Философские аспекты моделирования

Почему моделирование? – дешевел и можно моделировать любую фантазию.

Адекватность модели – "приближенность"к реальному миру.

Объект — это все то, на что направлена человеческая деятельность. Свойства объекта характеризуются данными.

Тип данных определяется множеством возможных значений и множеством операций над ним.

Методика – упорядоченная совокупность методов.

Выработка методологии направлена на упорядочение получения и обработки информации об объектах, которые существуют вне нашего сознания и взаимодействуют между собой и внешней средой.

Научно-техническое развитие в любой области обычно идет следующим путем:

1. Наблюдение и эксперимент.
2. Теоретическое исследование.
3. Организация производственного процесса.

Гипотезы – предсказания, основанные на небольшом количестве опытных данных, наблюдений, догадок. Быстрая и полная проверка гипотез может быть проведена в ходе специально поставленного эксперимента.

Аналогия — суждение о частном сходстве двух объектов.

Современная научная гипотеза создается как правило по аналогии с проверенными на практике положениями. Следовательно, именно аналогия связывает гипотезу и эксперимент.

Гипотезы и аналогии, отражающие реальный, объективно существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам. Такие логические схемы, упрощающие рассуждения и логические построения или позволяющие проводить эксперименты, уточняющие природу явлений, называются моделями.

Модель — это объект-заместитель объекта-оригинала, обеспечивающий изучение определенных свойств последнего.

Моделирование — это процесс замещения одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели.

Классификация видов моделирования

Самое главное при классификации — критерий классификации.

Модерирование систем:

1. ***Детерминированное*** отражает процессы, в которых предполагается отсутствие случайных воздействий.
2. ***Стохастическое*** отображает вероятностные процессы.

1. ***Статическое*** служит для описания поведения объектов в какой-либо момент времени
2. ***Динамическое*** отображает состояние системы во времени.

1. ***Дискретное*** — в дискретные моменты времени
2. ***Непрерывное*** — на протяжении некоторого времени
3. ***Дискретно-непрерывное*** моделирование используется для случаев, когда необходимо выделить наличие как дискретных, так и непрерывных процессов.

1) ***Мысленное*** моделирование применяется для исследования объектов, которые либо практически нереализуемы в заданном интервале времени, либо существуют вне условий, возможных для их физического создания
   a) При ***наглядном*** моделировании на базе представлений человека о реальных объектах создаются различные наглядные модели, отображающие явления и процессы, протекающие в объекте.
      i) ***Гипотетическое*** - исследователем закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте, которая отражает уровень знаний исследователя об объекте и базируется на причинно-следственных связях между входом и выходом изучаемого объекта. Используется, когда знаний об объекте недостаточно для построения формальных моделей.
      ii) ***Аналоговое*** - основывается на применении аналогий различных уровней.
      iii) ***Мысленный макет*** может применяться в случаях, когда протекающие в реальном объекте процессы не поддаются физическому моделированию, либо может предшествовать проведению других видов моделирования. В основе построения мысленных макетов также лежат аналогии, однако обычно базирующиеся на причинно-следственных связях между явлениями и процессами в объекте.
   b) ***Символическое*** - искусственный процесс создания логического объекта, который замещает реальный и выражает основные свойства его отношений с помощью определенной системы знаков или символов.
      i) Если ввести условное обозначение отдельных понятий (знаки), а также определенные операции между этими знаками, то можно реализовать ***знаковое*** моделирование и с помощью знаков отображать набор понятий — составлять отдельные цепочки из слов и предложений.
      ii) В основе ***языкового*** моделирования лежит тезаурус — словарь, в котором каждому слову может соответствовать лишь единственное понятие.
   c) Под ***математическим*** - процесс установления соответствия реальному объекту некоторого математического объекта, называемого математической моделью, и исследование этой модели, позволяющее получить характеристики объекта-оригинала.
      i) Для ***аналитического*** - процессы функционирования элементов системы записываются в виде функциональных соотношений или логических условий.

Алгоритм — это упорядоченная последовательность действий, направленная на достижение конечной цели.

Аналитическая модель может быть исследована тремя способами.

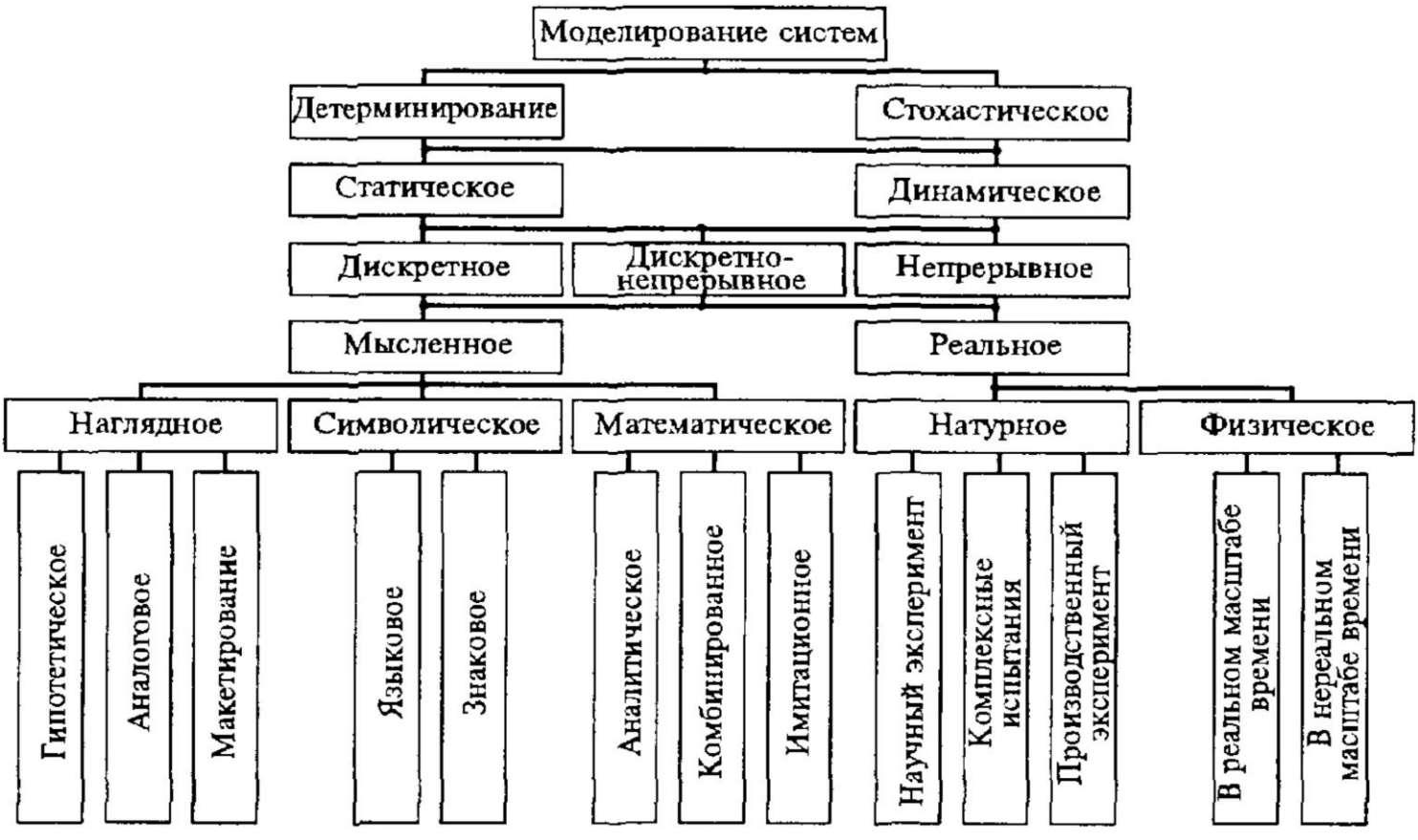
1. Аналитическим. Здесь стремятся получить в общем виде зависимости от искомых характеристик.
2. Численным. Когда нельзя решить уравнение в общем виде, то получают результаты для конкретных начальных данных.
3. Качественным. Когда не имея решения можно найти свойства решения.

ii) ***Имитационное*** моделирование - реализует модель алгоритма. Воспроизводят процесс функционирования системы во времени, причем имитируются элементарные явления составляющих процесса с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени, дающие возможность оценить характеристики системы.

Основным преимуществом имитационного моделирования по сравнению с аналитическим является *возможность решения более сложных задач*.

iii) ***Комбинированное*** (аналитико-имитационное). Проводится предварительная декомпозиция процесса функционирования объекта на составляющие подпроцессы, и для тех из них, где это возможно, используются аналитические модели, а для остальных подпроцессов строятся имитационные модели.

2) При ***реальном*** моделировании исследуется реальный объект целиком или частично.
   a) ***Натурным*** моделированием называют проведение исследования на реальном объекте с последующей обработкой результатов эксперимента на основе теории подобия.
   b) **Физическое** моделирование отличается от натурного тем, что исследование проводится на установках, которые сохраняют природу явлений и обладают физическим подобием.



Трудности использования имитационного моделирование:

- Обеспечение адекватности описания системы
- Интерпретация результатов стохастической сходимости моделирования
- Большая трудоемкость

Самый сложный этап имитационного моделирования — формализация модели. Необходимо четко определить входные данные, внутреннее содержание модели, выходные данные и воздействия внешней среды.
Нерешенной проблемой остается присутствие в процессе имитационного моделирования неформального этапа перевода поставленной задачи на язык математики. Этот этап не может быть решен ни программистом, ни аналитиком самостоятельно — необходима работа нескольких человек разной квалификации.

Виды имитационного моделирования

Агентное моделирование

Используется для исследования децентрализованных систем, динамика функционирования которых определяется результатом индивидуальной активности членов группы. Целью является определение глобальных правил и законов и общем поведении системы исходя из предположений о частном поведении агентов и их взаимодействий в системе.
Объект (Агент) — некая сущность, обладающая активностью и способная принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружающей средой и самостоятельно изменяться.
Пример: автомобильный трафик (объект — движущееся средство).

Дискретно-событийное моделирование

Рассматриваются только основные события: например, ожидание или обработка заказа. Абстрагирование от непрерывной природы объекта и событий, присущих данным объектам.
Пример: производственные процессы, логистика.

Системная динамика

Для исследования системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров объектов на другие. Основным параметром является время.
Пример: бизнес-процессы, динамика популяций и тд

Основные этапа средств имитационного моделирования:

1. Создание имитационной модели на языке программирования.
2. Использование при разработке имитационных моделей проблемно-ориентированных систем. Эти системы, как правило, не требуют знания программирования.
3. Использование методов искусственного интеллекта.

Технические средства мат. моделир-я (цифровые и аналоговые машины)

- как средства расчета (по полученным аналитическим моделям);
- как средства разработки программ.

При моделировании используется два вида вычислительной техники:

Цифровые машины – принцип счета

(процессор, память, ОС)

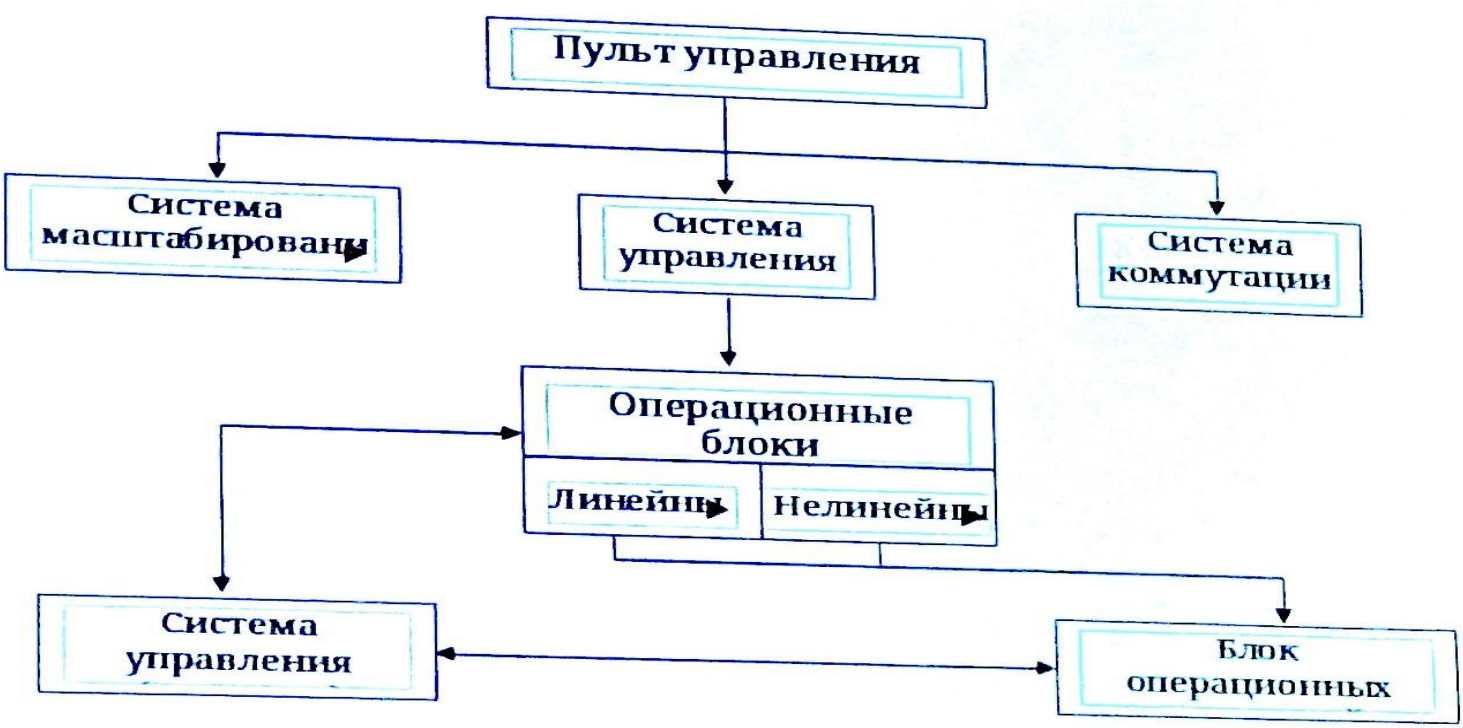
Самая большая проблема ограничение быстродействия

Аналоговые машины – принцип моделирования.

Каждой переменной величине задачи ставится в соответствие определенная переменная величина электронной цепи. При этом основой построения такой модели является изоморфизм — *подобие исследуемой задачи и соответствующей ей электронной цепи*. *Аналоговая вычислительная машина* — это совокупность электрических элементов, организованных в систему, позволяющую изоморфно (подобие) моделировать динамику исследуемого объекта.

+ быстродействие
– низкая точность вычислений
– неуниверсальность

Структурная схема АВМ:



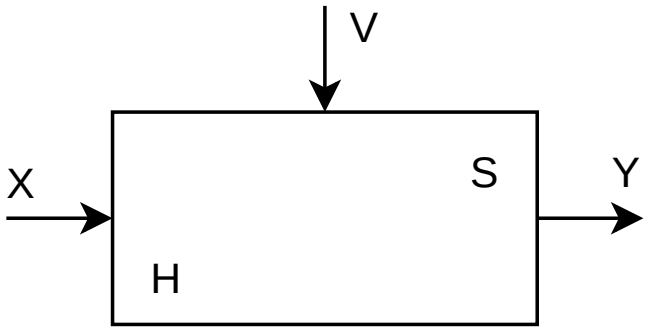
Гибридные вычислительные машины (ГВМ)

Это широкий класс вычислительных систем, использующий как аналоговую так и дискретную форму представления и обработки информации. Цель – сочетать быстродействие аналоговой и точность цифровой техники.

АВМ <–> Электронная система согласования <–> ЭВМ

Сравнение аналоговой и цифровой техники

	АВМ	**ЭВМ**
Тип информации	Непрерывный	Дискретный
Изменение значений	Величиной напряжения	Числовым значением
Базовые операции	Арифметические операции и интегрирование	Арифметические операции
Принцип вычисления	Высокопараллельный	Ограничен
Динамическое изменение решаемой задачи	Посредством системы коммутации	В диалоговом режиме
Требования к пользователю	Профессиональные знания, методика моделирования	Знание основ ПО ЭВМ
Уровень формализации задачи	Ограничен моделью решаемой задачи	Высокий
Способность к решению задач	Ограничена	Высокая
Точность вычисления	Ограничена (10^{-4})	Ограничена разрядностью (10^{-40})
Диапазон представления чисел	$1…10^4$	Зависит от разрядности
Класс решаемых задач	Алгебраические и дифф. уравнения	Любые
Специальные функции	Ограниченный набор	Неограниченный набор
Уровень миниатюризации	Ограничен	Высокий
Сфера применения	Ограничена	Практически любая



Модель объектного моделирования можно представить в виде множества величин, описывающих процесс функционирования реальной системы и образующих в общем случае следующие подмножества.

1. Совокупность входных воздействий $x_i \in X$, $i = \overline{1, n_x}$.
2. Совокупность воздействий внешней среды $v_i \in V$, $l = \overline{1, n_V}$.
3. Совокупность внутренних (собственных) параметров системы $h_k \in H$, $k = \overline{1, n_H}$.
4. Совокупность выходных характеристик системы $y_j \in Y$, $j = \overline{1, n_Y}$.

В общем случае x_i, v_l, h_k и y_j являются элементами непересекающихся подмножеств и содержат как детерминированные, так и стохастические составляющие. При моделировании функционирования такой системы входные воздействия, воздействия внешней среды и внутренние параметры являются независимыми (экзогенными):

$$\overrightarrow{x(t)} = (x_1(t), x_2(t), \dots, x_{n_X}(t))$$
$$\overrightarrow{v(t)} = (v_1(t), v_2(t), \dots, v_{n_V}(t))$$
$$\overrightarrow{h(t)} = (h_1(t), h_2(t), \dots, h_{n_H}(t))$$

Выходные характеристики системы являются зависимыми (эндогенными):

$$\overrightarrow{y(t)} = (y_1(t), y_2(t), \dots)$$

Процесс функционирования системы S описывается во времени:

$$\overrightarrow{y(t)} = F_S(\vec{x}, \vec{v}, \vec{h}, t)$$

Эта зависимость F_s называется ***законом функционирования системы***.

Алгоритм функционирования — метод получения выходных характеристик с учетом входных воздействий, воздействий внешней среды и соответствующих внутренних параметров системы.

Закон функционирования может быть рассмотрен как последовательную смену состояний $z_1, z_2, \dots, z_p$, которые интерпретируются как координаты точки в p-мерном пространстве (каждой реализации процесса будет соответствовать некоторая фазовая траектория):

$$\vec{z} = (z_1(t), z_2(t), \dots, z_p(t))$$

Пространство состояний объекта моделирования — это совокупность всех значений состояний.

Состояние системы в некоторый момент времени t полностью определяется некоторыми начальными условиями, входными воздействиями, внутренними параметрами, воздействиями внешней среды, которые имели место за время $t - t_0$:

$$\vec{z^0} = (z_1^0, z_2^0, \ldots, z_p^0)$$
$$\overrightarrow{z(t)} = \Phi(\vec{z^0}, \vec{x}, \vec{v}, \vec{h}, t)$$
$$\overrightarrow{y(t)} = F(\vec{z}, t)$$

функционирования через состояние:

$$\overrightarrow{y(t)} = F(\Phi(\vec{z^0}, \vec{x}, \vec{h}, t))$$

Под математической моделью реальной системы понимают конечное множество переменных x(t), v(t) и h(t) (входные, внешние и внутренние параметры) вместе с математическими связями между ними.

Типовые математические схемы

В практике моделирования на первоначальных этапах формализации объекта используют *типовые математические схемы*, к которым можно отнести хорошо разработанные и многократно проверенные на практике математические объекты.

Процесс функционирования системы	**Типовая математическая схема**	**Обозначение**
Непрерывно-детерминированный подход	стандартные ДУ	D-схема
Дискретно-детерминированный подход	конечные автоматы	F-схема
Дискретно-стохастический подход	вероятностные автоматы	P-схема
Непрерывно-стохастический подход	система массового обслуживания	Q-схема
Обобщенные (универсальный)	агрегативная система	A-схема

Непрерывная (т е изменяется плавно во времени) (против дискретная). Детерминированная (т е известны законы (уравнения), которые управляют её поведением и мы можем точно рассчитать состояние системы в любой будущий или прошлый момент времени.) (против стохастическая)

В **A-схеме** какие угодно блоки, самое главное определить операции над ними. Переходя со слоя на слой пытаемся построить математическую схему для каждого слоя. Самая важная операция для аггрегативных систем композиция и декомпозиция.

D-схема (Непрерывно-детерминированный подход)
отражают динамику изучаемой системы, т. е. ее поведение во времени.
Т.о. использование D-схем позволяет формализовать процесс функционирования непрерывной детерминированной системы и оценить их основные характеристики.
Применяя аналитический или имитационный подход, реализованный на соответствующем языке моделирования.
Такие системы чаще всего моделируются с помощью **обыкновенных дифференциальных уравнений (ОДУ)**.
Стандартная форма: dx/dt = f(x, t)
Пример:

Процесс малых колебаний маятника описывается обыкновенным дифференциальным уравнением

$$m_m l_m^2 \frac{d^2\theta(t)}{dt^2} + m_m g l_m \theta(t) = 0$$

где m_m, l_m — масса и длина подвеса маятника; *g* — ускорение свободного падения; *в \theta(t)* — угол отклонения маятника в момент времени *t.*

$$T_m = 2\pi\sqrt{\frac{l_m}{g}}$$

Из этого уравнения свободного колебания маятника можно найти период колебания маятника

Базовые понятия теории систем

Система — множество элементов, находящихся в отношениях и связанных между собой.
Элемент — часть системы, представление о которой нецелесообразно подвергать дальнейшему членению.
Сложная система — система, характеризующаяся большим числом элементов и (что наиболее важно) большим числом взаимосвязей элементов. Сложная система определяется также видом взаимосвязей элементов, свойствами целенаправленности, целостности, членимости, иерархичности, многоаспектности.
Подсистема — часть системы (подмножество элементов и их взаимосвязей), которая имеет свойства системы.
Надсистема — система, по отношению к которой рассматриваемая система является подсистемой.
Структура — отображение совокупности элементов системы и их взаимосвязей.
Понятие структуры отличается от понятия самой системы также еще и тем, что при описании структуры принимают во внимание лишь типы элементов и связей без конкретизации значений их параметров.
Параметр — величина, выражающая свойства системы или ее части.
Имитационное моделирование - реализует модель алгоритма. Воспроизводят процесс функционирования системы во времени, причем имитируются элементарные явления составляющих процесса с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени, дающие возможность оценить характеристики системы.

Характеристики сложных систем (главные)

1. Целенаправленность — свойство искусственной системы, выражающее назначение системы. Необходимо для оценки вариантов системы.
2. Целостность — свойство системы, характеризующее взаимосвязанность элементов и наличие зависимости выходных параметров от параметров элементов, причем большинство выходных параметров не являются простым повторением или суммой параметров элементов.
3. Иерархичность — свойство сложной системы, выражающее возможность и целесообразность ее иерархического описания, то есть представления в виде нескольких уровней, между компонентами которых имеется отношение «часть — целое».

Моделирование сложных дискретных систем имеет две задачи:

- создание моделей
- анализ свойств систем на основе исследования их моделей

Последовательность разработки и компьютерной реализации

Сущность компьютерного моделирования систем состоит в проведении эксперимента с моделью — программой, описывающей формально или алгоритмически поведение элементов системы в процессе функционирования системы, то есть в их взаимодействии друг с другом и внешней средой.

Основные требования, предъявляемые к модели:

- ***полнота*** модели должна предоставлять пользователю возможность получения необходимого набора характеристик, оценок системы;
- ***гибкость*** модели должна давать возможность воспроизведения различных ситуаций при варьировании структуры, алгоритмов и параметров моделей, причем структура должна допускать возможность замены, добавления, исключения некоторых частей без изменения всей модели;
- компьютерная реализация модели должна ***соответствовать имеющимся техническим ресурсам.***

Основные этапы моделирования больших систем

Первый этап - формулируется модель, и строится ее *формальная схема*.
Цель -- переход от содержательного описания объекта к его математической модели.
Исходные данные— содержательное описание объекта.
Последовательность действий:

1. Проведение границы между системой и внешней средой.
2. Выделения основных составляющих процесса функционирования по отношению к цели моделирования.
3. Исключение из рассмотрения второстепенных элементов описания, не оказывающих существенного влияния на ход процессов, исследуемых с помощью модели.
4. Оставшиеся элементы модели (системы) группируются:
    - имитатор воздействия внешней среды;
    - имитатор воздействия модели;
    - качественная обработка результатов.
5. Процесс функционирования сложной системы разбивается на подпроцессы так, чтобы построение модели отдельных процессов было элементарно и не вызывало особых трудностей.

Второй этап алгоритмизация модели (модель реализуется в виде программы).

1. Разработка схемы моделирующего алгоритма.
2. Разработка схемы программы (модели).
3. Выбор технических средств для реализации программной модели.
4. Программирование, отладка.
5. Проверка достоверности на тестовых примерах.
6. Составление технической документации (логические схемы, текст программы, спецификации).

Третий этап -- получения и интерпретации результатов.

1. Планирование компьютерного эксперимента. Составление плана эксперимента с указанием комбинаций переменных, параметров. Главная задача — дать максимальный объем информации об объекте при минимальных затратах компьютерного времени.
2. Проведение рабочих расчетов (контрольная калибровка модели).

3. Статистическая обработка результатов эксперимента.
4. Составление технической документации.

Классы ошибок

Выделяют три класса ошибок.

1. Ошибки формализации: недостаточно полная модель.
2. Ошибки решения: метод не позволяет достичь заданной точности.
3. Ошибки задания параметров модели.

Нарушение адекватности:

- проверка адекватности модели некоторой системы заключается в анализе ее соразмерности и равнозначности системы
- адекватность нарушается из-за идеализирования выполнения условий, особенностей режима функционирования, пренебрежение некоторыми случайными факторами
- считаем, что модель адекватна системе если вероятность того что отклонение выходного параметра не превышает предельной величины больше допустимой вероятности

Нужно проводить оценку по всем параметрам!

Проверка:

- модели элементов
- модели входных условий
- концептуальной модели
- как измерили/оценили значения выходных характеристик
- програмируемые модели

Проверка приводит к изменениям модели

1. глобальные (методические) ошибки - ошибки в концепте или матем. модели.
2. локальные - уточнение алгоритмов
3. параметрические

Зафиксировать область применения модели – множество условий при соблюдении которых точность результатов моделирования находится в допустимых пределах.

Рис. 4.1: Схема взаимосвязи технологических этапов моделирования

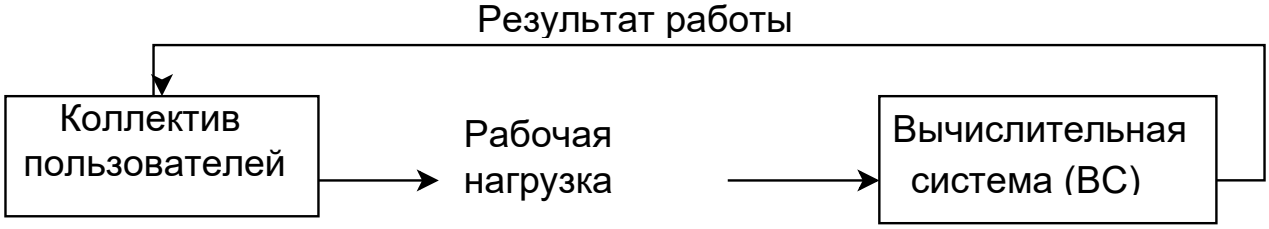
Уровни (проектирования) моделирования

1. Структурный – контроль функционирования всей системы (процессор, память, каналы, передача данных)
2. Функционально-логический уровень проектирования
    a. Регистровый подуровень – АЛУ конвейерная обработка
    b. Логический подуровень – каждое устройство выполняет свои действия
3. Схематический
4. Конструктивный – оптимальное размещение, отведения тепла

Каждый уровень использует разные математические модели

Моделирование на системном(структурном) уровне

Структура — отображение совокупности элементов системы и их взаимосвязей.
Понятие структуры отличается от понятия самой системы также еще и тем, что при описании структуры принимают во внимание лишь типы элементов и связей без конкретизации значений их параметров.
Вычислительная система — комплекс аппаратных и программных средств, которые в совокупности выполняют определенные рабочие функции.
Операционная система — набор ручных и автоматических процедур, которые позволяют группе людей эффективно использовать вычислительную установку. Основная черта операционных систем — согласованность работы управляющих программ.
Коллектив пользователей — сообщество людей, которое использует систему для удовлетворения своих нужд по обработке информации.
Входная информация — программы, данные, которые создаются коллективом пользователей, и называются *рабочей нагрузкой*.



Индекс производительности — описатель, который используется для представления производительности в самом широком смысле слова. Различают количественные и качественные индексы производительности.

- **Качественные индексы производительности**, например, являются удобство использования, субъективная легкость использования системы, мощность системы команд.
- **Количественные индексы производительности**, являются:
    - пропускная способность — объем информации, обрабатываемый в единицу времени;
    - время ответа (реакции) — время между предъявлением системе входных данных и появлением соответствующей выходной информации.
    - Коэффициент использования оборудования — отношение времени использования заданной части оборудования к определенному временному интервалу.

Концептуальная модель вычислительной системы включает также сведения о выходных и конструктивных параметрах системы, ее структуре, особенностях работы каждого элемента, характере взаимодействия между ресурсами.
Основные решаемые задачи:

- определение принципов организации вычислительной системы;
- выбор архитектуры;
- уточнение функций вычислительной системы;
- разделение функций на подфункции, реализуемые аппаратным и программным путем;
- разработка структурной схемы (определение состава устройств и способов их взаимодействия);
- определение требований к выходным параметрам устройств.

(Q-схема) Особенности непрерывного стохастического подхода

Q-схема реализует непрерывно-стохастический подход, используется для моделирования систем массового обслуживания.

рассмотрим на примере использования в качестве типовой математической схемы систем массового обслуживания (СМО). Характерным для таких объектов является случайное появление заявок на обслуживание и случайное завершение обслуживания. Случайность относится только к моментам времени.

В любом элементарном акте обслуживания можно выделить две основные составляющие:

- ожидание обслуживания
- само обслуживание.

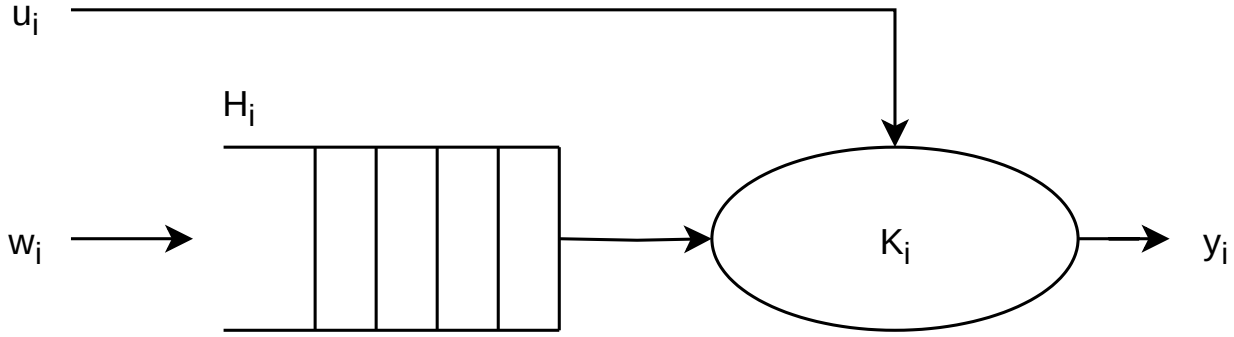


Рис. 5.1: i-ый прибор обслуживания

В накопителе H_i может находиться от 0 до емкости накопителя заявок.

Поток событий

Поток событий — последовательность событий, происходящих одно за другим в случайные моменты времени.

- Поток называется ***однородным***, если он характеризуется только моментами поступления.
- Поток называется ***неоднородным***, если он задается последовательностью, в которой кроме вызывающих моментов есть набор признаков события (например, приоритетность).

Если интервалы времени между сообщениями не зависят между собой и являются случайными, то поток имеет ***ограниченное последействие***.

Поток событий называется

- ***ординарным***, если вероятность того, что на малый интервал времени Δt попадает больше одного события, пренебрежительно мала по сравнению с вероятностью того, что на этот интервал времени попадает ровно одно событие.
- ***стационарным***, если вероятность появления того или иного числа событий на интервале времени зависит лишь от длины участка и не зависит от того, где на оси времени располагается этот участок. Для стационарного потока его интенсивность не зависит от времени и представляет собой постоянное значение, равное числу средних событий.

Среднее число сообщений, поступающих в единицу времени, — **интенсивность ординарного потока**:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{P_1(t, \Delta t)}{\Delta t}$$

В i-ом обслуживающем приборе имеем:

- поток заявок w_i — интервалы времени между появлением заявок на входе команд;
- поток обслуживания u_i — интервалы времени между началом и окончанием обслуживания заявки.

Заявки, обслуженные каналом, и заявки, покинувшие прибор необслуженными, образуют выходной поток y_i.

Процесс функционирования i-ого прибора обслуживания можно представить как процесс изменения состояний его элементов во времени. Переход в новое состояние для i-ого прибора означает изменение количества заявок, которые могут находиться либо в канале, либо в накопителе. Вектор состояния этого прибора состоит из двух компонентов: состояния накопителя и состояния канала.
Состояние канала — канал свободен или занят обслуживанием заявки.
В практике моделирования элементарные Q-схемы обычно объединяют, при этом, если каналы различных приборов обслуживания соединены параллельно, то имеет место *многоканальное* обслуживание, а если последовательно — *многофазное*. Следовательно, для задания Q-схемы необходимо использовать оператор сопряжения, отражающий взаимосвязь элементов структуры.

Различают **разомкнутые** (выходной поток не может поступать заново на обслуживание) и **замкнутые Q-схемы**.
Собственные внутренние параметры Q-схемы:

- количество фаз;
- количество каналов в каждой фазе;
- количество накопителей в каждой фазе;
- емкости накопителей.

Если емкость накопителя равна 0, то **система *с потерями***. Если емкость равна бесконечности — **система *с ожиданием***.
Для задания Q-схемы также необходимо описать алгоритм ее функционирования, который определяет набор правил поведения заявок в системе в различные моменты времени.
Неоднородность заявок, отражающая процесс в той или иной реальной системе, может быть определена классами приоритетов.
Весь набор возможных алгоритмов поведения заявок в Q-схемах можно представить в виде некоторого оператора алгоритма.

$$Q = (W,U,R,A,H,Z)$$

Здесь
W — подмножество входящих потоков,
U — подмножество потоков обслуживания,
R — оператор сопряжения,
A — оператор алгоритма,
H — вектор внутренних параметров,
Z — множество состояний.

Марковские процессы

Случайный процесс, протекающий в некоторой системе, называется ***марковским***, если для каждого момента времени вероятность любого состояния системы в будущем зависит только от состояния системы в настоящем и не зависит от того, когда и каким образом система пришла в это состояние. (Примера таких систем в реальности нет)

В марковском случайном процессе будущее развитие зависит только от настоящего состояния и не зависит от предыстории процесса. Для марковского случайного процесса составляют *уравнения Колмогорова*, представляющие собой соотношения следующего вида:

$$F(P'(t), P(t), \Lambda) = 0$$

Здесь $P(t)$ — вероятность нахождения в состоянии для сложной системы, Λ — коэффициенты, показывающие, с какой скоростью система переходит из одного состояния в другое (интенсивность).

$\Phi(P(t), \Lambda) = 0$ -- для стационарного состояния

$Y = Y(P(t), \Lambda) = Y(P(\Lambda))$ -- зависимость выходных параметров от внутренних параметров модели и является *базисной моделью*.

$Y = Y(X,V,H)$

$\Lambda = \Lambda(X,V,H)$ — интерфейсная модель.

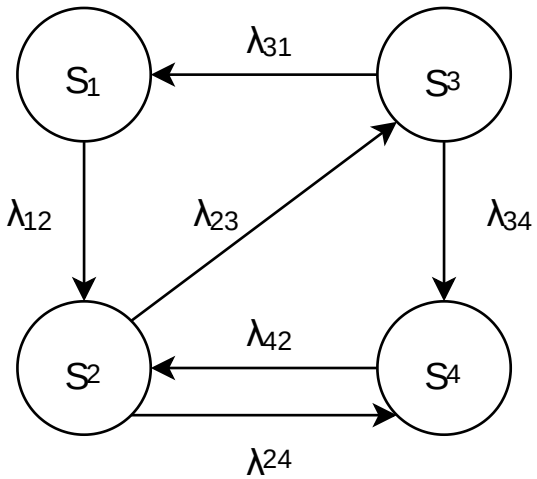
Следовательно, математическая модель системы строится как совокупность базисной и интерфейсной моделей, что позволяет использовать одни и те же базисные модели для различных задач моделирования, осуществляя настройку на соответствующую задачу посредством изменения интерфейсной модели.

Все уравнения Колмогорова построены по следующим правилам.

1. В левой части каждого уравнения стоит производная вероятности, а правая часть содержит столько членов, сколько стрелок связано с данным состоянием.
2. Если стрелка направлена из состояния, соответствующий член имеет знак «−», если в состояние — «+».
3. Каждый член равен произведению плотности вероятности перехода (интенсивности), соответствующей данной стрелке, на вероятность того состояния, из которого выходит стрелка.

Пример:

Пусть есть некоторая сложная система S.



Найдем вероятность $P_1(t)$, то есть вероятность того, что в момент времени t система будет находиться в состоянии S_1.

Придадим t малое приращение Δt и найдем вероятность того, что в момент $t + \Delta t$ система будет находиться в состоянии S_1. Вероятность того, что система была в состоянии S_1 и за

время Δt не вышла из него, найдем как вероятность $P_1(t)$, то есть того, что в момент t система была в состоянии S_1, на условную вероятность того, что, будучи в состоянии S_1 система не перейдет из него в S_2. Эта условная вероятность с точностью до бесконечно малых высших порядков равна

$$P_1(t+\Delta t) = P_1(t)\cdot(1-\lambda_{12}\Delta t) + P_3(t)\lambda_{32}\cdot\Delta t$$

$$\frac{P_1(t+\Delta t)-P_1(t)}{\Delta t} = -P_1(t)\lambda_{12} + P_3(t)\lambda_{31}$$

$$\lim_{\Delta t\to 0}\left(\frac{P_1(t+\Delta t)-P_1(t)}{\Delta t}\right) = -P_1(t)\lambda_{12} + P_3(t)\lambda_{31}$$

$$P_1'(t) = -P_1(t)\lambda_{12} + P_3(t)\lambda_{31}$$

Состояние S_2 может быть определено как:

- система перешла из S_1;
- система перешла из S_4;
- система, будучи в S_2, не перешла в S_3;
- система, будучи в S_2, не перешла в S_4.

В таком случае уравнения Колмогорова:

$P_2'(t) = -P_2(t)\cdot(\lambda_{23}+\lambda_{24}) + P_1(t)\lambda_{12} + P_4(t)\lambda_{42}$

$P_3'(t) = -P_3(t)\cdot(\lambda_{31}+\lambda_{34}) + P_2(t)\lambda_{23}$

$P_4'(t) = -P_4(t)\lambda_{42} + P_2(t)\lambda_{24} + P_3(t)\lambda_{34}$

Интегрирование данной системы дает искомые вероятности состояний как функции времени. Начальные условия берутся в зависимости от того, каково было начальное состояние системы. Например, если при $t = 0$ система находилась в состоянии S_1, то начальные условия:

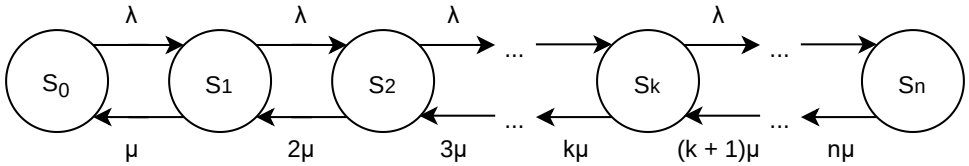
$P_1(0) = 1, P_2(0) = 0, P_3(0) = 0, P_4(0) = 0$

В эту же систему можно добавить уравнение (условие) нормировки: когда сумма всех вероятностей равна 1 в любой момент времени.

Пример многоканальной СМО с отказами:

Будем нумеровать состояния системы по числу занятых каналов (по числу заявок, связанных с системой):

- S_0 — все каналы свободны;
- S_1 — занят 1 канал, остальные свободны;
- S_k — занято k каналов, остальные свободны;
- S_n — заняты все n каналов.



Разметим граф, то есть проставим у стрелок интенсивности соответствующих потоков событий. Поток заявок (слева направо) имеет интенсивность λ.

Определим интенсивность потока событий, которые переводят систему по стрелкам справа налево. Пусть система была в состоянии S_1, тогда, как только закончится обслуживание заявки, занимающей этот канал, система перейдет в состояние S_0. Данный поток обслуживания заявок имеет интенсивность μ. Очевидно, что если заняты 2 канала, а не 1, то поток обслуживаний, переводящий систему из S_2 в S_1, будет вдвое интенсивней.

$$p_0 = \frac{1}{1 + \frac{\lambda}{\mu} + \frac{\left(\frac{\lambda}{\mu}\right)^2}{2!} + \cdots + \frac{\left(\frac{\lambda}{\mu}\right)^n}{n!}}$$

$$p_k = \frac{\left(\frac{\lambda}{\mu}\right)^k}{k!} p_0$$

$$\frac{\lambda}{\mu} = \rho$$

$$p_{\text{отк}} = p_n = \frac{\rho^n}{n!} p_0$$

$$q = 1 - p_n$$

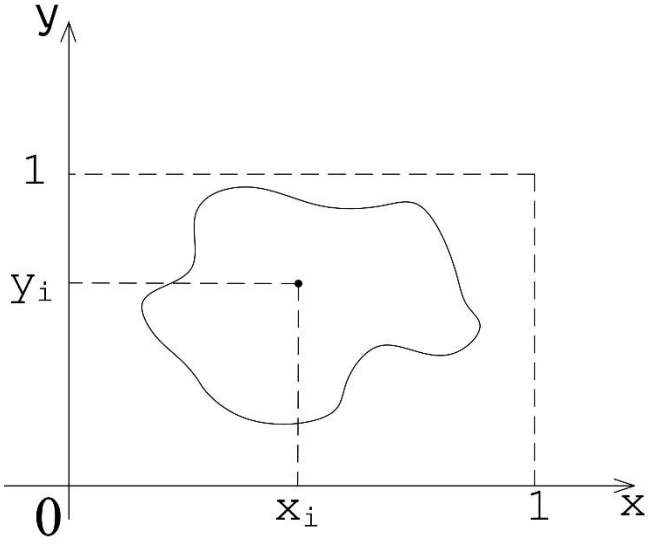
$$A = \lambda q = \lambda(1 - p_n)$$

Отказ: все n каналов заняты $p_{\text{отк}} = p_n$. Относительная пропускная способность системы равна вероятности того, что заявка будет принята к обслуживанию.
Параметр μ — это, как правило, функция технических характеристик объекта и средств реализаций (характеристики решаемых задач).
В простейшем случае, если время ввода-вывода каждого задания или процесса мало по сравнению со временем решения этого задания, принимают, что $t = \frac{1}{\mu}$, то есть среднее число операций, выполненных процессором, приходящихся на одно задание к среднему быстродействию процессора (речь идет о среднем времени).

Метод Монте-Карло (статистических испытаний)

На практике далеко не все процессы являются марковскими или сводящимися к марковским. В системах массового обслуживания поток заявок не всегда бывает пуассоновским (простейшим), реже наблюдается постоянное распределение времени обслуживания. Для произвольных потоков событий, переводящих систему из состояния в состояние, аналитические решения получены только для отдельных частных случаев. В общем случае применяют метод статистических испытаний (Монте-Карло).

Идея метода Монте-Карло: вместо того, чтобы описывать случайные явления с помощью аналитических зависимостей, производится моделирование случайного явления с помощью некоторой процедуры, которая дает «случайный» результат. Произведя такой розыгрыш очень большое количество раз, получаем статистический материал — множество реализаций случайного явления.



Суть метода:

1. Любым способом получаем два числа — x_i и y_i, подчиняющихся равномерному распределению на интервале $[0,1]$.
2. Считаем, что одно число определяют координату точки по Ox, второе — по Oy.
3. Анализируем, принадлежит ли точка заштрихованной поверхности. Если принадлежит, то в счетчик добавляем 1.
4. Процедура генерации двух случайных чисел с заданным законом распределения и проверка принадлежности точки поверхности повторяется n раз.

Площадь можно определить как отношение количества точек, попавших в область, к общему числу сгенерированных точек.

Точность метода: $\epsilon < \sqrt{\frac{1}{n}}$.

+ универсальность, обуславливающая возможность всестороннего статистического исследования объекта.

– для реализации нужны достаточно полные статистические сведения о параметрах;
– большой объем вычислений.

Уменьшение количества испытаний повышает скорость вычислений, но ухудшает их точность.

Немарковские случайные процессы, сводящиеся к марковским

Известно, что реальные процессы обладают последействиями и поэтому не являются марковскими. Иногда при исследовании таких процессов удается воспользоваться методами, разработанными для марковских цепей. К ним относят:

- метод разложения случайного процесса на фазы (метод псевдосостояний);
- метод вложенных цепей Маркова.

Метод псевдосостояний

Суть: состояния системы, потоки переходов из которых являются немарковскими, заменяются эквивалентной группой фиктивных состояний, потоки из которых уже являются марковскими. Созданная таким образом новая система статистически эквивалентна или близка к реальной системе.

Пример процессов которые можно точно свести к марковским относятся процессы, происходящие под воздействием потоков Эрланга. Сведение потока Эрланга k-ого порядка к пуассоновскому осуществляется введением k псевдосостояний. Интенсивность перехода между состояниями равна соответствующему параметру потока Эрланга. Полученный таким образом случайный процесс (он эквивалентен исходному) является марковским, так как интервалы времени нахождения его в различных состояниях подчиняются показательному закону.

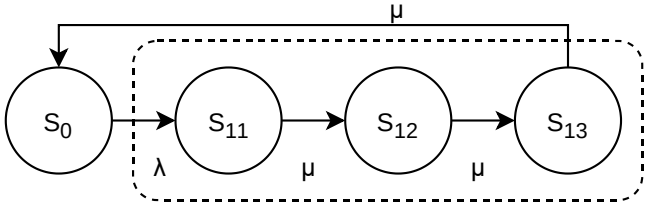
Пример:

Некоторые устройства выходят из строя с интенсивностью λ. Поток отказов — пуассоновский. После отказа устройство восстанавливается. Время восстановления распределено по закону Эрланга 3-го порядка. Найти предельные вероятности возможных состояний системы.

Система может принимать два возможных состояния:

- S_0 — устройство исправно;
- S_1 — устройство отказало и восстанавливается.

Переход из S_0 в S_1 — пуассоновский закон, наоборот — эрланговский.



Время восстановления можно представить в виде суммы трех времен, они распределены по показательному закону и имеют интенсивность μ.

Для установившегося состояния составляем уравнения Колмогорова и решаем полученную систему.

$$\begin{cases} p_0' = 0 = -\lambda p_0 + \mu p_{13} \\ p_{11}' = 0 = -\mu p_{11} + \lambda p_0 \\ p_{12}' = 0 = -\mu p_{12} + \mu p_{11} \\ p_{13}' = 0 = -\mu p_{13} + \mu p_{12} \\ p_0 + p_1 = 1 \\ p_1 = p_{11} + p_{12} + p_{13} \end{cases}$$

$$p_{13} = \frac{\lambda}{\mu} p_0, \; p_{11} = \frac{\lambda}{\mu} p_0$$

$$p_{12} = p_{11} = \frac{\lambda}{\mu} p_0$$

$$p_1 = \frac{3\lambda}{\mu} p_0$$

$$p_0 + \frac{3\lambda}{\mu} p_0 = 1 \Rightarrow p_0 = \frac{\mu}{\mu + 3\lambda}$$

$$p_1 = \frac{3\lambda}{\mu + 3\lambda}$$

Метод вложенных цепей Маркова

Вложенные марковские цепи образуются следующим образом. В исходном случайном процессе выбираются такие случайные процессы, в которых процесс является марковским. Эти моменты времени обычно являются случайными и зависят от свойств исходного процесса. Процесс исследуется только в этих точках

Частным случаем данного метода является полумарковский случайный процесс. Случайный процесс с конечным (счетным) множеством состояний называется *полумарковским*, если заданы вероятности перехода системы из состояния в состояние и распределение времени пребывания процесса в каждом состоянии в виде функции распределения или функции плотности распределения.

Способы получения последовательности случайных чисел

При имитационном моделировании сложных систем одним из основных вопросов является учет стохастических воздействий. Для этого способа моделирования нам необходимо большое число операций со случайными числами. Здесь характерна зависимость результатов от качества исходных (базовых) последовательностей.

На практике используется три основных способа генерации случайных чисел:

- аппаратный;
- табличный (файловый);
- алгоритмический (программный).

Аппаратный

Случайные числа вырабатываются специальной электронной приставкой — генератором случайных чисел. Как правило, в качестве нее служит одно из внешних устройств. Реализация этого способа не требует дополнительных вычислительных операций (необходима только операция обращения к устройству). В качестве физического эффекта чаще всего используются электрические шумы в полупроводниковых приборах.

+ Запас чисел не ограничен.
+ Расходуется очень мало вычислительных операций.
+ Не занимает место в памяти.
− Требуется периодическая проверка на случайность.
− Нельзя воспроизводить последовательности.
− Используются специальные устройства.

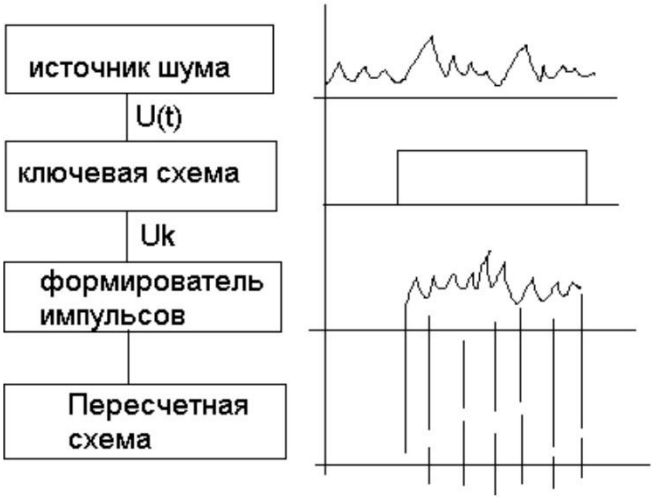


Рис. 7.1: Структурная схема аппаратного генератора случайных чисел

Табличный

Случайные числа оформляются в виде файла (таблицы).

+ Требуется однократная проверка на случайность.
+ Можно воспроизводить последовательности.
− Запас чисел ограничен.
− Занимает место в памяти.
− Требуется время на обращение.

Алгоритмический

Основан на специальных алгоритмах.

+ Однократная проверка на случайность.
+ Можно многократно воспроизводить последовательности псевдослучайных чисел.
+ Занимает очень мало места в памяти.
+ Не требует специальных устройств.
− Запас чисел ограничен периодом последовательности чисел.
− Требуются существенные затраты вычислительных ресурсов.

Алгоритмы генерации последовательности псевдослучайных чисел

Одним из первых способов получения последовательности псевдослучайных чисел было выделение значения дробной части у многочлена первой степени:

$$y_n = \text{Ent}(an + b)$$

Если значение n пробегает натуральный ряд чистел, то y_n – хаотично. При рациональном коэффициенте a множество y_n конечно, а при иррациональном — бесконечно и всюду плотно на интервале от 0 до 1.

Фон Нейман

Каждое последующее случайное число образуется возведением предыдущего в квадрат и отбрасывании цифр с обоих концов.

Лемер

Алгоритм генерирует последовательность чисел по следующей рекуррентной формуле:
$X_{n+1}=(a\cdot X_n+c) \bmod m$
Где:
Xn — текущее значение (начальное значение называется **зерном**, *seed*).
a — множитель, $0<a<m$. c — приращение, $0\le c<m$. m — модуль, $m>0$.
От отчаяния используют 2 или 3 различных генератора, смешивая их значения. Если разные генераторы независимы, то сумма их последовательностей обладает дисперсией, равной сумме дисперсий. *В современных системах обычно используют конгруэнтные генераторы по алгоритму, предложенному национальным бюро стандартов США, длина периода которого* 2^{24}.

Целочисленная арифметика

Берем бесконечный ряд Фибоначчи и, если брать только последнюю цифру каждого числа и использовать понятие бита переноса, то получается случайное число.
Для имитации равномерного распределения на интервале от a до b используется обратное преобразование функции плотности:

$$\frac{x-a}{b-a}=R$$

$$x=a+(b-a)R$$

Здесь R изменяется от 0 до 1.

Генерация случайных чисел с законом распределения отличным от равномерного

В основе построения программы, генерирующей случайные числа с законом распределения, отличным от равномерного, лежит **метод преобразования последовательности случайных чисел с равномерным законом распределения в последовательность с заданным**.

$$F(t)=\int_{-\infty}^{t} f(x)dx=R$$

Метод основан на утверждении, что случайная величина x, принимающая значения, равные корню уравнения выше, имеет плотность распределения $f(x)$. R — равномерно распределенная случайная величина на интервале от 0 до 1.
Значение случайной величины, распределенной по показательному закону, может быть вычислено следующим образом:

$$1-e^{-\lambda x}=R$$

$$x=-\frac{1}{\lambda}\ln(1-R)$$

Распределение Пуассона относится к дискретному с мат. ожиданием и дисперсией, равной $\lambda > 0$. Для генерирования пуассоновских переменных используется метод точек, в основе которого лежит генерирование случайной переменной R_i, равномерно распределенной на [0, 1], до тех пор, пока не будет выполнено следующее соотношение.

$$\prod_{i=0}^{x} R_i \geqslant e^{-\lambda} > \prod_{i=0}^{x+1} R_i$$

При получении случайной величины, функция распределения которой не позволяет найти решение уравнения в явной форме, можно произвести кусочно-линейную аппроксимацию, а затем вычислять приближенное значение корня. Кроме того, при получении случайных величин часто используют те или иные свойства распределения.

Распределение Эрланга определяется параметрами λ и k. При вычислении случайной величины воспользуемся тем, что поток Эрланга может быть получен прореживанием потока Пуассона k раз. Поэтому достаточно получить k значений случайной величины, распределенной по показательному закону, и усреднить их:

$$x = \frac{1}{k}\sum_{i=1}^{k}\left(-\frac{1}{\lambda}\ln(1-R_i)\right) = -\frac{1}{k\lambda}\sum_{i=1}^{k}\ln(1-R_i)$$

Нормально распределенная случайная величина может быть получена как сумма большого числа случайных величин, распределенных по одному и тому же закону и с одними и теми же параметрами:

$$x = \sigma_x\sqrt{\frac{12}{n}}\left(\sum_{i=1}^{n} R_i - \frac{n}{2}\right) + M_x$$

На практике берут n = 12.

Методика построения программной модели

Для разработки программной модели исходная система должна быть представлена как стохастическая система массового обслуживания. Это объясняется тем, что информация от внешней среды поступает в случайные моменты времени, длительность обработки различных типов информации может быть в общем случае различна. Следовательно, внешняя среда является как бы генератором сообщений, а комплекс вычислительных средств — обслуживающими устройствами.

БП – буферная память, ОА – обслуживающий аппарат, ПИ – программный имитатор, ИИ – источник информации, ПСС – программа сбора статистики, А – абонент.

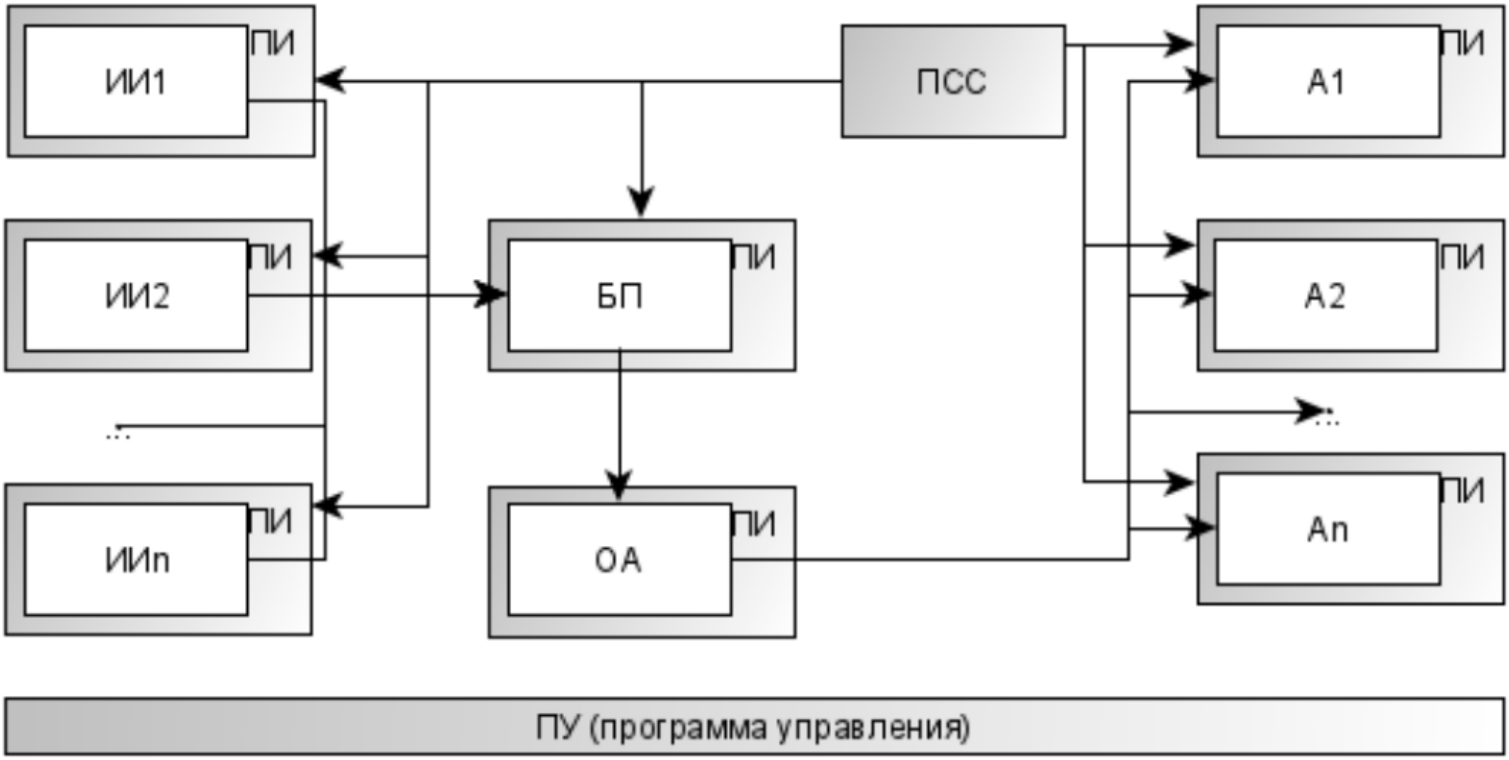


Рис. 9.1: Структурная схема программно-алгоритмического комплекса

ИИ выдают на вход БП независимо друг от друга сообщения. Закон появления сообщений произволен, но, как правило, формализован и задан.

В памяти сообщения записываются подряд и выбираются по одному на ОА по принципу FIFO. Длительность обработки одного сообщения также может быть случайна, но закон обработки должен быть задан. Быстродействие ОА ограничено, поэтому возникает очередь на обработку. Смысл программы синхронизации (управления) заключается в *протяжке* модельного времени.

Моделирование потока сообщений

Поток сообщений обычно моделируется моментами появления очередного сообщения в потоке, на которое мы продвигаем модельное время.

Таблица 9.1: Формулы генерации моментов времени

Распределение	**Момент времени t_i =**
Равномерное	$a + (b - a)R$
Экспоненциальное	$\frac{1}{\lambda}\ln(1-R)$
Нормальное	$\sigma_t\sqrt{\frac{12}{n}}\left(\sum_{i=1}^{n} R_i - \frac{n}{2}\right) + M_x$
Эрланга	$-\frac{1}{k\lambda}\sum_{i=1}^{k}\ln(1-R_i)$

Моделирование работы обслуживающего аппарата (ОА)

Программа, моделирующая работу обслуживающего аппарата, — это набор функций, вырабатывающих случайные отрезки времени, соответствующие времени обслуживания.

$$T_{\text{обр}} = M_x + \left(\sum_{i=1}^{12} R_i - 6 \right) \delta_x, \text{ где } X \sim N(M_x, \delta_x)$$

R – псевдослучайное число от 0 до 1
T_обр - время обработки очередного сообщения

Моделирование работы абонентов (А)

Абонент может рассматриваться как обслуживающий аппарат, поток информации на который поступает от процессора.
Для имитации работы необходимо написать программу выработки длительности обслуживания. Кроме того, абонент сам может быть источником заявок на те или иные ресурсы вычислительной системы. Эти заявки могут имитироваться с помощью генератора сообщений по заданному закону.

Моделирование работы буферной памяти (БП)

Моделирование начинаем с разработки концептуальной модели.
Память — электромеханическое устройство, предназначенное для хранения, записи и чтения информации.
Блок БП должен производить запись и считывание числа, выдавать сигнал переполнения и отсутствия данных, в любой момент модельного времени располагать сведениями о количестве находящихся в блоке требований.
Сама запоминающая среда в простейшем случае имитируется одномерным массивом, размер которого определяется объемом. Каждый элемент этого массива может быть либо свободен, либо занят, и в этом случае в качестве эквивалента требований ему присваивается значение времени появления этого требования.

Рис. 9.2: Структурная схема памяти

Здесь:

- P — массив обращения;

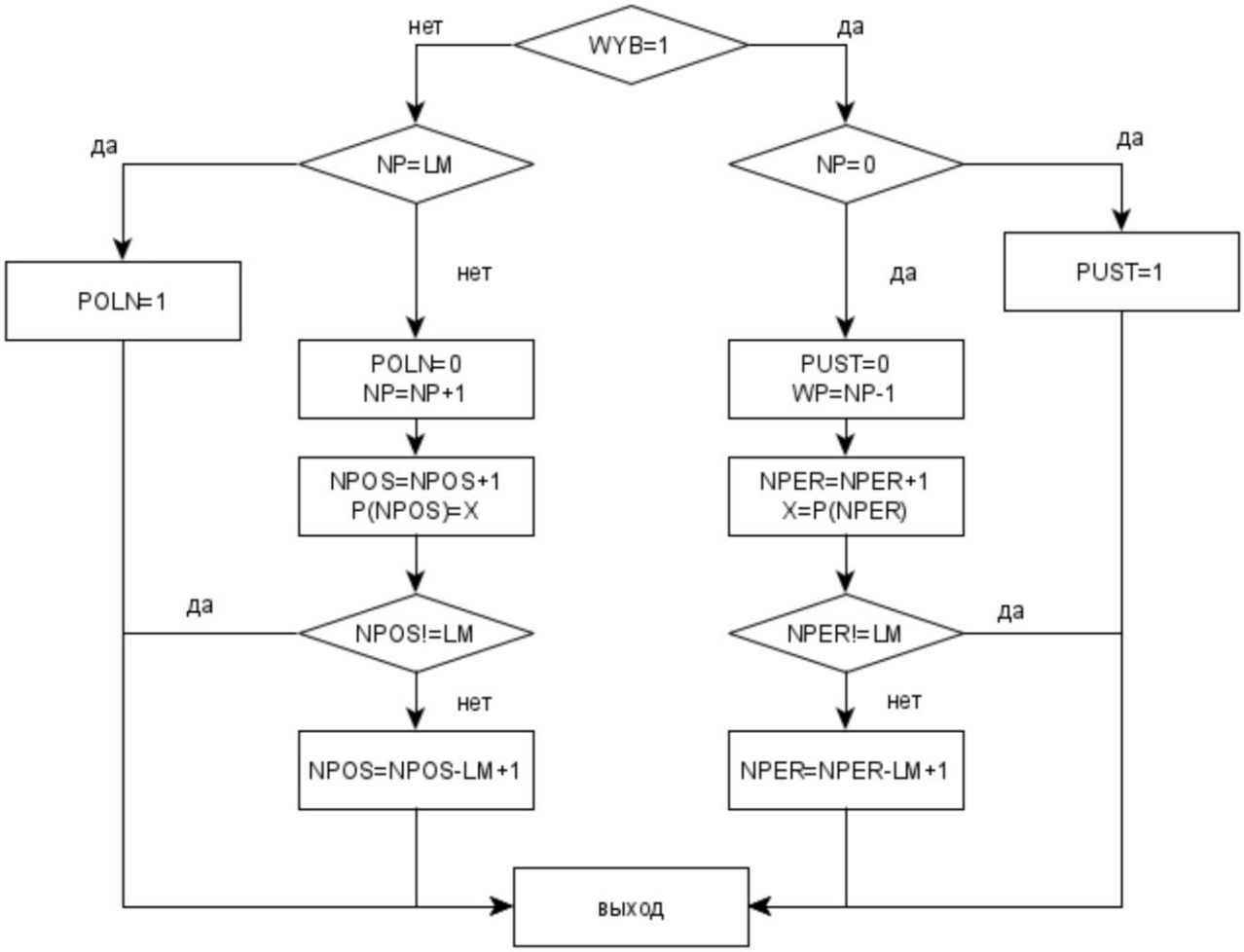


Рис. 9.3: Схема алгоритма буферной памяти

- WYB — признак обращения к буферной памяти (0 — режим записи, 1 — режим выбора сообщения);
- NP — число сообщений в памяти;
- LM — объем буферной памяти;
- NPOS — номер последнего сообщения в памяти;
- NPER — номер первого сообщения в памяти;
- POLN — признак переполнения (1 — нет свободных ячеек);
- X — ячейка для сообщения;
- PUST — признак отсутствия сообщения (1 — в памяти нет сообщения).

Моделирование программы сбора статистики (БС)

Задача блока статистики (БС) заключается в накоплении численных значений, необходимых для вычисления статистических оценок заданных параметров работы моделируемой вычислительной системы.

При моделировании простейшей СМО интерес представляет среднее время ожидания в очереди. Для каждого сообщения время ожидания в очереди равно разности между моментом времени, когда оно было выбрано на обработку, и моментом, когда оно пришло в систему от источника информации.

Коэффициент загрузки обслуживающего аппарата определяется как отношение времени работы обслуживающего аппарата к общему времени моделирования.

Подсчитав число потерянных сообщений, можно определить **вероятность отказа** или потери сообщения: $N_{потерь} / (N_{потерь} + N_{обработано})$

Управляющая программа имитационной модели

Если программа-имитатор от источника информации обслуживающего аппарата буферной памяти отображает работу отдельных устройств, то управляющая программа *имитирует алгоритм взаимодействия всех устройств системы.*

Управляющая программа реализуется по следующим принципам.

Принцип Δt

Принцип Δt заключается в последовательном анализе состояний всех блоков системы в момент времени $t + \Delta t$ по заданному состоянию блоков в момент времени t. При этом новое состояние определяется в соответствии с их алгоритмическим описанием с учетом случайных факторов, задаваемых распределениями вероятности. В результате этого решения проводится анализ, позволяющий определить, какие общесистемные события должны имитироваться в программной модели на данный момент времени.

Основной недостаток принципа Δt — значительные затраты машинного времени на анализ и контроль правильности функционирования всей системы. При недостаточно малом Δt появляется опасность пропуска отдельных событий в системе, что исключает возможность получения правильных результатов при моделировании.

Событийный принцип

Характерное свойство моделируемых систем обработки информации — то, что состояния отдельных устройств изменяются в дискретные моменты времени, совпадающие с моментами поступления сообщений в систему, окончания решения той или иной задачи,возникновения аварийных сигналов. Поэтому моделирование и продвижение текущего времени в системе удобно производить, используя *событийный принцип*, при реализации которого *состояния всех блоков имитационной (программной) модели анализируется лишь в момент появления какого-либо события*. Момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность ближайшего изменения состояния каждого из блоков системы.

Методика реализации событийной модели

Обозначения:

- t_{11}, t_{12} — моменты появления сообщений на выходе источника информа-

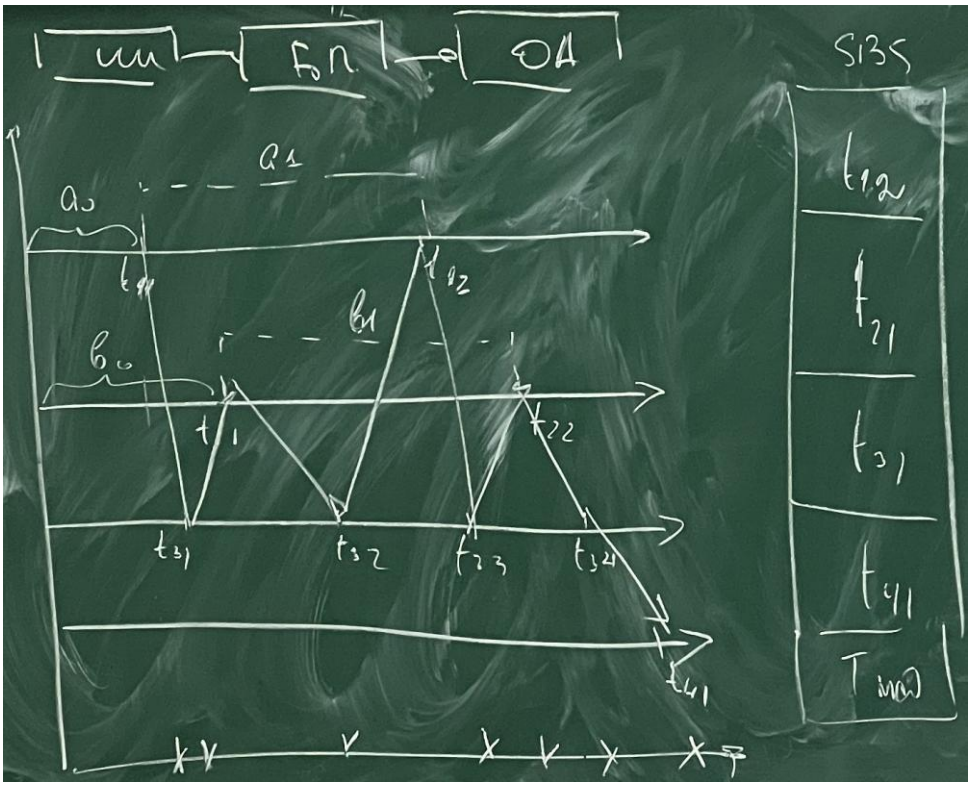


Рис. 10.1: Схема событийного принципа

ции;

- b_1 — время обслуживания первого сообщения;
- t_{3i} — моменты времени сбора статистики; • t_{41} — момент времени окончания моделирования;
- SBS — список будущих событий.

1. Для всех активных блоков (блоков, порождающих события) заводят свой элемент в одномерном массиве — списке будущих событий (СБС).
2. В качестве подготовительной операции в СБС заносят время ближайшего события от любого активного блока. Активизируя программный имитатор источника, вырабатывают псевдослучайную величину a_0, определяющую момент появления первого сообщения t_{11}, и эту величину заносят в СБС. Активизируя имитатор обслуживающего аппарата, вырабатывают псевдослучайную величину b_0, определяющую момент времени t_{21}, которую также заносят в СБС. Момент времени t_{31} (момент первого сбора статистики) определяется равным стандартному шагу сбору статистики и также заносится в СБС. В этот же список заносят время окончания моделирования t_{41}. На этом подготовительный этап заканчивается, и далее начинается «протяжка» модельного времени в соответствии с алгоритмом.
3. В списке будущих событий определяем минимальное значение и его порядковый номер.
4. Реализуются события, порождаемые блоком со временем t_{11}. Реализация этого события заключается в том, что само сообщение записывается в буферную память, и с помощью программного имитатора источника вырабатывается момент появления следующего сообщения t_{12}. Это время помещается в список будущих событий вместо t_{11}. Повторяется до достижения времени окончания моделирования.

Способ борьбы с недостатками – комбинаторный метод
Во многих реальных системах распределение событий далеко не однородно (события группируются во времени). Образование групп связано с наступлением какого-либо

«значимого» события, которое начинает определенную последовательность действий с соответствующими событиями, имеющими высокую плотность на близлежащем временном интервале. Такой интервал называется ***пиковым***, а распределение событий — ***квазисинхронным***.

Пример: цифровая сеть, в которой синхронизирующие сигналы переключают большое количество триггеров.

Этот алгоритм был разработан для специальных дискретных систем, в которых присутствует квазисинхронное распределение событий. Особенностью алгоритма является автоматическая адаптация к распределению событий. Метод реализуется так, что на пиковых интервалах он приближается к методу Δt, а вне пиковых — к событийному методу с большим шагом времени.

Алгоритм основан на использовании иерархической структуры циркулярных списков. Список первого уровня содержит n_1 элементов и описывает планируемое событие в пиковых интервалах. Число n_1 представляет собой разбиение пикового интервала на более мелкие участки, с каждым из которых связан список событий, произошедших за этот интервал. Списки второго уровня и выше являются масштабирующими списками, количество элементов которых равно константному значению n_2, которое характеризует коэффициент масштабирования временных интервалов.

Алгоритм протяжки модельного времени заключается в последовательном поиске непустых элементов в самом верхнем циркулярном списке с большим шагом и дальнейшим спуском на нижние уровни, тем самым уменьшая шаг протяжки модельного времени.

Классификация СМО

СМО классифицируется по следующим признакам

- закону распределения входного потока
- число обслуживающих приборов
- закону распределения времени (в обслуживающих приборах)
- числу мест в очереди
- дисциплине обслуживания

Для краткости в литературе часто записывается СМО которое характеризуется следующими характеристиками A/B/C/D/E, где

A — закон распределения интервалов времени между поступлением заявок, наиболее часто используемые (М-экспоненциальная, Е-эрланговская, Н-гиперэкспоненциальная, Г-гамма, D-детерминированное распределение) для обозначения произвольного характера распределения используется G.

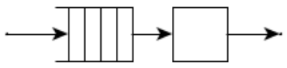
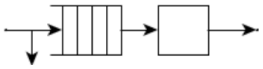
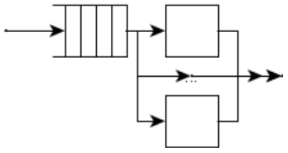
B – закон распределения времени обслуживания в приборах СМО

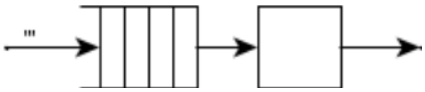
C – число обслуживающих приборов (для одноканальных записывается 1, для многоканальных в общем случае l)

D – число мест в очереди (если неограничено, то может отсутствовать) для конечного числа мест в очереди принято обозначение либо r либо m.

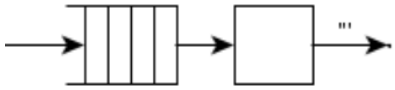
E — Дисциплина обслуживания наиболее часто используются следующие варианты (fifo, lifo, random) fifo если не указано

Для моделирования вычислительных систем и сетей чаще всего используются следующие типы СМО

1. Одноканальная СМО с ожиданием. Представляет собой один обслуживающий аппарат с бесконечной очередью. Является наиболее распространенным формализмом при моделировании практически любых узлов сетей и вычислительных систем
   G/G/1 
2. Одноканальные СМО с потерями. Представляют собой один обслуживающий аппарат с конечным числом мест в очереди Используются при моделировании каналов передачи.
   G/G/1/r 
3. Многоканальное СМО с ожиданием. Представляют собой несколько параллельно работающих обслуживающих приборов с общей бесконечной очередью. Используются при моделировании групп абонентских терминалов работающих в диалоговом режиме.
   
   G/G/l
4. Многоканальное СМО с потерями. Представляют собой несколько параллельно работающих обслуживающих приборов с общей очередью число мест в которой ограничено. Используются для моделирования каналов связи.
   G/G/l/r
5. Одноканальные СМО с групповым поступлением заявок. Представляют собой один обслуживающий прибор с бесконечной очередью. Перед обслуживанием заявки группируются в пакеты по определенному правилу.

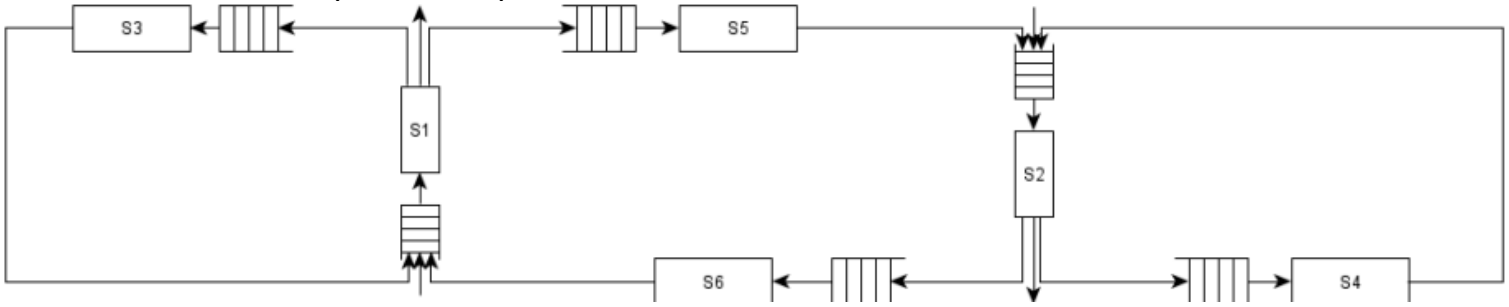
Gr/G/1 

6. Одноканальные СМО с групповым обслуживанием заявок. Представляют собой один обслуживающих прибор с бесконечной очередью. Заявки обслуживаются пакетами. Два последних типа СМО могут использоваться для моделирования центров коммутации.

G/Gr/1 

Вычислительные сети могут быть в общем виде представлены в виде **сети массового обслуживания**. Различают открытые, замкнутые и смешанные сети.

Открытая сеть – сеть массового обслуживания состоящая из m узлов причем хотя бы в один из узлов сети поступает извне входящий поток заявок и имеется сток заявок. Для открытых сетей характерным является то что интенсивность поступления заявок в сеть не зависит от состояния сети. Такие сети как правило используются для вычислительных сетей или систем работающих в не оперативном режиме.



S1 S2 – моделируют работу узлов комутации

S3 S4 – серверов, S5 S6 – межузловых каналов

В сети циркулирует 2 потока заявок, каждая заявка поступает на вход соответствующего узла коммутации, где определяется место ее обработки. Затем заявка передается на «свой» сервер или по каналу связи на соседний, где и обслуживается. После чего возвращается к источнику и покидает сеть.

Замкнутая сеть -- сеть массового обслуживания с множеством узлов без источника и истока, в которой циркулирует постоянное число заявок.

Смешанная сеть -- сеть массового обслуживания в которой циркулирует несколько различных типов заявок, причем относительно одних типов сеть замкнута, относительно других открыта. С помощью смешанных сетей массового обслуживания моделируется части абонентов котторые работают в диалоговомм а часть в непрерывном режиме (неоперативность).

Языки моделирования вычислительных систем

Алгоритмические языки при моделировании систем служат вспомогательным аппаратом разработки компьютерной реализации и анализа характеристик систем. Отсюда огромнейшее значение имеет вопрос правильного выбора языка.

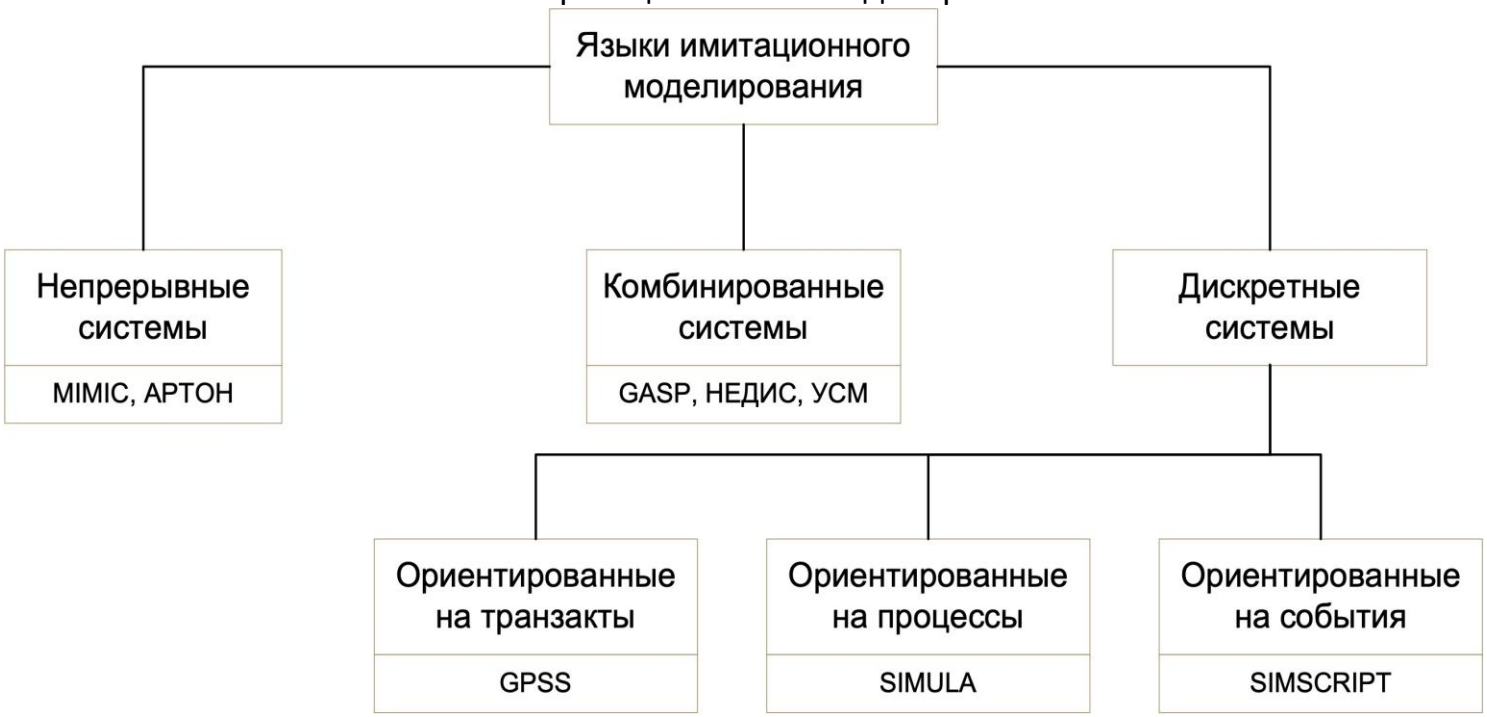
Язык моделирования, как и любой другой язык, имеет тщательно разработанную систему абстракций. Причем эти абстракции должны быть полностью направлены на анализ и контроль правильности функционирования объекта моделирования.

Качество языков моделирования характеризуется:

- удобством описания процесса функционирования;
- удобством ввода исходных данных, варьирования структуры, алгоритмов, задания параметров модели;
- анализом вывода результатов моделирования;
- простотой отладки и контроля работы моделирующей программы;
- доступностью восприятия и использования языка.

Все современные языки моделирования определяют поведение систем во времени с помощью событийного алгоритма.

Классификация языков моделирования



Непрерывное представление систем сводится к представлению дифференциальных уравнений, с помощью которых устанавливается связь вход-выход. Если выходные переменные модели принимают дискретные значения, то уравнения являются разностными.

Комбинированный подход предполагает что в системе могут происходить 2 типа событий:

- события, зависящие от состояния системы.
- события, зависящие от времени

Состояние системы описывается набором переменных, некоторые из которых меняются непрерывно. Пользователь такого языка описывает условия наступления событий, законы изменения непрерывных величин, правила перехода от одного состояния к другому.

Дискретные системы используются в основном для имитационного моделирования, при этом используются разные способы описания динамического поведения системы.

Формальное описание динамики моделируемого объекта

Будем считать, что любая работа в системе совершается путем выполнения *активностей*. Принимают, что активность является наименьшей единицей работы. Ее рассматривают как единый дискретный шаг.

Активность — это единый динамический объект, указывающий на совершение ед. работы.

Процесс — это логически связанный набор активностей. Например, запись в память.

Событие — это мгновенное изменение состояния объекта.

Рассмотренные объекты (активности, процессы , события) являются конструктивными элементами для динамического описания поведения системы. На их основе строятся языки моделирования. В то время, когда динамическое поведение системы формируется в результате выполнения большого числа взаимодействующих процессов, сами эти процессы образуют небольшое число классов. Следовательно, чтобы описать поведение систем, достаточно описать поведение каждого класса процессов и задать значения атрибутов для конкретных процессов.

Построение модели состоит из двух основных задач.

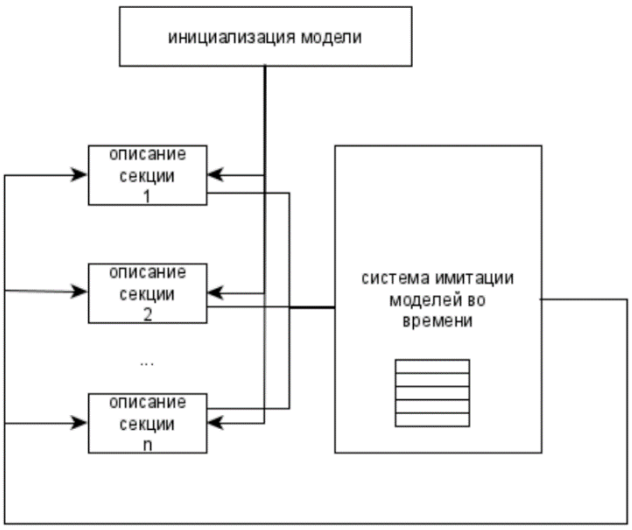
1. Описание правил, задающих виды процессов, происходящих в системе.
2. Описание значений атрибутов таких процессов или правил генерации этих значений. При этом система описывается на определенном уровне детализации в терминах множества описаний процессов, каждый из которых включает множество правил и условий возбуждения активностей.

В реальной системе совместно выполняется несколько активностей, причем как связанных с данным процессом так и независимых. Имитация этих действий на модели проходит в строгой последовательности в соответствие с алгоритмом.

Модель служит для отображения временного поведения системы, поэтому даже языки дискретного моделирования должны обладать средствами отображения времени.

SIMSCRIPT

Язык ориентированный на события. Моделирующая программа организована в виде секций событий. Событие состоит из набора операций, которые выполняются после завершения активности. Выполнение этих процедур или секций синхронизируется во времени списком будущих событий.



Инициализация модели -> описание секции N -> Список будущих событий -> в описание секции обратно

SIMULA

Язык ориентированный на процессы. Моделирующая программа организуется в виде набора описаний процессов, каждый из которых описывает один класс. Описание процесса устанавливает атрибуты активности. Протяжка модельного времени осуществляется с помощью списка будущих событий.

GPSS

Можно отнести к группе языков, ориентированных на процессы. Его используют для описания пространственного движения объектов. Такие динамические объекты в языке GPSS называются *транзактами* и представляют собой элементы потока.
Программа на GPSS представляет диаграмму движения транзактов.

Сравнение универсальных и специализированных языков моделирования

Универсальные:

+ Минимум ограничений на выходной формат
+ Широкое распространение
– Значительное время, затрачиваемое на программирование
– Значительное время, затрачиваемое на отладку

Специализированные:

+ Меньше затраты времени на программирование
+ Более эффективные методы выявления ошибок
+ Краткость, точность понятий, характеризующих имитируемые конструкции
+ Возможность заранее строить стандартные блоки, которые могут использоваться в любой имитационной модели
+ Автоматическое формирование определенных типов данных, необходимых именно в процессе имитационного моделирования Удобство накопления и представления выходной информации
+ Эффективное использование ресурсов
– Необходимость точно придерживаться ограничений на форматы данных
– Меньшая гибкость модели

Возможности программного обеспечивания и имитационного моделирования:

1) анимация и динамическая графика:
   a) представление сути имитационного моделирования;
   b) отладка моделируемой программы;
   c) обучение обслуживающего персонала.
2) статистические возможности:
   a) генераторы псевдослучайных чисел.

Существует два типа анимаций:

- совместная (осуществляется во время прогонки модели);
- реальная.

Объектно-ориентированное моделирование

Система формируется из объектов
которые взамиодействуют друг с другом. Объекты имеют данные и содежрат методы.
Данные описывают состояние объекта, методы взаимодействуют с объектом.

+ возможность повторного использования объектов и удобный способ их изменения;
+ реализация иерархического подхода;
+ упрощение изменения модели;
+ возможность декомпозиции задачи на подзадачи.
– трудность изучения.

ЛР. Моделируем информационный центр. Смоделировать процесс обработки 300 запросов. Определить вероятность отказа.

Рисунок 1 – Концептуальная модель

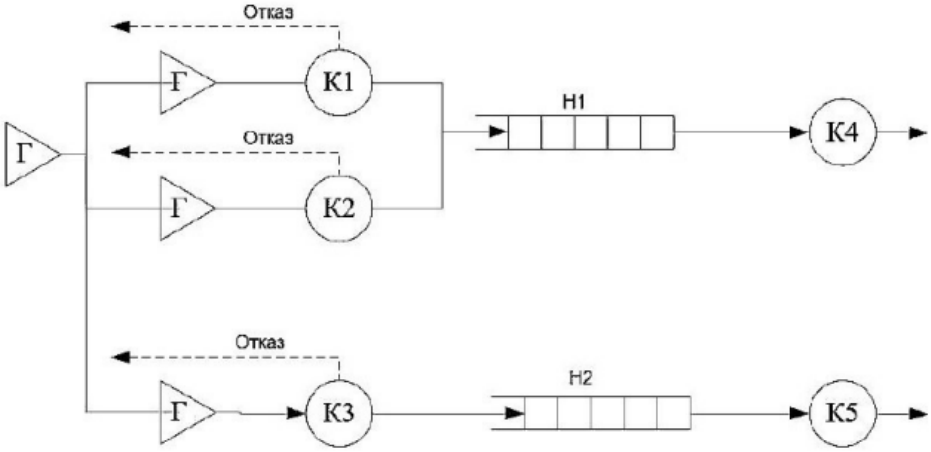


Рисунок 2 – Модель в терминологии СМО

Эндогенные переменные:
— Время обработки задания i-м оператором
— Время решения задания j-м компьютером
Экзогенные переменные:
— Число обслуживаемых клиентов
— Число клиентов, получивших отказ
Вероятность получения отказа:

$$P_{\text{отказа}} = \frac{N_{\text{rejected}}}{N_{\text{total}}}$$

Nrejected – количество клиентов получивших отказ,
Ntotal – количество обработанных клиентов и получивших отказ в сумме.

GPSS

GPSS: (General Purpose Systems Simulator — общецелевая система моделирования) — язык программирования, используемый для имитационного моделирования систем (в основном, массового обслуживания). Можно отнести к группе языков, ориентированных на процессы. Его используют для описания пространственного движения объектов. Такие динамические объекты в языке GPSS называются *транзактами* и представляют собой элементы потока. Программа на GPSS представляет диаграмму движения транзактов.

Транзакты - заявки (сообщения) Динамические объекты, которые представляют собой единицы исследуемых потоков и производят ряд определенных действий, продвигаясь по фиксированной структуре, представляющей собой совокупность объектов других категорий.

Операторы (блоки) GPSS имеют следующий формат:

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

Блоки

Блоки, используются для описания функций и управляют движением транзактов.

Аналогия

- Транзакт = деталь на конвейере
- Блоки = станки/операции на конвейере
- GPSS-модель = весь технологический маршрут

Практически все изменения состояния модели возникают в результате поступления транзактов в соответствующие блоки. После выполнения блока транзакт либо движется к следующему блоку, либо задерживается, либо уничтожается. Блоки измениют атрибуты транзакта.

Объекты

— это сущности модели, чье состояние меняется. Блоки воздействуют на эти объекты.

Классификация блоков GPSS

Блоки, используются для описания функций и управляют движением транзактов. У каждого блока имеется 2 стандартных числовых атрибута(СЧА):

- счетчик входа в блок. Номер транзакта, входящего в данный блок.
- Общий счетчик транзактов, поступивших в блок с начального момента моделирования.

1. **Блоки осуществляющие модификацию атрибутов транзакта**
    - Временная задержка ADVANCE
    - Генерация и уничтожение транзактов GENERATE TERMINATE SPLIT ASSEMBLE
    - Синхронизация движения нескольких транзактов MATCH GATHER
    - Изменение параметров транзакции ASSIGN INDEX MARK
    - Изменение приоритета PRIORITY
2. **Блоки, изменяющие последовательность передвижения транзактов**, т.е. блоки передачи управления. TRANSFER, LOOP, TEST, GATE
3. **Блоки, связанные с группирующей категорией**. JOIN, REMOVE, EXEMINE, SCAN, ALTER
4. **Блоки, сохраняющие значения** для дальнейшего использования. SAVEVALUE
5. **Блоки, организующие использование объектов аппаратного устройства**. SEIZE - RELEASE, PREEMPT - RETURN
6. **Специальные блоки** HELP, TRACE, UNTRACE, PRINT, ...?
7. **Блоки для организации цепей** LINK, UNLINK
8. **Вспомогательные блоки** LOAD, SAVE

Блоки создания и уничтожения транзактов

GENERATE A,B,C,D,E

A - время между поступлением транзактов
B - модификатор интервала a
C - начальная задержка
D - число порождаемых транзактов
E - приоритет транзакта

Прим. GENERATE 10,5 — каждые 10±5 единиц времени

TERMINATE A -- Уничтожает странзакт с изменением счетчика на A (если A нет, то счетчик не изменяется)

ASSEMBLE A -- число A транзактов объединяется в ансамбль, и в результате после набора числа A транзактов выходит первый его транзакт, а остальные уничтожаются.

Прим. ASSEMBLE 5 — ждать 5 транзактов, собрать в 1

SPLIT A,B,C,D -- создат A копий транзакта, входящих в блок B, записываемых в параметры C, с числом параметров D

Прим. SPLIT 1,BRANCH — создать 1 копию и отправить на BRANCH

Блоки синхронизации транзактов

MATCH A -- Синхронизация двух транзактов

Прим. MATCH PHASE — ждать парный транзакт с таким же параметром

GATHER A -- Сборка транзактов из разных веток

Прим. GATHER 3 — ждать 3 транзакта, затем выпустить все

Блоки работы с очередями

QUEUE A,B -- Встать в очередь A (увеличить ее длину на B)

DEPART A,B -- Покинуть очередь A (уменьшить ее длину на B)

Блоки работы с устройствами

SEIZE A -- Захватить устройство (если оно свободно)

RELEASE -- Освободить устройство

ADVANCE A,B -- Задержка транзакта на время. Прим. ADVANCE 15,3 — задержка 15±3

PREEMPT A,B,C,D,E -- Захватить устройство с вытеснением другого захваченного транзакта

- A — имя устройства
- B — приоритет прерывания (необязательно)
- C — параметр для сохранения времени (необязательно)
- D — параметр для сохранения состояния (необязательно)
- E — параметр для сохранения причины (необязательно)

RETURN A -- Вернуть вытесненное устройство

Управляющие блоки

START 300 ; Остановиться после 300 срабатываний TERMINATE 1

START A, B, C, D

- A -- целое число, задающее начальное значение счетчика завершений (количество заявок) .

- B может быть записано NP – признак подавления формирования отчета по завершении моделирования. Если пусто, то формируется отчет со стандартной статистической информацией обо всех объектах модели (как в лабах).
- C -- не используется,
- D может содержать 1 для включения в отчет списков текущих и будущих событий. Если пусто, то выдача в отчет содержимого этих списков не производится.

SIMULATE A

- A — максимальное время выполнения в минутах

Объекты запоминающей категории, ячейки.

SAVEVALUE A,B,C

A – имя ячейки. Если после A стоит знак + или -, то значение поля B прибавляется или вычитается из текущего содержимого ячейки A. Если знак не указан, то значение поля B присваивается ячейке A.

B – присваиваемое значение,

C – тип ячейки. Поле C определяет тип ячейки и может принимать значения: XH – полусловная, XF – полнословная, XL – с плавающей точкой. Default = XF

Начальное значение ячейки по умолчанию равно нулю. Для изменения начального значения применяется оператор инициализации

INITIAL Ячейка1, Значение1,…, Ячейка К, ЗначениеК

Изменение параметров транзакции

ASSIGN A,B,C

- A — номер параметра (1, 2, 3... или специальные символы)
- B — значение для добавления или замены
- C — номер модификатора-функции (необязательно)

Прим.

```
ASSIGN 1+,5      ; P1 = P1 + 5
ASSIGN 2-,3      ; P2 = P2 - 3
ASSIGN 1,0,PH1;  P1 = P1/2
```

MARK A

Где A — номер параметра для записи времени. Используется для измерения интервалов времени

```
MARK    1        ; P1 = время создания транзакта
…
MARK    2        ; P2 = время окончания обслуживания
ASSIGN 3,P2-P1    ; P3 = общее время в системе (P2 - P1)
```

PRIORITY 100 ; Установить приоритет = 100

Table, tabulate

TABLE и TABULATE — это инструмент сбора статистики и построения гистограмм в GPSS.

СОЗДАНИЕ ТАБЛИЦЫ (**TABLE**)

ИмяТаблицы TABLE Аргумент, НижняяГраница, Ширина, КоличествоИнтервалов, Режим

ЗАПОЛНЕНИЕ ТАБЛИЦЫ (**TABULATE**)
TABULATE ИмяТаблицы, Вес
Вес — это множитель, который указывает, сколько раз данное значение должно быть занесено в таблицу.

Отладка в GPSS

Для пошагового выполнения модели с целью ее отладки можно воспользоваться командой STEP A(выполнить шаг).
A -- задает количество входов активного транзакта в блоки, которое производится при каждом выполнении команды. Обычно этот операнд равен 1, и каждое выполнение команды STEP приводит к продвижению активного транзакта к следующему блоку. Отладку с использованием команды STEP удобно проводить, находясь в окне блоков. Для продолжения моделирования после прерывания следует ввести в командную строку команду CONTINUE (продолжить).

Для наблюдения за процессом моделирования используются 9 графических окон.

1. Блоки -- Движение транзактов между блоками, Текущее состояние каждого блока (активен/не активен), Значения Wn (кто сейчас в блоке) и Nn (сколько всего вошло)
2. Выражения -- Значения любых выражений в реальном времени
3. Устройства
4. Логические ключи
5. Матрица
6. График
7. Очереди
8. Ячейки памяти
9. Таблицы

```
            GENERATE   (UNIFORM(1, 2, 8))
start_queue QUEUE      QUEUE1              ; Поставить в очередь
            SEIZE      OP1                 ; Захват OP1
            DEPART     QUEUE1              ; Извлечь из очереди
            ADVANCE    (UNIFORM(1, 2, 5))
; (EXPONENTIAL(1, 0, 10) ) ; scale = 1/lambda
; ( NORMAL (1,9,2) )   ; Задержка (нормальное распределение m=10, d=2)
; (POISSON(1, lambda)) ;
; UNIFORM(stream, min, max) ;
            RELEASE    OP1                 ; Освобождение OP1
            TRANSFER   0.8,repeat,finish

repeat      SAVEVALUE  REPEAT_COUNT+,1
            TRANSFER   ,start_queue
finish      SAVEVALUE  SERVED+,1
            TERMINATE  1
            START      1000
```

```
        GENERATE        10,2,,          ; Бесконечный генератор заявок каждые 10+-2

op1     GATE NU         OPERATOR1,op2   ; Если OPERATOR1 занят, то переход к oper2
        SEIZE           OPERATOR1       ; Захват OPERATOR1
        ADVANCE         20,5            ; Задержка 20+-5
        RELEASE         OPERATOR1       ; Освободить OPERATOR1
        TRANSFER        ,proc1          ; Передать заявку в блок proc1

op2     GATE NU         OPERATOR2,op3   ; Если OPERATOR2 занят, то переход к oper3
        SEIZE           OPERATOR2       ; Захват OPERATOR2
        ADVANCE         40,10           ; Задержка заявки
        RELEASE         OPERATOR2       ; Освободить OPERATOR2
        TRANSFER        ,proc1          ; Передать заявку в блок proc1

op3     GATE NU         OPERATOR3,reject ; Если OPERATOR3 занят, то переход к reject
        SEIZE           OPERATOR3       ; Захват OPERATOR3
        ADVANCE         40,20           ; Задержка заявки
        RELEASE         OPERATOR3       ; Освободить OPERATOR3
        TRANSFER        ,proc2          ; Передать заявку в блок proc2

proc1   QUEUE           QUEUE_PROC1     ; Поставить в очередь
        SEIZE           PROCESSOR1      ; Захват PROCESSOR1
        DEPART          QUEUE_PROC1     ; Извлечь из очереди
        ADVANCE         15              ; Задержка заявки
        RELEASE         PROCESSOR1      ; Освободить PROCESSOR1
        TRANSFER        ,serv
proc2   QUEUE           QUEUE_PROC2     ; Поставить в очередь
        SEIZE           PROCESSOR2      ; Захват PROCESSOR2
        DEPART          QUEUE_PROC2     ; Извлечь из очереди
        ADVANCE         30              ; Задержка заявки
        RELEASE         PROCESSOR2      ; Освободить PROCESSOR2
        TRANSFER        ,serv

serv    SAVEVALUE       SERVED+,1       ; Увеличить счетчик SERVED
        SAVEVALUE       TOTAL+,1        ; Увеличить счетчик LOST
        TERMINATE       1               ; Уничтожить заявку и уменьшить счетчик

reject  SAVEVALUE       LOST+,1         ; Увеличить счетчик LOST
        SAVEVALUE       TOTAL+,1        ; Увеличить счетчик LOST
  ;SAVEVALUE RES,LOST/TOTAL      ; Увеличить счетчик LOST
        TERMINATE                       ; Уничтожить заявку БЕЗ уменьшения счетчика

        START           300             ; Остановиться после 300 срабатываний TERMINATE 1
```