

Лекция 2

Преимущества моделирования

1. Гораздо дешевле
2. Можно реализовать фантастические условия

Философские аспекты моделирования

Методологическая основа моделирования – диалектический метод познания и научного исследования.

Всё то, на что направлена человеческая деятельность — объект.

Научно-техническое развитие в любой области обычно идёт следующим путём:

- наблюдение и эксперимент;
- теоретическое исследование;
- организация производственных процессов.

В научных исследованиях большую роль играет такая категория как **гипотеза**, то есть определенное предсказание, основывающееся на небольшом количестве опытных данных, наблюдениях, догадок. Быстрая и полная проверка гипотезы может быть проведена в ходе специально поставленного эксперимента.

Аналогия (важное понятие для методологов основы моделирования) – суждение о каком-либо частном сходстве или различии двух объектов.

Современная гипотеза формируется по аналогии:

Гипотеза → Аналогия → Эксперимент

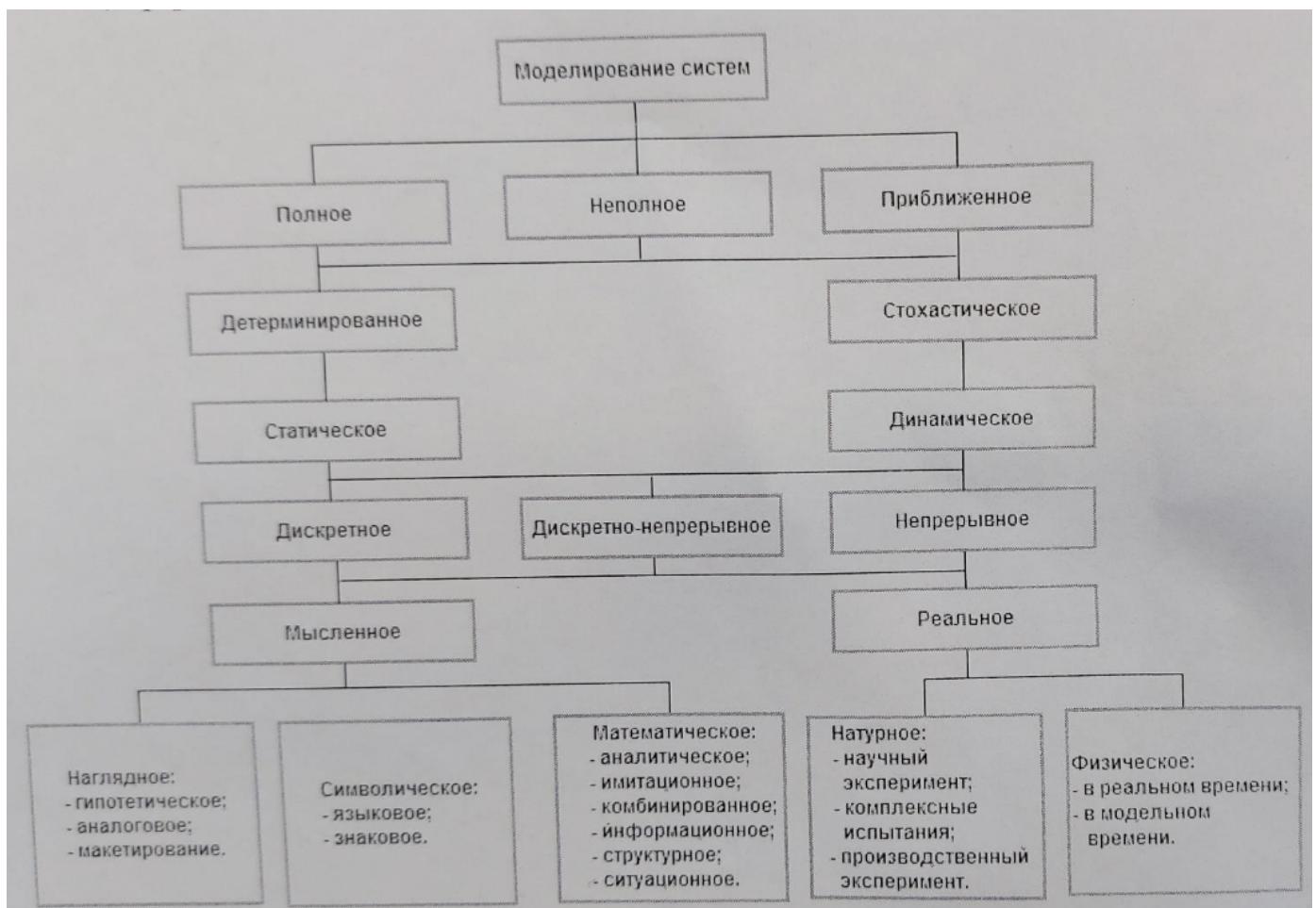
Гипотезы и аналогии, отражающие реальный, объективно существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам. Такие логические схемы, упрощающие рассуждения и логические построения или позволяющие проводить эксперименты, уточняющие

природу явлений, называются *моделями*.

Модель – объект-заместитель объекта оригинала, обеспечивающий изучение некоторых свойств оригинала.

Замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели называется **моделированием**. В основе моделирования лежит теория подобия, которая утверждает, что абсолютное подобие имеет место только при замене объекта таким же объектом.

Классификация видов моделирования



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

Для аналитического моделирования характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраических, интегро-дифференциальных, конечно-разностных и т.д.).

Аналитическая модель может быть исследована тремя способами.

1. Аналитический – стремится получить в общем виде в зависимости от исходных характеристик.
2. Численный метод – нельзя решить уравнение в общем виде, получаем результаты для конкретных начальных данных.
3. Качественный метод – когда, не имея решения можно найти свойства этого решения.

При имитационном моделировании реализующий модель алгоритм воспроизводит процесс функционирования системы, причем имитируются элементарные явления, составляющие процесс с сохранением их логической структуры и последовательности протекания, что позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени, дающие возможность оценить характеристики системы.

Лекция 3

Виды имитационного моделирования

1. **Агентное моделирование** – используется для исследования децентрализованных систем, динамика которых определяется не глобальными правилами и законами, как в других парадигмах моделирования, а, можно сказать, наоборот, когда эти глобальные правила и законы являются результатом индивидуальной активности членов групп.

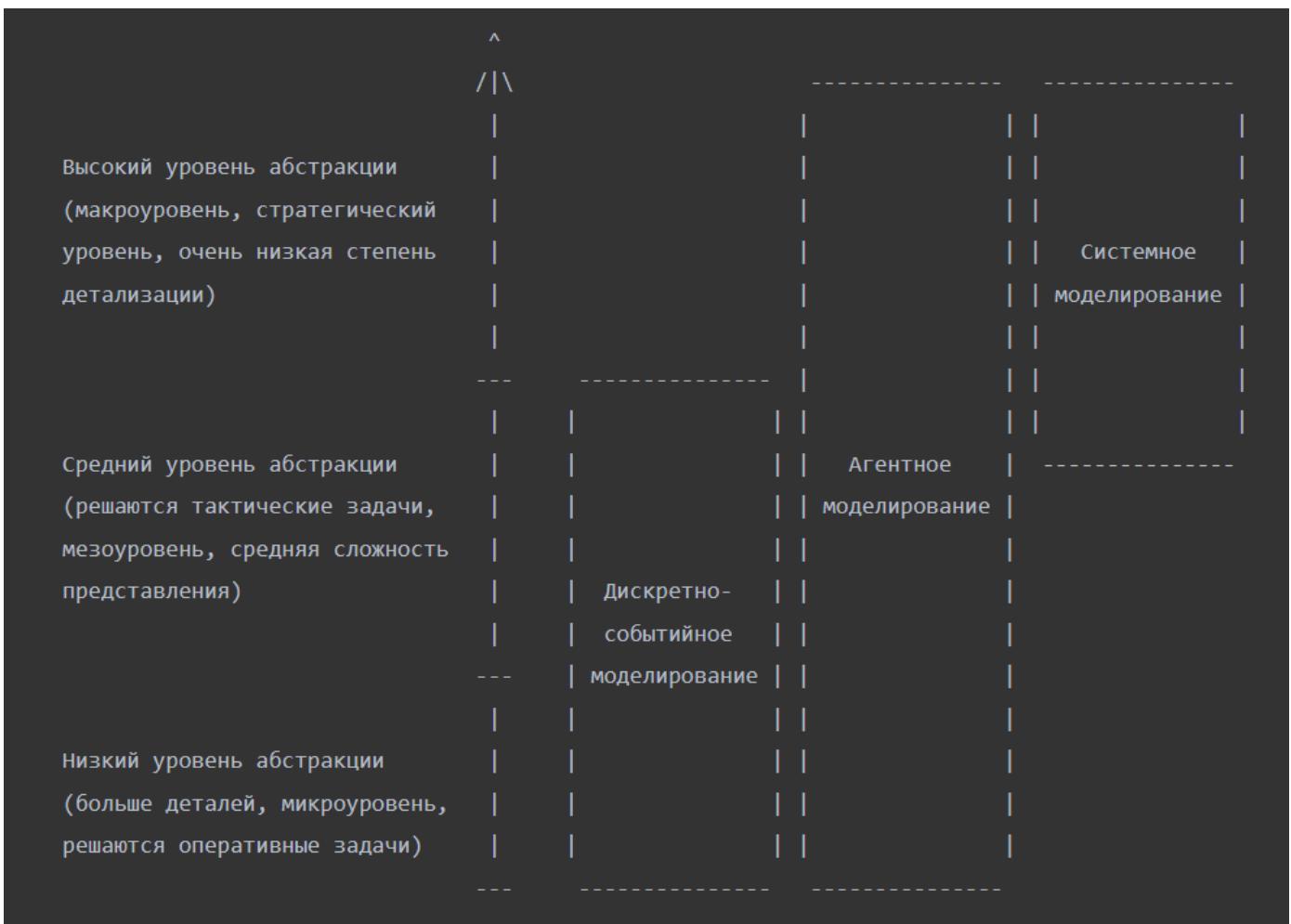
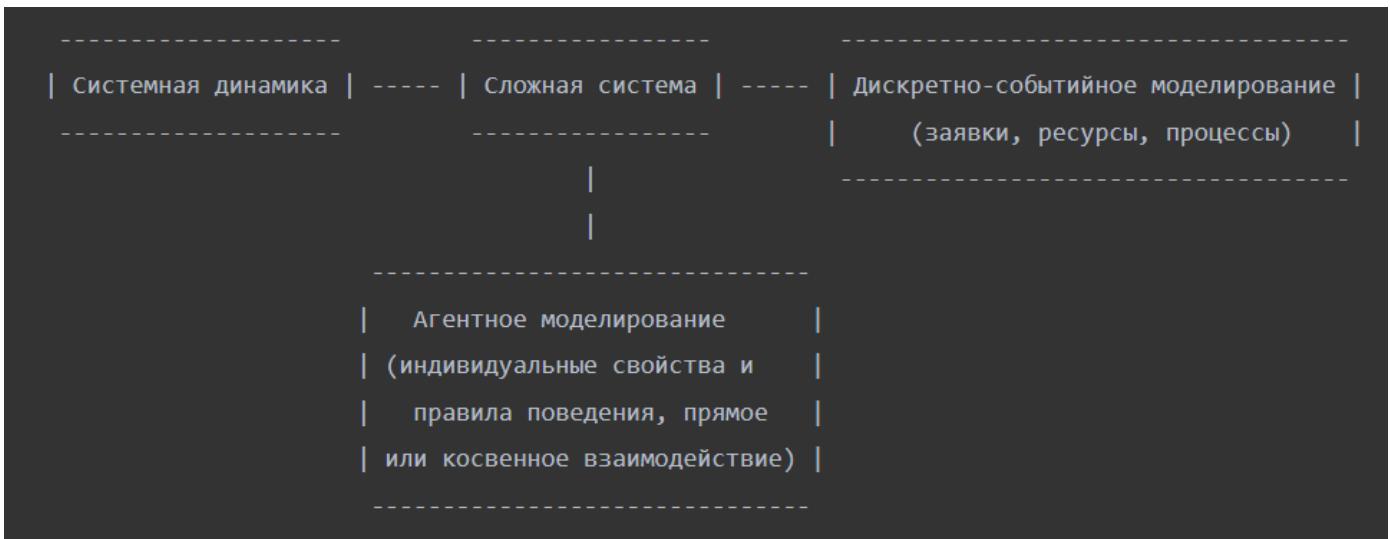
Следовательно, **цель** агентных моделей – получить представление об этих глобальных правилах, общем поведении системы исходя из предположений об индивидуальном частном поведении ее отдельных активных объектов. И взаимодействие этих объектов в системе.

Таким образом, **агент** – это некая сущность, обладающая активностью, автономным поведением, может принимать решения в соответствии с выбором некоторых правил, взаимодействовать с окружением, а также самостоятельно изменяться.

Главное свойство – активность поглощает модельное время.

2. **Дискретно-событийное моделирование** – предполагается абстрагироваться от непрерывной природы событий и рассматривать только основные события, происходящие в системе. Например, ожидание, обработка какого-то заказа и другие логистические проблемы. Оно наиболее развито и практически везде имеет приложение – от логистики и системы массового обслуживания до транспортных и производственных систем.

3. **Системная динамика** – для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие во времени, а затем созданная диаграмма моделируется с помощью компьютера. Например, бизнес-процессы.



Недостатки имитационного моделирования

Трудности имитационного моделирования связаны с обеспечением адекватности описания системы, интерпретации результатов, обеспечении стохастической сходимости моделирования и т.д., а также с большой трудоемкостью метода. Очень часто перед построением имитационной модели, являющейся

динамической по своей сути, оказывается полезным, а иногда и просто необходимым осуществить предварительный статический анализ исследуемой системы. При этом выявляются и специфицируются функции, выполняемый в системе, их взаимосвязи, потоки работ и т.д. Именно эти спецификации позволяют нашей модели выявить некоторый определенный объем знаний о всей системе до создания полной имитационной модели. Тем самым уменьшаем только вероятность ошибок. Для выполнения такого анализа используем кейс-технологии (предварительная оценка, можем определить жизненный цикл).

Выделяет три этапа в развитии средств компьютерного моделирования. В качестве признаков рассмотрим возможности и функции исследователя.

1. Создание имитационной модели с помощью языков:

- на универсальном языке (н-р, C++),
- специфицированном языке имитационного моделирования (н-р, GPSS),
- на объектно-ориентированном языке имитационного моделирования (н-р, ModelSim);

2. Использование при разработке имитационных моделей проблемно-ориентированных систем и средств (Matlab). Не требуют от пользователя знаний программирования и позволяют моделировать лишь относительно узкие классы сложных систем. При этом имитационная модель генерируется самой системой. Самое главное, что мы сами участвуем в создании настраиваемой модели

3. Использование собственного искусственного интеллекта. Речь идет о знаниях при принятии решений в процессе управления моделью при имитационном эксперименте. Самый важный этап – формализация модели.

Как используем вычислительную технику в моделировании

1. Как средство расчета по полученным аналитическим моделям
2. Как средство имитационного моделирования

Вычислительная техника

1. Цифровая
2. Аналоговая
3. Гибридная

Центральный процессор

Центральный процессор – устройство, предназначенное для реализации множества машинных команд и управления вычислительными процессами.

Лекция 4

Два основных использования технических средств

1. Как средство расчета по полученным аналитическим моделям (ЭВМ и АВМ)
2. Средство имитационного моделирования

АВМ – ускоряет процесса решения, но не обеспечивает высокую точность.

Гибридно-вычислительные комплексы соединяют в себе высокую точность и для некоторых объектов повышают их скорость.

Гибридно-вычислительные машины объединяют в себе узлы и блоки типовых и специализированных вычислительных машин с использованием различных форм представления информации и различных методов ее переработки. Для гибридно-вычислительной техники типичными являются различные преобразования формы представления информации (прецизионные коммутаторы, запоминающие устройства непрерывных сигналов, различные компараторы и т.д.).

Применение вычислительных гибридных машин

- Моделирование дискретных систем и случайных процессов
- Решение задачи оптимизации (в том числе многокритериальных)
- Для исследования, управления подвижным объектом

Основные направления гибридных вычислительных машин

1. На основе дискретно-управляемых элементов, меняющие свои параметры под воздействием управляемого кода
2. Разрядно-аналоговые – обеспечивают высокую точность и быстродействие за счет цифровой формы представления информации и аналогового способа ее переработки

3. Цифровые интегрирующие машины - фактически спец. ЭВМ, но, в отличие от цифровой техники, в качестве основной операции - интегрирование

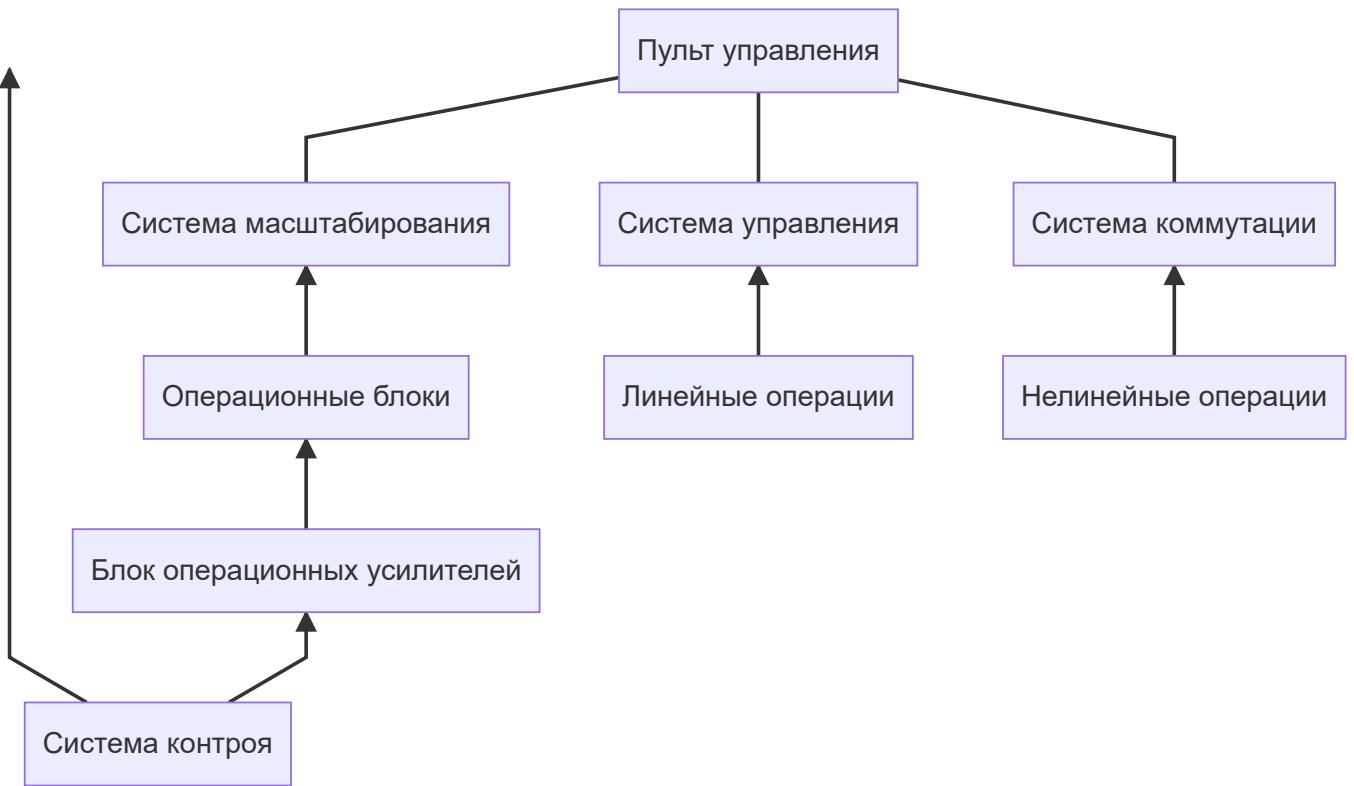
Аналоговые вычислительные машины (АВМ)

В отличие от дискретной (принцип счета), в основе аналоговой вычислительной технике заложен принцип **моделирования**. При использовании в качестве модели некоторой задачи электронных цепей каждой переменной величине задачи ставится в соответствие определенная переменная величина электрической цепи. При этом основой построения такой модели является **изоморфизм** (подобие исследуемой задачи и соответствующей ей электронной модели). В большинстве случаев при определении критериев подобия используются специальные приемы **масштабирования** соответствующих значений параметров модели и переменных нашей задачи. *АВМ реализует изоморфную модель исследуемой задачи.* Согласно своим вычислительным возможностям, АВМ наиболее приспособлена для исследования объектов, динамика которых описывается обыкновенными и в частных производных дифференциальными уравнениями, а также алгебраическими и др. Следовательно под **АВМ** будем понимать совокупность электрических элементов, организованных в систему, позволяющих изоморфно моделировать динамику изучаемого объекта.

АВМ делят на (по их возможности решать задачи, описываемые уравнениями n-го порядка):

- малые (меньше 10)
- средние (от 10 до 20)
- большие (больше 20)

Структурная схема АВМ



Под **гибридной вычислительной машиной** будем понимать широкий класс вычислительных систем, использующих как аналоговые, так и дискретные формы представления и обработки информации.



Подклассы гибридных вычислительных машин

1. АВМ, использующие цифровые методы численного анализа
2. АВМ, программируемые с помощью ЦВМ
3. АВМ с цифровым управлением и логикой
4. АВМ с цифровыми элементами

Сравнительная характеристика АВМ и ЦВМ

Показатель	АВМ	ЦВМ
Тип информации	Непрерывный	Дискретный
Изменение значений	Величиной напряжения	Числовым значением
Базовые операции	Интегрирование	Суммирование
Принцип вычислений	Высоко-параллельный	Последовательно-параллельный
Режим реального времени	Без ограничений	Ограниченные возможности
Динамическое изменение решаемой задачи	По средствам системам коммутации	В диалоговом режиме
Основные профессиональные требования пользователей	Профессиональные знания в области ИТ, методика моделирования	Алгоритмизация
Уровень формализации задачи	Ограничен моделью решаемой задачи	От самого низа до самого верха (высочайший уровень)
Способность решать логические задачи	Ограниченные	Высочайшие
Точность	Меньше, чем в 10^{-4}	Зависит от разрядной сетки
Диапазон	От 1 до 10^{-4}	Зависит от разрядной сетки
Класс решаемых задач	Алгебраические и дифференциальные уравнения	Любые задачи
Специальные функции	Ограниченный набор	Широкий класс
Уровень миниатюризации	В карман не поместится	Высочайший уровень
Сфера применения	Ограниченная	Почти везде
Пользовательский интерфейс	Очень низкий уровень	Интуитивно понятный графический интерфейс

Лекция 5

Базовые понятия в теории систем

Система – это множество элементов, находящихся в отношениях и связях между собой.

Элемент – часть системы, представление о которой нецелесообразно подвергать дальнейшему членению.

Сложная система – это система, характеризующаяся большим числом элементов (2 или более) и, что наиболее важно, большим числом взаимосвязей элементов. Сложность системы определяется также видом взаимосвязей элементов, свойствами целенаправленности, целостности, членности, *иерархичности*, многоаспектности.

Подсистема – часть системы (подмножество элементов и их взаимосвязи), которая имеет свойства системы.

Надсистема – система, по отношению к которой рассматриваемая система является подсистемой.

Структура – отображение совокупности элементов системы и их взаимосвязи. Понятие структуры отличается от понятия системы также тем, что при описании структуры принимают во внимание лишь типы элементов и их связи без конкретизации значений их параметров.

Параметр – величина, выражающая свойства или системы, или ее части, или влияющие на систему среды.

Характеристики сложной системы

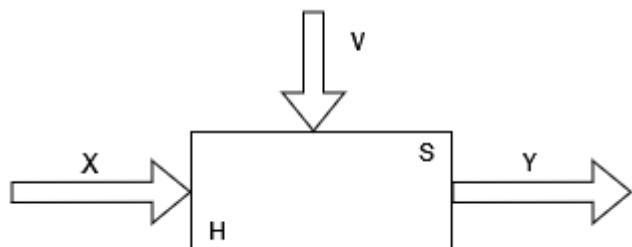
1. Целенаправленность – свойство искусственной системы, выражающее назначение системы (необходима для оценки эффективности вариантов системы).

2. Целостность – свойство системы, характеризующее взаимосвязанность элементов и наличие зависимости выходных параметров от параметров элементов. При этом большинство выходных параметров не являются простым повторением или суммой параметров элементов.
3. Иерархичность – свойство сложной системы, выражающее возможность и целесообразность ее иерархичного описания, то есть представление в виде нескольких уровней, между компонентами которых имеются отношения "целые – часть".

Задачи сложной системы

1. Modeling – создание моделей сложных систем.
2. Simulation – анализ свойств систем на основе исследования их моделей.

Основы теории моделирования



Модель объекта можно представить как совокупность множества величин, описывающих процесс функционирования реальной системы и образующие в общем случае следующие подмножества:

- совокупность входных воздействий ($x_i \in X, i = \overline{1, n_X}$)
- совокупность воздействий внешней среды ($v_i \in V, i = \overline{1, n_V}$)
- совокупность собственных (внутренних) параметров системы ($h_i \in H, i = \overline{1, n_H}$)
- совокупность выходных характеристик системы ($y_i \in Y, i = \overline{1, n_Y}$)

В общем случае x_i , v_i , h_i , y_i являются элементами непересекающихся подмножеств и содержат как недетерминированные, так и стохастические составляющие. При моделировании системы S входные воздействия, воздействия внешней среды и внутренние параметры являются независимыми – экзогенными, которые в общем случае имеют вид:

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

А выходные характеристики системы являются зависимыми, то есть эндогенными элементами:

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

Процесс функционирования системы 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 является **законом функционирования сложной системы S**. В общем виде он может быть задан в виде функции, функционала, логических условий в алгоритмическом или табличном виде.

Под алгоритмом функционирования сложной системы подразумевается метод получения выходных характеристик с учетом входных воздействий $x(t)$, воздействий внешней среды $v(t)$ и внутренних параметров системы $h(t)$.

Отношение может быть получено и через понятие состояния системы (через свойства системы конкретной модели). Эти же состояния характеризуются вектором состояний:

$$\overrightarrow{z} = (z_1, z_2, \dots, z_k), k = \overline{1, n_Z}$$

Если рассматривать процесс функционирования системы S как последовательную смену состояний, то они могут быть интерпретированы как координаты точки в k -мерном фазовом пространстве, причем каждой реализации процесса будет соответствовать некоторая фазовая траектория. Совокупность всех

возможных состояний системы называется **пространством состояний объекта моделирования**.

Состояние системы (момент времени от t_0 до t_k) полностью определяется начальными условиями \vec{z}^0 :

$$\vec{z}^0 = (z_1^0, z_2^0, \dots, z_k^0),$$

где z_1^0 – свойство системы в момент времени t_0 .

Состояние системы S в момент времени t_0 определяется входными воздействиями, внутренними параметрами и воздействиями внешней среды, которые имели место за промежуток времени $t - t_0$ с помощью векторных уравнений:

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

В общем случае время модели может быть непрерывным на интервале, а может быть дискретным. Отсюда появляется понятие интервала числа интервалов (это тоже параметр системы).

Следовательно, под **математической моделью** реальной сложной системы принимают конечное множество элементов вместе с математическими связями между ними и характеристиками $\vec{y}(t)$. Это математическая схема общего вида.

Типовая математическая схема

В практике моделирования на первоначальных этапах формализации объектов используют так называемые **типовыематематические схемы**, к которым относят такие хорошо проработанные (разработанные) математические объекты, как дифференциальные алгебраические уравнения, конечные вероятностные автоматы и т.д.

Процесс функционирования системы	Типовая математическая схема	Обозначение
Непрерывно-детерминированный подход	Стандартные ДУ	D-схема

Процесс функционирования системы	Типовая математическая схема	Обозначение
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

Лекция 6. Алгоритмизация и формализация процессов функционирования сложных систем. Последовательность разработки и компьютерной реализации модели системы.

Сущность компьютерного моделирования состоит в проведении на компьютере эксперимента с моделью, которая представляет собой программный комплекс, описывающий формально или алгоритмически поведение элементов системы в процессе функционирования.

Основные требования, предъявляемые к модели

1. Полнота модели – должна предоставлять пользователю возможность получения необходимого набора характеристик, оценок системы с требуемой точностью и достоверностью.

Модель дает характеристики. Нужно, чтобы они были достоверными (правильными), должны отвечать требованиям заданной точности.

2. Гибкость модели – должна давать возможность воспроизведения различных ситуаций при варьировании структуры, алгоритмов и параметров модели (причем структура должна быть блочной, то есть допускать возможность замен, добавления, исключения некоторых частей без переделки всей модели).
3. Компьютерная реализация модели должна соответствовать имеющимся техническим ресурсам (в первую очередь, быстродействие).

Процесс моделирования, включая 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 поставленной задачи.

Основные этапы моделирования больших систем

1. Построение концептуальной (описательной) модели системы и ее формализация. Формулируется модель и строится ее формальная схема. То есть основным назначением этого этапа является *переход от содержательного описания объекта к его математической модели*. Это наиболее ответственный и наименее формализованный этап. **Исходные материалы** – содержательное описание объекта. **Последовательность действий:**

- проведение границы между системой и внешней средой;
- исследование моделируемого объекта с точки зрения выделения основных составляющих процессов функционирования системы по отношению к цели моделирования;
- переход от содержательного описания системы к формализованному описанию свойств процесса функционирования системы (собственно к концептуальной модели);

Переход от содержательного описания к формализованному в данной интерпретации сводится к исключению из рассмотрения некоторых второстепенных элементов описания.

- оставшиеся элементы модели группируются в блоки;
- Блоки первой группы – имитатор воздействия внешней среды, второй – являются собственно моделью исследуемой системы.
- процесс функционирования системы так разбиваются на подпроцессы, чтобы построение модели отдельных подпроцессов было элементарно и не вызывало особых трудностей (то, что реализуется подбором типовых математических схем).

2. Алгоритмизация модели и ее компьютерная реализация – *математическая модель, сформулированная на первом этапе, воплощается в конкретную компьютерную модель*. Должна быть блочная логическая схема нашей модели. **Последовательность действий:**

- разработка схем моделирующего алгоритма;
- разработка схем программы;

- выбор технических средств для реализации компьютерной модели;
- программирование и отладка;
- тестирование и отладка (второе включает в себя первое);
- составление технической документации.

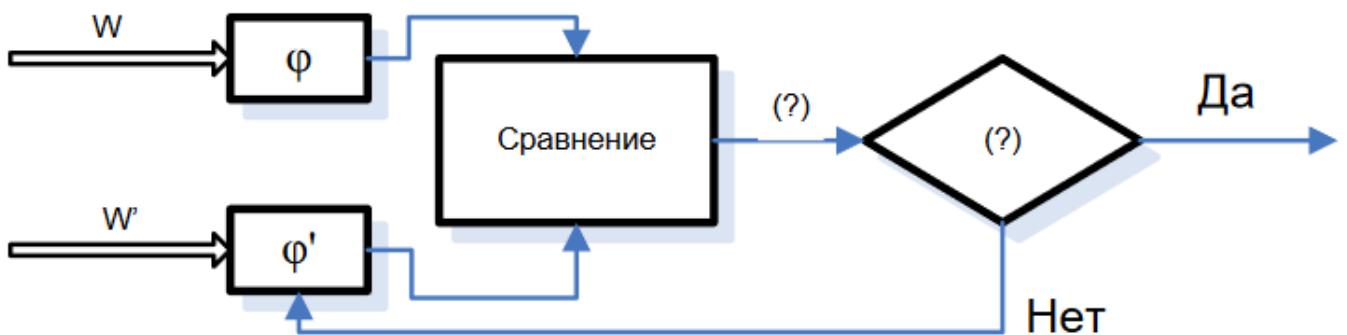
3. Получение и интерпретация результатов моделирования. Самое важное – проведение рабочих расчетов по готовой программе. Результаты этих расчетов позволяют проанализировать и сделать выводы о характеристиках процессов функционирования системы. **Последовательность действий:**

- планирование машинного эксперимента с моделью системы, активный и пассивный эксперименты, составление плана проведения эксперимента с указанием комбинаций переменных и параметров, для которых должен проводиться эксперимент;

Главная задача – дать максимальный объем информации при минимальных затратах вычислительных ресурсов.

- проведение рабочих расчетов;
- обработка статистических результатов расчетов, представление результатов.

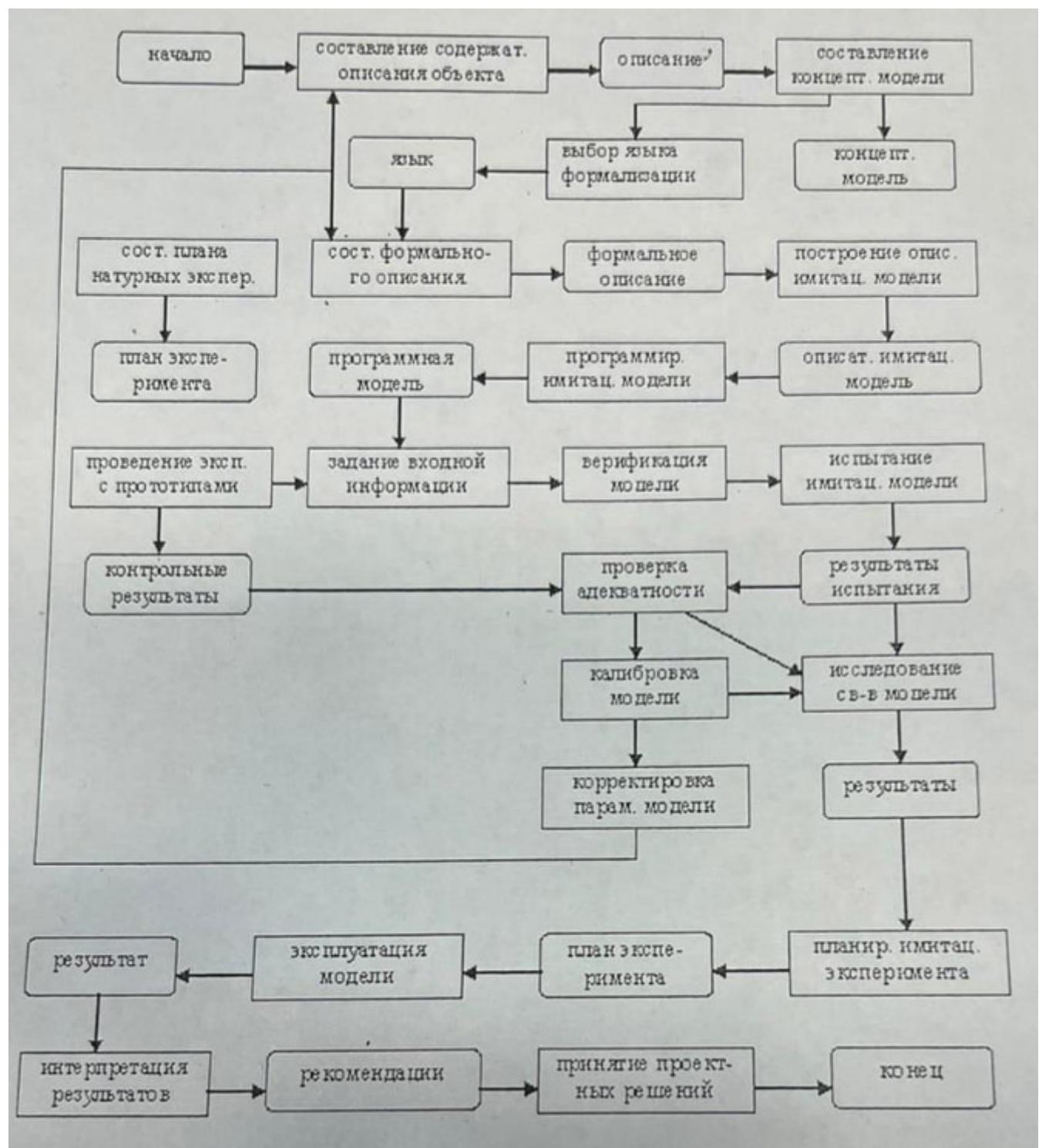
Калибровка модели



Возникает **три** основных типа ошибки:

1. Ошибка формализации – нужно заново делать модель, неполная модель.
2. Ошибки решения – взяли упрощенный, некорректный метод решения задачи.
3. Ошибки задания параметров модели.

Данный этап завершается определением и фиксацией области пригодности модели, под которой будем понимать множество условий, при соблюдении которых точность результатов моделирования находится в допустимых пределах.



Лекция 7. Вычислительная система как объект моделирования

Уровни проектирования

1. Структурный.
2. Функционально-логический уровень:
 - подуровни регистровых передач,
 - логический уровень.
3. Схемотехнический уровень.
4. Конструкторский.

Моделирование на системном уровне

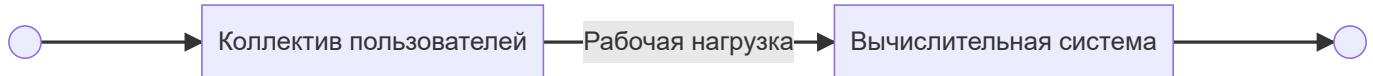
При моделировании новых и модернизации действующих вычислительных систем и сетей необходимо предварительно оценивать их возможности по функционированию с учетом различных вариантов структурной организации. Эти варианты могут отличаться составом и характеристиками модулей (наших устройств): структурами межмодульных связей, режимами работы и **алгоритмами управления**. *Именно для оценок и используются модели вычислительных систем.*

Под **вычислительной системой** будем понимать комплекс аппаратных и программных средств, которые в совокупности выполняют определенные рабочие функции.

Коллектив пользователей – это сообщество таких людей, которые используют нашу систему для удовлетворения нужд для обработки информации.

Входные сигналы (программы, данные, команды), которые создаются коллективом пользователей, называются *рабочей нагрузкой*.

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



Индекс производительности – описатель, который используется для представления производительности системы. Различают *количественные* и *качественные* индексы производительности.

1. Качественные:

- легкость использования системы;
- мощность системы команд.

2. Количественные:

- пропускная способность – объем информации, обрабатываемый в единицу времени;
- время ответа (реакции) – время между предъявлением системе входных данных и появлением соответствующей выходной информации;
- коэффициент использования оборудования – отношение времени использования указанной части оборудования в течение заданного интервала времени к длительности этого интервала.

Концептуальная модель вычислительной системы включает сведения о выходных и конструктивных параметрах системы, её структуре, особенностях работы каждого элемента и ресурса, постановка прикладных задач, определение цели моделирования.

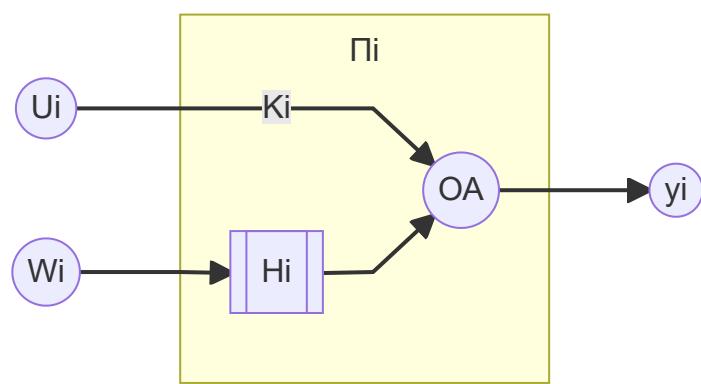
Основные задачи

1. Определение принципов организации вычислительной системы (ВС).
2. Выбор архитектуры, уточнение функций ВС и их разделение на подфункции, реализация аппаратным и программным путем.
3. Разработка структурной схемы – определение состава устройств и способов их взаимодействий.
4. Определение требований к выходным параметрам устройств и формирования технического задания на разработку устройств для функционально-логического уровня проектирования.

Непрерывно стохастические модели (Q-схемы)

Особенности непрерывно-стохастического подхода в дальнейшем рассматривается только на примере использования в качестве типовых математическим моделям системы массового обслуживания. При этом исследуемая система формализуется как некоторая система обслуживания. Характерным для таких объектов является **случайное появление заявок на обслуживание и завершение обслуживания в случайные моменты времени**.

В любом элементарном акте обслуживании можно выделить ожидание обслуживания и собственно обслуживание.



Два типа потоков:

- поток-накопитель – характеризуется своей емкостью;
- поток обслуживания.

Поток событий – последовательность событий, происходящих одно за другим в какие-то случайные моменты времени. Поток событий называется однородным, если он характеризуется только моментами поступления этих событий (вызывающие моменты). Поток называется неоднородным, если он задается не только вызывающими моментами, но и признаками этих событий. Если интервалы времени между сообщениями независимы между собой и являются случайными величинами, то это поток с ограниченным действием.

Поток событий называется ординарным, если вероятность того, что на малый интервал времени Δt попадает более одного события, пренебрежительно мала по сравнению с вероятностью того, что на этот же интервал времени Δt попадает ровно одно событие.

Поток называется стационарным, если вероятность появления того или иного события на некотором интервале времени зависит лишь от длины этого интервала и не зависит от того, где на оси времени взят этот интервал.

Для ординарного потока среднее число событий, поступающих за малый интервал Δt :

$$P_{>1}(t, \Delta t) + P_1(t, \Delta t) = P_1(t, \Delta t)$$

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

Для стационарного потока его интенсивность не зависит от времени и представляет собой постоянное значение, равное среднему числу событий, поступивших в единицу времени.

Лекция 8

В i -м приборе обслуживания имеем два потока:

1. Поток заявок – интервал времени между моментами появления заявки на входе канала k_i
2. Поток обслуживания – интервал времени между началом и концом обслуживания заявки в канале.

Заявки, обслуженные каналом, и заявки, покинувшие i -й прибор необслуженными, образуют выходной поток y_i . Процесс функционирования i -го прибора можно представить как *процесс изменения его состояния во времени*. Переход в новое состояние означает изменение количества заявок. Они могут находиться как в накопителе, так и в канале. Следовательно, вектор состояния для i -го прибора имеет следующий вид: $\vec{Z}_i = (Z_k^n, Z_i^k)$

Состояние накопителя:

- если 0, то, накопитель пуст;
- если Z_i равно емкости накопителя, то накопитель полностью занят.

Состояние канала:

- если 0, то канал свободен;
- если 1, то канал занят.

В практике моделирования сложных систем элементарные и сложные приборы обычно объединяются, при этом, если каналы различных приборов обслуживания соединены параллельно, то имеет место многоканальное обслуживание, если последовательное – то многофазное обслуживание. Поэтому для задания схемы необходимо использовать оператор сопряжения, отражающий взаимосвязь элементов между собой. Различают разомкнутые, замкнутые и смешанного типа.

Собственными внутренними параметрами Q-схемы будут являться:

- количество фаз
- количество каналов в каждой фазе

- количество накопителей в каждой фазе
- емкость i -ого накопителя

Для задания Q-схемы также необходимо описание алгоритма функционирования Q-схемы, который в данной интерпретации определяет поведение заявок в системе в различных ситуациях – неоднородность заявок. Следовательно, Q-схема, описывающая процесс функционирования системы массового обслуживания, может быть записана в виде кортежа:

$$Q = (W, U, R, H, Z, A), \text{ где}$$

1. **W** - подмножество входных потоков.
2. **U** - подмножество потоков обслуживания.
3. **H** - подмножество собственных параметров.
4. **Z** - множество состояний элементов системы.
5. **R** - оператор сопряжения элементов структуры (вектор сопряжения).
6. **A** - оператор алгоритмов обслуживания заявок.

Для получения соотношений, связывающих характеристики описывающих 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 с успехом применен математический аппарат в теории вероятности для, так называемых, марковских случайных процессов.

Случайный процесс, протекающий в некоторой системе S , называется **марковским**, если он обладает следующим свойством: для каждого момента времени вероятность любого состояния системы в будущем зависит только от ее состояния в настоящем и не зависит от того, когда и каким образом система пришла в это состояние.

Марковских систем в реальности нет

Для марковского процесса обычно составляются уравнения Колмагорова:

$$F = (p'(t), p(t), \Lambda) = 0, \text{ где}$$

- Λ – некоторый набор коэффициентов,

- p – вероятность,
- p' – производная вероятности.

$$\Phi = (p(t), \Lambda) = 0$$

$$p = p(\Lambda)$$

$$Y = Y(p(\Delta))$$

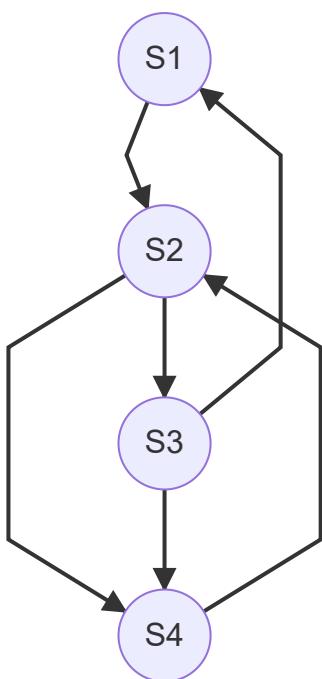
$$Y = Y(X, V, L)$$

Следовательно нам нужно осуществить связь внутренних параметров модели: Λ с конструктивными параметрами X , неуправляемыми параметрами V , учитывая по внутренним параметрам H . Таким образом:

$$\Lambda = \Lambda(X, V, L)$$

То есть получаем интерфейсную модель. Следовательно, математическая модель системы строится как совокупность базисной модели и интерфейсной, что позволяет использовать одни и те же базисные модели для решения разных задач по моделированию, осуществляя настройку на соответствующую задачу посредством изменения интерфейсной модели.

Определения вероятностей состояния. Пусть наша система S имеет 4 различимых состояния.



где имя стрелки $Sx - Sy == \Lambda_{xy}$.

Найдем вероятность P_1 , то есть вероятность того, что система будет находиться в состоянии S1.

Это можно найти двумя способами:

- в момент t система была уже в состоянии S1;
- в момент времени t уже была в состоянии S3 и за Δt пришла в состояние S1.

1. Вероятность первого способа как произведение вероятностей $P_1(t)$ на условную вероятность того, что будучи в состоянии S1 система не перейдет в состояние S2.

Таким образом мы получаем первое уравнение Колмогорова:

$$P_1(t + \Delta t) = P_1(t)(1 - \lambda_{12}\Delta t) + P_3(t)\Lambda_{31}\Delta t$$

$$\lim_{\Delta t \rightarrow 0} \frac{P_1(t + \Delta t) - P_1(t)}{\Delta t} = -P_1(t)\lambda_{12} + P_3(t)\lambda$$

$$P'_1(t) = -P_1(t)\lambda_{12} + P_3(t)\lambda_{31}$$

2. Найдем вероятность того, что система находится в состоянии S2:

$$P_2(t + \Delta t) = P_2(t)(1 - \lambda_{24}\Delta t - \lambda_{23}\Delta t) + P_1(t)\lambda_{12}\Delta t + P_4(t)\lambda_{42}\Delta t$$

$$\lim_{\Delta t \rightarrow 0} \frac{P_2(t + \Delta t) - P_2(t)}{\Delta t} = -P_2(t)\lambda_{24} - P_2(t)\lambda_{23} + P_1(t)\lambda_{12} + P_4(t)\lambda_{42}$$

$$P'_2(t) = -P_2(t)\lambda_{24} - P_2(t)\lambda_{23} + P_1(t)\lambda_{12} + P_4(t)\lambda_{42}$$

3. Найдем вероятность того, что система находится в состоянии S3:

$$P'_3(t) = -P_3(t)\lambda_{31} - P_3(t)\lambda_{34} + P_2(t)\lambda_{23}$$

4. Найдем вероятность того, что система находится в состоянии S4:

$$P'_4(t) = -P_4(t)\lambda_{42} - P_2(t)\lambda_{24} + P_3(t)\lambda_{34}$$

Все уравнения Колмогорова:

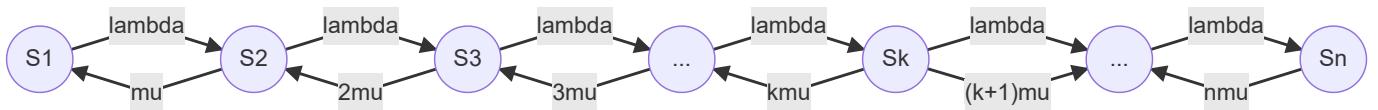
$$\begin{cases} P'_1(t) = -P_1(t)\lambda_{12} + P_3(t)\lambda \\ P'_2(t) = -P_2(t)\lambda_{24} - P_2(t)\lambda_{23} + P_1(t)\lambda_{12} + P_4(t)\lambda_{42} \\ P'_3(t) = -P_3(t)\lambda_{31} - P_3(t)\lambda_{34} + P_2(t)\lambda_{23} \\ P'_4(t) = -P_4(t)\lambda_{42} - P_2(t)\lambda_{24} + P_3(t)\lambda_{34} \end{cases}$$

Интегрирование данной системы дает искомые. Начальное условие берется в зависимости от того, какого было изначальное состояние системы. Обязательно нужно добавить условие нормировки.

Уравнения Колмогорова строятся по следующим правилам.

1. В левой части каждого уравнения стоит производная вероятности состояния, а в правой части содержится столько членов, сколько стрелок связано с этим состоянием.
2. Если стрелка направлена из состояния, соответствующий член имеет знак "-", если в состояние, то знак "+".
3. Каждый член равен произведению плотности вероятности перехода (интенсивность), соответствующий данной стрелке, и вероятности того состояния, из которого выходит стрелка.

Лекция 9



Разметим граф, то есть расставим стрелки интенсивности соответствующих потоков. По стрелкам слева направо системы переводит один и тот же поток, это поток заявок с интенсивностью λ . Пусть система была в состоянии S_1 и тогда как только закончится обслуживание заявки занимающего этот канал, система перейдет в состояние S_0 , интенсивность перехода μ . Если занято 2 канала, а не один, то интенсивность перехода составит 2μ .

$$\begin{cases} p'_0(t) = -\lambda p_0 + \mu p_1 \\ p'_1(t) = -(\lambda + \mu)p_1 + \lambda p_0 + 2\mu p_2 \\ p'_2(t) = -(\lambda + 2\mu)p_2 + \lambda p_1 - 3\mu \\ \dots \\ p'_k(t) = -(\lambda + k\mu)p_k + \lambda p \end{cases}$$

Предельные вероятности состояний p_0 и p_n характеризуют установившийся режим работы системы массового обслуживания при $t \rightarrow \infty$.

$$\begin{aligned} p_0 &= \frac{1}{1 + \frac{\lambda/\mu}{1!} + \frac{(\lambda/\mu)^2}{2!} + \dots + \frac{(\lambda/\mu)^n}{n!}} \\ p_k &= \frac{(\lambda/\mu)^k}{k!} p_0 \\ p_0 &= \left[1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \frac{\rho^2}{2!} \right] \\ p_k &= \frac{\rho^k}{k!} p_0 \end{aligned}$$

λ/μ - среднее число заявок, приходящих в систему за среднее время обслуживания одной заявки.

Зная все вероятности состояний p_0, \dots, p_n , можно найти характеристики СМО:

- вероятность отказа – вероятность того, что все n каналов заняты:
- $$p_{1OE} = p_n = \frac{\rho^n}{n!} p_0$$

- относительная пропускная способность – вероятность того, что заявка будет принята к обслуживанию:

$$q = 1 - p_n$$
- среднее число заявок, обслуженных в единицу времени: $A = \lambda q$

Полученные выражения могут рассматриваться как базисная модель оценки характеристик производительности систем. Входящий в эту модель параметр $\lambda = \frac{1}{\text{время обработки}}$ является усредненной характеристикой пользователей, а параметр μ – это функция технических характеристик компьютера, не только их, но еще и решаемых задач. Связь между ними должна быть установлена с помощью интерфейсной модели.

В простейшем случае, когда время ввода / вывода информации по каждой задачи мало по сравнению с временем ее решения, то можно принять, что время решения $= \frac{1}{\mu}$, где время решения – среднее время решения задачи процессом. Оно равно дроби, в числителе которой будет равно среднее число операций, выполняемых процессом на одну задачу.

Немарковские случайные процессы, сводящиеся к марковским

Реальные процессы весьма часто обладают последействием и поэтому не являются марковскими. Очень редко удается воспользоваться методами, разработанными для марковских цепей. Наиболее распространённые:

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

Метод псевдосостояния

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

Условия статистической эквивалентности реального состояния и фиктивных в каждом конкретном случае подбираются индивидуально. Очень часто может использоваться следующее: $\min \int_{t_1}^{t_2} (\lambda_{\text{экв}}(\tau) - \lambda_i(\tau)) dt$, где λ – эквивалентная интенсивность перехода в i -ой группе переходов, заменяющей реальный переход, обладающий интенсивностью λ_i .

За счет расширения числа состояний системы некоторые процессы удается точно свести к марковским. Созданная таким образом система статистически эквивалентна или будет близка к реальной системе, и она подвергается обычному исследованию с помощью аппарата марковских цепей.

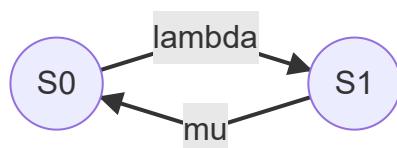
К числу процессов, которые введением фиктивных состояний можно точно свести к марковским, относятся процессы под воздействием потоков Эрланга. В случае потока Эрланга k -ого порядка интервал времени между соседними событиями представляет собой сумму k независимых случайных интервалов, распределенных по показательному закону. Поэтому переход потока Эрланга k -го порядка к пуассоновскому осуществляется введением k псевдосостояний. Интенсивности переходов между псевдосостояниями равны соответствующему параметру потока Эрланга. Полученный таким образом эквивалентный случайный процесс является марковским, так как интервалы времени нахождения его в различных состояниях подчиняются показательному закону.

Пример. Устройство S выходит из строя с интенсивностью λ , причем поток отказов пуассоновский. После отказа устройство восстанавливается и время восстановления распределено по закону Эрланга 3 порядка с функцией плотности $f_2(t) = 0.5\mu(\mu t)^2 e^{-\mu t}$.

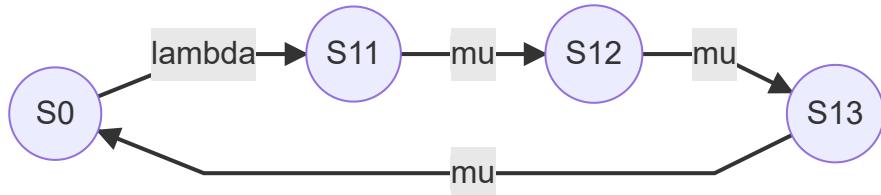
Система может принимать два возможных состояния:

- S_0 – устройство исправно;
- S_1 – устройство отказалось и восстанавливается.

Переход из S_0 в S_1 осуществляется под воздействием пуассоновского потока, а обратный – потока Эрланга.



Представим случайное время восстановления в виде суммы 3х случайных временных интервалов, распределенных по показательному закону с интенсивностью μ .



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

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

$$\text{Ответ: } P_0 = \frac{\mu}{\mu + 3\lambda}, P_1 = \frac{3}{3 + \frac{\mu}{\lambda}}$$

Метод вложенных цепей Маркова.

Вложенные цепи Маркова образуются следующим образом: в исходном случайном процессе выбираются такие случайные процессы, в которых характеристики образуют марковскую цепь. Моменты времени обычно являются случайными и зависят от свойств исходного процесса. Затем обычными методами теории марковских цепей исследуются процессы только в эти характерные моменты. Случайный процесс называется полумарковским (с конечным или счетным множеством состояний) если заданы переходы состояний из одного состояния в

другое и распределение времени пребывания процессов в каждом состоянии. Например, в виде функции распределения или функции плотности распределения.

Метод статистических испытаний. Метод Монте-Карло.

Преимущество метода статистических испытаний: его универсальность, обуславливающая его возможность всестороннего статистического исследования объекта. Но для реализации этого исследования необходимы довольно полные статистические сведения о параметрах элемента входящих в системы.

Недостаток: большой объем требующихся вычислений, равный количеству обращений к модели. Поэтому вопрос выбора величины n имеет важнейшее значение. Уменьшая n , повышаем экономичность расчетов, но одновременно ухудшаем их точность.

Лекция 10

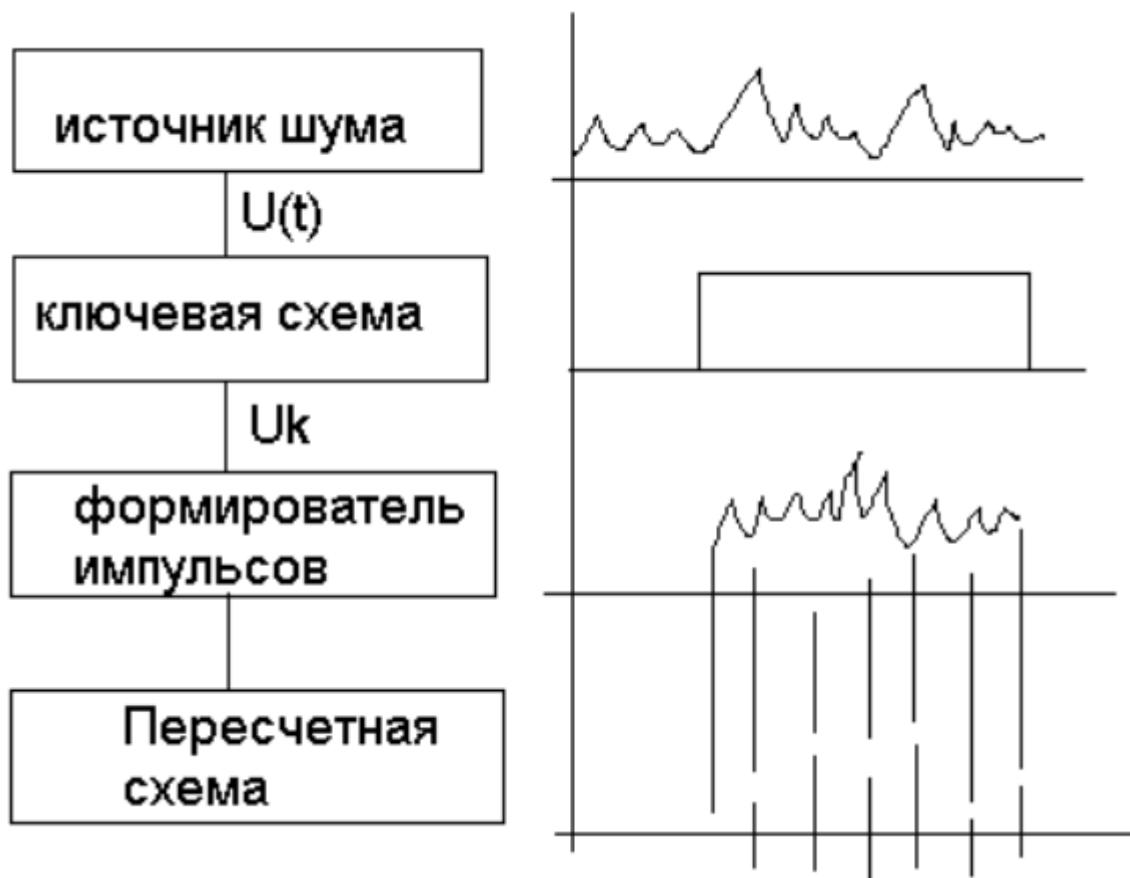
Методы получения последовательности случайных чисел.

При имитационном моделировании сложных систем одним из основных вопросов является учет стохастических воздействий. Для этого метода характерно большое число операций со случайными числами и зависимость результатов от качества исходных последовательностей случайных чисел. Способы получения случайных:

1. Аппаратный (физический).
2. Табличный (файловый).
3. Алгоритмический (программный)

Аппаратный способ.

Случайные числа вырабатываются специальной электронной приставкой (генератор случайных чисел - внешнее устройство).



Табличная схема.

Случайные числа оформляются в виде таблицы и помещаются во внешнюю или оперативную память.

Алгоритмический способ.

Способ основан на формировании случайных чисел с помощью специальных алгоритмов.

Преимущества и недостатки методов генерации случайных чисел

Способ	Достоинства	Недостатки
Аппаратный	1. Запас чисел неограничен 2. Расходуется мало операций 3. Не занимается место в оперативной памяти.	1. Требуется периодическая проверка на случайность 2. Нельзя воспроизводить последовательности 3. Используются специальные устройства. Надо стабилизировать
Табличный	1. Требуется однократная проверка 2. Можно воспроизводить последовательности	1. Запас чисел ограничен 2. Занимает место в оперативной памяти и требуется время на обращение к памяти
Алгоритмический	1. Однократная проверка 2. Можно многократно воспроизводить последовательности чисел 3. Относительно малое место в оперативной памяти 4. Не используются внешние устройства	1. Запас чисел последовательности ограничен её периодом 2. Требуются затраты машинного времени

Рассмотрим некоторую случайную величину X , которая может принимать любые значения из интервала $[a, b]$ и имеет плотность распределения $f_X(x) = \frac{1}{b-a}$.

Алгоритмы генерации последовательности псевдослучайных чисел

Одним из первых способов получения последовательности псевдослучайных чисел было выделение дробной части многочлена первой степени:
 $y_n = Ent(an + b)$

Если n пробегает значения натурального ряда чисел, то поведение y_n выглядит весьма хаотично. Физик Якоби доказал, что при рациональном коэффициенте a множество y конечно, а при иррациональном – бесконечно и всюду плотно в интервале $[0, 1]$.

Для многочленов больших степеней такую задачу решил Герман Вей, он предложил критерий равномерности распределения любой функции от натурального ряда чисел. Называется это эргодичностью и заключается в том, что среднее по реализациям псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1. Эти результаты далеки от практики получения последовательностей псевдослучайных чисел, поэтому она используется только для действительных чисел, что затрудняет практическую её реализацию. Попытки замены настоящего иррационального числа его приближением на компьютере привели к тому, что полученные последовательности оканчиваются циклом с коротким периодом.

1. 1946 год, Фон Нейман.

Каждое последующее число образуется возведением предыдущего в квадрат и отбрасыванием цифр. Способ с точки зрения случайности оказался нестабильным.

2. Лимер.

$$g_{n+1} = g_n k + c \mod m$$

Для подбора коэффициентов k , c , m потрачены десятки лет. Подбор почти иррациональных k ничего не дает. Разумнее вести все вычисления в целых числах. Установили, что при $c = 0$ и $m = 2^n$ наибольший период достигается при нечетном начальном числе и при $k = 3 + 8i$, $k = 5 + 8i$.

3. Форсайд

В 1977 году показал, что тройки последовательности чисел лежат на 15 параллельных плоскостях.

```

var n, i: integer;
x, R: double;
Const m34: double = 28395423107.0;
m35: double = 34359738368.0;
m36: double = 68719476736.0;
m37: double = 137438953472.0;

function Rand(n: integer): double;
var S, W: double;
i: integer;
begin
  if n = 0 then
begin
  x := m34; Rand := 0; exit;
end;
S := -2.5;
for i := 1 to 5 do
begin
  x := 5.0 * x;
  if x > m37 then x := x - m37;
  if x > m36 then x := x - m36;
  if x > m35 then x := x - m35;
  W := x / m35;

  if n = 1 then
begin
  Rand := W; exit;
end;
  S := S + W;
end;

S := S * 1.54919
Rand := (sqr(S) - 3.0) * S * 0.01 + S;
end;

begin
R := Rand(0)
for i := 1 to 200 do
writeln(Rand(2):18:10)
end;

```

От отчаяния используют 2 и даже 3 разных генератора, смешивая их значения. Если бы разные генераторы были независимыми, то сумма их последовательностей обладала дисперсией, равной сумме дисперсий. Иначе случайность рядов возрастет при суммировании. Сейчас в системах программирования обычно используют конгруэнтные генераторы по алгоритму, предложенному национальным бюро стандартов США, который имеет длину 2^{24} .

4. Генерация случайных чисел по алгоритму Зеймана.

$$\{1, 1, 2, 3, 5, 8, 13, 21, \dots\}$$

$\bmod 10$

$$\{1, 1, 2, 3, 5, 8, 3, 1, \dots\}$$

Переименовываем с помощью какого-либо ГСЧ; пусть всё так и осталось:

$$\{1, 1, 2, 3, 5, 8, 3, 1, \dots\}$$

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 случайных чисел с заданным законом.

$$F(t) = \int_{-\infty}^t f(x)dx = R \quad (1)$$

Метод основан на утверждении, что случайная величина x , принимающая значения, равные корню уравнения (1), имеет плотность распределения $f(x)$. R – равномерная случайная величина от 0 до 1.

Значение случайной величины распределенной по показательному закону исходя из (1) может быть вычислено следующим образом:

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

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

Распределение Пуассона

Распределение Пуассона относится к числу дискретных, то есть таких, при которых переменная может принимать лишь целочисленные значения, включая норму с мат. ожиданием и дисперсией равной $\lambda > 0$.

Для генерации пуассоновских переменных можно использовать метод точек, в основе которого лежит генерируемое случайное значение R_i , равномерно распределенное на $[0, 1]$, до тех пор, пока не станет справедливым

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

При получении случайной величины, функция распределения которой не позволяет найти решение уравнения (1) 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

Распределение Эрланга характеризуется двумя параметрами: λ и k . Поэтому при вычислении случайной величины в соответствии с данным законом 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.

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

Нормальное (Гауссово) распределение

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.

Случайная величина X имеющая нормальное распределение с математическим ожиданием M_X и среднеквадратичным отклонением σ_X может быть получена по следующей формуле:

$$x = \sigma_X \cdot \sqrt{\frac{12}{N}} \cdot \left(\sum_{i=1}^N R_i - \frac{N}{2} \right) + M_X$$

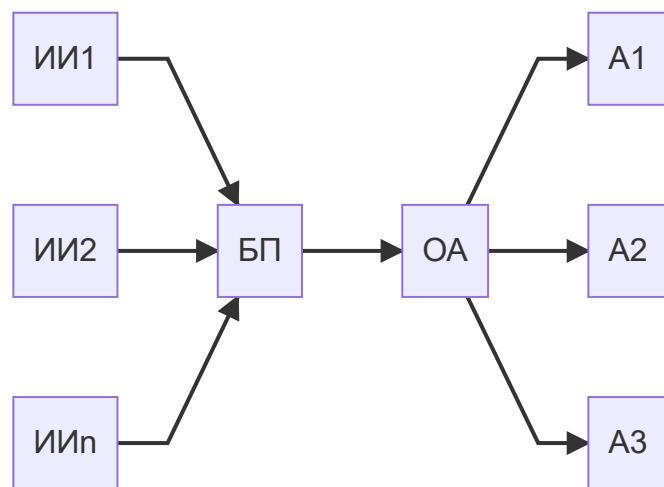
Для сокращения вычислений по нормальному закону распределения на практике часто принимают $N = 12$. Что дает довольно точные результаты.

Лекция 11. Методика построения имитационной программной модели вычислительной системы

Для разработки имитационной модели исходная система должна быть представлена как **стохастическая система массового обслуживания**.

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 устройством.

Обобщенная структурная схема ВС.



- ИИ (источники информации) – выдают на вход буферной памяти (БП) независимые друг от друга сообщения. Закон появления сообщений – произвольный, но задан наперед.
- В БП (буферной памяти) сообщения записываются «внавал» и выбираются по одному в обслуживающий аппарат (ОА) по принципу FIFO/LIFO. Длительность обработки одного сообщения в ОА в общем случае так же может быть случайной, но закон обработки сообщений должен быть задан. Так как быстродействие ОА ограничено то на входе системы в БП возможно сложение данных ожидающих обработки.
- А – абоненты.

Программная модель из этой системы создается следующим образом:



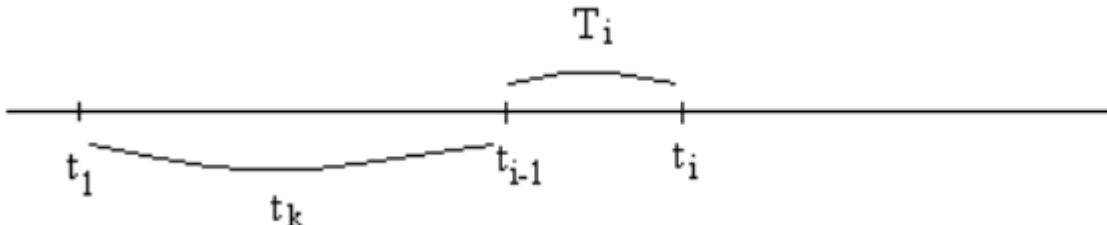
- Должна быть обязательно программа сбора статистики (ПБССт – программный блок сбора статистики). Причем статистику программа должна собирать по каждому из объектов модели.
- Также должна быть программа, которая позволит "оживить" систему – это программа синхронизации (блок синхронизации), которая покажет когда и в какое время будут активизированы те или иные фрагменты модели. Программа синхронизации – управляющая программа (осуществляет связь между каждым из этих блоков).

Если описать каждый блок ВС с помощью программного имитатора и задать соответствующие связи между ними, то получаем имитационную или программную модель системы.

Моделирование потока сообщений

Поток сообщений обычно имитируется *моментами появления очередного сообщения в потоке*. Момент текущего времени вычисляется как:

$$t_k = \sum_{k=1}^{i-1} T_k + T_i$$



Тип распределения	Выражение
Равномерное на отрезке $[a, b]$	$t_k = \sum_{k=1}^{i-1} T_k + T_i$
Экспоненциальное	$T_i = -\frac{1}{\lambda} \ln(1 - R)$
Нормальное (Гауссовское)	$T_i = \sigma_X \sqrt{\frac{12}{n}} (\sum_{i=1}^n R_i - \frac{n}{2}) + M_X, n = 12$
Эрланга	$T_i = \frac{1}{k\lambda} \sum_{i=1}^k \ln(1 - R_i)$

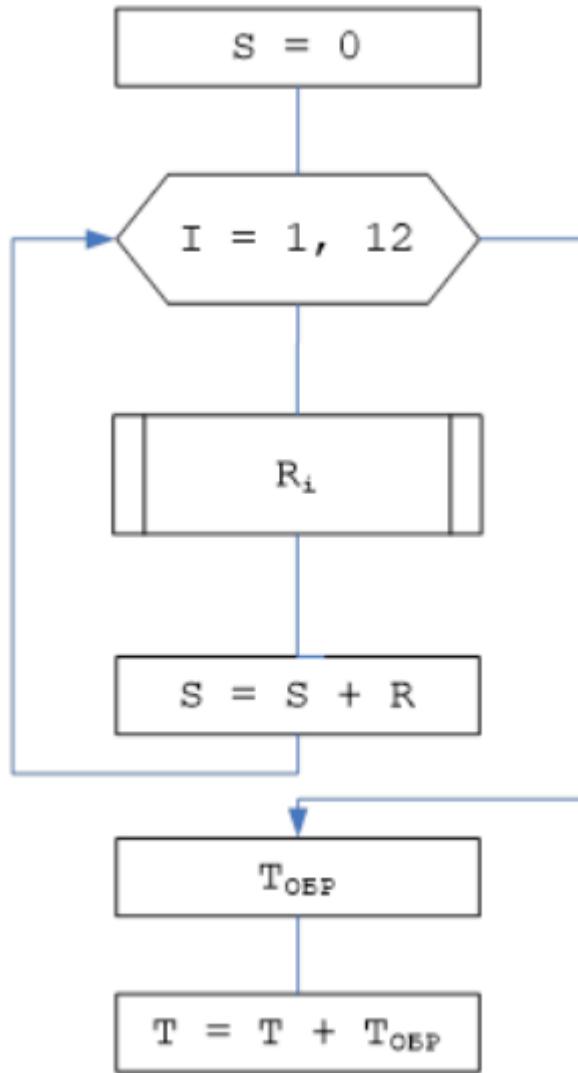
Моделирование работы обслуживающего аппарата

Программа, имитирующая работу обслуживающего аппарата – это набор программ, вырабатывающих случайные отрезки времени, соответствующие длительностью обслуживания требованиям.

Например, если требования от источника обрабатываются в ОА по нормальному закону с параметрами M_X и σ_X , то длительность обработки i -ого требования:

$$T_{\text{обр}} = M_X + \left(\sum_{i=1}^{12} R_i - 6 \right) \cdot \sigma_X$$

Схема алгоритма имитатора:



R_i – случайное число с равномерным законом распределения;

$T_{обр}$ – время обработки очередного сообщения;

T – время освобождения ОА;

MX – мат. ожидание для заданного закона обратки;

DX – СКО (средне квадратичное отклонение) для заданного закона обратки.

$$T_{обр} = MX + (S - 6) \cdot DX$$

Моделирование работы абонента

Абонент может рассматриваться как ОА, поток информации на который поступает от процессора.

Для имитации работы абонентов необходимо составить программу выработки длительности обслуживания требования. Кроме того, абонент сам может быть источником заявок вообще на любые ресурсы вычислительной системы. Эти заявки могут имитироваться с помощью генератора сообщений, распределенными по заданному закону. Таким образом, абонент либо имитируется как ОА, либо как генератор.

Моделирование работы буферной памяти

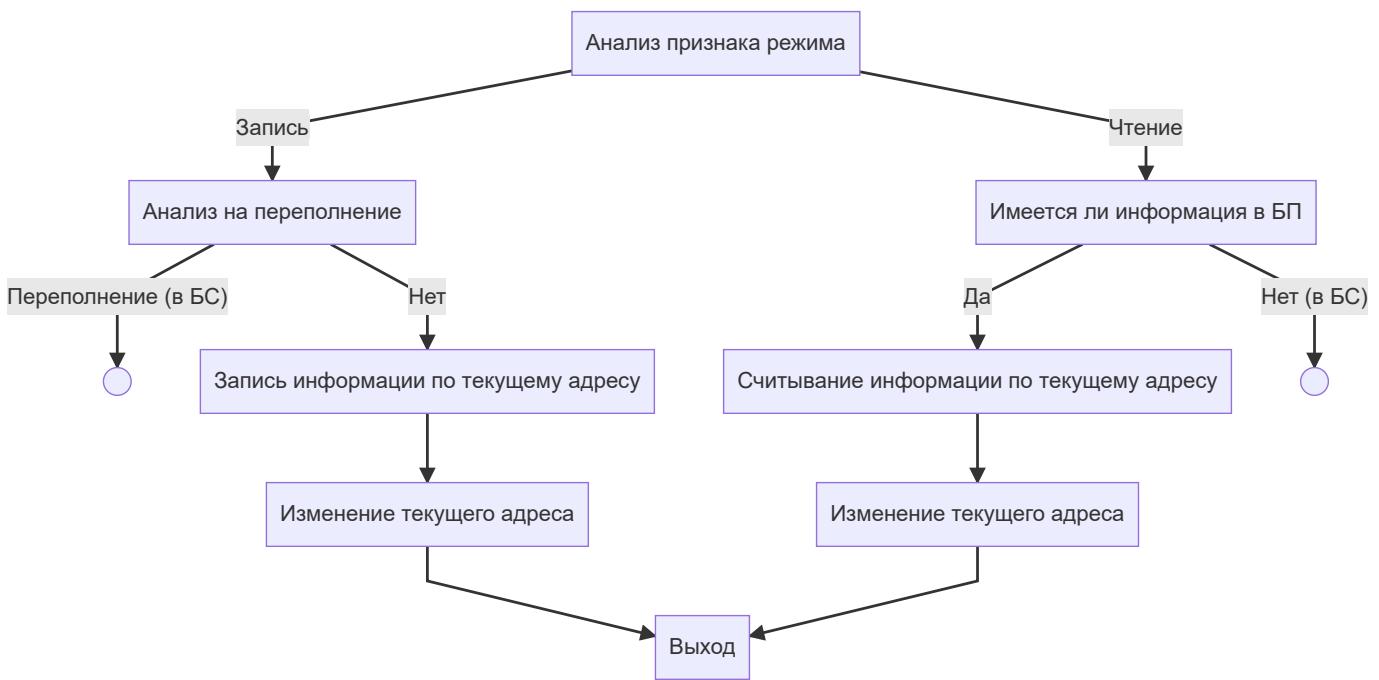
Память – относится к электромеханическому устройству, предназначенное для хранения, записи, считывания и включающее в себя среду для запоминания, устройство управления (информация находится по адресу база + смещение + [индекс]).

Свойства памяти: предназначена для хранения, чтения и записи информации.

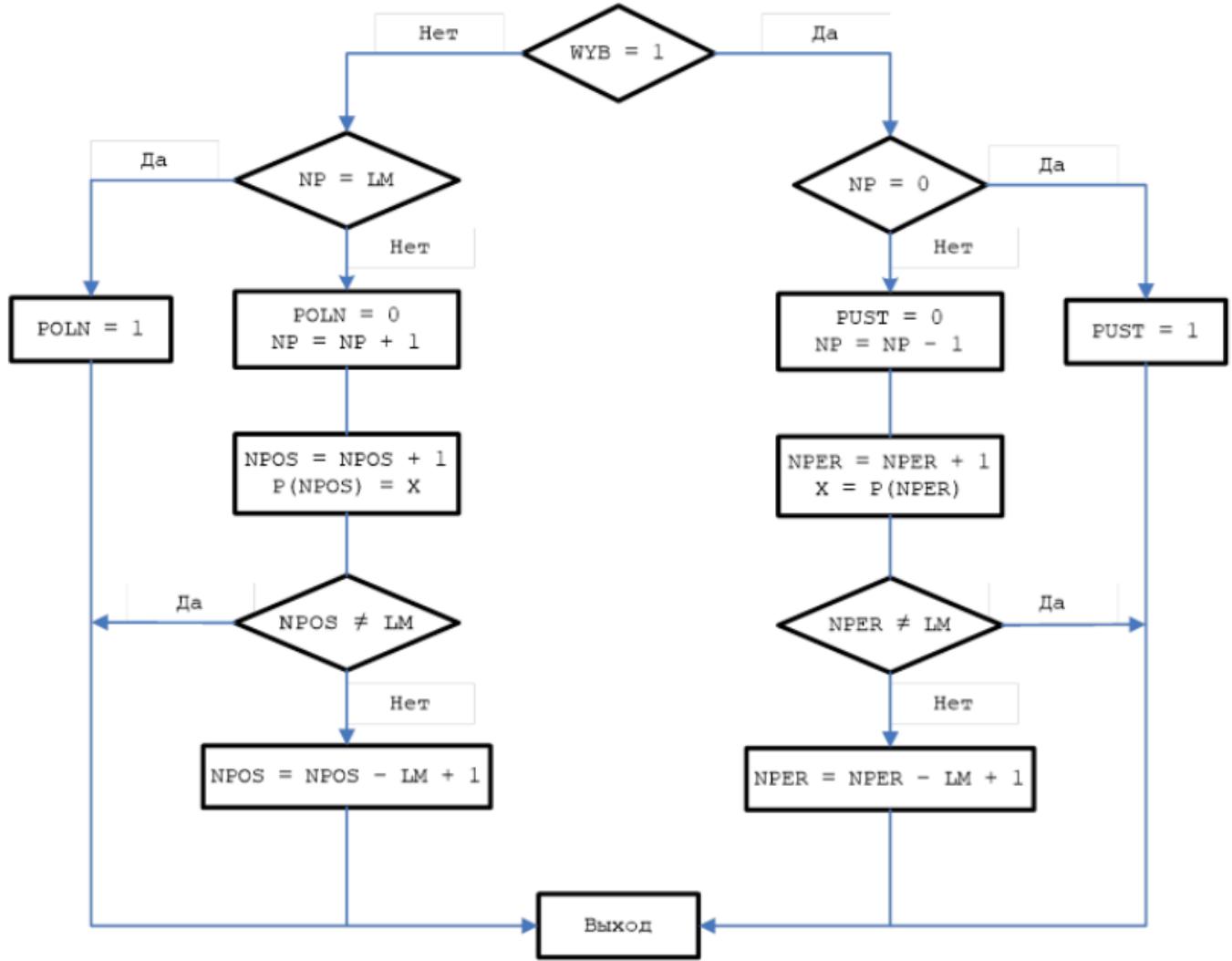
В блок статистики: ошибки записи, ошибки чтения.

Блок памяти должен производить запись и считывание числа, выдавать сигнал переполнения и отсутствия данных в любой момент времени, располагать сведениями о количестве находящихся в ней (в блоке) заявок. Сама запоминающая среда в простейшем случае имитируется одномерным массивом, размер которого определяет ёмкость памяти. Каждый элемент этого массива может быть либо "свободен" и в этом случае мы считаем, что он равен 0, либо "занят", в этом случае в качестве эквивалента требования ему присваивается значение времени появления этой записи (заносим время обращения к памяти).

Концептуальная модель памяти



Алгоритм реализации работы буферной памяти



Лекция 12

Разработка программы для сбора статистики

Задача блока статистики заключается в накоплении численных значений, необходимых для вычисления статистических оценок, заданных параметров работы моделируемой системы. При моделировании простейшей модели СМО, наибольший интерес представляет, **среднее время ожидания в очереди**. Для каждого сообщения *время ожидания в очереди* равно разности между моментами времени когда оно было выбрано на обработку обслуживающим аппаратом и моментом времени, когда оно пришло в систему от источника информации.

Суммируя количество сообщений в блоке памяти через небольшие промежутки времени и разделив полученную сумму на число суммирований, получим *среднее значение длины очереди*.

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

Рассчитав число потерянных сообщений, можно определить *вероятность потери сообщений* в системе. Нужно разделить количество потерянных сообщений на сумму потерянных и обработанных сообщений в системе.

Управляющая программа имитационной модели

Если программные имитаторы работы источника или буферной памяти обслуживающего аппарата отображают работу отдельных устройств, то управляющая программа имитирует алгоритм взаимодействия отдельных устройств системы.

Управляющая программа реализуется по следующим принципам:

1. Принцип Δt .
2. Событийный принцип.

Принцип Δt

Принцип Δt заключается в последовательном анализе состояний всех блоков в момент $t + \Delta t$ по заданному состоянию блоков в момент t . При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов, задаваемых распределениями вероятности. В результате такого анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью на данный момент времени.

Основной недостаток этого принципа: значительные затраты машинного времени на реализацию моделирования системы. А при недостаточно малом Δt появляется опасность пропуска отдельных событий в системе, что исключает возможность получения адекватных результатов при моделировании.

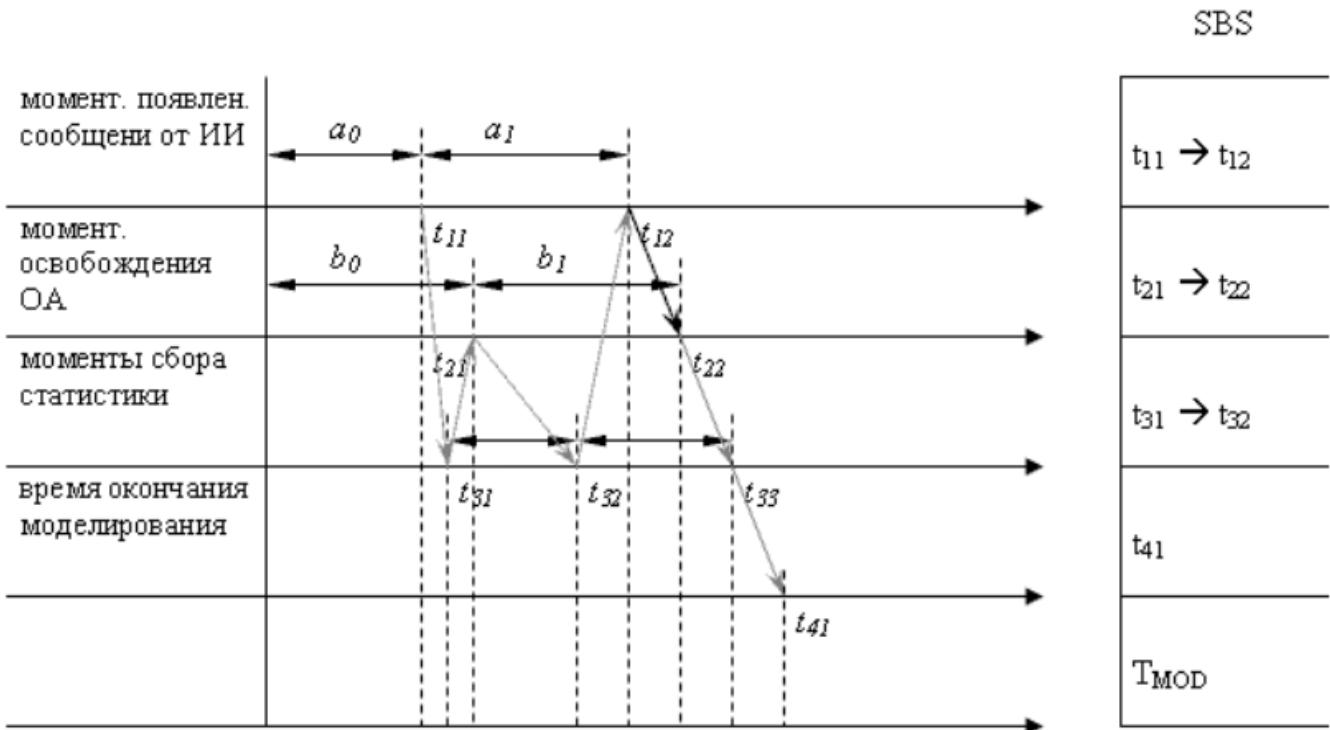
Достоинство: равномерная протяжка времени.

Событийный принцип.

Характерным свойством моделируемых систем обработки информации является то, что состояние отдельных устройств изменяются в дискретные моменты времени, совпадающие с моментами времени поступления сообщений в систему, временем поступления / окончания задачи, времени поступления аварийных сигналов и т.д. Следовательно, моделирование и продвижение времени в системе удобно проводить, используя событийный принцип. При использовании данного принципа состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. Момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов ближайшего изменения состояния каждого из блоков системы.

Недостаток событийного принципа: самостоятельная обработка.

Схема событийного принципа

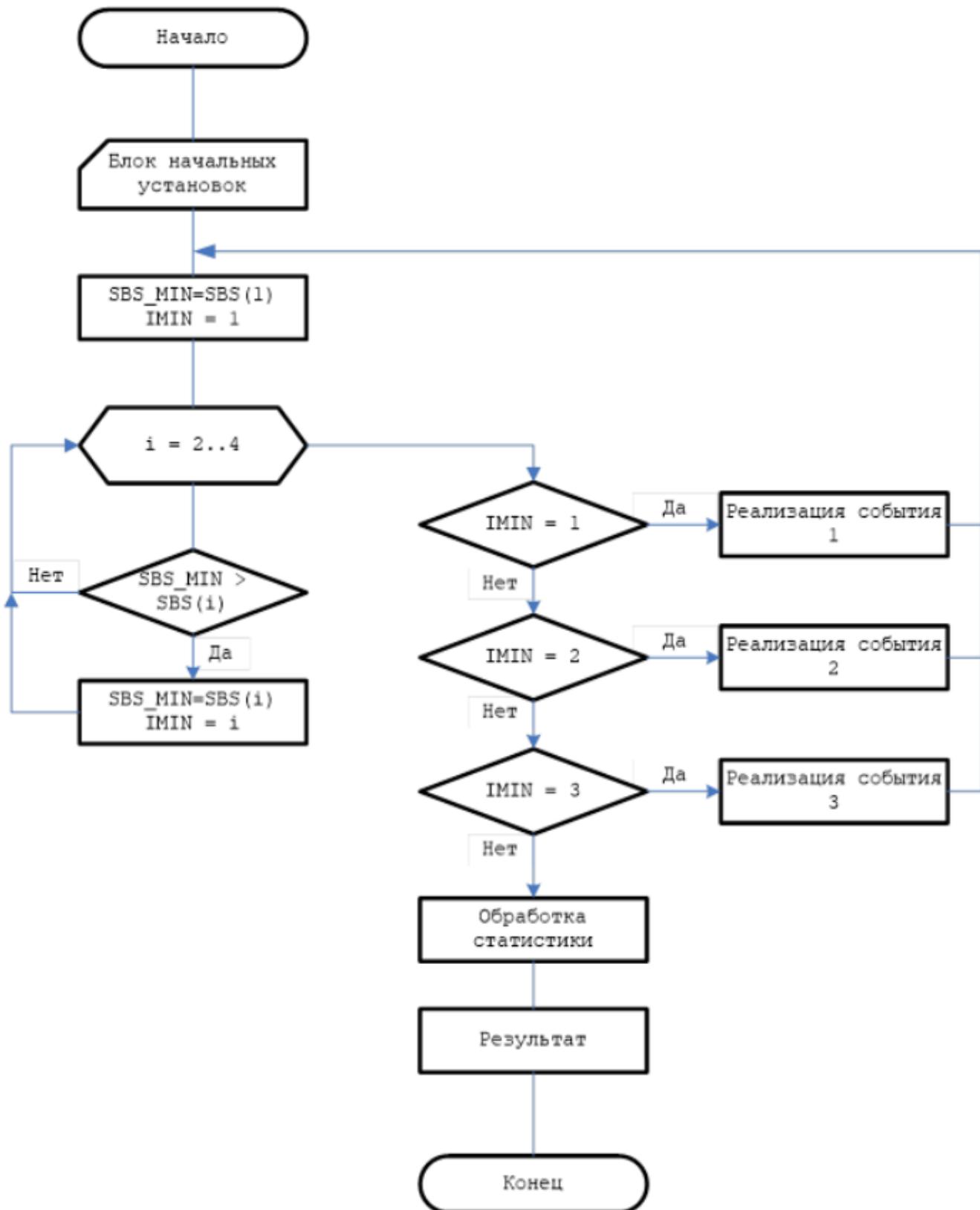


- Первая ось: момент появления сообщений.
- Вторая ось: момент освобождения обслуживающего аппарата.
- Третья ось: момент сбора статистики (здесь абсолютно равные интервалы, мы сами определяем, когда собирать статистику).
- Четвертая ось: время окончания моделирования.
- Пятая ось: текущее время.
- t_{11}, t_{12} – моменты появления сообщений на выходе генератора (источника информации).
- b_1 – интервал времени обслуживания первого сообщения.
- t_{3n} – момент сбора статистики.
- t_{41} – момент окончания моделирования.
- SBS – список будущих событий.

Методика реализации событийной модели

- Для всех активных блоков (порождающих события) заводят свой элемент в одномерном массиве – в списке будущих событий (СБС).
- В качестве подготовительной операции в СБС заносят время ближайшего события от любого активного блока. Активизируя программный имитатор

источника событий, вырабатывают псевдослучайную величину a_0 , определяющую момент появления первого сообщения t_{11} от источника информации и эту величину заносят в СБС. Активизируя программу-имитатор, ОА вырабатывает псевдослучайную величину b_0 , определяющую момент времени t_{21} , которую также заносят в СБС. В момент времени t_{31} (момент первого сбора статистики) определяется равным стандартному шагу сбору статистики $t_{\text{СТАТ}}$ и заносится так же в СБС. В этот же список заносим время окончания моделирования t_{41} . На этом подготовительный этап заканчивается и далее протяжка времени осуществляется по следующему алгоритму:



1. В SBS определяется минимальное числовое значение и его номер.
2. Реализуется событие, порождаемое блоком с соответствующим номером, то есть модельное время = t_{11} . Далее реализуется событие с номером 1, связанное с появлением нового сообщения в ИИ. Реализация этого события заключается в том, что само сообщение записывается в память, а с

помощью имитатора ИИ, вырабатывается момент появления следующего события t_{12} . Это время помещается в соответствующую ячейку SBS вместо t_{11} .

3. Затем вновь организуется поиск минимального элемента в SBS. Для данного примера реализуется событие 3, после чего выражение момента времени t_{32} – новое время сбора статистики. Так до тех пор, пока минимальное время не станет равным t_{41} .

Комбинированный метод

Два приведенных метода являются универсальными алгоритмами протяжки модального времени. Причем для некоторых предметных областей один принцип может работать быстро и без потерь, а другой будет работать неэффективно. Выбор метода необходимо производить исходя из распределения событий по времени. В реальных системах распределение событий, как правило, *неоднородно*. События, как бы группируются по времени. Образование таких групп связано с наступлением какого-то "значимого" события, которое начинает определенную последовательность действий с соответствующими событиями, имеющими высокую плотность на следующем временном интервале. Такой интервал называется **пиковым**. А распределение событий **квазисинхронным**. Примером может являться – цифровая сеть, в которой синхронизирующие сигналы переключают большое количество триггеров. Для сложных дискретных систем, в которых присутствуют квазисинхронное распределение событий, был разработан алгоритм с название **Delft**. Особенностью данного метода является автоматическая адаптация к распределению событий. Метод реализуется таким образом, что на пиковых интервалах он приближается к методу Δt , а вне пиковых – к событийному. В основе лежит использование иерархической структуры циркулярных списков.

Лекция 13

Моделирование системы и языки моделирования

Алгоритмические языки при моделировании систем служат вспомогательным аппаратом в разработке машинной реализации и анализа характеристик моделей.

Основная задача – это выбор языка.

Каждый язык имеет свою систему абстракций, позволяющих реализовать формальную модель системы.

Качество языков моделирования характеризуется:

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

Все современные языки программирования определяют поведение систем во времени (событийный алгоритм и его модификации). В большинстве своем языки моделирования определяют поведение систем во времени с помощью модифицированного событийного алгоритма. Как правило, он включает в себя список текущих и будущих событий.

Классификация языков имитационного моделирования.

Основа классификации – принцип формирования системного времени.



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.

Предполагается, что в системе могут наступать события двух типов:

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

Состояние системы описывается набором переменных, причем некоторые из них меняются непрерывно. При таком подходе пользователь должен использовать функции, описывающие условия наступления событий. Обязательно нужно описать: законы изменения непрерывных переменных, правила перехода от одного состояния к другому, т.е. реализуется классический принцип ДУ (дифференциальные уравнения).

Формальное описание динамики моделируемого объекта

Будем считать, что любая работа в системе совершается путем выполнения **активностей**. Т.е. активность является наименьшей единицей работы. Именно её рассматривают как единый дискретный шаг. Она имеет свое время выполнения. Следовательно, активность является единственным динамическим объектом, указывающим на совершение единицы работ.

Процесс – это логически связанный набор активностей.

Пример (считывание информации с жесткого диска): активность установки головки жесткого диска, активность ожидания запроса к диску, активность передачи информации с жесткого диска.

Активность проявляется в результате совершения событий. **События** – это мгновенное изменение состояния некоторого объекта системы.

Рассмотренные объекты (активности, процессы, события) являются **конструктивными элементами** для динамического 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 относительно небольшое число классов. Следовательно, чтобы описать поведение сложной системы, достаточно описать поведение каждого класса процессов и задать значение атрибутов для конкретных процессов (т.е. выбрать параметрическую модель).

Задачи построения модели

Построение модели состоит из решения двух основных задач:

1. Первая задача сводится к тому, чтобы описать правила, задающие виды процессов, происходящих в системе. Указать значения атрибутов таких процессов или сгенерировать. При этом система описывается на определенном уровне детализации в терминах множества описаний процессов, каждый из которых включает множество правил и условий возбуждения активности. Такое описание системы может быть декомпозировано. Именно такой подход обеспечивает многоуровневое исследование системы.
2. Вторая задача заключается в том, чтобы описать правила задания атрибутов или задать правила генерации этих значений. При этом система определяется на конкретном уровне детализации в терминах множества описаний процессов, каждый из которых в свою очередь включает множество правил и условий возбуждений активности. Такое описание системы может быть детализировано на более подробном или более

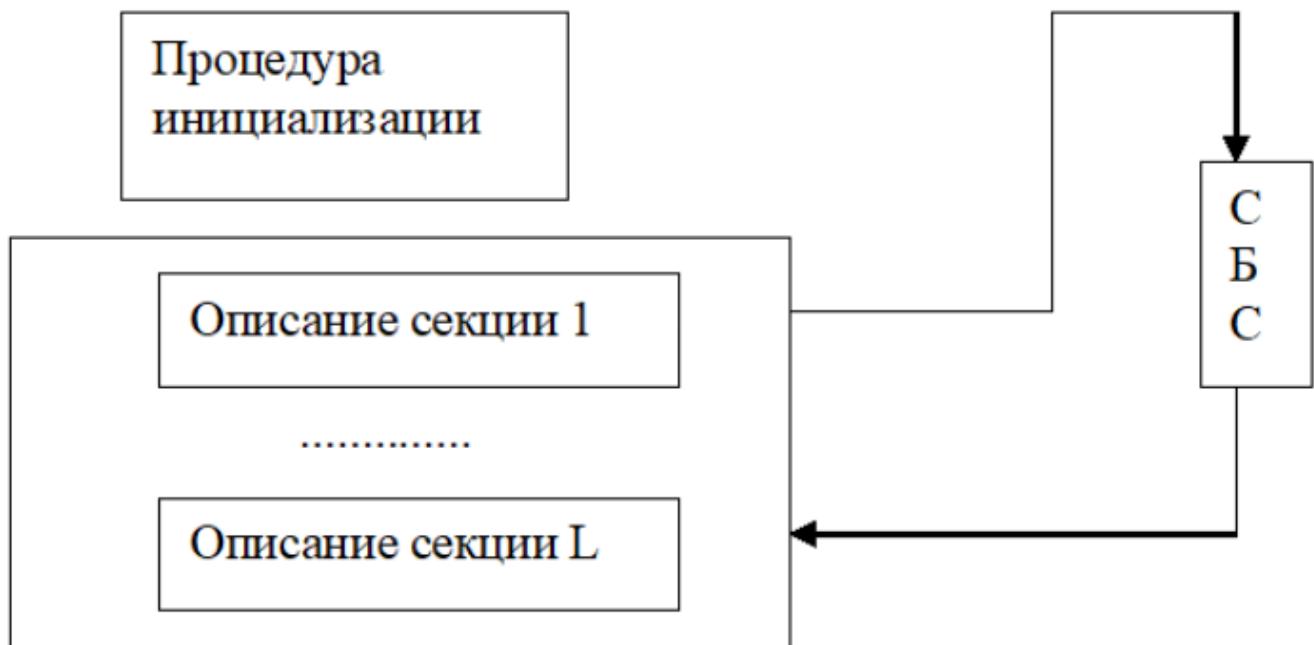
иерархическом уровне представления с помощью декомпозиции процессов (в идеальном случае в активности). Это обеспечивает многоуровневое исследование системы.

Так как модель в общем случае служит для описания временного поведения системы, то язык моделирования должен обладать средствами отображения времени.

Языки, ориентированные на события

Моделирующая программа реализована в виде совокупности секций событий. Секция событий состоит из операций, которые в общем случае выполняются после завершения какой-либо активности. Выполнение функций синхронизировано списков будущих событий.

Структура программы на языке SIMSCRIPT:

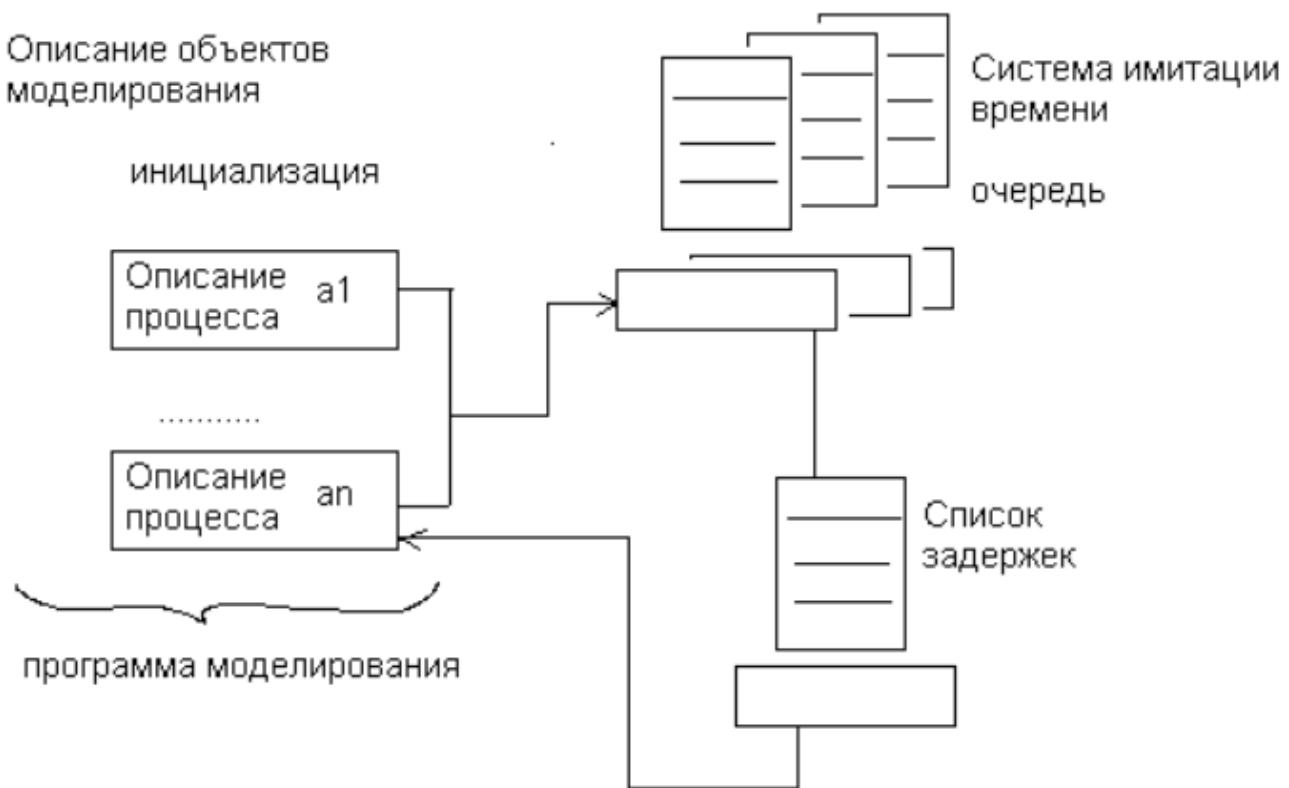


Первичное описание – это описание секций событий.

Языки, ориентированные на процессы

Моделирующая программа организуется в виде набора описаний процесса. Каждый из которых описывает один класс. Описание процесса функционирования устанавливает атрибуты и активности всех процессов. Синхронизация операций во времени (языка ориентированного на процессы) реализуются так же с помощью списка будущих событий, который содержит точку возобновления конкретного процесса (точка прерывания).

На примере языка SIMULA:



Критерии

- возможности языка (SIMULA → SIMSCRIPT → GPSS → C → Python)
- простота применения (GPSS → SIMSCRIPT → SIMULA → Python → C)
- предпочтения пользователя (SIMSCRIPT → GPSS → SIMULA → Python → C)

Сравнение универсальных и специализированных языков программирования при моделировании

Преимущества	Недостатки
Универсальные	
Минимум ограничений на выходной формат	Значительное время, затрачиваемое на программирование
Широкое распространение	Значительное время, затрачиваемое на отладку
Специализированные	
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	

Лекция 14. Обыкновенные сети Петри

Формальное 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.)

Определение сети Петри – формально в терминах теории систем это кортеж (т.е. набор элементов)

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

В этом определении:

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

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

- F – **функция инцидентности**;

$F : (P \times T) \cup (T \times P) \rightarrow \{0, 1, 2, \dots, k, \dots\}$, где k – кратность дугиж

- M_0 - начальная маркировка позиций: $M_0 : P \rightarrow \{0, 1, 2, \dots\}$.

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

1. $F^p(p, t) = P \times T \rightarrow \{0, 1, 2, \dots\}$, т.е. для каждой позиции указываются связанные с ней переходы (с учетом их кратности);
2. $F^t(t, p) = T \times P \rightarrow [0, 1, 2, \dots]$, т.е. 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*:

$$F^p = \begin{bmatrix} t_1 & t_2 & \dots & t_m \\ f_{11}^p & f_{12}^p & \dots & f_{1m}^p \\ \dots & \dots & \dots & \dots \\ f_{n1}^p & f_{n2}^p & \dots & f_{nm}^p \end{bmatrix} p_i$$

$$F^t = \begin{bmatrix} p_1 & p_2 & \dots & p_n \\ f_{11}^t & f_{12}^t & \dots & f_{1n}^t \\ \dots & \dots & \dots & \dots \\ f_{m1}^t & f_{m2}^t & \dots & f_{mn}^t \end{bmatrix} t_j$$

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 является выходным переходом, будем обозначать $P^j : \{p_k, f_{ij} > 0\}$. Иными словами, $P^j = \{p_k : f_{li}^t > 0\}$.

Аналогично из каждой вершины перехода $t_j \in T$ дуга ведёт в вершину-позицию

$p_i \in P$, тогда и только тогда, когда $f_{ji}^t > 0$. При этом говорят что p_i является выходной позицией перехода t_j .

Множество всех переходов t_l , для которых 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. Этим ресурсом могут быть студенты (число фишек = число студентов), сданные работы (число фишек = число сданных работ) и т.п. Вектор $M = \{\mu_1, \mu_2, \dots, \mu_k\}$ называют **разметкой (маркировкой) сети Петри**. Каждая маркировка – это отображение $M : P \rightarrow \{0, 1, 2, \dots\}$. Начальная маркировка M_0 определяет стартовое состояние сети Петри.

Динамика поведения моделируемой системы описывается, в отличие от конечных автоматов, в терминах функционирования сетей Петри. Как было сказано, сеть функционирует в дискретном времени $\tau = 0, 1, 2, 3, \dots$ в асинхронном режиме, переходя от разметки к разметке.

Смена разметок (начиная с M_0) происходит в результате срабатывания переходов сети. Переход $t_j \in T$ может сработать при разметке M , если для всех $p_i \in P_j$ выполняется условие $\mu_i(\theta) - f_{ij}^p(\theta) \geq 0$, т.е. если каждая входная позиция для данного перехода $p_i \in P_j$ содержит как минимум столько фишек, какова кратность дуги ведущей к t_j переходу в момент времен θ .

В результате срабатывания перехода t_j в момент времени θ разметка $M(\theta)$ сменяется разметкой $M(\theta + 1)$ по правилу:

$$\mu_i(\theta + 1) = \mu_i(\theta) - f_{ij}^P(\theta) + f_{ji}^\tau(\theta)$$

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

Если может сработать несколько переходов, то срабатывает один, любой из них. Функционирование сети останавливается, если при некоторой маркировке (*тупиковая маркировка*) 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 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.

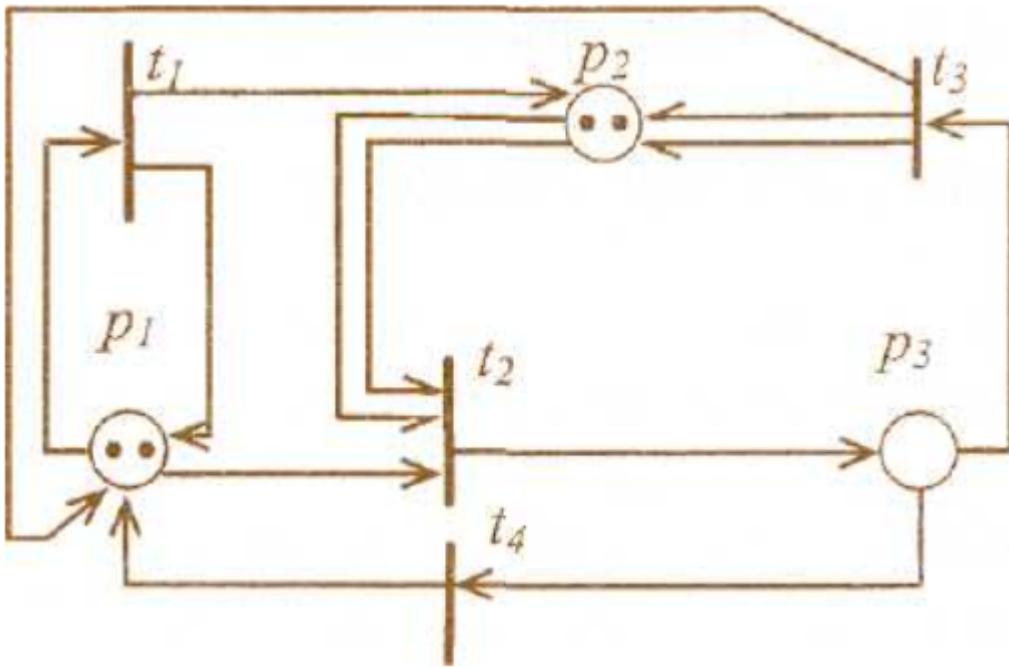
Графы сетей Петри

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

Поэтому ниже функционирование сетей Петри изложено две позиции теории графов. Теоретико-графовым представлением сети Петри является двудольный ориентированный мультиграф данной сети. Этот граф содержит:

- позиции (места), обозначаемые кружками;
- переходы, обозначаются вертикальными планками;
- ориентированные дуги (стрелки), соединяющие позиции с переходами и переходы 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:



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

$$F^p = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}, \quad F^t = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix}$$

Начальная маркировка, как видно из рисунка, $M_0 = [2, 2, 0]$. Очевидно, 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, т.е. E'' . Изменение в состоянии, вызванное запуском перехода, определяется функцией перехода S или функцией следующего состояния. Когда эта функция применяется к маркировке M и переходу t_j (если он разрешен), то в соответствии с нашим

условием работы 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 $f_{lk}^t \geq 0$.

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 $M(\tau)$ 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** $R(PN, M)$ сети Петри PN с маркировкой M есть множество всех M_k , достижимых из M .

Запуск:

$\theta = 0$ $M_0 = [2 \ 2 \ 0]$ $\theta \approx 1$

t 1: [2 3 0]

 $\theta = 2$ $\theta = 3$ $\theta = 4$

t 1: [2 4 0]

t 1: [2 5 0]

t 1: [2 6 0]

t 2: [1 3 1]

t 2: [1 2 1]

t 1: [1 3 1]

t 2: [0 0 2]

t 3: [2 4 0]

t 4: [2 2 0]

t 2: [1 1 1]

t 1: [1 2 1]

повтор

t 3: [2 3 0]

t 1: [2 4 0]

t 2: [1 1 1]

t 4: [2 1 0]

t 1: [2 2 0]

t 2: [1 0 1]

t 1: [1 1 1]

повтор

t 3: [2 2 0]

повтор

t 4: [2 0 0]

t 1: [2 1 0]

повтор

Лекция 15

Основные свойства сети Петри

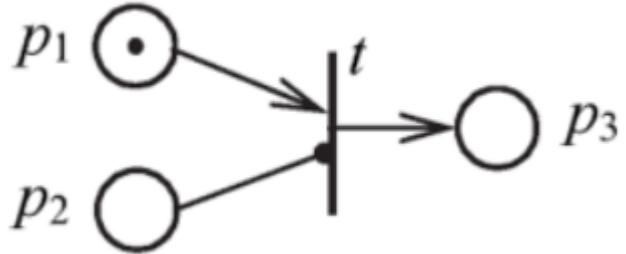
1. Свойство ограниченности – позиция в сети называется ограниченной, если для любой достижимой в сети маркировки существует такое k , что маркировка $\mu \leq k$, то есть сеть называется ограниченной, если все ее позиции ограниченные (сеть, которую мы рассматривали на предыдущем рисунке, неограниченная).
2. Свойство безопасности – в безопасной сети вектор маркировок состоит только из нулей и единиц, то есть является двоичным
3. Свойство консервативности – сеть называется консервативной, если сумма фишек во всех позициях остается постоянной при работе с ней, то есть сумма по всем τ для данной сети остается постоянной.
4. Свойство живости – переход называется потенциально живым, если в начальной маркировке существует некая маркировка, при которой переход может сработать. Если переход является потенциально живым для любой достижимой в сети маркировки, то он называется живым. Если не является – то называем мертвым, маркировка в этом случае будет называться тупиковой. То есть при тупиковой маркировки не может сработать ни один переход. Переход называется устойчивым, если никакой другой переход не может лишить его возможности сработать при наличии необходимых условий.

Ингибиторные сети

Это сети Петри, для которых функция инцидентности имеет вид F^P т.е. она дополнена специальной функцией инцидентности

Ингибиторные дуги связывают только позиции с переходами, кратность этих дуг равна 1. Правила срабатывания перех.

Это сети Петри, для которых функция инцидентности имеет вид $F = F^P \cup F^t \cup F^I$, т. е. 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, эти дуги на рисунках заканчиваются не стрелками, а кружочками. Кратность этих дуг всегда равна 1.



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 .

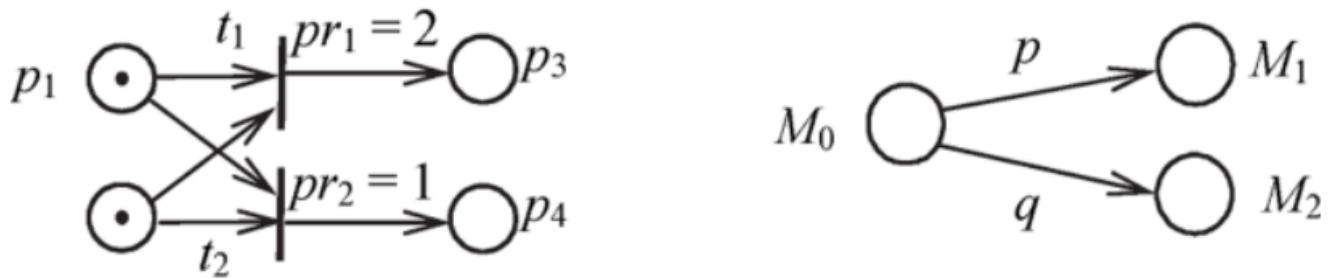
$$(\mu_i \geq f_{ij}^p) \wedge (\mu_k \cdot f_{kj}^I = 0),$$

то есть введено дополнительное условие: позиция p_k , соединенная с переходом t_j ингибиторной дугой, не должна содержать фишек (должна иметь нулевую маркировку). Так, переход t на рисунке выше 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 первым должен сработать переход t_2 ,

имеющий приоритет $pr_2 = 1$, поскольку приоритет перехода $t_1 \circ t_2 = 2$, то есть ниже.



Сети со случайными срабатываниями переходов

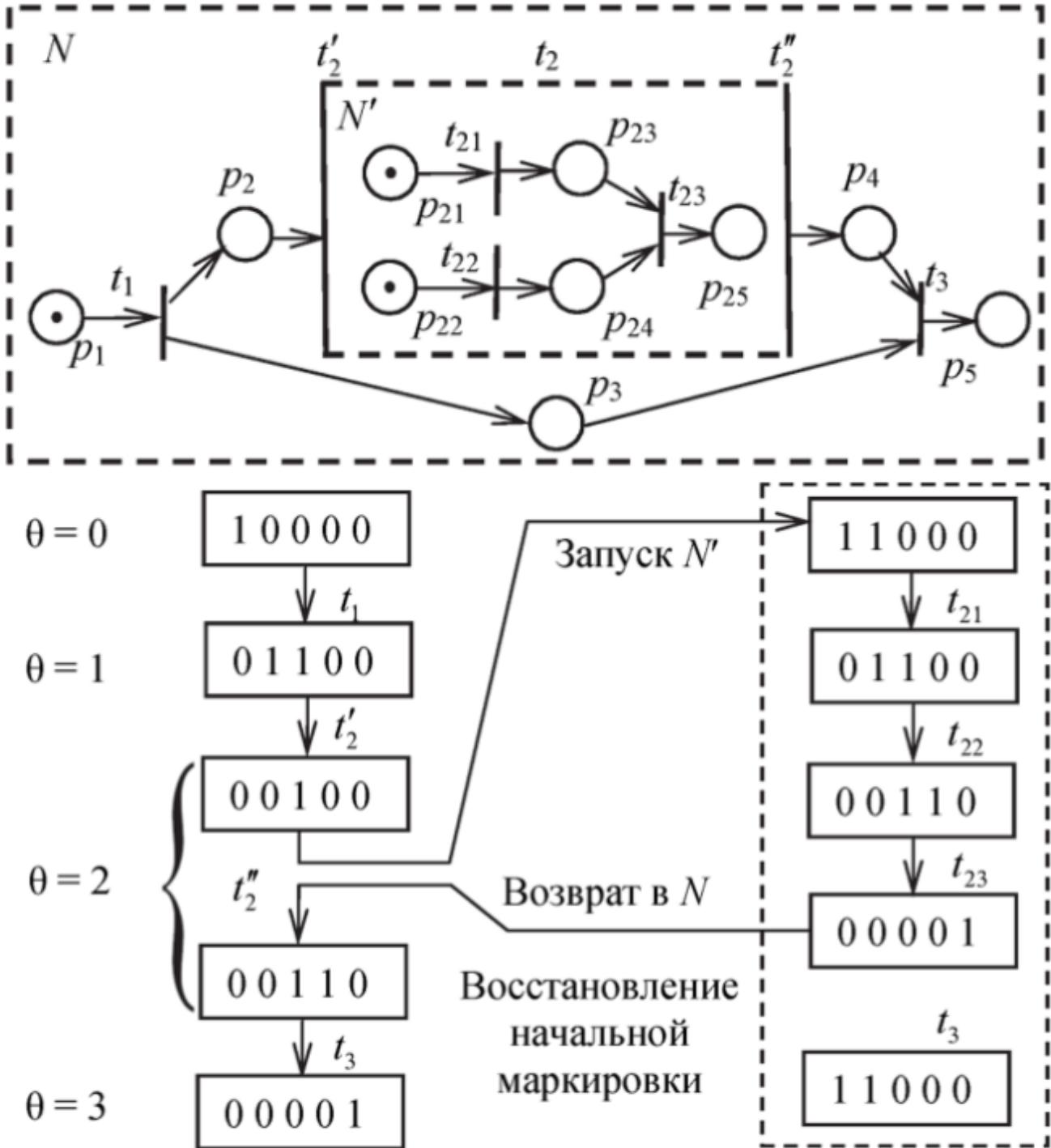
В описанной выше ситуации, когда имеется возможность срабатывания нескольких переходов t_i, t_j, \dots, t_s , их приоритет можно задавать вероятностями срабатывания каждого их переходов p_i, p_j, \dots, p_s , причем $p_i + p_j + \dots + p_s = 1$. Тогда исходная маркировка M_0 приведет на следующих шагах 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 предположив, что вероятности не зависят от работы сети в предыдущие такты, мы получим цепь Маркова, описывающую вероятностное поведение системы.

Иерархические сети

Сети представляют собой многоуровневые структуры, в которых выделяют сети различных уровней (позволяют моделировать различные многоуровневые иерархические системы).

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

Пример иерархической сети N:



Здесь приведен пример иерархической сети который имеет составной переход t_2 , содержащий внутри себя сеть N' этот составной переход имеет голову (t'_2) и хвост (t''_2). Между ними заключена вся сеть. Она состоит из позиций p_{21}, p_{25} и переходов t_{21}, t_{23} . Иерархическая сеть функционирует как обыкновенная сеть, переходя от одной маркировки к другой, обмениваясь фишками (в том числе между сетями различного уровня). Исключение составляют правила работы составных переходов.

Срабатывание составных переходов является не мгновенным событием, как в обычновенных сетях Петри, а некоторым составным действием. Поэтому говорят не о срабатывании составного перехода, а о его работе. На каждом шаге дискретного времени θ 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 θ , если до этого он был пассивен и имеются условия для срабатывания его головного перехода. При этом производится изменение маркировки в сети верхнего уровня по известным нам правилам 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. Составной переход возвращается в пассивное состояние, а в сети нижнего уровня восстанавливается начальная маркировка.

На шаге $\theta = 2$ происходит работа составного перехода и сети N' в следующем порядке: срабатывает t'_2 , запуск сети N' , окончание работы N' , восстановление начальной маркировки, срабатывание t_2 и продолжение работы сети.

Описанный процесс напоминает выполнение подпрограммы при программировании, где срабатывание перехода t'_2 соответствует вызову подпрограммы, а t''_2 срабатывание - возврату в основную программу.

Раскрашенные (цветные) сети

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.

Моделирование дискретных систем, формализованных сетей Петри

При описании сетей Петри выделяют два понятия: события и условия.

События – это действие в системе. В сетях Петри они моделируются переходами.

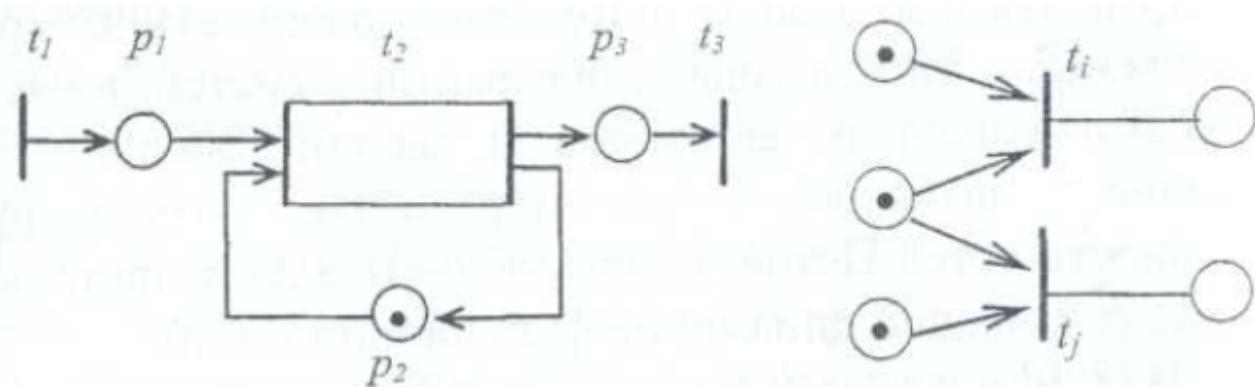
Условие – предикат или логическое описание системы, принимающее значение "истина" или "ложь". Условия моделируются позициями и условиями на дугах. Различаются предусловия и постусловия.

Предусловие – это условие до срабатывания перехода, **постусловие** – соответственно, условие после срабатывания перехода.

Если процесс в системе достаточно сложный, то его подсистемы можно представить в виде **непримитивных событий**.

Особенность Сети Петри – **одновременность**. Если переходы t_i и t_j не влияют друг на друга, то в возможный словарь языка сети Петри входят как слова, начинающиеся с t_i так и слова, начинающиеся с t_j .

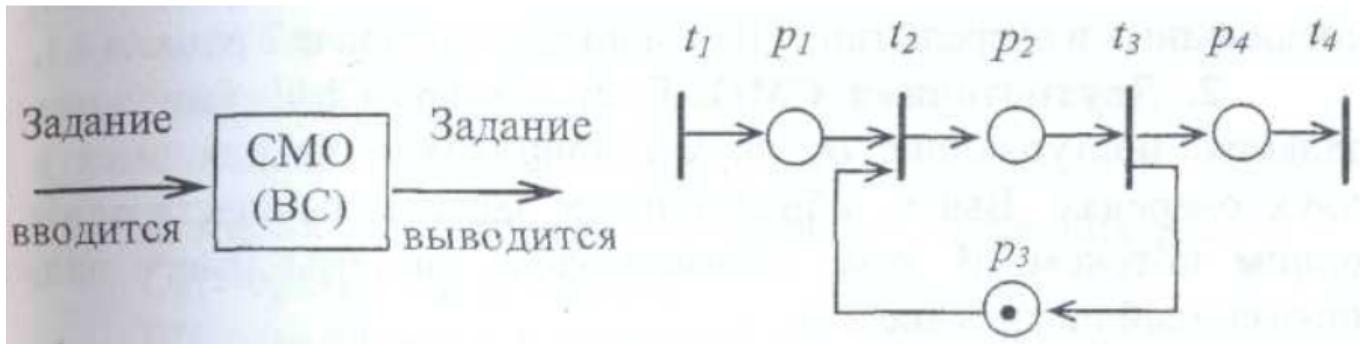
Еще одна ситуация называется **конфликтом**. Переходы t_i , и t_j находятся в конфликте, если запуск одного из них блокирует запуск другого.



Простейшая система массового обслуживания.

Система имеет входной поток заданий (или заявок), и пока она занята выполнением очередного задания, она не может ввести следующее.

Рассмотрим множество условий и событий, характеризующих нашу СМО.



Условия:

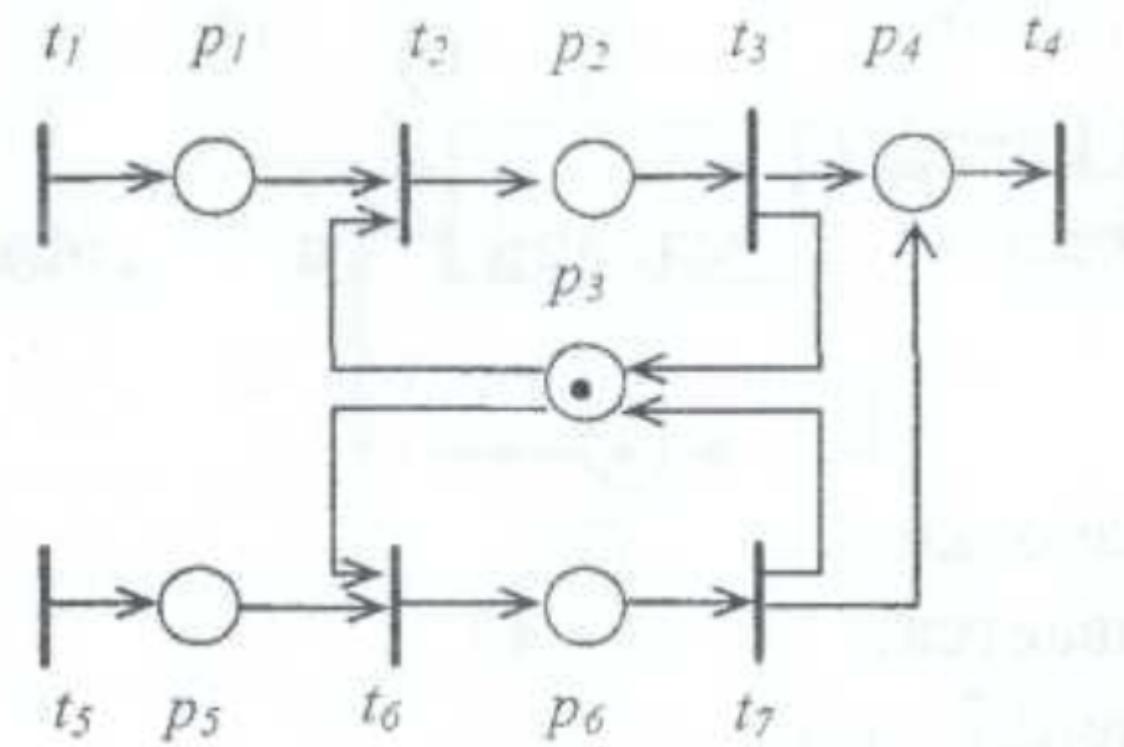
- P_1 - задание ждет обработки;
- P_2 - задание обрабатывается;
- P_3 - процессор свободен;
- P_4 - задание ожидает вывода.

События (это действия):

- t_1 - задание помещается во входную очередь;
- t_2 - начало выполнения задания;
- t_3 - конец выполнения задания;
- t_4 - задание выводится.

Поясним работу данной сети. Показанная на рисунке начальная маркировка $M_0 = [0, 0, 1, 0]$ соответствует состоянию, когда система свободна и заявки на обслуживание отсутствуют. При срабатывании перехода t_1 (от внешнего источника) поступает задание и получается маркировка $M_1 = [1, 0, 1, 0]$. При этом может сработать переход t_2 , что означает начало обслуживания задания и приводит к маркировке $M_2 = [0, 1, 0, 0]$. Затем может сработать переход t_3 , что означает окончание обслуживания задания и освобождение системы, т.е. переход к маркировке $M_3 = [0, 0, 1, 1]$. Переходы t_1 и t_4 могут работать независимо от t_2 и t_3 , моделируя поступление и вывод заданий.

Двухпоточная СМО. Пусть теперь СМО выполняет задания, поступающие от двух источников и находящиеся в двух очередях. Вывод обработанных заданий осуществляется одним потоком.



Здесь введены дополнительные **условия**:

- P_5 – задание из второй очереди ждет обработки;
- P_6 - задание из второй очереди обрабатывается.

Также введены дополнительные **события**:

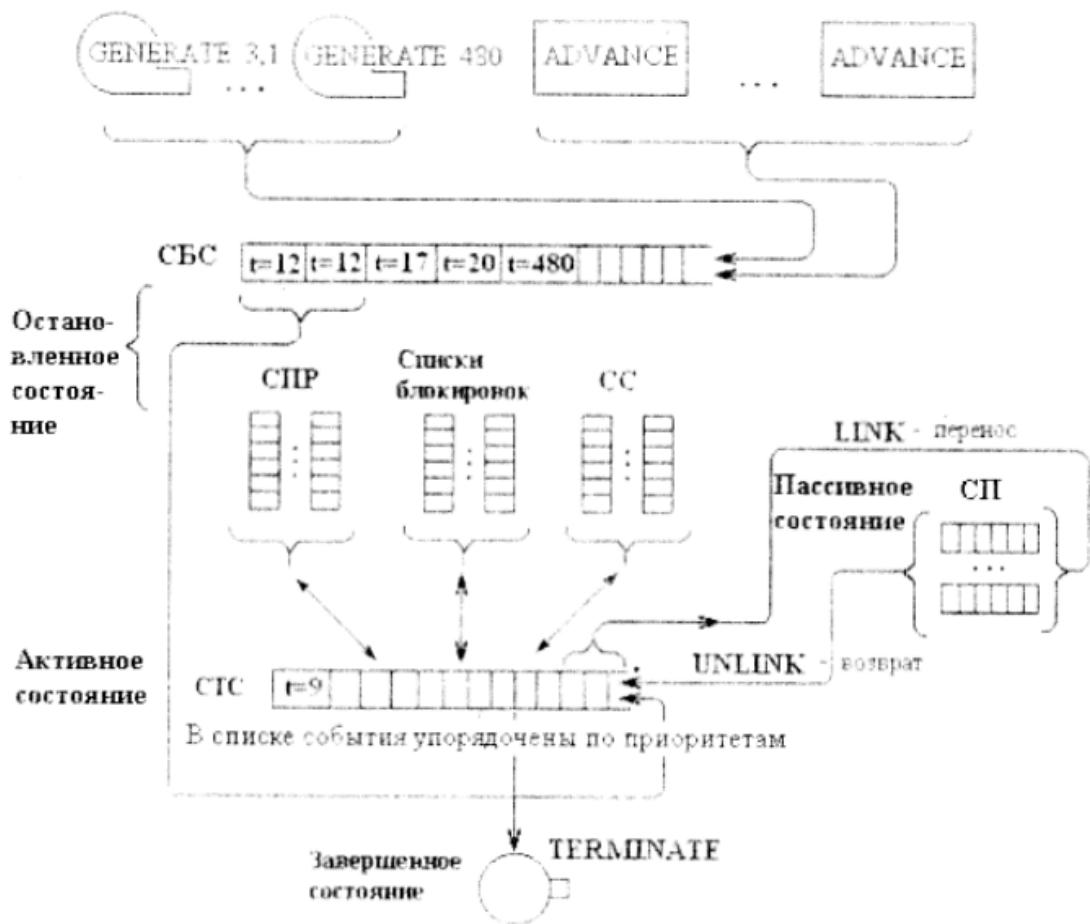
- t_5 - задание помещается во вторую очередь;
- t_6 - начало выполнения задания из второй очереди;
- t_7 - завершение выполнения задания из второй очереди.

Лекция 16. Язык General Purpose System Simulation (GPSS)

Язык GPSS – общецелевая система моделирования, предназначенная для того, чтобы выполнить анализ правильности функционирования любой системы с очередями.

Система GPSS построена в предположении, что моделью сложной дискретной системы является описание ее элементов и логических правил их взаимодействия в процессе функционирования моделируемой системы. Для определенного класса моделируемых систем в GPSS можно выделить конечный набор абстрактных элементов, называемых **объектами**, причем набор логических правил также ограничен и может быть описан небольшим числом стандартных операций. Объекты языка подразделяются на 7 категорий и 14 типов.

Категория	Типы
Динамическая	Транзакция
Операционная	Блоки
Аппаратная	Устройства памяти, ключи
Вычислительная	Переменные, функции
Статическая	Очереди, таблицы
Запоминающая	Ячейки, матрицы ячеек
Группирующая	Списки, группы



Моделирование заканчивается тогда, когда счетчик завершений (стандартный числовой атрибут), инициализированный оператором START будет равен 1. Или когда в списках СТС и СБС не будет ни одного транзакта.