

Простейшие алгоритмы генерации последовательности псевдослучайных чисел.

Одним из первых способов было выделение значения дробной части у многочлена первой степени.

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

Карл Якоби доказал, что при рациональном a множество y конечно, а при иррациональном - бесконечно и всюду плотно в интервале от 0 до 1.

Для многочленов большей степени Вейлер придумал критерий равномерности распределения любой функции от натурального ряда чисел. Заключается в том, что среднее по реализации псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1.

Фон Нейман придумал способ: каждое последующее случайное число образуется возведением предыдущего в квадрат и отбрасыванием чисел с обоих концов.

Лемер предложил способ:

$$g_{n+1} = kg_n + c \text{ mod } M$$

$$g_{n+1} = \text{ent}(g_n \sqrt{2}) \text{ - в таком случае у чисел есть период, который зависит от начального заполнения.}$$

Дальше решили вести все вычисления в целых комплексных числах. При $c = 0$ и $M = 2^n$ наибольший период достигается при $k = 3 + 8i$; $k = 5 + 8i$.

Через 5 лет Форсайд показал, что тройки этой последовательности лежат на 15 параллельных плоскостях. От отчаяния используют 2 или даже более генераторов, смешивая их значения. Если разные генераторы независимы, то сумма их последовательностей обладает дисперсией, равной сумме дисперсий ("случайность" возрастает). В современных системах используют конгруэнтные генераторы по алгоритму, предложенному Бюро стандартов США.

Метод, основанный на числах Фибоначчи.

Фрагмент программы на Фортране

```
var n, i: integer;
    x, R: double;
const m34: double = 28395423107.0;
      m35: double = 34359738368.0;
      m36: double = 68719476736.0;
      m37: double = 137438953472.0;
function RAND(n: integer): double;
var S, W: double;
    i: integer;
BEGIN
    if n = 0 then
begin
        x:=m34; Rand:=0; exit ;
    end;
    s:=-2.5;
    for i:=1 to 5 do
begin
        x:=5.0*x;
        if x>=m37 then x:=x-m37;
        if x>=m36 then x:=x-m36;
        if x>=m35 then x:=x-m36;
        w:= x/m35;
        if n = 1 then
begin
            RAND := w, exit
        end;
        S:=S+W;
    end;
    s = s*1.54919;
    RAND=(sqr(s)-3.0)*s*0.01+s;
END;
//Программа:
BEGIN
    R:=RAND(0); // настройка программы. n = 1 - равномерное, n = 2 гауссовое
    for i:=1 to 2 do
        writen (RAND(2):12:8)
```

END

```
SUBROUTINE RANDOM(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
```

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

IY - полученное число, используется при дальнейших обращениях к подпрограмме.

RN - равномерно распределенное случайное на число на (0;1).

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

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

Случайная величина $x = \sqrt{r(b-a) + a}$. r - равномерно распределенная случайная величина.

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

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

Распределение Пуассона. В основе метода лежит генерация случайных значений некоторой переменной r_i на интервале $\prod_{i=0}^x r_i \geq e^{-8} > \prod_{i=0}^{x+1} r_i$

При определении случайной величины, функция распределения которой не даёт решения уравнения в явной форме, можно произвести кусочную аппроксимацию и искать приближенные значения. Кроме того, при получении случайных величин часто используют те или иные свойства распределений.

Случайная величина, распределенная по закону Эрланга k -ого порядка. При генерации воспользуемся тем, что поток Эрланга может быть получен прореживанием потока Пуассона k раз. Поэтому достаточно получить k значений случайной величины, распределенной по показательному закону и усреднить их.

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.

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

Методика построений программной модели.

Для разработки программной модели исходная система должна быть представлена как стохастическая система массового обслуживания. Это объясняется следующим: информация от внешней среды поступает в случайные моменты времени, длительность обработки различных типов информации в общем случае так же будет различна, следовательно, среда является генератором сообщений, а комплекс вычислительных сред - обслуживающими устройствами. Источники информации выдают на вход блока памяти незасимо друг от друга сообщения. В памяти сообщения записываются в навал и выбираются по одному обслуживающим аппаратом (FIFO). Длительность обработки одного сообщения обслуживающим аппаратом в общем случае случайна. Быстродействие ограничено, следовательно создается очередь.

Моделирование потока сообщений. Поток сообщений обычно моделируется моментами появления очередного сообщения в потоке. Текущий момент времени появления очередного сообщения.

Закон распределения	Вычисление выражений
Равномерное распределение	$T_i = a + (b-a)r$
Экспоненциальное	$T_i = -\frac{1}{\lambda} \ln(1 - R_i)$
Нормальное	
Эрланга	$T_i = \frac{1}{k\lambda}$

Моделирование работы обслуживающего аппарата (10.11.2014)

Программа, имитирующая работу обслуживающего аппарата - это набор программ, вырабатывающих случайные отрезки времени.

По нормальному закону время обработки $t = MX \sum_{i=1}^1 (R_i - 6) \sigma_x$



Figure 1:

Моделирование абонента. Абонент может рассматриваться как обслуживающий аппарат, поток информации о котором поступает от процессора. Поэтому при моделировании его работы необходимо также вырабатывать длительности обслуживания. Кроме того, абонент сам может быть источником заявок на те или иные ресурсы вычислительной системы. Эти заявки можно имитировать с помощью генератора. Следовательно, абонент может быть промоделирован либо как обслуживающий аппарат, либо как генератор.

Моделирование работы буферной памяти. (рисунок 2)

Признак выбора равен 1 - режим выборки сообщения из буферной памяти. Признак выбора равен 0 - режим записи. NP - число сообщений, хранящихся в памяти. NPOS - номер последнего сообщений, NPR - номер первого из имеющихся в памяти сообщений. POL - признак переполнения, 1 - в памяти нет ни одного сообщения. X - текущее значение в памяти

Разработка программы сбора статистики. Задача сбора статистики заключается в накоплении численных значений, необходимых для вычисления статистических оценок параметров исследуемой схемы. При исследовании простейшей СМО интерес представляет. Для каждого сообщения время ожидания в очереди равно разности между моментом времени, когда оно было выбрано на обработку и моментом времени, когда оно пришло в систему. Необходимо все эти времена усреднить, то есть надо найти общее число обслуживаемых сообщений. Суммируя количество сообщений в памяти через небольшие промежутки времени и разделив эту сумму на число суммирований получим среднее число значения длины очереди.

Коэффициент загрузки обслуживающего аппарата - время, которое аппарат потратил на обслуживание делить на всё время моделирования.

Управляющий блок. Основное значение управляющей программы - узнавать, когда что запускать. Для определения, как в нужный момент времени активировать тот или иной программный имитатор существует несколько алгоритмов. Разбить время на интервалы и смотреть, какие из событий попали в конкретный интервал времени, то есть просматривать систему с чётко фиксированным шагом. Из-за недостаточно большого шага можно пропустить некоторые события. Изменения состояния системы связаны с событиями, значит, можно выполнить некоторые начальные установки и создать список событий на начальный момент времени. В список заносим ближайшие времена для каждого объекта. Недостаток алгоритма: слишком много времени будет тратиться на просмотр длинных списков событий.

Для экономии времени был разработан новый алгоритм. Все события группируются вокруг некоторого квази.

Лабораторная работа 4. Есть генератор, который генерирует по первому исследуемому нами распределению (равномерному), вводить любые циферки. Есть память и есть обслуживающий аппарат, который обслуживает все заявки по второму закону. Нужно посмотреть данную систему. Добавляются следующие вещи: промоделировать систему, найти минимальный объем памяти, при котором будут отсутствовать потери в системе. Второй случай: Заявки перенаправляем в очередь, таким образом происходит суммирование



Figure 2:

потоков заявок. Был один равномерный и добавился еще неравномерный. Вводится два параметра, которые контролируют количество заявок, которые перенаправляются и количество заявок, которые выходят. Реализовать событийным алгоритмом и алгоритмом дельта тэ. (рисунок 6)

Лабораторная работа 6. Составить концептуальную модель объекта, формализация которого может быть выполнена с помощью СМО. Минимально десять аппаратов обслуживания.

Управляющая программа имитирует алгоритм взаимодействия отдельных устройств системы. Управляющая программа в основном реализуется по следующим двум принципам:

1. Принцип Δt . Последовательность всех блоков в момент времени $t + \Delta t$, по состоянию в момент времени t . При этом новое состояние блоков определяется их алгоритмическим описанием с учетом действующих случайных факторов, задаваемых распределением вероятностей. В результате данного анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью в данный момент времени. Основной недостаток принципа Δt : значительные затраты времени моделирования системы при малом Δt , а при недостаточно малом Δt появляется опасность пропуска отдельных событий и искажение результатов моделирования.
2. Событийный. Характерное свойство моделируемой системы обработки информации заключается в том, что состояние отдельных устройств, изменяющееся в отдельный дискретный момент времени, совпадающее с моментами поступления сообщений в систему (окончание решения задач, возникновение аварийных сигналов, моменты сбора статистики). Время удобно связывать с событиями в системе. При данном принципе состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. Момент поступления следующего события определяется минимальным значением из списка будущих событий, представляющих собой совокупность моментов ближайшего изменения состояния каждого из блоков системы. Недостаток: большое время тратится на обработку списка. (рисунок с графиком). b_1 - время обслуживания первого сообщения. t_{3i} - момент сбора статистики по обслуживающей модели. t_{41} —время окончания моделирования. Для всех активных блоков (блоки, порождающие события) заводят свой элемент списка. В качестве подготовительной операции в список будущих событий заносят время ближайшего события от любого (или от всех) активных блоков. Активизируя программный имитатор генератора или источника информации, вырабатываем псевдослучайные значения a_0 , определяющие момент времени t_{11} . Эту величину времени заносят в список. Вырабатываем величину b_0 , определяющую t_{21} . Момент t_{31} известен. Также в список заносится время окончания моделирования. Алгоритм протяжки. Ищем минимальное время и номер блока, которому оно принадлежит. Обрабатываем блок. Как только обработаны все значения, переходим в блок обработки статистической информации. Реализация события заключается

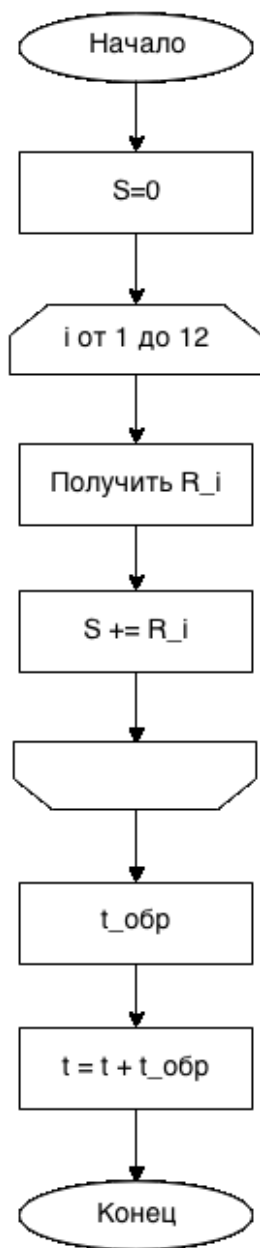


Figure 3:

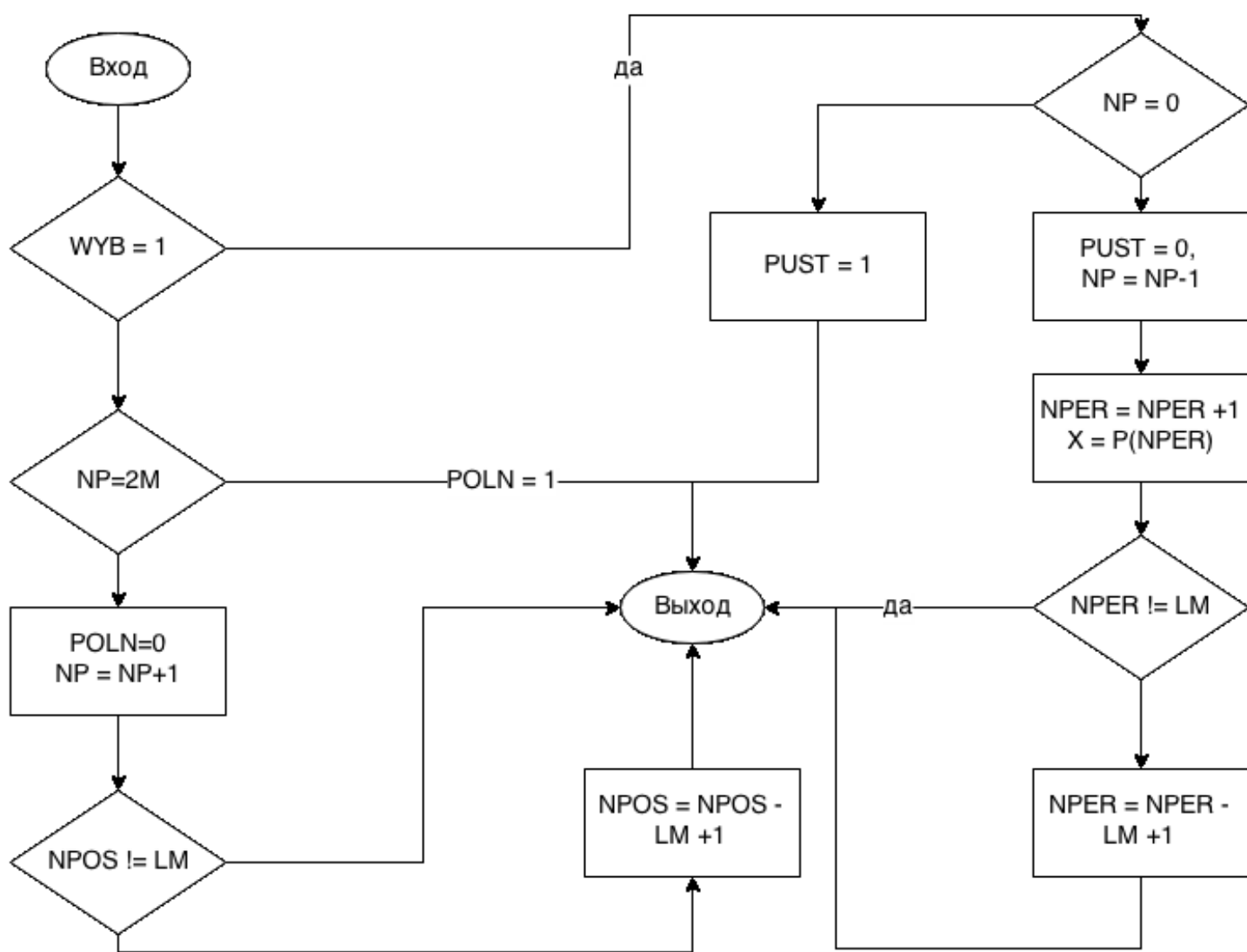


Figure 4:

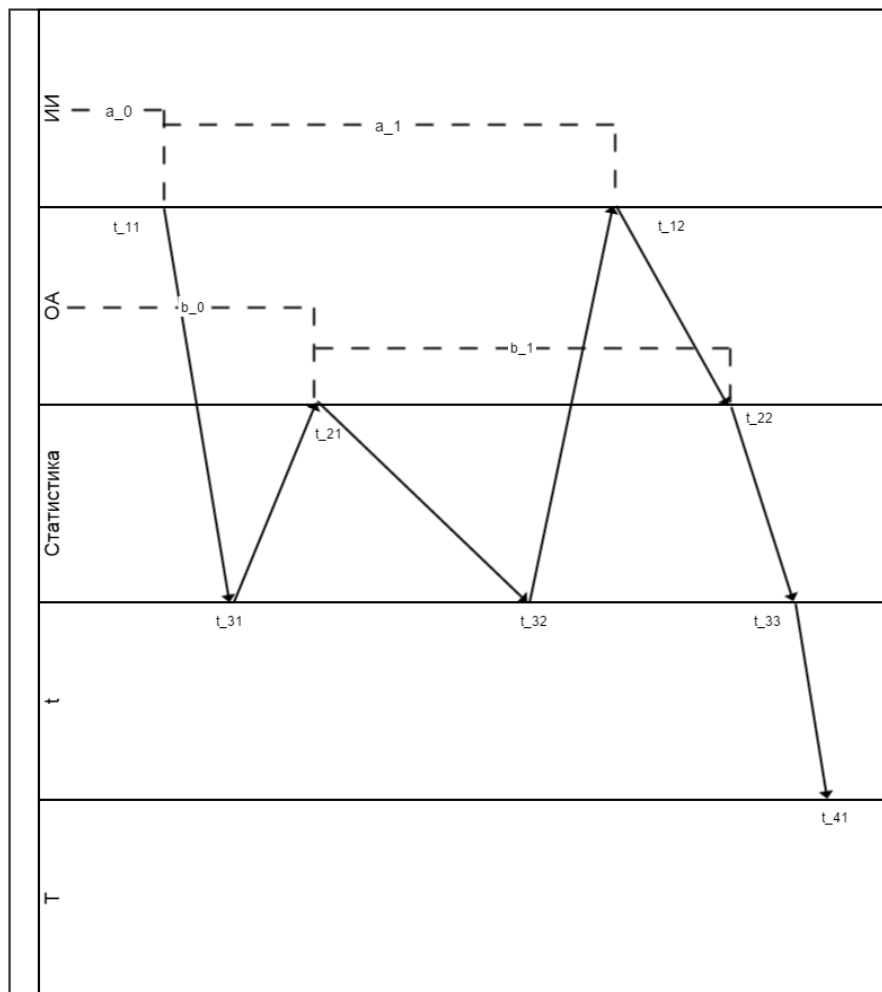


Figure 5:

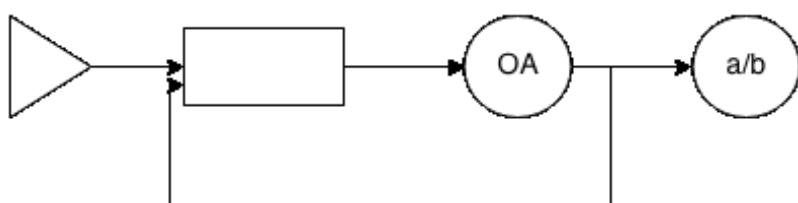


Figure 6:

в том, что само сообщение записывается в память и с помощью имитатора вырабатывается момент появления следующего сообщения и это время помещается в список событий вместо t_{11} . Далее вновь осуществляется поиск минимума и т.д.

3. Дельфт. Во многих случайных системах распределение событий неоднородно, они группируются по времени. Образование групп связано с наступлением какого-либо “значимого” события (которое потянет за собой эту цепочку).

Классификация систем массового обслуживания (17.11.2014)

Принято классифицировать СМО по следующим признакам:

1. По закону распределения входного потока
2. Числу обслуживающих приборов
3. Закону распределения времени обслуживания в обслуживающих приборах
4. Числу мест в очереди
5. Дисциплине обслуживания

Для краткости записи при обозначении любой СМО $A|B|C|D|E$, где

A - закон распределения интервалов времени между поступлениями заявок. Наиболее часто используются следующие:

- Экспоненциальная - M
- Эрланговская - E
- Гиперэкспоненциальное - H
- Гамма-распределение - Γ
- Детерминированное - D

Для обозначения произвольного характера распределения используется G.

B - закон распределения времени обслуживания в приборах. Здесь приняты такие же обозначения, как и для интервалов поступления заявок.

C - число обслуживающих приборов. Для одноканальной системы - 1. Для многоканальных в общем случае l .

D - число мест в очереди. Если число мест в очереди не ограничено, данное значение может опускаться. Для конечного числа мест можно написать r или n .

E - дисциплина обслуживания. FIFO, LIFO, random.

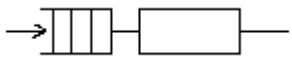
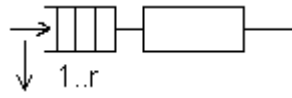
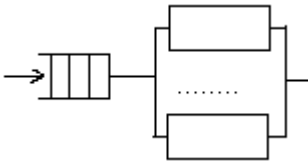
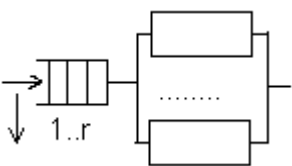
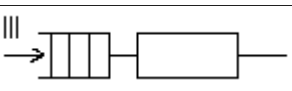
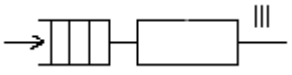
Пример: $M|M|1$ - СМО с одним обслуживающим прибором, бесконечной очередью, экспоненциальный закон распределения интервалов времени между поступлениями заявок и времени обслуживания, дисциплина обслуживания FIFO. $G|G|L$ - СМО с несколькими обслуживающими приборами, бесконечной очередью, произвольным распределением, дисциплиной FIFO. $E|H|L|r|LIFO$ - с бесконечной очередью, многоканальная, эрланговское и гиперэкспоненциальное распределение, LIFO.

Используются следующие типы СМО:

1. Одноканальная СМО с ожиданием. Представляет собой один обслуживающий прибор с бесконечной очередью, является наиболее распространенной при моделировании в той или иной степени приближения с ее помощью можно моделировать любой узел вычислительной системы или локальной вычислительной сети
2. Одноканальная СМО с потерей. Представляет собой один обслуживающий прибор с конечным числом мест в очереди. Если число заявок превышает число мест в очереди, то лишние заявки теряются. Используется при моделировании каналов передачи.

3. Многоканальные СМО с ожиданием. Представляют собой несколько параллельно работающих обслуживающих приборов с общей бесконечной очередью. Данный тип СМО часто используют при моделировании групп абонентских терминалов, работающих в диалоговых режимах.
4. Многоканальные СМО с потерями. Представляют собой несколько параллельно работающих обслуживающих приборов с общей очередью, число мест в которой ограничено. Этот класс СМО, как и одноканальные с потерями, часто используется для моделирования каналов связи.
5. Одноканальные СМО с групповым поступлением заявок. Представляют собой один обслуживающий прибор с бесконечной очередью. Перед обслуживанием заявки группируются в пакеты по определенному правилу.
6. Одноканальные СМО с групповым обслуживанием заявок. Представляют собой один обслуживающий прибор с бесконечной очередью. Заявки обслуживаются пакетами, предоставляемыми по определенному правилу.

Последние 2 типа СМО могут использоваться для моделирования центров коммутации.

Наименование	Обозначение	Схема
Одноканальная СМО с ожиданием	$G G 1$	
Одноканальная СМО с потерей	$G G 1 r$	
Многоканальная СМО с ожиданием	$G G L$	
Многоканальная СМО с потерями	$G G L r$	
Одноканальная СМО с групповым поступлением заявок	$G_r \frac{G}{r} 1$	
Одноканальные СМО с групповым обслуживанием заявок	$G \frac{Gr}{r} 1$	

Вычислительные сети

Любая вычислительная сеть может быть формализована прибором массового обслуживания (также можно формализовать конечным или вероятностным автоматом). Различают открытые, замкнутые и смешанные сети.

Открытой называется сеть массового обслуживания, состоящая из m узлов, причем хотя бы в один из узлов сети поступает извне входной поток заявок и обязательно должен быть сток заявок из сети. Для открытых сетей характерным является то, что интенсивность поступления заявок в сеть не зависит от состояния сети (от числа заявок уже поступивших в сеть). Такие сети, как правило, используются для моделирования вычислительных сетей, работающих в неоперативном режиме.

На рисунке S1, S2 (приборы обслуживания) моделируют работу узлов коммутации. S3, S4 - работу сервера. S5, S6 - работу межузловых каналов. В данной сети циркулируют 2 потока заявок. Каждая заявка поступает на вход соответствующего узла коммутации, где определяется место ее обработки. Затем заявка передается на "свой" сервер или по каналу связи на соседний, где и обрабатывается, после чего возвращается к источнику и покидает сеть.

Замкнутой называется сеть массового обслуживания с множеством узлов без источника и стока, в которой циркулирует постоянное число заявок. Замкнутые сети массового обслуживания используются

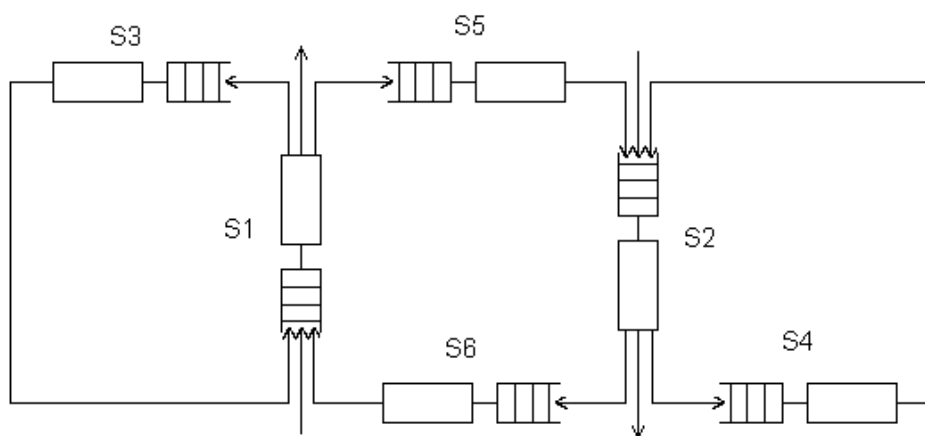


Figure 7:

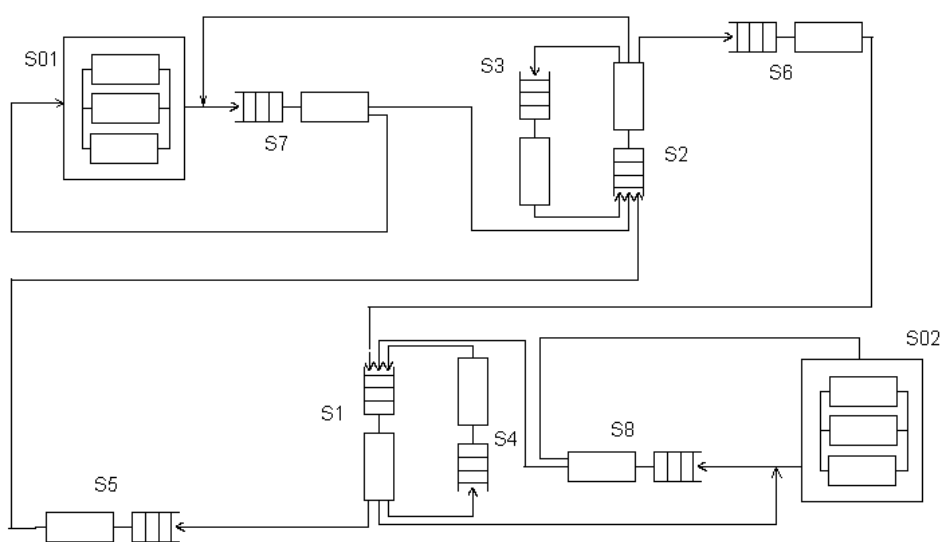


Figure 8:

для моделирования таких вычислительных сетей источниками информации для которых служат абонентские терминалы, работающие в диалоговом режиме. В этом случае каждая группа абонентских терминалов представляется в виде многоканальной СМО с ожиданием и включается в состав устойчивой сети. Различают два режима работы диалоговых абонентов:

- Простой. Абоненты не производят никаких действий, кроме генерации заданий ВС и обдумывания ответов. рисунок 8. S01, S02 - группа абонентских терминалов; S7, S8 - каналы связи с абонентами; S2, S4 - узлы коммутации; S5, S6 - каналы межузловой связи. Абоненты с терминалов посылают запросы, которые по каналам связи поступают на узлы коммутации, а оттуда на обработку на свой или соседний.
- Сложный. Работа абонента представляется в виде совокупности операций у некоего процесса, называемого технологическим. В каждой операции технологического процесса моделируется соответствующий СМО. Часть операций предусматривает обращение к вычислительной сети, в части операций такого обращения может и не быть. (рисунок 8)
- Смешанной называется сеть массового обслуживания, в которой циркулирует несколько различных типов заявок. Причем, относительно одних типов заявок сеть замкнута, а относительно других типов - сеть открыта. С помощью смешанных сетей массового обслуживания моделируются такие вычислительные системы, часть абонентов которых работают в диалоговом режиме, а часть в неоперативном. Причем для диалоговых абонентов также в свою очередь различают простой и сложный режимы работы. Часто смешанные СМО моделируют такие сети, в которых сервер дополнительно загружается задачами, решаемыми на фоне работы самой сети.

Лабораторная работа 5. Информационный центр. В информационный центр приходят клиенты через интервалы времени 10 ± 2 минуты (равномерное распределение). Если все три из имеющихся операторов заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 20 ± 5 , 40 ± 10 , 40 ± 20 минут. Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы сдаются в приемный накопитель, откуда будут выбираться для обработки. Для первого компьютера запросы пойдут от 1 и 2 операторов, на 2 - запросы от третьего. Время обработки запросов на 1 и 2 вычислительной машине равны 15 и 30 минут соответственно. Смоделировать процесс **обработки** 300 запросов, определить вероятность заказа.

Эндогенные переменные: время обработки i -ым оператором t_{0i} , время обработки задачи на j -ом компьютере t_{rj} . Экзогенные переменные: число обслуженных клиентов, число клиентов, получивших отказ. Уравнение вероятности отказа. За единицу модельного времени выбрать $1/100$ минуты.

Сети Петри 24.11.2014

Сеть Петри - математическая модель дискретных динамических систем (параллельных программ, операционных систем, компьютеров и их устройств, вычислительных сетей и т.д.) ориентированная на качественный анализ и синтез таких систем (обнаружение блокировок, тупиковых ситуаций и узких мест, автоматический синтез параллельных программ и компонентов компьютера и т.д.). Формально в терминах теории систем сеть Петри может быть определена следующим набором элементов (кортежем):

$$PN = \{\theta, P, T, F, M_0\}$$

$\Theta = \{0, 1, 2, \dots\}$ - множество моментов времени

$P = \{p_1, p_2, p_3, \dots\}$ - непустое множество элементов сети, называемых позициями

$T = \{t_1, t_2, t_3, \dots\}$ - непустое множество элементов сети, называемых переходами

$F : (P \times T) \cup (T \times P) \rightarrow \{0, 1, 2, \dots, k\}$ - функция инцидентности, k - кратность дуги

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

F может быть представлена в виде $F^P \cup F^T$ и задает

1. $F^P(p, t) = P \times T \rightarrow \{0, 1, 2, \dots\}$. Для каждой позиции указываются связанные с ней переходы с учетом их кратности

2. $F^T(t, p) = T \times P \rightarrow \{0, 1, 2, \dots\}$. Для каждого перехода указываются связанные с ним позиции

Эти функции могут быть представлены матрицей инцидентности

$$F^P = \begin{pmatrix} f_{11}^P & f_{12}^P & \dots & f_{1n}^P \\ f_{21}^P & f_{22}^P & \dots & f_{2n}^P \\ \dots & \dots & \dots & \dots \\ f_{41}^P & f_{42}^P & \dots & f_{mn}^P \end{pmatrix}, F^T = \begin{pmatrix} f_{11}^T & f_{12}^T & \dots & f_{1n}^T \\ f_{21}^T & f_{22}^T & \dots & f_{2n}^T \\ \dots & \dots & \dots & \dots \\ f_{41}^T & f_{42}^T & \dots & f_{mn}^T \end{pmatrix}$$

$$p_i \in p, t_j \in T, f_{ij}^P > 0$$

t_j - выходной переход позиции p_i

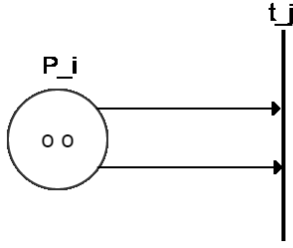


Figure 9: Простой переход

Каждая позиция p_i может содержать некоторый целочисленный ресурс $\mu(p) \geq 0$, отображаемое числом точек (фишек) внутри позиции.

Вектор $M = [\mu_1, \mu_2, \dots]$ будет называться маркировкой, или разметкой, сети Петри. Маркировка внутри позиции.

Начальная маркировка M_0 описывает начальное состояние сети.

Сеть функционирует в дискретном времени в асинхронном режиме, переходя от одной маркировки к другой. Смена маркировок происходит в результате срабатывания переходов сети. Переход t_j может сработать при некоторой маркировке M , если $\forall t_i \in T, p_i \in P^j \mu_i(\Theta) - f_{ij}^p(\Theta) \geq 0$

То есть если все позиции 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 сменяется по следующему правилу:

$$\mu_i(\Theta + 1) = \mu_i(\Theta) - f_i^P(\Theta) + f_{ij}^t(\Theta)$$

Переход t изымает из своей позиции число фишек, равное кратности числа входных дуг и посылает в 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 (тупиковой маркировке) ни один из переходов не может сработать. При одной и той же начальной маркировке сеть Петри может порождать в силу недетерминированности ее функционирования различные последовательности срабатывания ее переходов. Множество всевозможных слов, порождаемых сетью Петри, называют языком сети Петри. Две сети Петри эквивалентны, если порождают один и тот же язык. В отличие от конечных автоматов, в терминах которых описываются глобальные состояния системы, сети Петри концентрируют внимание на локальных событиях, локальных позициях и локальных связях между событиями и условиями.

Теоретико-графовым представлением сети Петри является двудольный ориентированный мультиграф, который содержит позиции (места), которые обозначаются кружками. Переходы, которые обозначаются планками и ориентированные дуги, которые соединяют позиции с переходами. Кратные дуги обозначаются несколькими параллельными дугами. Благодаря наличию кратных дуг сеть Петри есть мультиграф. Благодаря двум типам вершин граф называется двудольным, а поскольку дуги имеют направление, граф является ориентированным.

$$F^P : \begin{matrix} & 1 & 1 & 0 & 0 & p_1 \\ 0 & 2 & 0 & 0 & 0 & p_2 \\ 0 & 0 & 1 & 1 & 1 & p_3 \end{matrix} ; F^t : \begin{matrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 0 \end{matrix}$$

$\mu_0 = [2, 2, 0]$ - начальная маркировка

Пространство состояний сети Петри

Пространство состояний сети, обладающей n позиций, есть множество маркировок E^n . Изменения состояний, вызванные запуском переходов определяется функцией перехода - δ или функцией следующего состояния. Функция M и переход t_j , если он разрешен, получается новая маркировка

$$M' = \delta(M, t_j)$$

Эта маркировка получается изъятием фишек из позиции p_i и помещением фишек в позицию p_k . Процесс создания новых маркировок продолжается до тех пор, пока в сети Петри при данной маркировке существует хоть один переход. Если при данной маркировке ни один переход не разрешен, маркировка называется тупиком.

Две последовательности: последовательность маркировок $\{M(0), M(1), M(2), \dots\}$, последовательность запущенных переходов $\{t_{j0}, t_{j1}, t_{j2}, \dots\}$

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

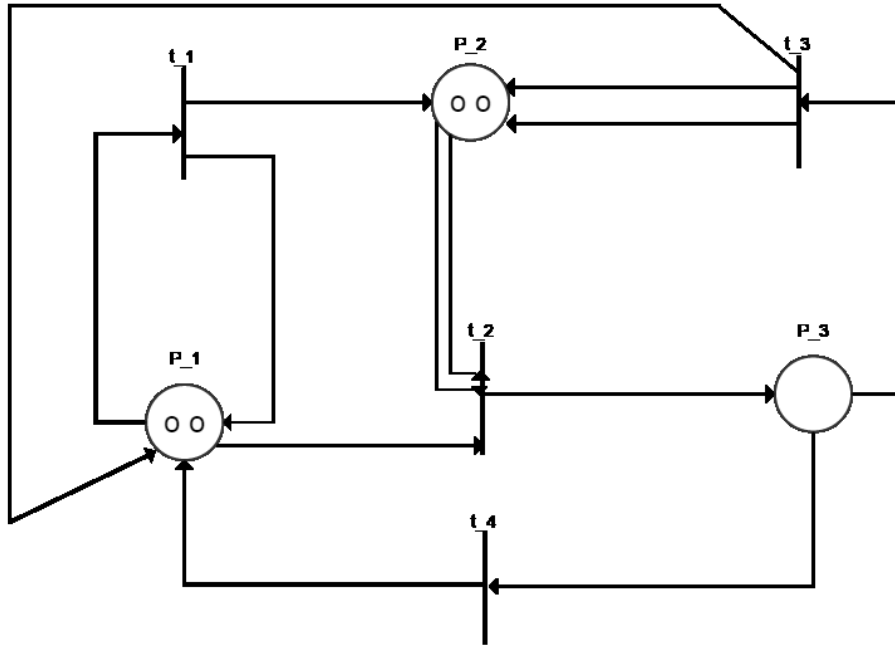


Figure 10:

Если при маркировке M образуется новая маркировка M' , говорят, что M' достижима из M .

Множество достижимости обозначается $R(PN, M)$. Есть некоторое множество M_k , достижимых из M . Множество достижимости в общем случае для сети Петри с маркировками M есть наименьшее множество маркировок, определенных следующим образом:

$$M' \in R(PN, M) \quad M'' = \delta(M', t_j), t_j \in T$$

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

$\Theta = 0$	$\Theta = 1$	$\Theta = 2$	$\Theta = 3$	$\Theta = 4$
$M(0)=[2,2,0]$	$t_1[2, 3, 0]$			
	$[1 -0 1]$	$t_1:[2 4 0]$		
			$t_1[2 5 0]$	
				$t_1[2 6 0]$
				$t_2[1 3 1]$
	$t_2[1 1 1]$	$t_2[1 1 1]$	$t_2[1 2 1]$	
		$t_1[1 1 1]$ повт	$t_1[1 2 1]$	$t_1[1 2 1]$
		$t_3[2 2 0]$ повт	$t_3[2 3 0]$	$t_2[0 0 2]$
		$t_4[2 0 0]$ повт		$t_3[2 4 0]$
				$t_4[2 2 0]$

01.12.2014

Неограниченная сеть.

Свойства безопасности. Сеть называется безопасной, если при любой достижимой маркировке $\mu_i \leq 1 \forall i$.

Свойство консервативности. Сеть называется консервативной, если сумма фишек во всех позициях остается постоянной при работе сети. $\sum_{i=1}^n \mu_i(\theta) = const$

Переход t_j называется потенциально живым, если существует достижимая из M_0 маркировка M' , при которой t_j может сработать. Если t_j потенциально жив при любой маркировке, то он называется живым. А переход t_j , не являющийся потенциально живым при начальной маркировке называется мертвым при этой маркировке; в этом случае маркировка M_0 является тупиковой, то есть при этой маркировке не может сработать ни один переход.

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

Последовательность маркировок M_0, M_1, \dots, M_S называется циклической, если $M_{k+1} = \delta(M_k)$; $k = 1, S$; $M_0 = M_S$.

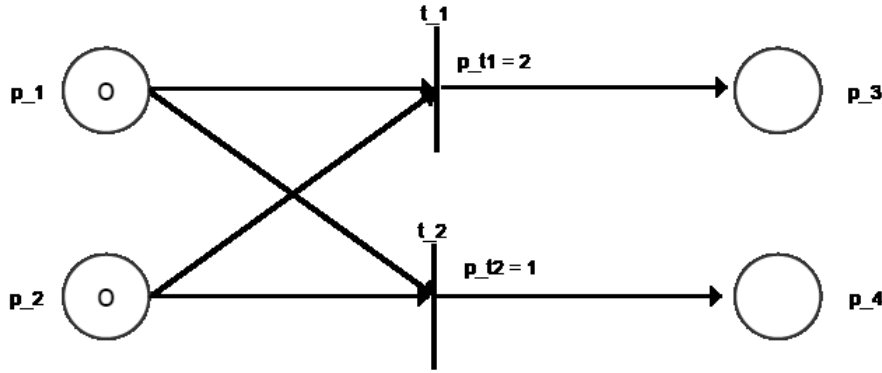


Figure 11: Сеть Петри с приоритетами

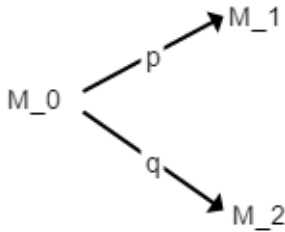


Figure 12: Маркировка сети Петри с приоритетами

Некоторые обобщения сетей Петри

Ингибиторные сети. Для ингибиторной сети $F = F^P \cup F^I \cup F^T$. Она дополняется функцией инцидентности, которая вводит ингибиторные дуги для тех пар $p_i t_j$, для которых $p_i t_j$ в пределах от 0 до 1. Эти дуги отмечаются кружочком. Кратность этих дуг всегда равна единице. Правила срабатывания переходов: t_j может сработать при маркировке M , если для всех связанных позиций выполняется следующее условие $(\mu_i \geq f_{ij}^P) \wedge (\mu_k f_{kj}^I = 0)$. На кружочке p_k фишек быть не должно.

Сети с приоритетом. Сети Петри недетерминированны: если имеется возможность срабатывания нескольких переходов, то срабатывает любой из них. При моделировании реальной сети могут возникнуть условия, когда последовательность срабатывания необходимо регламентировать. Это можно сделать введя множество приоритетов. Каждому переходу приписать соответствующее целочисленное значение. Если на некотором такте работы сети имеется возможность для срабатывания нескольких переходов, то срабатывает тот из них, у которого приоритет наивысший. (рисунок). Из двух переходов первым должен сработать переход t_2 , имеющий приоритет, равный 1, поскольку приоритет t_1 , равный 2, ниже.

Сети со случайным срабатыванием переходов. Сумма вероятностей нормирована и равна 1. Тогда исходная маркировка M_0 приведет к срабатыванию и появлению маркировок M_1, M_s . Причем каждая из этих маркировок будет помечена соответствующей вероятностью. Отождествив маркировки с состоянием сети и предположив, что вероятности сети не зависят от предыдущих состояний, получим: маркировка $M_0 = (1 \ 1 \ 0 \ 0)$, $M_1 = (0 \ 0 \ 1 \ 0)$, $M_2 = (0 \ 0 \ 0 \ 1)$. (рисунок)

Иерархические сети Петри. Представляют собой многоуровневые структуры, позволяют адекватно отображать функционирование иерархических структур, в том числе программ. В отличие от обыкновенных в иерархических сетях имеются два типа переходов: простые и составные. Простые переходы ничем не отличаются от рассмотренных ранее, а составные содержат внутри себя сеть более низкого иерархического уровня. Формально иерархический переход состоит из входного и выходного (хвостового) перехода. Между ними находится некоторая сеть Петри, которая в свою очередь так же может быть иерархической. (рисунок)

Составной переход t_2 . Иерархическая сеть функционирует как обыкновенная сеть Петри, переходя

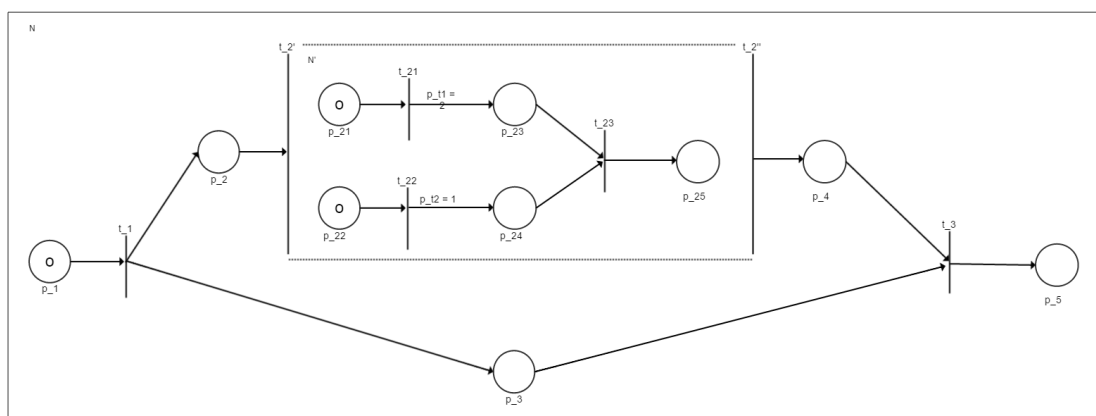


Figure 13: Иерархическая сеть Петри

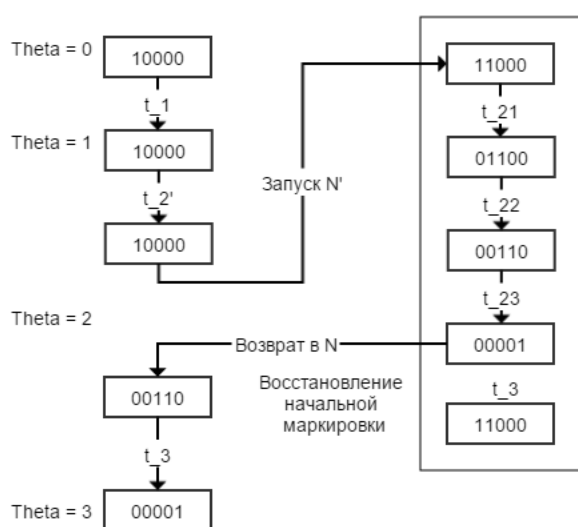


Figure 14: Маркировка иерархической сети Петри

от одной маркировки к другой и обмениваясь фишками в том числе между сетями различного уровня. Исключения составляют только правила работы составных переходов. Срабатывание составных переходов является не мгновенным событием, а составным действием. На каждом шаге дискретного времени Θ составной переход может находиться в одном из двух состояний: пассивном или активном. Начальное состояние всех переходов - пассивное. Составной переход может быть активирован в момент времени Θ , если до этого он был пассивен и имеются условия для срабатывания его головного перехода. При этом производится изменение маркировки в сети верхнего уровня по обычным правилам и одновременно запускается работа сети, находящейся внутри составного перехода. Во время работы функционирование сети верхнего уровня блокируется. Сеть нижнего уровня работает с учетом своей начальной маркировки до тех пор, пока все ее переходы не станут пассивными, то есть не смогут сработать. После этого происходит срабатывание хвостового перехода и изменение маркировки сети верхнего уровня. Составной переход возвращается в пассивное состояние, а в сети нижнего уровня восстанавливается начальная маркировка.

$\Theta = 0$	1 0 0 0 0	t'_2	1 1 0 0 0 - запуск N
$\Theta = 1$	0 1 1 0 0	t_{21}	0 1 1 0 0
$\Theta = 2$	$\{0, 0, 1, 0, 0 \quad 0, 0, 1, 1, 0\}$	t_{22}	0 0 1 1 0
$\Theta = 3$	0 0 0 0 1	t_{23}	0 0 0 0 1 - возврат в N
		t_3	1 1 0 0 0 - восстановление начальной маркировки

Цветные сети Петри. $CPN = \{\Theta, \Sigma, P, T, A, N, C, G, E, I\}$

- Θ —множество дискретных моментов времени или шагов, которые происходят в функционировании

сети

- Σ - все цвета, перечислимый тип
- P - конечное множество позиций. Маркировка, представляет собой мультимножество
- T - конечное множество переходов. В отличие от обыкновенных сетей Петри, механизм реализации более сложен.
- A - конечное множество дуг, связывающих между собой позиции и переходы. Перебор, цикл с параметром.
- N - узловая функция, которая для каждой дуги указывает её конечный и начальный узел.
- C - цветовая функция, определяющая множество типов цветов, разрешенных для каждой позиции.
- G - блокировочная или спусковая функция, описывающая дополнительные условия, которые должны быть выполнены для срабатывания перехода t_j . Эта функция представляет собой предикат, составленный из переменных, принадлежащих типам цветов.
- E - функция, задающая выражения на дугах. Для каждой дуги определяет мультимножество, состоящее из элементов, описанных в множестве цветов. Это мультимножество помечает дугу. Является развитием кратности дуги.
- I - функция инициализации. Для каждой позиции указывается цветное мультимножество.

08.12.2014

Цветные сети Петри с временным механизмом.

В модель системы вводится некоторый временной механизм, показывающий глобальное время. Обычно это время считается дискретным, то есть соответствует тактам, такты выдает тактовый генератор всей системы моделирования. Глобальное время отличается от времени тетта. Срабатывание сети Петри может случиться в любой момент времени. Сеть запускается либо когда придет фишка, либо когда сработает временной сигнал. Ресурсы сети (фишки) могут получать временные метки. Такие ресурсы задаются мультимножествами с временными метками (timed multiset). При описании множества цветов добавляются пометки timed. Переменные снабжаются значком @. Пример: @[500] - 500 единиц модельного времени, @[500,600] - временная задержка равномерно распределена на интервале от 500 до 600 единиц.

О моделирующих возможностях сетей Петри.

Свободные языки представляют собой некоторое подмножество всех слов в алфавите T . Множество свободных языков представляют собой класс свободных языков. В ряде случаев сеть можно изменить, связав с некоторыми переходами сети определенные символы из алфавита, а часть переходов оставить помеченными. Помеченная сеть Петри. Кorteж $(PN, \backslash \text{Sum})$ - помечающая функция. Ставит соответствие перехода некоторому символу. Помеченную сеть Петри можно рассматривать как генератор слов и изучать ее возможности с точки зрения математической лингвистики. Классы языков сравнивают с языками, порождаемыми иными типами абстрактных систем. В частности, это конечные автоматы и машины Тьюринга. Такое сравнение позволяет характеризовать моделирующие возможности сетей Петри, их способность адекватно описывать системы со сложной динамикой функционирования. В теории сложных дискретных систем доказано, что:

1. Класс помеченных сетей Петри строго мощнее класса конечных автоматов и строго менее мощен, чем класс машин Тьюринга.
2. Классы ингибиторных сетей и сетей с приоритетами строго мощнее класса сетей Петри и равномощные классу машин Тьюринга.
3. Класс раскрашенных сетей Петри при конечном количестве цветов равномошен классу сетей Петри.
4. Класс самомодифицируемых сетей эквивалентен классу ингибиторных сетей и сетей с приоритетом.

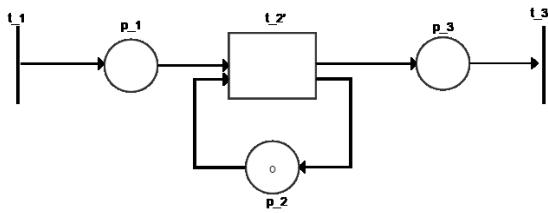


Figure 15:

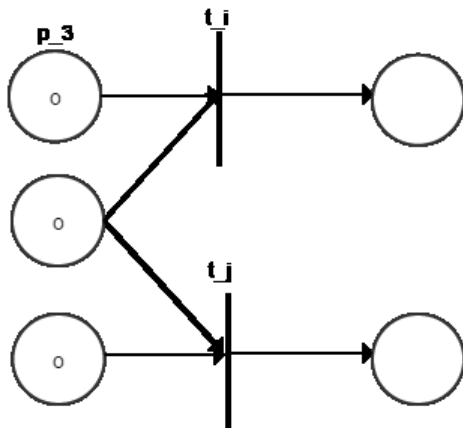


Figure 16: Конфликт

Практическое моделирование дискретных систем.

Сети Петри были разработаны и используются для исследования и моделирования сложных дискретных систем. С помощью различных модификаций этих сетей можно описывать достаточно широкий класс реальных систем, в особенности системы с независимыми элементами. Например, аппаратное и программное обеспечение компьютера, системы телекоммуникаций, OSI, физические и химические системы, социальные системы.

Событие - это действие в системе, которое при использовании формализма сетей Петри моделируется переходами.

Условие - это предикат или логическое описание системы; условие моделируется позициями. Различают предусловие и постусловие.

Предусловие - это условие до срабатывания перехода.

Постусловие - условие после срабатывания перехода. Если процесс в системе достаточно сложный, то его подсистемы можно представить в виде непримитивных событий. (рисунок 15) В сети появился составной переход t_2 , который отображает функционирование непримитивного события, моделируемого отдельной сетью Петри. Практически это иерархическая сеть Петри.

Одновременность. Если некоторые переходы t_i и t_j не влияют друг на друга, то в возможный словарь языка сети Петри входят как слова, начинающиеся с перехода t_i , так и слова, начинающиеся с перехода t_j .

В сложных системах часто возникают конфликты.

Конфликт: ситуация, когда переходы t_i и t_j находятся в конфликте, если запуск одного из них блокирует запуск другого

Промоделируем с помощью сетей Петри простейшую систему массового обслуживания (рисунок 17). СМО, на нее поступает задание и задание выводится. Входной поток заданий, пока система занята выполнением очередного задания, она не может ввести следующее задание. Множество условий и событий,

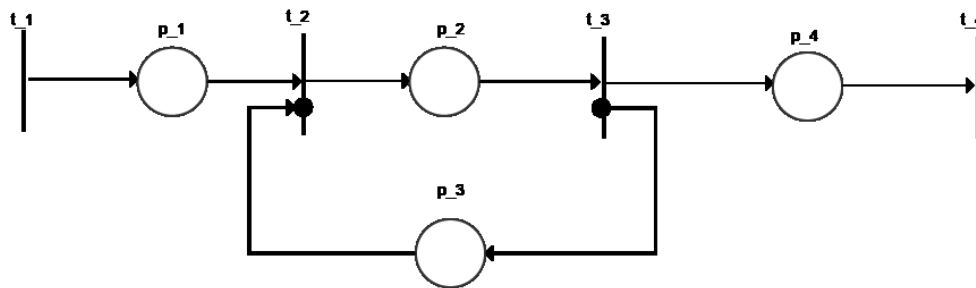


Figure 17:

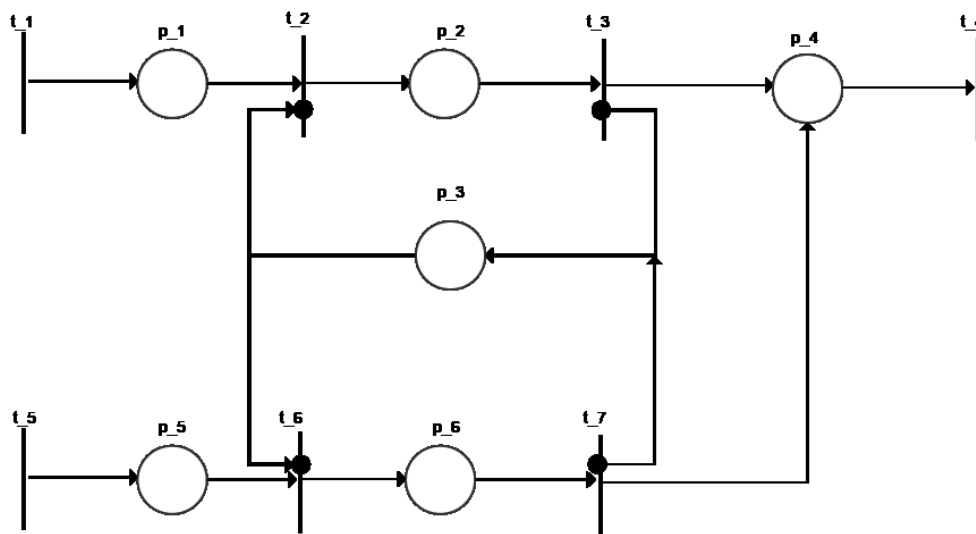


Figure 18:

которые характеризуют выполнение системы: p_1 - задание ждет обработки, p_2 - задание обрабатывается, p_3 - процессор свободен, p_4 - ожидает вывод. События: t_1 - задание помещается во входную очередь, t_2 - начало выполнения задания, t_3 - конец выполнения задания, t_4 - задание выводится.

$M_0 = [0 \ 0 \ 1 \ 0]$; $M_1 = [1 \ 0 \ 1 \ 0]$; $M_2 = [0 \ 1 \ 0 \ 0]$; $M_3 = [0 \ 0 \ 1 \ 1]$

Двухпоточная СМО (рисунок 18). СМО выполняет задания, поступающие от двух источников и находящиеся в двух очередях. Вывод обработанных заданий осуществляется одним потоком. Условие p_5 - задание из второй очереди ждет обработки, p_6 - задание из второй очереди обрабатывается. Дополнительные события: t_5 - помещается на вторую очередь, t_6 - начало выполнения задания из второй очереди, t_7 - завершение выполнения задания из второй очереди.

Конвейер. Сложение двух чисел в виде мантисса и порядок (рисунок 19). Числа должны быть нормализованы. Каждый этап выполняется отдельным устройством. Связь между функциональными устройствами и синхронизация их работы осуществляются двумя регистрами.

Условия для i -ого функционального устройства. P_{i1} - входной регистр свободен, P_{i2} - входной регистр заполнен, P_{i3} - блок занят, P_{i4} - выходной регистр свободен, P_{i5} - выходной регистр свободен, P_{i6} - пересылка в следующий блок возможна. Событие t_{i1} - начало работы i -ого блока, t_{i2} - завершение работы i -ого блока, t_{i3} - начало пересылки в $i+1$ блок, t_{i4} - завершение пересылки в $i+1$ блок. полностью сделать эту сеть

Языки моделирования (15.12.2014)

Языки моделирования помогают при разработке реализации и анализа характеристик модели. Качество языков моделирования характеризуется:

1. Удобством описания процесса функционирования
2. Удобством ввода используемых данных. Варьированием структур, алгоритмов работы, параметров

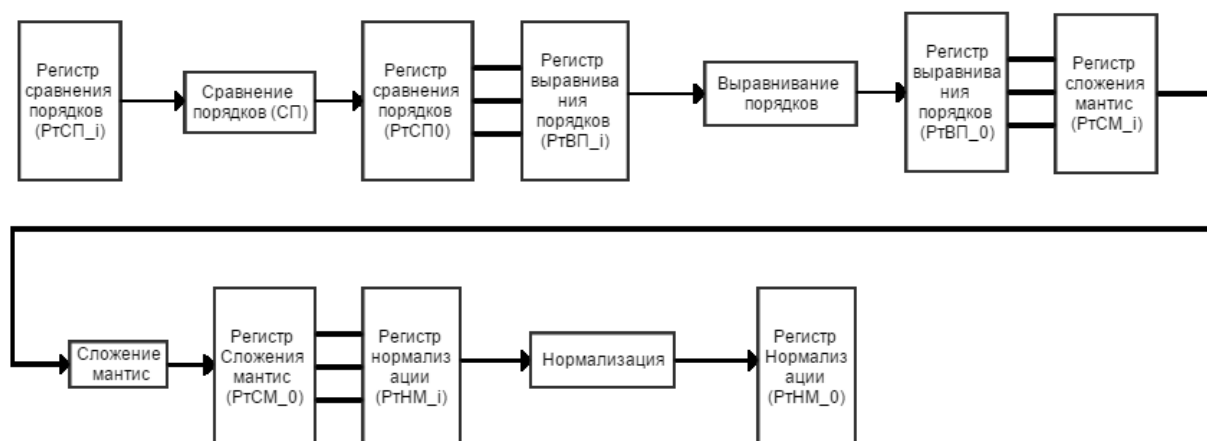


Figure 19:

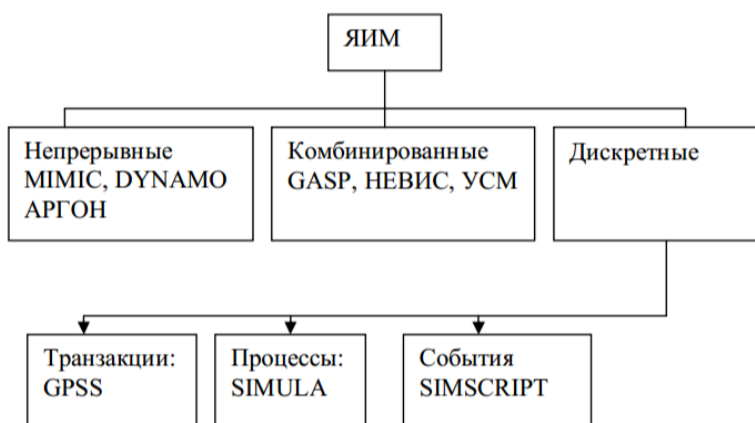


Figure 20: Классификация языков моделирования по принципу формирования системного времени.

модели

3. Эффективностью анализа и выводом результатов
4. Простотой отладки и контроля моделирующей программы
5. Доступностью восприятия и использования языка

Все современные языки в качестве управляющей программы реализуют либо событийный поток моделирования, либо

Классификация языков моделирования по принципу формирования системного времени.

Непрерывное представление систем сводится, как правило к составлению дифференциальных уравнений, с помощью которых осуществляется вход-выход моделей. Если выходные переменные модели имеют дискретный характер, то уравнения являются разностными.

Комбинированный подход - GASP. Предполагается, что в системе могут наступать события двух типов: зависящие от состояния, зависящие от времени. Состояния системы описываются некоторым набором переменных, причем некоторые из них меняются непрерывно. То есть, состояния - последовательность смен времен и дифференциальные уравнения, описывающие суть процесса.

Формальное описание динамики моделируемого процесса (объекта).

Будем считать, что любая работа в системе совершается путем выполнения активностей. Активность является наименьшей единицей работы и ее рассматривают, как единый дискретный шаг. Активность является динамическим объектом, указывающим на совершение единицы работы. Процесс - логически связанный набор активностей. Активность проявляется в результате совершения событий. Событие - мгновенные изменения состояния некоторого объекта системы. Рассмотренные объекты, активности, процессы и события являются конструктивными элементами для динамического описания поведения системы. В то время, когда динамическое поведение системы формируется в результате изменения

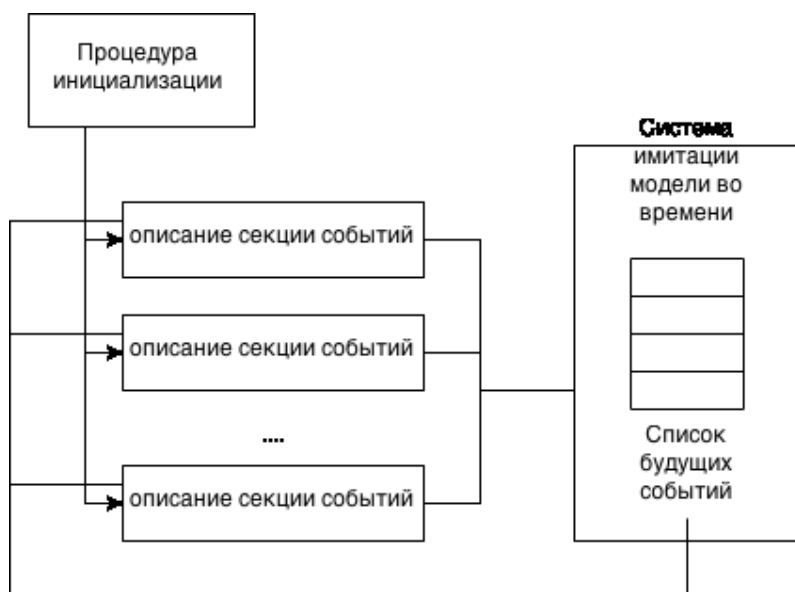


Figure 21:

большого числа взаимодействующих процессов, сами эти процессы образуют относительно небольшое число классов. Чтобы описать поведение систем, достаточно описать поведение каждого класса процессов и задать значения атрибутов для конкретных процессов. Построение модели состоит из решения следующих двух задач:

1. Первая задача сводится к тому, чтобы описать правила, задающие виды процессов, происходящих в системе.
2. Указать значения атрибутов таких процессов или задать правила генерации этих значений.

При этом система описывается на определенном уровне детализации в терминах множества описаний процессов, каждый из которых включает из себя множество правил и условий возбуждения активностей. Такое описание системы может быть детализировано на более подробном уровне представления с помощью декомпозиции процессов активностей. Тем самым мы реализуем иерархическое исследование. В общем случае, модель служит для отображения временного поведения системы. В языке моделирования должен присутствовать механизм продвижения времени. В реальной системе совместно выполняется несколько активностей, принадлежащих как связанным, так и несвязанным процессам. В соответствии с алгоритмом функционирования имитация всех действий происходит в строгой последовательности. Таким образом, модель системы можно рассматривать как взаимодействие (описание взаимодействий) активностей, событий и процессов. Языки моделирования можно разделить по данному классификатору.

Языки, ориентированные на события. Моделирующая программа организована в виде совокупности секции событий (процедуры событий). Каждая процедура состоит из набора операций, которые в общем случае выполняются после завершения какой-либо активности. Выполнение процедуры синхронизируется во времени списком будущих событий. (рисунок 21 - программа моделирования)

Языки, ориентированные на процессы Описание набора процессов. Каждый из этих наборов описывает один класс процессов. Описания процесса функционирования устанавливает атрибуты активностей всех процессов. Синхронизация операций во времени в языках, ориентированных на процессы, реализуется так же с помощью списка будущих событий, в которых содержит точку возобновления конкретного процесса или точку прерывания. (рисунок 22 - структура программы, ориентированной на процесс).

SIMSCRIPT, SIMULA, GPSS, C, Pascal - простота моделирования.

Сравнение универсальных и специализированных языков. Универсальные языки. Преимущества: минимум ограничений на выходной формат, широкое распространение. Недостатки: значительное время затрачивается на процесс программирования и отладку.

Специализированные языки. Преимущества: существенные меньшие затраты на программирование; более эффективные методы выявления ошибок; краткость, точность выражения понятий, характеризующих имитируемые процессы; возможность заранее строить библиотеки стандартных действий; автоматическое

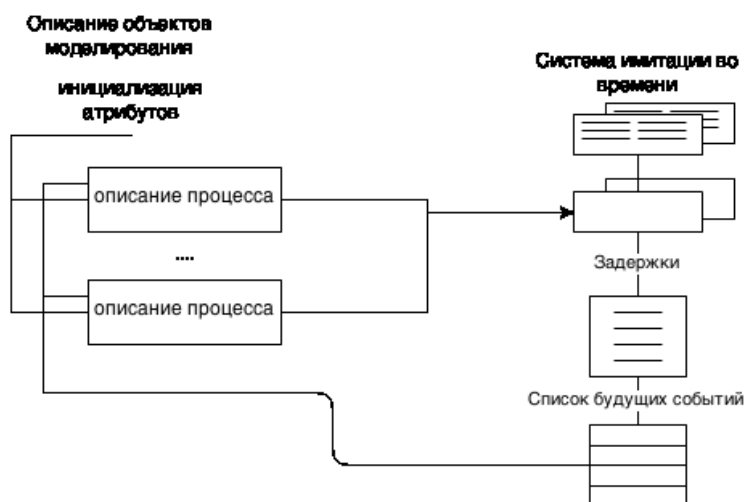


Figure 22:

формирование определенных типов данных; обеспечение управления и контроля над ресурсами вычислительных блоков. Недостатки: необходимость твердо придерживаться документации - ограничение на выходные форматы; меньшая гибкость модели.

Возможности современного ПО:

- Анимация и динамическая графика. Анимацию используют для представления сути имитационной модели, для отладки моделирующей программы, для обучения обслуживающего персонала. Существуют два основных вида анимации. Совместная анимация осуществляется во время прогона имитационной модели. Раздельная анимация: состояние системы сохраняется в отдельном файле и используется для управления графикой после завершения процесса моделирования
- Статистические возможности. Наличие генератора псевдослучайных чисел.

Объектно-ориентированные языки. Modsim 3 и выше, Simple 2+. Преимущества: обеспечивают возможность повторного использования объектов и удобный способ их изменения; реализация иерархического подхода для сложных структур; упрощает изменение модели. Недостатки: сложность согласования с заказчиком.