменных состояния. Такая модель создается посредством задания уравнений для совокупности переменных состояния, динамическое поведение которых и моделирует реальную систему. Эти уравнения нередко бывают дифференциальными, и математическая модель представляет собой задачу Коши для них.

В большинстве новейших непрерывных имитационных языков используется операторная ориентация, когда дифференциальные или разностные уравнения кодируются в явном виде. В настоящее время разработан набор стандартов для таких языков.

Комбинированные дискретно - непрерывные модели

В рамках методологии комбинированного моделирования исследуемая система описывается с помощью элементов, их атрибутов и переменных состояния, вычисляемых через небольшие отрезки времени.

В комбинированном моделировании применяют два типа событий: временные события, свершение которых планируется в определенные моменты временили события состояния, которые не планируются, а происходят тогда, когда система достигает определенного состояния.

Понятие событие состояниям аналогично понятию «сканирование активностей», в котором событие также не планируется, а инициируется определенным состоянием системы и возможность его возникновения проверяется при каждом продвижении имитационного времени.

Первым комбинированным языком является GASP-4 [7], [14]. Его возможности были затем введены в язык SIMSCRIPT [7], [8], [16]. Позже появились языки SMOOTH [3], SAINT [3] и CROPS [3]. Это направление в моделировании продолжает оставаться перспективным и в настоящее время.

Языки имитационного моделирования

Имитационное моделирование начинается с изучения моделируемой системы и описания ее в виде логических схем и функциональных взаимосвязей. Однако в конечном счете встает задача описания модели на том языке, который понятен машине. Модель можно описать на любом универсальном алгоритмическом языке, однако применение специализированных языков имитационного моделирования может дать значительные преимущества в смысле легкости, производительности и эффективности. В общем случае специализированные языки моделирования различаются методами организации времени и операций:

- наименованиями и структурой блоков модели;
- способами проверки операций и условий взаимодействия элементов;
- видами статистических испытаний, которым можно подвергнуть имеющиеся данные: легкостью изменения структуры модели.

Эволюционное развитие языков имитационного моделирования, начавшееся в конце 50-х годов, означало что в моделировании использовали либо языки, ориентированные на конкретную ЭВМ, либо универсальные языки. В начале 60-х годов несколько групп ученных почти одновременно пришли к идее создания специализированных языков. В своем развитии эти языки прошли через ряд стадий: от языков типа ассемблера с некоторыми специальными свойствами к более широким проблемно-ориентированным языкам, получившим значительное распространение на коммерческом рынке пользователей, и наконец к сложным специализированным языкам имитационного моделирования. Для целей имитации подходит любой алгоритмический язык программирования, но языки, специально созданные для имитации на ЭВМ, обладают рядом особых преимуществ такими как: снижение трудоемкости написания программы: обеспечение гибкости, необходимой для изменения программы: возможность четко классифицировать элементы системы: обеспечение более строгому следованию выбранной концепции; возможность корректировать число элементов модели и т.д. Среди возможностей языков имитационного моделирования нужно отметить следующие:

- способность генерировать случайные числа;
- возможность генерировать случайные переменные;
- возможность «продвигать» время либо на одну единицу, либо до следующего события: способность накапливать выходные данные;
- способность проводить статистический анализ накапливаемых данных;

- возможность выявлять и регистрировать логические несоответствия и другие ситуации, связанные с ошибками.

Некоторые из языков имитационного моделирования являются языками в широком смысле слова, т.е. служа пользователю средством общения с ЭВМ, облегчают формулирование задач. Имея словарь и синтаксис, они являются языками описательными.

Каждый язык имитационного моделирования базируется на определенной совокупности понятий, с помощью которой исследователь описывает состав, структуру и процессы функционирования моделируемой системы. Элементы моделируемой системы отображаются в имитационной модели в виде компонентов, которые в различных языках называются по-разному: ресурсы, транзакты, средства обслуживания, склады, сущности, процессы. В зависимости от характера поведения различают пассивные и активные, временные и постоянные компоненты. Компоненты активного типа, проходя через систему, вступают во взаимодействия с компонентами пассивного типа, чье поведение сводится к реакции на воздействия. Постоянные компоненты существуют в течение всего функционирования модели, временные компоненты генерируются и уничтожаются в течение этого периода.

Для обозначения характеристик компонентов используются понятия "атрибуты", "дескриптор", "параметр", "указатель". Каждой характеристике компонента в модели соответствует числовая или логическая переменная, значение которой определяет состояние компонента по данной характеристике. Значения всех характеристик определяют состояние отдельного компонента и состояние имитационной модели в целом.

Компоненты, обладающие одинаковым набором характеристик, образуют классы компонентов, и отдельные компоненты одного и того же класса отличаются друг от друга значениями характеристик. Обычно, в разных имитационных системах возникает необходимость сгруппировать компоненты, принадлежащих к одному или разным классам, по некоторым признакам. Поэтому языки имитационного программирования обладают для этого специальными средствами, называемыми цепь, очередь, набор, файл или склад .

В ходе имитации компоненты изменяют свои состояния, т.е. изменяют значения своих характеристик. Процесс функционирования моделируемой системы представляется как определенная последовательность взаимодействий, которая реализуется при выполнении определенных условий. Как независимые, так и зависимые переменные могут изменяться либо дискретно, либо непрерывно. Поскольку в большинстве имитационных моделей имитируется поведение системы на некотором отрезке времени, одной из наиболее важных задач при создании модели и выборе языка программирования является определение механизма регламентации событий и процессов. В имитационном моделировании понятие "регламентация" включает в себя две функции: "продвижение" по времени и обеспечение согласованности различных блоков и событий в системе.

Функционирование модели должно протекать в искусственном времени, обеспечивая появление событий в надлежащем порядке и с надлежащими временными интервалами между ними. Существуют два основных метода задания времени - с помощью фиксированных и переменных интервалов времени. Их называют соответственно методами фиксированного шага и шага до следующего события. По методу фиксированного временного шага отсчет системного времени ведется через заранее определенные временные интервалы постоянной длины. При использовании метода переменного шага состояние моделируемой системы обновляется с появлением каждого существенного события независимо от интервалов времени между ними.

Имитационные модели удобно классифицировать по двум основным категориям:

- модели с непрерывным изменением состояния:
- модели с дискретным изменением состояния;
- непрерывно-дискретные.

В первых используются механизмы фиксированных приращений временных интервалов; ими удобно описывать поведение систем, представляемых непрерывными потоками информации. Модели второго рода находят применение тогда, когда исследователя интересует поведение отдельных элементов в системе. В большинстве моделей с дискретным изменением состояния, используется метод отсчета времени до следующего события.

Поэтому же принципу можно разделить и языки имитационного моделирования. Непрерывные языки предназначены для моделирования систем, состояние которых изменяется непрерывным образом.

Язык CSMP/360 [7] является одним из первых непрерывных языков имитационного моделирования высокого уровня, создан на базе языков CSMP/1130 и PACTOLUS [7] и представляет собой пакет программ на ФОРТРАНЕ. CSMP/360 обладает возможностью гибкого внутри программного изменения параметров и диалогового режима работы с доступным к структуре самой модели. Интегрирование дифференциальных уравнений может производиться одним из семи алгоритмов различной сложности. В языке имеется возможность сортировки с помощью операторов NOSORT и SORT, имеется диагностика ошибок. Ввод и вывод осуществляется с помощью соответствующих операторов ФОРТРАНА.

В общем случае языки моделирования включают в себя средства обработки языковых конструкций (компилятор, транслятор или интерпретирующая программа) и систему выполнения имитационного процесса во времени. Поэтому термин "язык моделирования" употребляется неверно. Более подходящим является термин "система моделирования".

В настоящее время существует уже достаточно большое число разнообразных систем моделирования, однако, несмотря на это многообразие, можно выделить две тенденции их развития. Системы моделирования

первого типа - это специализированные языки моделирования с собственным синтаксисом и семантикой. Второй тип систем моделирования - это фактически пакеты прикладных программ предназначенные для моделирования и использующие в качестве базового некоторый алгоритмический язык.

Характерным представителем второго типа систем моделирования является GASP-4, являющийся дальнейшим развитием дискретного языка GASP-2. Пакет GASP-4 (General Activity Simulation Program) за короткое время завоевал себе широкую популярность за рубежом, которая объясняется наличием следующих качеств:

- он использует в качестве базового языка ФОРТРАН-4 и поэтому, не требует собственного компилятора: реализуем на любой ЭВМ, в состав математического обеспечения которого входит транслятор с этого языка;
- предназначен для реализации дискретных, непрерывных, а также комбинированных моделей: легко изучается и просто модифицируется.

Основой системы моделирования GASP-4 является концепция моделирования систем в двух измерениях: во времени и в пространстве состояний. Основными элементами языка являются компоненты моделей, их характеристики и множества. GASP-4 обеспечивает исследователя всеми основными функциями по проведению машинных экспериментов с имитационными моделями различных типов: установление начального состояния моделей: управление ходом имитационного эксперимента; продвижение имитационного времени различными способами в зависимости от типа моделей; интегрирование дифференциальных уравнений: сбор и вычисление статистических данных о наблюдаемых переменных; генерация случайных чисел с различными законами распределения: выдача данных экспериментов в виде таблиц, гистограмм и графиков.

Язык GASP-4 осуществляет различные способы продвижения имитационного времени. При непрерывной имитации исполнительная программа использует пошаговую процедуру продвижения времени. Если в модели содержатся только разностные уравнения, то исполнительная программа будет вычислять значения переменных в конце каждого шага. Если в модель включены дифференциальные уравнения, тогда вычисление значений переменных происходит неоднократно в течении шага. При каждом вычислении значений проводится проверка условий возникновения событий. Когда определено, какое событие должно произойти, тогда исполнительная программа вызывает соответствующую программу события.

При дискретной имитации исполнительная программа использует иную процедуру продвижения имитационного времени. Поскольку состояние системы может меняться только в моменты совершения событий, то имитационное время продвигается от момента свершения текущего события до момента свершения следующего. Эта процедура осуществляется с помощью файла (календаря) событий, в который помещаются записи (уведомления) событий, упорядоченные по времени их возникновения.

В непрерывно-дискретных моделях исполнительная программа в конце каждого шага вычисляет значения переменных для определения выполнения условий, обусловливающих возникновение событий. Событие будет пропущено, если хотя бы одна переменная своим значением пересекла предписанную границу. Это означает, что шаг был велик и он уменьшается. Исполнительная программа автоматически устанавливает размер шага так, чтобы внутри его не произошло ни одного события. Это достигается подбором размера шага таким образом, чтобы он заканчивался в момент свершения события.

Окончание имитации может быть определено либо посредством определения условий в терминах состояния модели, либо указанием времени окончания имитации. Данные о ходе эксперимента могут быть получены в моменты времени, определенные исследователем.

Одним из наиболее известных непрерывных языков имитационного моделирования является язык ДИНАМО [7], [14]. Он предназначен для моделирования систем и процессов на высоком уровне, где отображение дискретности отдельных элементов становится ненужным. Другой областью применения языка ДИНАМО является моделирование систем, процессы функционирования которых непрерывны.

Переменные модели и скорости их изменения определяют состояние имитационной модели и задаются системой разностных уравнений. В языке используется индексация переменных, соотносящая их с предшествующим, настоящим и будущим моментами имитационного времени: I - прошедший, К - настоящий, L - будущий моменты времени. Интервалы между этими точками обозначаются IK и KL. Длина интервала фиксирована и определяется исследователем посредством задания параметра DT. В процессе имитации в каждый момент К происходит вычисление значений переменных модели в конце очередного интервала KL .

Результаты имитационных экспериментов выводятся в форме таблиц и графиков. Имитационные модели, запрограммированные на языке ДИНАМО, легко могут быть перепрограммированы в термины GASP-4.

Непрерывный язык имитационного программирования MIMIC [7] базируется на подмножестве языка ФОРТРАН со специальными операторами описания и содержит аппарат логических меток для изменения структуры и управления выполнением программы в ходе имитации. Управление структурой модели осуществляется путем задания логических переменных, параметров и констант. Интегрирование дифференциальных уравнений производится методом Рунге - Кутта четвертого порядка. Ввод-вывод производится автоматически и с помощью специальных операторов.

Для моделирования систем, изменение состояния которых происходят в дискретные моменты времени, применяются дискретные языки имитационного программирования.

Важной составной частью дискретных языков является механизм синхронизации событий на данном временном интервале. Существуют два

основных метода синхронизации событий - поисковый метод выбора очередного события и метод планирования событий.

Поисковый метод выбора события (или метод сканирования) основан на проверке в определенные моменты времени условий возникновения событий. При выполнении заранее определенных условий происходит событие, соответствующее этим условиям. На каждом шаге имитационного времени происходит проверка всех условий, и, поскольку размер шага должен быть достаточно мал, данный метод требует больших затрат машинного времени.

Метод планирования событий состоит в определении в процессе имитации моментов совершения будущих событий. При достижении имитационным временем момента свершения запланированного события происходит передача управления к программе, соответствующей данному типу события. Имитационное время не изменяется до тех пор, пока все запланированные события с одинаковым временем свершения не будут выполнены.

Можно выделить три основных подхода к рассмотрению множества компонентов имитационной модели: планирование событий, сканирование активностей и взаимодействие процессов.

В подходе, ориентированном на планирование событий, в центре внимания находится событие. Языки, базирующиеся на подходе планирования событий, включают в себя списки текущих событий, в которые помещаются записи, соответствующие определенным событиям. Запись представляет собой уведомление о времени совершения события. При чем все записи располагаются в порядке увеличения времени совершения событий, что обеспечивает реализацию событий в хронологическом порядке. В этом случае имитационное время продвигается от события к событию. При использовании подхода сканирования активностей основным компонентом имитационной модели является активность (совокупность взаимодействий, свершаемых над определенным объектом). Все взаимодействия, относящиеся к одному активному компоненту, описываются в программе активности, которая состоит из контрольного и функционального блоков. В контрольном блоке определяются условия начала и окончания активностей, в функциональном блоке описываются взаимодействия между компонентами.

В процессе функционирования модели после каждого продвижения имитационного времени исполнительная программа просматривает (сканирует) контрольные блоки всех программ активностей. Если все условия, содержащиеся в контрольном блоке, выполнены, то управление передается функциональному блоку и данная активность выполняется. Если хотя бы одно из условий нарушено, то обращения к функциональному блоку не происходит.

Подход взаимодействия процессов включает некоторые элементы двух предыдущих подходов, но особенно акцентирует свое внимание на прохождении активного компонента через систему. Функционирование

моделируемой системы в этом случае представляется совокупностью процессов, каждый из которых описывает жизненный цикл компонента. Каждый процесс имеет несколько активных фаз, каждая из которых является событием, и несколько точек взаимодействия с другими процессами. Для каждого процесса периоды активности чередуются с неактивными периодами, во время которых активны другие процессы. Процесс может вновь стать активным с помощью точек реактивации, которые указывают те места в программе процесса, с которых ее надо начинать после того, как истечет указанное время прерывания или удовлетворится набор определенных условий. Отдельные части программы процесса могут выполняться в различные моменты имитационного времени на разных активных фазах.

Язык SMPL [8] является событийно-ориентированным и вполне пригоден для решения задач моделирования объектов с малой или средней размерностью модели.

Моделирующая программа на языке SMPL состоит из инициирующей секции, управляющей секции и ряда событийных секций. Эти секции могут существовать как части программы или подпрограммы.

В функции инициирующей секции входит возбуждение системы имитации во времени, определение языковых средств и очередей, а также планирование первого события. После этого управление передается в управляющую секцию. Эта секция устанавливает, какое событие следует первым, и инициирует выполнение соответствующей событийной секции. В общем случае события в языке SMPL представлены завершениями активностей. Событийная секция выполняет действия, соответствующие концу активности, определяет, какие активности могут выполняться в текущий момент, и планирует события, представленные завершениями этих активностей. После этого управление вновь передается в управляющую секцию.

Во время планирования событий процессы представляются целочисленными значениями; это значение является индексом массива, элементы которого задают значения атрибутов соответствующих процессов. В языке SMPL процессы рассматриваются как транзакты, а индекс, указывающий на соответствующий набор значений атрибутов, называется номером транзакта.

Упорядочение событий в языке SMPL достигается с помощью списка событий. Планирование события включает указание имени события, времени события и номера транзакта. Вся эта информация помещается в список событий, который подвергается сортировке в порядке возрастания значений времени. Выбрав очередное ближайшее событие, управляющая секция удаляет первый элемент из списка событий и передает управление событийной секции, номер которой был указан в этом элементе.

Постоянными объектами языка SMPL являются средства и очереди. Набор операторов языка позволяет определять эти объекты, резервировать

средства для использования определенным транзактом и освобождать их, а также включать транзакты в очереди и удалять их из очередей.

Выходные данные моделирования распечатываются в форме таблиц, содержащих имена очередей и средств, время ожидания, среднее время занятости и другу информацию.

Язык SIMSCRIPT основан на подходе планирования событий и включает два типа компонент: постоянные и временные. Каждому временному компоненту соответствует запись. Записи можно генерировать, стирать, помещать в различные списки и множества, используя операторы языка.

Каждый тип компонента описывается определенным набором характеристик. Характеристики компонентов могут быть временными и постоянными в зависимости от того, временный или постоянный компонент они определяют.

В языке SIMSCRIPT состояние моделируемой системы описывается в терминах компонентов, характеристик компонентов и целых множеств компонентов. Каждый тип компонента, характеристики и множества определяются исследователем в специальной стандартной форме.

Для управления ходом имитационного эксперимента используется сообщение о событии, в котором содержится идентификатор события и момент его возникновения. Сообщение о событии включается в общий список событий. Сообщение о событии уничтожается после его выполнения, а для следующего события создается новое сообщение. Программа-календарь следит за ходом имитационного времени и вызывает различные программы событий в нужной последовательности. Когда выполнение определенной программы события заканчивается, имитационное время продвигается до следующего события.

В языке SIMSCRIPT применяются события двух типов: внутрисистемные и внесистемные. Внутрисистемные события вызываются предшествующими событиями внутри модели и отображают внутренние связи компонентов. Внесистемные события определяются внешними связями и процессами, происходящими вне модели и влияющими на нее. В программу событий включаются специальные операторы для сбора статистики. Выдача данных о проведенном имитационном эксперименте осуществляется специальным генератором сообщений.

Язык МОДИС [7] относится к языкам имитационного моделирования, ориентированным на планирование событий. Описание моделируемой системы на языке МОДИС включает перечисление ее компонентов, связей между ними и определение внешних воздействий на систему. В языке имеется возможность моделировать многоуровневые иерархические системы. Компоненты системы, которая рассматривается как имеющая нулевой уровень (ранг), являются подсистемами первого ранга. Они в свою очередь состоят из подсистем второго ранга и т.д.

Подсистема максимального ранга содержит алгоритмы, которые образуются из блоков и предложений. Блоки и предложения складываются

из операторов, описывающих изменения значений переменных модели, а также порядок и условия их изменений.

Описание внешних воздействий является динамической частью модели, которая имеет вид временной диаграммы подаваемых на систему сигналов в форме упорядоченной последовательности.

Язык МОДИС включает две группы операторов: вычислительные, которые непосредственно изменяют значения переменных модели, и управляющие, которые служат для управления ходом имитации.

Имитационное время в модели изменяется за счет смены событий. Очередное событие определяется по списку будущих событий как событие, время наступления которого является ближайшим к текущему моменту имитационного времени. Характерной особенностью языка является то, что каждая компонента модели может существовать в своем, присущем только ей масштабе времени. Выравнивание масштабов в единый осуществляется указанием масштабного коэффициента времени для каждой компоненты.

Рассмотрим систему моделирования MISS [9]: Multi-lingual Integrated System for Simulation. Система ориентирована на такое моделирование, при котором важно явное воспроизведение хода времени. Эта система не первое средство, разработанное для применений в указанной области. Хорошо известен широкий класс алгоритмических языков имитационного моделирования, имеющих такое же назначение: СИМУЛА-67, СИМСКРИПТ и GSL [9].

MISS относится к процессно-ориентированным системам и в ее модельной концепции не трудно усмотреть элементы сходства с концепцией СИМУЛА-67 и с SDL-технологией проектирования и имитации систем связи, развиваемой за рубежом уже второй десяток лет. Технологически MISS выделяется прежде всего тем, что это не языковая система, как все упомянутые выше. Других инструментальных систем имитации практически не существует. Такова, в частности, созданная в начале 80-х годов во ВИНИСИ АН СССР система МИМ (Монитор Имитационного Моделирования). Однако, ее возможности существенно беднее предлагаемых MISS. Это можно сказать и про созданную в ИПУ АН СССР систему СИМОД [9].

Набор включенных в MISS инструментальных средств в основном определился из опыта конкретного моделирования. В этом наборе можно выделить группы средств "низкого" и "высокого" уровней. К низкоуровневым относятся средства, позволяющие виртуализировать память, оперировать списковыми структурами. По сути они совершенно универсальны и служат элементной базой системы MISS. Средства высокого уровня ориентированы на потребности имитации.

Высокоуровневую составляющую MISS можно разбить на блоки управления процессов, поддержки данных и графики. Первый включает развитый аппарат планирования машинного времени, а также службу передачи сигналов и определения правил чередования процессов. Второй обеспечивает автоматическое генерирование баз данных модели и удобный

доступ к их содержимому, как из программ имитации, так и в режиме диалога с терминала. Графический блок MISS предоставляет редакторы и процедуры для формирования пиктограмм модельных объектов.

Программы алгоритмов моделируемых процессов должны формироваться в одной из допускаемых MISS языковых систем программирования: МОДУЛА-2/SDS. МОДУЛА-2/JPI. TURBO-C, TURBO-C++.

Модельная концепция MISS объектно-ориентированна и исходит из представления о многокомпонентности предмета имитации.

При построении в MISS модели основой структуризации данных является разделение их по принадлежности приборам, объектам и группам.

Предметом имитации MISS является совокупность объектов. Концепция MISS также включает в себя понятие классов объектов, приборов и групп, а величины, чьими значениями исчерпываются различия между экземплярами одного класса, называются фазовыми переменными. По признаку постоянства состава на протяжении всего периода имитации в любом из наборов объектов могут выделяться статистические данные и динамические, состав которых с течением времени может меняться. Первые объединяются в одну запись. Что касается динамических данных, то для их представления в MISS предусмотрены средства работы с записями.

Функционирование модели видится слиянием деятельностей его объектов, а деятельность объекта представляется несколькими параллельно текущими процессами. В эволюции каждого процесса выделяются последовательные стадии, именуемые элементами.

Учитывая взаимно - однозначное соответствие процесс - прибор, можно сказать, что функционирование прибора есть последовательное выполнение им присущих ему элементов. Приборы во время своего функционирования могут вырабатывать сигналы, предназначенные конкретным адресатам (другим приборам или самому себе) и сопровождающиеся сообщениями, а также принимать сигналы и сообщения, посланные им. Связь адресатов с отправителями обеспечивается каналами, фиксируемыми в описаниях приборов, объектов и групп. Обычно сигналами моделируются импульсы, которые возникают в автоматической части управления исследуемого комплекса, а сообщениями - связанные с ними пересылки данных. Формально пара (сигнал, сообщение), являясь продуктом деятельности прибора, относится к его фазовым переменным.

По отношению к модельному времени все элементы всех приборов делятся на три категории: сосредоточенные, условно распределенные и распределенные, (сосредоточенные) элементы в модельном времени всегда выполняются мгновенно, выполнение же любого из прочих элементов имеет большую продолжительность.

Система MISS допускает квазипараллельное течение модельных процессов, которые реализуются через точки синхронизации. Это - точки оси модельного времени, к которым привязываются все выполняемые вычисления. Последние группируются в такты имитации. Это отрезки временной оси между точками синхронизации.

Значения модельного времени в точках синхронизации определяется взаимодействием двух механизмов. Первый - это назначение элементов на очередной такт, а второй - планирование моментов прерывания выполнения элементов.

В MISS каждый сосредоточенный элемент трактуется как один алгоритм, а условно распределенный как пара, включающая таймер и основной алгоритм, таймеры служат специальными процедурами планирования времени. При вызове таймера должен определяться очередной нужный элементу момент прерывания. Простейшая схема планирования моментов прерываний стоит на том, чтобы вызвать таймеры распределенных и условно распределенных элементов в начале каждого такта их выполнения.

В MISS реализован подход, согласно которому прибор формирует свою последовательность элементов как конечный автомат, имеющий входами номер предыдущего элемента и полученные сигналы, а выходом номер следующего) элемента. т.е. когда прибор завершил выполнение очередного элемента, выбор следующего элемента определяется тем, какой именно элемент завершился и какие из совокупности всех входных сигналов прибора поступили на момент принятия решения.

В пакете прикладных программ ДИСМ [7], [11] язык моделирования представляет собой совокупность универсального языка программирования PL-l и специальных процедурных средств пакета, предназначенных для реализации специфических операций имитационного моделирования. Такой синтез позволяет обусловить проблемную ориентацию языка (описание модели классов систем с дискретными событиями). Сохранив в то же время его универсальность и гибкость. При установлении специальных средств машинной имитации, предусмотренных в пакете, анализировались такие языки и системы дискретного моделирования как СИМУЛА, СИМСКРИПТ. SOL и СЛЭНГ [8], [16].

Для написания программы на этом языке функционирование моделируемой системы представляется в виде последовательности событий и между ними устанавливаются причинно-следственные связи.

Управление функционированием модели осуществляется управляющей программой, которая и определяет порядок выполнения событий. Язык ДИСМ содержит календарь событий, в котором уведомления о будущих событиях упорядочены по времени и приоритетам. В ситуациях, когда на один и тот же момент времени запланировано больше одного события, первым обрабатывается событие с меньшим приоритетом, а при их равенстве - в порядке занесения в календарь. Имитационное время в ДИСМ принимает только целые значения.

При использовании языка ДИСМ исследователь разрабатывает программу, состоящую из процедур событий и главный блок модели. Процедура события создается для каждого класса событий. Класс событий характеризуется набором параметров, включающих номер класса, к которому относится событие, указатель рабочего поля, приоритет события и момент наступления события. Указатель рабочего поля служит для ссылки

на некоторую структуру данных, относящихся к конкретному событию. В каждом конкретном случае исследователь сам устанавливает, требуется ли связывать с отдельным событием некоторую структуру данных или эта структура может относиться к ряду событий.

Главный блок модели, выполняющий запуск модели, содержит следующие разделы: нормирование модели для подготовки ее к прогону, порождение начальных событий и обращение к управляющей программе.

Для управляющей программы предусмотрены два режима функционирования: рабочий и тестовый. В отличие от рабочего при тестовом режиме прогон модели сопровождается выдачей контрольных сообщений об активном событии и состоянии календаря в моменты выполнения событий. Тестовый режим предназначен для отладки модели.

В языке ДИСМ имеются четыре группы процедур: процедуры для планирования событий и работы с очередями событий, процедуры генерации и преобразования случайных чисел, процедуры генерации статистической обработки результатов машинного эксперимента и процедуры для повышения качества разрабатываемых моделей и проведения их отладки.

Характерной чертой ДИСМ является наличие механизма планирования эксперимента. В ходе машинного эксперимента блок статистической обработки определяет объем выборки для прекращения процесса моделирования. Выдача результатов моделирования выполняется автоматически при достижении заданной точности и доверительной вероятности для исследуемых характеристик.

Моделирующий комплекс АЛСИМ-2 [10] представляет собой многоуровневую структуру, элементами которой является пакет прикладных программ и система автоматизации программирования, обеспечивающие поддержку создания программ имитационного моделирования, решение задач планирования эксперимента, сбора и обработки статистической информации, получаемой при проведении модельного эксперимента.

В моделирующем комплексе плюется универсальное средство создания моделей дискретных систем - система автоматизации программирования АЛСИМ-2/3. Эта система содержит входной язык, аналогом которого является язык описания моделей системы АЛСИМ - БЭСМ. В системе есть средства, позволяющие создавать библиотеки моделей в виде рабочих программ и собирать требуемые модели, используя эти библиотеки, перед началом интерпретации модели.

Основными средствами языка АЛСИМ являются следующие:

- модульной организации структуры программ
- программирования параллельных процессов
- сбора статистики обще алгоритмические.

К средствам модельной организации относятся понятия «модель» и оператор формирования модели. Описание модели определяет некоторый класс моделей. Между моделями различных классов могут существовать иерархические отношения. Средством генерации моделей является оператор

формирования модели. Модельная структура определяется доступностью к данным различных моделей, существованием различных связей между ними и наличием параметрических связей.

Можно выделить естественный доступ к данным внешних моделей из внутренних и специальный, обратный естественному.

Динамические связи между моделями устанавливаются с помощью входов и выходов операторами связи. Параметрические связи образуются при передаче некоторых параметров формируемым моделям. Эти параметры могут в процессе моделирования влиять на формирование внутренней структуры модели, устанавливать дополнительные связи к уже имеющимся объектам другой модели.

Средства программирования параллельных процессов являются традиционными для языков дискретного моделирования. Описание входа в модели определяет некоторый класс сообщений. Сообщения в программе моделирования служат для образования информационных потоков в модельном времени. Они характеризуются структурой данных и программой поведения. Исполнение сообщениями своих программ осуществляется в модельном времени параллельно. Сообщение может находиться в одном из четырех состояний: активном, ожидающем, пассивном и завершенном. Управление сообщением в модельном времени осуществляется с помощью операторов ожидания, планирования, пассивации, удаления. Указание конкретных сообщений данного класса осуществляется через ссылочные переменные и списки.

В качестве средств сбора статистики используются устройства памяти, таблицы, процедуры оценки случайных величин и процедуры генерации псевдослучайных последовательностей.

Общеалгоритмические средства языка включают в себя такие типы данных, как целые, вещественные, логические, кодовые и строковые переменные, а также возможность их обработки. Система допускает раздельную трансляцию отдельных частей программы. Такими частями могут быть описание моделей, входов, процедур.

Ввод-вывод представлен в языке средствами последовательного потоко -ориентированного ввода-вывода. Типичным представителем языков имитационного программирования, ориентированных на сканирование активностей, является язык CSL [7]. В этом языке использован поисковый метод синхронизации событий. Активность в CSL описывается программой, состоящей из контрольного и функционального блоков. В процессе имитации после каждого продвижения имитационного времени исполнительная программа сканирует контрольные блоки всех программ, определяя выполнение условий их выполнения. Если все условия, содержащиеся в контрольном блоке, выполнены, то управление передается функциональному блоку. Если хотя бы одно из условий не выполнено, то обращения к функциональному блоку не происходит.

Сбор, обработка и выдача статистических данных о поведении имитационной модели частично автоматизированы и позволяют исследователю получать результаты в стандартной форме.

Дальнейшим развитием CSL является язык ECSL [7], в котором содержится ряд новых операторов для манипулирования множествами данных.

Язык GPSS ориентирован на исследование систем, описываемых в терминах теории массового обслуживания, и основан на подходе взаимодействия процессов. Модель на языке GPSS включает компоненты активного (транзакты) и пассивного (приборы, накопители). Прибор может взаимодействовать только с одним транзактом, и к нему могут образовываться очереди, тогда как накопитель может взаимодействовать с несколькими транзактами. Максимальное число транзактов, одновременно взаимодействующих с накопителем, определяется исследователем. С каждым транзактом связано некоторое число параметров, которые определяются исследователем для задания характеристик этого транзакта.

Различные режимы функционирования моделируемой системы в модели отображаются с помощью функциональных блоков. В языке несколько десятков таких блоков, например: блоки генерации, продвижения, окончания, захвата, освобождения, переключения и т.д. В модели на языке GPSS транзакты продвигаются от одного блока к другому. Блок выполняет свои действия только тогда, когда через него проходит транзакт. Вход некоторого транзакта в блок составляет событие. Модель отображает моделируемую систему путем создания потока транзактов и управления ими по мере их продвижения по совокупности блоков.

Важной чертой языка GPSS является то, что на этапе разработки модели исследователь может описывать моделируемую систему в виде блок-схемы, используя стандартные символы. Каждый блок имеет определенный символ, которому соответствует оператор языка, что облегчает и процесс разработки модели, и ее программирование.

Взаимосвязи между компонентами представляются посредством явного или неявного определения элементов различных списков. Наиболее важными из них являются цепь текущих событий, цепь будущих событий и цепь задержки. Цепь текущих событий включает те транзакты, запланированное время наступления которых равно или меньше текущего имитационного времени. Цепь будущих событий включает транзакты. наступление которых планируется в будущем. Транзакты в цепи текущих событий размещены не только по времени наступления события, но и с учетом их приоритетов. Цепи задержек связаны с каждым прибором, накопителем или логическим переключателем и содержат те транзакты, обслуживание которых этим компонентом задерживается.

Продвижение имитационного времени в модели организовано по методу "от события к событию". Вначале каждого шага транзакт с наименьшим временем активизации извлекается из цепи будущих событий и

помещается в цепь текущих событий, далее предпринимается попытка продвинуть транзакт.

Так как операторы GPSS выполняются в режиме интерпретации, то имитационные программы на этом языке работают медленнее, чем на других языках.

Другим примером языков имитационного программирования, ориентированных на моделирование производственных систем в терминах теории массового обслуживания, является язык СИМДИС [7]. При использовании этого языка моделируемая система представляется в виде сети, узлами которой являются истоки и стоки требований с различными свойствами, обслуживающие устройства с известными характеристиками и устройства, управляющие потоками требований. Дуги, связывающие между собой узлы сети, имеют смысл потоков требований в системе обслуживания и указывают на схеме направление движения этих потоков.

В языке СИМДИС различают два типа элементов: временно действующие и постоянно действующие процессы. Элементы временного действия отображают потоки требований - их возникновение, поступление на обслуживание, ожидание в очереди, уничтожение после обслуживания. Элементы постоянного действия связаны с выполнением функций обслуживания потоков требований.

Элементы временного действия (потоки требований) реализуются в СИМДИС на базе специальных моделирующих элементов - активаторов. Активаторы используются для представления материальных сырьевых потоков, потоков заготовок или информационных процессов. При моделировании нарушений производственного процесса, например при выходе из строя технологического оборудования, связанные с этим процессы рассматриваются как требования особого вида и также представляются в модели активаторами.

Элементы постоянного действия, связанные с функционированием элементов моделируемой системы, описываются с помощью набора специальных моделирующих элементов. Такие элементарные средства моделирования процессов обслуживания могут быть использованы в модели как в качестве блоков прямого назначения для моделирования событий, так и в комбинации с другими элементами для представления в модели процессов большей сложности.

Подобно моделирующему элементу временного действия, моделирующие элементы постоянного действия реализуются в СИМДИС с помощью специальных структур данных: обслуживающее устройство, многоканальный узел обслуживания, склад, очередь ограниченной длины и т.д. Каждый элемент постоянного действия задается вектором признаков, компонентами которого являются идентификатор элемента и данные о текущих значениях его параметров. Язык СИМДИС обладает средствами графического вывода данных машинных экспериментов.

Язык ASPOL [7], [8] относится к языкам, ориентированным на взаимодействие процессов. Он был создан специально для моделирования вычисли-

тельных систем и обладает значительными возможностями моделирования широкого класса дискретных систем. В нем предусмотрен широкий диапазон средств моделирования. Описание процессов и синхронизация их выполнения языковыми средствами ASPOL реализованы на основе методов, применявшихся в проектировании операционных систем. Поэтому язык ASPOL хорошо приспособлен для моделирования систем такого рода.

Важной особенностью языка ASPOL является возможность определения множеств объектов и работы с ними, что значительно упрощает моделирование параллельных систем. Наличие в языке средств макрос - писания обеспечивает расширяемость языка. Синтаксис языка ASPOL напоминает языковые конструкции АЛГОЛА.

В языке ASPOL исследуемая система представляется совокупностью взаимодействующих процессов. Модель системы конструируется из набора описаний процессов. Описание процесса в модели представлено в форме процедуры, которая может выполняться одновременно для всех существующих в системе процессов описываемого класса. В действительности это описание выполняется в квазипараллельном режиме с передачей управления в различные точки процедуры. Таким образом, в модели каждый процесс реализуется отдельным случаем выполнения определенной процедуры, называемой описанием процесса.

В состав модели, написанной на языке ASPOL. входят основная программа, описание процессов и процедуры.

Структура моделирования на основе этого языка основана на событиях следования или окончания активностей. Поэтому не требуется явно определять события следования, так как правильное следование операторов, находящихся в различных процедурах, обеспечивается самой системой имитации.

Кроме основных конструктивных элементов языка ASPOL - процессов и событий в его основе существуют объекты, называемые средствами и памятями, которые бывают необходимы для построения моделей малого и среднего размеров. Язык ASPOL обладает рядом средств, позволяющих собирать, печатать в форме отчетного документа и чертить в виде диаграмм данные о поведении моделируемой системы. В языке предусмотрены также развитые средства отладки программ, включая контроль ошибок.

До середины 70-х годов дискретные и непрерывные языки развивались независимо друг от друга. Однако в дальнейшем наметилась тенденция к интеграции дискретного и непрерывного подходов к рассмотрению моделируемых систем, в результате чего появились комбинированные языки, предназначенные для исследования непрерывно-дискретных систем. Непрерывно-дискретная система, обобщающая дискретную и непрерывную системы, представляет собой изменяющуюся во времени конечную совокупность элементов, взаимодействия между которыми имеют как дискретный, так и непрерывный характеры. В отличии от дискретных систем, состояние которых может быть определено только в моменты свершения событий, в непрерывно-дискретных системах состояние может быть определено и в

промежутках между событиями. Большинство комбинированных языков появилось в результате дальнейшего развития уже существующих дискретных языков.

Одним из первых полностью документированных комбинированных языков является GASP-4.

Язык НЕДИС [7] объединяет подход планирования событий и взаимодействия процессов и предназначен для моделирования широкого класса непрерывно-дискретных систем. Элементы моделируемой системы в НЕДИС описываются в терминах объектов из наборов их атрибутов. Каждый объект активно выполняет некоторые функции. т.е. является процессом. Процесс, имеющий конечную протяженность во времени, состоит из последовательных мгновенных событий. Также как и в GASP-4, в НЕДИС новое событие может планироваться предыдущим событием, либо указанием длительности между ними, либо условием окончания этого интервала. В интервалах между событиями значения атрибутов объектов могут изменяться в соответствии с действующей на данный момент системе дифференциальных уравнении.

При написании модели на языке НЕДИС исследователь определяет классы объектов-процессов, одновременно задавая и атрибуты объектов данного класса, и алгоритм их функционирования. Управляющая программа языка осуществляет ведение календаря событий, продвижение имитационного времени и назначение текущего процесса.

Для решения дифференциальных уравнений в языке НЕДИС имеются два алгоритма: быстрый алгоритм Эйлера - Коши с постоянным шагом, используемый в период отладки модели, и более точный алгоритм Рунге - Кутта четвертого порядка.

В отличии от GASP-4 реализация языка НЕДИС является более сложной задачей, предусматривающей разработку специального компилятора и библиотеки программ.

Возможности языка НЕДИС во многом тождественны возможностям языка СИМУЛА-67, что делает его достаточно универсальным языком программирования, пригодным не только для моделирования, но и для использования в других областях нечисловой обработки информации.

В связи с расширяющимся использованием на практике имитационных моделей и методов языки имитационного программирования постоянно развиваются и совершенствуются. Так, на базе языка GASP-4 разработаны его версии УСМ. GASP-4/E и GASP-5.

Универсальная система моделирования для дискретной и непрерывной имитации (УСМ) [7] является программным обеспечением оптимизационно - имтационного подхода к синтезу структур сложных систем. В УСМ язык GASP-4 дополнен средствами генерации и выбора оптимальных вариантов структуры. Так как все возможности GASP-4 в УСМ сохранены, то она также используется для решения различных задач анализа функционирования сложных систем.

В языке GASP-4/E по сравнению с исходной версией добавлены возможности графопостроения в режиме реального времени, ряд интеграционных алгоритмов и логических функций. Существенным преимуществом этого языка являются средства интерактивного моделирования. Язык GASP-5 является развитием GASP-4 в части непрерывного моделирования.

В последнее время наметилась тенденция создания новых языков имитационного моделирования на основе уже известных языков. Это позволяет создать язык, обладающий всеми преимуществами тех языков, на базе которых он был разработан.

Язык SLAM [3], [7] является примером языка, созданного на основе интеграции других языков. В основу его положены языки Q-GERT [7] и GASP-4, возможности которых сохранены и значительно расширены. Язык SLAM предоставляет исследователю возможность программировать непрерывные, дискретные и комбинированные модели, основанные на различных моделирующих подходах. Каждый из этих подходов предполагает определенную структуру понятий. Такое разнообразие подходов сохраняется потому, что каждый из них имеет свои преимущества. Например, в дискретном имитационном моделировании подход, ориентированный на взаимодействия процессов, легок в изучении, но не обладает достаточной гибкостью. В то же время подход, ориентированный на планирование событий более сложен, но обеспечивает гибкую систему моделирования.

В языке SLAM альтернативные подходы комбинируются. Так, дискретно изменяющаяся система может быть смоделирована в рамках событийного подхода, процессного подхода или их обоих. Непрерывно изменяющиеся системы при использовании языка SLAM могут описываться дифференциальными и разностными уравнениями. Комбинированные дискретно-непрерывные системы моделируются путем интеграции событийного или процессного подходов с непрерывным подходом. Кроме того, язык SLAM включает ряд черт, которые соответствуют подходу сканирования активностей. Процессный подход в этом языке базируется на сетевом представлении моделируемой системы.

Важным преимуществом языка SLAM является то, что альтернативные моделирующие подходы могут быть скомбинированы в рамках одной и той же имитационной модели. Существует шесть возможных взаимодействий между сетевым, дискретно-событийным и непрерывными подходами, которые могут быть реализованы на этом языке:

- компоненты в сетевой модели могут инициироваться возникновением события;
- события могут отменить поток компонентов в сетевой модели;
- компоненты в сетевой модели могут вызвать мгновенные изменения значений переменных;
- переменные состояния, достигая определенных значений, могут инициировать компоненты в сетевой модели;

- события могут вызвать мгновенные изменения значений переменных;
- переменные состояния, достигая определенных значений, могут инициировать события.

Язык SLAM обладает улучшенными по сравнению с Q-GERT и GASP-4 средствами сбора и обработки статистических данных, ввода и вывода информации.

Язык GGC [7] является другим примером интегрированных языков имитационного программирования и создан на основе языков GPSS и GASP-4. Он объединяет легкость изучения языка GPSS с гибкостью использования комбинированного языка GASP-4. Язык GGC сохраняет возможности этих языков и позволяет создавать имитационные модели, базирующиеся на подходе планирования событий, подходе взаимодействия процессов или их комбинации. Моделирующий алгоритм языка GGC обеспечивает возможность разработки имитационных моделей на основе: только языка GPSS; только языка GASP-4 или обоих языков одновременно.

Совершенствование языков имитационного моделирования идет не только по пути расширения класса моделирующих систем, но и в области разработки средств имитационного моделирования. Выполнение имитационных экспериментов на ЭВМ приводит к разрыву во взаимосвязи “исследователь-модель”. Исследователь не имеет возможности наблюдать функционирование модели во время эксперимента, останавливать процесс вычисления и изменять некоторые параметры.

Интерактивные языки обеспечивают исследователя средствами для: ввода и модификации параметров; выбора алгоритмов интегрирования до и между прогонами в хадже эксперимента; графического отображениями и распечатки результатов с дисплея результатов эксперимента и промежуточных данных; хранения, восстановления и комбинирования имитационных программ.

Глава 2

Моделирование на языке Симула - 67

Язык СИМУЛА ориентирован на взаимодействие процессов и представляет собой расширение универсального языка АЛГОЛ в части средств описания и анализа сложных систем. Минимальное число основных понятий, используемых в языке, обеспечивает ему большую гибкость.

В основе описания моделируемой системы лежит представление о ней как совокупности процессов. Процессом в языке СИМУЛА называется объект, работа которого привязана к системному времени, т.е. последовательности активных фаз такого объекта соответствует последовательность моментов системного времени. Исполнение активной фазы процесса называется событием. События отображают изменения состояния моделируемой системы. Каждому событию при работе модели соответствует определенный момент системного времени, при чем в ходе события системное

время не меняется. Нескольким событиям модели может соответствовать один и тот же момент системного времени, что позволяет отображать одновременные события, происходящие в моделируемой системе. Работа процессов и их взаимодействие отражают функционирование системы. Процесс может находиться в одном из четырех состояний: активном, приостановленном, пассивном, законченном.

Активным в каждый момент времени является только один процесс, на который указывает первое уведомление в управляющем списке (календаре событий). Этот процесс в данный момент непосредственно исполняет операторы, описывающие его правила действии.

Для приостановленных процессов характерно наличие ссылающихся на них уведомлений в управляющем списке. Это означает, что для каждого из них запланировано событие и оно будет исполнено, т.е. рано или поздно приостановленный станет активным.

Пассивные процессы, в отличие от приостановленных, не представлены уведомлениями в управляющем списке, т.е. события для них не запланированы. Пассивный процесс может быть переведен в активное или приостановленное состояние. Завершенный процесс также как и пассивный, не представлен в управляющем списке, но не может перейти ни в какое другое состояние, т.е. его работа не может быть возобновлена и он присутствует в модели как структура данных.

Количество процессов и их взаимосвязи могут меняться в ходе работы модели, что позволяет легко описывать динамические системы с переменной структурен.

Важной особенностью языка являются средства структурированного, иерархического описания классов процессов. Эти средства позволяют описывать типовые модели и дают возможность постепенно наращивать, детализировать описания типовых моделей. Исследователь разрабатывает программы для каждого класса процессов, к которому относятся все процессы с одинаковой структурой данных и операционным алгоритмом.

Управление ходом имитации осуществляется с помощью "уведомлений о событии", в которых указывается, в какой момент имитационного времени наступит событие и к какому процессу оно относится. Уведомления помещаются в специальный последовательный управляющий список. Уведомления представляют собой объекты, содержащие информацию о запланированном времени события и ссылку на процесс, очередная активная фаза которого должна исполняться в это время. Во время исполнения своей очередной активной фазы процесс может запланировать события для других процессов, а также следующую активную фазу для себя самого. Кроме того, процесс может отменять некоторые из ранее запланированных, но еще не исполненных событий.

Во время активной фазы процесс может "присоединять" другие процессы, получая доступ к их структурам данных для изменения значений характеристик, а также планировать себе или другому процессу с помощью специальных операторов следующую активную фазу.

Для облегчения программирования типовых приемов статистического анализа в языке имеются процедуры, позволяющие накапливать гистограммы значений случайных величин и вести в ходе работы модели интегрирование по системному времени. Выдача данных экспериментов осуществляется с помощью операторов генерации отчетов.

К сказанному добавим, что в дополнении к средствам Алгола-60 язык содержит удобный аппарат описания новых понятий, средства обработки текстовой информации, стандартные средства ввода-вывода, средства для организации квазипаралельного исполнения компонентов программы. Важной также является возможность построения на его основе и его средствами специализированных языков программирования и пакетов прикладных программ, содержащих основные понятия, выработанные в некоторой предметной области. Язык разработан в Норвежском вычислительном центре и наиболее широко используется для решения задач имитационного моделирования.

2.1. Понятие объекта в языке Симула-67

2.1.1 Объекты и классы объектов

В языке Симула - 67 (точнее, в Симула - 1, 1960 год) впервые было введено такое понятие как объект и поэтому его справедливо можно считать первой системой объектно - ориентированного программирования. Объект - это программный компонент, обладающий атрибутами и способный выполнять определенные действия. Атрибутами могут быть переменные, массивы, процедуры, ссылки на другие объекты. Действия, выполняемые объектом, задаются последовательностью операторов, называемой сценарием его функционирования, поведения или правилами действий. Симула-объекты являются удобным средством для отображения существующих или проектируемых объектов реальности, а также абстрактных понятий. Каждый объект обладает системным атрибутом, указывающим на исполняемый оператор его правил действий. Этот атрибут называется локальным управлением (ЛУ).

В Симула-программе могут одновременно существовать несколько объектов, находящихся на разных стадиях исполнения своих сценариев поведения. Среди них только один объект в каждый момент времени выполняет операторы правил действий. Этот объект называется активным или текущим. ЛУ этого объекта совпадает с центральным управлением и двигается по операторам сценария его поведения. ЛУ любого другого объекта при этом остается неподвижным и указывает на оператор с которого начнется работа этого объекта, когда он станет текущим. Понятно, что операторы правил действий объекта могут быть использованы за несколько активных фаз, в промежутках между которыми работают другие объекты. Такой способ взаимодействия объектов получил название квазипаралельного исполнения, а сами объекты, в этом случае называются сопрограммами.

Объекты могут принадлежать к одному или нескольким множествам родственных объектов, называемых классами. Объекты первого класса имеют общую структуру атрибутов и одинаковые правила действий, но могут различаться конкретными значениями атрибутов. Классы объектов определяются в Симула-программах описаниями, которые называются декларациями классов. Синтаксически декларация класса очень похожа на описание процедуры.

Рассмотрим строение декларации класса на примере описания автобусов, различающихся своими параметрами :

```
                | class АВТОБУС    (P,L,B,H,N,T,S);
(1)             |        имя класса    формальные параметры
                | real P , L , B , H , T , S ; integer N ; - спецификации
                параметров

(2)             | begin real ПУТЬ;
                |       procedure ТЕХОСМОТР;…;
                |       procedure СТОП (V);real V;...;
                |. . . . . . . . . . . . . . . . . . . . . . . . . . . . .|
                | if ПУТЬ >=10000 then ТЕХОСМОТР;.    | (3)
                |. . . . . . . . . . . . . . . . . . . . . . . . . . . . .|
                | if КРАСНЫЙ then СТОП (60);          |
                |. . . . . . . . . . . . . . . . . . . . . . . . . . . . .|
                | end ( декларации класса автобус ) .
```

Здесь (1) - заголовок декларации класса, в котором формальные параметры P,..,S трактуются как вес, длина, ширина, высота, пассажировместимость рейсового автобуса, срок его эксплуатации, длина маршрута соответственно, (2) - тело декларации класса. (3) - описание сценария поведения автобуса. В роли формальных параметров деклараций классов могут использоваться переменные и массивы.

Телом декларации класса может быть любой оператор языка, чаще всего блок. Все переменные, массивы, процедуры, классы, описанные в декларации класса, считаются атрибутами объектов определяемого класса.

2.1.2 Создание объектов и их наименование

Опоздание конкретного объекта, принадлежащего к определенной декларации класса и имеющего определенные значения атрибутов, выполняется с помощью генератора объектов имеющего вид: new <имя класса> (<совокупность фактических параметров>) или new <имя класса>, для декларации класса без формальных параметров.

Результатом вычисления генератора объектов является ссылка на новый объект, указанного в нем класса с конкретными значениями атрибутов. Эта ссылка может быть присвоена переменной типа real - "

ссылка на объект ", которая в дальнейшем служит для обращения к сгенрированному объекту и его атрибутам.

Описания переменных, массивов, процедур - функций, а так же спецификации параметров, значениями которых могут быть ссылки на объекты, начинается с указателя типа, который имеет вид: ref (< имя класса >) Конструкция (< имя класса >) называется квалификацией определяемых переменных. Роль квалификации состоит в ограничении области возможных значений ссылочных переменных. Переменная может ссылаться только на область классов, указанных в ее квалификации.

Присваивание ссылочной переменной ссылки на конкретный объект производится с помощью оператора присваивания ссылок, в левой части которого пишется переменная, а в правой - генератор объекта. Части разделяются между собой знаком " : - " Пример:

АВТ1:-new АВТОБУС(8,10,3,3,120,5,45);
ИКАРУС :- АВТ1;

В Симула - 67 объекту можно дать несколько имен, присвоив ссылку на этот объект нескольким переменным.

Созданный объект существует в программе до тех пор пока на него имеется хотя бы одна ссылка. Уничтожить ее можно путем присваивания ссылочной переменной значения "none". Кроме того, хранящаяся в ссылочной переменной, ссылка на объект уничтожается при выходе из блока, в котором описана эта переменная и при присваивании ей нового значения.

Между ссылками объекта определены два отношения: идентичность == и неидентичность =/=. Отношение x == y истинно в том случае, когда ссылочные переменные x, y указывают на один и тот же объект или одновременно равны none. Отношение x =/= y является отрицанием предыдущего отношения.

Иногда, при описании правил действий объекта, возникает необходимость сослаться на токующий объект, т.е. на себя самого. Это можно сделать с помощью конструкции локального объекта. Она имеет вид: this A. где A - имя класса. Значением этой конструкции, которая может употребляться только в теле декларации класса A и декларациях его подклассов, является ссылка на объект A, исполняющий в данный момент свои правила действии.

Приведем ряд примеров иллюстрирующих введенные языковые конструкции: 1. Пример декларации класса.

```
class ПЕШКА ( НВ , НГ , БЕЛАЯ ) ;
integer НВ,НГ;      boolean БЕЛАЯ;
begin integer ТВ,ТГ; ТВ:=НВ; ТГ:=НГ;
if СВОБОДНА (ТВ,ТГ) then goto ОСТАНОВКА;

          | ТГ:=ТГ + (if БЕЛАЯ then 1 else -1 );
(*)       | if (ТГ>8 or ТГ<1) or not СВОБОДНА(ТВ,ТГ) then
```

```
    | go to ВСТАТЬ НЕКУДА;

ОСТАНОВКА: ЗАНЯТЬ (ТВ,ТГ); goto КОНЕЦ;
ВСТАТЬ НЕКУДА: ТВ:=ТГ:=0; КОНЕЦ;
end ПЕШКА;
```

Здесь описано поведение пешки, которая пытается встать на произвольную клетку шахматной доски: если клетка занята, то пешка пытается встать на клетку доступную из начальной за один ход; если же и она оказывается занятой, то пешка на доску не встает. Клетка задается парой чисел - номерами вертикали и горизонтали шахматной доски. Атрибутами объекта класса ПЕШКА являются: НВ, ТВ - начальная и текущая вертикали. НГ, ТГ - начальная и текущая горизонтали. БЕЛАЯ - булевский атрибут задающий цвет пешки.

В данной декларации класса использованы операторы процедуры функции СВОБОДНА и процедуры ЗАНЯТЬ, описания которых не приводим.

2. Генератор объектов new ПЕШКА(3,2,true);
отображает постановку белой пешки на клетку (3,2).

3. Обозначить через П1 крайнюю левую белую пешку можно задав описание ref(ПЕШКА) П1;
и выполнив оператор присваивания ссылок:
П1 :- new ПЕШКА (l,2,true); .

4. Начальная расстановка всех черных пешек может быть выполнена с помощью фрагмента программы:

```
ref (ПЕШКА) array ЧЕРНЫЕ ПЕШКИ; integer i;
for i :- 1 step 1 until 8 do
ЧЕРНЫЕ ПЕШКИ[i] :- new ПЕШКА (i,7,false);
```

2.1.3 Иерархическое описание классов объектов

Рассмотрим, имеющиеся в Симула - 67 средства отображения иерархических понятий на примере описания шахматных фигур, пытающихся встать на одну из свободных клеток, доступных за один ход из заданной начальной клетки. Решение этой задачи даст описание ряда не связанных между собой деклараций классов, построенных аналогично декларации класса ПЕШКА. каждая из которых полностью описывает поведение соответсвующей фигуры и отличается от нее только наименованием класса и операторами (*) - правилами хода фигуры. Общие части деклараций классов, описывающих различные фигуры. можно выделить в отдельную декларацию, задающую класс ФИГУРА. которая имеет атрибуты и правила действий общие для всех фигур:

```
class ФИГУРА (НВ, НГ, БЕЛАЯ);
integer НВ,НГ; boolean БЕЛАЯ;
begin integer ТВ,ТГ; ТВ:=НВ; ТГ:=НГ;
```

```
if СВОБОДНА (ТВ,ТГ) then goto
ОСТАНОВКА; inner ;
ОСТАНОВКА:ЗАНЯТЬ(ТВ,ТГ); goto
КОНЕЦ; ВСТАТЬ НЕКУДА : =ТВ :
=ТГ : =0 ; КОНЕЦ : end ФИГУРА;
```

В приведенной декларации символ inner - представляет дополнительные сведения - действия, задаваемые в декларациях классов, описывающих поведение любых конкретных фигур.

Теперь эти декларации можно определить на основе класса ФИГУРА и задавать в них только дополнительные, характерные для данного вида фигур признаки (атрибуты и правила действий). Определенные таким образом классы называются подклассами класса ФИГУРА и образуют вместе с ним иерархию классов. Класс ФИГУРА называют префикс-классом (надклассом) для классов ПЕШКА, ЛАДЬЯ и т.д. Вся иерархия классов, описывающих шахматные фигуры задается набором деклараций:

(1) class ФИГУРА…;
(2) ФИГУРА class ПЕШКА…;
ФИГУРА class КОРОЛЬ...;

В симула-программе тот факт, что класс ПЕШКА является подклассом класса ФИГУРА отображается употреблением имени ФИГУРА в качестве префикса к декларации класса ПЕШКА, которая теперь примет вид:

```
ФИГУРА class ПЕШКА;
    begin
    ТГ:=ТГ + ( if БЕЛАЯ then I else -1 );
    if (ТГ>8 or ТГ<1) or not СВОБОДНА ( ТВ , ТГ )
    then go to ВСТАТЬ НЕКУДА;
    end ПЕШКА;
```

Работа объектов принадлежащих классу определенной декларации с префиксом происходит в соответствии с эквивалентной декларацией без префикса, полученной путем объединения атрибутов и правил действий заданных в декларации класса и подкласса. Этот процесс объединения называется сочленением и выполняется по определенным правилам называемым алгоритмом сочленения. В процессе сочленения список формальных параметров декларации класса дополняется. При объединении inner заменяется (оператором правил действий из декларации подкласса. Замена символа происходит не буквально, а организацией передач управления.

Введем некоторые термины, связанные с иерархиями деклараций классов. Префиксальным уровнем декларации класса и объектов, принадлежащих этому классу, называется номер яруса, соответствующем ей вершины в дереве иерархии.

Цепочкой префиксов некоторой декларации класса называется совокупность классов, лежащих на пути из корня дерева иерархии в вершину, соответствующую этой декларации.

Введение аппарата иерархий деклараций классов требует уточнения правил использования квалификационных переменных, генераторов объекта и присваивания ссылок. При создании объекта, принадлежащего классу, декларация которого имеет префикс, в соответствующем генераторе объектов, необходимо указать фактические значения для формальных параметров, присутствующих в эквивалентной декларации. В теле декларации класса можно непосредственно использовать все атрибуты, заданные в декларациях из цепочки префиксов этого класса. Атрибуты подкласса из декларации надкласса непосредственно недоступны, хотя к ним можно обратиться благодаря специальным средствам. Взаимо недоступны и атрибуты деклараций лежащих на разных ветвях в дереве иерархий. В том случае, если в декларации классов принадлежащих первой цепочке префиксов определены одноименные атрибуты, то при употреблении их имен, в некоторой декларации обращение будет происходить к тому атрибуту, который определяет ее в цепочке префиксов. Причем в декларации с верхним префиксальным уровнем над обозначением объектов именами классов в языке определены два отношения. позволяющие определить принадлежность объекта к некоторому классу или его подклассу.

Отношение X if A. где X - объектное выражение. A - имя класса, истинно, если объект тождественно равный X принадлежит классу A.

Отношение X in A истинно если объект, на который указывает выражение X принадлежит классу A или его подклассу.

2.2 Средства доступа к атрибутам объектов

2.2.1 Дистанционные идентификаторы

Дистанционные идентификаторы записываются в виде конструкции: E.X, где E - обозначение объекта, X - имя одного из атрибутов этого объекта. При описании атрибута X объекта E, к которому производится обращение посредством дистанционного идентификатора E.X. важнейшую роль играет квалификация объектного выражения E. Если "E" имеет вид переменной или указателя функции, то его квалификацией считается квалификация, указанная в описании этой переменной. Квалификация генератора объектов или локального объекта является идентификатор класса, записанный после new или this.

Пусть C - квалификация объекта. Тогда E.X обращается к тем атрибутам объекта которые заданы в декларации класса C или его подклассов.

С помощью дистанционного идентификатора можно обращаться к атрибутам, массивам, атрибутам - процедурам.

2.2.2 Присоединяющие операторы.

Еще одним средством доступа к атрибутам объектов служат присоединяющие операторы, которые удобно использовать там, где часто приходится обращаться к атрибутам одного и того же объекта. Наиболее

употребительная форма присоединяющего оператора имеет вид: inspect E do S; где E - объектное выражение, S - оператор, в котором можно обращаться к атрибутам объекта E непосредственно по их именам. Объект E принято называть присоединенным объектом, а S - присоединяющим блоком.

2.2.3 Виртуальные атрибуты

Некоторые из атрибутов - процедур (а так же переключателей и меток), определенных в декларации класса, могут быть объявлены в ней виртуальными. Виртуальные атрибуты при формировании эквивалентной декларации ведут себя по-другому: описание виртуальных атрибутов, расположенное в декларации с префиксальным уровнем K, замещается описанием с таким же идентификатором, но расположенным в декларациях с большим, чем K уровнем (в декларации подклассов). Таким образом объявление атрибута виртуальным, в некоторой декларации класса C позволяет переопределять его в декларациях подклассов и обращаться посредством этого атрибута к другим атрибутам подклассов, которые из C непосредственно не доступны.

2.3 Ввод - вывод в Симула-67

Средства иерархического описания классов объектов открывают очевидную возможность предоставления программисту доступа к любым системным классам введенным в язык. Набор средств ввода-вывода описан системным классом под названием BASICIO и для того, чтобы получить доступ к средствам этого класса, пользователю ничего не надо делать. Программа действует будто она включена в блок:

```
BASICIO (n) begin inspect
SYSINP do inspect SUSOUTP
do
<Программа пользователя>
end .
```

Параметр n - целая константа - указывает максимальную длину строки печатающего устройства и определяется при конкретной реализации языка. Переменные SYSINP и SYSOUTP представляют стандартные устройства ввода - вывода. Эти файлы описаны внутри, то есть вне программы пользователя.

2.4 Средства имитационного моделирования

Совокупность средств имитационного моделирования описана в системном классе SIMULATION, который является подклассом другого системного класса SIMSET позволяющего эффективно обрабатывать упорядоченные множества, организованные в виде циклических двунаправленных списков - наборов входящих в них объектов. Поэтому*

сначала дадим определение набора и опишем детали приведенной ниже структуры декларации класса SIMSET :

```
class simset; begin class linkage;
    begin ref (linkage) SUCC, PREDD;
          ref (link) procedure SUC; . . . .
       . . ;
          ref (link) procedure prod; . . . .
       . ;
    end linkage;
linkage class link;
    begin procedure out; . . . . ;
      procedure  follow(x)  ;  ref  (linkage)
x; . . . . ;
      procedure precede(x) ; ref (linkage)
    x; . . . . ;
      procedure into(s) ;ref (head) s; . . . . . ;
    end link;
linkage class head;
    begin ref (link) procedure first; . . . ;
       ref(link) procedure last; . . . . ;
       boolean procedure empty; . . . ;
       integer procedure cardinal; . . . . . ;
    procedure clear; . . . ;
  SUCC:-PREDD : - this linkage;
    end head;
end simset;
```

Набор - циклическая последовательность элементов из которых все, кроме одного, имеют ссылки на процесс и на своего преемника (SUC) и предшественника (PREDD). Элемент без ссылки на процесс называется головой набора.

Здесь следует пояснить термин "процесс". Процесс является фундаментальным элементом системы программирования Симула - 67. Он служит для организации и классификации действий, происходящих в системе с дискретными событиями синхронизированными по системному времени. Процессом называется объект класса, имеющего process своим префиксом. Класс process имеет префикс link и описан в декларации класса simulation.

Голова набора всегда присутствует в нем, является одной и той же для набора. Если в наборе нет других элементов, он считаются пустым. В непустом наборе преемник головы набора называется первым элементом, а предшественник - последним. Набор имеет фиксированное наименование, которое называется указателем, указатель может быть простым или с индексами.

Объявления наборов синтаксически записывается аналогично объявлению простых переменных или переменных с индексами. Однако

указатель набора не является переменной в том смысле, что набору нельзя делать явных присваиваний. Описания набора недостаточно для его создания. Для этого одним из первых исполняемых операторов в программе должен присутствовать оператор : < указатель набора > : - new head :

Результатом работы этого оператора является порождение головы набора. Первоначально набор пуст. Элемент набора, не являющийся головой называется просто членом набора.

Опишем класс simset: подкласс linkage является общим названием для голов наборов и для членов наборов. Атрибут SUC является ссылкой на преемника данного объекта класса linkage в наборе, а переменная PREDD является ссылкой на предшественника.

Значение этих ссылок можно получить посредством процедур sue и predd. Эти процедуры имеют значение попе, если указанный объект не может быть членом никакого набора. Атрибуты SUC и PREDD можно модифицировать только посредством процедур, определенных внутри классов link и head. Поясним значение процедур описанных в подклассах класса SIMSET:

1. out класса link исключает объект, атрибутом которого она является, из содержащего его набора.

2. процедуры follow(X) и precede(X) включают объект в набор рядом после и перед объектом X.

3. into*) - вставляет объект в конец набора S.

4. процедура first и last дают доступ к крайним элементам набора.

5. empty* - дает значение true, если набор пуст.

6. cardinal - дает число членов набора.

7. clear - исключает из набора все его члены - делает набор пустым.

Приведем фрагмент симула -программы, использующей средства класса simset для имитации постановки двух автомобилей в очередь, отображаемую набором СТОЯНКА.

```
simset begin integer k;
    link class АВТОМОБИЛЬ ( . . . ) ;
        ref(head) СТОЯНКА; ref (АВТОМОБИЛЬ ) АВТI,АВТ2;
        СТОЯНКА :-- new head;
        АВТ1 : - new АВТОМОБИЛЬ( . . . );
        АВТ2 : - new АВТОМОБИЛЬ ( . . . );
        АВТ1.into(СТОЯНКА) ;
        АВТ2 . precede(АВТ1) ;
        ……..
        АВТ1.OUT;
        ……
        k := СТОЯНКА . cardinal ;
        end ( декларации класса автомобиль ) .
```

Средства класса simset могут быть дополнены и даже переопределены программистом. Но имеющиеся средства этого класса вполне достаточны для имитации. Вернемся к классу SIMULATION.

В основе класса SIMULATION, кроме понятия процесс лежит понятие системного времени, которое отображает течение времени в моделируемой системе и представляет собой неубывающую и неотрицательную величину. Основным инструментом синхронизации процессов, применяемым в классе, является управляющий список. Он представляет собой упорядоченный по системному времени набор, членами которого состоят объекты, отображающие запланированные, но еще не исполненные к этому времени события. Эти объекты называются уведомлениями о событиях. Каждое уведомление содержит информацию о запланированном времени события и ссылке на процесс, очередная активная фаза которого должна исполняться в это время. Первое уведомление в управляющем списке указывает на активный в данный момент процесс. Время события, указанное в нем равен токующему системному времени. Во время исполнения своей текущей активной фазы процесс может запланировать события для других процессов, а так же следующую активную фазу для самого себя и, кроме того, отменить ранее запланированные события. Планирование и отмену выполняют специальные операторы, определенные в классе, которые включают в управляющий список уведомления и соответственно исключают их из него.

Описание модели на Симула - 67 оформляется в виде вложенного блока с префиксом SIMULATION : simulation begin .... end. В ходе работы модели процессы могут находиться в активном, приостановленном, пассивном и завершенном состоянии. В активном состоянии в каждый момент времени может находиться только один процесс, он исполняет свои правила действий. По окончании текущего события он может перейти в любое из перечисленных состояний.

В приостановленном состоянии находится процесс, на который есть уведомление в управляющем списке, но оно не первое в этом списке. В пассивном состоянии процесс не имеет уведомлений в управляющем списке, то есть событий для него не запланировано, но его локальное управление указывает на один из операторов его правил действий и активная фаза для него может быть запланирована. Другими словами: пассивный процесс может быть переведен в активное или приостановленное состояние. Непосредственно после генерации процесс находится в пассивном состоянии, его ЛУ* указывает на первый оператор правил действии.

В завершенном состоянии процесс не представлен в управляющем списке и он не может перейти ни в какое другое состояние, то есть он присутствует в модели только как структура данных. Завершенным считается процесс при выходе управления через end его декларации класса. Рассмотрим общую структуру класса simulation. В нем есть два подкласса : EVENT NOTICE (уведомление о событии) PROCESS и ряд процедур управляющих моделированием .

Определен так же набор SQS членами которого являются объекты класса event notice, составляющие управляющий список. Создание головы набора и занесение туда первого уведомления выполняется операторами тела

класса, работа которого предшествовала началу моделирования. Структура декларации класса имеет следующий вид:

```
simset class
simulation; begin
ref(head) SQS:
link class EVENT
NOTICE(EVTIME,PROC); real
EV*TIME:ref(process) PROG;...;
link class process:.....;end:
procedure passivate : . . . :
procedure cancel(X): ref(process)
X;...: procedure hold (T); real
T:.....;
```

SQS: - new head; (другие операторы выполняющие ряд действий по организации моделирования)

ABTI . OUT;

k := СТОЯНКА . cardinal ; end (декларации класса автомобиль) .

Средства класса simset могут быть дополнены и даже переопределены программистом. Но имеющиеся средства этого класса вполне достаточны для имитации. Вернемся к классу SIMULATION.

В основе класса SIMULATION, кроме понятия процесс лежит понятие системного времени, которое отображает течение времени в моделируемой системе и представляет собой неубывающую и неотрицательную величину. Основным инструментом синхронизации процессов, применяемым в классе, является управляющий список. Он представляет собой упорядоченный по системному времени набор, членами которого состоят объекты, отображающие запланированные, но еще не исполненные к этому времени события. Эти объекты называются уведомлениями о событиях. Каждое уведомление содержит информацию о запланированном времени события и ссылке на процесс, очередная активная фаза которого должна исполняться в это время. Первое уведомление в управляющем списке указывает на активный в данный момент процесс. Время события, указанное в нем равен токующему системному времени. Во время исполнения своей текущей активной фазы процесс может запланировать события для других процессов, а так же следующую активную фазу для самого себя и, кроме того, отменить ранее запланированные события. Планирование и отмену выполняют специальные операторы, определенные в классе, которые включают в управляющий список уведомления и соответственно исключают их из него.

Описание модели на Симула - 67 оформляется в виде вложенного блока с префиксом SIMULATION : simulation begin .... end. В ходе работы модели процессы могут находиться в активном, приостановленном, пассивном и завершенном состоянии. В активном состоянии в каждый момент времени может находиться только один процесс, он исполняет свои правила

действий. По окончании текущего события он может перейти в любое из перечисленных состояний.

В приостановленном состоянии находится процесс на который есть уведомление в управляющем списке, но оно не первое в этом списке. В пассивном состоянии процесс не имеет уведомлений в управляющем списке, то есть событий для него не запланированно, но его локальное управление указывает на один из операторов его правил действий и активная фаза для него может быть запланирована. Другими словами: пассивный процесс может быть переведен в активное или приостановленное состояние. Непосредственно после генерации процесс находится в пассивном состоянии, его ЛУ указывает на первый оператор правил действии .

В завершенном состоянии процесс не представлен в упраляющем списке и он не может перейти ни в какое другое состояние, то есть он присутствует в модели только как структура данных. Завершенным считается процесс при выходе управления через end его декларации класса. Рассмотрим общую структуру класса simulation. В нем есть два подкласса : EVENT NOTICE (уведомление о событии) PROCESS и ряд процедур управляющих моделированием .

Определен так же набор SQS членами которого являются объекты класса event notice, составляющие управляющий список. Создание головы набора и занесение туда первого уведомления выполняется операторами тела класса, работа которого предшествовала началу моделирования. Структура декларации класса имеет следующий вид:

```
simset class simulation;
begin ref(head) SQS;
link class EVENT NOTICE (EVTIME,PROC);
real EVTIME; ref(process) PROC;...;
link class process:.......;end;
procedure passivate ;...;
procedure cancel(X); ref(process) X;...;
procedure hold (T); real T;.....;
…….;
SQS: - new head; ( другие операторы выполняющие ряд действий по
организации моделирования )
inner;
end simulation;
```

Декларация класса EVENT NOTICE описывает уведомление о событиях, характеризуемых временем события и ссылкой на процесс. При планировании очередной активной фазы для пассивного процесса X на системное время T программа класса создает уведомление о событии при помощи генератора объектов new EVENT NOTICE (T,X) и включает его в управляющий список, с помощью процедур класса simset. Деклорация класса process задает общие свойства для всех объектов, работа которых связана с системным временем.

Задание конкретных классов процессов выполняется внутри блока с префиксом simulation. Для этого достаточно употребить имя process, в качестве префикса в декларации определяемого класса. Вместе с тем, объекты класса process и его подклассов могут быть членами наборов, т.к. декларация класса process имеет префикс link.

Среди атрибутов процессов заданных в декларации класса process присутствует TERM (имеет значение true для завершенных процессов) и EVENT (указывает на уведомление, ссылающийся на данный процесс), а также ряд процедур позволяющих определить состояние процесса. Например, процедура a terminated – true только для совершенных процессов. Процедура evtime выдает в качестве значения число равное системному времени на которое запланировано для ее процесса (в котором она является атрибутом). Значение текущего системного времени дает процедура – функция time, а процедура – функция carrent дает ссылку на активный процесс, и т.д..

Кроме того, в теле декларации класса simulation определены ряд операторов (процедур), позволяющих управлять процессами.

Одновременно, со средствами отображения параллельно работающих объектов в языке есть и другие средства. Это операторы detach и resume (открепить и возобновить). В Симула-67 выделяют следующие состояния объекта (не процесса): прикреплен, самостоятелен, завершен. В прикрепленном состоянии объект находится с момента начала работы соответствующего генератора объектов до первого выполнения оператора detach в его правилах действий. «Откреплен» и «прикреплен» отображают тот факт, что в процессе вычисления генератор объектов создаваемый объект считается частью объекта, содержащий его генератор, то есть прикреплен к нему. При откреплении его локальное управление останавливается на операторе, следующим за оператором detach с которого начинается исполнение правил действий в активной фазе, то есть объект становится самостоятельным. Возобновление работы самостоятельного объекта осуществляется с помощью оператора resume (X), где X – объект, исполнение правил действий которого нужно возобновить. При выполнении этого оператора в правилах действий некоторого объекта Y, переходит передача центрального управления от объекта Y к X. Локальное управление объекта Y останавливается на операторе, следующим за оператором resume(X) и он перестает быть активным, а объект X начинает выполнять свои правила действий с того оператора, на который указывает его локальное управление, то есть становится активным.

Вся программа является самостоятельным объектом, который получает управление от операционной системы, он называется главной программой.

Далее опишем операторы задержания и отмены процессов (событий). При описании работы объектов моделируемых систем, отображаемых процессами, часто возникает необходимость задержать дальнейшее выполнение этих процессов на некоторый интервал системного времени или вовсе отменить их, если, например, в ходе моделирования сложились условия

делающие невозможным выполнение процессов. Кроме того, задержкой можно имитировать затраты времени на выполнение какого-либо действия моделируемым объектом.

Так, для того, чтобы обеспечить исполнение новой активной фазы некоторого процесса через время T, после окончания его текущей активной фазы достаточно употребить в качестве ее последующего исполняемого оператора: hold(T): T- арифметическое выражение равное требуемому времени задержки.

В результате выполнения этого оператора время события в уведомлении текущего процесса будет увеличено на T, а само оно будет установлено в УС с первого места в позицию соответствующую системному времени time +T, после уведомлений с таким же временем. При этом текущая активная фаза процесса заканчивается, его ЛУ останавливается пере оператором следующим за hold(T), а активным становится процесс, уведомление о котором в результате стало первым в УС. Если же при выполнении оператора hold(T) оказалось, сто в интервале времени (time, time+T) не запланировано ни одного события, то результатом действия этого оператора будет просто изменение текущего времени на величину T.

Следует отметить, что hold(T) не является необходимым и эквивалентен оператору: reaktivate current delay T

Для отменны запланированных активных фаз процессов в классе simulation предусмотрен оператор cansel(X), где X - процесс активная фаза которого должна быть отменена. Этот оператор переводит процесс в разряд пассивных, исключая уведомление о нем из управляющего списка. Если X уже пассивен или завершен, то оператор не производит никаких действий. Если же исключается уведомление о приостановленном процессе, то его ЛУ не меняется, тогда как для активного процесса ЛУ, в этом случае, устанавливается на операторе следующем за cansel.

Перевести активный процесс в пассивное состояние можно также с помощью оператора pssivate, который эквивалентен оператору cansel(carent).

Тогда, когда требуется включить процесс в набор S, отображающий, например, очередь, а затем сделать его пассивным удобно пользоваться оператором wait, который эквивалентен составному оператору begin carrent.into(S); passivate end В заключение несколько слов о планирующих операторах. Синтаксис планирующих операторов определяется следующим образом

activate	X		at T	[prior]
reactivte			delay T	
			after Y	
			before Y	
			< пусто > ,	

где X - объектное выражение, обозначающее процесс, для которого планируется событие ; T - арифметическое выражение, значение которого трактуется как абсолютное системное время (в случае at T) или как время задержки планируемого события относительно текущего момента

системного времени (в случае delay T): Y объектное выражение, обозначающее активный или приостановленный процесс; prior
- указатель приоритета, необязательный параметр: вертикальные линии - квазискобки указывают на альтернативу для конструкций помещенных в строках.

Оператор reaktivate планирует события только для активных и приостановленных процессов, тогда как activate - только для пассивных. И тот и другой вставляют уведомления в управляющий список.

2.5 0 средствах для сбора статистики и стохастического моделирования

Средства стохастического моделирования представим здесь только операторами uniform и randind, которые работают используя процедуру главной выборки, позволяющей получать последовательность псевдо-случайных чисел равномерно распределенных на интервале [0,1]:

```
real procedure psrand(V);integer V;
begin integer R: R:=V*5**(2*p+l);
V:=R-(R-2**N)*2**N;
psrand : =V/2**N;
end ;
```

p, N целые константы, значения которых зависят от максимального целого числа, представляемого на той ЭВМ, где используется симула - программа.

Понятно, что при многократных обращениях к этой процедуре ее значения составят последовательность чисел AI,A2,...,AN,... которая однозначно определится начальным значением VO переменной V. В теории вероятностей показывается, что тогда, когда *O > O целое нечетное число, то такими же будут и все следующие значения этой переменной VI,V2,...,VN,..., получаемые в результате побочного эффекта процедуры главной выборки. Вместе с тем последовательность VI,V2,...,VN,.. будет периодической с периодом 2N-2 И хотя, последовательность чисел AI,A2,...,AN,... не является случайной, она служит хорошим приближением к последовательности случайных чисел равномерно распределенных на интервале [0,1]. Именно поэтому, получаемые таким образом числа и называют псевдослучайными.

В Симула - 67 имеются встроенные процедуры случайного выбора, которые позволяют получать псевдослучайные числа, распределенные по различным законам распределения. Почти все они работают по схеме:

- производится обращение к процедуре, которая выдает псевдослучайное число из интервала [0,1) и изменяет значение своего параметра.
- вычисляется некоторое функциональное преобразование числа, полученного на предыдущем шаге с целью получения заданного закона распределения.

Средства для сбора статистики в языке Симула - 67 представлены только двумя процедурами, позволяющими накапливать гистограммы значении

случайных величин и вести интегрирование по системному времени. Это процедуры histo и accum. описывать которые здесь не имеет смысла. Отметим далее, что на языке Симула 67 различными авторами написаны программы, оформленные в виде классов и позволяющие строить графики на АЦПУ (класс SIMTAPL). эффективно собирать статистические данные (класс SIMSTAT). моделировать непрерывно - дискретные системы (DISCONT).

Более подробную информацию о системе имитационного моделирования Симула - 67 можно получить в книгах: А.Н. Андрианов, С.П. Бычков, А.И. Хорошилов. "Программирование на языке Симула-67" и У. Дал, Б. Мюрхауг, К. Нюгорд "Симула - 67 универсальный язык программирования", которые можно найти почти в каждой технической библиотеке.

2.6 Программа моделирования работы магазина

В заключении этой главы рассмотрим пример Симула - программы моделирующей работу магазина.

2.6.1 Постановка задачи

Пусть требуется промоделировать работу магазина самообслуживания с одной кассой, нагруженного потоком покупателей, входящих в магазин через случайные промежутки времени, равномерно распределенные в интервале от 3 до 7 минут. Время, затраченное покупателем на покупки, и количество покупок будем считать случайными величинами, распределенные равномерно в интервалах [10,20], [4,12] соответственно. Положим, что кассир тратит 30 секунд на расчет за одну покупку и что промоделировать необходимо 8 часов работы магазина.

Прежде, чем описывать всю программу моделирования приведем описание объектов: ПОКУПАТЕЛЬ, КАССИР и ГЕНЕРАТОР ПОКУПАТЕЛЕЙ. Объекты ПОКУПАТЕЛЬ и КАССИР описывают следующие правила поведения покупателя и кассира в магазине. Покупатель за время ТПОК делает КП покупок и оплачивает их у кассира за время, пропорциональное КП. Время оплаты одной покупки - С Покупатель фиксирует в атрибуте ТПЛ время, затраченное на ожидание в очереди и оплату покупок, а кассир подсчитывает суммарное время простоя из-за отсутствия покупателей (атрибут ПРОСТОИ).

```
П О К У П А Т Е Л Ь;
ref( head ) очередь ; ref (кассир) КАССА;
process class ПОКУПАТЕЛЬ(ТПОК,КП);rea1 ТПОК ;
begin real ТВХ,ТПЛ: integer КП;
 hold (ТПОК);into (очередь);
activate КАССА delay О;
 ТВХ := time : passivate;
```

comment покупатель встает в очередь и переходит в пассивное состояние, отметив время входа в очереди и толкнув кассира.

ВЫХОД: ТПЛ := time-ТВХ;

comment покупатель расплатился, вычисляется ТПЛ;

end ПОКУПАТЕЛЬ;

К А С С И Р:

```
process class КАССИР(С); real С;
begin real ТНАЧПР.ПРОСТОЙ; ref (ПОКУПАТЕЛЬ) П;
РАБОТА: for П :- 0ЧЕРЕДЬ.first while П =/= none
do begin hold(П.КП*С); П.out;
aktivate П; end:
```

comment кассир обслужил покупателя, удалил его из очереди и активизировал его, дав ему возможность продолжать свои действия.

ОТДЫХ : ТНАЧПР := time; passivate;

КОНЕЦ ОТДЫХА: ПРОСТОЙ:=ПРОСТОЙ+(time-ТНАУП); goto РАБОТА;

end КАССИР;

Для организации моделирования необходим процесс генерирующий и запускающий через случайные промежутки времени объекты класса ПОКУПАТЕЛЬ:

```
process class ГЕНЕРАТОР ПОКУПАТЕЛЕЙ;
begin СОЗДАНИЕ; activate new ПОКУПАТЕЛЬ
(uniforms (10,20,vl), randint(4,12,v2)); hold (uniform 3.7.v3);
goto СОЗДАНИЕ;
end .
```

2.6.2 Описание программы моделирования

Рассмотрим назначение каждой строки в приведенной ниже программе.

1.Начало самого внешнего блока программы. Описываются глобальные переменные, определяющие условия работы модели.

2. Задание конкретного значения переменной ТМОД. определяющей интервал времени, в течении которого моделируется работа магазина.

3. Задание начальных значений переменных, определяющих последовательности псевдослучайных чисел .

4. Начало блока с префиксом simulation, содержащего описание модели. Описываются ссылочные переменные КАССА и ОЧЕРЕДЬ для обозначения процесса отображающего кассира и набора, соответствующего очереди покупателей.

5-7.Декларации классов КАССИР, ПОКУПАТЕЛЬ, ГЕНЕРАТОР ПОКУПАТЕЛЕЙ. Создание головы набора ОЧЕРЕДЬ.

9. Создание процесса класса КАССИР с временем расчета за одну покупку равным 0.5 минут. Присваивание созданному процессу имени КАССА.

10. Создание генератора покупателей и планирование его первой активной фазы на нулевой момент системного времени вслед за текущей активной фазой главной программы, планирование начала работы кассира .

Данная строка завершает подготовку модели к работе: созданы процессы, отражающие исходное состояние моделируемой системы, и запланировано начало их работы.

11. Задержка дальнейшего исполнения главной программы на время ТМОД. Центральное управление переходит на процесс класса ГЕНЕРАТОР ПОКУПАТЕЛЕЙ, а локальное управление главной программы устанавливается на строку.

12. Теперь работа программы вплоть до момента времени ТМОД будет состоять во взаимодействии процессов, отображающих покупателей, приходящих в магазин, и кассира, выполняющего расчет с покупателями.

12. Вывод результатов моделирования. В результате исполнения приведенных операторов будет напечатана следующая строка (в предположении, что кассир простаивал 45 мин.30 с.). ПРОСТОЙ КАССИРА = 45.30

13. Конец блока с префиксом simulation. При выходе управления, через этот символ end моделирование прекращается независимо от того, есть ли в управляющем списке уведомления о запланированных, но еще не исполненных событиях.

14. Конец симула - программы, возврат управления в операционную систему.

Программа моделирования работы магазина:

```
1 begin real ТМОД ; integer vl,v2,v3;
2 ТМОД := 480; vl := l; v2 := 2; v3 := 3;
3 simulation begin ref (КАССИР) КАССА; ref (head) ОЧЕРЕДЬ;
4 process class ПОКУПАТЕЛЬ(ТПОК,КП) ; . . . . .. . . ;
5 process class КАССИР;.....;
6 process class ГЕНЕРАТОР.....
7 ОЧЕРЕДЬ : = new head:
8 КАССА : = new КАССИР(0.5);
9 activate new ГЕНЕРАТОР ПОКУПАТЕЛЕЙ delay O;
10 activate КАССА delay O; hold (ТМОД);
11 outtext (‘ПРОСТОЙ КАССИРА =’);
12 outtext(КАССА.ПРОСТОЙ,2,10);
13 end;
14 end.
```

Список литературы

1. Марков А.А., Пирумов Н.Р., Гудзенко Д.Ю., Общецелевая система моделирования на Паскале , М., 1988, 72 стр.
2. Шеннон Р., Имитационное моделирование систем - исскуство и наука, М.,"Мир", 1978, 418 стр.
3. Прицкер А., Введение в имтационное моделирование и язык СЛАМ II, М.,"Мир", 1987, 644 стр.
4. Шрайбер Т.Дж., Моделирование на GPSS. М.,"Машиностроение", 1980, 592 стр.
5. Томашевский В.Н., Жданова Е.Г. Имитационное моделирование на GPSS. М. Бестселлер, 2003 г.
6. Варжапетян А.Г. Имитационное моделирование на GPSS/Y: Учебное пособие.-СПб.: ГУАП, 2007.- 384 с.
7. Имитационные системы принятия экономических решений, М.,"Наука", 1979, 463 стр.
8. Автоматизация проектирования вычислительных систем. Языки моделирования и БД, М., 1979, 463 стр.
9. Бродский Ю.И., Лебедев В.Ю., Инструментальная система имитации MISS, М., ВЦ АН СССР, 1991, 179 стр.
10. Модзолевский В.И., Средства имитационного моделирования в моделирующем комплексе АЛСИМ-2, Киев,"Знание", 1981, 48 стр.
11. Мановицкий В.И., Сурков Е.М., Система имитационного моделирования дискретных процессов ДИСМ, Киев-Одесса,"Вища школа", 1981, 95 стр.
12. Андрианов А.Н., Бычков С.П., Хорошилов А.И., Программирование на языке Симула-67, М.,"Наука", 1985, 288 стр.
13. У. Дал, Б. Мюрхауг, К. Нюгорд "Симула - 67 универсальный язык программирования",
14. Григорян К., Моделирование системы реального времени на языке Симула-67, Дубна, ОИЯИ, 1976, 13 стр.
15. Азаров С.С., Шемшур А.В., Моделирование непрерывно--дискретных систем с использованием пакета GASP-IV, Киев, Инс-т кибернетики, 1979, 35 стр.
16. Бутомо И.Д., Дробинцев Д.Ф., Котляров В.П., Методы имитационного моделирования вычислительных систем, Ленинград, ЛПИ, 1979, 72 стр.
17. Гусев В., Языки имитационного моделирования и некоторые тенденции их развития, Киев, Инс-т кибернетики, 1972, 29 стр.
18. Нейлор Т.Х., Имитационные эксперименты с моделями экономических систем, М.,"Мир", 1975.
19. Gershefski G.M., Corporate Models}, The State of the Art, University of Washington, Seattle, Wash., 1970.
20. Meadows D.L., Dynamics of Commodity Production Cycles, Wright-Allen Press, Cambridge, Mass., 1970.
21. Dutton J.M., Starbush W.H., Computer Simulation of Human Behavior, Wiley, Inc., New York, 1971.

22. Armstrong R.H., Taylor J.L., eds., Instructional Simulation System in Higher Education, Cambridge, "Monographs on Teaching Methods", N 2,1970.
23. Kresge D.T., Roberts P.O., Techniques of Transportation Planning: System Analysis and Simulation Models, Brooklings Institution, Washington, D.C., 1971.
24. Siegal A.J., Wolf J.J., Man-machine Simulation Models, Intercience Publishers, New York, 1969.
25. Guetzkow H., Simulation in International Relations: Developments for Reseach and Teaching, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.
26. Cherryholmes C.H., Shapiro M.J., Represantatives and Roll-calls: A Computer Simulation of Voting in the Eighty-eighth Congress, Bobbs-Merrill Cj., Inc., New York, 1969.