МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение высшего профессионального образования
«Нижегородский государственный университет им. Н.И.Лобачевского»

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

Кафедра информатики и автоматизации научных исследований

Н.Н.Чернышова

Имитационное моделирование бизнес - процессов

Учебно – методическое пособие

Рекомендовано методической комиссией факультета вычислительной математики и кибернетики для студентов ННГУ, обучающихся по направлению подготовки 080800 «Прикладная информатика (в информационной сфере)»

Нижний Новгород

2010

УДК 519.8

Н.Н.Чернышова ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ БИЗНЕС - ПРОЦЕСОВ :Учебно – методическое пособие. - Нижний Новгород: Издательство Нижегородского государственного университета, 2010. – 28с.

Рецензент : кандидат физ.-мат. наук, доцент А.В.Баркалов

Материал предназначен для студентов, обучающихся по направлению подготовки 080800 «Прикладная информатика (в информационной сфере)» факультета ВМК ННГУ как пособие при изучении аппарата сетей Петри и языка имитационного моделирования GPSS.

Материал может быть использован при подготовке к практическим занятиям в курсах «Сети Петри», «Моделирование информационных ресурсов», «Моделирование информационных процессов и систем».

УДК 519.8

Нижегородский государственный университет имени Н.И.Лобачевского

2010 г.

Часть 1. Подходы к имитационному моделированию бизнес-процессов

Моделирование — один из способов исследования и устранения проблем, возникающих в окружающем нас мире. Модель является реальным или абстрактным объектом, который заменяет (представляет) объект исследования в процессе его изучения, находится в отношении сходства с последним (аналогия, физическое подобие и т. п.) и более удобен для экспериментов. Наиболее естественная и важная сфера применения моделирования — анализ сложных систем, в том числе социотехнических (производственных, финансовых и т. д.).

Традиционно различают аналитическое и имитационное моделирование.

Аналитическая модель, как правило, статическая (ее выходы функционально зависят от входов) и поэтому в ряде практических случаев может быть реализована даже с помощью электронных таблиц.

К *имитационным* моделям прибегают тогда, когда объект моделирования настолько сложен, что адекватно описать его поведение математическими уравнениями невозможно или затруднительно. Имитационное (динамическое) моделирование рассматривает модель как совокупность правил (дифференциальных уравнений, конечных автоматов, сетей Петри и т.п.), которые определяют, в какое состояние в будущем перейдет моделируемый объект из некоторого предшествующего состояния.

В имитационном моделировании сформировались и наиболее часто применяются три основных подхода — дискретно-событийное моделирование, системная динамика и агентное моделирование [1].

Аппарат системной динамики обычно оперирует непрерывными во времени процессами, а дискретно-событийное и агентное моделирование используются для дискретных во времени процессов. Системная динамика предполагает максимальный уровень абстракции модели, дискретно-событийное моделирование отражает абстракции низкого и среднего уровня. Агентное моделирование может применяться на любом уровне модели любого масштаба.

Дискретно-событийное моделирование

Дискретно-событийное моделирование обязано своим рождением Дж. Гордону, который в начале 1960-х спроектировал и реализовал на мэйнфреймах[1] IBM систему GPSS. Основной объект в этой системе — пассивный транзакт (заявка на обслуживание), который может определенным образом представлять собой работников, детали, сырье, документы, сигналы и т. п. «Перемещаясь» по модели, транзакты становятся в очереди к одноканальным и многоканальным устройствам, захватывают и освобождают эти устройства, расщепляются, уничтожаются и т. д. Таким образом, дискретно-событийную модель можно рассматривать как глобальную схему обслуживания заявок. Аналитические результаты для большого количества частных случаев таких моделей рассматриваются в теории массового обслуживания.

Сегодня существует целый ряд инструментов, поддерживающих такой подход в моделировании: GPSS/PC, GPSS/H, GPSS World, Object GPSS, Arena, SimProcess, Enterprise Dynamics, Auto-Mod и др.

GPSS World — типичный современный представитель GPSS-семейства, реализованный для работы в среде MS Windows. Наличие встроенных инструментов статистической обработки результатов моделирования, встроенного языка программирования расчетов PLUS и др. позволяет создавать средствами GPSS World не только простые обучающие модели, но и более полезные приложения.

Несмотря на изначальную ориентацию GPSS на моделирование систем массового обслуживания, система оказалась удивительно долгоживущей и способной к развитию. Трудоемкость описания моделируемых систем в терминах бизнес-процессов может быть снижена за счет применения таких продуктов, как Object GPSS или ISS 2000 (пакет ISS 2000 представляет собой лингвистический процессор, с помощью которого пользователь в

---

[1] Мэйнфрейм (mainframe) – высокоэффективная вычислительная машина с повышенным размером оперативной памяти, вместительными жёсткими дисками, способная производить множество трудоёмких вычислений одновременно и непрерывно в течении продолжительного времени. Основная сфера использования мэйнфреймов – госкорпорации, крупные коммерческие организации, научные исследования. С начала роста популярности и миниатюризации персональных компьютеров отношение к громоздким, хотя и производительным, мэйнфреймам становилось весьма скептическим. В начале девяностых годов прошлого века даже высказывались доводы и о скорой кончине этого типа устройств. Однако, не смотря ни на что, рынок мэйнфреймов жив и успешен по сей день. Также научные исследования показывают, что при использовании глобальных информационных массивов, обработка данных будет производиться значительно легче и экономически выгодней с помощью мэйнфрейма, чем при участии сети персональных устройств.
Мэйнфрейм обходит обычные современные компьютеры практически по всем показателям. Отдельно стоит уделить внимание высокой надёжности самого устройства и данных, с которыми оно работает. Наличие резервных составляющих устройств системы и возможность их горячего замена обеспечивают непрерывность работы. А стандартная величина загруженности процессора без особых усилий переваливает за отметку в 85 процентов. Управление таким устройством происходит с помощью цепочки терминалов, а с недавних пор и через сетевой интерфейс. Лидирующие позиции в производстве мэйнфреймов занимает компания IBM.

диалоговом режиме создает автоматически GPSS-программу и запускает ее на выполнение).

Системная динамика

Системная динамика как методология была предложена в 1961 году Дж. Форрестером в качестве инструмента исследования информационных обратных связей в производственно-хозяйственной деятельности, для того чтобы выяснить, каким образом взаимодействуют организационная структура, усиления (в политиках) и задержки (в принятии решений и действиях), оказывая влияние на эффективность предприятия. Процессы, происходящие в реальном мире, в системной динамике представляются в терминах накопителей (фондов) и потоков между ними. Системно-динамическая модель описывает поведение системы и ее структуру как множество взаимодействующих обратных положительных и отрицательных связей и задержек. Математически такая модель выглядит как система дифференциальных уравнений.

Методы системной динамики поддерживаются такими инструментами, как DYNAMO, Stella, Vensim, PowerSim, iThink, ModelMaker и др.

Пакет *Vensim* представляет собой инструмент для визуального моделирования, поддерживающий разработку концептуальной модели, документирование, собственно моделирование, анализ результатов и оптимизацию моделей динамических систем. Он позиционируется на рынке программных продуктов как простое и гибкое средство для построения имитационных моделей систем с причинно-следственными связями, фондами и потоками. Следует отметить, что Vensim существует и в версии для академического использования в образовательных целях. Пакет имеет графический редактор для построения с помощью мыши классических форрестеровских моделей, Equation Editor для завершения формирования модели, а также развитые средства визуализации поведения модели.

Программные комплексы Stella и iThink предназначены для преобразования моделей принятия решений в имитационные модели. Основной упор делается на формирование у пользователя умения принимать решения, необходимые для исследования систем со сложными взаимозависимыми связями между подсистемами. Указанные программы широко используют графические функциональные элементы для графического изображения потоков, фондов, эффектов влияния неформализованных факторов. Динамика процессов и объектов выражается с помощью пяти типов базовых параметров:

увеличение фондов, исчерпание фондов, рабочий процесс, соединение потоков, адаптация фондов. Соответственно, модели представляются тремя иерархическими уровнями: блок-схемы, базовые потоковые схемы, формальные спецификации.

Одна из наиболее показательных сфер применения аппарата системной динамики — имитационное моделирование финансово-кредитной деятельности. Так, существует ряд моделей банковских и страховых учреждений, выполненных с помощью PowerSim и iThink, обеспечивающих расчет показателей текущего и будущих периодов, прогнозы состояния отдельных сделок и состояния финансового учреждения в целом, оценку привлекательности направлений инвестиционной деятельности, оценку эффективности кредитного и депозитного портфелей банка и т. п. Накоплен положительный опыт оптимизации структуры холдингов с помощью имитационного моделирования в среде iThink.

Агентное моделирование

Агентное моделирование предполагает работу с децентрализованной моделью. В такой модели нет единой точки, определяющей поведение системы в целом. Агентная модель состоит из множества индивидуальных объектов (*агентов*) и их окружения. Поведение системы описывается на индивидуальном уровне; глобальное поведение рассматривается как результат совокупной деятельности агентов, каждый из которых действует сообразно собственному «уставу», существует в общей среде, взаимодействует со средой и другими агентами. Для описания поведения агентов используются *карты состояний*, являющиеся стандартным инструментом UML.

Для систем, содержащих большое количество активных объектов с отчетливо выраженным индивидуальным поведением, агентное моделирование является более универсальным подходом, т. к. позволяет учесть структуру и поведение любой сложности.

Другое важное достоинство агентного моделирования — возможность разработки модели даже в отсутствие априорной информации о глобальных зависимостях. Зная индивидуальную логику поведения участников процесса, можно построить агентную модель и спрогнозировать ее глобальное поведение. Помимо этого, агентная модель проще в сопровождении, поскольку уточнения вносятся на локальном уровне по мере накопления данных.

Концепция агентного моделирования позволяет осуществить переход от моделей системной динамики и дискретно-событийных моделей к агентным моделям с помощью

процедуры конвертации. Для системно-динамических моделей может потребоваться деагрегация накопителей на множества агентов (при условии активности и различимости этих агентов).

Другие подходы

Группа зарубежных исследователей, усматривая аналогии между фундаментальными процессами взаимодействия молекул (перераспределение импульсов и энергии) и взаимодействия участников рынка (перераспределение денег и товаров), применили *методы статистической физики* к исследованию колебания цен на фондовом рынке. Более того, сравнительно давно было отмечено, что закономерности процессов обмена в экономических системах сходны с закономерностями равновесных состояний в термодинамике.

Существуют и *узкоспециализированные методологии*, предназначенные исключительно для моделирования и анализа бизнес-процессов, например, ARIS (Architecture of Integrated Information Systems). Организация в ARIS рассматривается с четырех точек зрения: организационной структуры, функциональной структуры, структуры данных, структуры процессов. Для описания бизнес-процессов предлагается около 80 типов моделей, каждая из которых отражает тот или иной аспект моделирования. Развитая репрезентативная графика делает модели в ARIS особенно удобными для представления руководству и принятия стратегических решений.

ARIS хорошо стыкуется с известными ERP-системами[2], в частности, позволяет описать структуру SAP R/3 в терминах управления бизнес-процессами и провести реинжиниринг (импортировав в ARIS текущие описания бизнес-процессов из R/3). Есть возможность проверки создаваемых моделей на соответствие методологии SAP и тестирования проекта на соответствие требованиям стандарта качества ISO 9000.

Как показывает практика, внедрению ARIS должна предшествовать серьезная «безмашинная» проектно-аналитическая подготовка. Обычно ARIS используется либо для

---

[2] **ERP-система** (*Enterprise Resource Planning System* — Система планирования ресурсов предприятия) — корпоративная информационная система (КИС), предназначенная для автоматизации учёта и управления. Как правило, ERP-системы строятся по модульному принципу и охватывают все ключевые процессы деятельности компании.

Исторически концепция ERP стала развитием более простых концепций MRP (Material Requirement Planning — Планирование материальных потребностей) и MRP II (Manufacturing Resource Planning — Планирование производственных ресурсов). Используемый в ERP-системах программный инструментарий позволяет проводить производственное планирование, моделировать поток заказов и оценивать возможность их реализации в службах и подразделениях предприятия, увязывая его со сбытом.

В основе ERP-систем лежит принцип создания единого хранилища данных, содержащего всю корпоративную бизнес-информацию и обеспечивающего одновременный доступ к ней любого необходимого количества сотрудников предприятия, наделённых соответствующими полномочиями. Изменение данных производится через функции (функциональные возможности) системы. ERP-система состоит из следующих элементов:

- модель управления информационными потоками (ИП) на предприятии;
- аппаратно-техническая база и средства коммуникаций;
- СУБД, системное и обеспечивающее ПО;
- набор программных продуктов, автоматизирующих управление ИП;
- регламент использования и развития программных продуктов;
- IT-департамент и обеспечивающие службы;
- собственно пользователи программных продуктов.

Основные функции ERP систем:

- ведение конструкторских и технологических спецификаций, определяющих состав производимых изделий, а также материальные ресурсы и операции, необходимые для их изготовления;
- формирование планов продаж и производства;
- планирование потребностей в материалах и комплектующих, сроков и объёмов поставок для выполнения плана производства продукции;
- управление запасами и закупками: ведение договоров, реализация централизованных закупок, обеспечение учёта и оптимизации складских и цеховых запасов;
- планирование производственных мощностей от укрупнённого планирования до использования отдельных станков и оборудования;
- оперативное управление финансами, включая составление финансового плана и осуществление контроля его исполнения, финансовый и управленческий учёт;
- управления проектами, включая планирование этапов и ресурсов

Применение ERP-системы позволяет использовать одну интегрированную программу вместо нескольких разрозненных. Единая система может управлять обработкой, логистикой, дистрибуцией, запасами, доставкой, выставлением счетов-фактур и бухгалтерским учётом.

Реализуемая в ERP-системах система разграничения доступа к информации предназначена для противодействия как внешним угрозам (например, промышленному шпионажу), так и внутренним (например, хищениям).

формирования бизнес-структуры с самого начала, либо для ее крупномасштабной комплексной перестройки. Методики оптимизации, предлагаемые ARIS, представляют собой только первичный этап оптимизации бизнес-процессов (т. к. стандартные алгоритмы анализа могут быть реализованы и без использования ARIS, а более сложные алгоритмы могут быть воплощены с помощью сервисной надстройки «Поиск решения» в MS Excel).

Успешность применения ARIS сильно зависит от наличия профессиональных бизнес-аналитиков и высокой управленческой культуры на предприятии.

Относительно имитационного моделирования в рамках ARIS отметим, что модуль ARIS Simulation может быть использован для проведения динамических экспериментов в целях определения узких мест в реализации процессов (несогласованность параллельно выполняемых процессов, нехватка ресурсов и т. п.). Для этого предварительно требуется формализовать временные характеристики исследуемых бизнес-процессов. Возможно, более удобным в этом плане является программный комплекс MATLAB/Simulink, специально предназначенный для моделирования динамических систем.

Выводы

Присутствие в экономико-математических моделях материального, финансового и социального факторов требует применения различных инструментов на соответствующем модельном уровне. Так, производственно-технологические модели (традиционно рассматриваемые как системы массового обслуживания) неплохо моделируются дискретно-событийными средствами типа GPSS; финансовые модели хорошо вписываются в рамки системной динамики; для имитационного моделирования трудовых ресурсов может быть полезен агентный подход.

Часть 2. Моделирование с помощью сетей Петри

Сети Петри — аппарат для моделирования динамических дискретных систем (преимущественно асинхронных параллельных процессов) [4,5]. Сеть Петри определяется как четверка $<\mathbf{P},\mathbf{T},\mathbf{I},\mathbf{O}>$, где $\mathbf{P}$ и $\mathbf{T}$ — конечные множества позиций и переходов, $\mathbf{I}$ и $\mathbf{O}$ — множества входных и выходных функций. Другими словами, сеть Петри представляет собой двудольный ориентированный граф, в котором позициям соответствуют вершины, изображаемые кружками, а переходам — вершины,

изображаемые утолщенными черточками; функциям $\mathbf{I}$ соответствуют дуги, направленные от позиций к переходам, а функциям $\mathbf{O}$ — от переходов к позициям.

Как и в системах массового обслуживания, в сетях Петри вводятся объекты двух типов: динамические — изображаются метками (маркерами) внутри позиций и статические — им соответствуют вершины и дуги сети Петри.

Распределение маркеров по позициям называют маркировкой. Маркеры могут перемещаться в сети. Каждое изменение маркировки называют событием, причем каждое событие связано с определенным переходом. Считается, что события происходят мгновенно и разновременно при выполнении некоторых условий.

Каждому условию в сети Петри соответствует определенная позиция. Совершению события соответствует срабатывание (возбуждение или запуск) перехода, при котором маркеры из входных позиций этого перехода перемещаются в выходные позиции. Последовательность событий образует моделируемый процесс.

Правила срабатывания переходов (рис. 1), конкретизируют следующим образом: переход срабатывает, если для каждой из его входных позиций выполняется условие $N_i \geq K_i$, где N_i — число маркеров в i-й входной позиции, K_i — число дуг, идущих от i-й позиции к переходу; при срабатывании перехода число маркеров в i-й входной позиции уменьшается на K_i, а в j-й выходной позиции увеличивается на M_j, где M_j — число дуг, связывающих переход с j-й позицией.

На рис.1 показан пример распределения маркеров по позициям перед срабатыванием, эту маркировку записывают в виде (2,2,3,1). После срабатывания перехода маркировка становится иной: (1,0,1,4).

Можно вводить ряд дополнительных правил и условий в алгоритмы моделирования, получая ту или иную разновидность сетей Петри. Так, прежде всего полезно ввести модельное время, чтобы моделировать не только последовательность событий, но и их привязку ко времени. Это осуществляется приданием переходам веса — продолжительности (задержки) срабатывания, которую можно определять, используя задаваемый при этом алгоритм. Полученную модель называют *временной сетью Петри*.

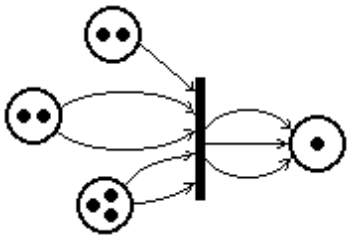

Рис. 1. Фрагмент сети Петри

Если задержки являются случайными величинами, то сеть называют *стохастической сетью Петри*. В стохастических сетях возможно введение вероятностей срабатывания возбужденных переходов. Так, на рис. 2 представлен фрагмент сети Петри, иллюстрирующий конфликтную ситуацию — маркер в позиции p может запустить либо переход t_1, либо переход t_2. В стохастической сети предусматривается вероятностный выбор срабатывающего перехода в таких ситуациях.

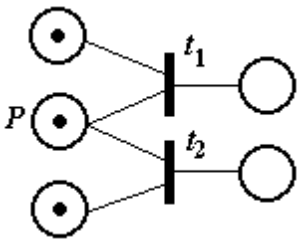

Рис. 2. Конфликтная ситуация

Если задержки определяются как функции некоторых аргументов, которыми могут быть количества маркеров в каких-либо позициях, состояния некоторых переходов и т.п., то имеем *функциональную сеть Петри*.

Во многих задачах динамические объекты могут быть нескольких типов, и для каждого типа нужно вводить свои алгоритмы поведения в сети. В этом случае каждый маркер должен иметь хотя бы один параметр, обозначающий тип маркера. Такой параметр обычно называют цветом; цвет можно использовать как аргумент в функциональных сетях. Сеть при этом называют *цветной сетью Петри*.

Среди других разновидностей сетей Петри следует упомянуть *ингибиторные сети Петри*, характеризующиеся тем, что в них возможны запрещающие (ингибиторные)

дуги. Наличие маркера во входной позиции, связанной с переходом ингибиторной дугой, означает запрещение срабатывания перехода.

Введенные понятия поясним на следующих примерах.

Пример 1

Требуется описать с помощью сети Петри работу группы пользователей на единственной рабочей станции WS при заданных характеристиках потока запросов на пользование WS и характеристиках поступающих задач. Сеть Петри представлена на рис. 3.

Здесь переходы связаны со следующими событиями: t_1 — поступление запроса на использование WS, t_2 — занятие станции, t_3 — освобождение станции, t_4 — выход обслуженной заявки; позиция p_4 используется для отображения состояния WS: если в p_4 имеется метка, то WS свободна и пришедшая заявка вызывает срабатывание перехода t_2; пока эта заявка не будет обслужена, метки в p_4 не будет, следовательно, пришедшие в позицию p_1 запросы вынуждены ожидать срабатывания перехода t_3.

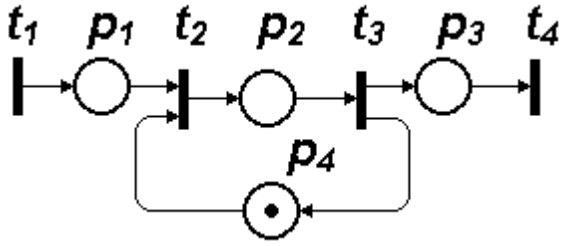

Рис. 3. Сеть Петри для примера 1

Пример 2

Требуется описать с помощью сети Петри процессы возникновения и устранения неисправностей в некоторой технической системе, состоящей из множества однотипных блоков; в запасе имеется один исправный блок; известны статистические данные об интенсивностях возникновения отказов и длительностях таких операций, как поиск неисправностей, замена и ремонт отказавшего блока. Поиск и замену отказавшего блока производит одна бригада, а ремонт замененного блока — другая бригада. Сеть Петри показана на рис. 5. Отметим, что при числе меток в позиции, равном M, можно в ней не ставить M точек, а записать в позиции значение M.

В нашем примере значение M в позиции p_2 соответствует числу имеющихся в системе блоков. Переходы отображают следующие события: t_1 — отказ блока, t_2 — поиск неисправного блока, t_3 — его замена, t_4 — окончание ремонта.

Очевидно, что при непустой позиции p_2 переход t_1 срабатывает, но с задержкой, равной вычисленному случайному значению моделируемого отрезка времени между отказами. После выхода маркера из t_1 он попадает через p_1 в t_2, если имеется метка в позиции p_6, это означает, что обслуживающая систему бригада специалистов свободна и может приступить к поиску возникшей неисправности. В переходе t_2 метка задерживается на время, равное случайному значению длительности поиска неисправности. Далее маркер оказывается в p_3 и, если имеется запасной блок (маркер в p_4), то запускается переход t_3, из которого маркеры выйдут в p_2, p_5 и в p_6 через отрезок времени, требуемый для замены блока. После этого в t_4 имитируется восстановление неисправного блока.

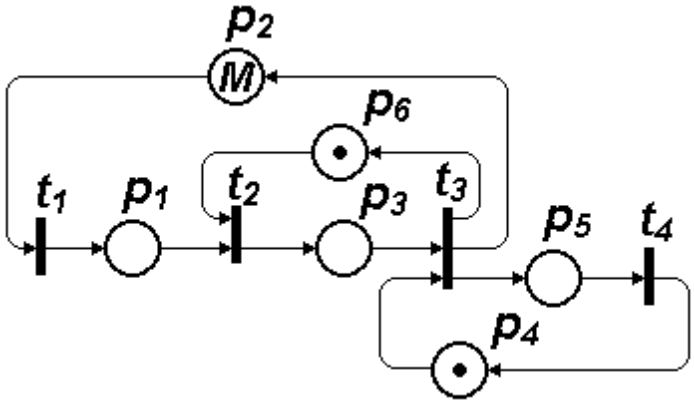

Рис. 4. Сеть Петри для примера 3

Рассматриваемая модель описывает функционирование системы в условиях, когда отказы могут возникать и в рабочем, и в неисправном состояниях системы. Поэтому не исключены ситуации, при которых более чем один маркер окажется в позиции p_1.

Анализ сетей Петри

Анализ сложных систем на базе сетей Петри можно выполнять посредством имитационного моделирования СМО, представленных моделями сетей Петри. При этом задают входные потоки заявок и определяют соответствующую реакцию системы.

Выходные параметры СМО рассчитывают путем обработки накопленного при моделировании статистического материала.

Возможен и другой подход к использованию сетей Петри для анализа объектов, исследуемых на системном уровне. Он не связан с имитацией процессов и основан на исследовании таких свойств сетей Петри, как ограниченность, безопасность, сохраняемость, достижимость, живость [4,5].

Ограниченность (или K-ограниченность) имеет место, если число меток в любой позиции сети не может превысить значения K. При проектировании автоматизированных систем определение K позволяет обоснованно выбирать емкости накопителей. Возможность неограниченного роста числа меток свидетельствует об опасности неограниченного роста длин очередей.

Безопасность — частный случай ограниченности, а именно это 1-ограниченность. Если для некоторой позиции установлено, что она безопасна, то ее можно представлять одним триггером.

Сохраняемость характеризуется постоянством загрузки ресурсов, т.е.

$$\sum A_i N_i = \text{const},$$

где N_i — число маркеров в i-й позиции, A_i — весовой коэффициент.

Достижимость $\mathbf{M}_k \to \mathbf{M}_j$ характеризуется возможностью достижения маркировки $\mathbf{M}_j$ из состояния сети, характеризуемого маркировкой $\mathbf{M}_k$.

Живость сети Петри определяется возможностью срабатывания любого перехода при функционировании моделируемого объекта. Отсутствие живости означает либо избыточность аппаратуры в проектируемой системе, либо свидетельствует о возможности возникновения зацикливаний, тупиков, блокировок.

В основе исследования перечисленных свойств сетей Петри лежит анализ достижимости.

Один из методов анализа достижимости любой маркировки из состояния $\mathbf{M}_0$ — построение *графа достижимости*. Начальная вершина графа отображает $\mathbf{M}_0$, а остальные вершины соответствуют маркировкам. Дуга из $\mathbf{M}_i$ в $\mathbf{M}_j$ означает событие $\mathbf{M}_i \to \mathbf{M}_j$ и соответствует срабатыванию перехода t. В сложных сетях граф может содержать чрезмерно большое число вершин и дуг. Однако при построении графа можно

не отображать все вершины, так как многие из них являются дублями (действительно, от маркировки $\mathbf{M}_k$ всегда порождается один и тот же подграф вне зависимости от того. из какого состояния система пришла в $\mathbf{M}_k$). Тупики обнаруживаются по отсутствию разрешенных переходов из какой-либо вершины, т.е. по наличию листьев — терминальных вершин. Неограниченный рост числа маркеров в какой-либо позиции свидетельствует о нарушениях ограниченности.

Моделирование программно-аппаратных "реактивных" систем раскрашенными сетями Петри

Под "реактивной" подразумевается программно-аппаратная система реального времени, осуществляющая прием внешних входных воздействий, обработку их в соответствии с внутренним алгоритмом и выдачу необходимых выходных реакций. Природа появления входных воздействий – событийная, сильно зависящая от внешних факторов, в то же время, формирование выходных реакций должно быть однозначным, установленным на стадии разработки. Часто при моделировании таких систем применяют теорию конечных автоматов, основанную на понятиях состояния и перехода. Интуитивно состояние можно представить как некое положение внутреннего "механизма" системы, обеспечивающее предопределенные выходные характеристики. Изменение положения этого внутреннего "механизма" достигается за счет внешних воздействий на него через систему специальных преобразователей.

"Реактивными" системами обычно являются программно-аппаратные комплексы, где аппаратная составляющая используется для согласования управляющей (программной) логики с реальной средой (механизмы, датчики и т.п.). Часто для повышения производительности таких систем или в целях получения независимости работы их подсистем применяют параллельное выполнение задач, как на программном, так и на аппаратном уровне.

Отличительными чертами этих систем является:

- Наличие ограниченного числа разделяемых ресурсов между параллельно работающими процессами (устройства ввода-вывода, коммутируемые устройства управления, каналы линий передачи информации и т.д.).

- Необходимость качественной обработки нештатных ситуаций, связанных с отказами аппаратной части или непредсказуемой комбинацией поступающих входных воздействий на систему.
- Акцентирование внимания на ресурсе, связанным со временем обработки поступающих данных или управлением внешними объектами по временной диаграмме.

Для разработчика важнейшими задачами анализа при этом являются:

- Доказательство непротиворечивости системы в плане согласованного использования разделяемых ресурсов и синхронизации работы параллельных процессов.
- Возможность рассмотрения последствий ложных внешних воздействий на систему и пути их устранения.
- Получение временных отсечек работы процессов (в т.ч. параллельных) при решении задач систем реального времени.

Архитектура разрабатываемого проекта представляется в виде разнообразных диаграмм, отражающих как статическую, так и динамическую составляющую системы. Фактически, в этом вопросе унифицированный язык моделирования UML (Unified Modelling Language) занял лидирующие позиции. Это связано с тем, что он предоставляет достаточный набор диаграмм для описания различных ракурсов системы. С помощью диаграмм UML можно описать и "реактивные" системы, но на довольно высоком уровне, что для анализа вышеперечисленных проблем не подходит. На диаграмме состояний, имеющихся в UML, невозможно показать, например, взаимодействие параллельных процессов и, следовательно, исследовать коллизии (в т.ч. временные). Также нет возможности наглядного представления процесса захвата разделяемых ресурсов разными процессами.

Поиск методов описания параллельных систем привел к рассмотрению сетей Петри как наиболее приемлемого инструмента для наглядного представления множества внутренних состояний и условий их изменения (функций перехода). В Европе активно развивается прикладное направление применения раскрашенных сетей Петри в промышленных проектах, поддерживаемое со стороны университетов как теоретически, так и практически.

Раскрашенные (цветные) сети Петри (РСП).

Теория раскрашенных сетей Петри (Coloured Petri Net, CP-net) разрабатывается более 20 лет рабочей группой (CPN Group) университета г.Орхуса (University of Aarhus, Denmark) под руководством профессора Курта Йенсена (Kurt Jensen). Этой группой разработана основная модель, включающая использование типов данных и иерархических конструкций, определены концепции динамических свойств, развивается теория методов анализа.

Раскрашенная сеть Петри (РСП) – это графоориентированный язык для проектирования, описания, имитации и контроля распределенных и параллельных систем [2,3]. Графическими примитивами показывается течение процесса, а конструкциями специального языка имитируется необходимая обработка данных.

В отличие от "классических" сетей Петри, в раскрашенных немаловажную роль играет типизация данных, основанная на понятии *множества цветов*, которое аналогично типу в декларативных языках программирования. Соответственно, для манипуляции *цветом* применяют переменные, функции и другие элементы, известные из языков программирования. Ключевой элемент РСП – позиция – имеет определенное значение из множества цветов.

Для отражения динамических свойств в сеть Петри введено понятие разметки сети, которая реализуется с помощью так называемых *фишек*, размещаемых в позициях. Цвет позиции определяет тип фишек, которые могут там находиться. Конкретизация фишки, находящейся в данной позиции, определяется инициализирующим выражением начальной разметки или формируется в результате правильного выполнения шага итерации сети Петри.

Сеть представляет собой асинхронную систему, в которой фишки перемещаются по позициям через переходы. Переход может сработать (т.е. переместить фишку из входной позиции в выходную для данного перехода), если во всех входных позициях для данного перехода присутствует хотя бы одна фишка и выполнено логическое выражение, ограничивающее переход (*спусковая функция*).

Дуги могут иметь пометки в виде выражений (переменных, констант или функций), определенных для множества цветов, и использоваться либо для "вычленения" компонентов сложного цвета фишек при определении условия срабатывания перехода, либо для изменения цвета фишки следующей позиции после срабатывания перехода.

Для анализа систем реального времени введен временной механизм, реализованный с помощью глобальных часов и так называемых *штампов*, которые несут

фишки. Временной штамп фишки назначается при ее инициализации в начальной разметке или при создании фишки переходом и наращивается выражениями на переходах или дугах. В результате фишка становится доступной для перехода, если ее штамп оказался меньше значения счетчика глобальных часов. Часы наращивают свое значение, если на данный момент времени ни один переход сети не разрешен.

Для "реактивных" систем позицию можно рассматривать как одно из состояний системы, уточняемое содержащейся в ней типизированной фишкой, причем отсутствие фишки указывает на “неактивность” данного состояния. Введение "цветной" фишки позволяет сократить число отображаемых однотипных состояний (позиций) и дает возможность проектировщику пользоваться дополнительной информацией, которую несет фишка.

Основным свойством сетей Петри, описывающих системы, является их способность отражать динамические характеристики моделей. Для этого проектировщику необходимо иметь программный инструмент, способный интерактивно вводить данные о позициях, переходах и дугах, описывать множества цветов, отражать процесс перемещения фишек. Одним из свободно распространяемых программных продуктов, позволяющий проводить указанные выше операции (а также ряд дополнительных), является программа CPNTools (http://www.daimi.au.dk/CPNTools/), разработанная в университете г.Орхуса (Дания). Программный продукт постоянно развивается и сопровождается группой, которая разрабатывает теорию раскрашенных сетей Петри.

Часть 3. Имитационное моделирование систем массового обслуживания

Для представления имитационных моделей можно использовать языки программирования общего применения, однако такие представления оказываются довольно громоздкими. Поэтому обычно применяют специальные языки имитационного моделирования на системном уровне. Среди языков имитационного моделирования различают языки, ориентированные на описание событий, средств обслуживания или маршрутов движения заявок (процессов). Выбор языка моделирования определяет структуру модели и методику ее построения.

Для описания имитационных моделей на системном уровне (иногда их называют *сетевыми имитационными моделями* — СИМ) чаще используют языки, ориентированные на события или процессы. Примерами первых могут служить языки Симскрипт, SMPL, GASP и ряд других. К числу вторых относятся языки Симула, SOL, а также популярный язык GPSS.

Языки имитационного моделирования реализуются в программно-методических комплексах моделирования СМО, имеющих ту или иную степень специализации. Так, комплексы на базе языка GPSS можно использовать во многих приложениях, но есть специализированные комплексы для моделирования вычислительных сетей (COMNET III, OPNET), вычислительных систем, производственных процесов (РДО) и т.п.

Примерами программ имитационного моделирования могут служить GPSS/H, GPSS/PC, Arena, SLX, ProcessModel, отечественная система РДО и др. В программе Arena использован входной графический язык. Пользователь может выбирать нужные блоки из меню и переносить их в поле модели.

При использовании языков, ориентированных на процессы, в составе СИМ выделяются элементарные части и ими могут быть источники входных потоков заявок, устройства, накопители и узлы.

Источник входного потока заявок представляет собой алгоритм, в соответствии с которым вычисляются моменты t_k появления заявок на выходе источника. Источники могут быть зависимыми и независимыми. В зависимых источниках моменты появления заявок связаны с наступлением определенных событий, например, с приходом другой

заявки на вход некоторого устройства. Типичным независимым источником является алгоритм выработки значений t_k случайной величины с заданным законом распределения.

Устройства в имитационной модели представлены алгоритмами выработки значений интервалов (длительностей) обслуживания. Чаще всего это алгоритмы генерации значений случайных величин с заданным законом распределения. Но могут быть устройства с детерминированным временем обслуживания или временем, определяемым событиями в других частях СИМ. Модель устройства отображает также заданную *дисциплину обслуживания*, поскольку в модель входит алгоритм, управляющий очередями на входах устройства.

Накопители моделируются алгоритмами определения объемов памяти, занимаемых заявками, приходящими на вход накопителя. Обычно объем памяти, занимаемый заявкой, вычисляется как значение случайной величины, закон и (или) числовые характеристики распределения может зависеть от типа заявки.

Узлы выполняют связующие, управляющие и вспомогательные функции в имитационной модели, например, для выбора направлений движения заявок в СИМ, изменения их параметров и приоритета, разделения заявок на части, их объединения и т.п.

Обычно каждому типу элементарной модели, за исключением лишь некоторых узлов, в программной системе соответствует определенная процедура (подпрограмма). Тогда СИМ можно представить как алгоритм, состоящий из упорядоченных обращений к этим процедурам, отражающим поведение моделируемой системы.

В процессе моделирования происходят изменения модельного времени, которое чаще всего принимается дискретным, измеряемым в тактах. Время изменяется после того, как закончена имитация очередной группы событий, относящихся к текущему моменту времени t_k. Имитация сопровождается накоплением в отдельном файле статистики таких данных, как количества заявок, вышедших из системы обслуженными и необслуженными, суммарное время занятого состояния для каждого из устройств, средние длины очередей и т.п. Имитация заканчивается, когда текущее время превысит заданный отрезок времени или когда входные источники выработают заданное число заявок. После этого производят обработку накопленных в файле статистики данных, что позволяет получить значения требуемых выходных параметров.

В программах имитационного моделирования СМО преимущественно реализуется *событийный метод* организации вычислений. Сущность событийного метода

заключается в отслеживании на модели последовательности событий в том же порядке, в каком они происходили бы в реальной системе. Вычисления выполняют только для тех моментов времени и тех частей (процедур) модели, к которым относятся совершаемые события. Другими словами, обращения на очередном такте моделируемого времени осуществляются только к моделям тех элементов (устройств, накопителей), на входах которых в этом такте произошли изменения. Поскольку изменения состояний в каждом такте обычно наблюдаются лишь у малой доли ОА, событийный метод может существенно ускорить моделирование по сравнению с инкрементным методом, в котором на каждом такте анализируются состояния всех элементов модели.

Рассмотрим возможную схему реализации событийного метода имитационного моделирования.

Моделирование начинается с просмотра операторов генерирования заявок, т.е. с обращения к моделям источников входных потоков. Для каждого независимого источника такое обращение позволяет рассчитать момент генерации первой заявки. Этот момент вместе с именем — ссылкой на заявку — заносится в список будущих событий (СБС), а сведения о генерируемой заявке — в список заявок (СЗ). Запись в СЗ включает в себя имя заявки, значения ее параметров (атрибутов), место, занимаемое в данный момент в СИМ. В СБС события упорядочиваются по увеличению моментов наступления.

Затем из СБС выбирают совокупность сведений о событиях, относящихся к наиболее раннему моменту времени. Эта совокупность переносится в список текущих событий (СТС), из которого извлекаются ссылки на события. Обращение по ссылке к СЗ позволяет установить место в СИМ заявки А, с которой связано моделируемое событие. Пусть этим местом является устройство X. Далее программа моделирования выполняет следующие действия (рис. 5):

1. изменяет параметры состояния устройства X; например, если заявка А освобождает X, а очередь к X не была пуста, то в соответствии с заданной дисциплиной обслуживания из очереди к X выбирается заявка В и поступает на обслуживание в X;
2. прогнозируется время наступления следующего события, связанного с заявкой В, путем обращения к модели устройства X, в которой рассчитывается продолжительность обслуживания заявки В; сведения об этом будущем событии заносятся в СБС и СЗ;

3. происходит имитация движения заявки А в СИМ по маршруту, определяемому заданной программой моделирования, до тех пор, пока заявка не придет на вход некоторого ОА; здесь либо заявка задерживается в очереди, либо путем обращения к модели этого ОА прогнозируется наступление некоторого будущего события, связанного с дальнейшей судьбой заявки А; сведения об этом будущем событии также заносятся в СБС и СЗ;

4. в файл статистики добавляются необходимые данные.

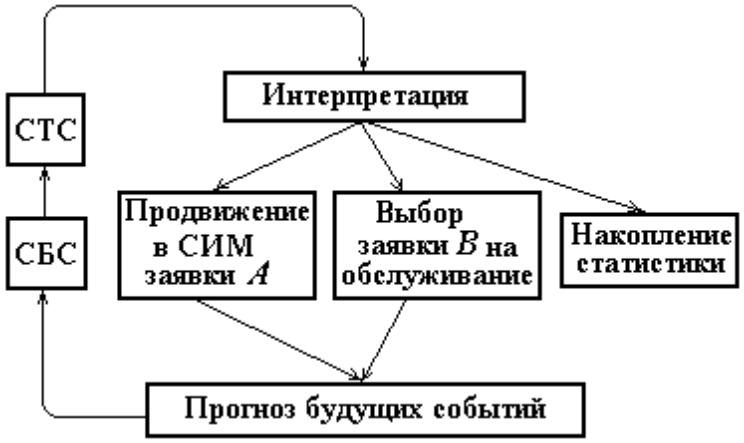

Рис. 5. Событийный метод имитационного моделирования

После отработки всех событий, относящихся к моменту времени t_k, происходит увеличение модельного времени до значения, соответствующего ближайшему будущему событию, и рассмотренный процесс имитации повторяется.

Примеры моделирования с помощью языка GPSS

Язык GPSS (General Purpose Simulation System), ориентированный на процессы, разработан еще в 1961 г., но продолжает широко использоваться [6,7]. Язык реализован в ряде программ имитационного моделирования, так, версия программы GPSS/PC в среде Windows создана в 2000 г.

Модель (программа) на языке GPSS представляет собой последовательность операторов (их называют блоками), отображающих события, происходящие в СМО при перемещениях транзактов. Поскольку в интерпретаторах GPSS реализуется событийный метод и в СМО может быть одновременно много транзактов, то интерпретатор будет попеременно исполнять разные фрагменты программы, имитируя продвижения транзактов в текущий момент времени до их задержки в некоторых устройствах или очередях.

Операторы (блоки) GPSS имеют следующий формат:

<метка> <имя_оператора> <поле_операндов> [<комментарий>]

Метка может занимать позиции, начиная со второй, имя оператора — с восьмой, поле операндов — с девятнадцатой, комментарий обязательно отделяется от поля операндов пробелом.

Поле операндов может быть пусто, иметь один или более операндов, обозначаемых ниже при описании блоков символами **A**, **B**, **C**,... Операндами могут быть идентификаторы устройств, накопителей, служебные слова и *стандартные числовые атрибуты* (СЧА). К СЧА относятся величины, часто встречающиеся в разных задачах. Это, например, такие операнды, как **S** — объем занятой памяти в накопителе, **F** — состояние устройства, **Q** — текущая длина очереди, **P** — параметр транзакта (каждый транзакт может иметь не более **L** параметров, где **L** зависит от интерпретатора), **V** — целочисленная переменная (вещественная и булева переменные обозначаются **FV** и **BV** соответственно), **X** — хранимая переменная (переменная, для которой автоматически подсчитывается статистика), **K** — константа, **AC1** — текущее время, **FN** — функция, **RN** — случайная величина, **RN1** — случайная величина, равномерно распределенная в диапазоне [0, 1] и др. При этом ссылки на СЧА записываются в виде **<СЧА>$<идентификатор>**. Например, **Q$ORD** означает очередь ORD или **FN$COS** — ссылка на функцию COS.

Рассмотрим наиболее часто встречающиеся операторы, сопровождая знакомство с ними простыми примерами моделей.

Источники заявок обычно описываются блоком

GENERATE A,B,C,D,E

Здесь **A** и **B** служат для задания интервалов между появлениями заявок, при этом можно использовать один из следующих вариантов:

- интервал — равномерно распределенная в диапазоне [A_B, A+B] случайная величина;
- интервал — значение функции, указанной в B, умноженной на A;

C — задержка в выработке первого транзакта; **D** — число вырабатываемых источником заявок; **E** — приоритет заявок. Если **D** пусто, то число вырабатываемых транзактов неограничено. Например:

GENERATE 6,FN$EXP,,15

Этот оператор описывает источник, который вырабатывает 15 транзактов с интервалами, равными произведению числа 6 и значения функции EXP;

GENERATE 36,12

Здесь число транзактов неограничено, интервалы между транзактами — случайные числа в диапазоне [24, 48].

Функции, на которые имеются ссылки в операторах, должны быть описаны с помощью блока следующего типа:

M FUNCTION A,B

За ним следует строка, начинающаяся с первой позиции :

X1,Y1/X2,Y2/X3,/.../Xn,Yn

Здесь метка **M** — идентификатор функции, **A** — аргумент функции, **B** — тип функции, **Xi** и **Yi** — координаты узловых точек функции, заданной таблично. Например:

EXP FUNCTION RN1,C12

0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000

Это описание непрерывной (**C**) функции EXP, заданной таблично 12-ю узловыми точками, аргументом является случайная величина (RN1), равномерно распределенная в диапазоне [0, 1].

Операторы занятия транзактом и освобождения от обслуживания устройства A:

SEIZE A

RELEASE A

Задержка в движении транзакта по СМО описывается оператором:

ADVANCE A,B

A и B имеют тот же смысл, что и в операторе **GENERATE**.

Пример 1

Обслуживание транзакта в устройстве WST продолжительностью a единиц времени, где a — равномерно распределенная в диапазоне [7,11] случайная величина, описывается следующим фрагментом программы

SEIZE WST

ADVANCE 9,2

RELEASE WST

Аналогично описывается занятие транзактом памяти в накопителе:

ENTER A,B

Здесь помимо имени накопителя (A) указывается объем занимаемой памяти (B).

Освобождение B ячеек памяти в накопителе A выполняется оператором:

LEAVE A,B

Для накопителей в модели нужно задавать общий объем памяти, что делается в следующем описании накопителя:

M STORAGE A

Здесь: M — имя накопителя, A — объем его памяти.

Если транзакт приходит на вход занятого устройства или на вход накопителя с недостаточным объемом свободной памяти, то транзакт задерживается в очереди к этому устройству или накопителю. Слежение за состоянием устройств и очередей выполняет интерпретатор. Но если в модели требуется ссылаться на длину очереди или собирать статистику по ее длине, то требуется явное указание этой очереди в модели. Делается это с помощью операторов входа в очередь и выхода из очереди:

QUEUE A,B

DEPART A,B

Согласно этим операторам очередь A увеличивается и уменьшается на B единиц соответственно, если B=1, то поле B можно оставить пустым.

Движение транзактов выполняется по маршруту, заданному последовательностью операторов в модели. Если требуется изменение естественного порядка, то используется оператор перехода. Оператор условного перехода имеет вид:

TEST XX A,B,C

В соответствии с ним переход к оператору, помеченному меткой C, происходит, если не выполняется условие A XX B, где XX $\in$ {**E**, **NE**, **L**, **LE**, **G**, **GE**}; **E** — равно; **NE** — неравно; **L** — меньше; **LE** — меньше или равно; **G** — больше; **GE** — больше или равно (XX всегда размещается в позициях 13 и 14).

Пример 2

Приходящие пользователи ожидают обслуживания, если длина очереди не более 4, иначе от обслуживания отказываются. Соответствующий фрагмент программы

```
TEST LE    Q$STR,4,LBL
QUEUE      STR
SEIZE      POINT
DEPART     STR
ADVANCE    50,16
RELEASE    POINT
```

LBL TERMINATE 1

В примере 2 использован оператор выхода транзактов из СМО:

TERMINATE A

Согласно этому оператору из итогового счетчика вычитается число А. С помощью итогового счетчика задается длительность моделирования. В начале исполнения программы в счетчик заносится число, указанное в операнде А оператора

START A,B,C

Моделирование прекращается, когда содержимое счетчика будет равно или меньше нуля. Операнд С — шаг вывода статистики на печать. Если В=0 и С=0, то выполняется только стандартная печать по окончании моделирования. В стандартную печать входят собранные за время моделирования статистические данные по основным параметрам модели: средние и максимальные значения длин очередей, объемов занимаемой памяти в накопителях, времени занятого состояния устройств и др. От печати можно отказаться, указав В=NP.

Пример 3

Общая структура программы на GPSS имеет вид

<описания, в том числе функций, накопителей, массивов и т.п.>

<операторы, моделирующие движение транзактов>

START A,B,C

END

Оператор безусловного перехода записывается следующим образом:

TRANSFER ,B

Здесь В — метка оператора, к которому следует переход.

Используется ряд других разновидностей оператора **TRANSFER**. Например:

TRANSFER P,B,C

Переход происходит к оператору с меткой, равной сумме значения параметра В транзакта и числа С.

TRANSFER FN,B,C

То же, но вместо параметра транзакта слагаемым является значение функции В.

TRANSFER PICK,B,C

Это оператор равновероятного перехода к операторам, метки которых находятся в интервале [B,C].

Важное место в СМО занимает переход по вероятности:

```
  TRANSFER  A,B,C
```

Здесь A — вероятность перехода к оператору с меткой C, переход к оператору с меткой B будет происходить с вероятностью 1 – A.

Пример 4

На вход производственной линии поступают и проходят обработку на станке TOOL1 детали типов X и Y. Далее детали типа X обрабатываются на станке TOOL2, а детали типа Y — на станке TOOL3. Интервал моделирования соответствует обработке 600 деталей (ниже у операторов **GENERATE** и **ADVANCE** значения операндов не конкретизированы):

```
  GENERATE  A,B
  ASSIGN    1,LBL4
  TRANSFER  ,LBL1
  GENERATE  A,B
  ASSIGN    1,LBL2
LBL1 SEIZE    TOOL1
  ADVANCE   A,B
  RELEASE   TOOL1
  TRANSFER  P,1
LBL4 SEIZE    TOOL2
  ADVANCE   A,B
  RELEASE   TOOL2
LBL3 TERMINATE 1
LBL2 SEIZE    TOOL3
  ADVANCE   A,B
  RELEASE   TOOL3
  TRANSFER  ,LBL3
START    600
END
```

Вычислительный оператор присваивает переменной с номером M значение арифметического выражения A:

```
 M   VARIABLE  A
```

Например, в следующем операторе переменной номер 3 присваивается разность числа 216 и объема занятой памяти в накопителе MEM2:

XINIT VARIABLE K216-S$MEM2

Знаки арифметических операций сложения, вычитания, умножения, деления +, -, #, / соответственно. В случае логических выражений имя оператора должно быть BVARIABLE, а знаками операций дизъюнкции и конъюнкции являются + и #. Если операции выполняются над числами типа real, то имя оператора FVARIABLE.

Часто сведения о некоторых величинах, характеризующих моделируемый процесс, удобно представлять в виде гистограмм. Задание гистограммы выполняют в разделе описаний с помощью оператора:

M TABLE A,B,C,D

Здесь M — имя гистограммы; A — табулируемая величина; B — верхняя граница левого интервала гистограммы; C — ширина интервалов; D — число интервалов. Формирование гистограммы происходит с помощью оператора:

TABULATE A

Выполнение этого оператора увеличивает на единицу число попаданий в i-й интервал гистограммы, имя которой указано в A. При этом i-й интервал соответствует текущему значению переменной, являющейся аргументом для гистограммы.

Пример 5

Требуется разработать модель процессов возникновения и устранения неисправностей в некоторой технической системе, состоящей из множества однотипных блоков; в запасе имеется один исправный блок; известны статистические данные об интенсивностях возникновения отказов и длительностях таких операций, как поиск неисправностей, замена и ремонт отказавшего блока. Поиск и замену отказавшего блока производит бригада TEAM1, а ремонт замененного блока — бригада TEAM2.

```
GENERATE   A,B   моделируется возникновение отказов
SEIZE      TEAM1
ADVANCE    A,B   поиск неисправности
ENTER      MEM,1 получение запасного блока из резерва
ADVANCE    A,B   замена блока
RELEASE    TEAM1
SEIZE      TEAM2
```

```
   ADVANCE    A,B   ремонт
   LEAVE      MEM,1 восстановление резерва
   RELEASE    TEAM2
   TERMINATE  1
START      1000
     END
```

Пример 6

Требуется разработать модель сборки изделия из 30 деталей типа А1 и 16 деталей типа А2, поступающих на сборочный участок от независимых экспоненциальных источников с интенсивностями λ , равными 0,1 и 0,04 мин$^{-}$ соответственно. Длительность сборочной операции находится в пределах [12,18] мин. Промоделировать выпуск 600 изделий. Табулировать наполнение входного бункера с деталями типа А2 перед началом сборки.

```
MEM1  STORAGE      30
 MEM2  STORAGE      16
 TAB   TABLE        MEM2,32,16,6
 EXP   FUNCTION     RN1,C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/
.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000
   GENERATE     10, FN$EXP
   ENTER        MEM1,1
   TRANSFER     ,MMM
   GENERATE     25, FN$EXP
   ENTER        MEM2,1
 MMM   TEST GE      S$MEM1,30,LLL
   TEST GE      S$MEM2,16,LLL
   TABULATE     TAB
   SEIZE        MONT
   ADVANCE      15,3
   RELEASE      MONT
   TERMINATE    1
 LLL   TERMINATE
```

START 600

END

Список литературы

1. М. Румянцев, Средства имитационного моделирования бизнес-процессов // Корпоративные системы, №2, 2007.
2. Верификация Estelle-спецификаций распределенных систем посредством раскрашенных сетей Петри.// Под ред. Непомнящего В.А., Шилова Н.В. - Новосибирск,1997.
3. Гома Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений. Пер.с англ. - М. ДМК Пресс 2002 704 с.
4. Котов В.Е. Сети Петри. - М.:Наука,1984.
5. Питерсон Дж. Теория сетей Петри и моделирование систем. - М.:Мир,1984.
6. Томашевский В., Жданова Е. Имитационное моделирование в среде GPSS. — М.: Бестселлер, 2003.
7. GPSS. — http://www.compmodel.ru/394/