

# Моделирование

7 семестр

Обновлено 5 января 2024 г.

## Содержание

<b>Лекция №1 (11.09.2023)</b>	<b>5</b>
Философские аспекты моделирования . . . . .	5
Классификация видов моделирования . . . . .	6
 <b>Лекция №2 (18.09.2023)</b>	<b>9</b>
Виды имитационного моделирования . . . . .	9
Агентное моделирование . . . . .	9
Дискретно-событийное моделирование . . . . .	9
Системная динамика . . . . .	9
Технические средства имитационного моделирования . . . . .	10
 <b>Лекция №3 (25.09.2023)</b>	<b>12</b>
Гибридные вычислительные машины . . . . .	12
Основы теории моделирования . . . . .	15
Типовые математические схемы . . . . .	17
 <b>Лекция №4 (02.10.2023)</b>	<b>18</b>
Характеристики сложных систем . . . . .	18
Последовательность разработки и компьютерной реализации . . . . .	19
Основные этапы моделирования больших систем . . . . .	20
Первый этап . . . . .	20
Второй этап . . . . .	20
Третий этап . . . . .	21
Классы ошибок . . . . .	21
 <b>Лекция №5 (09.10.2023)</b>	<b>23</b>
Лабораторная работа №1 . . . . .	23
Моделирование на системном уровне . . . . .	23
 <b>Лекция №6 (16.10.2023)</b>	<b>29</b>
Лабораторная работа №2 . . . . .	33
 <b>Лекция №7 (23.10.2023)</b>	<b>34</b>

Метод Монте-Карло (статистических испытаний) . . . . .	34
Способы получения последовательности случайных чисел . . . . .	35
Аппаратный . . . . .	35
Табличный . . . . .	36
Алгоритмический . . . . .	37
Алгоритмический способ получения случайных чисел . . . . .	37
<b>Лекция №8 (30.10.2023)</b>	<b>39</b>
Немарковские случайные процессы, сводящиеся к марковским . . . . .	39
Метод псевдосостояний . . . . .	39
Лабораторная работа №3 . . . . .	41
Метод вложенных цепей Маркова . . . . .	41
Алгоритмы генерации последовательности псевдослучайных чисел . .	41
Фон Нейман . . . . .	42
Лемер . . . . .	42
Целочисленная арифметика . . . . .	42
<b>Лекция №9 (13.11.2023)</b>	<b>45</b>
Методика построения программной модели . . . . .	45
Моделирование потока сообщений . . . . .	46
Моделирование работы обслуживающего аппарата . . . . .	46
Моделирование работы абонентов . . . . .	46
Моделирование работы буферной памяти . . . . .	47
Моделирование программы сбора статистики . . . . .	48
<b>Лекция №10 (20.11.2023)</b>	<b>50</b>
Управляющая программа имитационной модели . . . . .	50
Принцип $\Delta t$ . . . . .	50
Событийный принцип . . . . .	50
Методика реализации событийной модели . . . . .	51
Лабораторная работа №4 . . . . .	54
<b>Лекция №11 (27.11.2023)</b>	<b>55</b>
Моделирование систем и языки моделирования . . . . .	55

Формальное описание динамики моделируемого объекта . . . . .	56
SIMSCRIPT . . . . .	57
SIMULA . . . . .	57
GPSS . . . . .	57
Объектно-ориентированное моделирование . . . . .	59
Лабораторная работа №5 . . . . .	59
<b>Лекция №12 (04.12.2023)</b>	<b>62</b>
<b>Лекция №13 (11.12.2023)</b>	<b>63</b>
Графы сетей Петри . . . . .	65
Пространство состояний сети Петри . . . . .	66
Основные свойства сетей Петри . . . . .	69
Свойство ограниченности . . . . .	69
Свойство безопасности . . . . .	69
Свойство консервативности . . . . .	69
Свойство живости . . . . .	69
Некоторые обобщения сетей Петри . . . . .	70
Ингибиторные сети . . . . .	70
Сети с приоритетами . . . . .	71
Сети со случайными срабатываниями переходов . . . . .	71
Иерархические сети . . . . .	72
Раскрашенные (цветные) сети Петри . . . . .	74
<b>Лекция №14 (18.12.2023)</b>	<b>75</b>
Моделирование дискретных систем . . . . .	75
Простейшая система массового обслуживания . . . . .	75

## Лекция №1 (11.09.2023)

Литература:

1. Советов, Яковлев «Моделирование систем» (2000+ г., лучше — последняя версия).
2. Градов, Рудаков «Компьютерное моделирование».
3. Кельтон «Имитационное моделирование».

Моделирование удешевляет весь проект. Есть возможность реализовать «фантастические» условия.

### Философские аспекты моделирования

Свойства объекта характеризуются данными.

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

Методологическая основа моделирования — диалектический метод познания.

**Определение.** Объект — это всё то, на что направлена человеческая деятельность.

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 идёт следующим путём: наблюдение и эксперимент, теоретическое исследование и организация производственного процесса.

В научных исследованиях большую роль играют *гипотезы*, то есть определённые предсказания, основывающиеся на небольшом количестве опытных данных, наблюдениях или догадках.

Быстрая и полная проверка гипотез может быть проведена в ходе специально поставленного эксперимента. В качестве метода суждения при формировании и проверке правильности гипотез большое значение имеет *аналогия*.

**Определение.** Аналогия — суждение о каком-либо частном сходстве двух объектов. Причём такое сходство может быть как существенным, так и не существенным. Существенность в основном зависит от уровня абстрагирования и в общем случае определяется конечной целью процесса исследования.

Современная научная гипотеза создаётся 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*.

**Определение.** Модель — это объект, заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.

**Определение.** 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.

Детерминированное моделирование отражает такие процессы, в которых предполагается отсутствие всяких случайных воздействий. Стохастическое отображает вероятностные процессы.

Статическое служит для описания поведения объектов в какой-либо момент времени, а динамическое отображает состояние системы во времени.

Дискретное — в дискретные моменты времени, непрерывные — на протяжении некоторого времени.

**Определение.** Под математическим моделированием будем понимать процесс установления данному реальному объекту некоторого математического объекта, называемого математической моделью, и исследование этой модели, позволяющее получить характеристики объекта-оригинала.

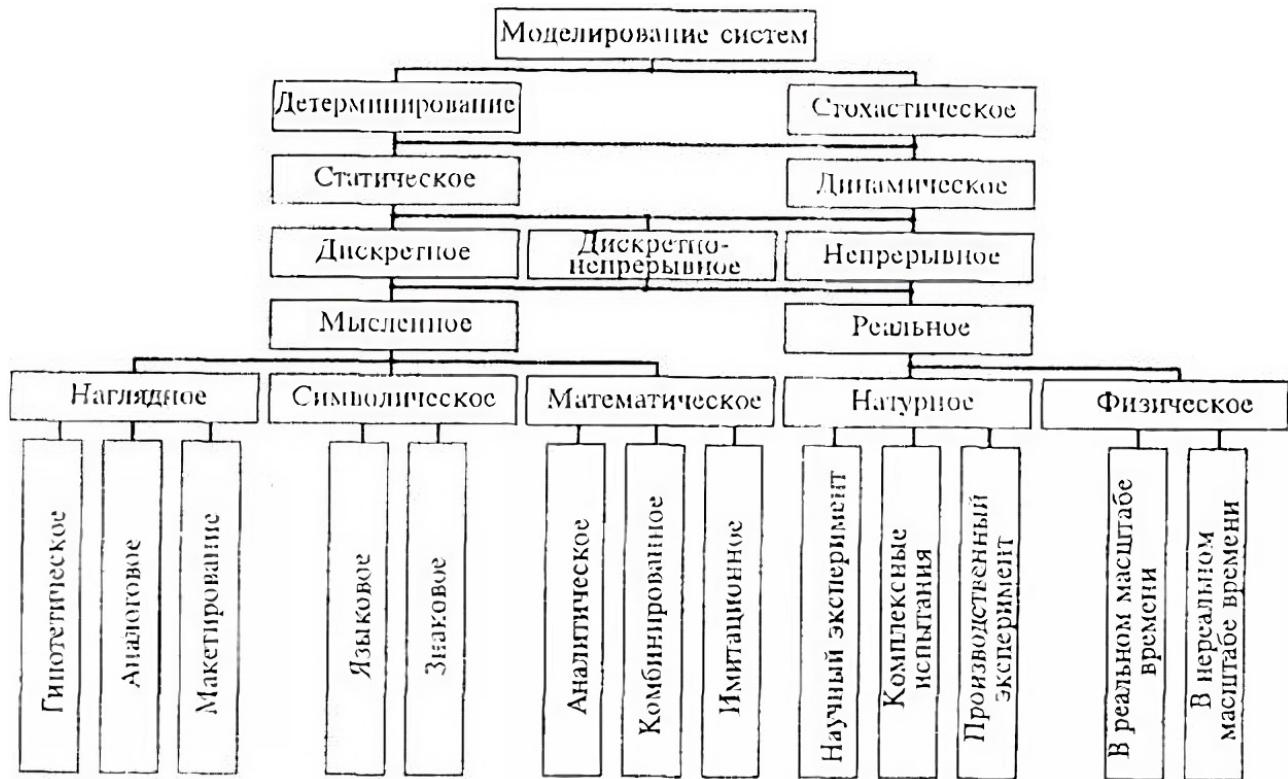


Рис. 1.1: Классификация видов моделирования сложных систем

Вид математической модели зависит от реального объекта и целей моделирования.

Для аналитического моделирования характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений или логических условий.

**Определение.** 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.

При имитационном моделировании, реализующем модель алгоритма, воспроизводят процесс функционирования системы во времени, причём 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 определённые моменты времени, дающие возможность оценить характеристики системы.

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

Имитационные модели позволяют достаточно просто учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики элементов, многочисленные случайные воздействия — что трудно описать аналитически.

## Лекция №2 (18.09.2023)

### Виды имитационного моделирования

#### Агентное моделирование

Используется для исследования децентрализованных систем, динамика функционирования которых определяется не глобальными правилами и законами, а, наоборот, когда эти глобальные правила и законы являются результатом индивидуальной активности членов группы.

Цель агентной модели — получить представление об этих глобальных правилах, общем поведении системы исходя из предположений об индивидуальном, частном поведении отдельных активных объектов и взаимодействии этих объектов в системе.

Таким образом, объект — это некая сущность, обладающая активностью. Может принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружающей средой и самостоятельно изменяться.

Пример: автомобильный трафик (объект — движущееся средство).

#### Дискретно-событийное моделирование

Предлагается абстрагироваться от непрерывной природы объекта и событий, присущих данным объектом. Рассматриваются только основные события: например, ожидание, обработка заказа. Данное направление наиболее развито приложениями.

Пример: производственные процессы, логистика.

#### Системная динамика

Для исследования системы строятся графические диаграммы причинных связей и глобальных влияний параметров объектов на другие. Основным параметром является время.

Выделяют три основных этапа в развитии средств имитационного моделирования.

1. Создание имитационной модели на универсальном языке программирования. Могут использоваться специализированные языки компьютерного моделирования (например, GPSS), объектно-ориентированные языки моделирования.

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

Самый сложный этап имитационного моделирования — формализация модели. Необходимо четко определить входные данные, внутреннее содержание модели, выходные данные и воздействия внешней среды.

Нерешённой проблемой остается присутствие в процессе имитационного моделирования **неформального** этапа перевода поставленной задачи на язык математики. Этот этап не может быть решён ни программистом, ни аналитиком самостоятельно — необходима работа нескольких человек разной квалификации.

## Технические средства имитационного моделирования

- как средства расчёта по полученным аналитическим моделям;
- как средства разработки программ.

При моделировании используется два вида вычислительной техники.

### 1. Цифровые машины:

- процессор;
- память;
- операционная система.

### 2. Аналоговые машины.

В отличие от цифровой, в основе аналоговой вычислительной технике заложен принцип моделирования, а не счета. 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).

**Определение.** Под аналоговой вычислительной машиной будем понимать совокупность электрических элементов, организованных в систему, позволяющую изоморфно моделировать динамику исследуемого объекта.

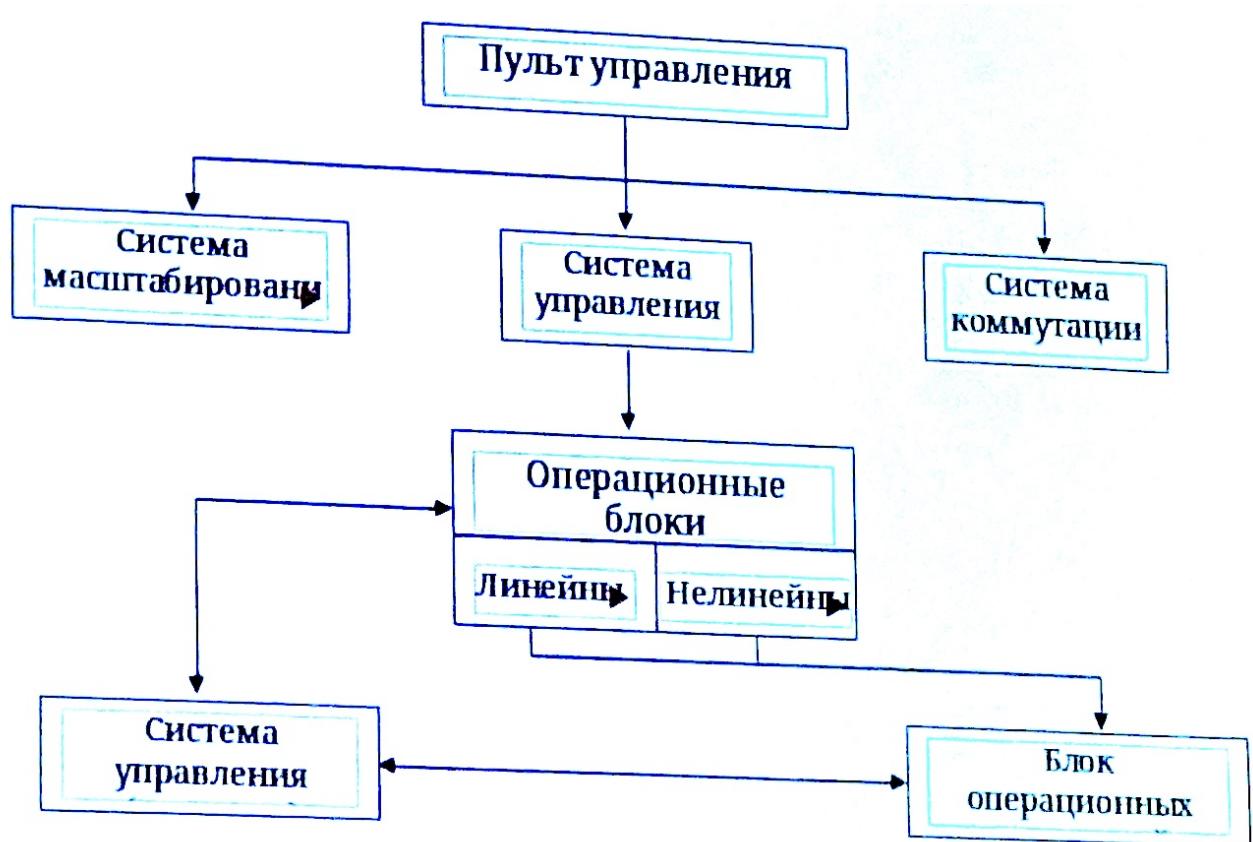
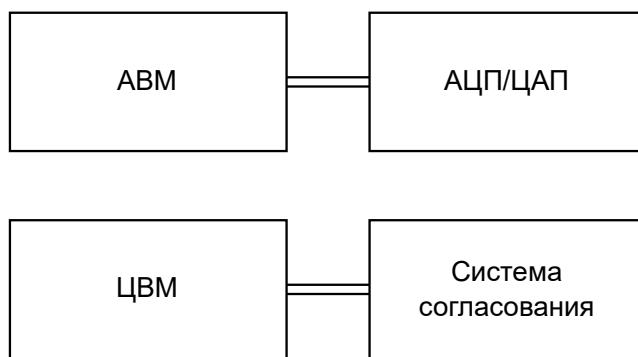


Рис. 2.1: Структурная схема АВМ

## Лекция №3 (25.09.2023)

### Гибридные вычислительные машины

Гибридная вычислительная машина объединяет в себе узлы и блоки типовых или специализированных аналоговых и цифровых вычислительных машин с использованием **различных форм представления информации** и методов её переработки.



Для гибридной вычислительной техники характерным является:

- различные преобразователи форм представления информации;
- прецизионные коммутаторы непрерывных сигналов;
- запоминающие устройства непрерывных сигналов;
- устройства сравнения.

Создание гибридной вычислительной машины ставит своей целью сочетать быстродействие аналоговой и точность цифровой техники. Например, путем многократного использования операционных блоков посредством использования цифрового управления.

Применение гибридных вычислительных машин:

- моделирование дискретных систем и случайных процессов;
- решение задач оптимизации;
- исследования в области управления подвижными объектами;
- моделирование систем «человек и машина».

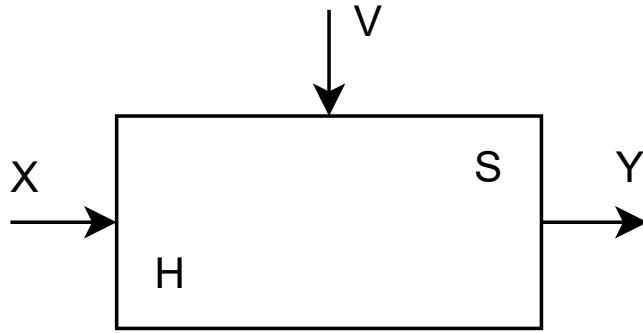
Можно выделить три основных направления развития гибридных вычислительных машин.

1. На основе дискретно управляемых регистров.
2. Разрядно-аналоговые. Высокая точность за счет цифровой формы представления сигнала. Перерабатываем информацию аналоговым способом.
3. Цифровые интегрирующие компьютеры. Основная операция — не суммирование, а интегрирование.

Таблица 1: Сравнительная характеристика аналоговой и цифровой техники

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

## Основы теории моделирования



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  $v_l \in V, l = \overline{1, n_V}$ .
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).

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

Выходные характеристики системы являются зависимыми (эндогенными):

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

Процесс функционирования системы  $S$  описывается во времени некоторым оператором, который в общем случае преобразует независимые переменные в зависимые

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

Эта зависимость называется *законом функционирования системы*. В общем случае она может быть задана в виде функции, функционала, логических условий, в алгоритмическом или табличном виде.

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

Закон функционирования может быть получен и через состояния системы, то есть через свойства системы в конкретные моменты времени:

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

Если рассматривать процесс функционирования системы как последовательную смену состояний  $z_1, z_2, \dots, z_p$ , то они могут быть интерпретированы как координаты точки в  $p$ -мерном пространстве, причем каждой реализации процесса будет соответствовать некоторая фазовая траектория.

**Определение.** Совокупность всех значений состояний называется *пространством состояний* объекта моделирования.

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

$$\begin{aligned}\vec{z}^0 &= (z_1^0, z_2^0, \dots, z_p^0) \\ \vec{z}(t) &= \Phi(\vec{z}^0, \vec{x}, \vec{v}, \vec{h}, t) \\ \vec{y}(t) &= F(\vec{z}, t)\end{aligned}$$

Закон функционирования через состояние:

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

В общем случае время в модели может быть непрерывным на некотором интервале, а может быть дискретным (квантованным на некотором отрезке).

**Определение.** Под математической моделью реальной системы понимают конечное множество переменных  $x(t)$ ,  $v(t)$  и  $h(t)$  вместе с математическими связями между ними.

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

## Типовые математические схемы

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

<...>

Таким образом, использование D-схем позволяет формализовать процесс функционирования непрерывно-детерминированных систем и оценить их основные характеристики, применяя аналитический или имитационный подход, реализованный в виде соответствующего языка моделирования непрерывных систем или АВМ/ГВМ.

## Лекция №4 (02.10.2023)

В теории систем имеются следующие базовые понятия.

**Определение.** Система — множество элементов, находящихся в отношениях и связанных между собой.

**Определение.** Элемент — часть системы, представление о которой нецелесообразно подвергать дальнейшему членению.

**Определение.** 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, членности, иерархичности, многоаспектности.

**Определение.** Подсистема — часть системы (подмножество элементов и их взаимосвязей), которая имеет свойства системы.

**Определение.** Надсистема — система, по отношению к которой рассматриваемая система является подсистемой.

**Определение.** Структура — отображение совокупности элементов системы и их взаимосвязей.

Понятие структуры отличается от понятия самой системы также ещё и тем, что при описании структуры принимают во внимание лишь типы элементов и связей без конкретизации значений их параметров.

**Определение.** Параметр — величина, выражающая свойства системы или её части.

## Характеристики сложных систем

К главным относятся следующие характеристики сложных систем.

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 её 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 (modeling);
- анализ свойств систем на основе исследования их моделей (simulation).

## 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 быть блочной, то есть допускать возможность замены, добавления, исключения некоторых частей без изменения всей модели;
- компьютерная реализация модели должна соответствовать имеющимся техническим ресурсам.

Необходимость определяет оценку системы с требуемой точностью и достоверностью.

Процесс моделирования включает разработку и компьютерную реализацию модели. Является итерационным. Этот итерационный процесс продолжается до тех пор, пока не будет получена модель, которую можно считать адекватной в рамках решения поставленной задачи.

## Основные этапы моделирования больших систем

### Первый этап

На первом этапе построения концептуальной (описательной) модели системы, формулируется модель и строится её формальная схема. Основным назначением данного этапа является переход от содержательного описания объекта к его математической модели. Это наиболее ответственный и наименее формализованный этап. Исходный материал — содержательное описание объекта.

1. Проведение границы между системой и внешней средой.
2. Исследование моделируемого объекта с точки зрения выделения основных составляющих процесса функционирования по отношению к цели моделирования.
3. Переход от содержательного описания системы к формализованному, который в данной интерпретации сводится к исключению из рассмотрения некоторых второстепенных элементов описания (не оказывающих существенного влияния на ход процессов, исследуемых с помощью модели).
4. Оставшиеся элементы модели (системы) группируются в следующие блоки:
  - первой группы — имитатор воздействия внешней среды;
  - второй группы — имитатор воздействия модели;
  - третьей группы — 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

На втором этапе — 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

Третий этап моделирования — этап получения и интерпретации результатов.

1. Планирование компьютерного эксперимента. Составление плана эксперимента с указанием комбинаций переменных, параметров. Главная задача — дать максимальный объём 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.

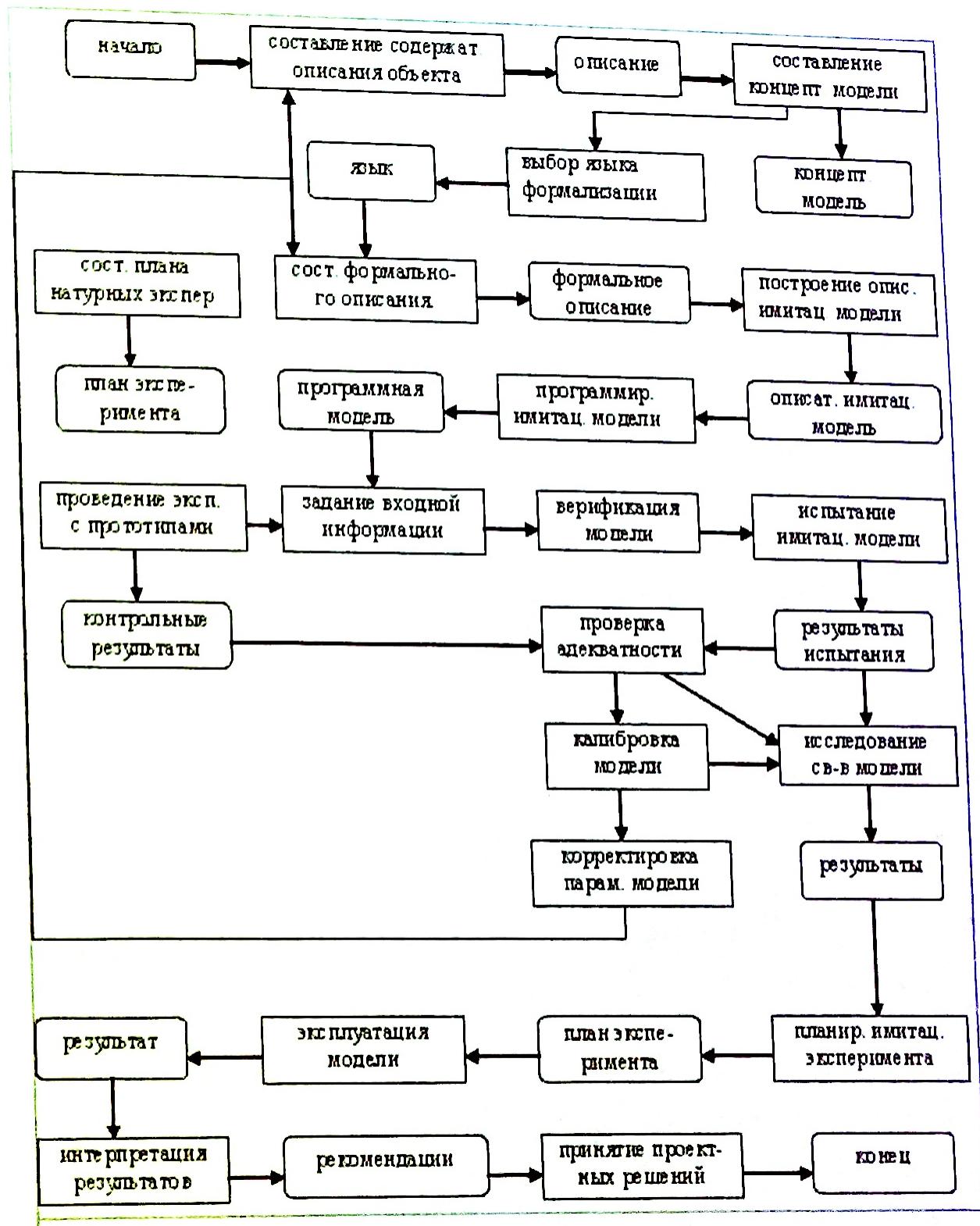


Рис. 4.1: Схема взаимосвязи технологических этапов моделирования

## Лекция №5 (09.10.2023)

### Лабораторная работа №1

Изучить равномерное распределение от  $a$  до  $b$  и распределение по варианту. Выслать отчёт (титульный лист, графики функций, скриншот запущенной программы) до четверга 14:00.

Иерархия:

1. Системный уровень. Рассматриваем всю вычислительную систему как единую модель, в качестве элементов — процессор, память, все внешние устройства. Используемые модели: вероятностные автоматы, системы массового обслуживания, графовые структуры (сети Петри), агрегативные модели. Решаются вопросы, связанные с производительностью системы.
2. ФЛУП (функционально-логический уровень):
  - a) РП (уровень регистровых передач). Рассматриваем, например, из чего состоит процессор. Типовая математическая схема — автоматы или дискретные сети. На этом уровне смотрим, как реализуется команда процессора.
  - б) ЛУ (логический уровень). Рассматриваем вопросы логики преобразования информации, разбиваем машинную команду на алгоритмы реализации этой операции в двоичной системе счисления.

Рассматриваем отдельные части системы.

3. Схемотехнический. Дискретные элементы рассматриваются в виде электронных реализаций (электронных схем). Математические модели — интегро-дифференциальные уравнения. Рассматриваются вопросы различных задержек и искажений сигнала.
4. Конструкторский уровень. Речь идет о «железе», основной вопрос — решение размещения всех элементов на плате. Самые сложные модели связаны с вопросами охлаждения и отведения тепла.

### Моделирование на системном уровне

Используются модели вычислительных систем и сетей.

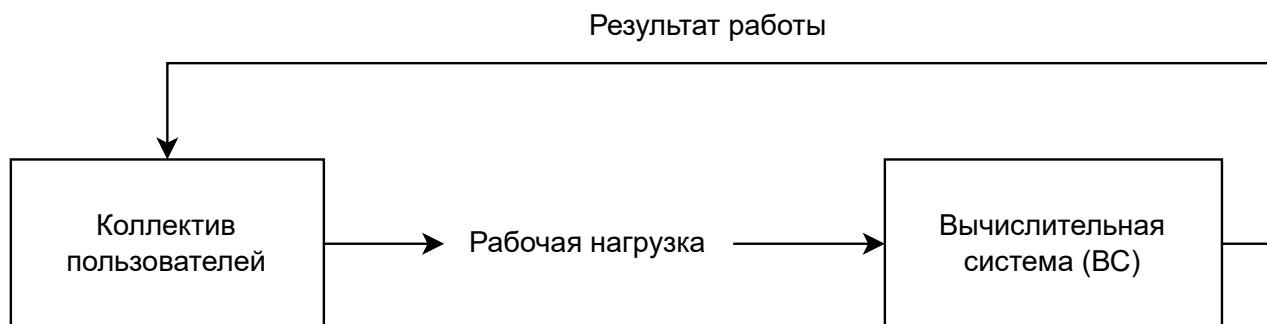
**Определение.** Вычислительная система — комплекс аппаратных и программных средств, которые в совокупности выполняют определенные рабочие функции.

**Определение.** Операционная система — набор ручных и автоматических процедур, которые позволяют группе людей эффективно использовать вычислительную установку.

Основная черта операционных систем — согласованность работы управляющих программ.

**Определение.** Коллектив пользователей — сообщество таких людей, которое использует систему для удовлетворения своих нужд по обработке информации.

**Определение.** Входная информация — программы, данные, которые создаются коллективом пользователей, и называются *рабочей нагрузкой*.



**Определение.** Индекс производительности — описатель, который используется для представления производительности в самом широком смысле слова.

Различают количественные и качественные индексы производительности. Качественными, например, являются удобство использования, субъективная легкость использования системы, мощность системы команд. Основными количественными индексами являются:

- пропускная способность — объем информации, обрабатываемый в единицу времени;
- время ответа (реакции) — время между предъявлением системе входных данных и появлением соответствующей выходной информации.

**Определение.** 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, её 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 путём;
- разработка структурной схемы (определение состава устройств и способов их взаимодействия);
- определение требований к выходным параметрам устройств.

Особенности непрерывного стохастического подхода рассмотрим на примере использования в качестве типовой математической схемы систем массового обслуживания (queueing-систем). При этом исследуемая система формализуется как некоторая система обслуживания (например, компьютер и поток заявок). 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 и само обслуживание.

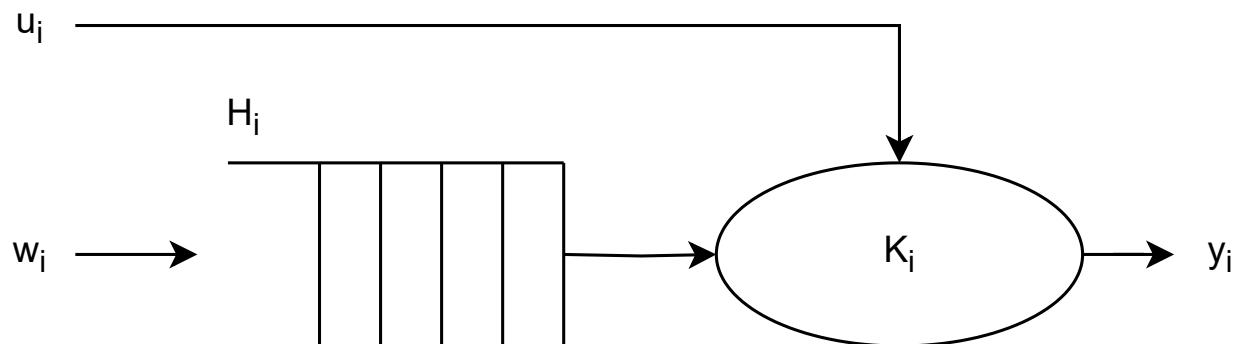


Рис. 5.1:  $i$ -ый прибор обслуживания

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

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

**Определение.** Поток называется *однородным*, если он характеризуется только моментами поступления (вызывающими моментами).

**Определение.** Поток называется *неоднородным*, если он задается последовательностью, в которой кроме вызывающих моментов есть набор признаков события (например, приоритетность).

**Определение.** Если интервалы времени между сообщениями не зависят между собой и являются случайными, то поток с *ограниченным последействием*.

**Определение.** Поток событий называется *ординарным*, если вероятность того, что на малый интервал времени  $\Delta t$  попадает больше одного события, пренебрежительно мала по сравнению с вероятностью того, что на этот интервал времени попадает ровно одно событие.

**Определение.** Поток называется *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.

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

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

Для стационарного потока его интенсивность не зависит от времени и представляет собой постоянное значение, равное числу средних событий.

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;
- ёмкости накопителей.

Если ёмкость накопителя равна 0, то система *с потерями*. Если емкость равна бесконечности — система *с ожиданием*.

Для задания Q-схемы также необходимо описать алгоритм её 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.

## Лекция №6 (16.10.2023)

Для получения соотношений, связывающих характеристики, описывающие функционирование Q-схем, вводят некоторые допущения относительно входных потоков, функций распределения, длительности обслуживания запросов, дисциплин обслуживания.

Для математического описания функционирования устройств, развивающихся в форме случайного процесса, может быть применён математический аппарат, разработанный в теории вероятностей для т. н. марковских случайных процессов.

**Определение.** 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<sup>a</sup>.

---

<sup>a</sup>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,  $\Lambda$  — коэффициенты, показывающие, с какой скоростью система переходит из одного состояния в другое (интенсивность).

Для стационарного состояния:

$$\begin{aligned} \Phi(P(t), \Lambda) &= 0 \\ Y &= Y(P(t), \Lambda) = Y(P(\Lambda)) \end{aligned}$$

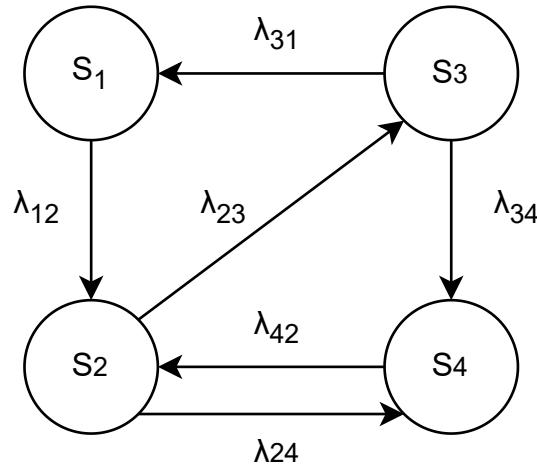
Последнее соотношение представляет собой зависимость выходных параметров от внутренних параметров модели и является *базисной моделью*.

$$\begin{aligned} Y &= Y(X, V, H) \\ \Lambda &= \Lambda(X, V, H) \end{aligned}$$

Последняя формула — 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.

Пусть есть некоторая сложная система  $S$ .



Найдём вероятность  $P_1(t)$ , то есть вероятность того, что в момент времени  $t$  система будет находиться в состоянии  $S_1$ . Придадим  $t$  малое приращение  $\Delta t$  и найдём 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  $\Delta t$  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

$$\begin{aligned}
 P_1(t + \Delta t) &= P_1(t) \cdot (1 - \lambda_{12}\Delta t) + P_3(t)\lambda_{32} \cdot \Delta t \\
 \frac{P_1(t + \Delta t) - P_1(t)}{\Delta t} &= -P_1(t)\lambda_{12} + P_3(t)\lambda_{31} \\
 \lim_{\Delta t \rightarrow 0} \left( \frac{P_1(t + \Delta t) - P_1(t)}{\Delta t} \right) &= -P_1(t)\lambda_{12} + P_3(t)\lambda_{31} \\
 P'_1(t) &= -P_1(t)\lambda_{12} + P_3(t)\lambda_{31}
 \end{aligned}$$

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:

$$\begin{aligned} P'_2(t) &= -P_2(t) \cdot (\lambda_{23} + \lambda_{24}) + P_1(t)\lambda_{12} + P_4(t)\lambda_{42} \\ P'_3(t) &= -P_3(t) \cdot (\lambda_{31} + \lambda_{34}) + P_2(t)\lambda_{23} \\ P'_4(t) &= -P_4(t)\lambda_{42} + P_2(t)\lambda_{24} + P_3(t)\lambda_{34} \end{aligned}$$

Интегрирование данной системы даёт 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:

$$\begin{aligned} P_1(0) &= 1 \\ P_2(0) &= 0 \\ P_3(0) &= 0 \\ P_4(0) &= 0 \end{aligned}$$

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

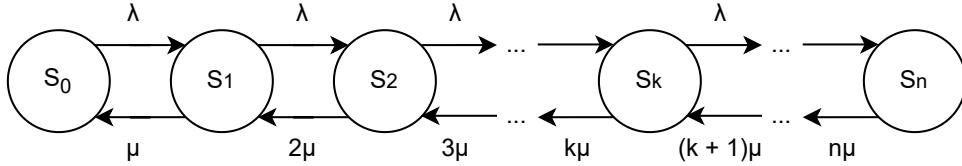
Все уравнения Колмогорова построены по следующим правилам.

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.

Рассмотрим многоканальную систему массового обслуживания 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 пропишем у стрелок интенсивности соответствующих потоков событий. Поток заявок (слева направо) имеет интенсивность  $\lambda$ . 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  $\mu$ . 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.

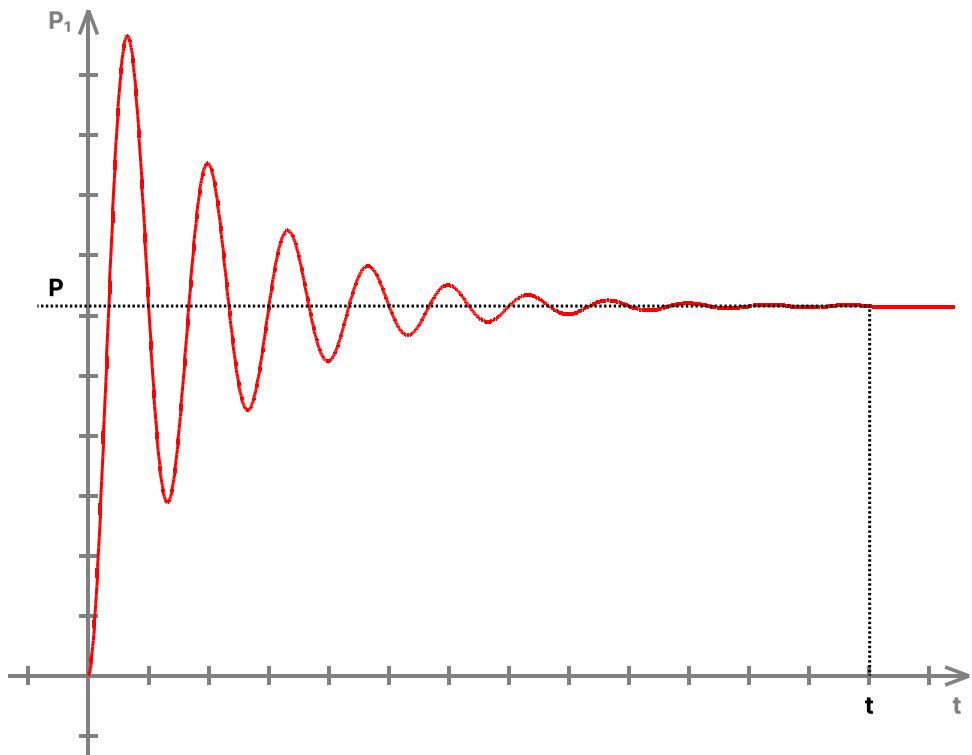
Параметр  $\mu$  — это, как правило, функция технических характеристик объекта и средств реализаций (характеристики решаемых задач).

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).

## Лабораторная работа №2

Предельные вероятности состояний  $P_0 \dots P_n$  характеризуют установленный режим работы. В результате получаем  $P_0$  и  $P_k$ .



Решив уравнения Колмогорова, получим  $P$ , в лабораторной работе нужно найти  $t$ .

Определить среднее относительное время пребывания системы в предельном стационарном состоянии. Интенсивности переходов из состояния в состояние задаются в виде матрицы максимальной размерности 10.

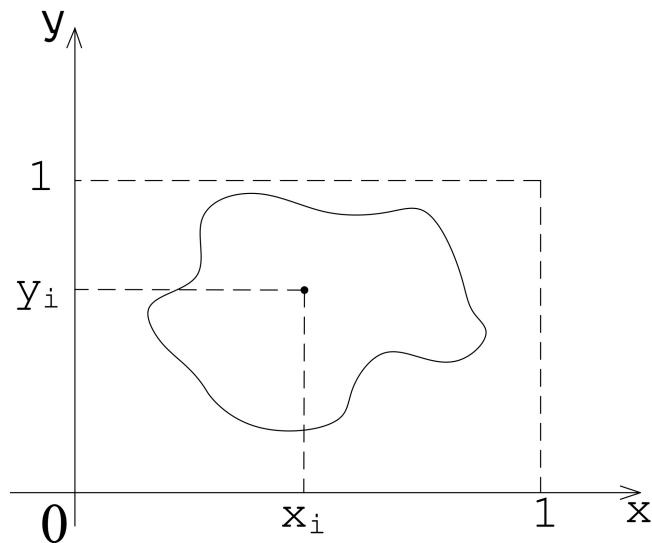
Определить вероятность нахождения системы в заданном состоянии.

## Лекция №7 (23.10.2023)

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

Преимущества:

- универсальность, обуславливающая возможность всестороннего статистического исследования объекта.

Недостатки:

- для реализации нужны достаточно полные статистические сведения о параметрах;
- большой объём вычислений.

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

## 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 числарабатываются специальной электронной приставкой — генератором случайных чисел. Как правило, в качестве неё 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.

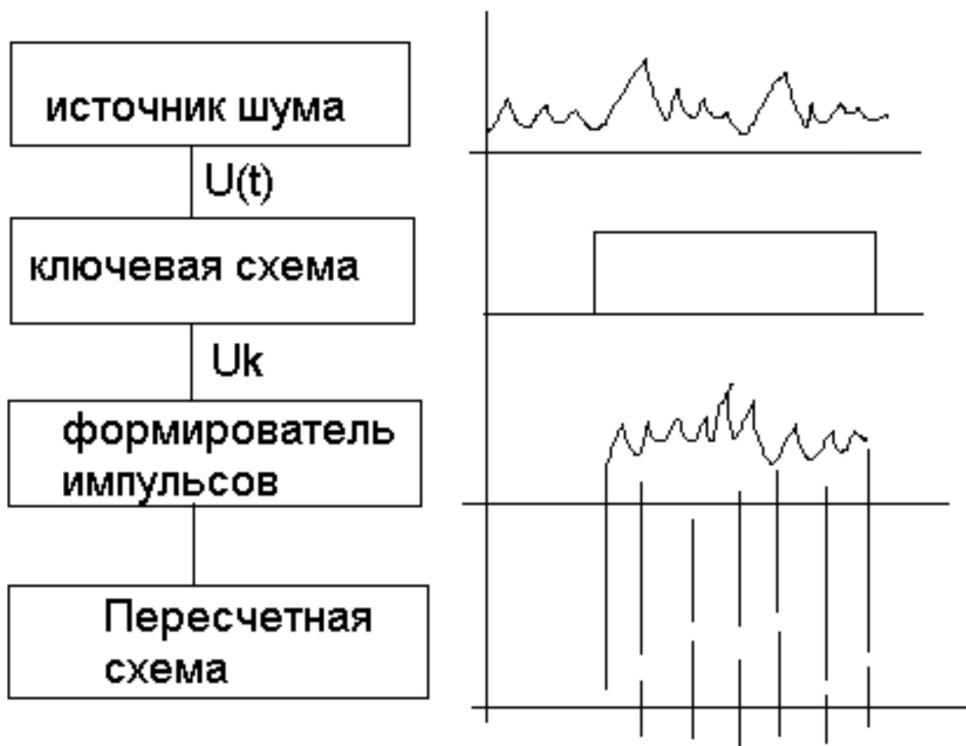


Рис. 7.1: Структурная схема аппаратного генератора случайных чисел

- + 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.

## 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.

Рассмотрим некоторую случайную величину  $X$ , которая может принимать любые значения из интервала  $[a, b]$  и имеет плотность распределения  $\frac{1}{b-a}$ . Найти функцию распределения вероятности, математическое ожидание и дисперсию.

$$F(x) = \int_a^x \frac{1}{b-a} dx = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ 1, & x > b \end{cases}$$

$$MX = \int_a^b x f(x) dx = \frac{a+b}{2}$$

$$DX = M(X^2) - (MX)^2 = \int_a^b x^2 f(x) dx = \frac{(b-a)^2}{12}$$

## Алгоритмический способ получения случайных чисел

Для получения случайной величины из последовательности случайных величин с заданным законом распределения обычно используют одно или несколь-

ко значений, равномерно распределенных случайных чисел. Поэтому вопрос получения равномерного распределения имеет наиважнейшее значение.

Как правило, равномерное распределение получаем с помощью некоторого рекуррентного соотношения. Это означает, что каждое последующее число  $a_{i+1}$  образуется из предыдущего числа  $a_i$  или из группы предыдущих чисел путем применения некоторого алгоритма, состоящего из арифметических и логических операций.

На языке FORTRAN необходимо было использовать следующие константы:

- 1220703125;
- 2147483647;
- $2147483647 + 1$  (почему не сложить сразу?);
- $<\dots>$ .

## Лекция №8 (30.10.2023)

### Немарковские случайные процессы, сводящиеся к марковским

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

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

#### Метод псевдосостояний

Сущность метода состоит в том, что состояния системы, потоки переходов из которых являются немарковскими заменяются эквивалентной группой фиктивных состояний, потоки из которых уже являются марковскими.

Условия статистической эквивалентности выбираются по-разному. За счет расширения числа состояний в системе некоторые процессы удается точно свести к марковским. Созданная таким образом новая система статистически эквивалентна или близка к реальной системе.

К числу процессов, которые введением фиктивных состояний можно точно свести к марковским, относятся процессы, происходящие под воздействием потоков Эрланга. В случае потока Эрланга  $k$ -ого порядка интервал времени между соседними событиями представляет собой сумму  $k$  независимых случайных интервалов, распределенных по показательному закону.

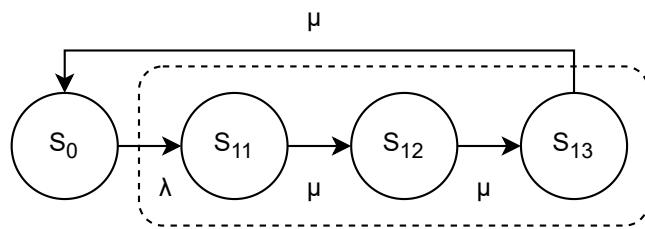
Сведение потока Эрланга  $k$ -ого порядка к пуассоновскому осуществляется введением  $k$  псевдосостояний. Интенсивность перехода между состояниями равна соответствующему параметру потока Эрланга. Полученный таким образом случайный процесс (он эквивалентен исходному) является марковским, так как интервалы времени нахождения его в различных состояниях подчиняются показательному закону.

Некоторые устройства выходят из строя с интенсивностью  $\lambda$ . 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 отказалось и восстанавливается.

Переход из  $S_0$  в  $S_1$  — пуассоновский закон, наборот — эрланговский. Время восстановления можно представить в виде суммы трёх времен, они распределены по показательному закону и имеют интенсивность  $\mu$ .



Для установившегося состояния составляем уравнения Колмогорова.

$$\begin{cases} p'_0 = 0 = -\lambda p_0 + \mu p_{13} \\ p'_{11} = 0 = -\mu p_{11} + \lambda p_0 \\ p'_{12} = 0 = -\mu p_{12} + \mu p_{11} \\ p'_{13} = 0 = -\mu p_{13} + \mu p_{12} \\ p_0 + p_1 = 1 \\ p_1 = p_{11} + p_{12} + p_{13} \end{cases}$$

Решаем полученную систему.

$$\begin{aligned} p_{13} &= \frac{\lambda}{\mu} p_0, \quad p_{11} = \frac{\lambda}{\mu} p_0 \\ p_{12} &= p_{11} = \frac{\lambda}{\mu} p_0 \\ p_1 &= \frac{3\lambda}{\mu} p_0 \\ p_0 + \frac{3\lambda}{\mu} p_0 &= 1 \Rightarrow p_0 = \frac{\mu}{\mu + 3\lambda} \\ p_1 &= \frac{3\lambda}{\mu + 3\lambda} \end{aligned}$$

## Лабораторная работа №3

Необходимо взять одно-, двух- и трехразрядные числа, сгенерированные табличным и алгоритмическим способами (три столбца). Дать возможность ввести 10 любых чисел и затем под каждым из столбцов вывести число, показывающее случайность данной последовательности — разработать количественный критерий случайности для чисел, сгенерированных табличным и алгоритмическим способами. Если числа будут подчиняться какому-либо закону, то они уже не случайны.

См. Кнут II том и Деон А. Ф.

## Метод вложенных цепей Маркова

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

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.

## Алгоритмы генерации последовательности псевдослучайных чисел

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

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

Здесь  $n \in \mathbb{N}$ . Якоби доказал, что при рациональном коэффициенте  $a$  множество  $y_n$  конечно, а при иррациональном — бесконечно и всюду плотно на интервале от 0 до 1.

Среднее по реализациям псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1.

## Фон Нейман

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

## Лемер

$$g_{n+1} = kg_n + c \bmod m$$

$<\dots>$

Принято решение, что работать нужно с целыми числами. При  $c = 0$  и  $m = 2^n$  наибольший период достигается при  $k = 3 + 8i$  или  $k = 5 + 8i$  и нечетном начальном числе.

В 1977 г. Форсайд показал, что тройки последовательности чисел лежат всего на 15 параллельных плоскостях.

От отчаяния используют два или даже три 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

Берём 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:

$$\begin{aligned}\frac{x - a}{b - a} &= R \\ x &= a + (b - a)R\end{aligned}$$

Здесь  $R$  изменяется от 0 до 1.

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:

$$\begin{aligned} 1 - e^{-\lambda x} &= R \\ x &= -\frac{1}{\lambda} \ln(1 - R) \end{aligned}$$

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 равномерно распределенной на  $[0, 1]$   $R_i$  до тех пор, пока не будет выполнено следующее соотношение.

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

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  $\lambda$  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  $N = 12$ .

## Лекция №9 (13.11.2023)

### 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  
 ИИ - источник информации      ПСС - программа сбора статистики  
 ПИ - программный имитатор      А - абонент

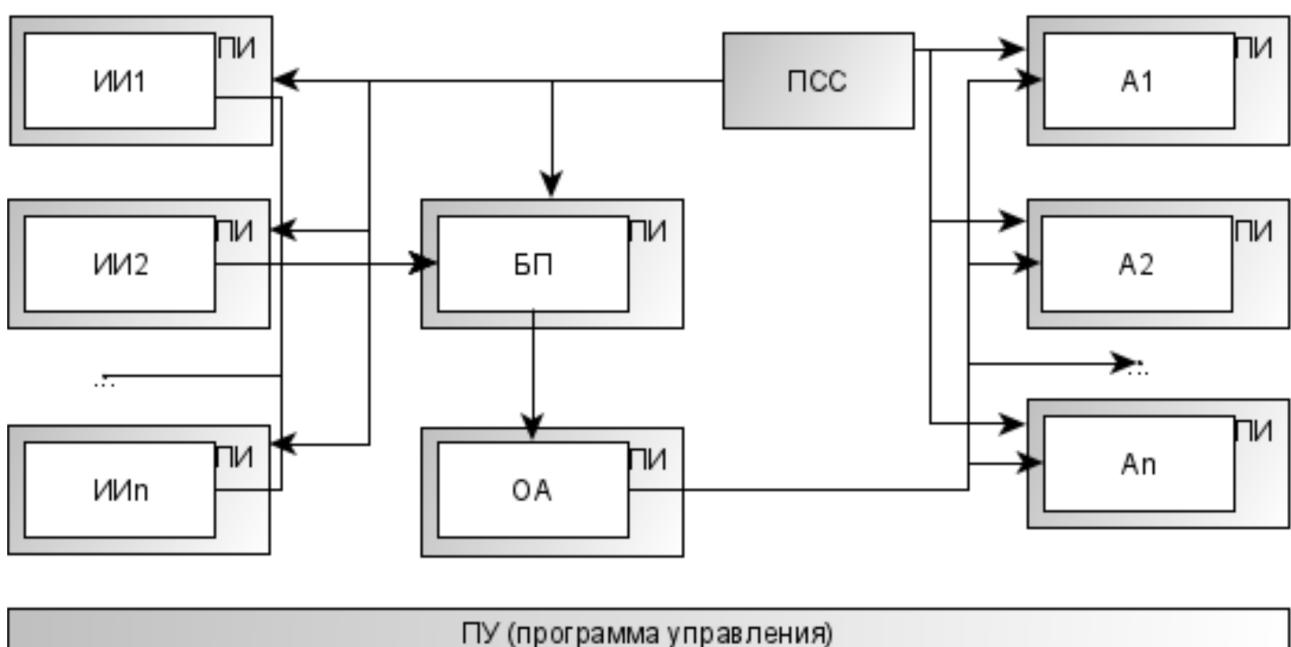


Рис. 9.1: Структурная схема программно-алгоритмического комплекса

Источники информации выдают на вход буферной памяти 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 обслуживающий аппарат по принципу FIFO. Длительность обработки одного сообщения также может быть случайна, но закон обработки должен быть задан.

Быстродействие обслуживающего аппарата ограничено, поэтому возникает

очередь на обработку.

Смысл программы синхронизации (управления) заключается в *протяжьске* модельного времени.

## Моделирование потока сообщений

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

Таблица 2: 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

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

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

## Моделирование работы абонентов

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

Моделирование начинаем с разработки концептуальной модели.

**Определение.** Память — электромеханическое устройство, предназначенное для хранения, записи и чтения информации.

Блок буферной памяти должен производить запись и считывание числа, выдавать сигнал переполнения и отсутствия данных, в любой момент модельного времени располагать сведения 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.

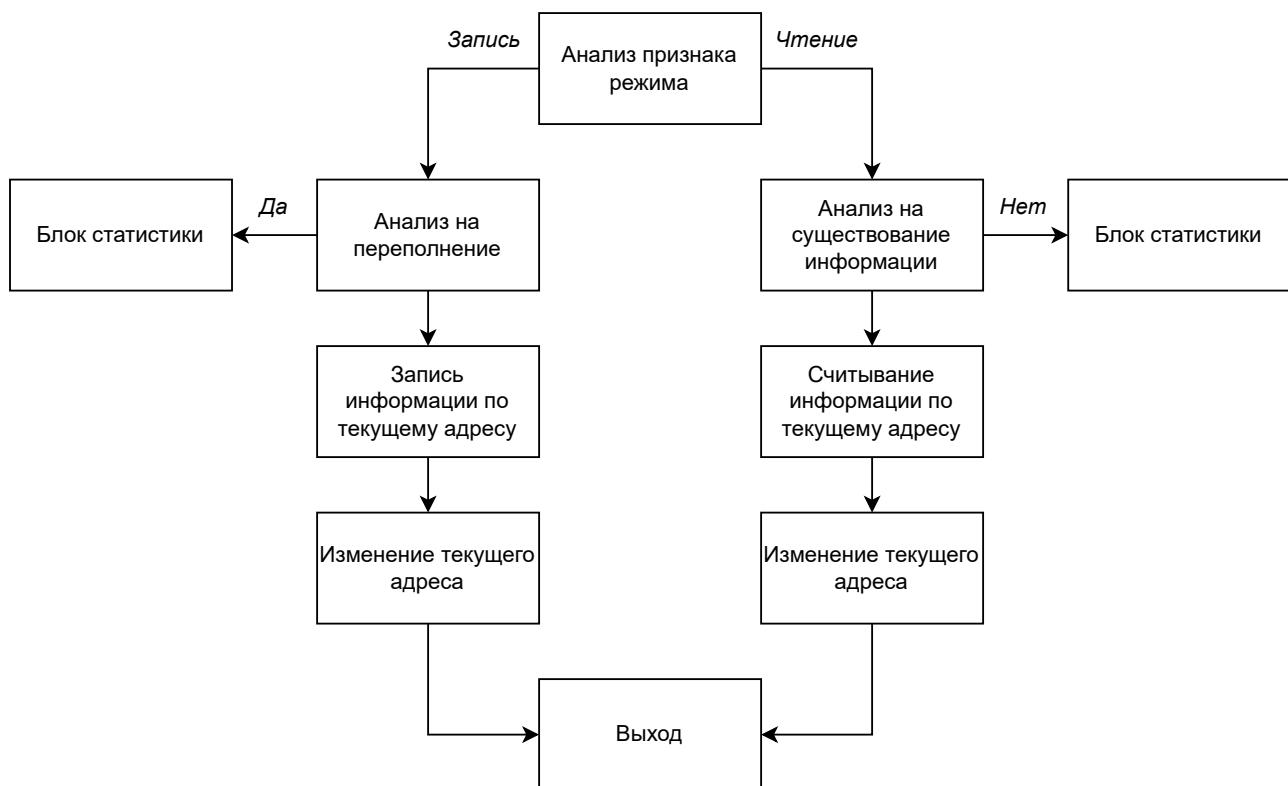


Рис. 9.2: Структурная схема памяти

Здесь:

- Р — массив обращения;
- WYB — признак обращения к буферной памяти (0 — режим записи, 1 — режим выбора сообщения);

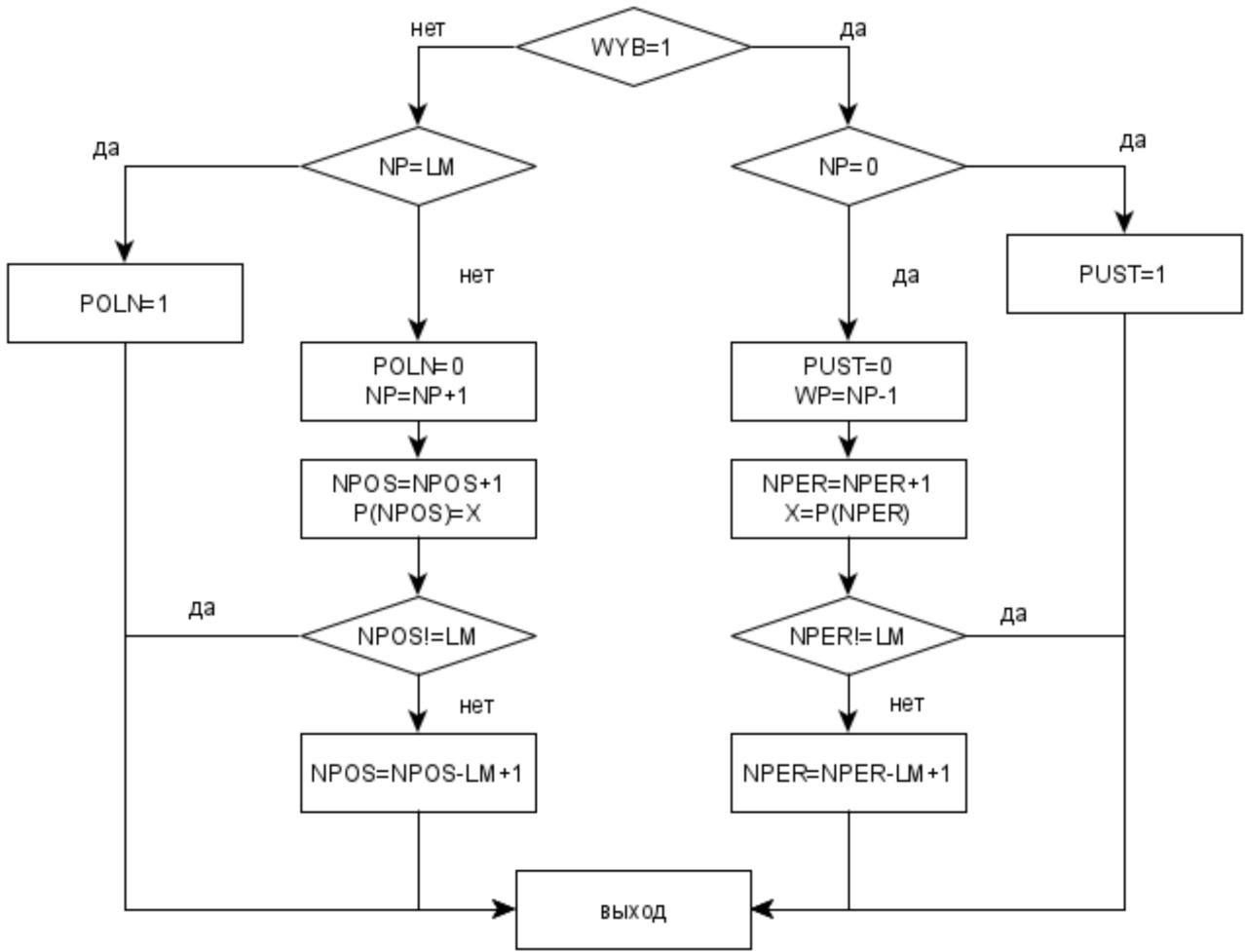


Рис. 9.3: Схема алгоритма буферной памяти

- NP — число сообщений в памяти;
- LM — объём 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

Задача блока статистики 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. 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:

$$\frac{N_{\text{потер}}}{N_{\text{потер}} + N_{\text{обраб}}}$$

## Лекция №10 (20.11.2023)

### 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 $\Delta t$

Принцип  $\Delta t$  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  $\Delta t$  — значительные затраты машинного времени на анализ и контроль правильности функционирования всей системы. При недостаточно малом  $\Delta t$  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.

Обозначения:

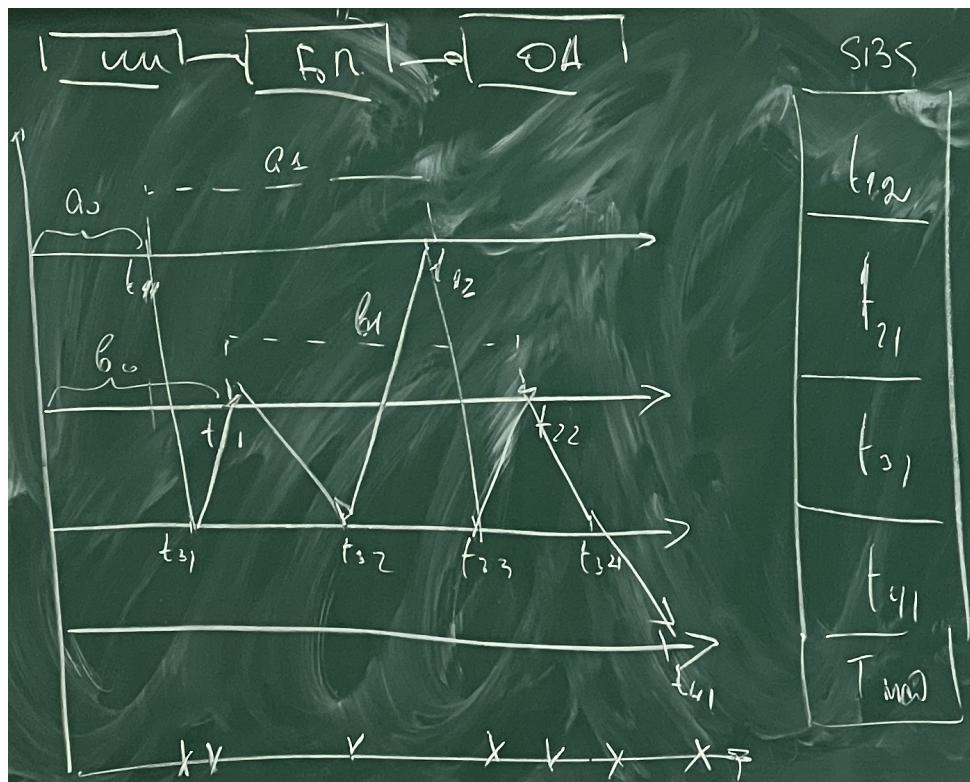


Рис. 10.1: Схема событийного принципа

- $t_{11}, t_{12}$  — моменты появления сообщений на выходе источника информации;
- $b_1$  — время обслуживания первого сообщения;
- $t_{3i}$  — моменты времени сбора статистики;
- $t_{41}$  — момент времени окончания моделирования;
- SBS — список будущих событий.

### Методика реализации событийной модели

Для всех активных блоков (блоков, порождающих событий) заводят свой элемент в одномерном массиве (списке будущих событий — СБС).

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.

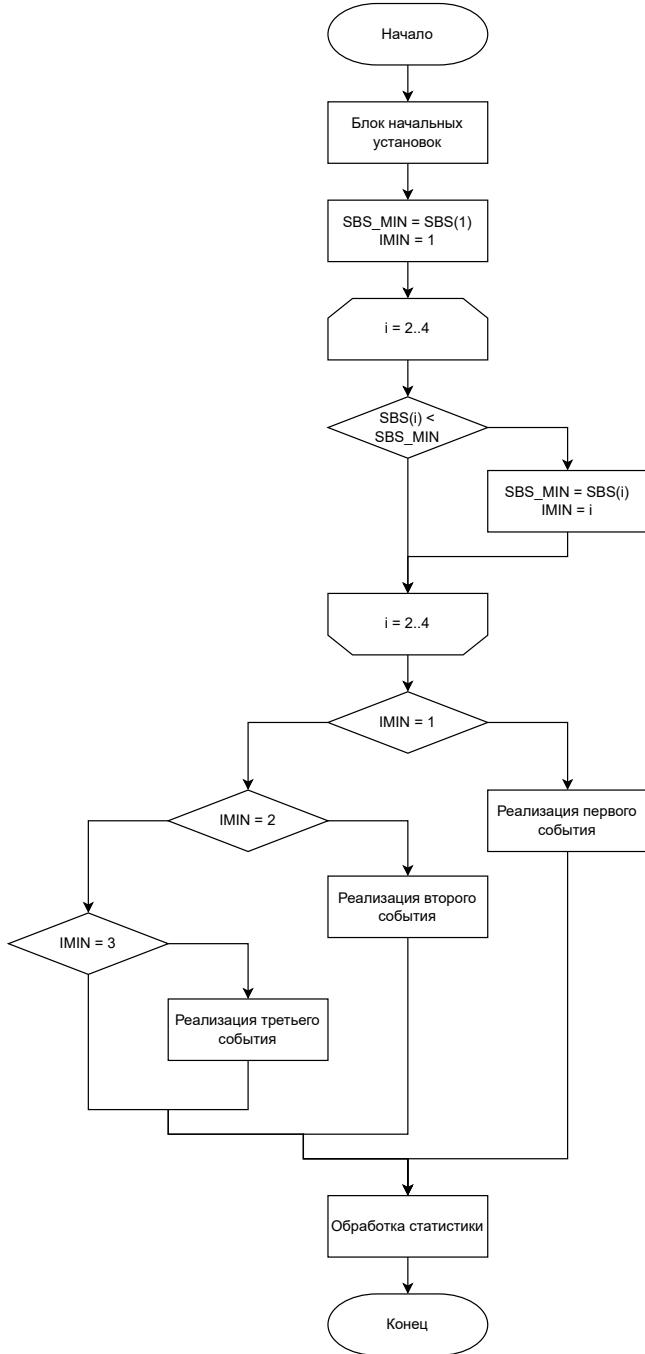


Рис. 10.2: Схема алгоритма

1. В списке будущих событий определяем минимальное значение и его порядковый номер.
  2. 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.

### 3. Выводим статистику по каждому устройству, связанному с событием.

Два описанных выше принципа являются универсальными методами протяжки модального времени. Выбор метода необходимо осуществлять исходя из распределения событий во времени. 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*. Примером таких сложных систем может являться 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  $\Delta t$ , 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.

## Лабораторная работа №4

Есть генератор, который выдает сообщения в систему, распределенные по равномерному закону. Есть очередь, куда сообщения поступают в режиме FIFO, и обрабатываются обслуживающим аппаратом. Обслуживающий аппарат работает по закону первой лабораторной работы (по варианту).

Задача — найти максимальный объем очереди, при котором сообщения не теряются, используя два алгоритма протяжки модельного времени —  $\Delta t$  и событийный.

Часть сообщений (заранее заданная в процентном соотношении от выходного потока) заново поступает в очередь на обработку. Таким образом, максимальная длина очереди должна быть другой.

Должен быть GUI.

## Лекция №11 (27.11.2023)

### Моделирование систем и языки моделирования

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.

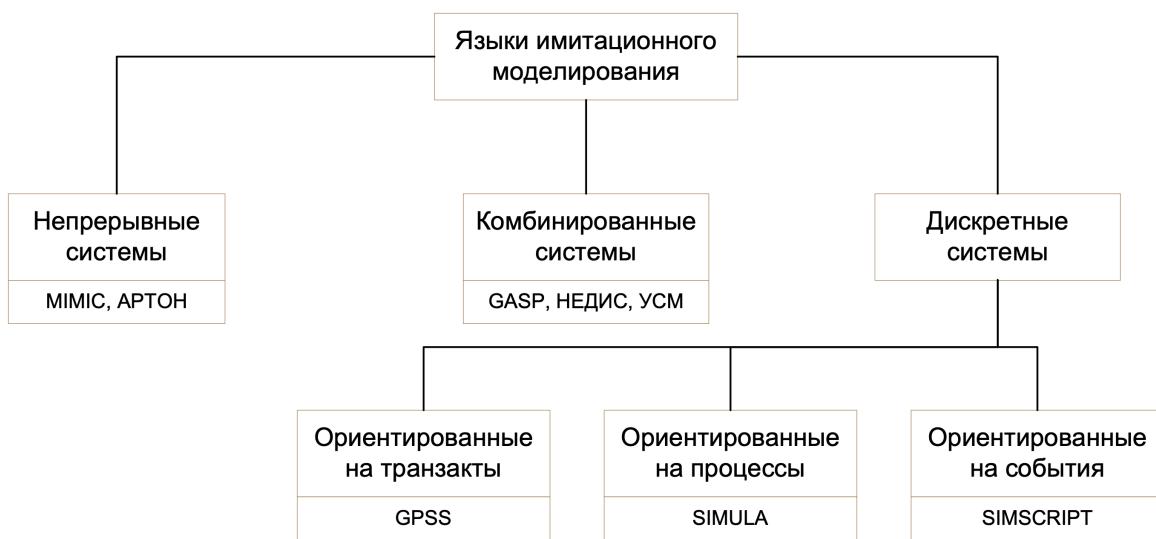


Рис. 11.1: Классификация языков моделирования

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.

## Формальное описание динамики моделируемого объекта

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

**Определение.** Активность — это единый динамический объект, указывающий на совершение единицы работы.

**Определение.** Процесс — это логически связанный набор активностей.

Пример процесса — запись в память.

**Определение.** 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.

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

## SIMSCRIPT

Моделирующая программа организована в виде секций событий. Событие состоит из набора операций, которые выполняются после завершения активности.

## SIMULA

Моделирующая программа организуется в виде набора описаний процесса, 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.

Таблица 3: Сравнение универсальных и специализированных языков моделирования

Преимущества	Недостатки
<b>Универсальные</b>	
Минимум ограничений на выходной формат	Значительное время, затрачиваемое на программирование
Широкое распространение	Значительное время, затрачиваемое на отладку
<b>Специализированные</b>	
Меньше затраты времени на программирование	Необходимость точно придерживаться ограничений на форматы данных
Более эффективные методы выявления ошибок	Меньшая гибкость модели
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	

Возможности программного обеспечивания и имитационного моделирования:

- анимация и динамическая графика:
  - представление сути имитационного моделирования;
  - отладка моделируемой программы;
  - обучение обслуживающего персонала.
- статистические возможности:
  - генераторы псевдослучайных чисел.

Существует два типа анимаций:

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

## Объектно-ориентированное моделирование

Преимущества:

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

Недостаток — трудность изучения.

## Лабораторная работа №5

Моделируем информационный центр. В информационный центр приходят клиенты (пользователи) через интервал времени  $10 \pm 2$  минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании.

Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса от пользователя за  $20 \pm 5$ ,  $40 \pm 10$  и  $40 \pm 20$  ед. времени (минут). Клиенты стараются занять свободного оператора с максимальной производительностью.

Полученные запросы сдаются в накопитель, откуда выбираются на обработку. На первый компьютер — от первого и второго операторов, на второй — от

третьего. Время обработки запроса в компьютерах — 15 и 30 минут соответственно.

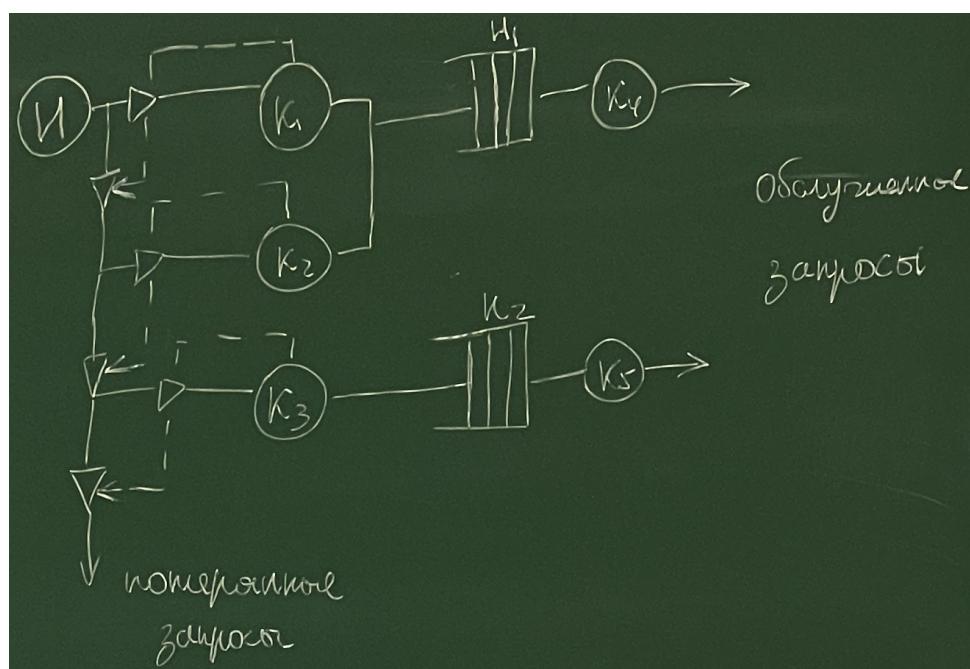
Смоделировать процесс **обработки** 300 запросов. Определить вероятность отказа.



Рис. 11.2: Концептуальная модель

В процессе взаимодействия клиентов с информационным центром возможны:

- режим нормального обслуживания, когда клиент выбирает одного из свободных операторов, отдавая предпочтение тому, у которого меньше номер.
- режим отказа в обслуживании, когда все три оператора заняты.



Переменные и уравнения имитационной модели: эндогенные (время обработки задания  $i$ -ым оператором и время решения задания  $j$ -ым компьютером) и экзогенные (число обслуженных клиентов  $N_0$  и число клиентов, получивших отказ, —  $N_1$ ).

Уравнение модели имеет вид (11.1).

$$P_{\text{отк}} = \frac{N_1}{N_0 + N_1} \quad (11.1)$$

За единицу системного времени выбрать  $\frac{1}{100}$  минуты.

## Лекция №12 (04.12.2023)

*Пропущена. Если есть желание — присылайте PR.*

## Лекция №13 (11.12.2023)

*Отменена, далее — высланный конспект. Судя по всему, это скан с пособия Г. А. Доррера «Методы моделирования дискретных систем», 2004 г.*

**Определение.** Сеть Петри — это математическая модель дискретных динамических систем (параллельных программ, операционных систем, ЭВМ и их устройств, сетей ЭВМ), ориентированная на качественный анализ и синтез таких систем (обнаружение блокировок, тупиковых ситуаций и узких мест, автоматический синтез параллельных программ и компонентов ЭВМ и др.).

Формально в терминах теории систем сеть Петри (Petri Net — PN) — это набор элементов (кортеж):

$$\text{PN} = \{\Theta, P, T, F, M_0\}$$

Здесь:

1.  $\Theta = \{0, 1, 2, \dots\}$  — множество дискретных моментов времени.
2.  $P = \{p_1, p_2, \dots, p_n\}$  — непустое множество элементов сети, называемых *позициями* (местами).
3.  $T = \{t_1, t_2, \dots, t_m\}$  — непустое множество элементов сети, называемых *переходами*.

Множества позиций и переходов не пересекаются:  $P \cap T = \emptyset$ .

4.  $F: (P \times T) \cup (T \times P) \rightarrow \{0, 1, 2, \dots, k, \dots\}$  — функция инцидентности, где  $k$  — кратность дуги.
5.  $M_0: P \rightarrow \{0, 1, 2, \dots\}$  — начальная маркировка позиций.

Функция инцидентности может быть представлена в виде  $F = F^p \cup F^t$  и фактически задает два отображения:

- $F^p(p, t) = P \times T \rightarrow \{0, 1, 2, \dots\}$ , то есть для каждой позиции указываются связанные с ней переходы (с учетом их кратности);
- $F^t(t, p) = T \times P \rightarrow \{0, 1, 2, \dots\}$ , то есть для каждого перехода указываются связанные с ним позиции (также с учетом кратности).

Эти функции, в общем случае зависящие от времени, могут быть представлены

матрицами инцидентности:

$$F^p = \begin{bmatrix} t_1 & t_2 & \cdots & t_m \\ f_{11}^p & f_{12}^p & \cdots & f_{1m}^p \\ \vdots & \vdots & \vdots & \vdots \\ f_{n1}^p & f_{n2}^p & \cdots & f_{nm}^p \end{bmatrix} \begin{matrix} p_1 \\ \vdots \\ p_n \end{matrix} \quad F^t = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \\ f_{11}^t & f_{12}^t & \cdots & f_{1n}^t \\ \vdots & \vdots & \vdots & \vdots \\ f_{m1}^t & f_{m2}^t & \cdots & f_{mn}^t \end{bmatrix} \begin{matrix} t_1 \\ \vdots \\ t_m \end{matrix}$$

Из вершины-позиции  $p_i \in P$  ведет дуга в вершину-переход  $t_j \in T$  тогда и только тогда, когда  $f_{ij}^p > 0$ . В этом случае говорят, что  $t_j$  — выходной переход позиции  $p_i$ .

Множество всех позиций  $p_k$ , для которых  $t_j$  — выходной переход, будем обозначать  $P^j$ . Иными словами,  $P^j = \{p_k : f_{kj}^p > 0\}$ .

Аналогично из каждой вершины перехода  $t_j \in T$  дуга ведет в вершину-позицию  $p_i \in P$  тогда и только тогда, когда  $f_{ji}^t > 0$ . При этом говорят, что  $p_i$  — выходная позиция перехода  $t_j$ .

Множество всех переходов  $t_l$ , для которых  $p_i$  — выходная позиция, будем обозначать  $T^i$ . Таким образом,  $T^i = \{t_l : f_{li}^t > 0\}$ . При  $f_{ij}^p > 0$  и  $f_{ji}^t > 0$  эти величины называются *кратностью* соответствующих дуг.

Каждая позиция  $p_i \in P$  может содержать некоторый целочисленный ресурс  $\mu(p) \geq 0$ , часто отображаемый соответствующим числом точек (фишек) внутри позиции — рисунок 13.1.

Вектор  $M = [\mu_1, \dots, \mu_n]$  называют *маркировкой (разметкой) сети Петри*. Каждая маркировка — это отображение  $M: P \rightarrow \{0, 1, 2, \dots\}$ . Начальная маркировка  $M_0$  определяет стартовое состояние сети Петри.

Смена маркировок (начиная с  $M_0$ ) происходит в результате срабатывания переходов сети. Переход  $t_j \in T$  может сработать при маркировке  $M$ , если для всех  $p_i \in P^j$  выполняется условие  $\mu_i(\theta) - f_{ij}^p(\theta) \geq 0$ , то есть если каждая входная позиция для данного перехода  $p_i \in P^j$  содержит как минимум столько фишек, сколько кратность ведущей к  $t_j$  дуги.

В результате срабатывания перехода  $t_j$  в момент времени  $\theta$  маркировка  $M(\theta)$

сменяется маркировкой  $M(\theta + 1)$  по правилу (13.1):

$$\begin{aligned} \mu_i(\theta + 1) &= \mu_i(\theta) - f_{ij}^p(\theta) + f_{ji}^t(\theta) \\ i &= \overline{1, n}, j = \overline{1, m}, i \in P^j, j \in T^i \end{aligned} \quad (13.1)$$

Иными словами, переход  $t$  изымает из каждой своей входной позиции число фишечек, равное кратности входных дуг, и посыпает в каждую свою выходную позицию число фишечек, равное кратности выходных дуг.

Если может сработать несколько переходов, то срабатывает один (любой из них). Функционирование сети останавливается, если при некоторой *тупиковой маркировке* ни один из ее переходов не может сработать. При одной и той же начальной маркировке сеть Петри может порождать, в силу недетерминированности ее функционирования, различные последовательности срабатывания ее переходов. Эти последовательности образуют *слова* в алфавите  $T$ .

Множество всевозможных слов, порождаемых сетью Петри, называют *языком* сети Петри. Две сети Петри эквивалентны, если они порождают один и тот же язык.

В отличие от конечных автоматов, в терминах которых описываются глобальные состояния систем, сети Петри концентрируют внимание на локальных событиях (переходах), локальных условиях (позициях) и локальных связях между событиями и условиями. Поэтому в терминах сетей Петри более адекватно, чем с помощью автоматов, моделируется поведение распределенных асинхронных систем.

## Графы сетей Петри

Формальное определение сети Петри, изложенное выше, полностью определяет ее функционирование. Однако при решении конкретных инженерных задач удобнее и нагляднее графическое представление этих сетей.

Теоретико-графовым представлением сети Петри является двудольный ориентированный мультиграф сети Петри, который содержит:

- позиции (места), обозначаемые кружками;
- переходы, обозначаемые планками;
- ориентированные дуги (стрелки), соединяющие позиции с переходами и переходы с позициями (кратные дуги обозначаются несколькими параллельными дугами).

Благодаря наличию кратных дуг сеть Петри есть мультиграф. Благодаря двум типам вершин граф называется двудольным. Поскольку дуги имеют направление, граф является ориентированным. Пример такого мультиграфа показан на рисунке 13.1.

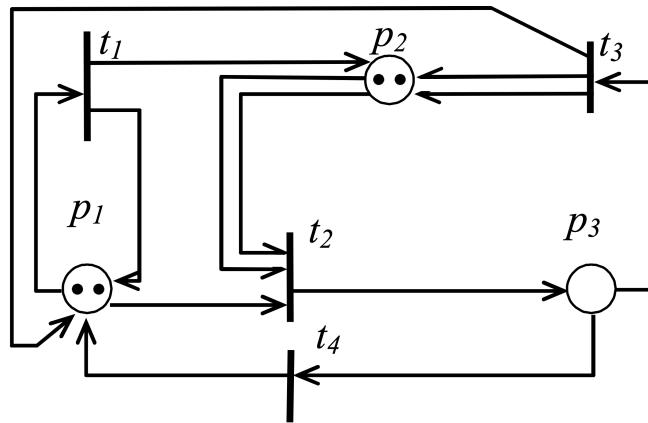


Рис. 13.1: Пример графа сети Петри

Для сети, изображенной на этом рисунке, матрицы инцидентности имеют вид

$$F^p = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{array}{l} p_1 \\ p_2 \\ p_3 \end{array}$$

$$F^t = \begin{bmatrix} p_1 & p_2 & p_3 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{array}{l} t_1 \\ t_2 \\ t_3 \\ t_4 \end{array}$$

Начальная маркировка, как видно из рисунка, —  $M_0 = [2, 2, 0]$ .

Нетрудно видеть, что матричное и графовое представления взаимно однозначно соответствуют друг другу.

В случае большой кратности дуг ее можно указывать цифрами на соответствующей дуге.

## Пространство состояний сети Петри

Состояние сети Петри определяется ее маркировкой. Пространство состояний сети Петри, обладающей  $n$  позициями, есть множество всех маркировок, то есть  $E^n$ .

Изменение в состоянии, вызванное запуском перехода, определяется функцией перехода  $\delta$  или функцией следующего состояния. Когда эта функция применяется к маркировке  $M$  и переходу  $t_j$  (если он разрешен), то в соответствии с (13.1) получается новая маркировка  $M' = \delta(M, t_j)$ . Она, как уже говорилось, получается изъятием фишек из позиции  $p_i$  таких, что  $f_{ij}^p \neq 0$  ( $\mu_i \geq f_{ij}^p$ ) и помещением фишек в позиции  $p_k$  такие, что  $f_{ik}^t \neq 0$ .

Процесс создания новых маркировок продолжается до тех пор, пока в сети Петри при данной маркировке существует хоть один разрешенный переход. Если же при некоторой маркировке ни один переход не разрешен, то такая маркировка называется тупиковой.

При выполнении сети Петри получается две последовательности:

- последовательность маркировок  $\{M(0), M(1), M(2), \dots\}$ ;
- последовательность запущенных переходов  $\{t_{j0}, t_{j1}, t_{j2}, \dots\}$ .

Эти две последовательности связаны следующим соотношением:

$$M(\theta + 1) = \delta(M(\theta), t_{j\tau})$$

Если в результате запуска перехода при маркировке  $M$  образуется новая маркировка  $M'$ , то говорят, что  $M'$  достижима из  $M$ .

Множество достижимости  $R(\text{PN}, M)$  сети Петри PN с маркировкой  $M$  есть множество всех  $M_k$ , достижимых из  $M$ .

Маркировка  $M'$  принадлежит  $R(\text{PN}, M)$ , если существует какая-либо последовательность запусков переходов, изменяющих  $M$  на  $M'$ .

Множество достижимости  $R(\text{PN}, M)$  для сети  $\text{PN} = \{\Theta, P, T, F, M_0\}$  с маркировками  $M$  есть наименьшее множество маркировок, определенных следующим образом:

- $M' \in R(\text{PN}, M)$ ;
- если  $M' \in R(\text{PN}, M)$  и  $M'' = \delta(M', t_j)$  для некоторого  $t_j \in T$ , то  $M'' \in R(\text{PN}, M)$ .

Вернемся к примеру на рисунке 13.1. При начальной маркировке  $M_0 = [2, 2, 0]$  могут сработать переходы  $t_1$  (в результате получаем  $M'_1 = [2, 3, 0]$ ) и  $t_2$  (получается маркировка  $M''_1 = [1, 0, 1]$ ). Каждая из полученных маркировок порождает новые, в результате чего получается *дерево маркировок*, фрагмент которого показан на рисунке 13.2. Обратим внимание на то, что в дереве

маркировок могут встречаться повторяющиеся маркировки. В этом случае дальнейшее построение дерева ведется только для одной из них.

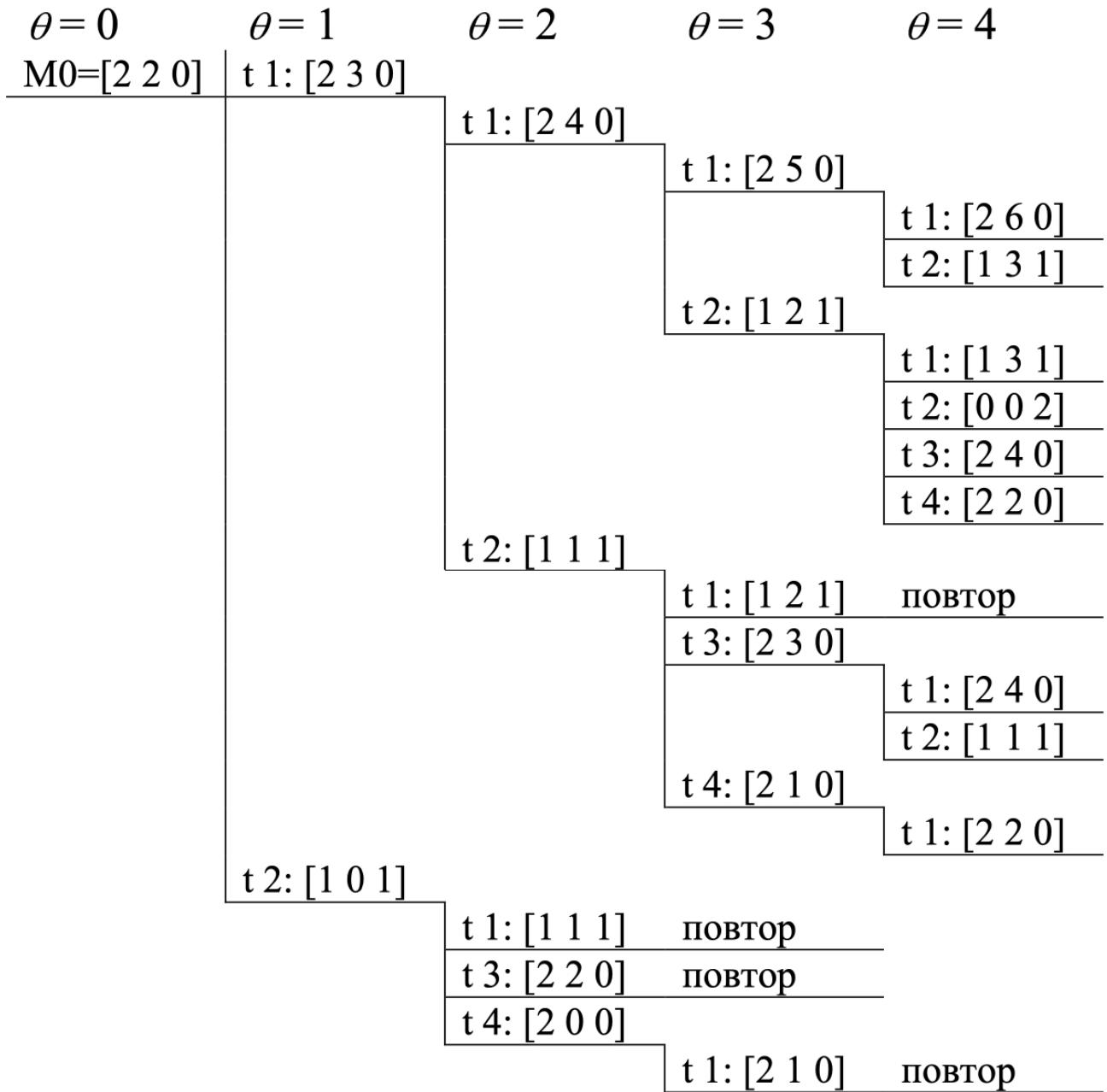


Рис. 13.2: Пример дерева сети Петри

Если выделить путь по дугам графа маркировок, начинающийся в вершине  $M_0$  и заканчивающийся в различных вершинах  $M'$ , и выписать подряд все встречающиеся символы переходов, то полученное слово образует последовательность срабатываний сети, а их совокупность — *свободный язык сети Петри*  $L(PN, M_0)$ .

Так, язык рассматриваемой сети включает слова

$$\begin{aligned} \{\lambda, t_1, t_2, t_1t_1, t_1t_2, t_2t_1, t_2t_3, t_2t_4, t_1t_1t_1, t_1t_1t_2, t_1t_2t_1, \\ t_1t_2t_3, t_1t_2t_4, t_2t_1t_1, t_2t_3t_1, t_2t_4t_1, t_1t_1t_1t_1, t_1t_1t_1t_2, \\ t_1t_1t_2t_1, t_1t_1t_2t_3, t_1t_1t_2t_4, t_1t_2t_1t_1, \dots\} \end{aligned}$$

Здесь  $\lambda$  — пустой символ, соответствующий начальной маркировке  $M_0$ .

## Основные свойства сетей Петри

Исходя из практических задач моделирования, можно установить ряд свойств сетей Петри, характеризующих поведение моделируемых систем.

### Свойство ограниченности

Позиция  $p_i$  в сети PN =  $\{\Theta, P, T, F, M_0\}$  называется *ограниченной*, если для любой достижимой в сети маркировки  $M$  существует такое  $k$ , что  $\mu_i \leq k$ . Сеть PN называется ограниченной, если все ее позиции ограничены. Сеть, показанная на рисунке 13.1, не ограничена, так как возможен неограниченный рост  $\mu_2$ . Для обозначения неограниченной маркировки используется специальный символ  $\omega$ . Так, на дереве маркировок (рисунок 13.2) можно выделить маркировку  $M = [1, \omega, 0]$ .

### Свойство безопасности

Сеть PN называется *безопасной*, если при любой достижимой маркировке  $\mu_i \leq 1$  для всех  $i = \overline{1, n}$ . Таким образом, в безопасной сети вектор маркировок состоит только из нулей и единиц (является двоичным словом).

### Свойство консервативности

Сеть называется *консервативной*, если сумма фишечек во всех позициях остается постоянной при работе сети:

$$\sum_{i=1}^n \mu_i(\theta) = \text{const}, \quad \theta = 0, 1, \dots$$

### Свойство живости

Рассмотрим теперь свойства переходов. Переход  $t_j$  в сети PN =  $\{\Theta, P, T, F, M_0\}$  называется *потенциально живым*, если существует достижимая из  $M_0$  маркировка  $M'$ , при которой  $t_j$  может сработать. Если  $t_j$  является потенциально

живым при любой достижимой в PN маркировке, то он называется *живым*. Переход  $t_j$ , не являющийся потенциально живым при начальной маркировке  $M_0$ , называется *мертвым* при этой маркировке. Маркировка  $M_0$  в этом случае называется  $t_j$ -*тупиковой*. Если маркировка  $M_0$  является  $t_j$ -тупиковой для всех  $j = \overline{1, m}$ , то она называется *тупиковой*. При тупиковой маркировке не может сработать ни один переход. На дереве маркировок тупиковая маркировка является листом.

Переход называется *устойчивым*, если никакой другой переход не может лишить его возможности сработать при наличии для этого необходимых условий.

Последовательность маркировок  $M_0, M_1, \dots, M_p$ , в которой  $M_{k+1} = \delta(M_k)$ ,  $k = \overline{0, p}$  образует *цикл*, если  $M_0 = M_p$ . Каждому циклу соответствует последовательность слов свободного языка сети Петри.

## Некоторые обобщения сетей Петри

Рассмотренное выше базовое определение сети Петри позволяет моделировать широкий класс дискретных систем. Однако в ряде случаев этих возможностей оказывается недостаточно, поэтому вводят обобщения этих сетей, которые обладают расширенными возможностями моделирования. Упомянем некоторые из них.

### Ингибиторные сети

Ингибиторные сети (ИСП, IPN) — это сети Петри, для которых функция инцидентности имеет вид  $F = F^p \cup F^t \cup F^I$ , то есть она дополнена специальной функцией инцидентности  $F^I: P \times T \rightarrow \{0, 1\}$ , которая вводит ингибиторные дуги для тех пар  $(p_i, t_j)$ , для которых  $f_{ij}^I = 1$ . Ингибиторные дуги связывают только позиции с переходами, на рисунках заканчиваются не стрелками, а кружочками (рисунок 13.3). Кратность этих дуг всегда равна 1.

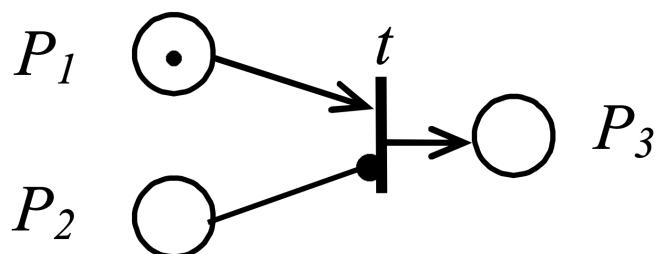


Рис. 13.3: Ингибиторная дуга

Правила срабатывания переходов в ингибиторной сети модифицируются следующим образом. Переход  $t_j$  может сработать при маркировке  $M$ , если для

всех связанных с ним позиций  $p_i$  и  $p_k$  выполняется (13.2):

$$(\mu_i \geq f_{ij}^p) \wedge (\mu_k \cdot f_{kj}^I = 0) \quad (13.2)$$

Так, переход  $t$  на рисунке 13.3 может срабатывать только при  $\mu_1 > 0$  и  $\mu_2 = 0$ .

### Сети с приоритетами

При определении сети Петри отмечалась недетерминированность ее работы: если имеется возможность срабатывания нескольких переходов, то срабатывает любой из них. При моделировании реальных систем могут сложиться ситуации, когда последовательность срабатываний необходимо регламентировать. Это можно сделать, введя множество приоритетов  $PR = T \rightarrow \{0, 1, \dots\}$  и приписав каждому из переходов  $t_j$  соответствующее целочисленное значение приоритета  $pr_j$ . Тогда правило срабатывания переходов модифицируется: если на некотором такте работы сети PN имеется возможность для срабатывания нескольких переходов, то срабатывает тот из них, который имеет наивысший приоритет. Так, из двух готовых к срабатыванию переходов  $t_1$  и  $t_2$  на рисунке 13.4а первым должен сработать переход  $t_2$ , имеющий приоритет  $pr_2 = 1$ , поскольку приоритет перехода  $t_1$   $pr_1 = 2$ , то есть ниже.

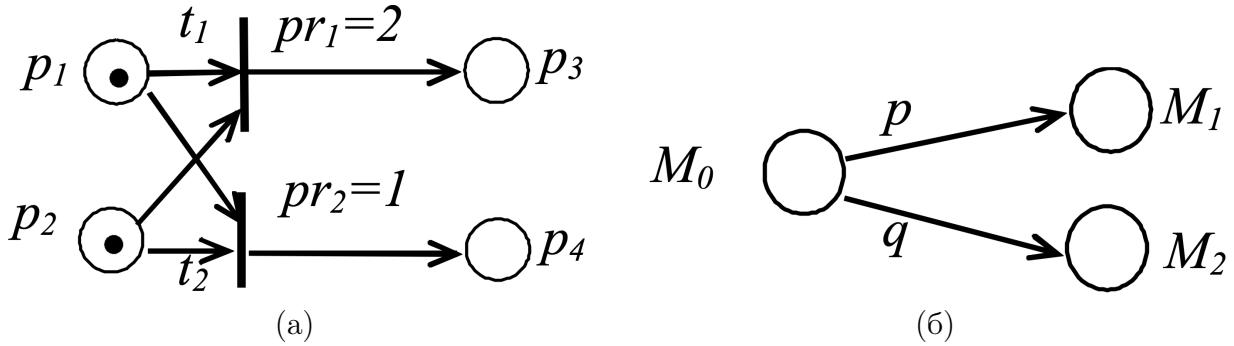


Рис. 13.4: Пример сети с приоритетами

### Сети со случайными срабатываниями переходов

В описанной выше ситуации, когда имеется возможность срабатывания нескольких переходов  $t_i, t_j, \dots, t_s$ , их приоритет можно задавать вероятностями срабатывания каждого из переходов  $p_i, p_j, \dots, p_s$ , причем  $p_i + p_j + \dots + p_s = 1$ . Тогда исходная маркировка  $M_0$  приведет на следующих шагах работы сети к набору маркировок  $M_i, M_j, \dots, M_s$ , каждая из которых будет помечена соответствующей вероятностью. Отождествив маркировки с состоянием сети и положив, что вероятности не зависят от работы сети в предыдущие такты, мы получим цепь Маркова, описывающую вероятностное поведение системы.

Пусть на рисунке 13.4а вероятность срабатывания перехода  $t_1$  равна  $p$ , а вероятность срабатывания  $t_2$  равна  $q = 1 - p$ . Тогда, обозначив маркировки  $M_0 = [1100]$ ,  $M_1 = [0010]$ ,  $M_2 = [0001]$ , получим цепь Маркова (рисунок 13.4б).

## Иерархические сети

Иерархические сети Петри представляют собой многоуровневые структуры, в которых выделяются сети различного уровня. Они позволяют моделировать различные многоуровневые (иерархические) системы.

В отличие от обычных сетей Петри, в иерархических сетях имеются два типа переходов: *простые* и *составные*. Простые переходы ничем не отличаются от рассмотренных ранее, а составные переходы содержат внутри себя сеть Петри более низкого уровня. Формально они состоят из входного («головного») и выходного («хвостового») переходов, между ними находится некоторая сеть Петри, которая, в свою очередь, также может быть иерархической.

Пример иерархической сети  $N$ , в которой имеется составной переход  $t_2$ , содержащий внутри себя сеть  $N'$ , показан на рисунке 13.5. Составной переход  $t_2$  имеет «голову»  $t'_2$  и «хвост»  $t''_2$ , между которыми заключена сеть  $N'$ , состоящая из позиций  $p_{21}-p_{25}$  и переходов  $t_{21}-t_{23}$ .

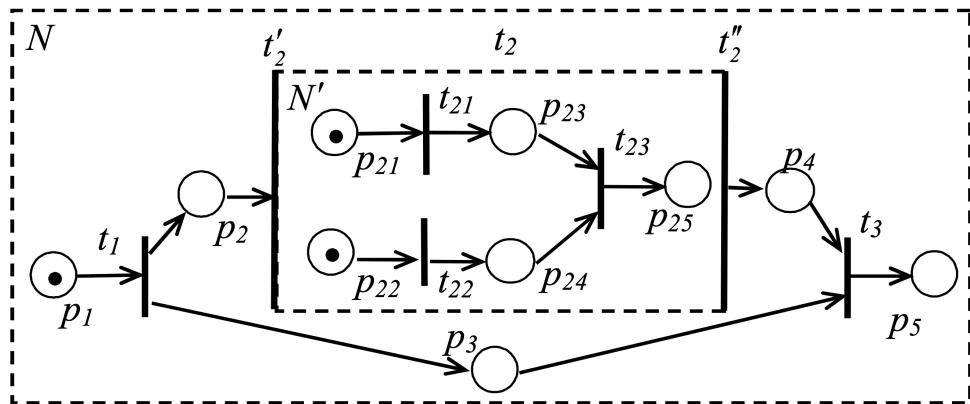


Рис. 13.5: Пример иерархической сети Петри

Иерархическая сеть функционирует, как и обыкновенная СП, переходя от одной маркировки к другой и обмениваясь фишками (в том числе между сетями различного уровня). Исключение составляют правила работы составных переходов.

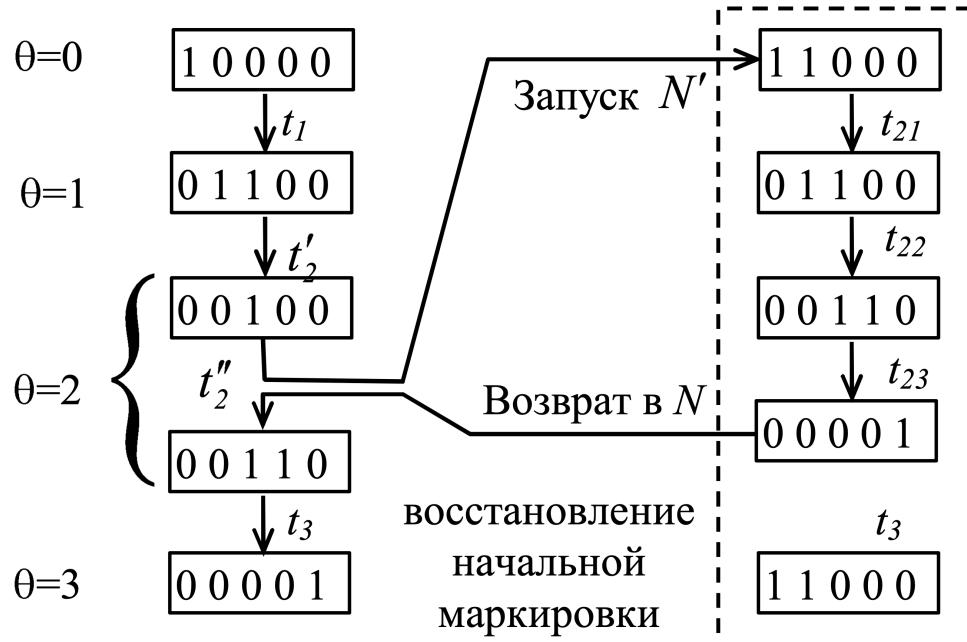
Срабатывание составных переходов является не мгновенным событием, как в обычных СП, а составным действием. Поэтому говорят не о срабатывании составного перехода, а о его работе. На каждом шаге дискретного времени  $\theta$  составной переход может находиться в одном из двух состояний — пассивном и активном. Начальное состояние всех переходов — пассивное. Составной

переход может быть активирован в момент времени  $\theta$ , если он до этого был пассивен, и имеются условия для срабатывания его головного перехода. При этом производится изменение маркировки в сети верхнего уровня по обычным правилам (13.1), и одновременно запускается работа в сети, находящейся внутри составного перехода. Во время ее работы функционирование сети верхнего уровня блокируется. Сеть нижнего уровня работает с учетом своей начальной маркировки до тех пор, пока все ее переходы не станут пассивными (то есть не смогут сработать). После этого происходит срабатывание хвостового перехода и изменение маркировки сети верхнего уровня согласно (13.1). Составной переход возвращается в пассивное состояние, а в сети нижнего уровня восстанавливается начальная маркировка.

Ниже приведено дерево маркировок сетей  $N$  и  $N'$  при указанных на рисунке 13.5 начальных маркировках.

Мы видим, что на шаге  $\theta = 2$  происходит работа составного перехода  $t_2$  и сети  $N'$  в следующем порядке: срабатывание  $t'_2 \rightarrow$  запуск  $N' \rightarrow$  окончание работы  $N'$  и восстановление ее начальной маркировки  $\rightarrow$  срабатывание  $t_2$  и продолжение работы сети  $N$ .

Отметим также, что описанный процесс напоминает выполнение подпрограммы при программировании на алгоритмических языках. Срабатывание перехода  $t'_2$  и соответствует вызову подпрограммы, а срабатывание  $t''_2$  — возврату в основную программу.



## Раскрашенные (цветные) сети Петри

Раскрашенные сети Петри (PCP, CPN – Coloured Petri Net). В ряде приложений перемещаемые в сети Петри ресурсы (фишки) требуется дифференцировать, и тогда приходится вводить фишки различных видов (например, разных цветов). В этом случае для каждого перехода необходимо указывать, при каких комбинациях фишек во входных позициях он может сработать, и какое количество фишек различных цветов помещается в выходные позиции.

Понятно, что моделирующая возможность раскрашенных сетей Петри выше, чем у обычных, однако при этом значительно усложняется их описание.

Следует отметить, что раскрашенные сети Петри могут быть преобразованы в обычные, но при этом возрастают размеры сети.

Упомянем еще некоторые расширения сетей Петри: синхронные, самомодифицируемые (с изменяющейся кратностью дуг, когда  $F^p$  и  $F^t$  зависят от  $\tau$ ), и другие.

## Лекция №14 (18.12.2023)

### Моделирование дискретных систем

Сети Петри были разработаны и используются для моделирования сложных систем. С помощью различных модификаций этих сетей можно создать концептуальные модели и реализовать функционирование таких моделей <...> систем с независимыми элементами:

- аппаратное и программное обеспечение;
- системы телекоммуникаций;
- физические, химические, социальные процессы.

При описании функционирования системы с помощью сетей Петри выделяют два понятия.

**Определение.** Событие — это действие в системе. В сетях Петри моделируются *переходами*.

**Определение.** Условие — это предикат или логическое описание системы, принимающее значение «истина» или «ложь». Условия моделируются *позициями*. Различают пред- и постусловия.

**Определение.** Предусловие — это условие до срабатывания перехода.

**Определение.** Постусловие — это условие после срабатывания перехода.

**Определение.** Переходы  $t_i$  и  $t_j$  находятся в *конфликте*, если запуск одного из них блокирует запуск другого.

### Простейшая система массового обслуживания

Система имеет входной поток заданий. Пока она занята выполнением очередного задания, она не может ввести следующее задание.

Рассмотрим множество условий и событий, характеризующих данную систему:

- условия:
  - $p_1$  — задание ждет обработки;

- $p_2$  — задание обрабатывается;
  - $p_3$  — процессор свободен;
  - $p_4$  — задание ожидает вывода.
- события:
- $t_1$  — задание помещается во вторую очередь;
  - $t_2$  — начало выполнения задания;
  - $t_3$  — конец выполнения задания;
  - $t_4$  — задание выводится.

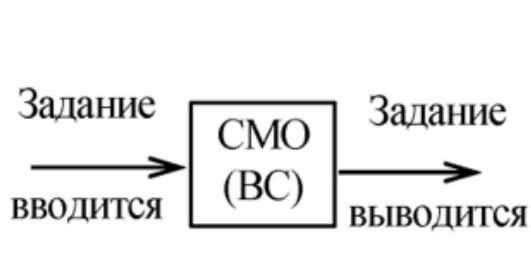
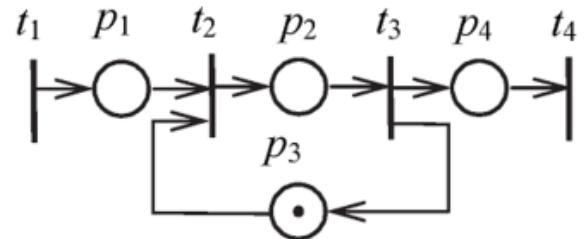
*a**b*

Рис. 14.1: Сеть Петри (однопоточная СМО)

Показанная на рисунке начальная маркировка соответствует состоянию, когда система свободна, а заявки обслужены ( $M_0 = [0, 0, 0, 0]$ ). При срабатывании перехода  $t_1$  от внешнего источника поступает задание ( $M_1 = [0, 1, 0, 1]$ ). При этом может сработать переход  $t_2$ , что означает начало обслуживания задания и приводит к маркировке  $M_2 = [0, 1, 0, 0]$ . Затем срабатывает переход  $t_3$ , то есть окончание обслуживания задания и освобождение системы ( $M_3 = [0, 0, 1, 1]$ ).

Переходы  $t_1$  и  $t_4$  могут работать независимо.

Конвейер состоит из следующих этапов:

- сравнение порядков;
- выравнивание порядков;
- сложение мантисс;
- нормализация результата.

Условия:

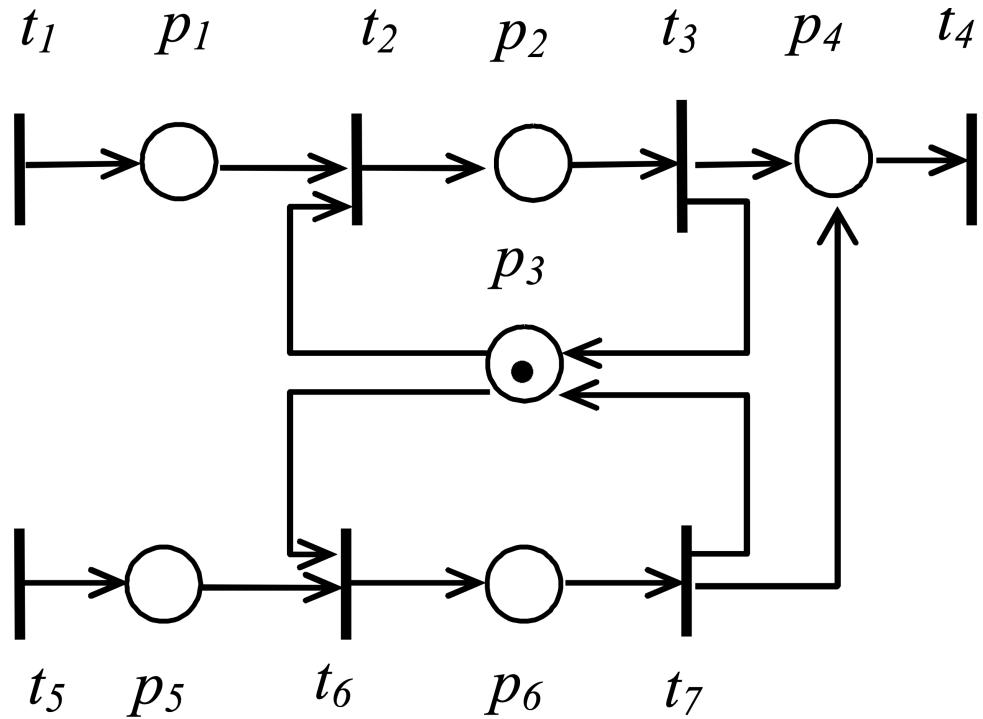


Рис. 14.2: Сеть Петри (двуихпоточная СМО)

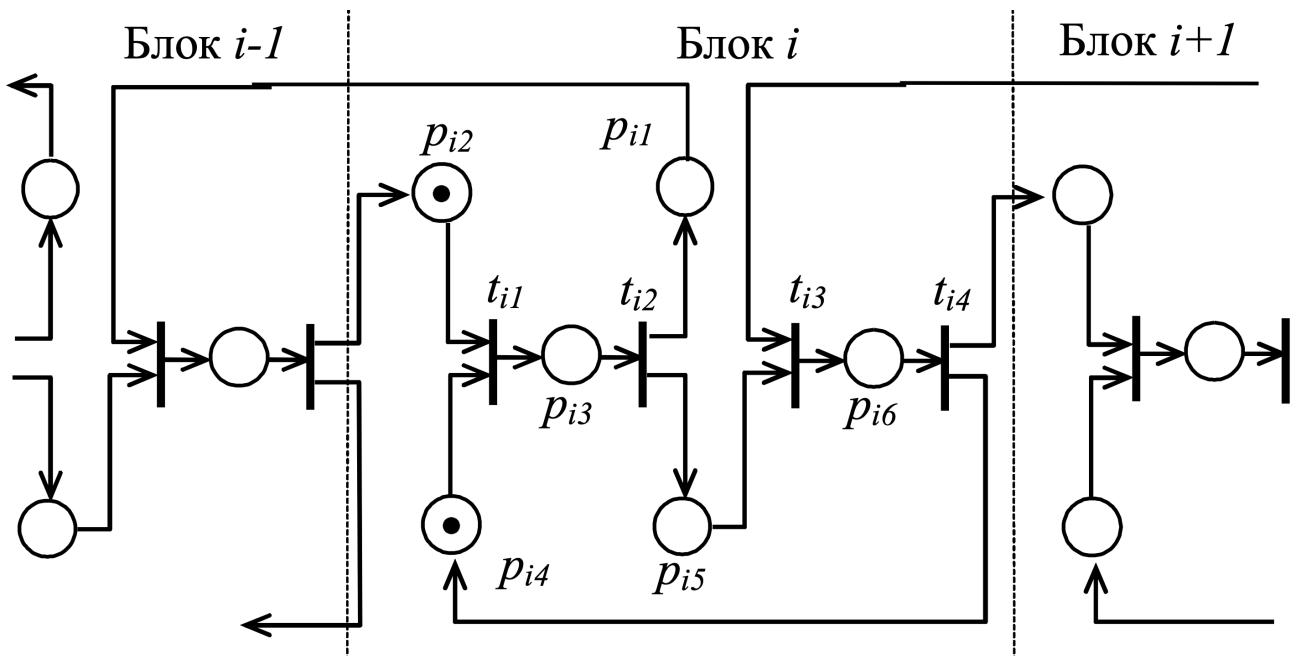


Рис. 14.3: Сеть Петри (конвейер)

- $p_{i1}$  — входной регистр свободен;
- $p_{i2}$  — входной регистр заполнен;
- $p_{i3}$  — блок занят;
- $p_{i4}$  — выходной регистр свободен;

- $p_{i5}$  — выходной регистр заполнен;
- $p_{i6}$  — пересылка в следующий блок возможна.

События:

- $t_{i1}$  — начало работы  $i$ -го блока;
- $t_{i2}$  — завершение работы  $i$ -го блока;
- $t_{i3}$  — начало пересылки в  $(i + 1)$ -ый блок;
- $t_{i4}$  — завершение пересылки в  $(i + 1)$ -ый блок.