Оглавление

Переходы по лекциям:

- *8 лабораторных начиная* с октября. Сдавать можно также, когда у Рудакова первый курс.

ЛР№1: "Исследование псевдослучайных чисел".

Таблица, генерируется табличным и алгоритмическим способом последовательность псевдослучайных чисел. Числа:, ,

1разрядные (0-9)		2разрядные (10-99)		3разрядные (100-999)	

Под каждой из колонок - количественный критерий, который показывает, который из способов является лучшим. Критерий обосновать в отчете.

Табличный способ: использовать таблицу случайных чисел by математики.

На форме - возможность ввода последовательности чисел, также оцениваемой по критерию.

ЛР№2: "Изучение функции и плотности распределения заданной случайной величины"

1. равномерный поток (распределение), построить функцию и плотность; знать аналитическое выражение равномерности

2. по вариантам, распределение пуассона (первые номера группы) или гауссово (2-е номера), нормальное, эрланга.

Генератор случайных чисел, работающий по нужному распределению; сравнить с теоретическим

Лабораторная №4: почти i-й прибор обслуживания. Сгенерировать в систему сообщения (по ранее взятому закону). Очередь сообщений имеет ограничение по ёмкости. ОА работает равномерно (a+-b). Написать программу, в которой параметрически настраиваем прибор – задать параметры генератора, очереди, дисциплину обслуживания. Запустить процесс и набрать статистику. Идеальный вариант: динамически выводить состояния объектов.
Обратная связь – вышибает закон генерации сообщений, выходные заявки суммируются с сгенерированными. Определить оптимальную длину буфера (емкость памяти), при которой не будет потерь.

Лабораторная №5: в информационный центр приходят клиенты (пользователи) через интервалы времени 10+-2 минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 20+-5, 40+-10 и 40+-20 минут. Клиенты стремятся занять очередь свободного оператора с максимальной производительностью. Полученные запросы отправляются в накопитель (память), откуда выбираются для обработки. На первый компьютер: запросы от 1 и 2 операторов, на второй: от 3 оператора. Время обработки запроса на компьютере равны 15 и 30 минут. Промоделировать процесс обработки 300 запросов (на выходе). Определить вероятность отказа.

Лабораторные 7,8 по GPSS: повторение 4-й и 5-й лабораторных;

- Лекция №1 07.09.2015
- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*

- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*
- *Ошибка! Источник ссылки не найден.*

8 лабораторных начиная с октября. Сдавать можно также, когда у Рудакова первый курс.

ЛР№1: "Исследование псевдослучайных чисел".

Таблица, генерируется табличным и алгоритмическим способом последовательность псевдослучайных чисел. Числа:, ,

1разрядные (0-9)		2разрядные (10-99)		3разрядные (100-999)	

Под каждой из колонок - количественный критерий, который показывает, который из способов является лучшим. Критерий обосновать в отчете.

Табличный способ: использовать таблицу случайных чисел by математики.

На форме - возможность ввода последовательности чисел, также оцениваемой по критерию.

ЛР№2: "Изучение функции и плотности распределения заданной случайной величины"

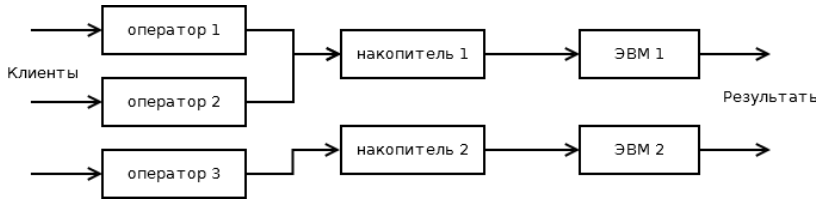
1. равномерный поток (распределение), построить функцию и плотность; знать аналитическое выражение равномерности

2. по вариантам, распределение пуассона (первые номера группы) или гауссово (2-е номера), нормальное, эрланга.

Генератор случайных чисел, работающий по нужному распределению; сравнить с теоретическим

Лабораторная №4: почти i-й прибор обслуживания. Сгенерировать в систему сообщения (по ранее взятому закону). Очередь сообщений имеет ограничение по ёмкости. ОА работает равномерно (a+-b). Написать программу, в которой параметрически настраиваем прибор – задать параметры генератора, очереди, дисциплину обслуживания. Запустить процесс и набрать статистику. Идеальный вариант: динамически выводить состояния объектов.
Обратная связь – вышибает закон генерации сообщений, выходные заявки суммируются с сгенерированными. Определить оптимальную длину буфера (емкость памяти), при которой не будет потерь.

Лабораторная №5: в информационный центр приходят клиенты (пользователи) через интервалы времени 10+-2 минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 20+-5, 40+-10 и 40+-20 минут. Клиенты стремятся занять очередь свободного оператора с максимальной производительностью. Полученные запросы отправляются в накопитель (память), откуда выбираются для обработки. На первый компьютер: запросы от 1 и 2 операторов, на второй: от 3 оператора. Время обработки запроса на компьютере равны 15 и 30 минут. Промоделировать процесс обработки 300 запросов (на выходе). Определить вероятность отказа.



Лабораторные 7,8 по GPSS: повторение 4-й и 5-й лабораторных;

Лекция №1----------07.09.2015

"Имитационное моделирование" 3 издание Лоу Кейтон (?)

"Имитационное моделирование систем" (РК9) Емельянов, Веселовский

Шрайдер "Моделирование на GPSS"

Советов, Яковлев "Моделирование систем" (МЭИ)

Бусленко "Моделирование сложных систем"

Общие вопросы теории моделирования

Философские аспекты моделирования

Моделирование позволяет:

1) сэкономить время и ресурсы

2) предсказать то, чего ещё нет

Объект – всё то, на что направлена человеческая деятельность. При определении объекта необходимо разобраться с "составом" объекта и его взаимодействием с "внешним миром".

Научно-техническое развитие в любой области обычно идёт по следующему пути:

- наблюдение и эксперимент
- теоретическое исследование
- организация производства

Большую роль играют **гипотезы** – определенные *предсказания и предположения*, основывающиеся на *небольшом* количестве опытных данных. Также используются наблюдения и догадки. При формировании и проверки правильности гипотез большое значение в качестве метода суждения имеет аналогия.

Аналогия – суждение о *частном сходстве* между двумя объектами. Современная научная гипотеза создаётся как правило по аналогии с проверенными на практике положениями. *//гипотеза -> аналогия -> эксперимент – аналогия связывает гипотезу и эксперимент.*

Гипотезы и аналогии, отражающие реальный объективно существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам.

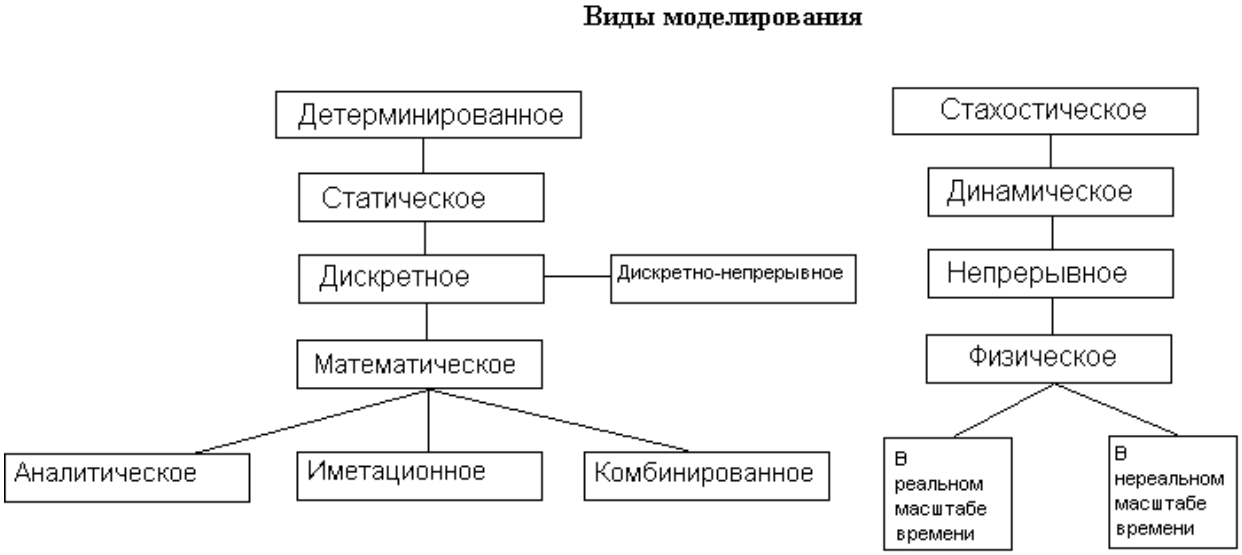
Такие логические схемы, упрощающие рассуждения и логические построения (либо позволяющие проводить эксперименты, уточняющие природу явлений), называются **моделями**. **Модель** – заместитель объекта-оригинала, обеспечивающий исследование некоторых его свойств. Замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели называется **моделированием**.

Лекция №2----------14.09.2015

Гносеология моделирования – показываются методы, с помощью которых мы познаём объект. Самое главное - понятие эксперимента. Только проведя группу экспериментов можно говорить, что модель отображает тот или иной аспект реального объекта.

Классификация методов моделирования

Моделирование систем:



полное неполное приближенное

+----------+

детерминированное стохастическое

| |

статическое динамическое

+----------+----------+

дискретное дискр-непр. непрерывное

| |

мысленное реальное----------натурное, физическое

наглядное символическое математическое

Наглядное: гипотетическое, аналоговое, макетирование.

Символическое: языковое, знаковое

Математическое: аналитическое, имитационное, комбинированное, информационное, структурное, ситуационное

Натурное: научный эксперимент, комплексные испытания, производственный эксперимент

Физическое: в реальном времени, в модельном времени

Основу классификации составляют критерии. В дальнейшем будем заниматься имитационным моделированием.

Классификация видов моделирования может быть проведена по *характеру* моделируемых объектов, *сферам приложения*, *глубине* моделирования.

По глубине методы делятся на две группы: материальное и идеальное.

Материальное – основывается на материальной аналогии объекта и модели. Оно осуществляется с помощью воспроизведения основных физических, функциональных или геометрических характеристик

изучаемого объекта. Частный случай - физическое моделирование, а его частный случай - аналоговое (на основе явлений, описываемых одинаковыми математическими соотношениями).

Идеальное – основывается на мысленной аналогии. Разбивается на два подкласса: знаковое (формализованное) и интуитивное (неформализованное). При знаковом, моделью являются формулы, чертежи; важнейшая форма - математическое.

В зависимости от типа носителя и сигнатуры модели различаются:

детерминированные – отображает процессы, в которых предполагается отсутствие случайных воздействий.

стохастическое – учитывает ***вероятностные*** процессы и события.

статическое – служит для описания состояния объекта в фиксированный момент времени.

динамическое – для исследования объекта во времени. При этом оперируют аналоговыми (непрерывными), дискретными и смешанными моделями.

В зависимости от формы реализации носителя моделирование делится:

мысленное - модели не реализуемы в заданном интервале времени, либо отсутствуют условия для их физического создания. Такое моделирование реализуется в виде наглядного, символического и математического.

реальное

В основу гипотетического моделирования закладывается гипотеза о закономерностях функционирования реального объекта. Базируется на причинно-следственной связи между входом и выходом.

Макетирование применяется, когда процессы в реальном объекте не поддаются физическому моделированию.

Символическое моделирование представляет собой искусственный процесс создания логического объекта, который замещает реальный и выражает его основные свойства с помощью определенных знаков и символов. Знак - условное обозначение отдельного понятия.

Математическое моделирование – процесс установления в соответствие данному *реальному* объекту некоторого *математического* объекта, который и называется математической моделью.

При создании модели реального объекта и осуществлении процесса исследования необходимо провести формализацию (определение математических схем) процесса функционирования. Для представления математических моделей могут использоваться различные формы записи, основными из которых являются:

- *инвариантная* – запись соотношений модели с помощью традиционного математического языка, безотносительно к методу решений уравнений. Модель может быть представлена как совокупность входов, выходов, переменных состояния и глобальных уравнений системы.
- *алгоритмическая*
- *схемная*
- *аналитическая* – запись модели в виде результата решения исходных уравнений модели. Обычно модели в аналитической форме представляют собой ***явные*** выражения выходных параметров как функций входов и переменных состояния.

Для аналитического моделирования характерно, что в основном моделируется только *функциональный* аспект системы. При этом глобальные уравнения системы, описывающие закон её функционирования, записываются в виде некоторых аналитических соотношений: алгебраические, интегро-дифференциальные, конечно-разностные и т.д., а также логические условия.

Аналитическая модель исследуется несколькими методами:

аналитический – стремятся получить в общем виде явные зависимости, связывающие искомые характеристики с начальными условиями, параметрами и переменными состояния системы.

численный – для конкретных начальных данных.

качественный – ничего не можем решить, но можем найти некоторые свойства решения (например, устойчивость).

Распространены компьютерные методы исследования сложных систем. Строится алгоритм функционирования сложного объекта, а затем среди класса алгоритмических моделей выделяется класс имитационных моделей.

* При имитационном моделировании воспроизводится алгоритм функционирования системы *во времени*, причем имитируются *элементарные* явления, составляющие процесс, с сохранением их в логические структуры и последовательности протекания. Это позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени, дающие возможность оценить характеристику системы.

Основное преимущество имитационного моделирования (по сравнению с аналитическим) – возможность решения более сложных задач: достаточно "просто" учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики этих элементов, многочисленные случайные воздействия.

При изучении объекта, "лучшесть" модели выражается в точности – следовательно, аналитические модели лучше – математические схемы проверены и работают. Имитация - попытка (жалкая) построить копию; всегда возникает большая погрешность. Часто такая ошибка настолько завуалирована, что тяжело оценить конечный результат. Основной закон использования имитационной модели - многократные эксперименты и статистический анализ результатов.

Типы имитационного моделирования.

1. Агентное.

Используется для исследования децентрализованных систем, динамика функционирования которых определяется не глобальными правилами и законами, а когда эти самые правила и законы являются результатом индивидуальной активности членов группы. Цель агентных моделей – получить представление об этих глобальных правилах и общем поведении системы, исходя из предположений об индивидуальном частном поведении её отдельных (активных) объектов и их взаимодействии в системе. Используется, если удаётся выделить индивидуальные свойства и правила поведения; прямое или косвенное взаимодействие.

Под **агентом** понимают некую сущность, обладающую *активностью*, автономным поведением, которая может принимать решения в соответствии с некоторым набором правил. Главное – агент может самостоятельно изменяться.

2. Дискретно-событийное.

(разработано в 60-х) Предлагается абстрагироваться от непрерывной природы событий и рассматривать только основные события моделируемой системы. Пример на аэропорте: обработка заказов, регистрация, загрузка-разгрузка, ожидание и т.п. ДС-моделирование наиболее развито и имеет приложение от логистики и массового обслуживания, до транспортных и производственных систем. В качестве квазиобъектов выступают "заявки", ресурсы, процессы.

3. Системная динамика.

Для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие во времени. Затем созданная модель имитируется на компьютере. Бизнес-процессы, когнитивное моделирование ("модели развития города" и т.п.). Необходимо учитывать связанные переменные, накопители и обратные связи.

Нельзя обойтись одной аналитикой, одной имитацией – плохо. Необходимо комбинировать модели. Что имитировать и что анализировать? Всё явное – аналитическое. Модель становится иерархической. Из такой композитной системы методами декомпозиции нужно дойти до элементарных действий (когда эти действия можно описать типовыми математическими средствами).

Лекция №3----------21.09.2015

Подходы имитационного моделирования как иерархического способа моделирования

Дискретно-событийное моделирование	**Агентное моделирование**	**Системная динамика**
Заявки	Активные объекты	Накопители
Потоковые диаграммы	Индивидуальные правила поведения	Обслуживающие аппараты
Сети, ресурсы	Динамика среды	Потоки данных

Каждая следующая строка ниже с точки зрения абстракции. Столбцы идут слева-направо от дискретного времени к непрерывному.

Имитационное моделирование является эффективным, но имеющим недостатки методом моделирования сложных объектов. Трудности использования ИМ связаны с обеспечением *адекватности* описания системы, *интерпретацией результатов*, обеспечением *стохастической сходимости* процесса моделирования. Если что-то описывается в матричной форме, проблемой может являться размерность и трудоёмкость данного метода.

Часто перед построением имитационных моделей, являющихся динамическими по своей сути, оказывается полезным (а иногда и необходимым) осуществить предварительный *статический анализ*. При этом выявляются функции, выполняемые в системе, взаимосвязи элементов системы, формализуются потоки событий. Эти спецификации позволяют выявить определенный объем знаний и характеристик системы для построения имитационной модели и уменьшения вероятности ошибок.

В имитационном моделировании с точки зрения средств выделяют по возможностям и функциям исследования три основных этапа:

1. Создание имитационной модели на:
   - универсальном языке высокого уровня (Си, Паскаль, ...)
   - специализированном языке моделирования (GPSS)
   - объектно-ориентированном языке (ModSym)
2. Использование при разработке имитационных моделей проблемно-ориентированных систем и средств.

Такие системы как правило не требуют от исследователя знаний программирования, но позволяют моделировать лишь узкие классы моделируемых систем. При этом имитационная модель генерируется самой системой в процессе диалога с пользователем: это позволяет сделать описание системы достаточно быстрым, эффективным, избежать ошибок программирования. Таких систем несколько десятков, а языков - около семисот.

3. Использование методов искусственного интеллекта.

Прежде всего, знаний при принятии решений в процессе имитации, управления имитационным экспериментом, реализация интерфейса взаимодействия с пользователем.

Нерешенной проблемой остаётся самый первый этап: концептуализация при ИМ, то есть перевод поставленной задачи в плоскость формализации (желательно на язык математики). Этот этап не может быть решён ни программистом, ни аналитиком в данной области самостоятельно. Необходимо создавать комплексную группу, включающую разных специалистов; на начальном этапе проекта вырабатывается формализм системы.

Ещё одна проблема - наличие обратной связи (системы управления).

В качестве примеров интеллектуальных средств моделирования можно рассмотреть системы RDO, AnyLogic и Arena.

* В теории систем встречаются следующие базовые понятия:

Система – множество элементов (Упорядоченная совокупность), находящихся в отношениях и связи между собой.

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

Сложная система – система, характеризуемая большим числом элементов и большим числом взаимодействий элементов. Сложность системы определяется также видом взаимосвязей элементов, свойствами (целенаправленности, целостности, членимости, иерархичности, многоаспектности).

Подсистема – часть системы (подмножество элементов и связей между ними), которая имеет свойства системы.

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

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

Параметр – величина, выражающая свойства системы или её части, или влияющая на окружающую среду.

Рассмотрим **свойства** сложной системы

- *Целенаправленность*. Свойство искусственной системы, выражающее назначение системы. Необходимо для оценки эффективности вариантов системы.
- *Целостность*. Свойство системы, характеризующее взаимосвязанность элементов и наличие зависимости выходных параметров от параметров элементов. Причём, большинство выходных параметров не являются простым повторением или суммой параметров элементов.
- *Иерархичность*. Свойство сложной системы, выражающее возможность и целесообразность её иерархического описания: т.е. представление системы в виде нескольких уровней, между компонентами которых имеется отношение "целое - часть".

В англоязычной литературе встречаются термины modelling (только процесс *построения* модели) и simulation («запуск» модели и *анализ* её свойств (и свойств системы, которую она отражает)).

Технические средства моделирования

Роль технических средств – *создание системы и проведение эксперимента*. Выполнять моделирование позволяют две структуры (два способа обработки информации): цифровое и аналоговое.

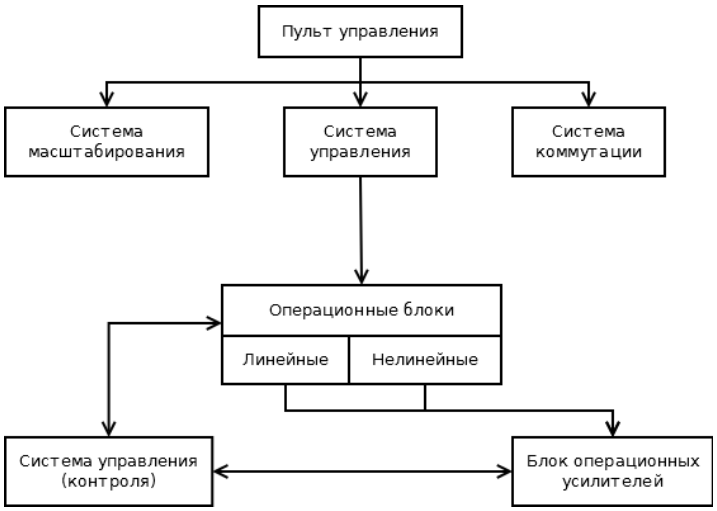
При построении цифровой вычислительной техники используются принципы Фон Неймана: программа представляется как данные; происходит разделение на ЦП (устройство для выполнения арифметическо-логических операций и управление вычислительным процессом), запоминающее устройство, ввод, вывод, устройство управления; двоичная арифметика (сложение и сдвиги, основное устройство - сумматор)...

Аналоговая техника. В основе заложен принцип моделирования, а не счёта. При использовании в качестве модели некоторой задачи электронных цепей, каждой переменной величине задачи ставится в соответствие определенная переменная величина электронной цепи. Основой построения такой модели является *изоморфизм* (подобие) *исследуемой задачи и соответствующей ей электронной модели*. В большинстве случаев при определении критериев подобия используются специальные приемы масштабирования соответствующих значений параметров модели и переменных задачи.

Аналоговая вычислительная машина реализует модель, изоморфную исследуемой задаче. Согласно своим вычислительным возможностям, АВМ наиболее приспособлены для исследования объектов, динамика которых описывается обыкновенными и в частных производных дифференциальными уравнениями. Также имеют место быть алгебраические уравнения. АВМ можно отнести к спецмашинам.

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

Главный элемент аналоговой техники – пульт управления.



С точки зрения моделирования, аналоговая техника быстрее. Она ограничена скоростью электронов; мгновенно получаем картинку (графики). Для обработки оных необходимо подключать цифровую технику, но результат получается сразу же. Главный недостаток: точность практически никакая.

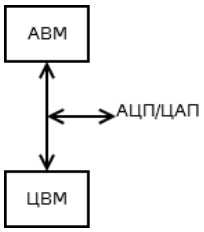
АВМ общего назначения делят по производительности на малые, средние и большие. Речь здесь не про быстродействие, а про степень в уравнениях: 0<10 малые, <100 средние, 100+ большие.

Лекция №4----------28.09.2015

Гибридная вычислительная машина – широкий класс вычислительных систем, использующих как аналоговую, так и дискретную форму представления сигнала (обработки сигнала). Подклассы:

- АВМ, использующие цифровые методы численного анализа (Итератор-1)
- АВМ, программируемые с помощью ЦВМ. Программируются масштабные коэффициенты, которые приводят физические параметры к параметрам модели
- АВМ с цифровым управлением и логикой (Хайдак)
- АВМ с цифровыми элементами (вольтметры, запоминающие устройства, но не решающие элементы)
- ЦВМ с аналоговыми арифметическими устройствами
- ЦВМ, допускающие программирование аналогового типа (дифференциальные анализаторы).

Общая схема ГВМ:

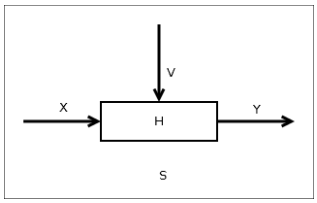


Сравнение характеристик АВМ и ЦВМ

Показатель	**АВМ**	**ЦВМ**
Тип информации	Непрерывный	Дискретный
изменение значений	величиной напряжения	числовым значением
базовые операции	арифметическое интегрирование	арифметические
принцип вычисления	высокопараллельные	последовательно-параллельные
режим реального времени	без ограничений	ограниченные возможности
динамическое изменение решаемой задачи	посредством системы коммутации	в диалоговом режиме
основные профессиональные требования к пользователю	профессиональные знания + методика моделирования	знание основ ПО ЭВМ + алгоритмизация
уровень формализации задач	ограничены моделью решаемой задачи	высокий
способность решения логических задач	ограниченная	высокая
точность вычислений	порядка 1e-4	ограничена разрядностью, ~1e-40
диапазон представления чисел	1..1e-4	1e-40..1e40
класс решаемых задач	дифференциальные и алгебраические уравнения	всё
специальные функции	ограниченный набор	широкий класс специализированных функций
уровень миниатюризации	ограниченный	высокий
сферы применения	спецмашины	практически везде
пользовательский интерфейс	низкого уровня	высочайший уровень

Основы теории моделирования

Моделируемый объект – сложная дискретная система, на которую подается множество воздействий



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

1. совокупность **входных** воздействий $x_i \in X, i = 1, n_x$
2. совокупность воздействий **внешней среды** $v_l \in V, l = 1, n_v$
3. совокупность **внутренних** собственных параметров системы $h_k \in H, k = 1, n_h$
4. совокупность **выходных** параметров системы $y_j \in Y, j = 1, n_y$

В общем случае X,V,H,Y являются элементами непересекающихся подмножеств и содержат как детерминированные, так и стохастические характеристики.

При моделировании функционирования сложной системы S, входные воздействия X, внутренние параметры H и воздействия внешней среды V, являются независимыми (экзогенными). В векторной форме можно записать следующим образом:

$$\vec{x}(t) = \left(x_1(t), x_2(t), \dots, x_{nx}(t)\right)$$
$$\vec{v}(t) = \left(v_1(t), v_2(t), \dots, v_{nv}(t)\right)$$
$$\vec{h}(t) = \left(h_1(t), h_2(t), \dots, h_{nh}(t)\right)$$

Выходные характеристики системы являются зависимыми (эндогенными):

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

Вводится параметр *времени* – наиважнейший элемент моделирования.

Процесс функционирования системы С описывается во времени некоторым оператором, который в общем случае преобразует независимые переменные в зависимые, в соответствии со следующим соотношением: $y(t) = F_s(\vec{x}, \vec{v}, \vec{h}, t)$.

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

^ запись позволяет записать задачу в общем виде, формализовать её. Выделяются наиболее важные подклассы воздействий, указываются ограничения. Действия можно преобразовывать с помощью множества алгоритмов.

Важным является понятие **алгоритма функционирования** A_S, под которым здесь подразумевается метод получения выходных характеристик У с учетом входных воздействий X, воздействий внешней среды V и соответствующих параметров системы H. Один и тот же закон функционирования F_S может быть реализован различными способами – т.е. с помощью множества различных алгоритмов.

^ зависимость может быть получена через *свойства* системы в *конкретные моменты* времени, которые называются **<u>состояния</u>** системы:

$$\vec{z} = \left(z_1, \dots, z_p\right), p = 1, n_z$$

Если рассматривать процесс функционирования сложной системы S как последовательную смену состояний $(z_1(t), z_2(t), \dots)$, то они могут быть интерпретированы как координаты точки в k-мерном фазовом пространстве. Причем каждой реализации процесса будет соответствовать некоторая фазовая траектория. *Совокупность* всех $z(t)$ называется **пространством состояний** объекта моделирования. Состояние системы в момент $t_{\text{нач}} \le t \le t_{\text{кон}}$ полностью определяется начальными условиями $\vec{z}^0 = (z_1^0, z_2^0, \dots)$, где z_1^0 – состояние системы в момент времени t_0.

Состояние системы S в момент времени t от t_0 до времени окончания моделирования полностью определяется: начальными условиями $\vec{z}^0 = (z_1^0, z_2^0, \dots)$, входными воздействиями, внутренними параметрами и воздействиями внешней среды, которые имели место за промежуток времени $t_0 \dots t$ с помощью двух векторных уравнений:

$\vec{z}(t) = \Phi(\vec{z}^0, \vec{x}, \vec{v}, \vec{h}, t)$;. выходной параметр $\vec{y}(t) = (\vec{z}, t)$. Наконец, если поставить одно в другое, то $\vec{y}(t) = F(\Phi(\quad), t)$.

В общем случае время в модели может быть непрерывным на интервале, а может быть конечным.

Таким образом, под **математической моделью реальной системы** понимают конечное множество переменных $x(t), v(t), h(t)$ вместе с математическими связями между ними и характеристиками $y(t)$.

Типовые математические схемы

Выделяются по характеру процесса функционирования.

Процесс F	**Типовая математическая схема**	**Обозначения**
непрерывно-детерминированный подход	дифференциальные уравнения	D-схема
дискретно-детерминированный	конечные автоматы	F-схема
дискретно-стохастический процесс	вероятностные автоматы	P
непрерывно-стохастический	системы массового обслуживания (queueing system)	Q
универсальный подход	аггрегативные системы	A

Данные схемы были введены, чтобы "не изобретать велосипед". Эти схемы при своей реализации дают правильный результат, подтверждая адекватность модели.

Рассмотрим примеры.

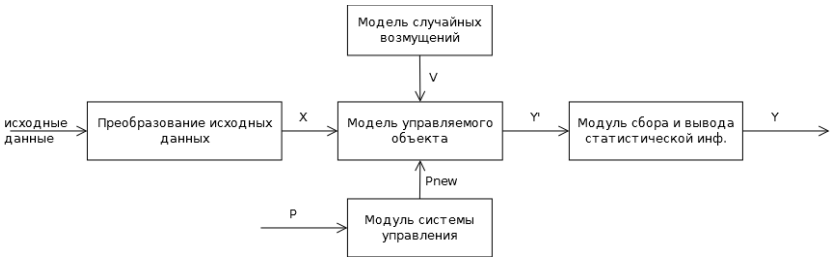
НДП: колебательный контур с ёмкостью C_k и индукцией L_k, или маятник массой M и длиной l.

В контуре: $L_k * \frac{d^2q(t)}{dt^2} + \frac{q(t)}{C_k} = 0$. $q(t)$ – заряд на конденсаторе. Период $T_k = 2\pi\sqrt{L_k C_k}$.

В маятнике: $m * l_M^2 * \frac{d^2\theta(t)}{dt^2} + m * g * l_M\theta(t) = 0$. Период $T = 2\pi\sqrt{\frac{l}{g}}$.

Суть объектов - разная, но процессы описываются обыкновенными ДУ. Введём понятия $h_2 = L_k = m * l_M, h_1 = 0 = 0, h_0 = \frac{1}{C_k} = m * g * l$. Если теперь положить состояние $z(t) = q(t) = \theta(t)$, то можно записать общий вид уравнения, описывающего функционирование данных объектов: $h_2 * \frac{d^2z(t)}{dt^2} + h_1 * \frac{dz(t)}{dt} + h_0 * z(t) = 0$. Нуль символизирует собой отсутствие внешних воздействий X (включающих в себя заодно и V).

Состав имитационной модели сложной дискретной системы. На входе: поток исходных данных, затем модули преобразования этих данных.



Лекция №5----------05.10.2015

Формализация и алгоритмизация процесса функционирования сложных систем

Сущность компьютерного моделирования – проведение эксперимента с моделью, который представляет собой некоторый программный комплекс, описывающий формально или алгоритмически поведение элементов системы в процессе её функционирования (т.е. взаимодействия их друг с другом и с внешней средой).

Основные требования, предъявляемые к модели в процессе функционирования системы:

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 модели. Структура должна быть *блочной* – допускать возможность замены, добавления, исключения некоторых частей схемы без переделывания всей модели.
3. компьютерная реализация модели должна *соответствовать* имеющимся техническим ресурсам (память, быстродействие, БД и т.п.).

Процесс моделирования, включая разработку и машинную реализацию модели, является итерационным. Этот итерационный процесс продолжается до тех пор, пока не будет получена модель, которую можно считать адекватной в рамках поставленной задачи. Вычленяется объект на базе некоторого анализа, синтезируется, и снова начинает анализироваться.

Основные **этапы** моделирования больших систем:

I. Построение **концептуальной** (описательной) модели.

- формулируется модель
- строится её формальная схема

Основное назначение - переход от содержательного описания объекта к его математической модели. Необходимо определить, какие накладываются ограничения, чтобы не рассматривать проблему в общем. Если накосячить – ошибка приведет к фатальному результату.

Исходный материал этапа: *содержательное описание объекта*. Последовательность действий:

- проведение границ между системой и внешней средой
- исследование моделируемого объекта с точки зрения выделения основных составляющих процесса функционирования системы (по отношению к цели моделирования)
- переход от содержательного описания системы к формализованному описанию свойств процесса функционирования (к концептуальной модели):
  - исключение из рассмотрения некоторых второстепенных элементов описания
- оставшиеся элементы модели группируются в блоки трех групп:
  i. имитатор воздействий внешней среды
  ii. собственно модель функционирования объекта
  iii. вспомогательные объекты

- процесс функционирования системы так разбивается на подпроцессы, чтобы построение отдельных подпроцессов было элементарно и не вызывало особых трудностей.

II. Этап **алгоритмизации** модели и ее программирование.

Исходный материал*: блочная логическая схема модели*. Последовательность действий:

- разработка схемы моделирующего алгоритма
- разработка схемы программы
- выбор технических средств для реализации компьютерной модели
- программирование модели и отладка программы
- проверка достоверности программы на тестовых примерах
- создание технической документации

III. Этап **получения и интерпретации** результатов моделирования

Компьютер используется для проведения рабочих расчетов по готовой программе. Результаты этих расчетов позволяют в результате анализа сделать вывод о характеристиках процесса функционирования моделируемой системы.

Последовательность действий:

- планирование машинного эксперимента.

Различают активный и пассивный эксперимент. Составление плана проведения эксперимента с указанием комбинаций переменных и параметров, для которых должен проводиться эксперимент. Необходимо получить максимальный объем информации об объекте моделирования при минимальных затратах машинного времени.

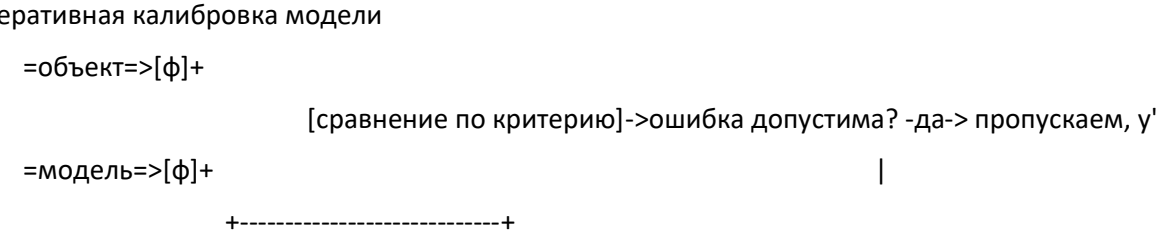
- проведение рабочих расчетов (контрольная калибровка модели)
- статистическая обработка результатов и представление их в наглядном (удобном пользователю) виде.
- составление технической документации

Различают **стратегическое** и **тактическое** планирование компьютерного эксперимента.

При **стратегическом** планировании ставится задача построения оптимального *плана* эксперимента для достижения цели, поставленной перед моделированием. (Оптимизация структуры с использованием генетических алгоритмов; параметрическая оптимизация; оптимизация алгоритмов)

Тактическое преследует частные цели оптимальной *реализации* каждого конкретного эксперимента из множества необходимых, заданных при стратегическом планировании.

Итеративная калибровка модели



Три класса ошибок (по убыванию страшности):

1. Ошибки формализации. Неполное \ недостаточно подробное модели объекта или предметной области.

2. Ошибки решения. Некорректный \ слишком упрощенный метод построения модели.

3. Ошибка задания параметров модели (на стадии эксперимента).

Проверка адекватности и корректировки модели

Проверка адекватности модели некоторой системы заключается в анализе ее соразмерности и равнозначности системы. Адекватность нарушается из-за идеализации внешних условий и режимов функционирования; пренебрежением некоторых случайных факторов.

Простейшая мера адекватности применительно к ^ схеме калибровки:

Yмодели, Yобъекта. \deltaY - разность по модулю, либо относительная погрешность. Считают, что модель адекватна с системой, если вероятность того, что отклонение \deltaY не превысит некоторой величины \delta, больше допустимой вероятности.

Практическое использование критерия невозможно, потому что:

для проектируемых \ модернизируемых систем отсутствует информация о характеристике Y объекта

система оценивается не по одной, а по множеству характеристик

характеристики могут быть случайными величинами и функциями

отсутствует возможность априорного точного задания предельных отклонений и допустимых вероятностей

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

Выделяют следующие виды проверок:

проверка моделей элементов

проверка моделей внешних воздействий

проверка концептуальной модели

проверка способов измерения и вычисления выходных характеристика

Корректировка модели. Если по результатам проверки адекватности выявляются недопустимые рассогласования (объекта и модели), необходимо вносить изменения:

глобальные - в случае обнаружения методических ошибок в концептуальной/математической модели

локальные - связаны с уточнением некоторых параметров и алгоритмов. Выполняются путем замены моделей компонентов системы и внешних воздействий на эквивалентные, но более точные.

параметрические - изменения некоторых специальных параметров, называемых калибровочными

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

Схема взаимосвязи технологических этапов моделирования.

/*скопипастить из лекции прошлого года + фото*/

добавления: описание => содержательное описание; язык => язык формализации; стрелочка (концептуаьной модели)->[составление формального описания]; стрелочка (план эксперимента)->[проведение натурэксперимента с прототипами]; стрелочки [интерпретация]->[составление концепт],->[построение описания имитационной модели],->[испытание ИМ]; стрелка (результаты)->[интерпретация]

12.10.15 ----------

{ доклад

Стратегическое планирование в эксперименте

Дана система, нужно получить информацию о ней. Строится модель, реализуемая на ЭВМ. Учитывается ограниченность вычислительных ресурсов. Отличие от тактического: тактическое - в деталях, как проводится серия экспериментов; стратегическое - обобщает.

//Стратегическое - внешнее проектирование по отношению к объекту, сбор общей информации о нем

Главная проблема - необходимо учитывать множество факторов (параметров) - внешней среды и т.п.

Многофакторная функция реакции - описывает, как модель реагирует на входные параметры; построить функцию сложно. В ряде случаев эту проблему можно обойти, представляя эксперимент над моделью как набор экспериментов, где в каждом наблюдается только одна реакция. Зачастую переменные реакции связаны друг с другом, из-за чего такое разбиене будет неправильно. Рационально использовать интегральные оценки и функцию полезности.

Также - стохастическая сходимость результатов эксперимента. Сходимость необходимо оценивать, обрабатывать и строить доказательства. Целью эксперимента является получение характеристик функционирования системы с помощью модели. Стохастическая сходимость может быть очень медленной; при уменьшении выборки может понадобиться значительно бОльшее число повторных прогонов.

СП состоит из двух этапов - построения структурной модели. Выбирается из того как система работает физически; определяются все факторы и их уровни. Уровень - некоторая характеристика, которая тем больше чем важнее фактор и чем большее число раз их нужно пересчитывать в эксперименте. Для простых факторов принимается 1, для переменных - число уровень не меньше двух. Если фактор изменяется часто, то уровень будет большим. Нужно расмотреть систему с разных точек зрения - либо исследовать поведение системы в целом, либо сосредоточиться на решении какой-то проблемы.

Второй шаг - определение уровней, на которых следует измерять каждый фактор. Учитывается, что каждый новый уровень увеличивает затраты ресурсов ЭВМ.

На основе структурной модели строится функциональная - "что может быть сделано с учетом ресурсов". Определяет количество элементов (структурный блок, простейший эксперимент с одним фактором и уровнем измерения). Бывают полные (участвуют все элементы структурной модели) и неполные. Полная как правило не реализуется из-за ограниченности ресурсов. Цель построения - компромисс между требуемыми действиями и ресурсами.

ФМ строится на основе варьировании числа факторов, уровней, повторений экспериментов, затрат времени на прогон и ресурсов.

Стратегические цели можно поделить на два класса - задача анализа и задача синтеза (известна реакция, ищется при каких факторых будет достигнута).

}

Исследование вычислительных систем

Выделяют следующие уровни проектирования:

1. структурный

В качестве типовых матсхем - системы массового обслуживания, А-схемы, сети Петри, вероятностные автоматы

2. функционально-логический уровень проектирования

1. подуровень регистровых передач

Элемент моделирования - регистры, сумматоры; всё разделяется на комбинационные и последовательные схемы.

2. логический уровень

Моделируем работы ЛЭ, когда они синтезированы на уровне одной кремниевой подложки.

Используются цифровые автоматы, Ф-схемы

3. конструкторский

ОДУ, ДУ в частных производных.

Характерен тем, что решаются в основном задачи отвода тепла, вентиляции, оптимального размещения интегральных схем (оптимизация длины проводников) либо вопросы технологичности многослойной печатной платы.

Мы сосредотачиваемся на структурном уровне, СМО и сети Петри.

Моделирование на системном уровне

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

Коллектив пользователей - сообщество людей, которые используют вычислительную систему для удовлетворения своих нужд по обработке информации.

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

Получаем структуру вычислительной установки:

-+->коллектив пользователей -рабочаянагрузка-> вычислительная система -+->

+----------+

обратная связь

Введем понятие индекса производительности. ИП - некоторый описатель, используемый для представления производительности системы. Различают качественные и количественные индексы:

качестсвенные

мощность системы команд (RISC\CISC)

легкость использования системы (интерфейсы)

количественные

пропускная способность (объем информации, обрабатываемой за единицу времени)

время реакции (ответа) системы - время между предъявлением системе входных данных и появлением соответствующих выходных данных

коэффициент использования оборудования - отношение времени использования указанной части оборудования к интервалу времени, за который производится наблюдение.

Концептуальная модель вычислительной системы включает в себя сведения о:

выходных и конструктивных параметрах системы

её структуре

особенностях работы каждого элемента

характере взаимодействия между ресурсами

постановке прикладной задачи, определяющей цели моделирования

исходных данных, необходимых для исследования

Основные задачи:

1. определение принципов организации вычислительной системы

2. выбор архитектуры, уточнение функций вычислительной системы, их разделение на подфункции, реализуемые аппаратным или аппаратным путем

3. разработка структурной схемы (состав устройств, способы их взаимодействий)

4. определение требований к выходным параметрам ФЛУП (опускаемся от уровня вычислительной системы на уровень регистровых передач). Входные данные - команды, реализуемые в АЛУ; преобразуются в нули и единицы; запуск алгоритма разбора последовательности и выдачи результата.

Системы массового обслуживания, Q-схемы

Особенности непрерывно-стохастического подхода рассматриваем на примере использования в качестве типоой математической схемы систем массового обслуживания (queuing system).

прибор обслуживания

Накопитель Канал

-v_i->|||||||] ( )->w_i

Канал в единицу времени может ообрабатывать только одну заявку.

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

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

Неоднородный поток - задается не только вызывающими моментами, но и функцией f - набором признаков события (приоритет, тип заявки и т.п.)

Если интервалы времени между сообщениями независимы между собой и являются случайными величинами - поток с ограниченным последействием.

Поток событий называется ординарным, если вероятность, что на малый интервал времени (\inc t, примыкающий к моменту времени t) попадает больше одного события, пренебрежительно мала по сравнению вероятностью попадания на интервал ровно одного события.

Поток называется стационарным, если вероятность появления некоторого числа событий на некотором интервале \tau зависит лишь от длины этого интервала и не зависит от того, где на оси времени взят интервал.

Среднее число сообщений на участке времени \inc t в единицу времени для ординарного потока:

P_(>1)(t,\inct) + P_1(t,\inct) = P_1(t,\inct)

=>

lim_(\inct->0)(P_1(t,\inct))/(\inct)=\lambda(t) - интенсивность ординарного потока.

Для стационарного потока интенсивность не зависит от времени, =\lambda.

В общем случае в i-м приборе мы имеем поток заявок - интервалы времени между моментами появления заявок на входе канала. Это подмножество неуправляемых параметров.

Поток обслуживания - интервалы времени между началом и окончанием обслуживания заявки. Подмножество управляемых параметров.

Заявки, обслуженные каналом и заявки покинувшие i-й прибор необслуженными, образуют выходной поток Y_i, т.е. интервалы времени между моментами выхода заявок.

Также процесс функционирования прибора можно представить через состояния. Могут быть введены подмножества состояний: накопителя и канала. Состояния накопителя - пуст, обрабатывает, полностью занят. Канал - занят или свободен.

В практике моделирования, элементарные Q-схемы обычно объединяются. Если каналы различных приборов соединены параллельно, то имеет место многоканальное обслуживание; если последовательно - многофазное.

Используется оператор сопряжения, отражающий взаимосвязь элементов структуры между собой.

Различают разомкнутые и замкнутые Q-схемы. В разомкнутых схемах выходной поток не может попасть на вход, а в замкнутых схемах количество заявок постоянно.

Собственные внутренние параметры Q-схемы:

количество фаз

количество каналов в каждой фазе

количество накопителей в каждой фазе

емкость i-го накопителя. В ТМО принято:

если емкость i-го накопителя = 0 - система с потерями

если емкость стремится к бесконечности - система с ожиданием

если емкость конечна - система смешанного типа

Для задания Q-схемы также необходимо описать алгоритм её функционирования (и не один, а много - схема может реализовывать несколько алгоритмов), которые определяют набор правил поведения заявок в системе. Неоднородность заявок, отражающая процесс в той или иной реальной системе, учитывается с помощью введения классов приоритетов (статических и динамических, относительных, абсолютных).

Весь набор возможных алгоритмов поведения заявок в Q-схеме можно представить также в виде оператора алгоритмов A:

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

поток, отображающий входные заявки

поток, отображающий выходные заявки

подмножество собственных внутренних параметров

множество состояний системы

оператор сопряжения

множество алгоритмов, которые могут реализовать данный подкласс Й-схем

* * * 19.10.2015 by vlad * * *

Для получения соотношений, которые связывают характеристики, описывающие функционирование Q-схем, вводят некоторые допущения относительно входных потоков, функции распределения, длительности обслуживания запросов и самих дисциплин обслуживания.

Для математического описания функционирования устройств, развивающихся в форме случайного процесса, может быть применен математический аппарат, разработанный в теории вероятностей для так называемых «марковских случайных процессов». Случайный процесс, протекающий в сложной системе S, называется марковским, если для каждого момента времени t_0 вероятность любого состояния системы в будущем зависит только от состояния системы в настоящем (т.е. не зависит от того, когда и каким образом система перешла в это состояние). Для марковского процесса составлены уравнения Колмогорова:

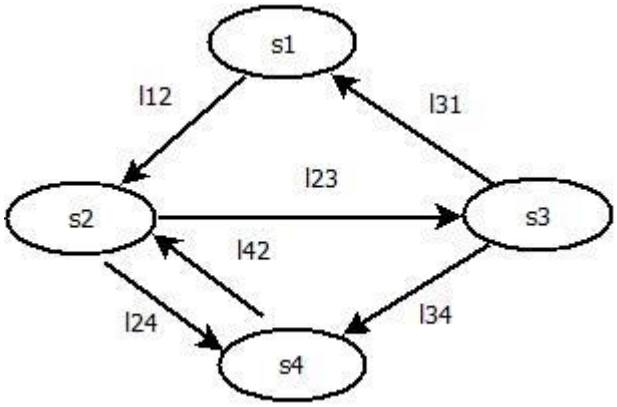
$$F = (P'(t), P(t), \lambda) = 0$$

λ – набор некоторых коэффициентов.

Для некоторого стационарного $\phi(P(t), \lambda) = 0$ имеем $P = P(\lambda)$ и $Y = Y\big(P(\lambda)\big)$ – базисная модель системы.

Задача – связать $Y(X, V, H)$ и $P(X, V, H)$, т.е. найти интерфейсную модель $\lambda = \lambda(X, V, H)$.

Нужно осуществить связь внутренних параметров модели с конструктивными. Математическая модель системы строится как совокупность базовой и интерфейсной модели – это позволяет использовать одни и те же базисные модели для решения различных задач моделирования, осуществляя настройку на соответствующую задачу с помощью изменения интерфейсной модели.



Найдем вероятность $P_1(t)$ что в момент времени t система будет находиться в состоянии s1. Придадим t малое приращение dt и найдем вероятность того, что система будет находиться в s1. Возможно два случая:

1. система уже находилась в s1 и не изменила состояния
2. система находилась в s3 и за dt перешла в состояние s1

Вероятность первого способа найдем как произведение вероятности $P_1(t)$ на условную вероятность того, что будучи в состоянии S1 система за время Δt ***не*** перейдет из него в состояние s2. Эта условная вероятность (с точностью до бесконечно малых величин высших порядков) будет равна $P_1(t)(1 - \lambda_{12}\Delta t)$ (на рисунке: l12).

Вероятность второго способа равна вероятности того, что в момент времени t система была в состоянии s3, умноженной на условную вероятность перехода 3-1:

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

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

Таким образом, получаем уравнение Колмогорова для первого состояния:

$$\boldsymbol{\frac{dP_1(t)}{dt} = -\lambda_{12}P_1(t) + \lambda_{32}P_3(t)}$$

Теперь выведем аналогичные уравнения для состояний S2, S3, S4:

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

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

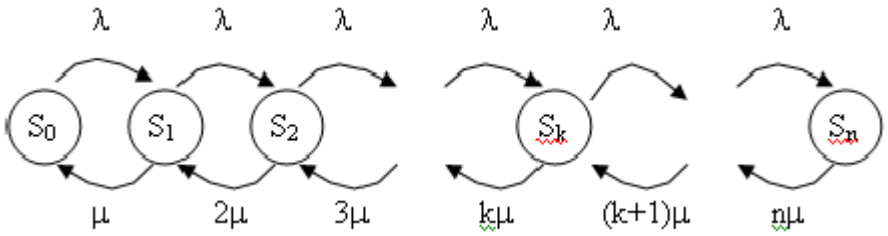
$$P_3'(t) = -\lambda_{31}P_3(t) - \lambda_{34}P_3(t) + \lambda_{23}P_2(t)$$

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

Проинтегрировав данную систему, получим вероятности состояний как функции от времени. Начальные значения будут зависеть от начального состояния (если в момент времени t=0 система находилась в состоянии S1, то начальное условие будет t_1). В систему должно быть добавлено условие нормировки: $\sum_i P_i = 1$.

Все эти уравнения могут быть построены по формальному правилу: слева – производная состояния, справа – сумма произведений плотности вероятности перехода (интенсивность стрелки) на вероятность того состояния, из которого происходит переход. Если переход происходит из самого рассматриваемого состояния, то значение берётся со знаком «минус».

Рассмотрим многоканальную систему массового обслуживания с отказами.



Введём следующие состояния: S0 – все каналы свободны; S1 – занят 1-й канал; Sk – занято k каналов, Sn – заняты все каналы. Разметим граф – проставим веса стрелок.
Слева-направо систему переводит один и тот же поток заявок с интенсивностью λ.
Справа налево: пусть было S1. Как только закончится обслуживание заявки, занимающей канал, система перейдет в S0 – поток, переводящий систему в это состояние, будет иметь интенсивность перехода μ. Если будет занято 2 канала, то интенсивность перехода (уже в S1) составит 2μ и так далее.

Уравнения Колмогорова для такой системы будут иметь вид:

$$\begin{cases} P_0' = -\lambda P_0 + \mu P_1 \\ P_1' = -\lambda P_1 + \lambda P_0 - \mu P_1 + 2\mu P_2 \\ \dots \\ P_k' = -\lambda P_k + \lambda P_{k+1} - k\mu P_1 + (k+1)\mu P_{k+1} \\ \dots \\ P_n' = -n\mu P_n + \lambda P_{n-1} \end{cases}$$

Предельные вероятности состояний P_0 и P_n характеризуют установившийся режим работы системы массового обслуживания при $t \to \infty$:

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

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

$(\lambda/\mu) = \rho$ – среднее число заявок, приходящих в систему за среднее время обслуживания одной заявки.

Зная вероятности всех состояний, можно найти следующие характеристики системы:

Вероятность отказа – вероятность, что все n каналов заняты: $p_{\text{отк}} = p_n = \frac{\rho^n}{n!} p_0$

Относительная пропускная способность – вероятность, что заявка будет принята к обслуживанию: $q = 1 - p_n$

Среднее число заявок, обслуженных в единицу времени: $A = \lambda * q$

В простейшем случае, если время ввода/вывода информации по каждой задаче мало по сравнению со временем её решения, то можно принять время решения за $t = \frac{1}{\mu}$.

26.10.15 ----------

Реальные процессы весьма часто обладают последействием и поэтому не являются марковскими. Тем не менее, иногда при исследовании таких процессов удается воспользоваться методами, разработанными для марковских цепей. Наиболее распространенными являются метод разложения случайного процесса на фазы (метод псевдосостояний) и метод вложенных цепей Маркова. Подробно рассмотрим первый и упомянем о втором.

Метод разложения:

Состояния, потоки переходов из которых являются не марковскими, заменяются эквивалентной группой некоторых фиктивных состояний, потоки переходов их которых уже являются марковскими. Условие статистической эквивалентности реального состояния и фиктивных могут в каждом конкретном случае выбираться по-своему.

Например - в качестве критерия можно выбрать min\integral_t1^t2(\lamba_iЭкв(\tau)-\lambda_i(\tau))dt. Здесь лямбда_иэкв - эквивалентная интенсивность перехода в i-й группе переходов, обладающей интенсивностью

\lambda_i(\tau). За счет расширения числа состояний системы некоторые процессы удается точно свести к марковским. Созданная таким образом новая система статистически эквивалентна или близка к реальной системе. Её можно исследовать обычными методами с помощью аппарата теории марковских цепей.

К числу процессов, которые введением фиктивных состояний можно точно свести к марковским, относятся процессы, формализовать которые можно с помощью закона Эрланга. В случае потока Эрланга k-го порядка, интервал времени между соседними событиями представляет собой сумму k независимых случайных интервалов, распределенных по показательному закону. Сведение ПЭ к-го порядка к пуассоновскому осуществляется введением k псевдосостояний. Интенсивности переходов между псевдосостояниями равны соответствующему параметру ПЭ.

Полученный таким образом эквивалентный процесс является марковским, т.к. интервалы времени нахождения в различных состояниях подчиняются показазтельному закону распределения.

Пример: некоторое сложное устройство S, которое выходит из строя с интенсивностью \lambda - причем поток отказов Пуассоновский. После отказа устройство восстанавливается - время восстановления распределено по закону Эрланга 3 порядка: f(t)=0.5\mu(\mu*t)^2*exp(-\mu*t). Найти предельные вероятности возможных состояний системы.

Система может принимать два возможных состояния: ЫО (исправно) и ЫЗ(отказало и восстанавливается. Переход 0-1 осуществляется под воздействием пуассоновского потока, 1-0 - Эрланга. Если есть ПЭ - случайное время восстановления можно представить в виде суммы 3 (порядок) случайных временных интервалов, распределенных по показательному закону с интенсивностью \mu: t=t1+t2+t3.

Состояние ЫЗ заменяем последовательной цепочкой из 3 псевдосостояний.

Граф - 4 состояния: ЫО, Ы11, Ы12, Ы13. 11,12,13 - всё состояние ЫЗ, разбито на псевдосостояния. Интенсивность 0-1 = \lambda, 11-12 = \mu, 12-13 = \mu, 13-0 = \mu.

Для установившегося состояния необходимо составить уравнения Колмогорова, и к ним добавить два доп.условия - условие эквивалентной замены состояния ЫЗ: p1=p11+p12+p13 и условие нормировки: p0+p1=1.

{ -\lambda p0 + \mu p13 = 0

{ -\mu p11 + \lambda p0 = 0

{ -\mu p12 + \mu p11 = 0

{ -\mu p13 + \mu p12 = 0

=>

{ p1 = p11 + p12 + p13

{ p0 + p1 = 1

{ -\lambda p0 + \mu p13 = 0

{ -\mu p11 + \lambda p0 = 0

{ -\mu p12 + \mu p11 = 0

{ -\mu p13 + \mu p12 = 0

=>

{ p11 = (\lambda p0) / \mu

{ p12 = (\mu p11) / \mu

{ p13 =

=> p11 = p12 = p13 = 1/3 p1 = 1/3 (1-p0)

p0 = (\mu p11) / \lambda

p11 = 1/3 (1 - (\mu p11) / \lambda)

3\lambda p11 = \lambda - \mu p11

p11 (3\lambda + \mu) = \lambda

p11 = 1 / (3\lambda + \mu)

p0 = 1 - 3\lambda/(3\lambda + \mu); p1 = 3\lambda / (3\lambda + \mu)

Метод вложенных цепей маркова:

Вложенные марковские цепи образуются следующим образом - в исходном случайном процессе выбираются такие случайные (по времени t_k) процессы, в которых значения процесса образуют марковскую систему. Моменты тк как правило являются случайными и зависят от свойств исходного процесса. Дальше обычными методами эти моменты используются для решения уравнений.

Частный случай - полумарковский случайный процесс; случайный процесс с конечным\счетным множеством состояний, если заданы вероятности перехода системы из одного состояния в другое (p_ij) и __распределение__ времени пребывания процесса в каждом состоянии, например в виде функции распределения или плотности распределения случайной величины.

Метод монте-карло - метод статистических испытаний:

Для произвольных потоков событий, переводящих систему из состояния в состояние, аналитические решения получены только для отдельных частных случаев. ММК используется, когда аналитику нельзя получить.

Вместо того чтобы описывать случайные явления, производится "розыгрыш", моделирование случайного явления с помощью некоторой процедуры, дающей случайный результат. Произведя такой розыгрыш большое число раз и собрав статистический материал, получаем множество реализаций случайного явления.

Возьмём единичный квадрат и посчитаем площадь произвольной фигуры в нем. Генерируем равномерно распределенные X,У, получаем кучу точек, считаем количество попавших в фигуру.

Суть метода:

1. любым способом получаем два числа, Xi, Yi, подчиняющихся равномерному распределению случайных чисел на отрезке [0..1]. Числа определяют координаты

2. анализируем принадлежность сгенерированной точки некоторой поверхности; если принадлежит - ++счетчик1, иначе - ++счетчик2.

Процедура генерации двух случайных чисел с заданным законом распределения и проверка принадлежности точки произвольной поверхности повторяется N раз. Дальше вычисляем площадь (случайная величина): количество попавших / количество сгенерированных.

Точность метода: \sqrt(1/n). Преимущество метода статистических испытаний - универсальность, обуславливающая возможность всестороннего статистического исследования объекта.

Однако для реализации этой возможности нужны достаточно полные статистические сведения о параметрах элементов, входящих в систему. Недостатки: большой объем вычислений.

Способы получения случайных чисел

1. Аппаратный

2. Табличный

3. Алгоритмический.

Аппаратный. Случайные числа как правило вырабатываются специальной электронной приставкой, генератором случайных чисел, служащей в качестве одного из внешних устройств компьютера. Реализация этого способа не требует дополнительных вычислительных операций - необходима только операция обращения к внешнему устройству.

[источник шума] |__шум___

[ключевая схема] |_прямоуг.участок__

[формирователь импульсов] |_наложение__

[пересчетная схема] |_значения t_i, t_i1_

КС - вырезает из шума нужный отрезок.

Случайное число - разница между ti и t_(i+1).

Табличный - СЧ формируются в виде таблицы и помещаются во внешнюю\оперативную память. Иногда формируют специальный файл.

Алгоритмический - рассмотрим потом более подробный

Способ Достоинства Недостатки

Аппаратный Запас чисел неограничен; в ОП не занимает места; небольшой объём вычислений. Требуется периодическая проверка на случайность; нельзя воспроизвести последовательность многократно; используется спецустройство; нужно обеспечивать стабильность (защиту от внешних значений), требует время на обращение.

Табличный Требуется однократная проверка на случайность; можно воспроизводить последовательность. Запас чисел ограничен; занимает место в ОП, требует время на обращение.

Алгоритмический Однократная проверка; многократное воспроизведение последовательности; места в ОП почти не занимает; не требуется внешнее устройство\файл. Запас чисел последовательности ВСЕГДА ограничен периодом; затраты машинного времени.

Алгоритмические способы ГПСЧ

Выделение значения дробной части многочлена первой степени (Yn = A*n + b). Если H пробегает значения всего натурального ряда чисел, то Yn выглядит весьма хаотично. Карл Якоби доказал, что при рациональном коэффициенте A множество Y конечно, а при иррациональном - бесконечно и всюду плотно в интервале 0..1. Для многочленов большей степени задача была решена Германом Вейлем. Критерий - равномерность распределения любой функции от натурального ряда чисел ("эргодичность" - среднее по реализации псевдослучайных чисел равно среднему по всему их можностью с вероятностью 1). Недостаток: в компах нет иррациональности, при попытке замены - короткий период.

(ФонНейман) Каждое последовательное число - образуется возведением предыдущего в квадрат и отбрасывание цифр с обоих концов

g_n1 = (K g_n + C) mod M. Наибольший период: при k = 3+8i или 5+8i (комплексные числа). Фарсайт доказал, что последовательности таки нехорошие.

Метод Фибоначчи - на одноименных числах и натуральной арифметике.

Наилучший способ - смешивать алгоритмы.

ДЗ: свойства функции\плотности распределения случайных величин; графики всех случайных величин.

02.11.15 ----------

Алгоритмический способ получения последовательности ПСЧ

Для получения значений СВ из последовательности случайных чисел с заданным законом распределения обычно используют одно или несколько значений равномерно распределенных случайных величин. Псевдослучайные равномерно распределенные случайные числа получаются в компьютере программным способом с помощью некоторого рекуррентного соотношения (i-е значение получается либо из i-1го либо из группы предыдущих) путем определенного алгоритма, состоящего как правило из арифметических и логических операций.

Программа на Фортране для генерации равномерно распределенной последовательности [0..1]:

```
subroutine randum(ix, iy, rn)
      iy = ix * 1220703125
      if( iy) 3, 4, 4
3:    iy = iy + 2147483647 + 1
4:    rn = iy
      rn = rn * 0.4656613e-9
      ix = iy
      return
end
```

Обращение:

call randum(ix, iy, yfl),

ix - число, которое при первом обращении должно содержать нечетное целое, состоящее менее чем из 9 цифр.

iy - полученное случайное число, используемое при обращении к последующим процедурам

третий параметр - полученное равномерно распределенное число в интервале [0..1]

На Паскале:

```
var
      ei: integer
      xr: 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 = 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 := 2;
                exit;
            end;
            s := s + w;
        end; //for
        s := s * 1.54919;
        rand := (sqr(s) - 3.0) * s * 0.01 + s;
    end;
```

Обращение:

```
r := rand(0);
for i := 1 to 200 do
    writeln( rand(2):12:8 );
```

Здесь при n=0 происходит "настройка" программы, при n=1 - равномерное распределение и при n=2 - гауссово.

Для имитации равномерного распределения на интервале [а..б] используется обратное преобразование функции плотности: (x-a)/(b-a)=R, где р - равномерно распределенная СВ [0..1]

x = a + (b-a)*r

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 СЧ с заданным законом распределения.

F(t) = \integral_(-\infty)^(t)(f(x))dx = R

Метод основан на теореме, утверждающей, что СВ X, принимающая значения, равные корню ^ уравнения, имеет плотность распределения f(x). R - равномерно распределенная [0..1].

$1 - \exp(-\lambda x) = R$

$x = (-1/\lambda) \ln(1 - R)$

Распределение Пуассона. Относится к числу дискретных (таких, при которых переменная может принимать только целочисленные значения, включая нуль. С мат.ожиданием и дисперсией $\lambda > 0$. Для генерирования пуассоновских переменных используют метод Точер, в основе которого лежит генерирование случайных значений некоторой переменной r_i, равномерно распределенной на интервале [0..1], до тех пор, пока не станет справедливым следующее соотношение:

$\Pi_{i=0}^{x}(r_i) \geq \exp(-\lambda) > \Pi_{i=0}^{x+1}(r_i)$

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

Пример на законе Эрланга (λ, k). При вычислении случайной величины воспользуемся тем, что результат может быть получен прореживанием потока Пуассона k раз. Достаточно получить k значений случайной величины, распределенной по показательному закону, и усреднить их.

$1/k * \sum_{i=1}^{k}((-1/\lambda)\ln(1-R_i)) = -1/(kx)\sum_{i=1}^{k}(\ln(1-R_i))$

Нормально распределенная случайная величина может быть получена как сумма большого числа случайных величин, распределенных по одному и тому же закону распределения с одними и теми же параметрами. СВ X, имеющая нормальное распределение с математическим ожиданием и среднеквадратичным отклонением может быть получена по следующей формуле:

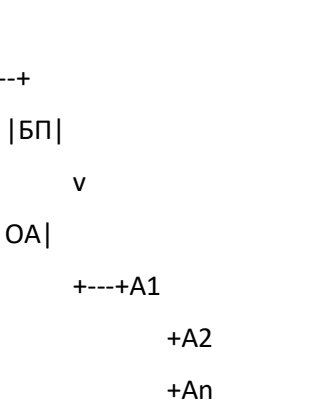
$x = \sigma x \sqrt{12/n} \sum_{i=1}^{n}(R_i-n/2) + Mx$

Для практического применения принимают n=12.

Методика построения программной модели

Для разработки программной модели (вычислительной системы) исходная система должна быть представлена как стохастическая система массового обслуживания (или сеть). Это объясняется следующим: информация от внешней среды поступает в случайные моменты времени. Длительность обработки различных типов информации может быть (в общем случае) различной. Таким образом, внешняя среда является своего рода генератором сообщений, а комплекс вычислительных средств.

Обобщённая структурная схема вычислительной системы:



(ИсточникИнформации, БлокПамяти, ОбслуживающийАппарат, Абоненты)

Источники информации выдают на вход буферной памяти независимо друг от друга сообщения. Закон появления сообщений - произволен, но должен быть известен заранее (в случае последовательности чисел - проверить какую-нибудь гипотезу). В память сообщения записываются "в навал" и выбираются по-одному в осблуживающий аппарат по принципу FIFO.

Длительность обработки одного сообщения в обслуживающем приборе в общем случае может быть также случайной, но закон обработки также должен быть задан.

Быстродействие обслуживающего аппарата ограничено, поэтому на входе системы возможны сложения данных, ожидающих обработки.

Для каждого источника данных необходимо создать собственную программную модель (ПМ генератора). Точно также ПМ пойдёт для памяти и обслуживающего аппарата, плюс для каждого абонента.

Необходимо добавить два компонента: программу сбора статистики (по каждому программному компоненту иметь статистику сколько сообщений, производительность и т.п.) и управляющую программу (на данной системе реализовать множество различных алгоритмов - когда и в какое время обращаться\активизировать тот или иной программный модуль, функцию и т.п.).

Моделирование потока сообщений

Поток сообщений обычно моделируется моментами появления очередного сообщения в потоке. Текущий момент времени появления вычисляется по формуле t_i = \sum_(k=1)^(n)(T_k) + T_i. T_i - интервал между сообщениями.

Тип распределенияВыражение для расчета времени T_i

Равномерное [а..б] T_i = A + (B-A) * R

Экспоненциальное T_i = 1/\lambda ln(1-R)

Нормальное Гауссово T_i = \sigma_x (\sum_(i=1)^(n)(R_j) - 6) + M_x

Эрланга T_i = 1/(k \lambda) \sum_(j=1)^(k)(ln(1-R_j))

09.11.15 ----------

Моделирование работы обслуживающего аппарата

Программа, имитирующая работу обслуживающего аппарата, вырабатывает случайные отрезки времени, соответствующие длительности обслуживания требований. Требования от источника обрабатываются по нормальному закону с параметрами: матожидание и сигма; длительность обработки i-го требования t_обработки = \sigma_x (\sum_(j=1)^(12)(R_j) - 6) + M_x

[s = 0]

/j = 1,12\

[r_j = f()]

[s = s + r_j]

\ /

[t_обработки = Mx + (s - 6) * \sigma_x]

[t_i = t_i + t_обработки]

Здесь r - случайное число с равномерным законом распределения, Mx - матожидание, \sigma_x - среднестатистическое отклонение от заданного закона, t_обр - время обработки очередного сообщения, t - время освобождения аппарата.

Моделирование работы абонентов

Абонент может рассматриваться как обслуживающий аппарат, поток информации на который поступает от процессора. Для моделирования работы абонентов необходимо вырабатывать длительности обслуживания требований. Кроме того, абонент может сам быть источником заявок.

Эти заявки могут имитироваться с помощью генератора сообщений (распределенных по определенному закону). Таким образом, абонент моделируется как генератор или обслуживающий аппарат.

Моделирование работы буферной памяти

Буферная память должна производить запись и считывание чисел, выдавать сигналы переполнения или отсутствия данных, в любой момент времени располагать сведениями о количестве находящихся в памяти требований.

Саму запоминающую среду можно моделировать в простейшем случае одномерным массивом (лучше списком), размер которого определяет объём памяти. Каждый элемент массива может быть свободен (=0) либо занят - в этом случае в качестве эквивалента требования ему присваивается значение времени появления этого требования.

Анализ признака запись/чтение

Запись: анализ на переполнение

Переполнение: сообщение в блок статистики (переполнение)

Иначе:

запись информации по текущему адресу.

изменение текущего адреса +1

чтение: анализ на наличие

пусто: сообщение в блок статистики (нет)

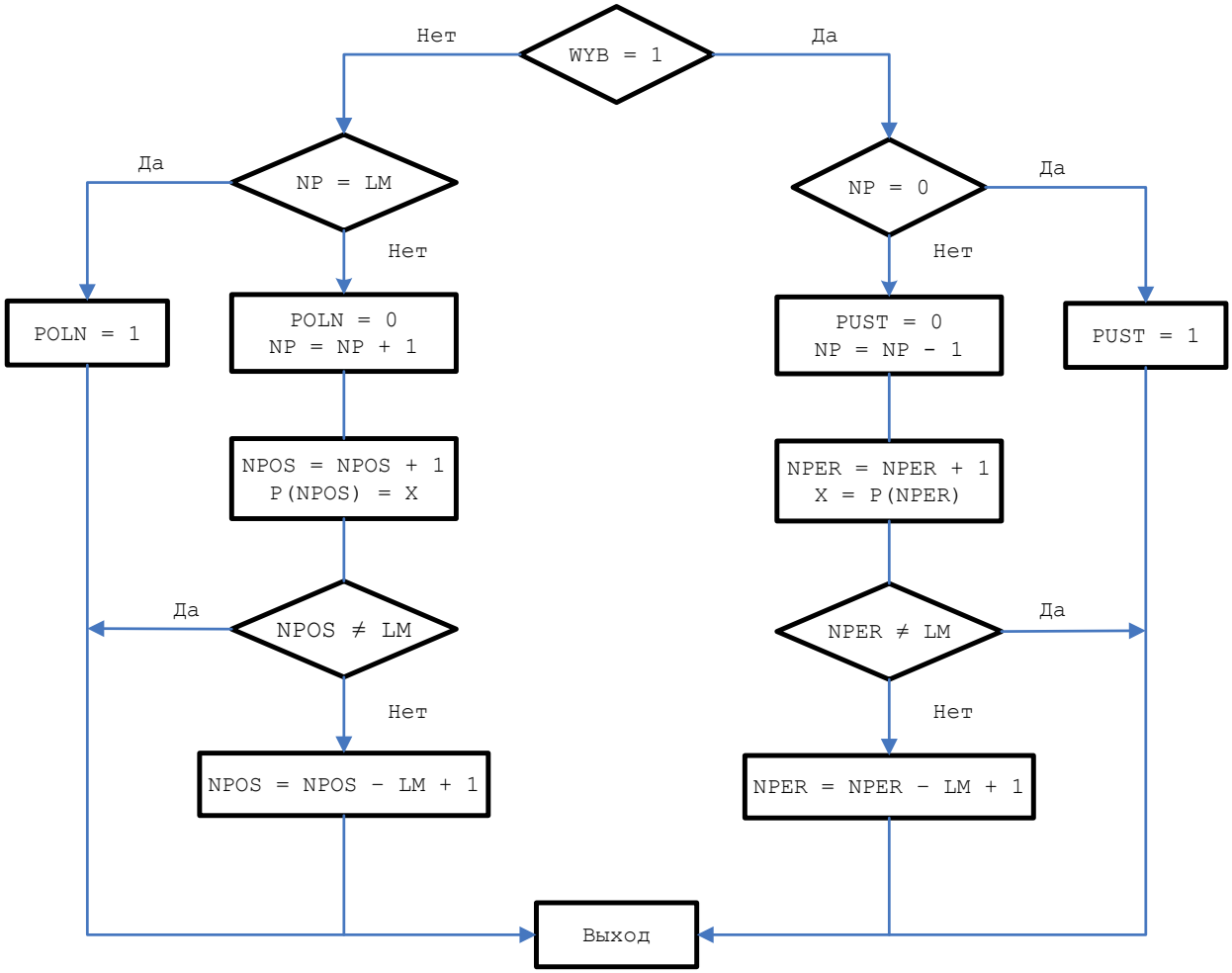
иначе:

чтение по текущему адресу

изменение текущего адреса -1

выход

Алгоритм реализации работы буферной памяти:



WYB – признак обращения к буферной памяти (1 – режим выборки, 0 – режим записи)

P(LM) – массив, в котором хранятся сообщения

LM – объём памяти

NP – число сообщений, хранящихся в памяти

NPOS – номер последнего сообщения, поступившего в память $NPOS = \begin{cases} NPOS + 1, npos < lm \\ NPOS - lm + 1 \end{cases}$

NPER – номер первого сообщения из имеющихся в памяти $NPER = \begin{cases} NPER + 1, NPER < LM \\ NPER - LM + 1 \end{cases}$

POLN – признак переполнения

PUST – признак отсутствия сообщений в памяти

X – некоторая входная ячейка памяти\

Разработка программы сбора статистики

Задача блока статистики заключается в накоплении численных значений, необходимых для вычисления статистических оценок данных, по каждому объекту, который у нас встречается. При моделировании работы простейшей СМО 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 было сгенерировано источником информации.

Дальше времена суммируются по каждому из сообщений, и делится на общее число.

Необходимо суммировать количество сообщений через небольшие промежутки времени и делить сумму на число суммирований.

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

Чтобы определить вероятность потери сообщений в системе, нужно разделить количество потерянных сообщений на сумму всех сообщений в системе.

Управляющая программа имитационной модели

Имитирует алгоритм работы. Реализуется по двум принципам:

Принцип дельты т. Заключается в последовательном анализе состояний всех блоков в момент времени t+\delta t по заданному состоянию блоков на момент времени t. При этом новое состояние определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов, задаваемых распределениями вероятностей.

В результате этого анализа принимается решение, какие общесистемные события должны имитироваться в программной модели на данный момент времени.

Основной недостаток принципа – значительные затраты машинного времени на реализацию моделирования системы, а при недостаточно малой дельте – появляется опасность **пропуска** отдельных событий в системе – промоделируется совсем не та система, что ожидалась. Достоинство – равномерное продвижение времени без скачков.

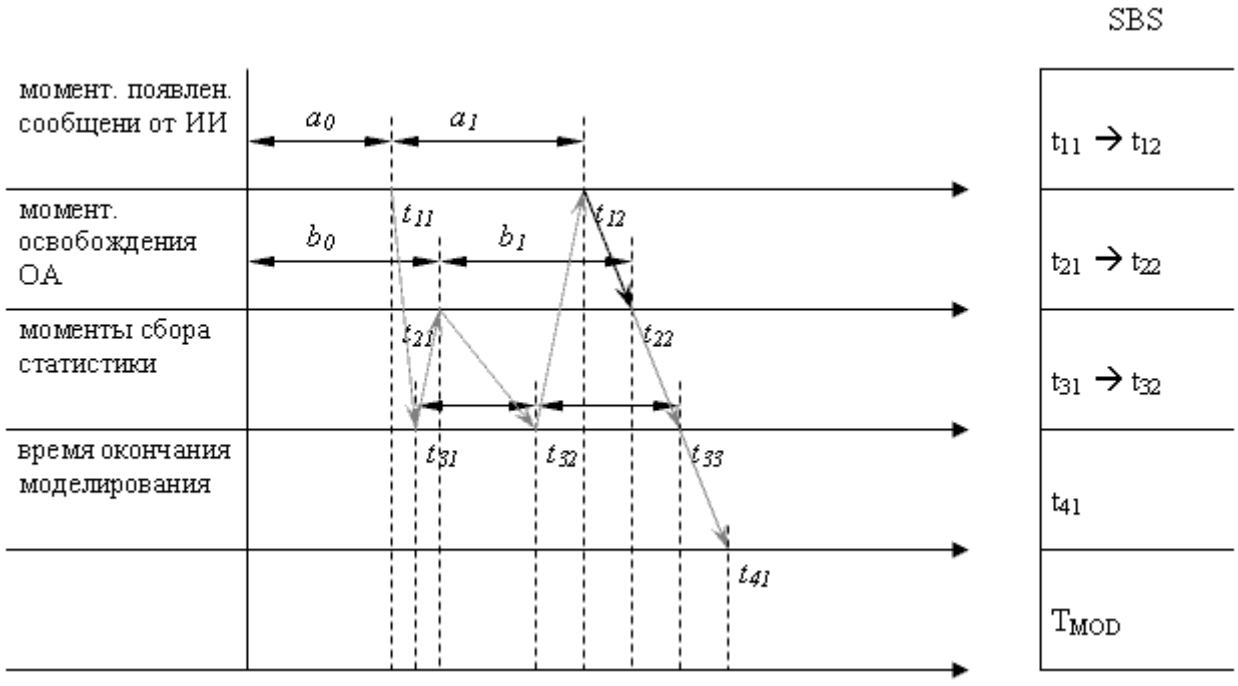
Событийный принцип. Характерное свойство моделируемых систем обработки информации – состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с: моментами времени поступления сообщений в систему; временем окончания решения задач; временем возникновение аварийных сигналов и т.п. Моделирование и продвижение текущего времени в системе удобно проводить с использованием событийного метода. Состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события.

Моменты наступления следующих событий определяются минимальным значением из списка будущих событий. Список представляет собой совокупность моментов ближайшего изменения состояния каждого из блоков системы.

Достоинство: не пропустим ни одного события

Недостаток: при большом количестве событий (сложная система) список необходимо просматривать постоянно (можно держать сортированным).

Схема событийного принципа:



Последняя графа – текущее время.

t11,t12 – моменты появления сообщений на выходе источника информации

b1 – время обслуживания первого сообщения

t3i – момент сбора статистики

t41 – момент окончания моделирования

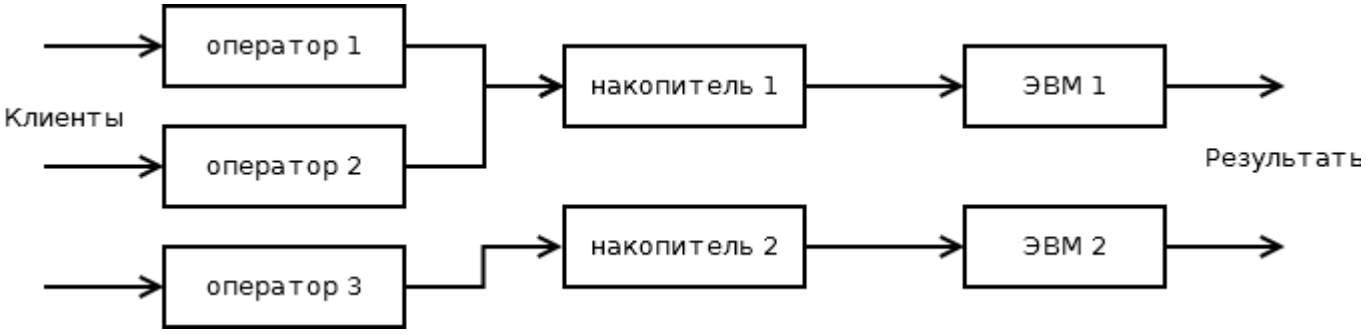
Методика реализации событийного принципа

1. Для всех активных блоков (порождающих события) заводят свой элемент в списке (массиве)
2. Заносим в СБС время ближайшего события от любого активного блока
3. Активизируем программный имитатор источника и вырабатываем псевдослучайную величину a0, определяющую момент появления первого сообщения (t11) и заносим её в СБС.
4. Активизируя имитатор процессора (ОА) вырабатываем псевдослучайную величину b0, которая определяет t21.
5. Момент времени t31 – момент времени первого сбора статистики (равен стандартному шагу). Заносится в СБС, как и время окончания моделирования t41
6. Подготовительная часть закончена, начинаем алгоритм:
    a. В СБС определяем минимальное значение и его порядоковый номер
    b. Реализуем события, порождаемые блоком, в соответствии с номером – реализуем событие, связанное с появлением сообщение от источника информации. Само сообщение записывается в буферную память и с помощью имитатора вырабатывается момент появления следующего сообщения (t12)
    c. Это время заносится в соответствующую ячейку СБС (на место т11)
    d. Вновь организуем поиск минимума в СБС
    e. Реализуется событие 31
    f. Вырабатываем момент времени t32 (новое время сбора статистики)
    g. Повторяем до тех пор, пока минимальное время не станет равным t41.

16.11.15 ----------

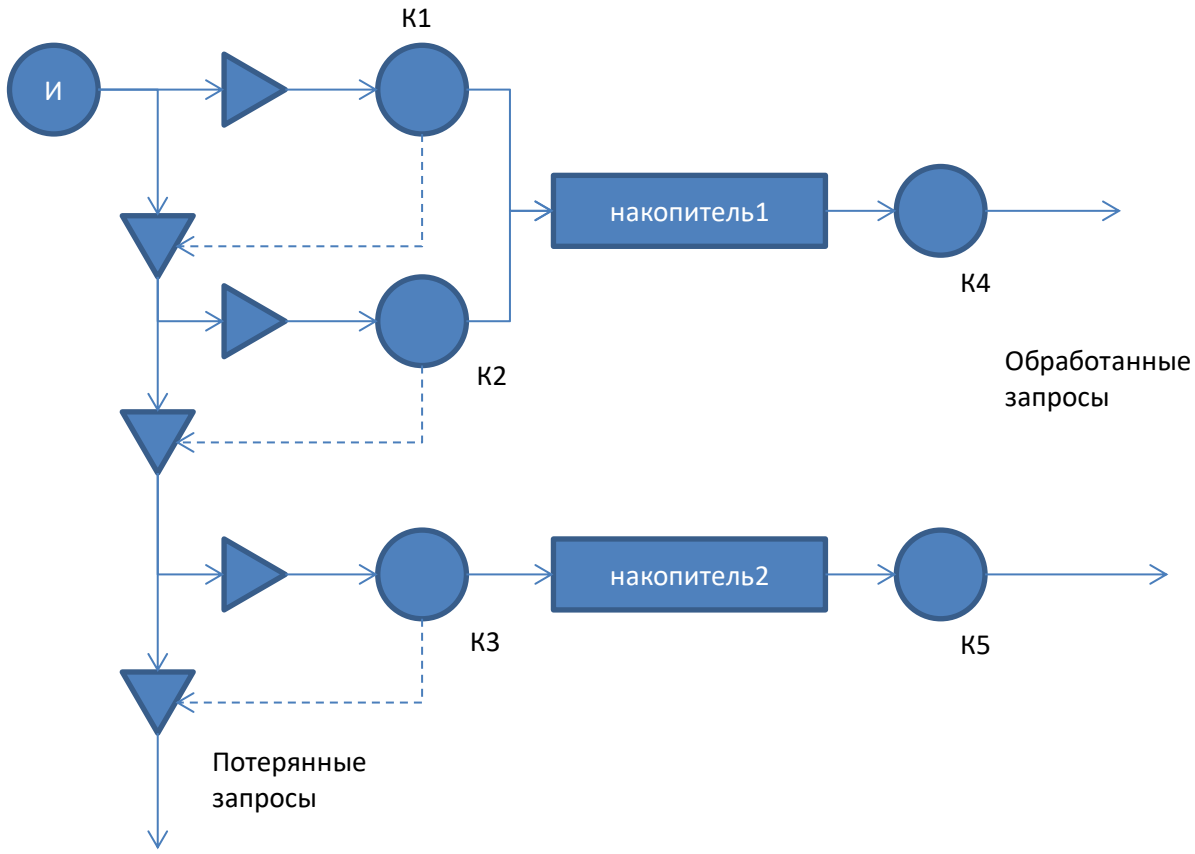
^ - концептуальная модель

Структурная модель:



В процессе взаимодействия возможно: режим нормального обслуживания (клиент выбирает одного из свободных операторов, отдавая предпочтение по меньшему номеру); режим отказа в обслуживании клиента (все операторы заняты).

В терминах СМО:



Необходимо составить уравнения моделирования и выделить переменные.

Эндогенные: время обработки задания i-м оператором; время решения задания на j-й машине

Экзогенные: число обслуженных клиентов N0 и число клиентов получивших отказ N1

Вероятность отказа: $\frac{N_0}{N_0+N_1}$

За единицу системного времени выберем 0.01 минуты (исходя из качества получения псевдослучайной последовательности на GPSS).

Комбинированный метод

Используются циклические иерархические списки. В каждой ячейке – время обработки. Для каждой ячейки можно построить соответствующий циклический список. От n-го уровня доходим до 1-го, когда происходят все события. На нем каждая ячейка – текущее время, будущее время. На этом уровне происходит моделирование с шагом Δt, на высших – событийное.

Сети Петри

Сеть Петри – математическая абстракция, один из формализмов – на практике занимает положение между цифровым автоматом и вероятностным. ПО любой сложности можно формализовать графом (автомат с памятью \ автомат мили-мура). Остаются нюансы, которые с помощью автоматного подхода не моделируются (проявляется на уровне регистровых передач) –событию нельзя придать какие-либо характеристики, *помимо* его собственно свершения (почему совершилось и т.п.).

Сети Петри позволяют ввести состояния, внутри которых используются «фишки». Моделирование производится с помощью «запуска» сети – формально, фишки передвигаются по графу. Можно получить цветную

сеть с использованием разноцветных фишек. Можем получить модель для определения работоспособности программы и поиска тупиковых ситуаций.

Сети Петри можно представлять с двух точек зрения:

1. теория множеств (абстрактная математика)
2. графовое представления

Обыкновенные сети Петри

Математическая модель дискретных динамических систем (параллельных программ, операционных систем, компьютеров и компонентов, вычислительных сетей), ориентированная на анализ и синтез таких систем. Даёт обнаружение блокировок, тупиковых ситуаций, узких мест при выполнении заданий, автоматический синтез параллельных программ, синтез компонент компьютера и т.д.

Формально, **сеть Петри** – кортеж $PN = (\theta, P, T, F, M_0)$

$\theta = (0,1,2, \dots)$ – множество дискретных *моментов времени*

$P = (p_1, p_2, \dots, p_{\boldsymbol{n}})$ – непустое множество элементов сети, называемых вершинами-**позициями**

$T = (t_1, t_2, \dots, t_{\boldsymbol{m}})$ – непустое множество элементов сети, называемых вершинами-**переходами**, $P \cap T = \emptyset$

$F: (P \times T) \times (T \times P) \to (0,1,2, \dots, k, \dots)$ – функция инцидентности, k – кратность дуги

$M_0 \to P(0,1, \dots)$ – начальная маркировка позиций

Функция инцидентности может быть представлена в виде $F = F^P \cup F^t$ и задает два отображения: $F^p(p,t) = P \times T \to \{0,1,2, \dots\}$ и $F^t(t,p) = T \times P \to \{0,1,2, \dots\}$. В первом случае для каждой позиции указываются связанные с ней переходы (с учетом кратности); во втором – для каждого перехода указываются связанные с ним позиции. В общем случае эти две функции могут быть представлены матрицами инцидентности:

$$\mathrm{F^p} = \begin{bmatrix} f_{11}^p & \cdots & f_{1m}^p \\ \vdots & \ddots & \vdots \\ f_{n1}^p & \cdots & f_{nm}^p \end{bmatrix}$$

Здесь, столбцы определяют переходы $t_1, \dots, t_m$, а строки – позиции $p_1, \dots, p_n$.

Аналогично можно представить матрицу инцидентности для второй функции:

$$F^t = \begin{bmatrix} f_{11}^t & \cdots & f_{1n}^t \\ \vdots & \ddots & \vdots \\ f_{m1}^t & \cdots & f_{mn}^t \end{bmatrix}$$

Здесь наоборот, столбцы определяют позиции, а строки – переходы.

Из вершины-позиции $p_i \in P$ ведет дуга в вершину-переход $t_j \in T$ тогда и только тогда, когда $f_{ij}^p > 0$. В этом случае говорят, что t_j – выходной переход позиции p_i. Множество всех позиций p_k, для которых t_j является выходным переходом будем обозначать P^j. В нашем случае:

$$P^j = \left\{ p_k : f_{ij}^p > 0 \right\}$$

Аналогично, из каждой вершины перехода $t_j \in T$, дуга ведёт в вершину-позицию $p_i \in P$ тогда и только тогда, когда $f_{ji}^t > 0$. При этом говорят, что p_i – выходная позиция перехода t_j. Множество всех переходов t_j, для которых p_i – выходная позиция, будет обозначаться t^i.

При $f_{ij}^p > 0, \text{имеем } f_{ji}^t > 0$. Эти величины называются **кратностью дуг**.

Каждая позиция $p_i \in P$ может содержать некоторый целочисленный ресурс $\mu(p) \geq 0$. Его часто отображают соответствующим числом точек (фишек) внутри позиции. Вектор $M = [\mu_1, \mu_2, \dots]$ называется маркировкой (разметкой) сети. Маркировки связаны с позициями, $M: P \to \{0,1,2, \dots\}$. **Начальная маркировка** определяет стартовое состояние сети Петри.

Динамика поведения моделируемой системы описывается в терминах функционирования сетей Петри. Сеть функционирует в дискретном времени, в асинхронном режиме переходя от одной маркировки к другой. Смена маркировок (начиная с маркировки M_0, начального состояния) проходит в результате срабатывания переходов

сети: $t_j \in T$ может сработать при маркировке M, если для всех $p_i \in P^j$ выполняется следующее условие: $\mu_i(\theta) \geq f_{ij}^{p(\theta)}$, т.е. если каждая входная позиция для данного перехода содержит как минимум столько фишек, какова кратность ведущей к t_j дуги.

В результате срабатывания перехода t_j в момент времени θ, маркировка $M(\theta)$ сменяется на маркировку $M(\theta + 1)$ по следующему правилу:

$$\boldsymbol{\mu_i(\theta + 1) = \mu_i(\theta) - f_{ij}^{p}(\theta) + f_{ji}^{t}(\theta)}$$

Здесь i изменяется от 1 до N, j – от 1 до M (в соответствии с матрицей инцидентности), $i \in P^j, j \in T^i$. 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 переходов, то срабатывает любой из них. Функционирование сети останавливается, если при некоторой маркировке (тупиковая маркировка) ни один из её переходов не может сработать.

При одной и той же начальной маркировке, сеть Петри может порождать различные последовательности срабатывания её переходов (в силу недетерминированности её функционирования). Эти последовательности образуют слова в алфавите T. Множество всех возможных слов, порождаемых сетью Петри, называется языком сети Петри. Две сети Петри эквивалентны, если порождают один и тот же язык.

В отличие от конечных автоматов, в терминах которых описывается глобальное состояние системы, сети Петри акцентируют внимание на локальных событиях (переходах), локальных условиях (позициях) и локальных же связях между оными. Поэтому в терминах сетей Петри, более адекватно, чем с помощью автоматов, моделируется поведение распределенных асинхронных систем.

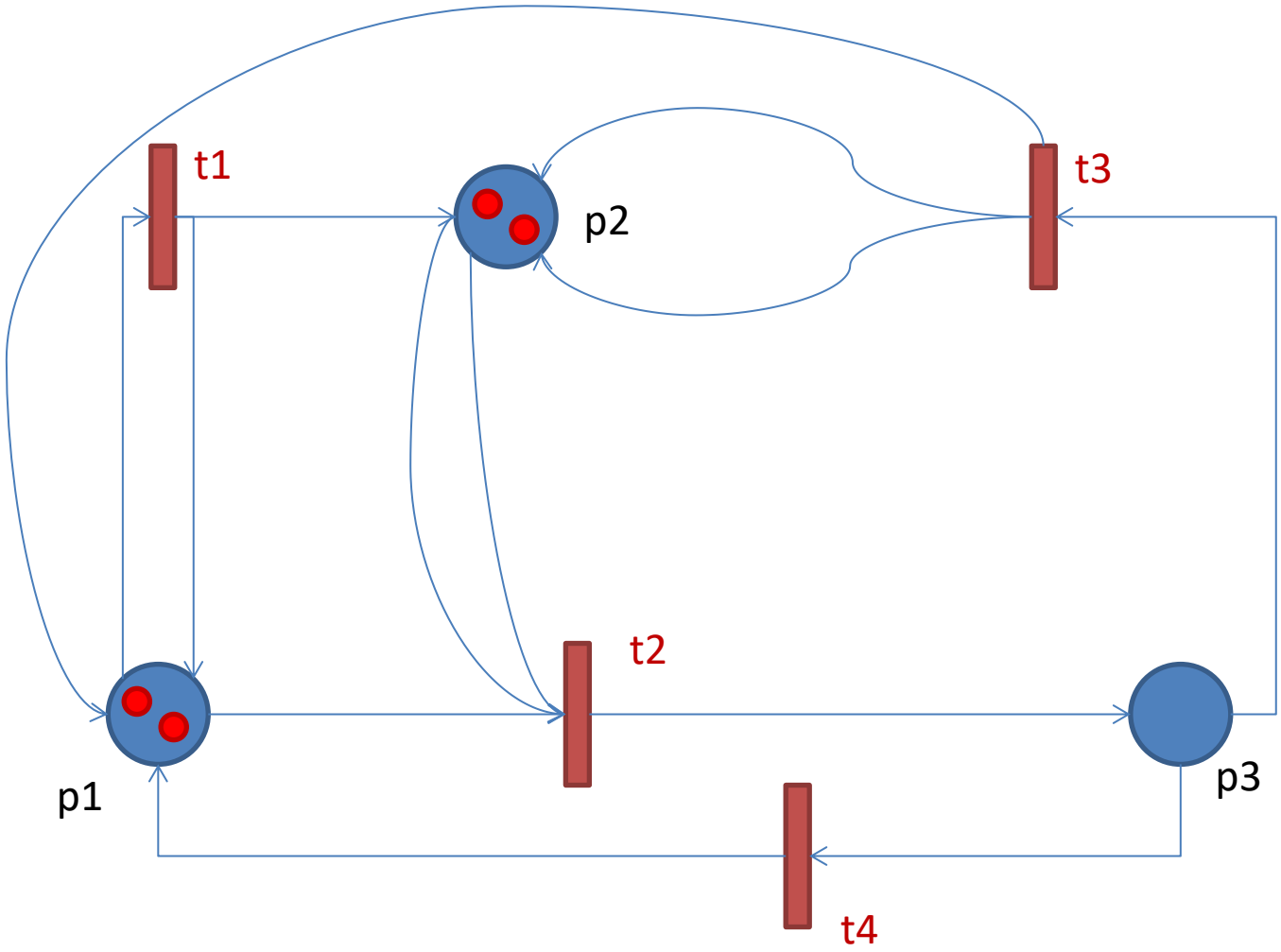
Графы сетей

Теоретико-графовым представлением сети Петри является двудольный ориентированный мультиграф. Этот граф содержит:

- позиции(места), кружки
- переходы, вертикальные планки
- ориентированные дуги, стрелки, соединяющие позиции с переходами и наоборот.

Кратные дуги обозначаются несколькими параллельными (можно указывать число).

Благодаря наличию кратных дуг, сеть Петри есть мультиграф. Благодаря двум типам вершин, граф называется двудольным. Поскольку дуги имеют направления, граф является ориентированным.



Матрица инцидентности F^p:

	t1	t2	t3	t4
p1	**1**	**1**	**0**	**0**
p2	**0**	**2**	**0**	**0**
p3	**0**	**0**	**1**	**1**

Матрица F^t:

	p1	p2	p3
t1	**1**	**1**	**0**
t2	**0**	**0**	**1**
t3	**1**	**2**	**0**
t4	**1**	**0**	**0**

Начальная маркировка: $M_0 = [2,2,0]$

23.11.15 ----------

Пространство состояний сети Петри

Определяется маркировкой сети и обозначается как E^n, где n – количество позиций сети. Изменение в состоянии, вызванное запуском перехода, определяется функцией перехода δ, или функцией следующего состояния. Когда эта функция применяется к маркировке M и переходу t_j (если он разрешен), то получается новая маркировка: $M' = \delta(M, t_j)$. ($M(\theta+1) = M(\theta) - f_{ij}^p + f_{ji}^t$). Эта новая маркировка получается извлечением фишек из позиций p_i таких, что F_{ij} (матрица инцедентности П) != 0 (должны быть точки внутри позиций) (Маркировка $\mu_i \geq f_{ij}^p$) и перемещением фишек в позицию p_k, т.е. $f_{jk}^t \neq 0$.

Процесс создания новых маркировок продолжается до тех пор, пока в сети Петри при данной маркировке существует хоть один разрешенный переход. Если же при некоторой маркировке $M(\tau)$ (τ – дискретный момент времени) ни один переход не разрешен, то такая маркировка называется тупиковой.

При выполнении (оживлении, реализации) сети Петри получается:

1. последовательность маркировок $(M_0, M_1, \dots)$

2. последовательность запущенных переходов $(t_{j_0}, t_{j_1}, \dots)$

Эти две последовательности связаны следующим соотношением: $M(\theta+1) = \delta(M(\theta), t_{j_\tau})$. Если в результате запуска перехода при некоторой маркировке M образуется новая маркировка M', то говорят, что M' достижима из M.

Множество достижимости

$R(PN, M)$ (PetryNet, Markings) – множество всех M_k, достижимых из M.

Маркировка M' принадлежит множеству достижимости, если существует какая-либо последовательность запусков переходов, изменяющих M на M'. Иначе, множество достижимости для сети Петри с маркировками M есть наименьшее множество маркировок, определенных следующим образом:

А) $M' \in R(PN, M)$;

Б) если $M' \in R(PN, M)$ и $M'' = \delta(M', t_j)$ для некоторого $t_j \in T$, то $M'' \in R(PN, M)$.

Вернемся к примеру ^ с сетью. При начальной маркировке M0=[2,2,0] могут сработать переходы: т1 в M1'=[2,3,0] и т2 в M1''=(1,0,1).

(photo)

$\theta = 0$	$\theta = 1$	$\theta = 2$	$\theta = 3$	$\theta = 4$	
M0=[2,2,0]	T1: [2,3,0]	T1: [2,4,0]	T1: [2,5,0]	T1: [2,6,0]	
	T2: [1,0,1]				

Постоянные переходы по Т1 приведут к бесконечному накоплению фишек в позиции 2.
Каждая из полученных маркировок порождает новые. В результате получается **дерево маркировок**, в котором дуги – переходы. В этом дереве могут встречаться повторяющиеся маркировки – в таком случае, дальнейшее построение дерева ведётся только для одной из них. Если выделить путь по дугам графа маркировок, начинающийся в вершине М0 и заканчивающийся в различных вершинах М', и выписать подряд все встречающиеся символы переходов, то полученное слово образует последовательность срабатываний сети.

Совокупность этих т-слов представляет собой свободный язык сети Петри. При этом нулевой компонент языка – некоторый пустой символ λ, соответствующий начальной маркировке.

Основные свойства сетей Петри

1. Свойство **ограниченности**. Позиция p_i в сети называется ограниченной, если для любой достижимой в сети маркировки М существует такое К, что $\mu_i \leq k$.

Сеть называется ограниченной, если все её позиции ограничены. ^ сеть – неограниченна, потому что есть неограниченный рост μ_2.

2. Свойство **безопасности**. Сеть называется безопасной, если при любой достижимой маркировке $\mu_i \geq 1\ \forall i = \overline{1,n}$ (н –число позиций). Следовательно, в безопасной сети вектор маркировок состоит только из нулей или единиц (является двоичным словом).
3. Свойство **консервативности**. Сеть называется консервативной, если сумма фишек во всех позициях остается постоянной при работе сети. $\sum_{i=1}^{n} \mu_i = const$.
4. Свойство **живости**. Переход t_j в сети ПН называется потенциально живым, если существует достижимая из М0 маркировка М', при которой t_j может сработать. Если t_j является потенциально живым при любой достижимой в сети маркировке, то он называется живым.

Переход t_j, не являющийся живым при начальной маркировке М0, называется мёртвым при этой маркировке. Маркировка М0 в этом случае называется $t_{j\ \text{тупиковая}}$ для перехода. Если маркировка М0 является $t_{j\ \text{туп.}}$ для всех j, то она называется тупиковой маркировкой. Другими словами, при тупиковой маркировке не может сработать ни один переход.

Если рассматривать *дерево* всех маркировок, тупик будет листовой вершиной. Переход называется устойчивым, если никакой другой переход не может лишить его возможности сработать при наличии для этого необходимых условий.

Последовательность маркировок $M_1, \dots, M_p$, в которой последующая маркировка через функцию переходов может быть выражена из предыдущей, образует цикл в том случае, когда $M_0 = M_p$. Фактически, каждому циклу соответствует последовательность слов свободного языка сети.

Некоторое обобщение сетей Петри

Ингибиторные сети (IPN, ИСП). Сети, для которых в функцию инцидентности добавляется ещё один компонент: $F = F^p \cup F^T \cup F^1$, в котором отображаются позиции и переходы только из множества 0-1 ($F^1 = P \times P \to \{0,1\}$. То есть, функция дополнена специальной функцией, которая вводит ингибиторные дуги для тех пар, для которых элемент матрицы инцидентности = 1.

Ингибиторные дуги связывают только позиции с переходами и на рисунках отображаются не стрелками, а черными кругами (не -> а -•). Кратность этих дуг всегда равна единицы.

$$\left(\mu_i \geq f_{ij}^{p}\right) V \left(\mu_k f_{kj}^{1} = 0\right)$$

Вводится дополнительное условие: позиция п катая соединенная с переходом т жтая ингибиторной дугой не должна содержать фишек.

Сеть с приоритетами. При определении СП отмечалась недетерминированность работы. Если имеется возможность срабатывания нескольких переходов, то срабатывает любой из них. При моделировании реальных систем, могут сложиться ситуации, когда последовательность срабатываний необходимо как-то регламентировать. Вводится множество приоритетов (относящихся к переходам) 0,1,2,3… 0 выше чем 1 и так далее. Приписав к каждому из переходов соответствующее значение приоритета пр_j. Тогда правило срабатывания модифицируется снова: если на некотором такте работе сети имеется возможность для срабатывания нескольких переходов, то срабатывает тот из них, который имеет наивысший приоритет.

П1(.) Т1-> пр1=2 ( )П3

X

П2(.) Т2-> пр2=1 ( )п4

Первым должен сработать переход т2, поскольку его приоритет выше.

Сети со случайным срабатыванием переходов. Если имеется возможность срабатывания нескольких переходов, их приоритет можно задавать вероятностями срабатывания. При этом вводится нормировочное ограничение – сумма всех вероятностей должна быть 1. Исходная маркировка М0 приведет на следующих шагах работы сети к набору маркировок $(M_1, \ldots, M_c)$, каждая из которых будет помечена своей вероятностью. Отождествив маркировки с состоянием сети и предположив что вероятности не зависят от работы сети в предыдущие такты, получаем марковскую сеть.

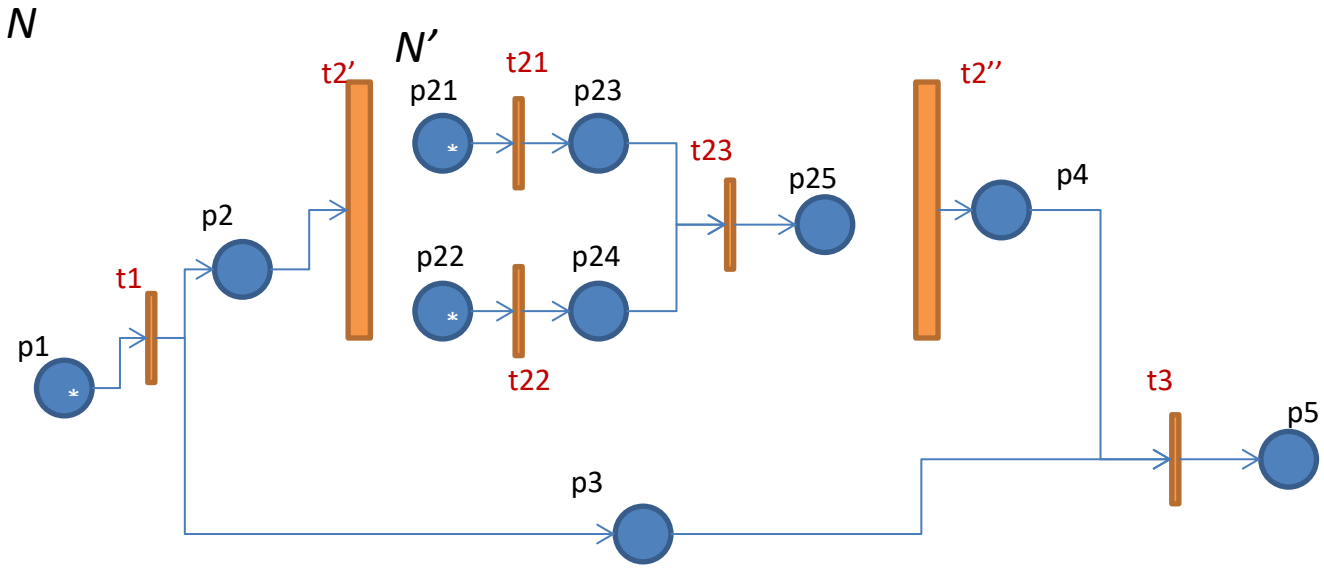
В ^ случае (с Х), если в М1 срабатываем с вероятностью П, получаем 0 0 1 0, в М2 – 0 0 0 1.

30.11.15 ----------

Иерархические сети Петри

Представляют собой многоуровневые структуры, в которых выделяются сети различного уровня. Они позволяют моделировать иерархические системы. В отличие от обыкновенных сетей, в иерархических имеется *два типа переходов*:

- **Простые**. Ничем не отличаются от рассмотренного ранее.
- **Составные**. Содержат внутри себя сеть Петри более низкого иерархического уровня. Формально они состоят из входного (головного) и выходного (хвостового) переходов. Между ними и находится более низкая сеть, которая в свою очередь также может быть иерархической.



Иерархическая сеть N’ функционирует как обыкновенная сеть Петри, переходя от одной маркировки к другой и обмениваясь фишками, в том числе между сетями различного уровня. Исключение составляет правило работы составных переходов.

Срабатывание составных переходов является не мгновенным событием, а составным действием – говорят о работе составного перехода, а не его срабатывании. На каждом шаге дискретного времени составной переход может находиться в одном из двух состояний:

- Пассивном
- Активном

Начальное состояние всех переходов – пассивное. Составной переход может быть активирован в некоторый момент θ, если до этого он был пассивен и имеются условия для срабатывания его головного перехода. При этом производится изменение маркировки в сети верхнего уровня (по обычным правилам) и одновременно запускается работа в сети, находящейся внутри составного перехода.

Во время работы внутренней сети, работа сети верхнего уровня блокируется. Сеть нижнего уровня работает с учетом своей начальной маркировки до тех пор, пока все её переходы не станут пассивными (не смогут дальше работать). После этого происходит срабатывание хвостового перехода и изменение маркировки верхней сети. Составной переход возвращается в пассивное состояние, а в сети нижнего уровня восстанавливается начальная маркировка.

Запустим ^ сеть.

$\theta = 0$: 1 0 0 0 0

T1

$\theta = 1$: 0 1 1 0 0

T2'

$\theta = 2$: 0 0 1 0 0

T2'' 1 1 0 0 0

T21

0 1 1 0 0

T22

0 0 1 1 0

T23

0 0 0 0 1

T3

[1 1 0 0 0]

$\theta = 3$: 0 0 1 1 0

T3

0 0 0 0 1

На шаге $\theta = 2$ происходит работа составного перехода T2' для сети N’ следующим образом. T2' – запуск сети более низкого иерархического уровня. Окончание работы сети и восстановление маркировки – срабатывание перехода T2''.

Данный процесс хорошо отображает функционирование какой-либо подпрограммы.

Цветные сети Петри

Colored Petri Net. Опираются на понятие мультимножества.

$$CPN = \{\theta, \Sigma, P, T, A, N, C, G, E, I\}$$

Здесь:

- θ – множество дискретных моментов времени, в которые происходит функционирование сети
- Σ – конечное множество (непустых типов), называемое цветами
- P – конечное множество позиций
- T – конечное множество переходов, которое можно представить как в обычной сети. Правила срабатывания гораздо сложнее.
- A – конечное множество дуг, связывающих между собой позиции и переходы. В отличие от обыкновенных сетей, дуги задаются не матрицей, а множеством P to T.
- N(a) – некоторая узловая функция, которая для каждой дуги $a \in A$ указывает её исходный и конечный узел.
- C(p) – цветовая функция, определяющая множество типов цветов, разрешенных для каждой позиции.
- G(t) – блокировочная (спусковая) функция, описывающая дополнительные условия, которые должны быть выполнены для срабатывания перехода $t \in T$. Эта функция представляет из себя предикат, составленный из переменных , принадлежащих типам цветов.
- E(a) – функция, задающая выражения на дугах. Функция для каждой дуги a определяет мультимножество, состоящее из элементов, описанных в множестве цветов.
- I(p) – функция инициализации цветной сети. По аналогии с обыкновенными сетями, задает начальную маркировку (разметку) сети M_0. Здесь для каждой позиции п эта функция указывает цветовое мультимножество.

$$\forall p \in P: \left(m(p) \geq \sum_{\forall a \in AP} E(a) \right) \wedge (G(t) = true)$$

$$m(p, \theta + 1) = m(p\theta) - \sum_{\forall a \in AP} E(a) + \sum_{\forall a \in AP} E(a)$$

В первом случае происходит поэлементное сравнение мультимножества, являющегося маркировкой позиции *p*, с мультимножеством, которое помещает дуги, ведущие от *p* к *t*. Суммирование мультимножества E(a) связано с те, что два узла могут быть соединены несколькими дугами и суммирование производится по всем таким дугам. Вторая скобка – отсутствие условий блокировки.

Вторая формула описывает изменение маркировки узла *p* при осуществлении одного шага работы цветной сети.

Класс сетей Петри строго мощнее класса конечных автоматов и строго менее мощен, чем класс машин Тьюринга.

Классы ингибиторных сетей и сетей с приоритетами строго мощнее класса сетей Петри и равномощны классу машин Тьюринга.

Класс раскрашенных сетей Петри (при конечном количестве цветов) равномощен классу сетей Петри.

Класс самомодифицируемых сетей эквивалентен классу ингибиторных сетей и сетей с приоритетами.

Моделирование дискретных систем с помощью сетей Петри

Функционирование систем с независимыми элементами (как правило) – аппаратное и программное обеспечение, телекоммуникации, физические процессы и так далее. При описании сетей выделяют два понятия:

- События. Любое действие в системе. В сетях Петри моделируются переходами. Непримитивное событие – сложный переход, иерархическая сеть.
- Условия. Предикат или логическое описание системы (принимающее значение истина/ложь). Моделируется позициями. Различают пред- и пост-условия.

- Пред – условия до срабатывания переходов
- Пост – условия после срабатывания переходов.

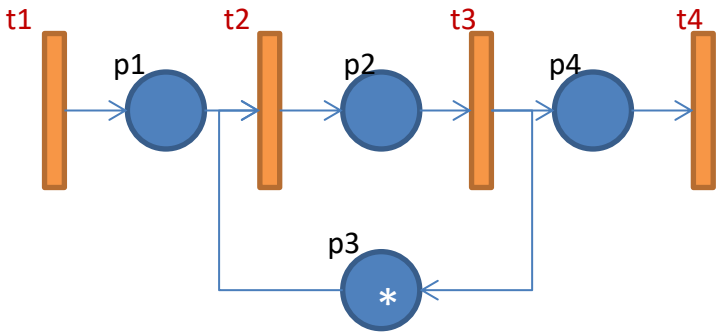
Ещё одна особенность сетей – одновременность. Если при своей работе некоторые переходы t_i и t_j не влияют друг на друга, то в словарь языка сети Петри входят как слова, начинающиеся с t_i, так и слова, начинающиеся с t_j.

Два перехода t_i и t_j находятся в конфликте, если запуск одного из них блокирует запуск другого.

Так вот, СМО.

Условия:

- p1 – задание ждёт обработки
- p2 – задание обрабатывается
- p3 – процессор свободен
- p4 – задание ожидает вывода
- t1 – задание помещается во входную очередь
- t2 – начинает выполняться
- t3 – завершает выполняться
- t4 – задание выводится



Начальная маркировка M_0 0 0 1 0 соответствует состоянию, когда система свободна и заявки на обслуживание отсутствуют. При срабатывании перехода t1 от внешнего источника поступает задание и получается маркировка M_1 1 0 1 0. При этом может сработать переход t2, что означает начало обслуживания задания и приводит к маркировке M_2 0 1 0 0. Затем может сработать переход t3, то означает окончание обработки задания и освобождение системы – переход к M_3 0 0 1 1. Переходы t1 и t4 могут работать независимо от переходов t2 и t3, моделируя поступления и вывод заданий.

07.12.15 ----------

Языки моделирования

При исследовании процесса функционирования сложных дискретных систем, язык имитационного моделирования является проблемно-ориентированным средством, позволяющим определять процесс функционирования СДС в терминах и категориях, базирующихся на методологии и технологии процесса имитации, на формализации системы в виде типовых математических схем элементов СДС и их взаимодействия.

В общем случае язык имитационного моделирования должен удовлетворять следующим требованиям:

- Обеспечивать простоту и легкость программирования при описании СДС для компьютерной реализации

- Концептуальная направленность языка на исследование СДС, обеспечивая модульный иерархический подход к анализу и контролю правильности функционирования СДС.

Архитектура языка ИМ в общем случае основывается на концепции взаимосвязи элементов.

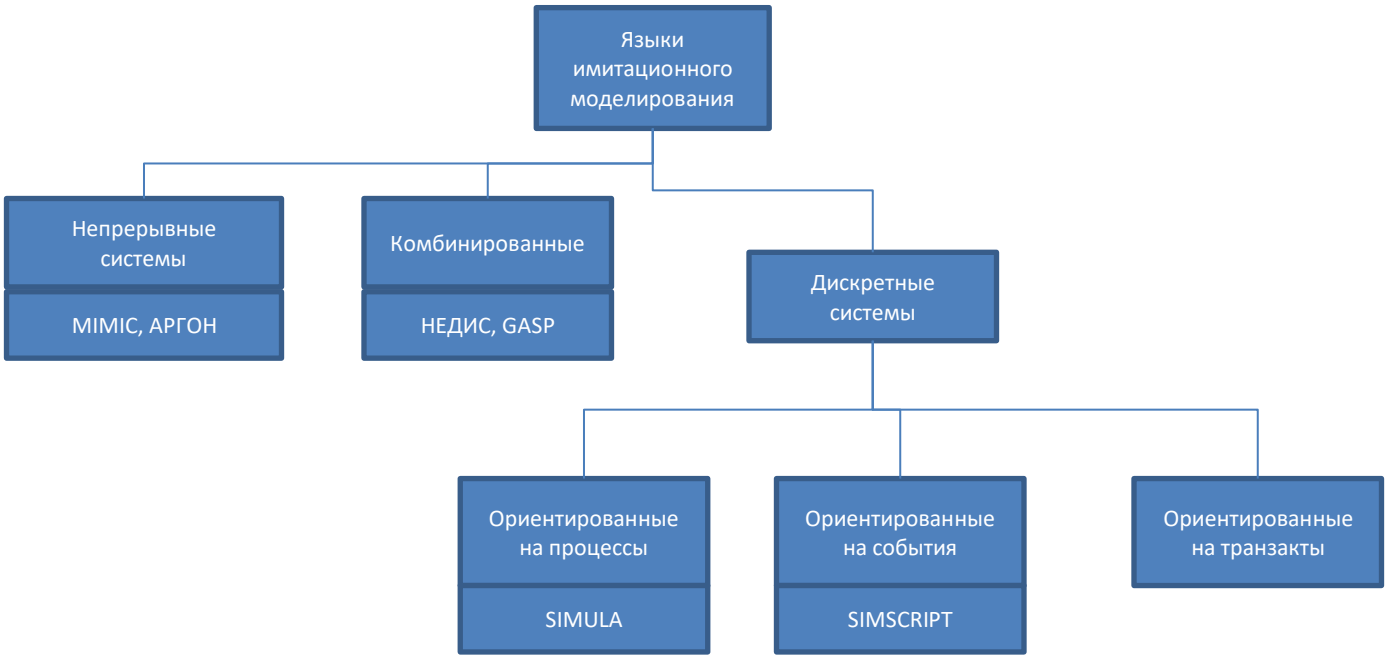
Объекты моделирования СДС описываются и определяются с помощью некоторых атрибутов языка, которые взаимодействуют с процессами (упорядоченная во времени совокупность активностей, адекватно отображающая процесс функционирования СДС). Процессы реализуются конкретными условиями, определяющими логическую основу и последовательность взаимодействия.

Язык ИМ должен:

- Обеспечивать визуальное программирование модели СДС с использованием стандартных и оригинальных графических компонентов
- Совмещать языковые средства ИМ с интеллектуальными системами поддержки принятия решений
- Обеспечивать возможности применения анимации при отображении процессов функционирования
- Интегрироваться с CASE-технологиями
- Создавать многоуровневые модели систем в рамках иерархического анализа

Существуют три классических подхода:

- Событийный (и различные вариации), рассматривался ранее.
- Сканирование активностей. Моделируемая система рассматривается с точки зрения её состояния. Описывается набор активностей, зависящих от состояния системы. При выполнении условия реализуется заданный набор действий по изменению состояния системы.
- Процессно-ориентированный. Моделируемая система рассматривается с точки зрения взаимодействия её процессов. Каждый процесс описывается своим алгоритмом.



При непрерывности выходных параметров (D-схемы) выделяют события двух типов: зависящие от состояния и зависящие от времени.

Важнейший аспект при рассмотрении языков имитационного моделирования

Формализация, описание динамики моделируемого объекта. Полагаем, что любая работа в системе совершается путем выполнения активностей; активность – минимальная единица работы. Её рассматривают как единый дискретный шаг. Таким образом, активность является единым динамическим объектом, указывающим на совершение единицы работы.

Процесс – логически связанный набор активностей.

Активность проявляется в результате совершения событий.

Событие – мгновенное изменение состояния некоторого объекта системы.

Рассмотренные объекты, активности, процессы и события являются конструктивными элементами для динамического описания поведения СДС. На их основе строятся языки моделирования таких систем. 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. В общем случае, построение модели состоит из решения двух задач:

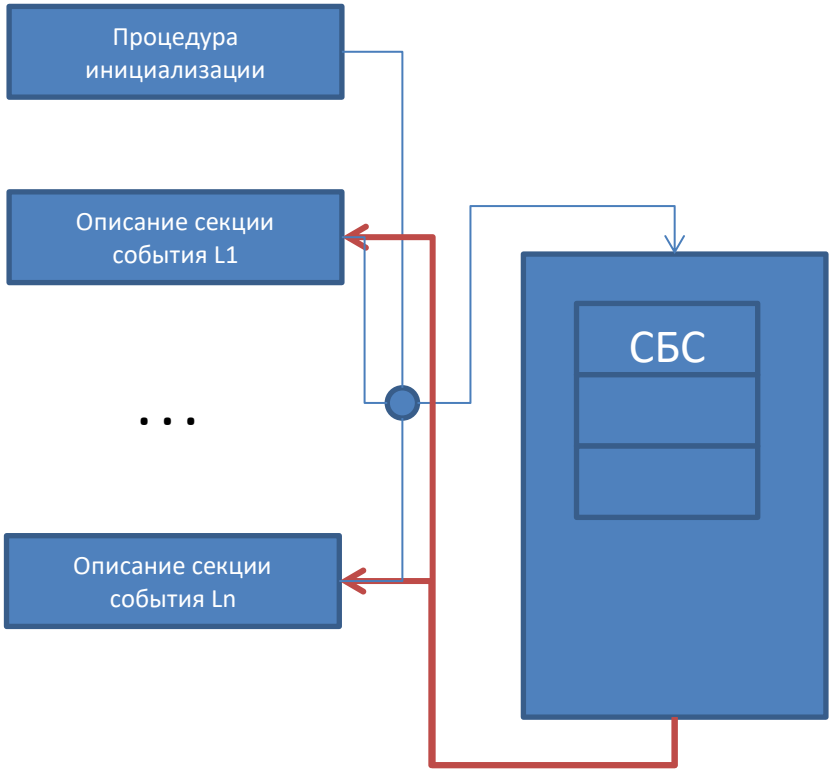
1. Первая сводится к описанию правил, задающих виды процессов, происходящих в системе.
2. Вторая заключается в указании значений атрибутов таких процессов или задании правил генерации этих значений.

При этом система описывается на определенном уровне детализации в терминах множества описания процессов. Каждое описание включает множество правил и условий возбуждения активностей.

Так как модель служит для отображения временного поведения системы, то язык моделирования должен отображать это время и иметь средства отображения оного. В реальной системе совместно выполняются несколько активностей, принадлежащих как связанным, так и не связанным процессам, а имитация должна соответствовать алгоритму \ множеству алгоритмов. Модель системы можно рассматривать как модель описания активностей, событий или процессов.

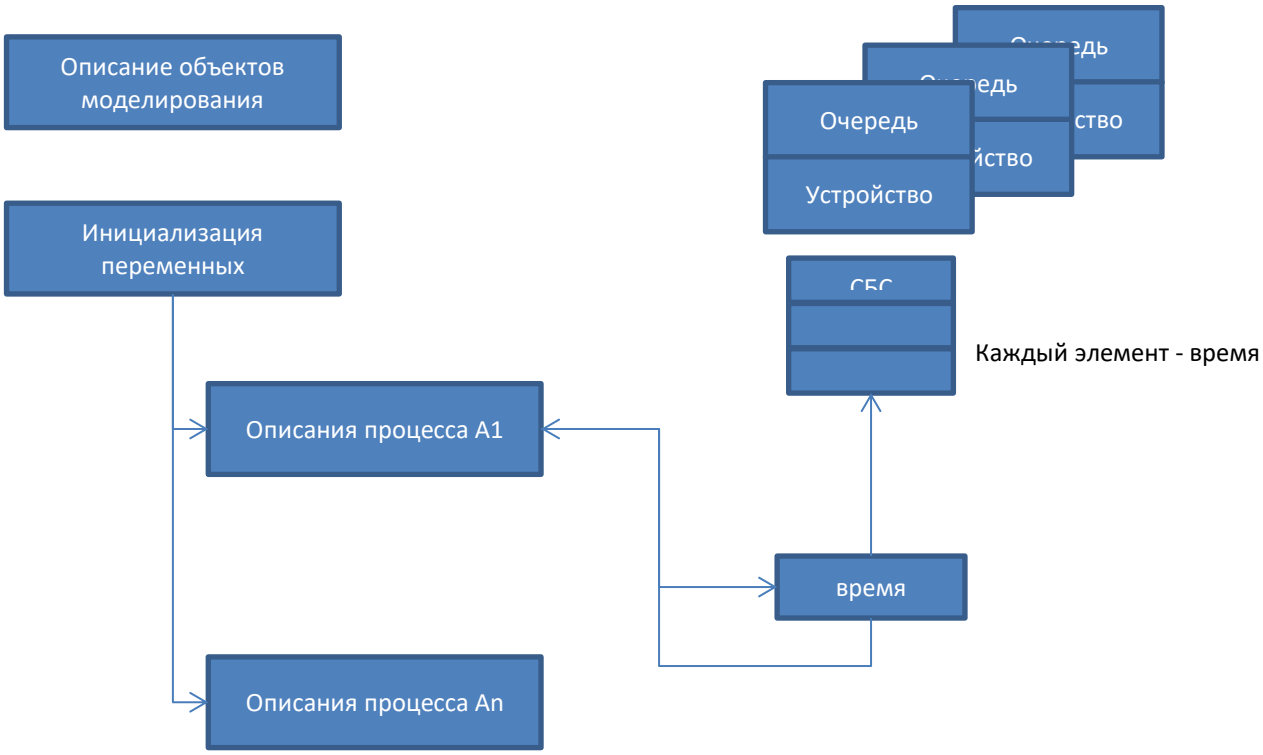
Языки, ориентированные на события

Моделирующая программа организована в виде совокупности секций событий (процедур событий), каждая из которых состоит из набора операций, которые в общем случае выполняются после завершения какой-либо активности. Синхронизация происходит с помощью списка будущих событий.



Языки, ориентированные на процессы

Моделирующая программа организована в виде набора описания процессов (каждый описывает сразу класс процессов). Описание устанавливает атрибуты и активности – происходит параметрическая настройка. Синхронизация операций во времени реализуется также с помощью списка будущих событий (СБС).



^ на примере языка SIMULA

Языки моделирования

Самый мощный – класс языков на базе Симулы. Простота применения – здесь выигрывает ГПСС. Предпочтение пользователей – язык на базе СимСкрипта.

ARENA.Предназначена для моделирования производственных и бизнес-процессов. Реализует дискретные модели. Имеется набор стандартных блоков и процессов при использовании графических компонент. Обеспечивает поддержку анимации.

EXTENT. В основном ориентирована на моделирование экономических процессов, стратегического планирования. Реализует дискретные модели. Имеются блоки функциональных преобразований при использовании графических компонент для визуального моделирования.

GPSS-world (General Purpose System Simulation). Предназначена для моделирования систем массового обслуживания, транспортных потоков, производственных линий. Реализует дискретные модели. Недостаток – не обеспечивает в должной степени поддержку анимации.

Группа систем IThinkAnalyst. Системы, предназначенные в основном для моделирования финансовой деятельности, реинжиниринга

MatLab + SIMULINK. Многофункциональная система моделирования информационных и управляющих систем.

Система РДО (ресурсы-действия-объекты). Заточена под моделирование производственных процессов.

AnyLogic. Многоцелевая система моделирования сложных систем широкого назначения. Моделируются как дискретные, так и непрерывные системы.

Возможности программного обеспечения

Наличие анимации и динамической графики. Анимация используется для представления сути ИМ для руководителя. Также отладка программы и определение неправильности модели.

Два основных типа анимации: совместная (осуществляется во время прогона ИМ) и раздельная.

Статистические возможности.

Наличие разнокалиберных генераторов случайных чисел.

Использование объектных технологий.

Преимущества объектно-ориентированного моделирования:

1. Обеспечивает возможность повторного использования объектов и удобство их параметрической настройки.
2. Реализуется иерархический подход к моделированию.
3. Упрощает изменения модели.
4. Предоставляет возможность разработки модели несколькими коллективами.

Недостаток – проблема обучения такому подходу.

14.12.15 ----------

General Purpose System Simulation

Общецелевая система моделирования. Весь пакет построен в предположении, что моделью сложной дискретной системы является описание её элементов и логических правил их взаимодействия в процессе функционирования системы. Для определения класса моделируемых систем можно выделить конечный набор абстрактных элементов, называемых объектами. Набор логических правил также ограничен и может быть описан небольшим количеством стандартных операций.

Объекты ГПСС делят на 7 категорий и 14 типов:

1. динамическая. Тип – транзакты
2. операционная. Тип – блоки
3. аппаратная. Устройства, память, ключи
4. вычислительная. Переменные (арифметические, булевские), функции
5. статистическая. Таблицы, очереди
6. запоминающая. Ячейки, матрицы
7. группирующая. Списки пользователя, группы

Перед изложением программы можно встретить блок-схему, где каждый макрос отображается графически.

Основой системы является программа, описывающая функционирование выделенного конечного набора объектов и специализированная диспетчеризирующая программа (программа-симулятор), которая выполняет следующие функции:

- Обеспечение заданных маршрутов продвижения динамических объектов (транзактов).
- Планирование событий, происходящих в модели, путем регистрации времени наступления каждого и выполнение в нарастающей временной последовательности.

Динамическими объектами являются транзакты (сообщения), которые представляют собой единицы исследуемых потоков и производят ряд определенных действий, продвигаясь по фиксированной структуре (всем остальным блокам, кроме динамических)

Операционные объекты – блоки, задающие логику функционирования модели и пути движения транзактов. Как правило, между объектами аппаратной категории.

Аппаратные объекты – абстрактные элементы, на которые может быть декомпозировано оборудование реальной системы. Воздействуя на эти объекты, транзакты могут изменять их состояния и влиять на движение других транзактов.

Вычислительная категория – служит для описания таких ситуаций в процессе моделирования, когда связи между компонентами моделируемой системы наиболее просто и компактно выражаются в виде математических соотношений.

В процессе моделирования системы и взаимодействия объектов друг с другом, происходят изменения атрибутов и преобразования их значений. Такие преобразования называются событиями.

Каждому объекту присущи атрибуты, которые описывают его состояние в данный момент времени. Значения атрибутов могут быть арифметическими или логическими. Бо̀льшая часть атрибутов недоступна для пользователя – атрибуты, которые необходимо адресовать, называются стандартными числовыми \ логическими атрибутами.

В языке делается рекомендация – название своих собственных переменных – не менее чем 3-4 символа, чтобы случайно не попасть в стандартный числовой атрибут.

Управляющие средства языка – команды (start, show). Есть определенные требования к написанию команд. Раньше существовало строгое разграничение по полям – вначале метки, затем команды, под конец комментарии.

Результаты можно видеть не только в отчете (стандартными средствами) но и в различных окнах, а также на строимых графиках (команда plot).

Принципы построения и организации

Имеется два типа основных объектов – транзакты и блоки. Практически все изменения состояния модели системы происходят в результате входа транзактов в блоки и выполнения блоками своих функций. С блоками непосредственно связаны:

- Операционные – изменяют процесс моделирования
- Вывода – печать промежуточных результатов
- Команды, управляющие процессом моделирования
- Команды редактирования

Команда ANOVA – реализует дисперсионный анализ по характеристике.

Всем блокам присваиваются номера, чтобы можно было к ним обратиться.

Транзакты представляют собой не только единицы потока, но и описания динамических процессов в реальных системах. Они могут описывать реальные физические объекты (автомобиль на бензоколонке) и нефизические (канальные программы).

Транзакты можно генерировать и уничтожать. Основным атрибутом транзакта являются параметры – 0..1020 штук. Параметр может быть словом, полусловом, байтом или плавающей точкой. Также важным атрибутом является приоритет – он изменяется от 0 до 127. В случае, когда два транзакта соперничают при занятии устройства, первым обрабатывается тот, у которого приоритет выше. Если приоритеты одинаковые, то сначала обрабатывается тот, у которого время ожидания обработки больше.

Классификация блоков GPSS

Блоки используются для описания функций и управляют движениями транзактов. У каждого блока имеются два стандартных числовых атрибута (СЧА): W_n – счетчик входов в блок (ожидающий счетчик), который содержит в себе номер текущего транзакта, находящегося в блоке. N – общий счетчик транзактов, поступивших в блок с начального момента моделирования или с момента обнуления.

Две управляющие команды – RESET, CLEAR. Первая – сбрасывает временные счетчики, вторая – обнуляет всю модель.

По назначению вводится следующая классификация блоков:

1. Блоки, осуществляющие модификацию атрибутов транзактов.

   Временная задержка:

   ADVANCE (при использовании функций – потребуется производить масштабирование)

   Генерирование и уничтожение транзактов:

   GENERATE (второй параметр – модификатор)

   TERMINATE

   SPLIT – берёт транзакт и создаёт копии одного «ансамбля» транзактов

ASSEMBLE

Синхронизация движения нескольких транзактов:

MATCH

GATHER

Изменение параметров транзактов:

ASSIGN

INDEX

MARK

Изменение приоритета

PRIORITY

2. Блоки, изменяющие последовательность передвижения транзактов (блоки передачи управления)
   - TRANSFER
   - LOOP
   - TEST (устройство – занято ли, память – свободна ли)
   - GATE
3. Блоки, связанные с группирующей категорией
   - JOIN
   - REMOVE
   - EXAMINE
   - SCAN
   - ALTER
4. Блоки, сохраняющие необходимые значения для дальнейшего использования
   - SAVEVALUE
   - MSAVEVALUE
5. Блоки, организующие использование объектов аппаратной категории

   Устройств:

   - SEIZE
   - RELEASE
   - PREEMPT
   - RETURN
   - FAVAIL (facilities avail)
   - MFAVAIL
   - FUNAVAIL

   Памяти:

   - ENTER
   - LEAVE
   - SAVAIL
   - SUNAVAIL

   Ключи:

   - LOGIC
6. Блоки, обеспечивающие получение статистических результатов
   - QUEUE
   - DEPART
   - TABLE
   - TABULATE
7. TRACE, SELECT, HELP
8. Блоки для организации цепей
   - LINK

   UNLINK
9. Вспомогательные блоки
   - WRITE
   - SAVE
   - LOAD
   - REPORT
   - UPDATE

Формант команды на языке GPSS

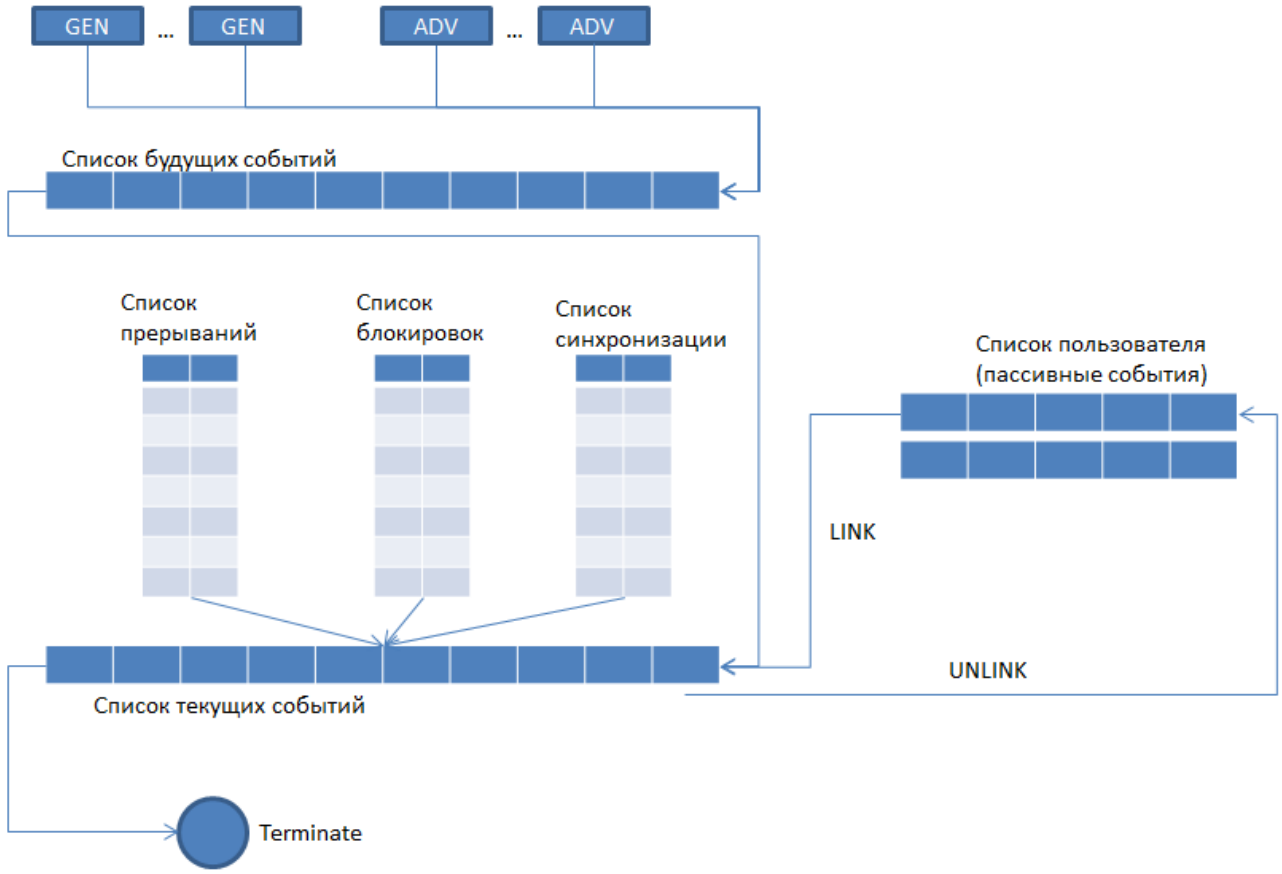
При описании схемы алгоритма с помощью языковых средств, каждый блок определяется с помощью отдельной команды независимо от редактора.

1. Поле метки: символический адрес блока
2. Поле операции: наименование типа блока
3. Поле операндов: перечисляются все операнды, присущие блоку, A..H
4. ;комментарий

21.12.15 ----------

Структура управляющей программы на языке GPSS

Generate … Generate Advance … Advance



Поддерживается сложная структура списков. С целью уменьшения времени на просмотр этих списков система ведёт два основных списка:

Список **текущих событий** – включает все события, запланированные на текущий момент модельного времени, независимо от того условные они или безусловные.

Управляющая программа:

1. Просматривает этот список
2. Пытается переместить по модели те транзакты, для которых выполнены условия (время).
3. Если в этом списке таких транзактов нет, то обращается к списку будущих событий
4. Переносит все события, которые запланированы на ближайший момент модельного времени
5. Заново просматривает список.

Такой перенос также осуществляется в случае совпадения текущего времени со временем первого события в списке будущих событий.

В СТС транзакты размещены в порядке уменьшения приоритетов. Транзакты с одинаковыми приоритетами размещаются в соответствии с последовательностью поступления в список. Каждый транзакт СТС может находиться или в активном состоянии, или в состоянии задержки.

1. В начальный момент (при выполнении оператора управления START) управляющая программа обращается ко всем блокам GENERATE
2. Каждый из них планирует момент появления транзактов и заносит их в СБС.
3. Программа просматривает СТС (на данный момент в нем пусто).
4. Программа просматривает СБС все транзакты на ближайшее время, заносит в СТС и начинает продвигать модельное время.
5. Если продвижение транзакта было задержано по какой-либо причине, не связанной с блоком ADVANCE, то он остается в списке текущих событий и управляющая программа требует заново перемещать его далее по блокам.
6. Если транзакт входит в блок адванс, то выход из этого блока для транзакта планируется и заносится в СБС.

СТС и СБС можно вывести на экран с помощью управляющей команды EVENTS(?) или в окне списков ГПСС.

Список **блокировок** – список транзактов, которые ожидают изменения состояния ресурса. Существует шесть видов таких списков, связанных с устройствами, семь связанных с памятями, и два связанных с логическими ключами.
С устройствами используются списки для: занятых и незанятых, доступных и недоступных, работающих с прерываниями и без прерываний.
Памяти: списки для заполненного, незаполненного, пустого непустого доступного недоступного.
Ключи: списки для включенных\выключенных ключей.

Список **прерываний** – содержит прерванные во время обслуживания транзакты, а также транзакты, вызвавшие прерывания. Данный список используется для организации обслуживания одноканальных устройств по абсолютным приоритетам, что дает возможность организовать приоритетные дисциплины обслуживания.

Список **синхронизации** – содержит транзакты, которые на текущий момент ждут сравнения. Список работает с транзактами, полученными с помощью блока SPLIT – создает транзакты-копии, принадлежащие одному ансамблю; синхронизацию движения транзактов одного семейства выполняется блоком MATCH (также ASSEMBLE собирает все транзакты-копии и выдает один начальный, GATHER собирает заданное количество транзактов и задерживает их пока не соберётся нужное число копий).

Остановленные процессы находятся в СБС, СС и списке блокировок.

Список пользователя содержит транзакты, выведенные пользователем с помощью LINK и помещенные в список прерываний как неактивные. Управляющей программе они будут недоступны до UNLINK.

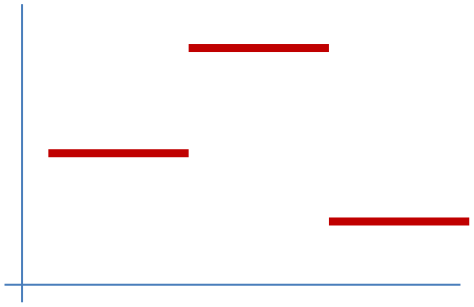
Функции GPSS

Функция – стандартный числовой атрибут, название и численная зависимость.

Два основных типа – дискретные и непрерывные. Пример: хранилище S с занятой памятью SMemory.
Дискретная функция:

Func1 FUNCTION S$Mem, D3 *; дискретная по трем точкам, ещё C, M, L, …, S – стандартный числовой атрибут*

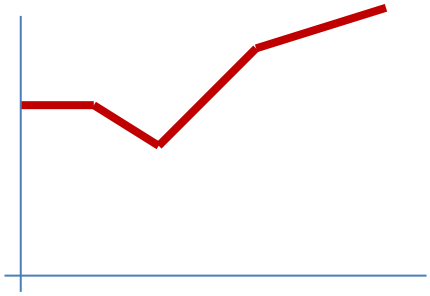
5,12 / 9,20 / 12,6 *; дробная часть различается точкой*



до 5 будет идти на уровне 12, до 9 на 20, до 12 на 6 (кусочная). Функция неявно продлевается влево и вправо если пользователь введёт соответствующий аргумент. На граничных участках берётся «всё ещё» значение (в 5 – 12).

В случае непрерывной – не набор горизонтальных отрезков, а ломаная.

4.5, 12.1 / 5.5, 10.5 / 7, 14 / 9, 15.3



Если задается много точек, то новая точка начинается со следующей строки, причем знак слэша в конце не ставится. Во всех незаданных участках (влево и вправо) функция продлевается горизонтально.

Для обращения к функции: FN$Func1