

1. Философские аспекты моделирования

Два преимущества моделирования:

1. дешевизна;
2. можно реализовать любые фантастические условия

Методологическая основа моделирования – это диалектический метод познания

(придумал Гегель, основан на признании всеобъемлющей взаимосвязи предметов и явлений и их непрерывного развития).

Адекватность модели – "приближенность" к реальному миру. 20% факторов определяют 80% функционирования объекта

Объект – все, на что направлена человеческая деятельность.

Научно-техническое развитие в любой области обычно идет следующим путем: наблюдение и эксперимент, теоретическое исследование, организация производственных процессов

Гипотеза – предсказание, основывающееся на небольшом количестве опытных данных, наблюдениях, догадках.

Быстрая и полная проверка выдвигаемых гипотез может быть проведена в ходе специально поставленного эксперимента

Аналогия – суждения о частном сходстве и различии объектов. Современная научная гипотеза создается по аналогии: гипотеза->аналогия->эксперимент.

Причем такое сходство может быть существенным или несущественным. Существенные сходства (различия) условны и относительно - зависит от уровня абстрагирования и в общем случае определяется конечной целью проводимого исследования.

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 эксперимент, уточняющий природу явления, называют **моделью**.

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

Моделирование – замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели.

В основе моделирования лежит теория подобия, которая утверждает, что абсолютное подобие имеет место только при замене одного объекта другим точно таким же.

Главный вопрос – можем ли мы познать объект с помощью методов моделирования?

Гносеологическая роль теории моделирования (её значение в теории познания), заключается в том, что изучение моделей, выступает в роли относительно самостоятельного квазиобъекта, позволяет получить при исследовании некоторые данные о самом объекте. Причем по отношению к моделям исследователь является экспериментатором, только эксперимент проводится не с объектом, а с моделью.

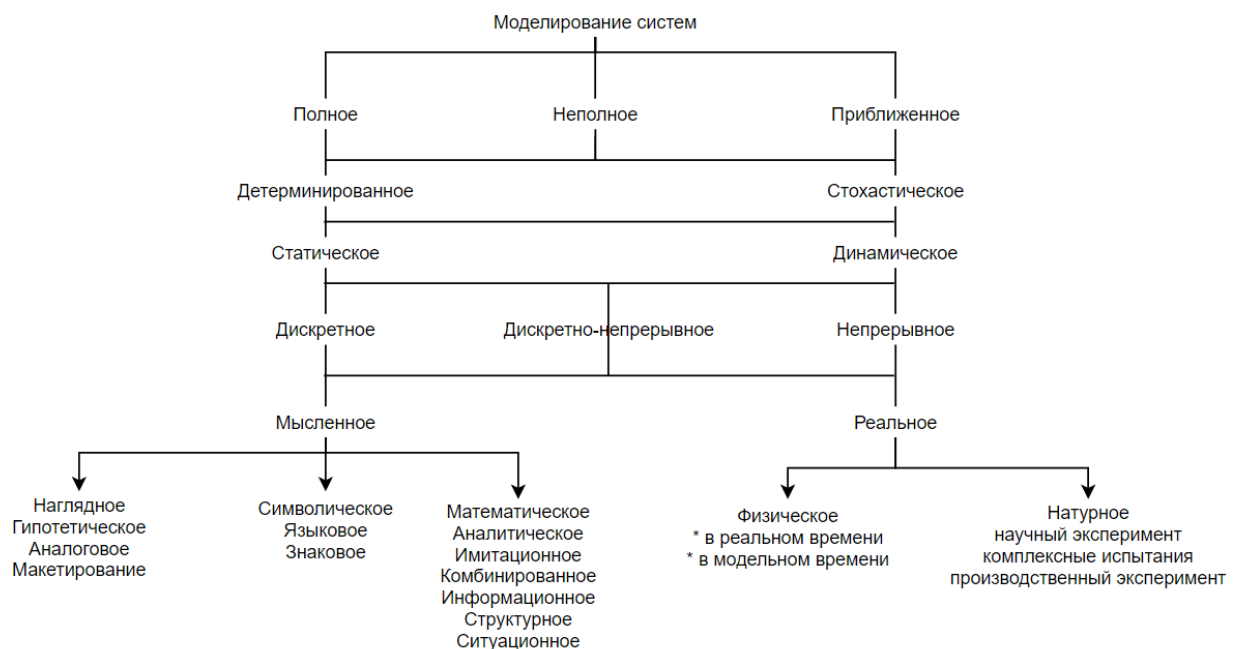
Любой эксперимент может иметь существенное значение в конкретной области науки и техники, только при его специальной обработке и обращении. Единичный эксперимент не может быть решающим для подтверждения гипотезы или теории.

Подобия:

1. Полное
2. Неполное
3. Приближенное

Методика – упорядоченная совокупность методов

2. Классификация видов моделирования



В качестве признака классификации берем характер изучаемых процессов.

Математическое

Под математическим моделированием будем понимать процесс установления данному реальному объекту некоторого математического объекта, называемой математической моделью и исследовании этого объекта с целью получения характеристик реального объекта.

Вид математической модели зависит как от природы рассматриваемого объекта, так и от цели моделирования. Математическая модель описывает реальный объект с некоторой степенью приближения.

Аналитическое

Для аналитического моделирования характерно то, что в процессе функционирования системы процессы записываются в виде логических условий и функциональных соотношений (алгебраических, интегро-дифференциальных, конечно-разностных и т.д.). Отображена связь вход-выход.

Аналитическая модель может быть исследована 3-я способами:

- аналитическим –получить в общем виде зависимости от исходных характеристик;
- численным – если уравнение в общем виде решить нельзя, но может получать решения уравнений для конкретных данных;
- качественным – не имея решения, можно найти некоторые свойства решения (например, оценить его устойчивость).

Имитационное

При имитационном моделировании реализующий модель алгоритм воспроизводит процесс функционирования системы во времени, причем имитируются реальные явления с сохранением их логических структур и последовательностей протекания во времени, что позволяет по исходным данным получить сведения о состоянии процесса в определенный момент времени, дающие возможность оценить характеристики системы.

Основное преимущество по сравнению с аналитическим методом – возможность решения более сложных задач.

Имитационная модель позволяет достаточно просто учитывать такие факторы как:

- наличие дискретных и непрерывных элементов;
- нелинейные характеристики;
- многочисленные случайные воздействия.

Имитационная модель часто подразумевается тем же самым что и программная.

Результаты выдачи имитационной модели должны быть статистически обработаны из-за большого числа вероятностных параметров, лежащих в основной модели.

Целесообразно в качестве метода компьютерной имитационной модели использовать метод стохастического моделирования – метод Монте-Карло. Это численный метод применяемый для моделирования случайных величин и функций, вероятностные характеристики которых совпадают с решением аналогичных задач.

Комбинированный метод позволяет объединить достоинства разных видов. Самое важное – декомпозиция процесса функционирования на подпроцессы так, чтобы описать часть из них аналитически, а часть - имитационно.

Виды имитационного моделирования

1. Агентное

Используется для исследования децентрализованных систем, динамика функционирования которых выделяется не глобальными правилами и их законами, а когда эти глобальные правила являются результатами индивидуальной активности членов группы.

Цель – получить представление об общем поведении системы исходя из предположений об индивидуальном поведении отдельных активных агентов

Агент – некоторая сущность, обладающая активностью, автономным поведением. Может принимать решение в соответствии с некоторым набором правил взаимодействия с окружением, а также самостоятельно изменяться.

Активность поглощает модельное время

Пример: транспортные потоки

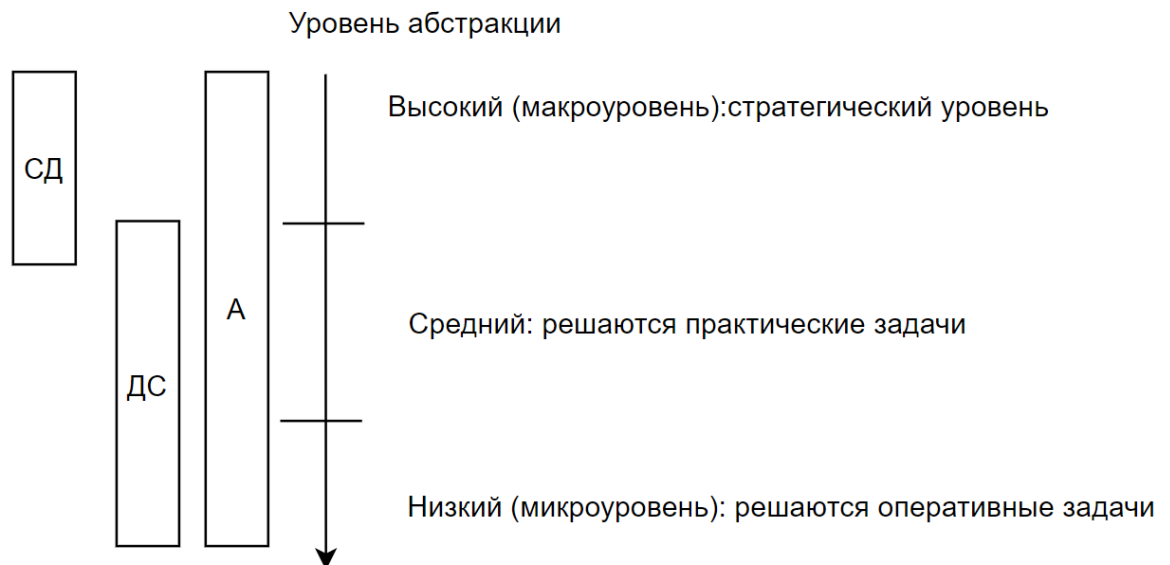
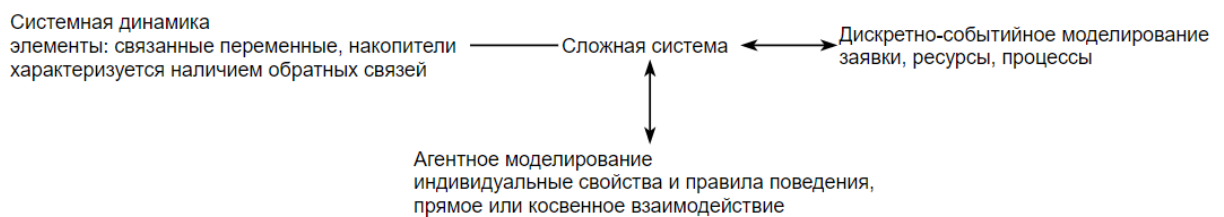
2. Дискретно-событийное

Предлагается абстрагироваться от непрерывной природы событий и рассматривать только основные события моделируемой системы (ожидание/обработка заказа и т.д.). *Наиболее развито*

3. системная динамика

Для исследования системы строятся графические диаграммы причинных связей и глобальных влияний одних событий на другие во времени. Связи – между объектами и явлениями. Созданная диаграмма моделируется с помощью компьютера.

Пример: бизнес-процессы, динамика популяций и т.д.



Недостатки имитационного моделирования:

Трудности имитационного моделирования связаны с:

- обеспечением адекватности описания системы;
- Интерпретацией результатов
- Обеспечением стохастической сходимости моделирования
- Уменьшением размерности
- Большой трудоемкостью

Перед построением модели (динамической по своей сути) полезно или даже необходимо осуществить предварительный статистический анализ системы.

При этом определяются и специфицируются функции, выполняющиеся в системе, взаимосвязи, потоки работ и тд. Именно эти спецификации позволяют модели выявлять необходимый объем знаний о системе до создания полной имитационной модели (уменьшение размерности имитационной модели, что повышает эффективность).

На этом этапе составляется тз можно ограничить входные данные

3 этапа развития средств имитационного моделирования (по возможностям и функциям исследователя)

1. создание имитационной модели с помощью ЯП: на специальном языке имитационного моделирования (GPSS, ...), на ОО языке имитационного моделирования (mod, sym)
2. использование при разработке имитационной модели проблемно-ориентированных систем и средств (matlab).

Как правило не требует навыков программирования, но могут моделировать только узкий класс систем, при этом модель строится самой системой (в ходе диалога с пользователем).
Настраиваем модель на объект, выделяя самые существенные признаки

3. использование методов ИИ, основанных на знаниях ...при принятии решений.
Продуманный интерфейс

Нерешенной проблемой неформального этапа остается перевод поставленной задачи на язык математики. Где одна проблема – наличие сложной системы управления.

3. Технические средства математического моделирования

Использование компьютера:

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

Принципиально можно выделить 2 типа: цифровые и аналоговые. А-быстрый (скорость ограничивается скоростью передвижения электрона в цепи), но неточный (всё представляется сигналами), Ц-точный (сложение и сдвиг)

АВМ ускоряет процессы принятия решений, но не обеспечивает высокую точность.

ЦВМ

Процессор – устройство, предназначенное для выполнения арифметических операций (базово сложение и сдвиг) и управления над ними. процессоры отличаются набором команд. Самая большая проблема ограничение быстродействия



Аналоговые вычислительные машины

В отличие от дискретной техники в аналоговых компьютерах заложен принцип моделирования, а не счета. В качестве модели определенной задачи используются электронные цепи. Каждой переменной величине задачи ставится в соответствие переменная величина электронной цепи. Основой построения такой модели является изоморфизм (подобие) исследуемой задачи и соответствующей ей электронной модели. В большинстве случаев при определении критерия подобия используются специальные приемы масштабирования, соответствующих значений параметров модели и переменных задач.

Согласно своим вычислительным возможностям аналоговые машины наиболее приспособлены для исследования объектов, динамика которых описывается обыкновенными и в частных производных дифференциальными уравнениями, а также алгебраическими и рядом других => АВМ можно отнести к специальным машинам.

Под **АВМ** будем понимать совокупность электрических элементов, организованных в систему, позволяющую изоморфно моделировать динамику изучаемого объекта.

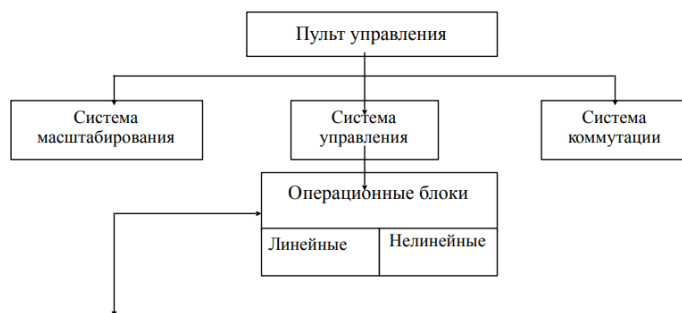
АВМ делятся по мощности (степень дифференциальных уравнений):

- малые ($n < 10$),
- средние ($10 \leq n \leq 20$),
- большие ($n > 20$)

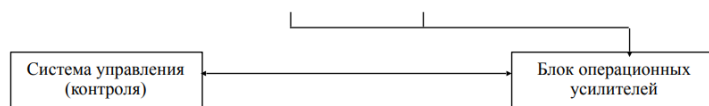
Структурная схема АВМ:

Система управления пронизывает все блоки

Пульт управления играет важнейшую роль: с его помощью задаются исходные данные



8



Сравнительная характеристика АВМ и ЦВМ:

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

Гибридные вычислительные машины

Под **ГВМ** будем понимать широкий класс вычислительных систем, использующих как аналоговую, так и дискретную формы представления и обработки информации.

ГВМ соединяют в себе высокую точность и для некоторых объектов повышают скорость. ГВМ объединяют в себе узлы и блоки типовых и специальных ВМ **с использованием различных форм представления информации и различных методов ее переработки.** (различный коммутаторы)

Применение гибридной вычислительной техники:

1. моделирование дискретных цифровых сигналов и случайных процессов;
2. решение задач оптимизации (в том числе многокритериальной)
3. исследование в области управления подвижными объектами
4. моделирование системы "человек - компьютер"

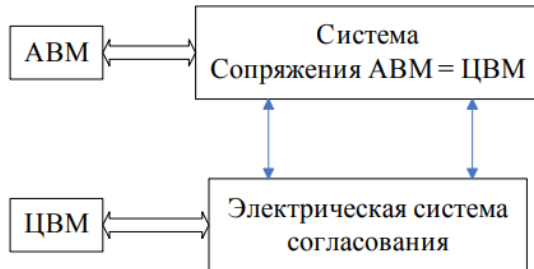
Направления развития ГВМ:

1. на основе дискретно-управляемых элементов, меняющих свои параметры под воздействием управляющего кода
2. разрядно-аналоговые: обеспечивают высокую точность за счёт цифровой формы представления информации и аналогового способа ее переработки

3. цифровые интегрирующие машины: специальные ЭВМ, но в отличие от ЦВМ в качестве основной операции идет интегрирование (а не суммирование и алгебра логики)

(тут электрическая система согласования аналоговой и цифровой информации)

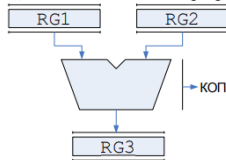
Гибридная ВМ:



Подклассы ГВМ (от аналоговой к цифровой):

1. АВМ, использующие цифровые методы численного анализа
2. АВМ, программируемые (вычисление масштабных коэффициентов) с помощью ЦВМ
3. АВМ с цифровым управлением и логикой
4. АВМ с цифровыми элементами (например, память, цифровые вольтметры)
5. ЦВМ с аналоговыми арифметическими устройствами
6. ЦВМ, допускающие программирование аналогового типа (например, дифференциальные анализаторы)

». ЦВМ с аналоговыми арифметическими



5

4. Основные понятия теории моделирования

В теории систем имеются специальные базовые понятия:

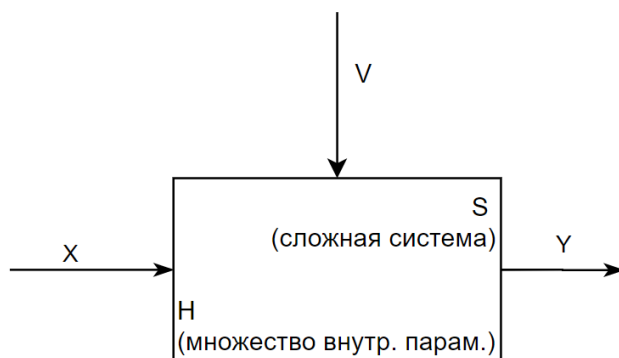
1. **Система** – это множество элементов, находящихся в отношениях и связях между собой
2. **Элемент** – часть системы, представление о котором нецелесообразно подвергать дальнейшему членению при моделировании.
3. **Сложная система** – система, характеризующаяся большим числом элементов и, что наиболее важно, взаимосвязей. Сложность системы определяется также видом взаимодействий элементов, свойствами:
 - a. Целенаправленности (свойство искусственной системы, выражающее назначение системы для оценки эффективности вариантов системы)
 - b. Целостности (свойство системы, характеризующее взаимосвязанность элементов и наличие зависимости выходных параметров от параметров элементов. При этом большинство выходных параметров не является повторением или суммой параметров элементов);
 - c. членимости;

- d. **иерархичности** (свойство сложной системы, выражающее возможность и целесообразность ее иерархического описания, т.е. представления в виде уровней, между компонентами которыми имеется отношение "целое - часть");
- e. **многоаспектности**
4. **Подсистема** – часть системы (подмножество элементов и их взаимосвязи), которая имеет свойства системы.
 5. **Надсистема** – система, по отношению к которой рассматривается система является подсистемой
 6. **Структура** – отображение совокупности элементов системы и их взаимосвязей. От понятия "система" отличается также тем, что при описании структуры принимают во внимание описание лишь типов элементов и связей, без конкретизации значений их параметров.
 7. **Параметр** – величина, или выражающая свойство системы или ее части, или влияющая на систему среды.

Задачи моделирования сложной системы:

1. modelling – создание модели системы
2. simulation – анализ свойств системы на основе использования ее модели

Основы теории моделирования



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

- совокупность входных воздействий $x_i \in X, i = 1, nx$
- совокупность воздействий внешней среды $vl \in V, l = 1, nl$
- совокупность внутренних собственных параметров системы $hk \in H, k = 1, nk$
- совокупность выходных характеристик системы $yj \in Y, j = 1, nj$

В общем случае, x_i, vl, hk, yj – элементы непересекающихся подмножеств и содержат как детерминированные, так и статистические составляющие.

При анализе функционирования сложной системы входные воздействия, воздействия внешней среды и внутренние параметры являются независимыми, то есть экзогенными переменными, которые в векторной форме имеют вид:

- $\vec{x}(t) = (x_1(t), x_2(t), \dots, x_{nx}(t))$
- $\vec{v}(t) = (v_1(t), v_2(t), \dots, v_{nl}(t))$

- $\vec{h}(t) = (h_1(t), h_2(t), \dots, h_n(t))$

А выходные характеристики являются зависимыми (эндогенными) переменными и имеют вид:

- $\vec{y}(t) = (y_1(t), y_2(t), \dots, y_n(t))$

Процесс функционирования системы S описывается во времени некоторым оператором, который в общем случае преобразует независимые переменные в зависимые. В общем случае он может быть задан функцией, функционалом, логическими условиями, в алгоритмическом или табличном виде. **Это закон функционирования системы.** Самый важный параметр – время.

- $\vec{y}(t) = F_s(\vec{x}, \vec{v}, \vec{h}, t) \quad (1)$

Под **алгоритмом функционирования** сложной системы будем понимать **метод** получения выходных параметров с учетом входных воздействий, воздействий внешней среды и внутренних параметров системы.

Один и тот же закон функционирования F_s может быть реализован с помощью множества алгоритмов функционирования (например, сортировки)

Соотношение 1 может быть получено через свойства системы в конкретные моменты времени, которые называются состояниями вроде и характеризуются вектором состояний

- $\vec{Z}(t) = (z_1(t), z_2(t), \dots, z_k(t)), k = 1, n_z$

Если рассматривать функционирование системы как последовательную смену состояний, то они могут быть интерпретированы как координаты точки в k -мерном фазовом пространстве, причем каждой реализации соответствует некоторая фазовая траектория.

Совокупность $\{\vec{Z}(t)\}$ всех возможных состояний системы называют **пространством состояний объекта моделирования**.

Состояние системы S в момент времени t от начала t_0 и до конца моделирования t_k

полностью определяется начальным состоянием (0 наверху),

- $\vec{Z}_0 = (z_{01}, z_{02}, \dots, z_{0k}), z_{01} = z_1(t_0).$

входными воздействиями, внутренними параметрами и воздействиями внешней среды, которые имеют место за промежуток времени $t - t_0$, с помощью 2 векторных уравнений

- $\vec{z}(t) = \Phi(\vec{z}_0, \vec{x}, \vec{v}, \vec{h}, t)$
- $\vec{y}(t) = F(\vec{z}, t)$

Откуда

- $\vec{y}(t) = F(\Phi(\vec{z}_0, \vec{x}, \vec{v}, \vec{h}, t))$

В общем случае время м.б. непрерывным или дискретным (квантованным), откуда появляется понятие числа интервалов дискретизации – еще один параметр системы

Под математической моделью реальной системы понимают конечное множество параметров \vec{x} , \vec{v} , \vec{h} вместе с математическими связями между ними и характеристиками $\vec{y}(t)$

Главное – введение закона функционирования в виде 1, оно позволяет:

- выделить множество входных характеристик, а главное внутренних парам;

- показать, как F_s переводит эти значения в выходные значения;
- даются ограничения на то, что делается

6. Типовые математические схемы сложных систем (см также 5)

На первоначальных этапах формализации используют типовые математические схемы

процесс функционирования системы	типовая математическая схема	обозначения
непрерывно детерминированный подход	дифференциальные уравнения	D-схемы
дискретно детерминированный подход	конечные автоматы, мур	F-схемы
непрерывно-стохастический подход	системы массового обслуживания, системы с очередями	Q-схемы
дискретно-стохастический подход	вероятностные автоматы	P-схемы
обобщенный (универсальный)	агрегатные системы	A-схемы

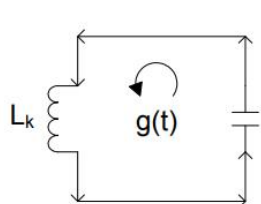
В А-схеме какие-угодно блоки, самое главное определить операции над ними. Переходя со слоя на слой пытаемся построить математическую схему для каждого слоя. Самая важная операция для агрегатных систем - композиция и декомпозиция.

!5. Непрерывно-детерминированные модели (D-схемы) (см 4,5)

(у нас вроде не было)

Если неизвестны функции многих аргументов частных производных, если одна неизвестная - обыкновенное дифференциальное уравнение.

Называется D-схема т.к. схемы такого вида отображают динамику изучения схемы. Независимая переменная обычно время



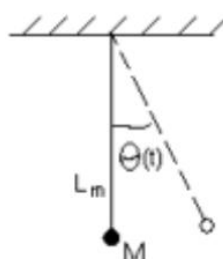
ДУ: $y' = f(y, t)$

$$L_k \frac{d^2 q(t)}{dt^2} + \frac{q(t)}{C_k} = 0$$

- описание процессов в электрическом колебательном контуре на базе ДУ.

$q(t)$ - заряд конденсатора в момент времени t

$T = 2\pi \sqrt{L_k C_k}$ - период колебаний



$$M_M I_M^2 \frac{d^2 \theta(t)}{dt^2} + m_M g l_M = 0$$

$T_n = 2\pi \sqrt{\frac{I_M}{g}}$ - период

$$h_2 = L_k = M_M l_M^2$$

$$h_1 = 0$$

$$h_0 = \frac{1}{C_k} = M_M g l_M$$

$$h_2 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_0 z(t) = 0$$

h_1, h_2, h_3 - внутренние параметры системы
 $z(t)$ - состояние системы в момент времени t

Если рассматриваемая система взаимодействует с внешней средой, то появляется входное воздействие $x(t)$ и непрерывно детерминированная система имеет вид:

$$h_2 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_0 z(t) = x(t)$$

При этом состояние системы S в данном случае можно рассматривать как выходную характеристику, т.е. полагать, что $y = z$., следовательно, использование D – схем позволяет формализовать процесс функционирования непрерывно детерминированных систем и оценить их характеристики применяя аналитический или имитационный подход, реализованный в виде соответствующего языка моделирования непрерывных систем или использования аналоговой или гибридных средств вычислительной техники.

7. Основные требования, предъявляемые к модели при её разработке и машинной реализации

Формализация и алгоритмизация процесса функционирования сложных систем.

Сущность компьютерного моделирования сложной системы состоит в проведении на компьютере эксперимента с моделью, которая представляет собой некоторый программный комплекс, описывающий формально или алгоритмически поведение элементов системы в процессе её функционирования, т.е. в их взаимодействии друг с другом и внешней средой.

Основные требования, предъявляемые к модели.

1. Полнота модели – должна предоставлять пользователю возможность получения необходимого набора характеристик, оценок системы, с требуемой точностью и достоверностью.
2. Гибкость модели – должна давать возможность воспроизведения различных ситуаций при варьировании структуры, алгоритмов и параметров системы. Причем структура должна быть блочной, т.е. допускать возможность замены, добавления, исключения некоторой части без переделки всей модели.
3. Компьютерная реализация модели должна соответствовать имеющимся техническим ресурсам.

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 решения поставленной задачи

Картинки не было



8. Основные этапы моделирования больших систем

1. Построение концептуальной (описательной) модели системы и её формализация (первые 6 слайдов)

На первом этапе формулируется модель и строится её формальная схема, т.е. основным назначением данного этапа является переход от содержательного описания объекта к его математической модели. Этот этап наиболее ответственный и наименее формализованный.

Последовательность действий:

1. Проведение границы между системой и внешней средой.
2. Исследование моделируемого объекта с точки зрения выделения основных составляющих функционирования системы **по отношению к цели моделирования**. Что является основным критерием выявления основных составляющих функциональной системы? – Цель!
3. Переход от содержательного описания модели к формализованному описанию свойств функционирования модели, т.е. к её концептуальной модели. Это сводится к исключению из рассмотрения некоторых второстепенных элементов. Пример: завышенные требования к модели – реализация на супер-компьютере.
4. Оставшиеся элементы модели группируются в блоки:
 - 4.1. имитатор событий внешних воздействий.
 - 4.2. собственно модель исследуемой системы
 - 4.3. вспомогательные блоки, служат для реализации блоков I и II группы, интуитивно понятный интерфейс, обеспечивают корректность ввода данных, приемлемость результатов и т.д.
5. Процесс функционирования системы, так разбивается на подпроцессы, чтобы построение модели подпроцесса было элементарно и не вызывало особых трудностей.

Элементарно=должно реализовываться подбором типовых математических схем

2. Алгоритмизация модели и её машинная реализация

Математическая модель, сформированная на первом этапе, воплощается в конкретную компьютерную модель. Исходный материал – блочная логическая схема.

Последовательность действий:

1. Разработка схемы моделирующего алгоритма.
2. Разработка схемы программы.
3. Выбор технических средств для реализации программной модели.
4. Процесс программирования и отладки.
5. Проверка достоверности программы на тестовых примерах. (тестирование, отладка> тестирование)
6. Составление технической документации.

3. Получение и интерпретация результатов моделирования;

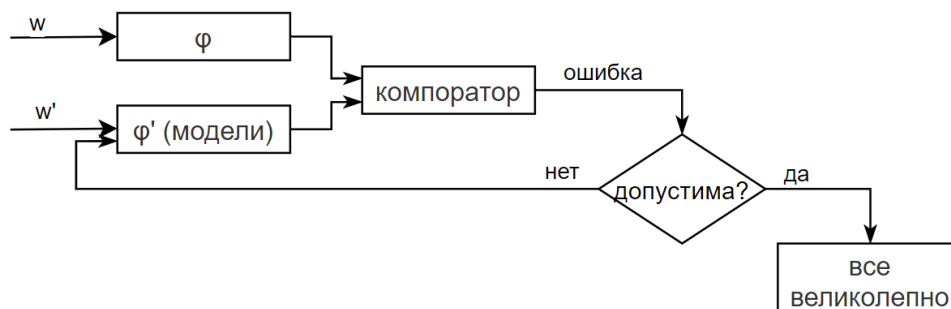
Проведение рабочих расчетов по готовой программе. Результаты этих расчетов позволяют проанализировать и сделать выводы о характеристиках процессов функционирования системы

Последовательность действий:

1. Планирование машинного эксперимента с моделью. Составление плана проведения эксперимента с указанием комбинаций, переменных и параметров для которых должен проводится эксперимент. Главная задача – дать максимальный объем информации об объекте моделирования при минимальных затратах машинного времени.
2. Проведение собственных рабочих расчетов (контрольная калибровка модели).
3. Статистическая обработка результатов расчетов и представление результатов в наглядной форме.
4. *Интерпретация результатов моделирования. Подведение итогов.*
5. Составление технической документации (сшивание РПЗ)

Завершается этот этап определением области пригодности модели. Под областью пригодности модели понимается множество условий, при соблюдении которых точность результатов моделирования находится в допустимых пределах.

Схема итеративной калибровки модели.



Ошибки:

1. Ошибки формализации. Как правило возникают, когда модель недостаточно подробно определена.
2. Ошибки решения. Некорректный или слишком упрощенный метод построения модели.
3. Ошибки задания параметров модели.

Проверка адекватности модели некоторой системы заключается в анализе её соизмерностей, а также равнозначности системы.

Адекватность часто нарушается из-за идеализации внешних условий и режимов функционирования, пренебрежения некоторыми случайными факторами. Простейшей мерой адекватности может служить отклонение ΔY некоторой характеристики Y -оригинала от Y -модели. Считают, что модель адекватна, если вероятность того что отклонение не превышает предельной величины Δ , больше допустимой вероятности.

Однако, фактическое использование данного критерия невозможно, т.к. для проектируемых или модернизируемых систем, отсутствует информация по выходным характеристикам объекта. Система отслеживается не по одной, а по множеству характеристик.

Замечание: Характеристики могут быть случайными величинами и функциями.

Замечание: Отсутствует возможность априорного точного задания предельных отклонений и допустимых вероятностей.

На практике оценка адекватности обычно проводится путем экспертного анализа разумности результатов моделирования.

Виды проверок:

- Проверка моделей элементов
- Проверка моделей внешних воздействий
- Проверка концептуальной модели
- Проверка формализованной и математической модели
- Проверка способов измерения и вычисления выходных характеристик.
- Проверка программной модели

Если по результатам проверки выделяется недопустимое рассогласование модели и системы, возникает необходимость в её корректировки или изменения.

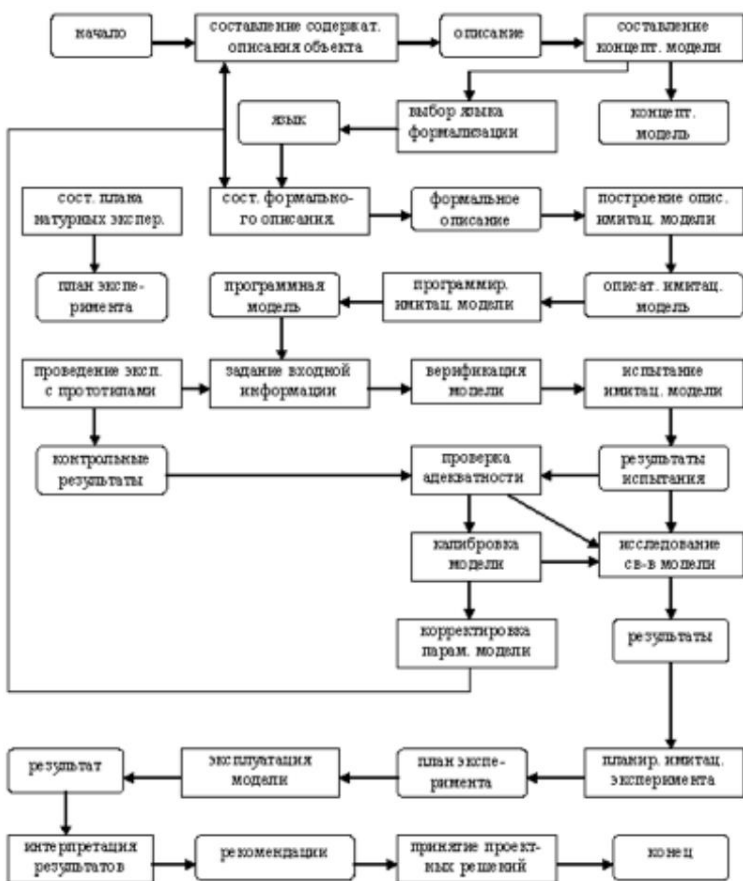
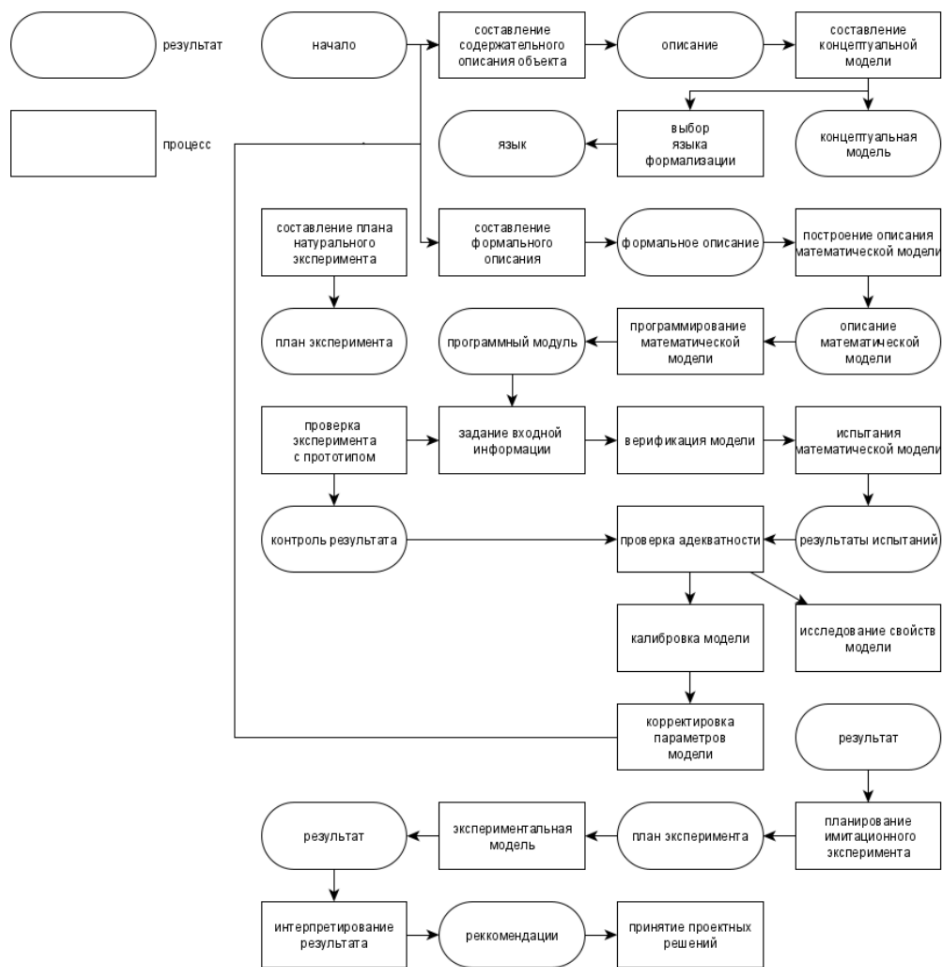
Выделяют следующие типы изменений:

1. Глобальные. Возникают в случае методических ошибок в концептуальной или математической модели (проще всё начать сначала, чем исправлять).
2. Локальные. Связаны с уточнением некоторых параметров и алгоритмов. Заменить компоненты модели на более точные.
3. Параметрические изменения некоторых специальных характеристик, называемых калибровочными. Как правило, эти характеристики мы задаем сами.

Схема взаимосвязи технологических этапов моделирования.

Всё что связано с процессом – это прямоугольник.

С результатом – овал.



TODO

3.1

Есть в большом файле

1. планирование компьютерного эксперимента (найти материал по)

- машинный эксперимент
- активный/пассивный эксперимент
- стратегическое планирование и тактика

Различают стратегическое и практическое планирование:

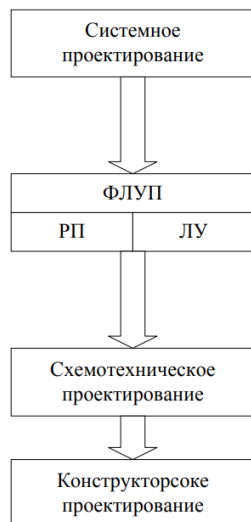
- При стратегическом планировании ставится задача построения оптимального плана эксперимента для достижения данной цели поставленной перед моделированием (оптимизация структуры алгоритмов и параметров системы).
- Тактическое планирование преследует частные цели оптимальной реализации каждого конкретного эксперимента из множества необходимых заданных при стратегическом планировании.

9. Проверка адекватности и корректности систем (см 8.3 про калибровку)

10. Моделирование на системном уровне. Схема вычислительной установки

Вычислительная система, как объект моделирования.

В теории проектирования ВТ принято выделять уровни проектирования (на каждом из них свои типовые математические модели). Если рассматривать процесс проектирования электронной техники как иерархический процесс



1. Системное проектирование. В качестве объектов нужно рассматривать процессор, память, каналы и т.д., а также ОС.
2. Функционально-логический уровень проектирование (ФЛУП). Входными характеристиками являются выходные параметры системного проектирования. ФЛУП делится на два уровня:
 - а. подуровень регистровых передач (АЛУ: 2 входных регистра, сумматор, выходной регистр, оперируем моделированием машинных команд)
 - б. логический уровень (прописать логику и проверить, получаем ли мы логику, которую ожидали. Минимизация: совершенные конъюнктивные и дизъюнктивные формы)
3. Схемотехнический уровень проектирования (УП). Проектирование интегральных схем, синтез логических элементов. Уровни 0 и 1 должны быть различимы, и в идеале прямоугольными (не выше транзисторов).
4. Конструкторский УП. В качестве математического формализма – теория графов. Здесь рассматриваются вопросы о теплообмене, охлаждении и т.д. Интегро-диф. Уравнения, описывающие ток жидкости.

Вопрос: можно ли начать проектирование этой схемы снизу-вверх? Ответ:

Моделирование на системном уровне.

При моделировании новых и модернизации существующих вычислительных систем и сетей необходимо предварительно оценивать эффективность их функционирования с учетом различных вариантов структурной организации. Эти варианты могут отличаться составом и характеристиками устройств. Структурой межмодульных связей, режимами работы и алгоритмами управления. Для оценок таких структур используют модели вычислительных систем.

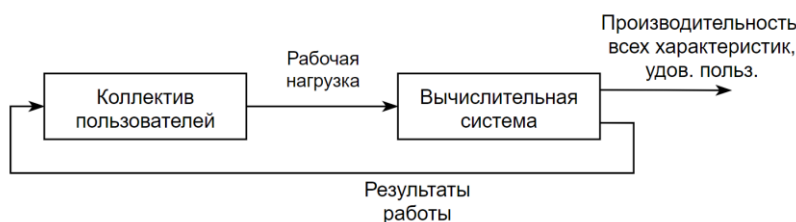
Под **вычислительной системой** будем понимать комплекс аппаратных (процессор, память, ...) и программных средств (ОС), которые в совокупности выполняют определенные рабочие функции.

ОС – набор ручных и автоматических процедур, который позволяет группе людей эффективно использовать вычислительную установку

Коллектив пользователей – это сообщество таких людей, которые используют систему для удовлетворения своих нужд по обработке информации.

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

Схема вычислительной установки:



Индекс производительности (ИП) – описатель, который используется для представления производительности системы. Различают:

- Качественные ИП.
 - Тип процессора – RISC/CISC,
 - «легкость использования системы и тд
- Количественные ИП.
 - 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 системы в течение заданного интервала времени к длительности этого интервала.

Концептуальная модель включает в себя сведения о выходных и конструктивных параметрах системы, ее структуре, особенности работы каждого элемента, характере взаимодействия между ресурсами. Включается постановка прикладной задачи, определяющей цели моделирования исходной системы, а также исходные данные для исследования системы.

Формализованная схема представляет собой, как правило, некоторую сложную систему массового обслуживания.

Основные задачи, которые необходимо решить:

1. Определение принципов организации вычислительной системы;
2. Выбор архитектуры, уточнение функций и их разделение на подфункции (что реализуется аппаратным, а что программным способом);
3. Разработка структурной схемы, т.е. определение состава устройств и способов их взаимодействия;
4. Определение требований к выходным параметрам устройств и формирование технического задания для разработки отдельных устройств.

11. Непрерывно-стохастические модели. Q-схемы

Особенность непрерывно стохастической модели будем рассматривать на примере систем массового обслуживания (СМО) в качестве типовых математических моделей. При этом используемая система формализуется как некая система обслуживания.

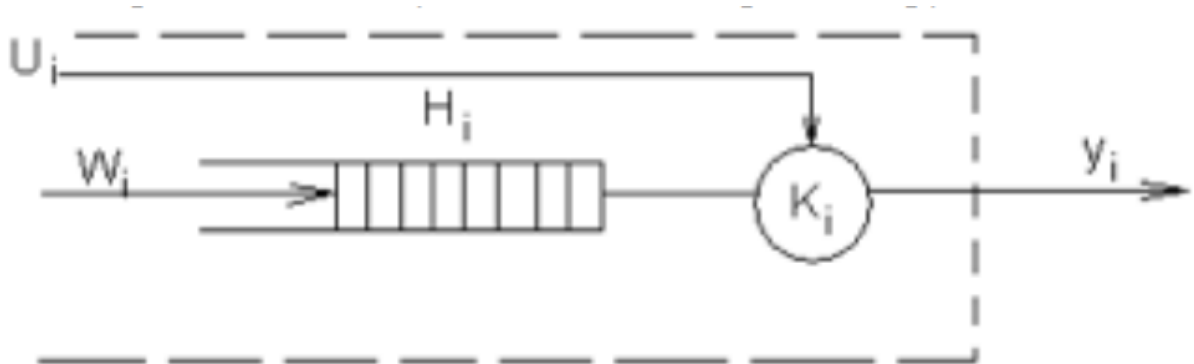
Характерным для таких объектов является случайное появление требований (заявок) на обслуживание и завершение обслуживания в случайные моменты времени. Т.е. характер функционирования системы носит стохастический порядок.

В любом элементарном акте обслуживания можно выделить две основные составляющие:

- 1) Ожидание обслуживания
- 2) Собственно, обслуживание

i -й прибор обслуживания:

- W_i, u_i – заявки
- H_i – накопитель
- K_i – канал (обслуживающий аппарат, ОА)
- Y_i – поток обслуженных заявок



Прибор обслуживания i состоит из:

- накопителя заявок емкостью L_i
- канала обслуживания заявок

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) и характеризуется только вектором времени

Поток называется **неоднородным**, если он задается не только временными моментами, но и признаками этих событий (наличие приоритета, принадлежность к тому или иному типу заявки).

Если интервалы времени между сообщениями независимы между собой и являются случайными величинами, то такой поток называется **поток с ограниченным последствием**.

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

Поток называется **стационарным**, если вероятность появления того или иного числа событий на некотором интервале времени зависит лишь от длины интервала и не зависит от того, где на оси времени взят этот участок.

Для ординарного потока среднее число сообщений, поступивших за малый интервал Δt от времени t

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

Тогда среднее число сообщений в единицу времени (**интенсивность ординарного потока**):

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

Интенсивность стационарного потока не зависит от времени и представляет собой постоянное значение, равное среднему числу событий, наступающих в единицу времени.

В i -м приборе обслуживания имеем 2 потока:

- Поток заявок w_i - интервалы времени между моментами появления заявок на входе канала k_i (*это подмножество неуправляемых переменных*)
- Поток обслуживания u_i - интервалы времени между началом и окончанием обслуживанием заявок в канале (*принадлежат подмножеству управляемых*).

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

Процесс функционирования i -ого прибора можно представить, как процесс изменения его состояний во времени. Переход в новое состояние для i -ого прибора означает изменение количества заявок, которые в нем находятся (в накопителе или канале) \Rightarrow вектор состояния i -го прибора $Z^i = (ZHi, Zki)$ (n -накопитель, k -канал, все наверху)

Состояния накопителя:

- 0 заявок (пуст)
- 1 заявок
- L_i (полностью занят)

Состояния канала:

- 0 – свободен
- 1 - занят.

В практике моделирования элементарные Q-схемы обычно объединяют.

- если каналы различных приборов обслуживания соединены параллельно, то имеет место **многоканальное обслуживание**.
- если последовательно – **многофазное обслуживание**.

Таким образом для задания Q-схемы необходимо использовать **оператор сопряжения R**, отражающий взаимосвязь элементов структуры.

Различают Q-схемы:

- Разомкнутые – 1 вход и 1 выход как минимум (*выходной поток заявок не может поступить к какому-либо элементу, т.е. отсутствует обратная связь*)
- Замкнутые – ни одного входа/выхода (*есть обратная связь*)
- Смешанного типа – есть и замкнутые, и разомкнутые

Собственные внутренние параметры Q-схемы:

- количество фаз
- количество каналов в каждой фазе
- количество накопителей в каждой фазе
- емкость i-го накопителя.

В зависимости от ёмкости накопителя в теории массового обслуживания применяют следующую терминологию:

- **Система с потерями** - емкость равна нулю (т.е. накопитель отсутствует, а есть только канал)
- **Система с ожиданием** - емкость стремится к бесконечности (т.е. очередь заявок неограниченна).
- **Система смешанного типа** – емкость ограничена

Для задания Q-схемы так же необходимо описать **алгоритм её функционирования**, который определяет набор правил поведения заявок в системе в различных ситуациях.

Неоднородность заявок описывается с помощью систем классов и приоритетов.

Функционирование Q-схемы как отображение СМО может быть описано кортежем:

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

- W - подмножество входных потоков;
- U - подмножество потока обслуживания;
- R - оператор сопряжения элементов структуры;
- H - подмножество собственных параметров;
- Z - множество состояний системы;
- A - оператор алгоритмов поведения и обслуживания заявок;

TODO самостоятельно изучить блочно-иерархический подход к моделированию Q-схем

Для получения соотношений, связывающих характеристики, которые определяют функционирование Q-схемы, вводят некоторые допущения относительно входных потоков, функций распределения, длительности обслуживания запросов, дисциплин обслуживания (fifo и др)

12. Понятие Марковского процесса. Уравнение Колмогорова

Для математического описания функционирования устройств, развивающихся в форме случайного процесса, могут быть применены математические модели для описания так называемых Марковских случайных процессов.

Случайный процесс называется Марковским, если он обладает следующим свойством – для каждого момента времени вероятность любого состояния системы в будущем зависит только от состояния системы в настоящем и не зависит от того, когда и каким образом система пришла в это состояние (то есть не зависит от прошлого).

Реально таких систем, конечно, не существует. Но существуют механизмы, которые позволяют свести к этим процессам.

Для Марковских процессов обычно составляют уравнения Колмогорова. В общем виде: $F = (P'(t), P(t), \lambda) = 0$, где λ - вектор, определяющий некоторый набор коэффициентов, присущих системе

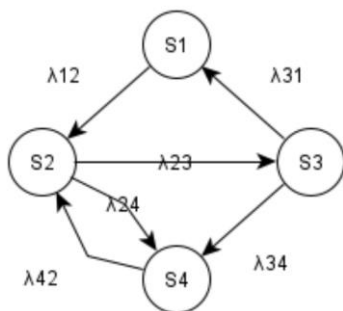
Для стационарного потока $\Phi = (P(t), \lambda) = 0$ - возможность получить соотношение $P = P(\lambda)$, и затем связать выходной поток через набор коэффициентов, соответствующих системе: $Y = Y(P(\lambda))$

Последнее соотношение представляет собой зависимость выходных параметров от некоторых внутренних параметров модели, и называется **базисной моделью**.

Для сложной системы $Y = Y(X, V, H) \Rightarrow$ нужно осуществить связь между внутренними параметрами модели λ и конструктивными параметрами X , неуправляемыми параметрами V и учитывать внутренние параметры H , то есть необходимо найти зависимость $\lambda = \lambda(X, V, H)$, которую называют **интерфейсной моделью**.

Следовательно, математическая модель системы строится как совокупность базисной и интерфейсной моделей, что позволяет использовать одни и те же базисные модели, для различных задач проектирования, осуществляя настройку на соответствующую задачу посредством изменения только интерфейсной модели. Для Q-схем математическая модель должна обеспечивать вычисление времени реакции на запрос и производительности системы.

Пример: пусть есть некоторая система S , имеющая 4 различных состояния



λ_{ij} - плотности вероятностей для множества состояний. Надо найти все вероятности P_1 - P_4 , т.е. вероятности того, что в момент времени t система будет в состояниях S_1 - S_4 .

Найдем $P_1(t)$ - вероятность того что в момент t система будет находиться в состоянии S_1 . Придадим t малое приращение и найдем вероятность того, что в момент времени $t + \Delta t$ система будет находиться в состоянии S_1 .

Это может быть реализовано двумя способами:

1. В момент t система S уже была в состоянии S_1 и за время Δt не вышла из него.
2. В момент t система была в состоянии S_3 и за время Δt перешла из него в состояние S_1 .

Вероятность первого способа найдем как произведение вероятности $P_1(t)$ на условную вероятность того, что будучи в состоянии S_1 система за время Δt не перейдет из него в состояние S_2 (с точностью до $o(\Delta t)$)

$$P_1(t)(1 - \lambda_{1,2}\Delta t)$$

Аналогично вероятность второго способа

$$p_3(t)\lambda_{3,1}\Delta t$$

Откуда

$$p_1(t)(1 - \lambda_{12}\Delta t) + p_3(t)\lambda_{3,1}\Delta t = p_1(t + \Delta t) - \text{вероятность нахождения системы в состоянии } S_1$$

$$\lim_{\Delta t \rightarrow 0} \frac{p_1(t + \Delta t) - p_1(t)}{\Delta t} = -\lambda_{12}p_1(t) + \lambda_{31}p_3(t)$$

$$p'_1(t) = -\lambda_{12}p_1(t) + \lambda_{31}p_3(t)$$

Мы вывели уравнение Колмагорова для первого состояния. Выведем далее для 2, 3 и 4 состояний.

$$p_2(t + \Delta t) = p_1(t)\lambda_{1,2}\Delta t + p_4(t)\lambda_{4,1}\Delta t + p_2(t)(1 - \lambda_{2,3}\Delta t) + p_2(t)(1 - \lambda_{2,4}\Delta t)$$

$$p'_2(t) = p_1(t)\lambda_{1,2} + p_4(t)\lambda_{4,1} - p_2(t)(\lambda_{2,3} + \lambda_{2,4})$$

$$p_3(t + \Delta t) = p_2(t)\lambda_{2,3}\Delta t + p_3(t)(1 - \lambda_{3,1}\Delta t) + p_3(t)(1 - \lambda_{3,4}\Delta t)$$

$$p'_3(t) = p_2(t)\lambda_{2,3} - p_3(t)(\lambda_{3,1} + \lambda_{3,4})$$

$$p'_4(t) = p_3(t)\lambda_{3,4} + p_2(t)\lambda_{2,4} - p_4(t)\lambda_{4,2}$$

Интегрирование данной системы дает искомые вероятности системы как функции времени. Начальные условия берутся в зависимости от того какого было начальное состояние системы. Кроме того, необходимо добавлять условие нормировки (сумма $P_i = 1$).

Уравнение Колмогорова строится по следующему правилам:

1. в левой части каждого уравнения стоит производная вероятности состояния, а правая часть содержит столько членов, сколько стрелок связано с данным состоянием. Если стрелка направлена из состояния, то соответствующий член имеет знак "-", в состояние – "+".
2. Каждый член равен произведению плотности вероятности перехода (интенсивности), соответствующий данной стрелке, умноженной на вероятность того состояния, из которого исходит стрелка.

13. Марковские случайные процессы (см 12). Многоканальные СМО с отказом

Многоканальные СМО с отказами

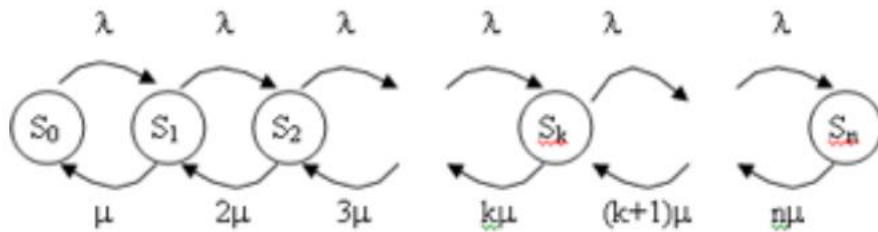
Будем нумеровать состояние системы по числу занятых каналов. Т.е. по числу заявок в системе.

Обозовем состояния:

- S_0 - все каналы свободны
- S_1 - занят один канал, остальные свободны
- ...
- S_k - занято k каналов, остальные свободны

- ...
- S_n – заняты все n каналов

Граф состояний:



Разметим граф, т.е. расставим интенсивности соответствующих событий. По стрелкам с лева на право система переводит один и тот же поток с интенсивностью λ .

Определим интенсивность потоков событий, переводящих систему справа на лево. Пусть система находится в S_1 . Тогда, когда закончится обслуживание заявки? занимающей этот канал, система перейдет в $S_0 \Rightarrow$ поток, переводящий систему в другое состояние, будет иметь интенсивность перехода μ . Если занято 2 канала, а не один, то интенсивность перехода составит 2μ .

Уравнения Колмогорова:

$$\begin{cases} p_0' = -p_0\lambda + p_1\mu \\ p_1' = -p_1\lambda - p_1\mu + p_0\lambda + p_2 2\mu \\ p_2' = -p_2\lambda - p_2 2\mu + p_1\lambda + p_3 3\mu \\ p_k' = -p_k\lambda - p_k k\mu + p_{k-1}\lambda + p_{k+1}(k+1)\mu \\ p_n' = p_{n-1}\lambda - p_n n\mu \end{cases}$$

Введем специальные характеристики, которые позволяют получить характеристики СМО **через интенсивности потоков**.

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

$$p_0 = \frac{1}{1 + \frac{\lambda/\mu}{1!} + \frac{(\lambda/\mu)^2}{2!} + \dots + \frac{(\lambda/\mu)^n}{n!}}$$

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

Среднее число заявок, приходящих в систему за среднее время обслуживания одной заявки:

$$\lambda/\mu = \rho$$

$$p_0 = \left[1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \dots + \frac{\rho^n}{n!} \right]^{-1}$$

$$p_k = \frac{\rho^k}{k!} p_0$$

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

- вероятность отказа – вероятность того, что все n каналов заняты

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

- 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 соотношения могут рассматриваться как базисная модель оценки характеристик производительности системы. Входящий в эту модель параметр $\lambda = 1/\text{время_обработки}$ – усредненная характеристика пользователей, а μ – функция технических характеристик компьютера и решаемых задач. Связь между ними должна быть учтена с помощью соотношений, называемых **интерфейсной моделью**.

В простейшем случае, если время ввода/вывода информации по каждой задаче мало по сравнению со временем решения задачи, то логично принять, что (среднее) время решения равно $1/\mu$ и равно отношению (среднего числа операций, выполненных процессором при решении одной задачи) к (среднему быстродействию процессора [операций в секунду])

Введен ряд существенных допущений => могут быть неадекватные результаты.

0. (мое). Не Марковские случайные процессы, сводящиеся к Марковским.

Реальные процессы весьма часто обладают последствием и поэтому не являются Марковским. Иногда (весьма редко) при исследовании таких процессов удастся воспользоваться методами, разработанными для Марковских цепей. Наиболее распространенными являются:

1. Метод разложения случайного процесса на фазы (метод псевдо состояний)
2. Метод вложенных цепей Маркова

Метод псевдо состояний.

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

Условие статистической эквивалентности реального и фиктивного состояния могут в каждом конкретном случае выбираться по-разному. Например,

$$\min_{t_1} \int_{t_1}^{t_2} (\lambda_{iэкв}(\tau) - \lambda_i(\tau)) d\tau,$$

где $\lambda_{iэкв}$ - 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 близка к реальной системе, и ее возможно подвергнуть обычному исследованию с помощью аппарата теории 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 сведение потока Эрланга k -го порядка к Пуассоновскому осуществляется введением k псевдо состояний. Интенсивности переходов между псевдо состояниями равны соответствующему параметру потока Эрланга. Полученный таким образом эквивалентный случайный процесс является Марковским, т.к. 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. Время восстановления распределено по закону Эрланга 3-го порядка с функцией плотности

$$f_2(t) = 0.5\mu(\mu t)^2 e^{(-\mu t)}$$

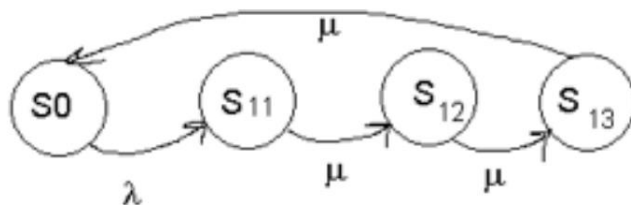
Найти предельные вероятности возможных состояний системы.

Решение.

Пусть система может принимать 2 возможных состояния: S_0 - устройство исправно; S_1 - устройство отказало и восстанавливается

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

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



(а было S0-лямбда-S1-мю-S0)

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

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

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

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

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

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

$$\text{Ответ: } p_0 = \frac{\mu}{\mu + 3\lambda}, \quad p_1 = \frac{3}{3 + \frac{\mu}{\lambda}}$$

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, когда наступает марковский процесс, являются случайными и зависят от свойств исходного процесса. Затем обычными методами теории Марковских цепей исследуются процессы только в эти характерные моменты.

Случайный процесс с конечным или счетным множеством состояний называется полумарковским, если заданы вероятности переходов системы из одного состояния в другое (p_{ij}) и распределение времени пребывания процессов в каждом состоянии (например, в виде функции распределения или функции плотности распределения)

Если ничего нет: Метод статистических испытаний. Метод Монте-Карло.

14. Метод Монте-Карло. Метод статистических испытаний

Если ничего нет: Метод статистических испытаний. Метод Монте-Карло.

В СМО поток заявок редко бывает Пуассоновским и еще реже наблюдается распределенный закон.

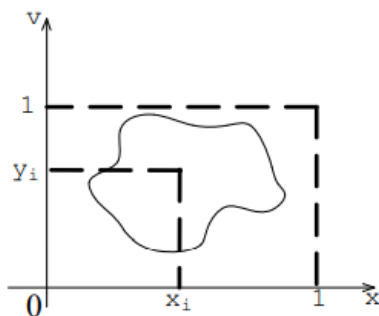
Для произвольных потоков событий, переводящих систему из состояния в состояние.

Аналитические решения получены только для отдельных частных случаев. Когда построение аналитической модели является по той или иной причине трудно осуществимым ставится метод статистических испытаний.

*Когда этот метод нашел реальное применение: с развитием компьютеров.

Идея метода: вместо того чтобы описывать случайные явления с помощью аналитической зависимости производится т.н. «розыгрыш», т.е. моделирование «случайного» явления с помощью некоторой процедуры, дающей «случайный» результат. Проведя такой розыгрыш достаточно большое количество раз, получаем статистический материал, т.е. множество реализаций случайного явления. Дальше эти результаты могут быть обработаны методами математической статистики.

Метод Монте-Карло был предложен в 1948 году Фон-Нейманом как метод численного решения некоторых математических задач.



Введем в некотором единичном квадрате случайную величину. Задача стоит в определении её площади.

Суть метода:

1. Вводим в некотором единичном квадрате любую поверхность S .
2. Любым способом получаем 2 числа x_i, y_i , подчиняющиеся равномерному закону распределения случайной величины на интервале $[0, 1]$.
3. Полагаем, что одно число определяет координату x , второе – координату y .
4. Анализируем принадлежность точки (x, y) фигуре. Если принадлежит, то увеличиваем значение счетчика на 1.
5. Повторяем n раз процедуру генерации 2х случайных чисел с заданным законом распределения и проверку принадлежности точки поверхности S .
6. Определяем площадь фигуры как количество попавших точек, к количеству сгенерированных.

Фон-Нейман доказал, что погрешность

$$\varepsilon \leq \sqrt{\frac{1}{n}}$$

Преимущество метода статистических испытаний в его универсальности, которая обуславливает его возможность статистического исследования объекта, причем всестороннего. Но для реализации этого исследования необходимы довольно полные статистические сведения о параметрах элементов, входящих в систему. Еще один недостаток - большой объем требующихся вычислений, равный количеству обращений к модели n .

Поэтому вопрос выбора величины n имеет важнейшее значение. Уменьшая n , повышаем экономичность расчетов, но одновременно ухудшаем их точность.

(15, 25). 15. Способы получения последовательности псевдослучайных чисел. 25. Алгоритмический способ получения случайных чисел

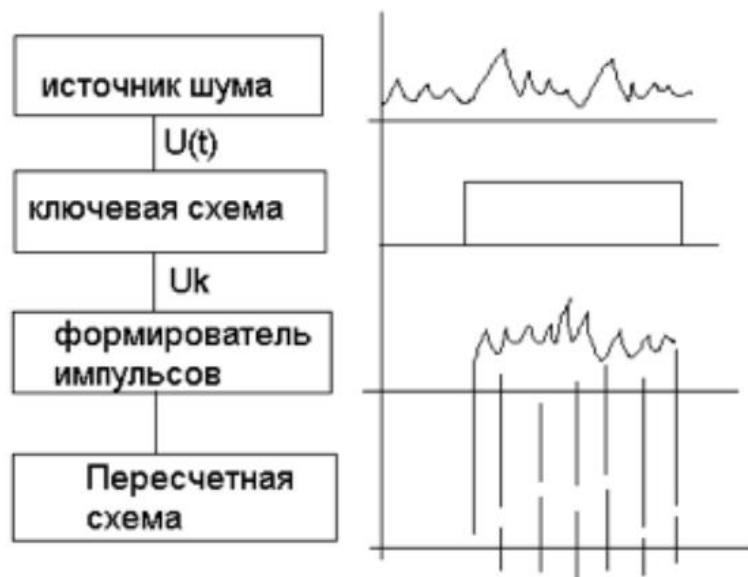
При имитационном моделировании системы одним из основных вопросов является учет стохастических воздействий. Для этого метода характерно большое число операций со случайными числами или величинами и зависимость результатов от качества последовательностей случайных чисел. На практике используется 3 основных способа:

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

Способ	+	-
Аппаратный	<ol style="list-style-type: none"> 1. запас чисел неограничен 2. мало вычислительных операций 3. не занимает место в памяти 	<ol style="list-style-type: none"> 1. требуется периодическая проверка на случайность 2. нельзя воспроизвести 3. использование специальных устройств: надо стабилизировать
Табличный	<ol style="list-style-type: none"> 1. однократная проверка на случайность 2. воспроизводимость 	<ol style="list-style-type: none"> 1. запас чисел ограничен 2. занимает место в памяти. время на обращение к памяти
Алгоритмический	<ol style="list-style-type: none"> 1. однократная проверка 2. воспроизводимость 3. относительно мало памяти 4. не используются внешние устройства 	<ol style="list-style-type: none"> 1. запас чисел ограничен периодом 2. ресурсы системы

Аппаратный способ.

Случайные числа вырабатываются случайной электронной приставкой (генератор случайных чисел), служащей, как правило, в качестве одного из внешних устройств. Реализация этого способа обычно не требует дополнительных вычислений, а необходима только операция - обращения к ВУ. В качестве физического эффекта, лежащего в основе таких генераторов чаще всего используют шумы в электронных приборах.



(по оси $x - t$, 2 – импульс, 1-3 – вырезать на графике то, что необходимо, 4 – $t_i - 1, t_i$)

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. В настоящее время с помощью рекуррентных математических уравнений реализовано несколько алгоритмов генерирования псевдослучайных чисел.

Псевдослучайными эти числа называются потому, что фактически они, даже пройдя все статистические испытания на случайность и равномерность распределения, остаются полностью **детерминированными**. Т.е. если каждый цикл работы генератора начинается с одними и теми же исходными данными (константами, начальными условиями и значениями), то на выходе мы получаем одни и те же последовательности.

0.

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

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

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 Вейль. 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 псевдослучайных чисел, так как теорема Якоби относится только к действительным числам, которые не могут быть использованы в вычислениях, так как иррациональные действительные числа требуют для своей записи бесконечного числа знаков.

1. 1946 год, Фон Нейман.

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

2. Линейный конгруэнтный метод, Лемер, 1949.

$$X_{n+1} = (aX_n + c) \bmod m,$$

Для подбора коэффициентов a , c , m потрачены десятки лет. Подбор почти иррациональных a ничего не дает. Если брать дробную часть от корня предыдущего значения (степень и мантисса), то полученные последовательности оканчиваются циклом с коротким периодом (1225-147). Иррациональные нельзя, дробные нельзя, следовательно, нужно брать целые.

Установили, что при $c = 0$ и $m = 2^n$ наибольший период достигается при нечетном начальном числе и при $a = 3 + 8i$ или $a = 5 + 8i$.

3. Форсайд, 1977

Доказал, что в этой последовательности тройки чисел лежат на 15 параллельных плоскостях. Ужас

От отчаяния используют 2 и даже 3 разных генератора, смешивая их значения. Если генераторы независимы, то сумма их последовательностей обладает дисперсией, равной сумме дисперсий. (то есть случайность рядов возрастет при суммировании).

4.

Сейчас в системах программирования обычно используют конгруэнтные генераторы по алгоритму, предложенному национальному бюро стандартов США с периодом 2^{24} .

5.

Следующий класс генераторов случайных чисел предложен А.Зейманом и Дж.Марсалиа. Генераторы этого типа основаны на использовании последовательности Фибоначчи. Пример такой последовательности {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,} (каждый последующий член равен сумме предыдущих). Если мы берем только последнюю цифру каждого числа в последовательности, то получим последовательность чисел {0, 1, 1, 2, 3, 5, 8, 3, 1, 4, 5, 9, 4 ...}. Если эта последовательность применяется для начального заполнения массива большой длины, то используя этот массив, можно создать генератор случайных чисел Фибоначчи с запаздыванием, где складываются не соседние, а удаленные числа.

Или

С начала $CF = 1$.

$$a_i = a_{i-1} - a_{i-2} + CF;$$

$$CF = (a_i \geq 10);$$

Пример:

```
randomize(231)
```

```
x = rnd();
```



```
randomize(231)
```

```
y = rnd();
```

```
// x != y
```

```
x = rnd(-231)
```

```
y = rnd(-231)
```

```
// x = y
```

6.

Для имитации равномерного распределения в интервале от [a, b] используется обратное преобразование функции плотности вероятности:

$$\frac{x-a}{b-a} = R$$
$$x = a + (b-a)R$$

где R – равномерно распределенное псевдослучайное число на [0, 1].

7.

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.

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

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,

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & \text{при } x \geq 0, \\ 0, & \text{при } x < 0, \end{cases}$$

исходя из (1), может быть вычислено следующим образом:

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

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

8. Распределение Пуассона.

$$p_k(\lambda) = \frac{\lambda^k}{k!} e^{-\lambda}$$

Распределение Пуассона относится к числу дискретных, т.е. таких при которых переменная может принимать лишь целочисленные значения, включая $MX=DX=\lambda > 0$.

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

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

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

9. Распределение Эрланга.

$$f(\tau) = \frac{\lambda(\lambda\tau)^{k-1}}{(k-1)!} e^{-\lambda\tau}$$

λ – среднее число заявок в единицу времени

k – порядок потока Эрланга

τ – длительность промежутка времени

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.

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

10. 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 MX и среднеквадратичным отклонением σX , может быть получена по следующей формуле:

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 и дает довольно точные результаты.

12.

Процедура генерирования псевдослучайных чисел (равномерный и нормальный законы распределения):

```
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 := ( sqrt(S) - 3.0 ) * S * 0.01 + S;
end;

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

При $n = 0$ происходит параметрическая настройка или т.н. «установка».

При $n = 1$ будем получать равномерно распределенную случайную величину.

При $n = 2$ будет гауссово (нормальное) распределение.

Программа генерации случайных чисел на Фортране для машин ES (~IBM 360)

```
SUBROUTINE RANDOM( IX, IY, RN)      // была придумана для 32 разрядной
машины
  IY = IX * 1220703125
  IF (IY) 3,4,4 // if ( IY < 0) then
3  IY = IY + 2147483647 + 1
4  RN = IY
  RN = RN * 0.4656613E-9
  IX = IY
  RETURN
END
```

```
// обращение к данной процедуре
CALL RANDOM(IX, IY, YFL)
```

IX – число, которое при первом обращении должно содержать нечетное целое число, состоящее менее чем из 9 цифр

IY - полученное случайное число, используемое при последующих обращениях к программе

YFL - полученное равномерно распределенное в интервале [0, 1] случайное число

16. Методика построения программной модели системы

Для разработки имитационной модели исходная вычислительная система (ВС) должна быть представлена как стохастическая СМО. Информация от внешней среды поступает в случайные моменты времени, длительность обработки различных типов информации может быть в общем случае различна. Т.о. внешняя среда является генератором сообщений. А комплекс вычислительных устройств – обслуживающими устройствами.

Концептуальная модель ВС.

(программная, но без «программа» и верхнего и нижнего блоков)

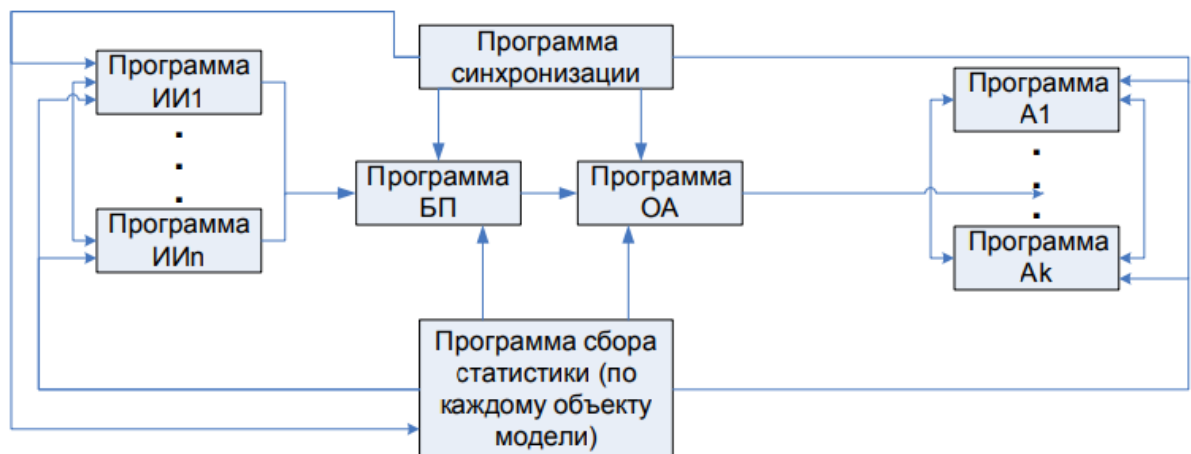
ИИ – источники информации – выдают на вход буферной памяти (БП) независимо друг от друга сообщения. Закон появления сообщений – произвольный, но задан наперед.

В БП сообщения записываются «в навал» 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. Быстродействие ОА ограничено, поэтому сообщения скапливаются в очередь

А – абоненты.

Программная модель



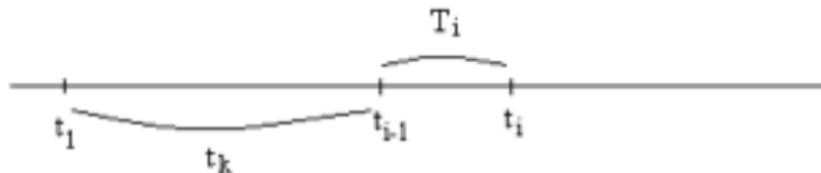
Если описать каждый блок с помощью программного имитатора и задать соответствующие связи между ними, то получаем имитационную модель системы

Главное – придумать программу управления (ПУ), которая определяет, когда обратиться к какому из устройств.

Рассмотрим каждый блок.

17. 1. Моделирование потоков сообщений

Поток сообщений обычно имитируется **моментами** появления очередного сообщения в потоке.



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

где T_i – интервал времени между появлением i -го и $(i-1)$ -го сообщения.

Выражения для вычисления времени с различным распределением

Нормальное	$t = \sigma_t \sqrt{\frac{12}{n} (\sum_{i=1}^n R_i - \frac{n}{2})} + m_x$
Равномерное	$t = a + (b - a)R$
Экспоненциальное	$t = -\frac{1}{\lambda} \ln(1 - R)$
Эрланга	$t = -\frac{1}{k\lambda} \sum_{i=1}^k \ln(1 - R_i)$

N примерно 12 для нормального закона

Программа – имитатор выработки таких интервалов:

1. Обратиться к генератору равномерно распределенных случайных величин на $[a, b]$
2. T_i – по заданному закону
3. К текущему времени прибавить T_i

```
// процедура равномерного распределения псевдослучайных чисел на интервале
[a,b]
// U - равном. распр. на [0, 1]
// x = a + (b - a)U
double get_time (int i)
{
    double S = 0;
    srand(seek);
    if ( i > 1 ) S += get_time(i - 1);
    S += a + (b - a)get_u();
    return S;
}
```

ИЛИ

```
double get_time (int i)
{
    double S = 0;
    srand(seek);
    for (int i = 0; i < .. ; i++)
        S += a + (b - a)rand();
    return S;
}
```

18. 2. Моделирование работы обслуживающего аппарата

Программа-имитатор работы ОА – набор программ, вырабатывающий случайные отрезки времени, соответствующие длительностям обслуживания требований.

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

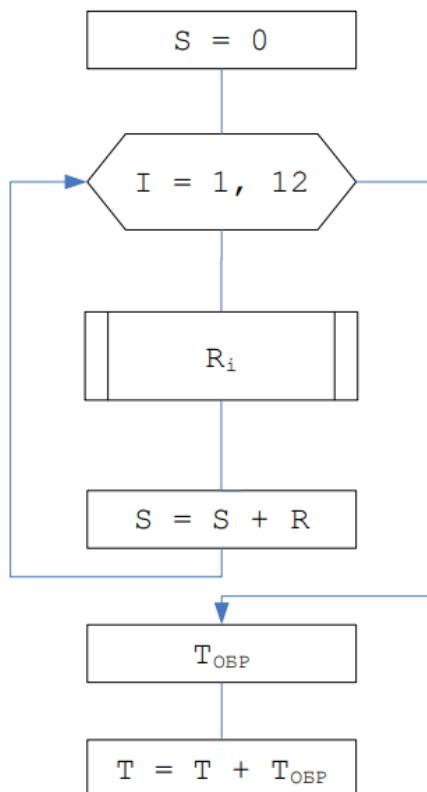
$$T_{обр} = M_x + \left(\sum_{i=1}^{12} R_i - 6 \right) \cdot \sigma_x$$

(так как

Нормальное	$t = \sigma_t \sqrt{\frac{12}{n}} \left(\sum_{i=1}^n R_i - \frac{n}{2} \right) + m_x$
------------	--

)

Начало и конец, нормальный цикл, R_i – случайное число с равномерным законом распределения
ТОБР – время обработки очередного сообщения (по формуле $ТОБР = MX + (S - 6) DX$), T – время освобождения ОА.



19. 3. Моделирование работы абонентов сети

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 на те или иные ресурсы вычислительной системы, вплоть до влияния на источник. Эти заявки могут моделироваться с помощью генератора сообщений, распределенными по заданному закону.

Таким образом, абонент либо имитируется как ОА, либо как генератор.

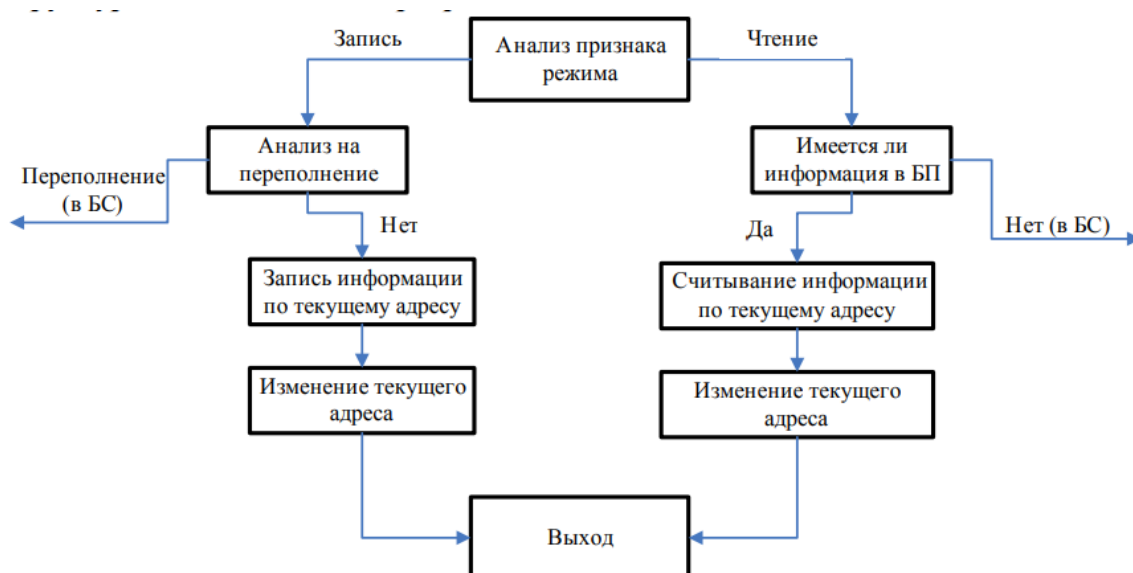
20. 4. Моделирование работы буферной памяти

Память - электромеханическое устройство, предназначенное для хранения, чтения и записи информации (включающее в себя: среду для запоминания, устройство управления, информация находится по адресу: база + смещение + [индекс]).

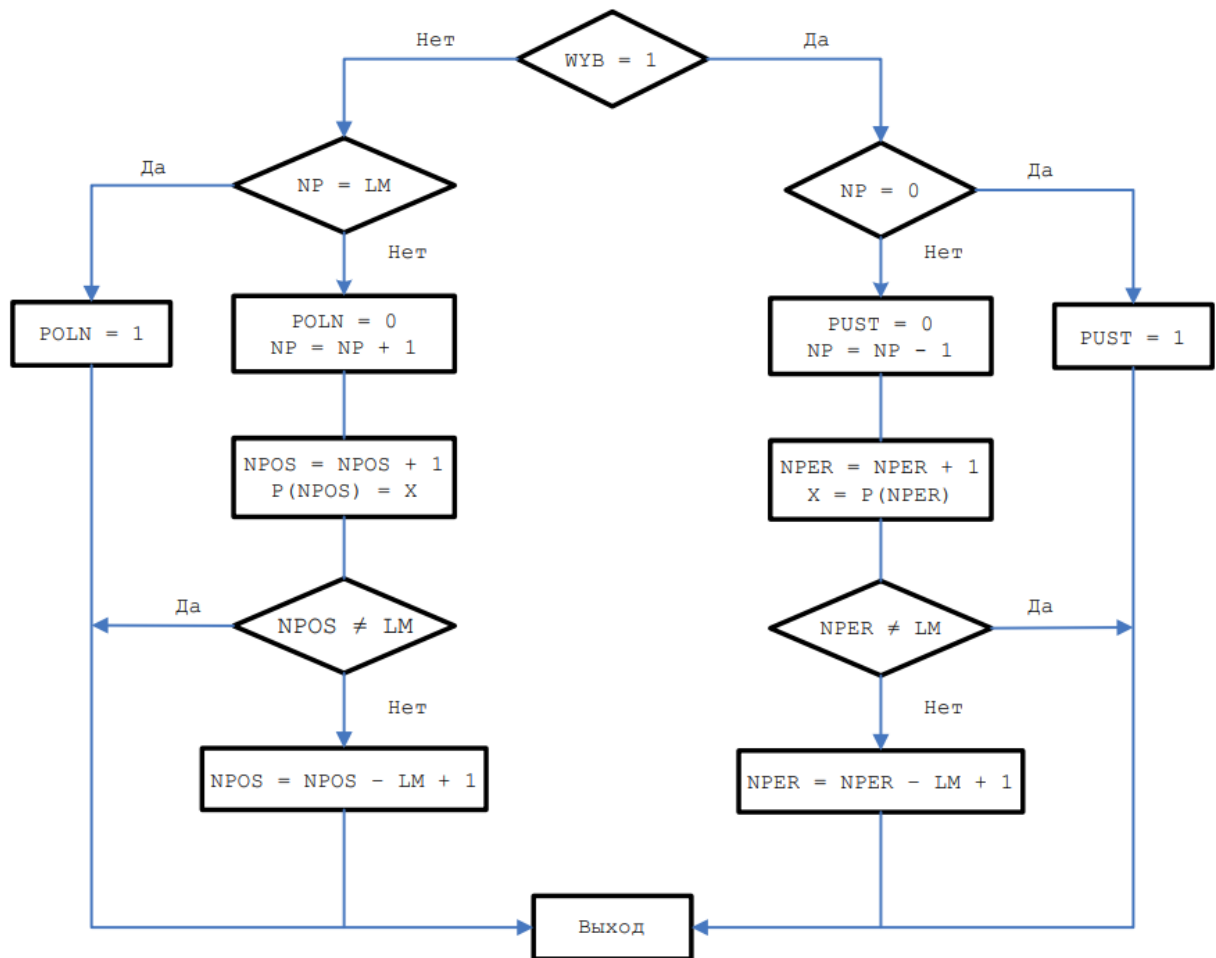
В блок статистики: ошибки записи, ошибки чтения.

Блок буферной памяти должен производить запись и считывание чисел, выдавать сигналы переполнения и отсутствия данных в любой момент времени располагать сведениями о количестве требований (заявок) в блоке. 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 требования.

Структурная схема модели буферной памяти



Алгоритм реализации работы буферной памяти



(зачем в конце эти формулы, там же просто 1)

9

P	массив обращения
WYB	признак обращения к буферной памяти (0 – режим записи, 1 – режим выбора сообщения)
NP	число сообщения в памяти
LM	объем буферной памяти
NPOS	номер последнего сообщения в памяти
NPER	номер первого сообщения в памяти
POLN	признак переполнения (1-нет свободных ячеек)
NPOS	=NPOS+1, если NPOS<LM NPOS-LM+1 иначе
X	ячейка для сообщения
PUST	признак отсутствия сообщения (1-в памяти нет сообщения)
NPER	=NPER-1, если NPER<1, =NPER-LM+1 иначе

21. 5. Разработка программы сбора статистики

Задача блока статистики заключается в накоплении численных значений необходимых для вычисления статистических оценок заданных параметров работы моделируемой системы.

- При моделировании простейшей 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. Среднее время = сумма всех времен / общее число сообщений
- Коэффициент загрузки обслуживающего аппарата (ОА) определяется как отношение времени работы ОА, к общему времени моделирования.
- вероятность потери сообщений в системе - отношение количества потерянных сообщений к сумме потерянных и обработанных сообщений в системе.

22. 6. Управляющая программа имитационной модели. 23.

Событийный принцип

Если программа-имитатор работы источника, буферной памяти, обслуживающего аппарата имитируют работу отдельных устройств, то управляющая программа имитирует алгоритм взаимодействия отдельных устройств системы.

Управляющая программа реализуется в основном по двум принципам:

1. Принцип Δt
2. Событийный принцип
3. Комбинированный

1. 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.

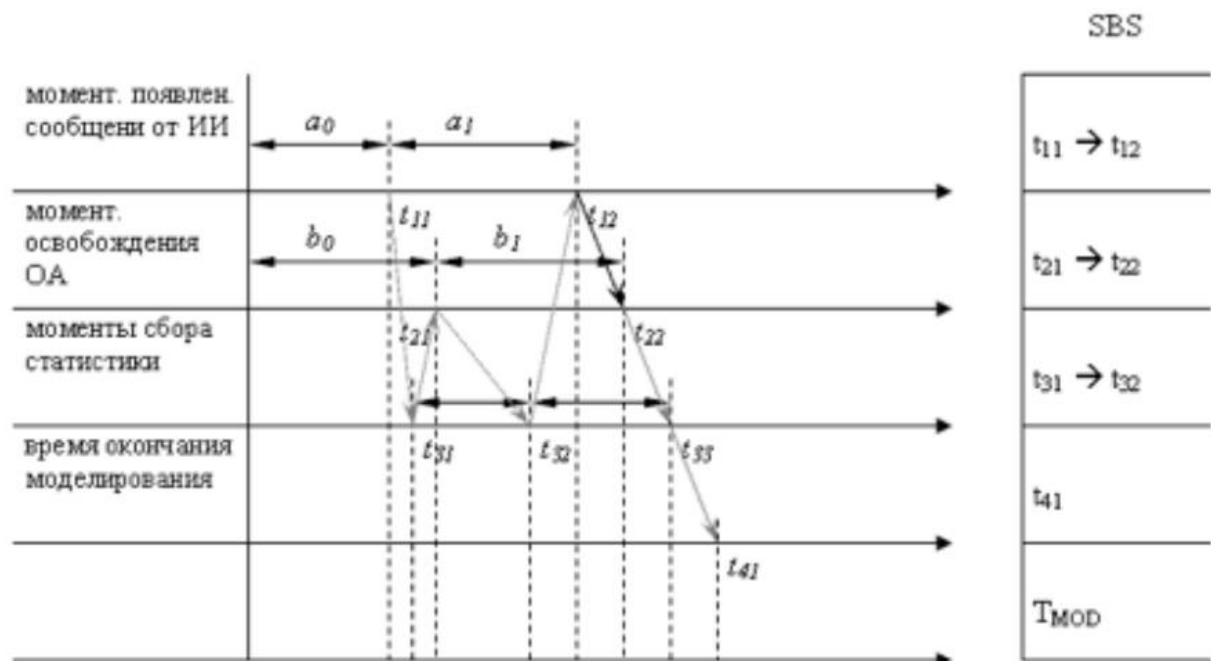
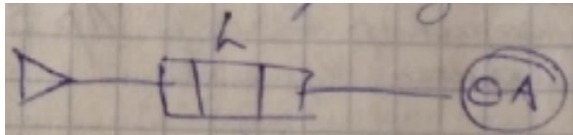
2. Событийный принцип.

Характерное свойство систем обработки информации - 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.

Поэтому моделирование и продвижение времени в системе удобно проводить, используя событийный принцип, при котором состояние всех блоков имитационной модели анализируется

лишь в момент появления какого-либо события. Момент поступления следующего события определяется минимальным значением из списка будущих событий (СБС), представляющего собой совокупность моментов ближайшего изменения состояния каждого из блоков системы.

Недостатки: много ресурсов (постоянно ищем минимальное, вычеркиваем и тд), неравномерная протяжка времени.

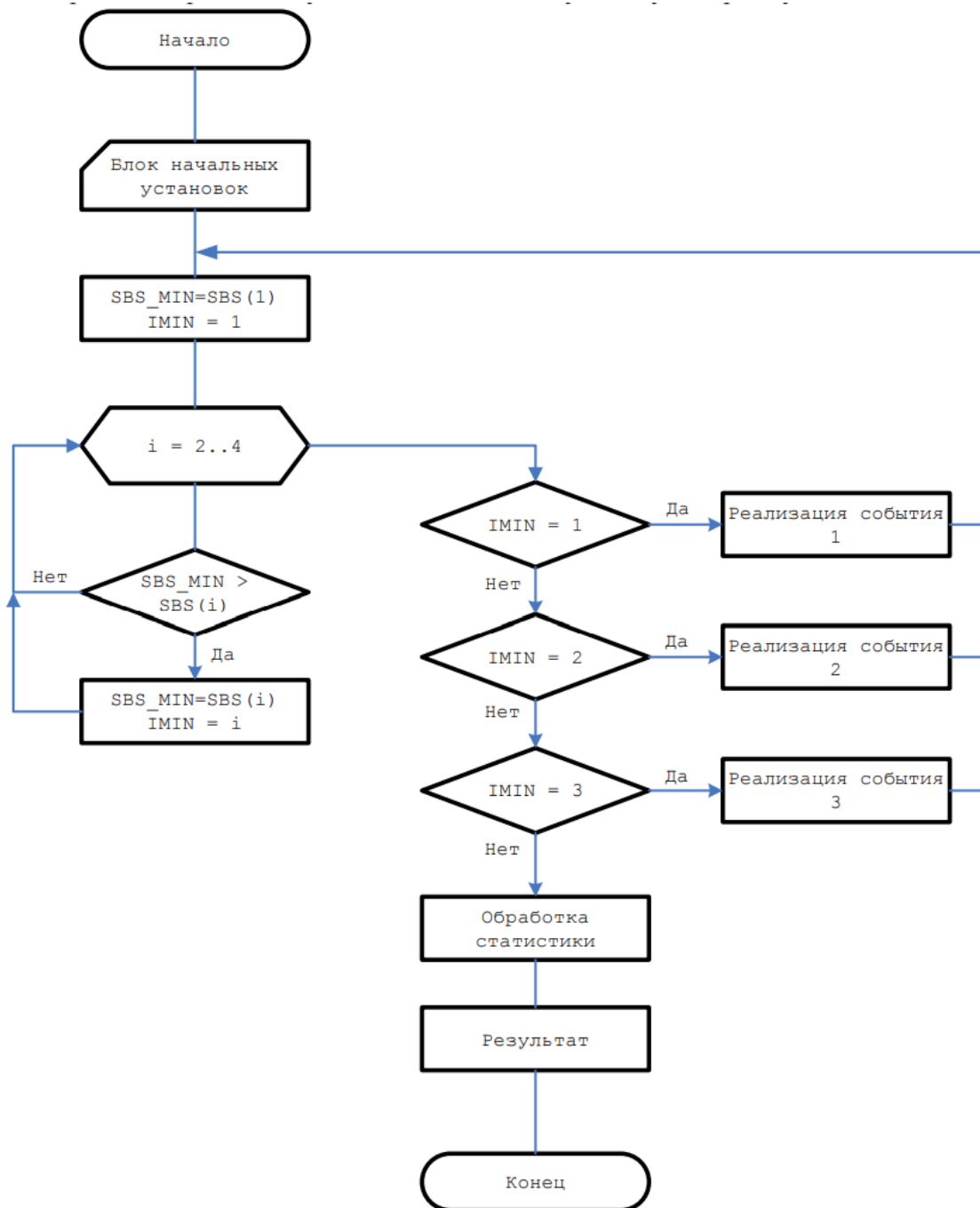


1. Первая ось: момент появления сообщений
 2. Вторая ось: момент освобождения обслуживающего аппарата
 3. Третья ось: момент сбора статистики (здесь абсолютно равные интервалы, мы сами определяем, когда собирать статистику)
 4. Четвертая ось: время окончания моделирования
 5. Пятая ось: текущее время
- 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.

1. Для всех активных блоков (блоки, порождающие события) заводят свой элемент в одномерном массиве – в списке будущих событий (СБС).
2. В качестве подготовительной операции в СБС заносят время ближайшего события от любого активного блока.
 - а. 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.

- b. Активизируя программу-имитатор, ОА вырабатывает псевдослучайную величину b_0 , определяющую момент времени t_{21} , которую также заносят в SBS.
 - c. момент времени t_{31} (момент первого сбора статистики) определяется равным стандартному шагу сбора статистики $t_{\text{СТАТ}}$ и заносится так же в СБС.
 - d. В этот же список заносим время окончания моделирования t_{41} .
3. протяжка времени осуществляется по следующему алгоритму (нормальный цикл)



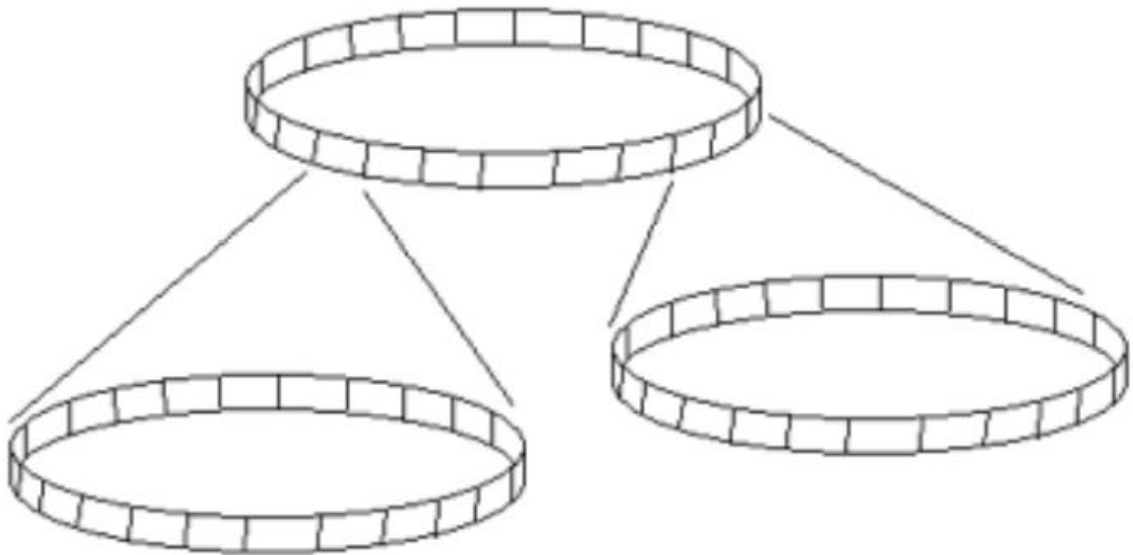
3. Комбинированный

Два приведенных метода являются универсальными алгоритмами протяжки модельного 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 названием **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.



Список уровня 1 содержит n_1 элементов и описывает планируемое событие в пиковых интервалах. Число n_1 представляет собой разбиение пикового интервала на более мелкие участки, с каждым из которых связан список событий происшедших за этот интервал. Списки второго уровня и выше являются масштабирующими списками, количество элементов которого равно константному значению n_2 , которое характеризует коэффициент масштабирования временных интервалов.

Собственно, алгоритм протяжки времени заключается в последовательном поиске непустых элементов в самом верхнем циркулярном списке с большим шагом и дальнейшим спуском на нижние уровни (иерархические), вследствие чего уменьшается шаг протяжки модельного времени.

24. Классификация языков имитационного моделирования по типу системы

Алгоритмические языки при моделировании систем служат инструментом реализации, анализа и контроля правильности функционирования модели. Поэтому выбор языка имеет большое значение

Каждый язык имеет свой синтаксис и семантику, то есть глубоко проработанный язык абстракций, позволяющий реализовать формальную модель системы.

Для программирования модели могут использоваться следующие языки:

1. Универсальные алгоритмические языки высокого уровня.
2. Специализированные языки моделирования: языки, реализующие событийный подход, подход сканирования активностей, языки, реализующие процессно-ориентированный подход.
3. Проблемно-ориентированные языки и системы моделирования.

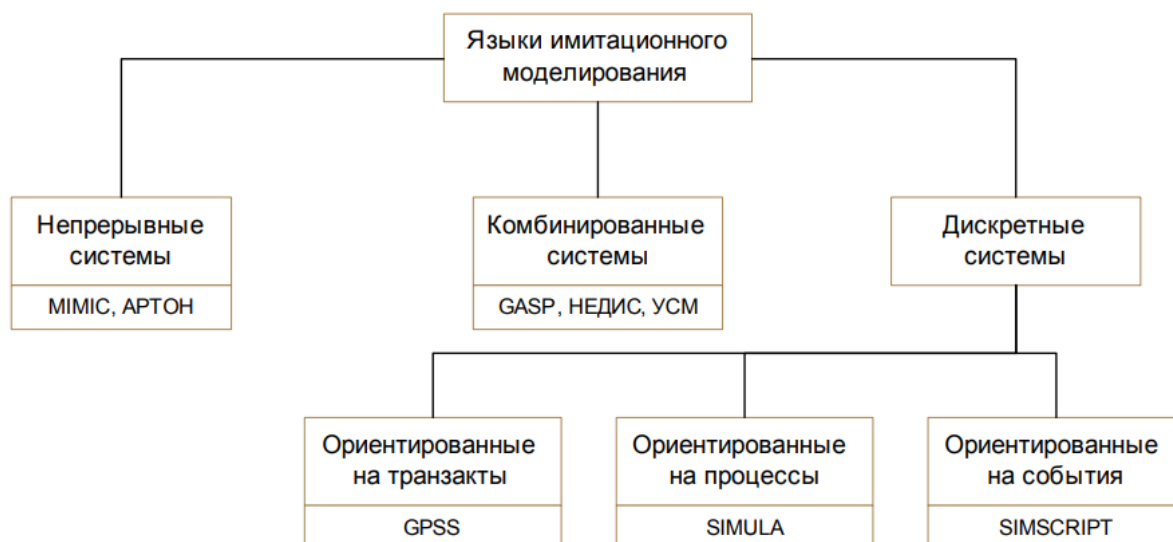
Качество языков моделирования характеризуется:

1. Удобством описания процесса функционирования;
2. Удобством ввода исходных данных, варьирования структуры, алгоритмов работы и параметров модели;
3. Эффективностью анализа и вывода результатов моделирования;
4. Простотой отладки и контроля работы моделирующей программы;
5. Доступностью восприятия и использования языка.

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

по принцип формирования системного времени.



1. Непрерывное

представление систем сводится к представлению дифференциальных уравнений, с помощью которых устанавливают связь между входной и выходной функциями. Если выходные переменные модели принимают дискретные значения, то уравнения являются разностными.

2. Комбинированные (GASP)

комбинированный, в основе лежит язык FORTRAN. 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 составлять процедуры, описывающие условия наступления событий. Законы изменения непрерывных переменных, правила перехода от одного состояния к другому, т.е. реализуется классический принцип дифференциальных уравнений.

3. Дискретные

Группы языков моделирования, ориентированные на **дискретное** время, используются при построении именно имитационных моделей, но при этом используются разные способы описания динамического поведения исследуемого объекта.

3.0. Формальное описание динамики моделируемого объекта.

Будем считать, что любая работа в системе совершается путем выполнения **активностей**. Т.е. активность является наименьшей единицей работы и её рассматривают как единый дискретный шаг. Следовательно, активность является единым динамическим объектом, указывающим на совершение единицы работ. Пример: активность установки головки жесткого диска, активность передачи информации с жесткого диска.

Процесс – это логически связанный набор активностей.

Активности проявляются в результате свершения событий. **События** – это мгновенное изменение состояния некоторого объекта системы.

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

Построение модели состоит из 2 основных задач:

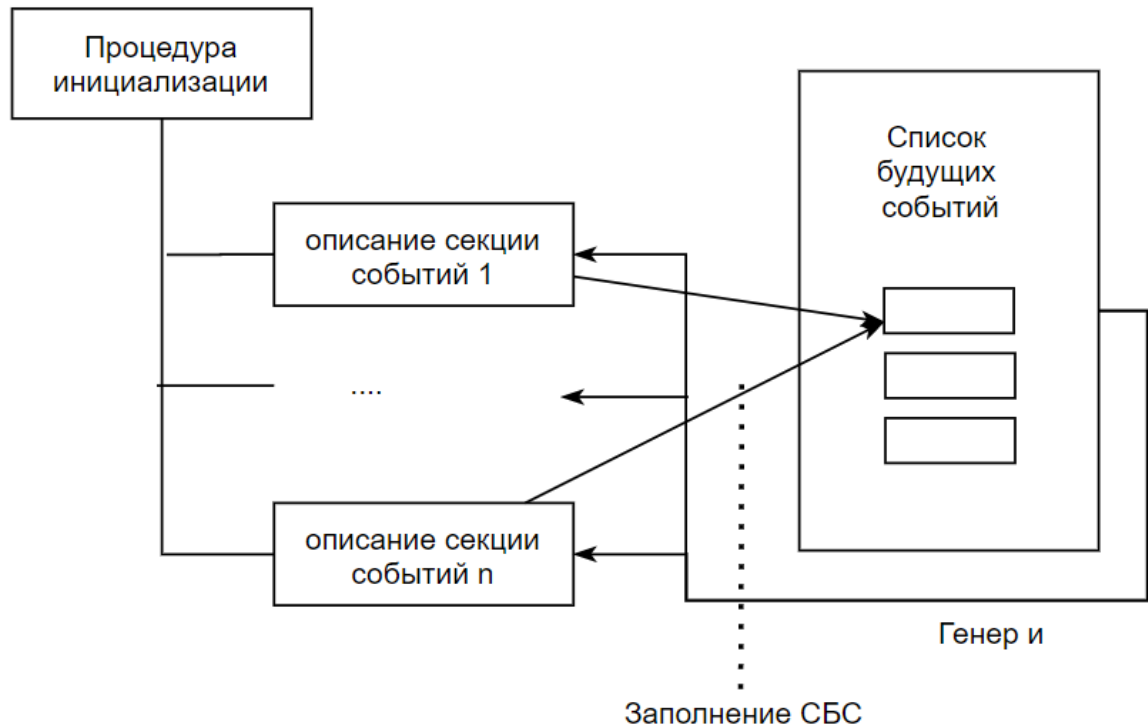
1. Первая задача сводится к тому, чтобы описать правила, задающие виды процессов, происходящих в системе.
2. Вторая задача заключается в том, чтобы описать правила задания атрибутов или задать правила генерации этих значений. При этом система определяется на конкретном уровне детализации в терминах множества 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.

Т.к. система в общем случае служит для описания временного поведения, то язык моделирования дискретных систем должен обладать средствами, отображающими время. В реальной системе совместно выполняются несколько активностей, принадлежащих как связанным, так и не связанным процессам. Имитация их действий должна быть строго последовательной. Таким образом, модель системы можно рассматривать как модель описаний, активностей, событий или процессов. Отсюда и деление языков моделирования.

3.1. Языки, ориентированные на события.

Моделирующая программа организована в совокупности в виде секций событий. Секция событий состоит из набора операций, которые выполняются после завершения какой-либо активности. Выполнение функций синхронизировано списком будущих событий.

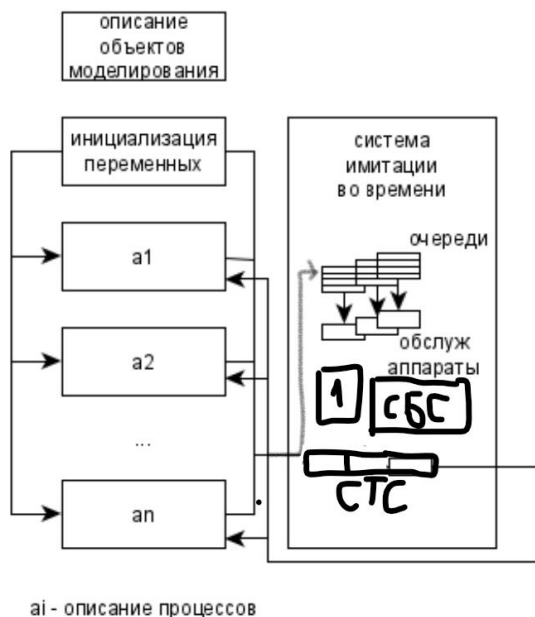
Структура программы на языке SIMSCRIPT.



3.2. Языки, ориентированные на процессы.

Моделирующая программа организуется в виде набора описаний процессов, каждый из которых описывает один класс процессов. Описание процесса функционирования устанавливает атрибуты и активности всех процессов. Синхронизация операций во времени реализуются так же с помощью списка будущих событий, который содержит точки активации/прерывания активности/процесса.

На примере языка SIMULA: CTC – список текущих событий, 1 – список задержек во времени (когда прервать процесс)



3.3. Языки, ориентированные на транзакты. GPSS

4. Сравнение языков

Результаты экспертных оценок сравнения различных языков при моделировании большого класса систем.

По убыванию

1. Возможность языка: SIMULA -> SIMSCRIPT -> GPSS -> C -> Python
2. Простота применения: GPSS -> SIMSCRIPT -> SIMULA -> Python -> C
3. Предпочтение пользователей: SIMSCRIPT -> GPSS -> SIMULA -> Python -> C

5. Сравнение универсальных и специализированных языков программирования при моделировании

Универсальный

- Преимущества:
 - минимум ограничений на выходной формат,
 - широкая распространенность
- Недостатки:
 - Значительное время, затраченное на программирование,
 - значительное время, затраченное на отладку

Специализированные языки

- Преимущества:
 - меньшие затраты на программирование;
 - более эффективные методы выявления ошибок;
 - краткость, точность выражений, понятий, характеризующих имитируемые конструкции;
 - возможность заранее строить стандартные блоки, которые могут использоваться в любой имитационной модели
 - автоматическое формирование в определенных типах данных, необходимых именно в процессе имитационного моделирования;
 - удобство накопления и представления выходных данных;
 - эффективное использование ресурсов
- Недостатки:
 - необходимо точно придерживаться ограничений на форматы данных;
 - меньшая гибкость модели

6. Возможности ПО и имитационного моделирования

- Анимация и динамическая графика
 - Представить суть имитационной модели (для руководителя)
 - С целью отладки моделируемой программы
 - Показать, что модель неправильная
 - Обучение обслуживающего персонала

Существуют два типа анимаций:

- Совместная: осуществляется в процессе имитационного моделирования
- Раздельная: состояния сохраняются в файле

- Статистические возможности
 - Необходимость поддерживать генератор псевдослучайных чисел

7. Объектно-ориентированное моделирование

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

Преимущества:

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

Недостатки: пакеты очень трудны для изучения

Назовем некоторые предметно-ориентированные пакеты имитационного моделирования:
Для производственных систем – *promodel* Для сетей связи – *itdecision*, *opnetmodelling*

Универсальные пакеты имитационного моделирования: *arena*, *extend*, *asim*, *somul8*, etc....

26. TODO Типы СМО для моделирования сложных систем

27. TODO Сети массового обслуживания, открытые, замкнутые, смешанные (написано самостоятельно)

Му Сети Петри

Формальное описание

Сеть Петри – это математическая модель дискретных динамических систем, ориентированная на качественный анализ и **синтез** таких систем (обнаружение блокировок, тупиковых ситуаций, "узких" мест ПО, автоматический синтез параллельных программ и компонентов компьютера)

Каких дискретных динамических систем: параллельных программ, ОС, устройства компьютеров, от структурного до схематического уровня, сетей компьютеров

Определения:

- теоретико-множественное
- графовое

Формальное описание в терминах теории систем

$PN = \{\theta, P, T, F, MO\}$

1. $\theta = \{\theta = 0, 1, 2, \dots\}$ – множество дискретных моментов времени, начиная с 0
2. $P = \{p_1, p_2, \dots, p_n\}$ – непустое множество: позиции, места
3. $T = \{t_1, t_2, \dots, t_m\}$ – непустое множество: переходы
4. $P \cap T = \emptyset$
5. F – функция инцидентности, связывающая множества вершин $F: (P \times T) \cup (T \times P) \rightarrow \{0, 1, 2, \dots, k\}$, где k -кратность дуги

6. M_0 – начальная маркировка позиции

Для каждой позиции есть свойства, количество свойств определяется начальной маркировкой.

F может быть представлена в виде 2 отображений:

1. $F^p(p, t): P \times T \rightarrow \{0, 1, 2, \dots\}$ – для каждой позиции указываются переходы с учетом кратности
2. $F^t(t, p): T \times P \rightarrow \{0, 1, 2, \dots\}$ – для каждого перехода указываются связанные с ним позиции с учетом кратности

Это все 2 матрицы

Из вершины позиции pi из P ведет дуга в вершину-переход tj из T тогда и только тогда, когда $f^p_{ij} > 0$. В этом случае говорят, что tj – выходной переход позиции pi . Множество позиций pk , для которых tj – выходной переход, будем обозначать $P^j = \{pk: f^p_{ij} > 0\}$.

Аналогично для f^t

Каждая позиция pi содержит некоторый целочисленный ресурс $\mu(pi) \geq 0$. Часто он отображается числом фишек/точек внутри позиции, каждая фишка отображает свойство вершины.

$M = (\mu_0, \mu_1, \mu_2, \dots)$ – вектор-маркировка/разметка сети Петри. $M: P \rightarrow \{0, 1, 2, \dots\}$

Начальная маркировка M_0 определяет стартовое состояние сети Петри.

Динамика поведения моделируемой системы описывается в терминах **функционирования сети Петри** в дискретном времени из кортежа в асинхронном режиме, переходя от одной маркировки к другой.

Смена маркировок, начиная с M_0 , происходит в результате **срабатывания переходов сети**

Переход $tj \in T$ может сработать при маркировке μ для всех $pi \in P$, если $\mu_i(\theta) - f^p_{ij}(\theta) \geq 0$ (если есть, что забрать), т.е. если каждая входная позиция для данного допустимого перехода содержит как минимум столько фишек, какова кратность ведущей к tj -му переходу дуги

Маркировка в след. момент времени $\theta+1$ сменяется на $\mu_i(\theta+1) = \mu_i(\theta) - f^p_{ij}(\theta) + f^t_{ij}(\theta)$, т.е. переход t изымает из каждой своей входной позиции число фишек, равное кратности входных дуг, и посылает в каждую свою выходную позицию число фишек, равное кратности выходных дуг

Если может сработать несколько, то срабатывает **1 любой**

Функционирование сети останавливается, если при некоторой маркировке ни один из переходов сработать не может (**тупиковая маркировка**)

В силу недетерминированного функционирования сеть Петри может порождать различные последовательности срабатывания переходов. Последовательности образуют **слова, алфавитом** которых является множество переходов T . Множество всевозможных слов, порождаемых сетью Петри, называют **языком** сети Петри. 2 сети Петри **эквивалентны**, если порождают один и тот же язык.

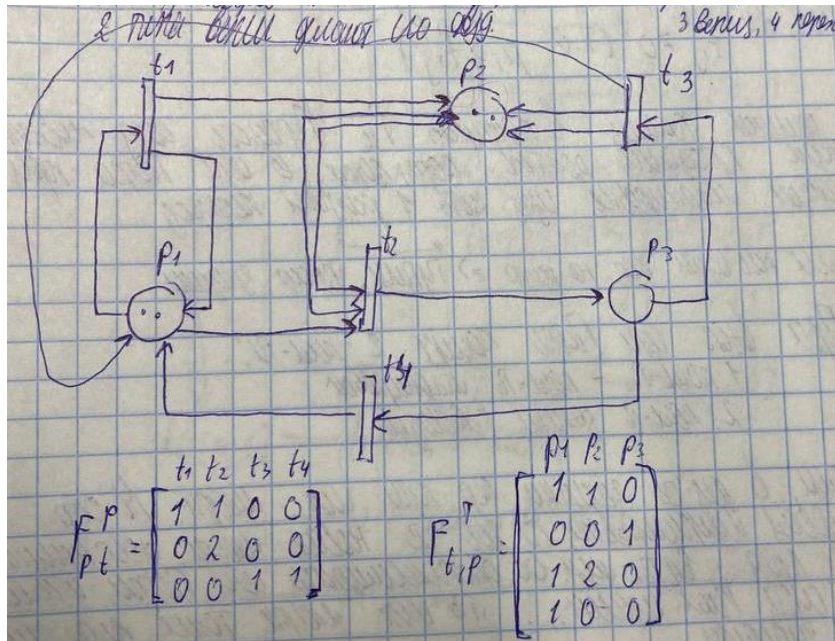
В отличие от конечных автоматов, в терминах которых описываются глобальные состояния системы, сети Петри концентрируют свое внимание на локальных событиях (переходы) (локальных условиях (позиции)) и локальных связях между событиями и условиями => в терминах сетей Петри более адекватно, чем с помощью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

Графы сетей Петри.

Теоретико-графовым представлением сетей Петри является двудольный ориентированный мультиграф (кратные ребра) данной сети

Граф содержит позиции (места) обозначенные кружочками; переходы, обозначенные планками; ориентированные дуги (стрелки), соединяющие позиции с переходами и переходы с позициями. Кратные дуги обозначаются несколькими параллельными дугами.

Матрицы и граф представляют взаимно-однозначное соответствие.



Пространство состояний сети Петри

Состояние сети определяется ее маркировкой. Пространство состояний, обладающее n позициями - множество всех маркировок E . Изменение в состоянии, вызванное запуском перехода, определяется функцией перехода или функцией следующего состояния.

Переход – изъятие фишек из позиции p_i и помещение в позицию p_k

Когда эта функция применима и срабатывает переход t_j (если он разрешен), то в соответствии с условием работы, получаем новую маркировку: $M(\theta + 1) = F(M(\theta), t_j)$, то есть новая маркировка получается изъятием фишек из позиций p_i таких, что $f^{\wedge}p_{ij} \neq 0$ (то есть $m_{ij} \geq f^{\wedge}p_{ij}$). Эти фишки помещаются в позицию p_k

Процесс создания новых маркировок продолжается, пока в выбранной сети Петри при данной маркировке существует хотя бы 1 разрешенный переход. Если же ни один переход не разрешен, то маркировка называется тупиковой

Получается 2 последовательности:

1. последовательность маркировок
2. последовательность запущенных переходов

Если в результате запуска перехода при исходной маркировке M образуется новая маркировка, то говорят, что новая маркировка **достижима** из предыдущей.

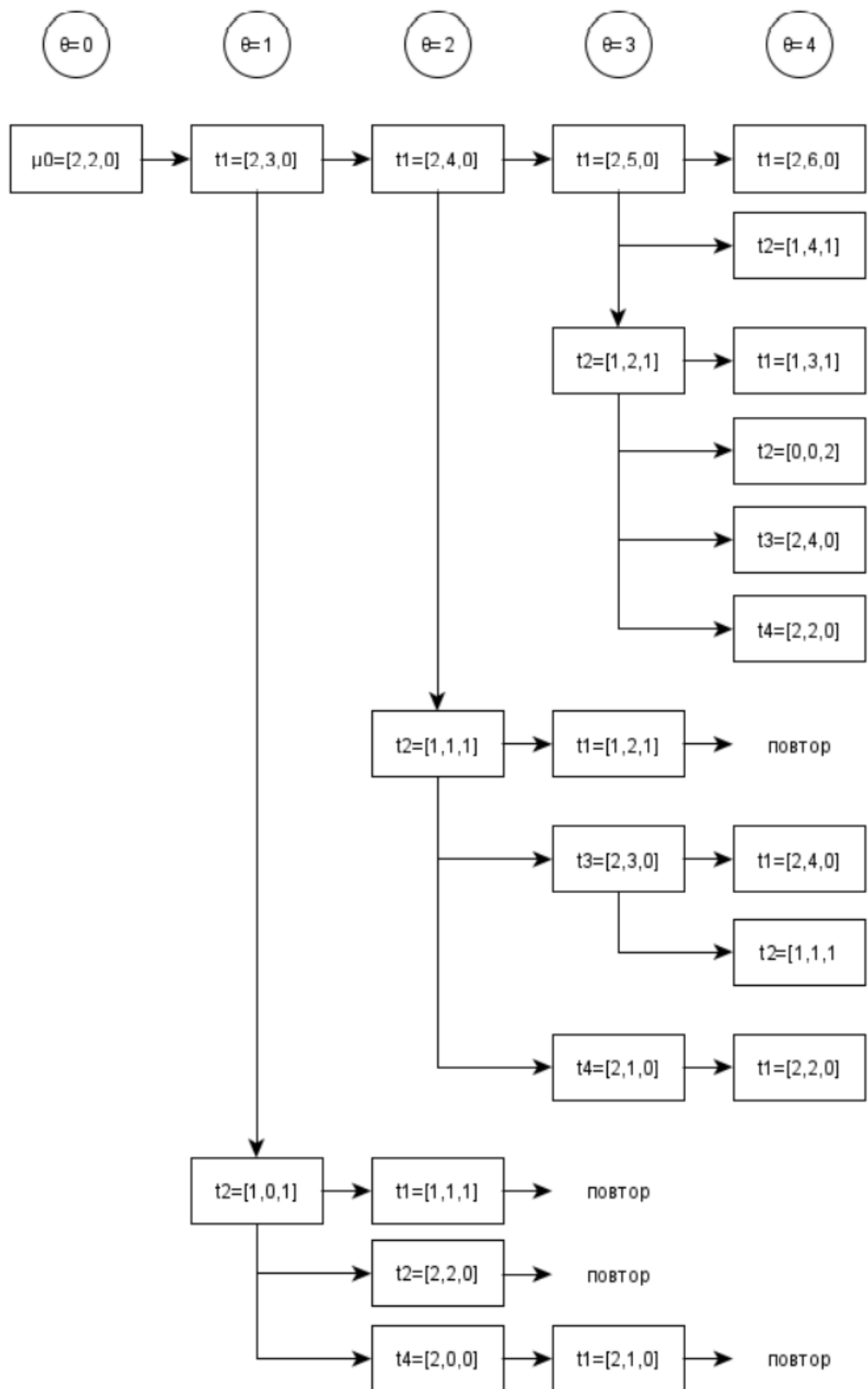
Множество достижимости $R(PN, M)$ – наименьшее множество маркировок, такое что:

$$M' \in R(PM, M)$$

$$M'' \in \delta(M', t_{j\tau})$$

$$M'' \in R(PN, M)$$

Пример (сх. выше)



Каждая из полученных маркировок порождает новую в результате чего получаем дерево маркировок. В дереве маркировок могут встречаться повторяющиеся маркировки, в этом случае дальнейшее построение дерева ведется только для 1 из них

Если выделить путь по дугам графа от μ_0 до μ'' и выписать подряд все символы переходов, то полученное слово образует последовательность срабатывания. А их совокупность – язык сети Петри. Он начинается со специального символа λ (пустое). Для примера $\lambda, t_1, t_1 t_2, \dots$

Свойства сетей Петри

1. свойство ограниченности

Позиция называется ограниченной, если для любой достижимой в сети маркировки μ существует такое k , что $\mu \leq k$. **Сеть называется ограниченной**, если все позиции ограничены

Пр – сеть не ограничена, т.к. возможен неограниченный рост μ_2 . Выделяется маркировка $\mu' = [2, \omega, 0]$

2. свойство безопасности

Сеть называется безопасной, если вектор любой достижимой маркировки двоичен (состоит из 0 и 1)

3. свойство консервативности

Сеть называется консервативной, если сумма фишек во всех позициях постоянна при работе сети (сумма по $\tau a_{\tau i} = \text{const}$)

4. свойство живости

Переход t_j называется потенциально живым, если существует достижимая из μ_0 какая-то маркировка μ' , при которой переход t_j может сработать.

Если t_j – потенциально живой при любой достижимой в сети маркировке, то он называется **живым**

Если не потенциально живой, то называется **мертвым**. **Маркировка μ_0** в этом случае называется **тупиковой** для перехода t_j . Если маркировка μ_0 – тупиковая для всех t_j то маркировка μ_0 – тупиковая (является листом)

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

Если начальная маркировка совпадает с конечной, то образуется цикл, и каждому циклу соответствует последовательность слов.

Некоторые сети Петри

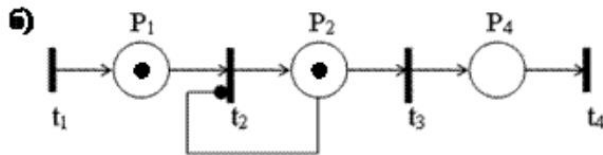
1. Ингибиторные сети Петри

Это сети Петри, для которых функция инцидентности имеет вид $F = F^{\wedge} p \cup F^{\wedge} t \cup F^{\wedge} \bar{t}$, т.е. она дополнена специальной функцией инцидентности $F' = P \times T \rightarrow \{0, 1\}$

Ингибиторные дуги имеют кратность 1 и связывают только позиции с переходами.

которая вводит ингибиторные дуги для тех пар (p, t) , для которых $F'(p, t) = 1$. Переход t ингибиторной сети может сработать, если каждое его входное место p , соединённое с t дугой кратности $W(p, t)$, содержит не менее $W(p, t)$ фишек, а каждое входное место, соединённое с t ингибиторной дугой, имеет нулевую разметку

Особой разновидностью сетей Петри являются ингибиторные сети, которые в дополнение к обычным дугам (ветвям) графа сети содержат запрещающие, так называемые ингибиторные ветви. Такая ветвь запрещает активацию перехода при наличии достаточного количества меток во входных вершинах обычных дуг до тех пор, пока в ее выходной вершине имеются метки. Пример реализации простейшего цикла обслуживания с использованием ингибиторной сети Петри представлен на рис.22-б. Здесь переход t_2 при наличии метки в позиции P_2 будет «заперт» не смотря на наличия метки в вершине P_1 до тех пор, пока метка не покинет P_2 через переход t_3 , что эквивалентно завершению очередного обслуживания.

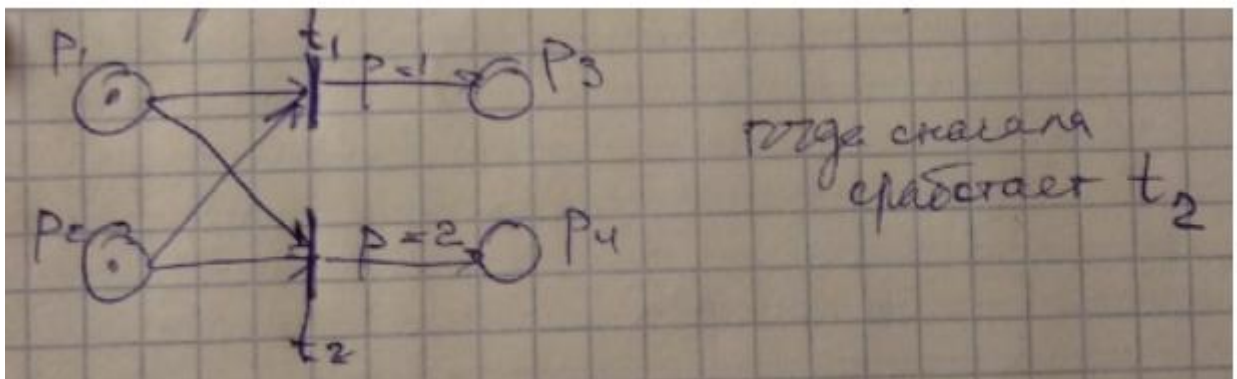


2. Сети Петри с приоритетами

Характер работы сети Петри недетерминированный т.е. если имеется возможность срабатывания нескольких переходов, то может сработать любой

Необходимо регламентировать последовательность срабатывания, введя понятие множества приоритетов и приписав каждому переходу соответствующее целочисленное значение

В этом случае правила срабатывания модифицируются: если на такте сети Петри могут сработать несколько переходов, то срабатывает тот из них, который имеет наивысший приоритет



3. Сети Петри со случайным срабатыванием переходов

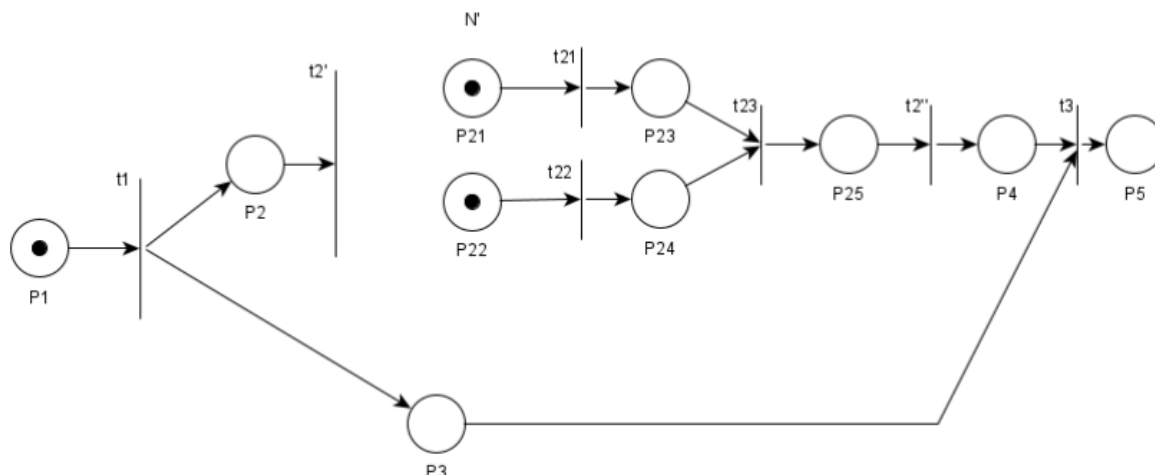
Указываются вероятности срабатывания переходов (сумма вероятностей = 1). Сеть Петри вырождается, если маркировки с состояниями сети (маркировки зависят только от предыдущей), то получим марковскую цепь.

4. Иерархические сети Петри

Представляют собой многоуровневые структуры, внутри которых присутствуют подсети различных уровней. Отсюда это замечательный инструмент формализации многоуровневых иерархических систем.

В отличие от обыкновенных сетей Петри в иерархических имеется 2 типа перехода: простые и составные. Простой переход ничем не отличается от того что мы рассматривали ранее. Составной переход содержит внутри себя сеть Петри более низкого иерархического уровня. Формально они состоят из входного (головного) и выходного (хвостового) перехода. Именно между ними находится сеть. Причем уровень вложенности не ограничен.

Пример иерархической сети



В данной сети имеется составной переход t_2 , содержащий сеть N' . В этой сети t_2' - голова t_2'' - хвост. То, что между, - это сеть, состоящая из позиций p_{21} - p_{25} и переходов t_{21} - t_{23} .

Иерархическая сеть функционирует как обычная сеть Петри, переходя от одной маркировки к другой, обмениваясь фишками (в том числе и между сетями различного уровня). Исключение составляет работа составных переходов.

Срабатывание составных переходов является не мгновенным событием, а составным действием. Поэтому мы говорим не о срабатывании перехода, а его работе.

На каждом шаге дискретного времени t_{eta} составной переход может находиться в одном из двух состояний. В пассивном и активном. Начальное состояние всех переходов пассивное. Составной переход может стать активным в некоторые моменты времени t_{eta} , если до этого он был пассивен и имеются условия для срабатывания головного перехода. При этом производится изменение маркировки в сети верхнего уровня по обычным правилам. И одновременно запускается работа сети, находящейся внутри перехода.

Как только запущена подсеть, работа верхней сети блокируется. Сеть нижнего уровня работает с учетом своей начальной маркировки до тех пор, пока все ее переходы не станут пассивными, после этого работает хвостовой переход и происходит изменение маркировки сети верхнего уровня. Составной переход принимает свое начальное пассивное состояние, а в сети нижнего уровня восстанавливается начальная маркировка.

На шаге $t_{eta} = 2$ происходит выполнение составного перехода в следующем порядке: срабатывает t_2' , запускается N' , окончание работы N' восстановление начальной маркировки, срабатывает t_2 и продолжение работы всей сети.

Этот процесс напоминает выполнение подпрограммы, где работа перехода t_2' соответствует вызову подпрограммы, а t_2'' - возврату в основную программу.

5. Цветные (раскрашенные) сети Петри CPN

В ряде приложений перемещаемые в сети ресурсы требуется дифференцировать, и тогда приходится вводить фишки различных видов (цветов). В этом случае для каждого перехода необходимо указывать, при каких комбинациях фишек во входных позициях переход может сработать, и какое количество фишек различных цветов перемещать в выходные позиции. Кортеж из 10 элементов

CPN – функционирует через мультимножество.

$CPN = (teta, sum, P, T, A, N, C, G, E, I)$

1. T – множество дискретных моментов времени, в которых происходит функционирование сети.
2. Sum – конечное множество, называемое «цветами»,
3. P – конечное множество позиций,
4. T – конечное множество переходов,
5. A – конечное множество дуг, связывающих между собой позиции и переходы.

В отличие от обыкновенных сетей Петри, где дуги задаются матрицами инцидентности, здесь дуги указываются с помощью множественного выражения. Такая нотация введена здесь для того, чтобы можно было пару элементов соединить несколькими дугами, с различными свойствами.

6. $N(A)$ узловая функция которая для каждой дуги указывает ее исходный и конечный узел. (дуга может быть указана! записью!)
7. C – цветовая функция, определяющая множество типа цветов, разрешенных для каждой позиции.
8. G – блокировочная или спусковая функция, описывающая дополнительные условия, которые должны быть выполнены для срабатывания перехода t . Эта функция представляет собой предикат, составленный из переменных, принадлежащих к типу цветов.
9. E – функция, задающая выражения на дугах, причем для каждой дуги она формализована в виде мультимножества, состоящих из элементов описанных в множестве цветов.
10. I – функция инициализации сети CPN .

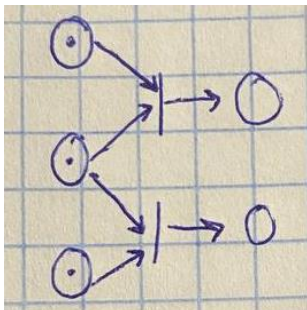
Моделирование дискретных систем, формализованных сетями Петри

При описании сетей Петри выделяют 2 понятия:

1. Событие – это действие системы – переходы
2. Условие – позиция. Предусловие – условие до срабатывания перехода, постусловие – после

Если процесс в системе достаточно сложный, то его подсистему можно представить в виде не примитивных событий. Если переходы не влияют друг на друга, то в словарь входят слова как с одного перехода, так и с другого.

Конфликт



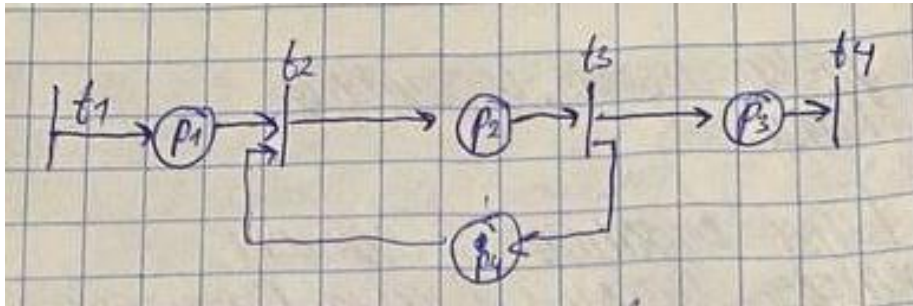
Промоделируем СМО. Концептуальная модель имеет входной поток заявок, пока выполняется заявка, нельзя взять следующую. Рассмотрим множество событий и условий, характеризующих нашу СМО.

Условия:

- P1 – задание ждет обработки
- P2 – задание обрабатывается
- P3 – процессор свободен
- P4 – задание ожидает вывода

События:

- T1 – задание помещено во входную очередь
- T2 – начало выполнения задания
- T3 – конец выполнения задания
- T4 – задание выводится



Начальная маркировка то соответствует состоянию, когда система свободна и заявки на обслуживание отсутствуют. При срабатывании t_1 от внешней среды поступает задание и получается маркировка $m_1=\{1010\}$, при этом может сработать переход t_2 (обслуживание) и $m_2=\{1100\}$, затем может сработать t_3 (окончание обслуживания) и $m_3=\{0011\}$

Переходы t_1 и t_4 работают независимо от t_2 и t_3 , моделируя поступление и вывод. Если задания поступают от 2 источников и попадают в 2 очереди, то имеет место двупоточная СМО. Вывод обработанных заданий осуществляется одним потоком.

Язык General Purpose System Simulation (GPSS)

GPSS – это общесистемная программная среда моделирования, предназначенная для того, чтобы ... с очередями

Система GPSS построена в предположении, что моделью сложной дискретной системы является описание ее элементов и логических правил их взаимодействия. Для определения классов моделируемых систем в GPSS выделяют конечный набор абстрактных элементов, называемых объектами. Набор логических правил также ограничен и описывается стандартными операциями.

Объекты языка делятся на 7 категорий и 14 типов

Язык GPSS –общецелевая система моделирования. Как и любой язык программирования, она содержит словари и грамматику, с помощью которых разрабатываются имитационные модели сложных дискретных систем версий 1, 2, V, GPSS /PC, GPSS WORLD.

Позволяет:

- Многозадачность
- Использование виртуальной памяти
- Интерактивность
- Графический интерфейс пользователя.

- Визуализация процесса моделирования.

Основное применение (то, что заявлено в качестве рекламы):

- Транспорт (самая известная модель: эксплуатация парка самолетов в авиационно-технической транспортной компании)
- Сетевые технологии. Исследование распределенной региональной сети передачи данных.

64

- Промышленность. Имитация автоматизированного металлургического производства.
- Финансовые и медицинские аспекты.

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