

Примеры:

Схема с конденсатором (рис. 1)

Маятник (рис. 2.)

Полная модель (с внутр. и внешними воздействиями):

Рис. 3

Формализация и алгоритмизация процесса функционирования сложных систем

Сущность компьютерного моделирования состоит в проведении эксперимента с моделью, которая обычно представляет собой некоторый программный комплекс, описывающий формально или алгоритмически поведение элементов в системе в процессе ее функционирования, т.е. во взаимодействии друг с другом и с внешней средой.

Основные требования, предъявляемые к модели, отображающей функционирование некоторой 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. При чем структура должна быть блочной – допускать возможность замены, добавления, исключения некоторых частей без переделки всей модели.
3. *машинная реализация модели* – должна соответствовать имеющимся ресурсам. Ресурсы - технические, экономические.
Пример: автомобиль, критерий – мин. расход топлива. В результате получаем... инвалидную коляску из Германии ☺

Процесс моделирования включает 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 решения поставленной задачи (исследования и проектирования).

Основные этапы моделирования больших систем:

1. Построение концептуальной (описательной) модели системы и ее формализация
2. Алгоритмизация модели и ее компьютерная реализация.
3. Получение и интерпретация результатов моделирования

1 этап

Формулируется модель и строится ее формальная схема.

Основное назначение – переход от содержательного описания объекта к его математической модели. Исходный материал – описание.

Последовательность действий

1. Проведение границы между системой и внешней средой
2. Исследование объекта с точки зрения основных составляющих процесса

3. Переход от содержательного описания системы к формализованному описанию свойств процесса функционирования – непосредственно к концептуальной модели. Переход от описания к модели сводится к исключению из рассмотрения некоторых второстепенных элементов описания. Причем полагается, что они не оказывают существенного влияния на ход процесса.
То, что осталось, разбивается на группы:
Блоки 1 группы – имитатор воздействия внешней среды.
Блоки 2 группы – модель процесса функционирования.
Блоки 3 группы – вспомогательные, служат для машинной реализации 1 и 2 групп.
4. Процесс функционирования системы так разбивается на подпроцессы, чтобы построение модели отдельных процессов было элементарно и не вызывало особых затруднений.

2 этап

Математическая модель реализуется в конкретную программу. Основной этап – блочная логическая схема.

Последовательность действий

1. Разработка схемы моделирующего механизма
2. Разработка схемы программы.
3. Выбор технических средств для реализации компьютерной программы
4. Программирование, отладка
5. Проверка достоверности программы на тестовых примерах.
6. Составление технической документации – логические схемы, схемы программы, текст, спецификация, иллюстрации и т.д.

3 этап

Компьютер используется для проведения рабочих расчетов по готовой программе. Результаты этих расчетов позволяют проанализировать и сделать выводы по характеристикам процесса функционирования исследуемой системы.

Последовательность действий

1. Планирования машинного эксперимента (активный/пассивный)
Составление плана проведения эксперимента, с указанием переменных и параметров, для которых должен проводиться эксперимент.
Главная задача – дать максимальный объем информации об объекте при минимальных затратах машинного времени.
2. Проведений рабочих расчетов (контрольная калибровка модели).
3. Статистическая обработка результатов и их интерпретация.
4. Составления отчетов

Различают стратегическое и тактическое планирования машинного эксперимента.

При *стратегическом планировании* ставится задача построения оптимального плана эксперимента для достижения цели, поставленной перед моделированием (оптимизация структуры, алгоритмов и параметров).

Тактическое планирование преследует частные цели оптимальной реализации каждого конкретного эксперимента из множества необходимых – оно задается при стратегическом планировании.

Схема интерактивной структуры калибровки

рис. 4.

3 основных класса ошибок:

1. ошибки формализации – неполная, недостаточная модель
2. ошибки решения – некорректный или слишком упрощенный метод построения модели.
3. ошибки задания параметров модели

Проверка адекватности и корректировка модели

Проверка адекватности модели в некоторой системе заключается в анализе ее соразмерности, а также равнозначности. Адекватность нарушается из-за идеализации внешних условий и режима функционирования, пренебрежением некоторых случайных факторов.

Простейшая мера адекватности – рис. 5.

Считают, что модель адекватна с системой, если вероятность того, что отклонение $\delta(y)$ не превышает предельной величины δ , больше допустимой вероятности P_d

Рис. 6.

Практически использовать невозможно, т.к.

1. для проектирования или модернизирования системы отсутствует информация у-объекта
2. система оценивается не по одной, а по множеству характеристик. Они могут быть случайными величинами.
3. Отсутствует возможность априорного точного задания предельных отклонений δ и допустимых вероятностей.

На практике оценка адекватности производится путем экспертного анализа *разумности результатов моделирования*. Выделяют следующие виды проверок:

1. Проверка модели элементов
2. Проверка модели внешних воздействий
3. Проверка концептуальной модели – ошибки в постановке задач.
4. Проверка формализованной и математической модели
5. Проверка способов вычисления выходных характеристик
6. Проверка программной модели.

Если по результатам проверки адекватности модели появляются недопустимые рассогласования, то возникает необходимость в калибровке или корректировке.

Выделяют следующие виды изменений.

1. Глобальные – возникают в случае обнаружения методических ошибок концептуальной или методической модели.
Исправления сводится к исправлению модели
2. Локальные, связанные с уточнение параметров и алгоритмов. Эти изменения выполняются путем замены модели на эквивалентные, но более точные.

3. Параметрические изменения некоторых специальных калибровочных параметров. Следует заранее выявить эти «влияющие» параметры и предусмотреть простые способы их варьирования.

Завершается определением и фиксацией области пригодности модели, под которой будем понимать множество условий, при соблюдении которых точность моделирования остается в рамках допустимой.

Схема взаимосвязей технологических этапов моделирования

рис. 7.

Вычислительная система как объект моделирования

В практике проектирования выявляют уровни проектирования:

1. *Системный уровень проектирования.*
В качестве элементов – процессор, внешние устройства
2. *ФЛУП – функционально логический уровень проектирования*
 - РП – регистровые передачи
 - ЛУ – логический уровень
3. *Схемотехнический – вероятностный конечный автомат.*
4. *Конструкторский – рассматривается конструкция системы. Важнейшие вопросы – охлаждение, надежность, размещение элементов на платах и т.д.*

Моделирование на системном уровне – под вычислительной системой будем понимать комплекс аппаратных и программных сред, которые в совокупности выполняют определенные рабочие функции.

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

Коллектив пользователей – сообщество людей, которые используют систему для удовлетворения своих нужд по обработке информации. Входные сигналы – программы, данные, команды, которые создаются коллективом пользователей, называются рабочей нагрузкой.

Термин установка – комплекс из определенной среды.

Схема вычислительной установки.

рис. 8.

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

Легкость использования системы, мощность системы команд.

Основные количественные индексы:

1. Пропускная способность – объем информации, обрабатываемой в единицу времени
2. Время реакции – время между предъявлением системе входных данных и выявлением соответствующей выходной информации
3. Коэффициент использования оборудования – отношение времени использования указанной части оборудования в течении заданного интервала времени, длительности этого интервала.

Концептуальная модель включает в себя:

сведения о выходных и конструктивных параметрах системы, ее структуре, особенности работы каждого элемента – ресурса, характере взаимодействия между ресурсами. Сюда же относится постановка прикладной задачи – цель моделирования, а также цель моделирования. Формализация процесса функционирования такой системы отображается например СМО – системах массового обслуживания.

Основные задачи, решаемые при моделировании вычислительных систем:

1. Определение принципов организации вычислительных систем.
2. Выбор архитектуры, уточнение функций вычислительной системы и их разделение на подфункции, реализуемые аппаратно или программным способом.
3. Разработка структурной схемы – определение состава устройств, и способов их взаимодействия.
4. Определение требований к выходным параметрам устройств
5. Формирование ТЗ на разработку отдельных устройств.

Непрерывно стохастические модели (Q-схемы)

В этом случае используемая система формализуется как некоторая система обслуживания. Характерным для таких объектов является случайное появление заявок (требований) на обслуживание и завершение обслуживания в случайные моменты времени.

В любом элементарном акте обслуживания можно выделить 2 основные составляющие

1. ожидание обслуживания
2. собственно обслуживание

Можно изобразить в виде некоторого прибора обслуживания.

рис 9.

Поток событий – последовательность событий, происходящих одно за другим.

Однородный - только моментами поступления этих событий. Задается временной последовательностью

Неоднородный – если он задается не только множеством вызывающих моментов, но и набором признаков-событий (наличие приоритета).

Если интервалы времени между сообщениями независимы между собой и являются СВ, то такой поток – поток с ограниченным последствием.

Поток событий – *ординарный*, если вероятность того, что на малый интервал времени dt , примыкающий к моменту времени t попадает больше одного события пренебрежительно

мала по сравнению с вероятностью того, что на этот же интервал попадает ровно одно событие.

Поток – *стационарный*, если вероятность появления того или иного числа событий на интервале времени зависит лишь от длины этого участка и не зависит от того, где на оси времени взят этот участок.

Пример

Для ординарного потока среднее число сообщений, поступающих за интервал dt , примыкающее к моменту времени t

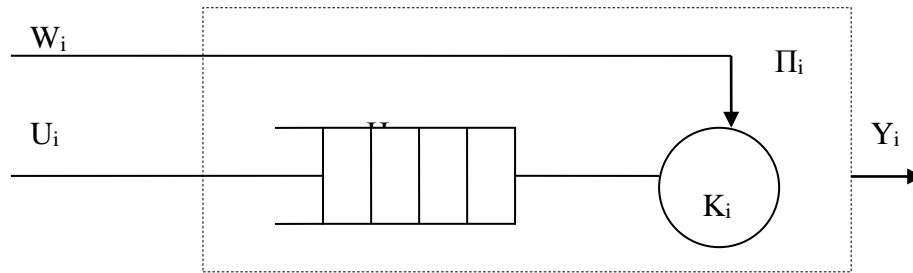
Условие
рис. 10.

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

Базовый – прибор обслуживания.

В любом элементарном акте обслуживания можно выделить 2 основные составляющие:

1. собственно, обслуживание
2. ожидание обслуживания заявки



К – канал, Н – накопитель, П – прибор обслуживания

В i -ом приборе обслуживания имеем:

- w_i – поток заявок т.е. *интервалы времени* между моментами появления заявок (вызывающие моменты) на входе канала K_i .
- u_i – поток обслуживания – интервалы времени между началом и окончанием обслуживания заявок.

W_i – интервалы времени между началом и концом времени обслуживания.

Заявки, обслуженные каналом, и заявки, покинувшие i -ый прибор не обслуженным, образуют *выходной поток* Y_i , т.е. интервалы времени между выходами заявок из системы.

Процесс функционирования i -ого прибора обслуживания можно представить как изменение состояний его элементов во времени. Переход в новое состояние для i -ого прибора означает изменение количества заявок, которые в нем находятся. Они либо в накопителе, либо в канале.

$$\vec{Z}_i(t) = (Z_i^H, Z_i^K, t)$$

$$Z_i^H = \begin{cases} 0 - \text{накопитель_свободен} \\ 1 - \text{накопитель_занят} \end{cases} \quad Z_i^K = \begin{cases} 0 - \text{канал_свободен} \\ 1 - \text{канал_занят} \end{cases}$$

Z_i – состояние накопителя в дискретные моменты времени.

В практике моделирования, элементарные схемы обычно объединяются, при этом если каналы соединены параллельно, то имеет место *многоканальное* обслуживание; если последовательно – *многофазное* обслуживание.

Таким образом, для формализации схемы функционирования Q-схемы необходимо использовать *оператор сопряжения* – R, отражающий взаимосвязь элементов структуры между собой.

Различают разомкнутые и замкнутые Q-схемы.

В *разомкнутых* выходной поток обслуживаемых заявок не может поступить снова на какой-либо элемент структуры, т.е. отсутствует обратная связь.

В *замкнутых* системах имеется обратная связь.

Собственные параметры Q-схемы:

- количество фаз
- количество каналов в каждой фазе

- Количество накопителей в каждой фазе
- Емкость каждого накопителя

$$R, Q = \begin{cases} L_\phi \\ L_{Kj}, j = \overline{1, L_\phi} \\ L_H \\ L_i^H \end{cases}$$

В зависимости от емкости накопителя, в теории массового обслуживания исп. след. терминологию:

$$L_i^H = \begin{cases} 0 - \text{система с потерями} \\ \rightarrow \infty - \text{бесконечная емкость, очередь заявок не ограничена} \end{cases}$$

H – вся система параметров

$$H = (L_\phi, L_{Kj}, L_H, L_i^H)$$

Для задания Q-схемы также необходимо описать ее алгоритм, который определяет набор правил поведения заявок в различных ситуациях. Неоднородность заявок, отражающая реальный процесс в той или иной мере с введением понятий классов и приоритетов.

Различают – *статические, динамические, относительные, абсолютные* приоритеты. При работе главная задача – минимально загрузить процессор.

Весь набор возможных алгоритмов поведения заявок можно представить в виде некоторого оператора алгоритма – A поведения заявок. Таким образом Q-схема, описывающая процесс функционирования системы массового обслуживания любой сложности, однозначно задается в виде:

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

W – подмножество входящих потоков

U – подмножество потоков обслуживания

R – оператор сопряжения элементов в системе

H – подмножество собственных параметров

Z – множество со стояний системы

A – оператор алгоритмов обслуживания заявок

Возможности использования характеристик с использованием аналитических методов, разработанных в теории массового обслуживания, является узкоограниченными по сравнению с требованиями практического исследования таких систем. Следовательно необходимо использование имитационных методов.

Эксперимент

Задача моделирования – исследование объекта оригинала по методу черного ящика.

Факторы – критерии системы, от которых зависит поведение системы.

В процессе моделирования мы задаем значения входных факторов, получаем значения выходных. Однако могут еще присутствовать случайные воздействия, которые мы можем не учесть.

Имитационная модель – сложная модель, вероятностный характер, все характеристики имеют вероятностные значения. На вход – определенная комбинация. То, что получается на выходе, по закону функционирования, может не соответствовать. Берем отдельный выходной параметр и наблюдаем при серии экспериментов, как он меняется.

Цель планирования – обеспечение минимального числа затрат. Как проводить, сколько испытаний, интервалы времени. Минимум затрат, максимум информации об объекте. На основе исследования модели получаем *поверхность отклика*.

рис.1

Закон распределения выходного параметра.

Планирование разделяют на *тактическое* и *стратегическое*.

Тактическое планирование – условия проведения единичного эксперимента: чаще всего определение объема выборки, обеспечивающего заданную статистическую погрешность целых числовых характеристик, распределение функции отклика.

Стратегическое планирование.

Если решается задача синтеза, задача определения параметров, обеспечивающих требуемое качество функционирования исследуемой системы, необходимо знание зависимости характеристик от параметров. Эти задачи с помощью тактического пл. Как правило существует прямая и обратная задачи.

Прямая задача – проводится серия экспериментов и по их результатам необходимо определить функциональную зависимость, связывающую характеристики исследуемой системы с параметрами. При этом обычно задаются классом исследуемые зависимости.

Рис. 2.

Минимизировать количество точек.

Обратная задача – необходимо так планировать эксперимент, чтобы при минимальном числе опытов, построить экспериментальную функцию отклика (реакции), связывающую отклик (реакцию) с системой параметров или с множеством входных независимых факторов Y_i , которые можно варьировать при постановке эксперимента. X_i – переменные, Y_i – вектор отклика системы. Рис. 3

Разложение в ряд Тейлора в окрестности точки X_0 .

Наряду с управляемыми факторами, на систему могут воздействовать неуправляемые и не контролируемые факторы, в результате чего Y – случайная величина, поэтому по результатам эксперимента мы можем построить только приближенную зависимость вида

Рис. 4.

Y – математическое ожидание.

Могут служить математическими оценками теоретических коэффициентов ряда Тейлора. Таким образом наша задача – определение выборочных коэффициентов регрессии по результатам опытов в M точках пространства.

Делаем 2 допущения:

1. результаты наблюдений отклика системы в различных точках факторного пространства рассматриваются как независимые случайные величины с одинаковыми дисперсиями.
2. погрешности задания варьируемых факторов существенно меньше погрешности измерения Y .

Рис.5.

(Эксперимент - продолжение)

Рис. 1.

Необходимо организовать m экспериментов, в котором каждому F_j назначается... каждому j -ому набору эффективных переменных V_j соответствует единичный эксперимент с номером G , в котором проводится n наблюдений отклика системы. После усреднения определяется математическое ожидание отклика Y_j .

Наша задача – построение плана такого эксперимента – выбор параметра F_j . Построим матрицу.

Рис.2.

Для нахождения наименьшего решения системы уравнений, относительно коэф. V_j необходимо и достаточно, чтобы определитель матрицы C не равен 0. Это условие выполняется в случае линейной независимости столбцов матрицы M .

Если C – диагональная матрица, то решение упрощается. Рис.3.

Условие диагональности C – попарная ортогональность столбцов матрицы F . Для получения независимых оценок, коэффициентов уравнения регрессии надо так спланировать эксперимент, т.е. построить такую матрицу планирования, чтобы выполнялось условие линейной независимости и ортогональности матриц.

Тактическое планирование

Связано с вопросом эффективности. Прежде связано с :

1. определение начальных условий в той мере, в которой они влияют на эксперимент.
2. сокращение размеров выборки при уменьшении размеров дисперсии

....

Для получения данных, связывающих характеристики, описывающие функционирование q -схемы вводят некоторые допущения относительно входных потоков, функции распределения, длительностей обслуживания запросов и дисциплин обслуживания.

Для таких систем в качестве типовой модели будем рассматривать теорию Марковских процессов. Случайный процесс, протекающий в некоторой системе S называется *Марковским процессом*, если он обладает следующим свойством: для каждого момента времени t_0 вероятность любого состояния системы в будущем (при $t > t_0$), зависит только от ее состояния в настоящем ($t = t_0$), и не зависит от того, когда и каким образом система пришла в это состояние. Т.е. в таком процессе будущее развитие зависит только от настоящего состояния и не зависит от предыстории процессов. Для Марковского процесса разработаны уравнения Колмогорова:

рис 4.

Для *стационарного* распределения – рис. 5.

Отсюда выводим: $P = P(\lambda)$

Выходная функция $Y = Y(P(\lambda))$ – Данное соотношение представляет собой зависимость выходных параметров от некоторых внутренних параметров. Данное соотношение – *базисная модель*.

Необходимо осуществить связь между внутренними параметрами модели, входными параметрами, внешним воздействием. Рис. 6.:

Интерфейсная модель.

Математическая модель Q-системы строится как совокупность базисной и интерфейсной модели. Это позволяет использовать одни и те же базисные модели при решении задачи проектирования, осуществляя настройку на соответствующую задачу посредством изменения интерфейсной модели. Параметрическая настройка на конкретную задачу.

Математическая модель должна обеспечить вычисление времени реакции на запрос и определить производительность системы.

Методика вывода уравнений Колмогорова.

Рис.7.

λ_{ij} – плотность вероятности для множества состояний.

Найдем вероятности $P_1(t)$, т.е. вероятность того, что в момент времени t система будет находиться в состоянии S_1 . Придадим t малое приращение dt , и найдем вероятность того, что в момент $(t+dt)$ система будет находиться в состоянии S_1 .

Это событие может произойти двумя способами:

1. В момент t система уже находилась в состоянии S_1 и за время dt не вышла из него
 $P_1(t)(1-\lambda_2 dt)$
2. Система находилась в состоянии S_3 .
 $P_3(t)\lambda_3 dt$

$$P_1(t+dt) = P_1(t)(1-\lambda_2 dt) + P_3(t)\lambda_3 dt$$

Раскроем скобки в правой части, P_1 в левую часть и все поделить на dt .

Рис 8.

Уравнение Колмогорова для состояния S_1 .

Рассмотрим состояние S_2 . $P_2(t)$ – вероятность того, что система в S_2 .

1. В момент t была в S_2 и за dt ни перешла ни в S_3 , ни в S_4
2. Система была в состоянии S_1 и за время dt пришла к состоянию S_2
3. В момент t – S_4 , и за время dt пришла в состояние S_2 .

Интегрирование этой системы дает искомые вероятности состояний как функции времени. Начальные условия берутся в зависимости от того, какого было начальное состояние системы. Если в момент времени t система в S_1 , то начальное условие при $t = 0$: $P_1=1$, $P_2=0$, $P_3=0$, $P_4=0$ – *условие нормировки*, т.е. сумма всех вероятностей равна 1. Из структуры полученных уравнений выводим правила построения уравнения Колмогорова.

1. В левой части каждого уравнения стоит произвольное вероятностное состояние, а правая часть содержит столько частей, сколько стрелок связано с данным состоянием.
2. Если стрелка направлена из состояния, то соответствующий элемент имеет “-”
3. Каждый член уравнения равен произведению плотности вероятности перехода (*интенсивность*), соответствующей данной стрелке, умноженной на вероятность того состояния, из которого приходит стрелка.

Рассмотрим систему массового обслуживания с отказами.

Будем формировать состояние системы по числу занятых каналов (по числу заявок).

Обозначим

S_0 – все каналы свободны,

S_1 – один канал занят, остальные свободны

S_k – занято k каналов, остальные свободны

S_n – заняты все каналы.

рис. 9.

Разметим граф, проставим у стрелок интенсивности соответствующих потоков событий. По стрелкам слева направо.

Пусть система в состоянии S_1 , как только закончится обслуживание этой заявки, система перейдет в S_0 . Интенсивность из S_2 в S_1 – μ_0

Предельные вероятности состояний P_0 P_n характеризуют устоявшейся режим работы:

рис.10

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

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

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

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

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

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

$$q = 1 - p_n$$

- среднее число заявок, обслуженных в единицу времени

$$A = \lambda q$$

Полученные соотношения могут рассматриваться как *базисная модель оценки характеристик* производительности системы. Входящий в эту модель параметр $\lambda = 1 / \text{ТОБРАБОТКИ}$, является усредненной характеристикой пользователей. Параметр μ является функцией технических характеристик компьютера (вычислительных средств) и решаемых задач. Эта связь должна быть установлена с помощью интерфейсной модели.

В простейшем случае, если время ввода-вывода информации мало по сравнению со временем решения этой задачи, то можно принять, что время решения равно $1 / \mu = N_{\text{пр}} / V_{\text{пр}}$.

$N_{\text{пр}}$ – среднее число операций, выполняемых процессором при решении одной задачи
 $V_{\text{пр}}$ – среднее быстродействие процессора, измеряется в операциях в секунду.

Задание на Лабу:

определить среднее время нахождения системы в состояниях при установившемся режиме работы. Должна строиться матрица (max 10x10). Количество состояний вводится пользователем: 2 – матрица 2x2. Потом вводятся все состояния:

| | S1 | S2 |
|----|----|-----|
| S1 | 2 | 1.5 |
| S2 | 0 | 3 |

При вычислении – расчет времен пребывания в каждом состоянии.

На практике далеко не все случайные процессы являются Марковскими или близкими к ним. В СМО поток заявок не всегда Пуассоновский, ещё реже наблюдается показательное или близкое к нему распределение времени обслуживания.

Для произвольных же потоков событий, переводящих систему из состояния в состояние, аналитические решения получены только для отдельных частных случаев. Когда построение аналитической модели по той или иной причине трудно выполнимо, применяют *метод статистических испытаний (Монте-Карло)*.

Идея метода: вместо того, чтобы описывать случайное явление с помощью аналитической зависимости производится «розыгрыш», т.е. происходит моделирование случайного явления с помощью некоторой процедуры, дающей случайный результат. Произведя такой розыгрыш n раз, получаем статистический материал, т.е. множество реализаций случайного явления, которое потом можно обработать обычными методами математической статистики. Метод Монте-Карло предложил Фон-Нейман в 1948 году, как метод численного решения некоторых математических задач.

Рис 11.

Суть:

1. В единичном квадрате – любая поверхность S .
2. Любым способом получаем 2 числа: X_i, Y_i , подчиняющихся равномерному закону распределения на интервале от 0 до 1.
3. Считаем, что одно число – координаты точки по X , другое – по Y . Анализируем, принадлежит ли точка поверхности. Если да, то счетчик на 1.

4. Процедура генерации 2 случайных чисел и проверки принадлежности точки поверхности повторяется N раз
5. Площадь - как отношение количества сгенерированных к количеству попавших точек.

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

Преимущества метода статистических испытаний в его универсальности, обуславливающей возможность всестороннего статистического исследования объекта, однако, для реализации этой возможности нужны довольно полные статистические сведения о параметрах элементов.

К *недостаткам* относится большой объем требуемых вычислений, равный количеству обращений к модели. Поэтому вопрос выбора величины n имеет важнейшее значение. Уменьшая n – повышаем экономичность расчетов, но одновременно ухудшаем их точность.

Способы получения последовательностей случайных чисел.

На практике – три основных способа:

1. *аппаратный* – случайные числа вырабатываются специальной электронной приставкой, генератором, служащей как правило в качестве одного из внешних устройств компьютера. Реализация данного способа не требует дополнительных вычислительных операций по выработке случайных чисел, а необходима только операция обращения к внешнему устройству. В качестве физического эффекта, лежащего в основе таких генераторов, чаще всего используют шумы в электронных приборах.

рис 12.

2. *табличный* – случайные числа оформляются в виде таблицы и помещаются в оперативную или внешнюю память.
3. *алгоритмический* – основан на формировании случайных чисел с помощью специальных алгоритмов.

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| 1 | 2 | 3 | 1 | 2 | 3 |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

выбрать порядка 1000.

выбрать критерий, чтобы определить, какой лучший – табличный или алгоритмический.

Простейшие алгоритмы генерации последовательности псевдослучайных чисел.

Наиболее часто используются – криптография, статистика, поиск объектов из космоса (нефть, метеориты).

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

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

Если n пробегает по ряду натуральных чисел, то поведение Y_n выглядит весьма хаотично. Карл Якоби доказал, что при рациональном a , множество значений Y_n конечно, а при иррациональном – бесконечно и всюду плотно в интервале $[0,1]$.

Критерий равномерности распределения любой функции от натурального ряда чисел. Это свойство *эргатичности* – среднее по реализации псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1. Такие результаты далеки от практики получения последовательностей псевдослучайных чисел, т.к. теорема Якоби относится к действительным числам, которые не могут быть использованы при вычислении, потому что иррациональные действительные числа требуют для своей записи бесконечного числа знаков.

Попытки замены настоящего иррационального числа его приближением на компьютере опасна, т.к. полученные последовательности оканчиваются циклом с коротким периодом.

Способы генерации:

1. 1946 год. Фон Нейман. Каждое последующее случайное число образуется возведением предыдущего в квадрат и отбрасыванием цифр с обоих концов. Не выдержал проверки
2. Лемер. $g_{n+1} = kg_n + (c \bmod m)$
Для подбора коэффициентов потрачены десятки лет. Подбор почти иррациональных почти ничего не дает.
Выбрав закон генерации при просчете с плавающей точкой 4 байта, получают псевдослучайные числа, обязательно заканчивающиеся циклами с периодом длиной всего лишь 1225 или 147, в зависимости от начального заполнения.
3. Разумнее в целых числах. Установлено, что при $c=0$ и $m=2^n$, наибольший период достигается при $k=3+8i$ или $k=5+8i$ и при нечетном начальном числе.
По данному алгоритму в 1972 IBM на 360 разработала программу. А в 1977 году Форсайд показал, что тройки чисел такой последовательности лежат на 15 параллельных плоскостях.

Лучше использовать несколько генераторов и смешивать их значения. Если разные генераторы независимы, то сумма их последовательностей обладает дисперсией равной сумме дисперсий.

В системах программирования обычно используют конгруэнтные генераторы, по алгоритму предложенному в США.

4. Метод целочисленной арифметики. Работу написал Зейнеман. Использовали Фибоначчи, брали в своем алгоритме только последнюю цифру числа и получали последовательности.


```
RANDOMIZE 231
Y=RND
RANDOMIZE 231
X=RND
```

X не равно Y, т.к. устанавливаются только 2 байта.

Для увеличения периода последовательности, используют

```
FUNCTION Rand(x, y)
  x = RND(-x)
  y = RND(-y)
  IF y = 0 THEN y = RND(-y)
  RAND = (x+y) mod 1
END FUNCTION
```

Период функции 2^{24} ($2^{41} - 1$).

Но свойства этого ряда не будут такими же, как у X, Y.

При создании с помощью встроенного генератора случайных объектов, имеющих число состояний больше, чем у генератора, его приходится использовать несколько раз, переустанавливая по заранее заданному ключу.

```
FOR I = 1 TO 5
  X = RND(-gamma(i))
  FOR J = 0 TO 32
    SWAP map(j), map(32xRND)
  NEXT J
NEXT I
```

Здесь происходит перестановка 33-х элементов массива map, которая может быть сделана 2^{11} способами. При длине генератора 2^{24} надо запустить не менее 5 раз, чтобы реализовать все варианты перестановки.

Для получения значения случайной величины из последовательности случайных чисел с заданным законом распределения, обычно используют одно или несколько значений равномерно распределенных случайных чисел. Псевдослучайные равномерно распределенные числа получают в компьютере программным способом с помощью некоторого рекуррентного соотношения. Это означает, что каждое последующее число образуется из предыдущего (или из группы предыдущих) путем реализации некоторого алгоритма, состоящего из арифметических и логических операций.

Процедура на языке Фортран, предназначенная для последовательности равномерно распределенных случайных чисел на интервале $[0,1]$ с помощью мультипликативного конгруэнтного метода для 32 разрядного компьютера.

```
SUBROUTINE RANDUM(IX, IY, RN)
  IY = IX*1220703125
  IF (IY) 3,4,4
  IY = IY+2147483647+1 // метод 3
  RN = IX // метод 4
  RN = RN * 0.4656613E-9
```

```
IX = IY  
RETURN  
END
```

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

< code here >

выдаст 200 чисел, Rand(1) – равномерно распределенные, Rand(2) – Гауссово.

Для имитации равномерного распределения на интервале [a,b] используется обратное преобразование.

рис.1.

В основе построения программы, генерирующей случайные числа, с законом отличным от равномерного, лежит метод преобразования последовательности случайных чисел с равномерным законом распределения в последовательность случайных чисел с заданным законом распределения. Метод основан на теореме, утверждающей, что некоторая случайная величина X: рис.2 имеет плотность распределения F(x), где R – случайная величина, равномерно распределенная в интервале [0,1]. Значение случайной величины, распределенной по показательному закону, может быть вычислено по рис.3

Распределение Пуассона.

Относится к числу дискретных, т.е. таких, при которых переменная может принимать лишь целое значение с математическим ожиданием и дисперсией равными lambda. Для генерируемых пуассоновых переменных, используя метод Точера, в основе которого лежит генерируемая случайная переменная Ri, равномерно распределенная до тех пор, пока не станет справедливо следующее соотношение. рис.4.

При получении случайной величины, функция распределения которой не позволяет найти решения уравнения в явной форме, можно произвести кусочно-линейную аппроксимацию и затем вычислить приближенные значения корня. Кроме того, при получении случайной величины часто используют те или иные свойства распределения.

Распределение Эрланга – 2 параметра: lambda, k. При вычислении случайной величины, воспользуемся тем, что поток Эрланга может быть получен прореживанием k раз. Или же потока Пуассона. Т.е. достаточно получить k значений случайной величины, распределенной по показателям и усреднить их.

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

Нормально распределенная случайная величина может быть получена как сумма большого числа случайных величин, распределенных по одному и тому же закону и одними и теми же параметрами.

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

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

Для сокращения вычислений на практике принимают $N=12$, что дает хорошее приближение.

Лаба 4.

Классификация систем массового обслуживания.

СМО классифицируются по следующим принципам:

1. закон распределения заявок
2. числу обслуживающих приборов
3. закону распределения времени обслуживания обслуживающих приборов
4. числу мест в очереди
5. дисциплине обслуживания

Для краткости записи при обозначении любой системы массового обслуживания:

A/B/C/D/E

где на место буквы ставится характеристики.

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

M – экспоненциальные

E - эрландовские

H – гиперэкспоненциальные

G - гамма-распределение

D - детерминированное распределение

P – произвольное

B – Закон распределения времени обслуживания приборов СМО.
приняты такие же обозначения

C – число обслуживающих приборов.

для одноканальных систем – 1

для многоканальных в общем случае – l

D – число мест в очереди

если число мест в очереди неограниченно, то оно может опускаться. Для конечного числа мест в очереди – R/N.

E - Дисциплина обслуживания. По умолчанию LIFO – в этом случае поле может опускаться.

Примеры:

M/M/1 – СМО с одним прибором, бесконечной очередью, экспоненциальным законом распределения, интервалом времени между поступлением заявок и временем обслуживания, дисциплина обслуживания FIFO.

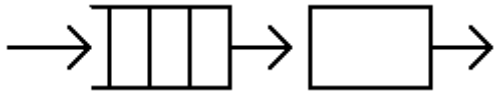
Е/Н/1/τ/LIFO - СМО с несколькими обслуживающими приборами, конечной очередью, законом распределения Эрланга интервалов между поступлением заявок, гиперэкспоненциальным законом распределения времени обслуживания заявок в приборах и дисциплиной обслуживания LIFO.

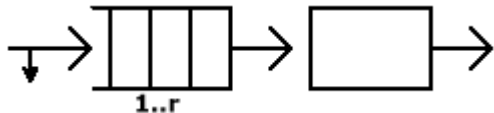
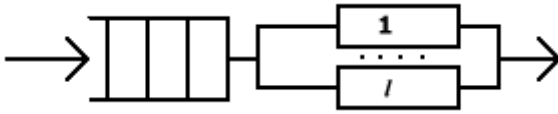
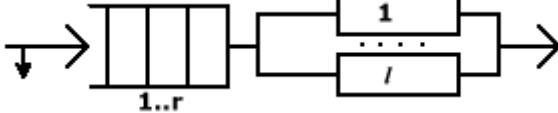
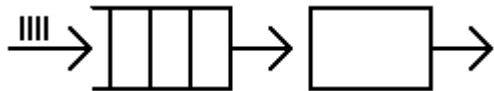

G / G / I

СМО с несколькими приборами, бесконечной очередью, произвольными законами распределения времени между поступлением заявок и времени обслуживания, FIFO.

Для моделирования вычислительной системы наиболее часто используются следующие типы СМО:

1. **Одноканальная СМО с ожиданием.**
 - один обслуживающий прибор с бесконечной очередью
 - структура является наиболее распространенной при моделировании
 - с той или иной степенью приближения с ее помощью можно моделировать практически любой узел вычислительной системы или ЛВС
2. **Одноканальная СМО с потерями.**
 - 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 передач данных в ВС и ЛВС
3. **Многоканальная СМО с ожиданием.**
 - несколько параллельно работающих обслуживающих приборов с общей бесконечной очередью
 - используется при моделировании групп абонентских терминалов, работающих в диалоговом режиме
4. **Многоканальная СМО с потерями.**
 - несколько параллельно работающих обслуживающих приборов с общей очередью, число мест в которой ограничено
 - часто используется, как и (2), при моделировании каналов связи
5. **Одноканальная СМО с групповым поступлением заявок**
 - один обслуживающий прибор с бесконечной очередью
 - перед обслуживанием заявки группируются в пакеты по определенному признаку или правилам
 - используется для моделирования центров коммутации
6. **Одноканальная СМО с групповым обслуживанием заявок**
 - один обслуживающий прибор с бесконечной очередью
 - заявки обслуживаются пакетами, составленными по определенному правилу
 - используется для моделирования центров коммутации

| <i>Наименование</i> | <i>Обозначение</i> | <i>Схема</i> |
|---------------------------|--------------------|--|
| Одноканальная с ожиданием | <i>G / G / I</i> |  |

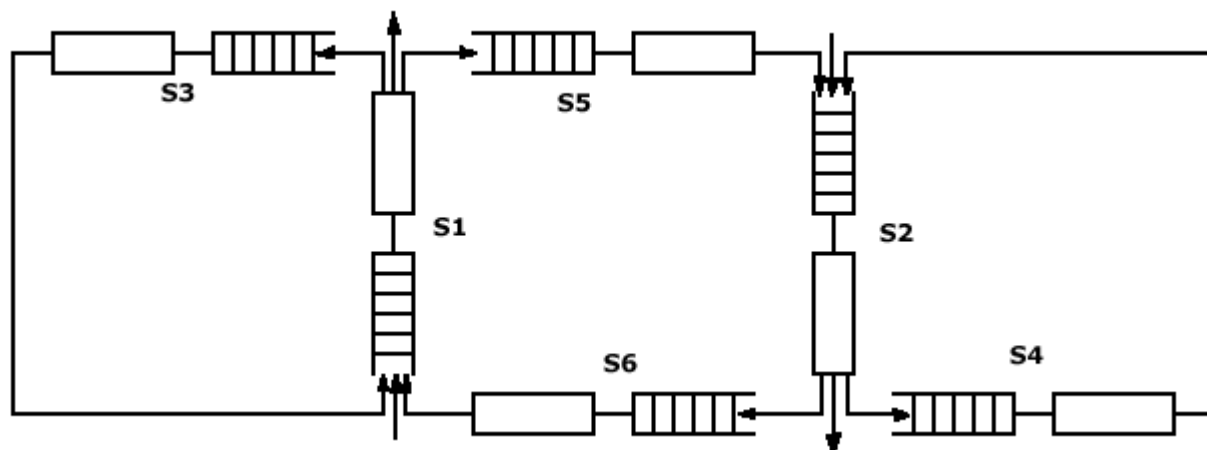
| | | |
|--|------------------|--|
| Одноканальная с потерями | $G / G / 1 / r$ |  |
| Многоканальная с ожиданием | $G / G / l$ |  |
| Многоканальная с потерями | $G / G / l / r$ |  |
| Одноканальная с групповым поступлением заявок | $Gr / G / 1 / R$ |  |
| Одноканальная с групповым обслуживанием заявок | $G / Gr / 1 / R$ |  |

Вычислительные сети в целом могут быть представлены в виде СМО. Различают следующие типы сетей:

1. **Открытые**
2. **Замкнутые**
3. **Смешанные**

Открытой называется СМО, состоящая из m узлов, причем хотя бы в один из узлов сети поступают извне входной поток заявок и обязательно имеется хотя бы один сток заявок из сети.

Для открытых сетей характерно то, что интенсивность поступления заявок не зависит от состояния системы, т.е. от числа заявок уже поступивших в сеть. Такие сети как правило используются для моделируемых ВС, работающих в неоперативном режиме.



S1, S2 – моделируют работу узлов коммутации
S3, S4 – работу сервера
S5, S6 – работу межузловых каналов

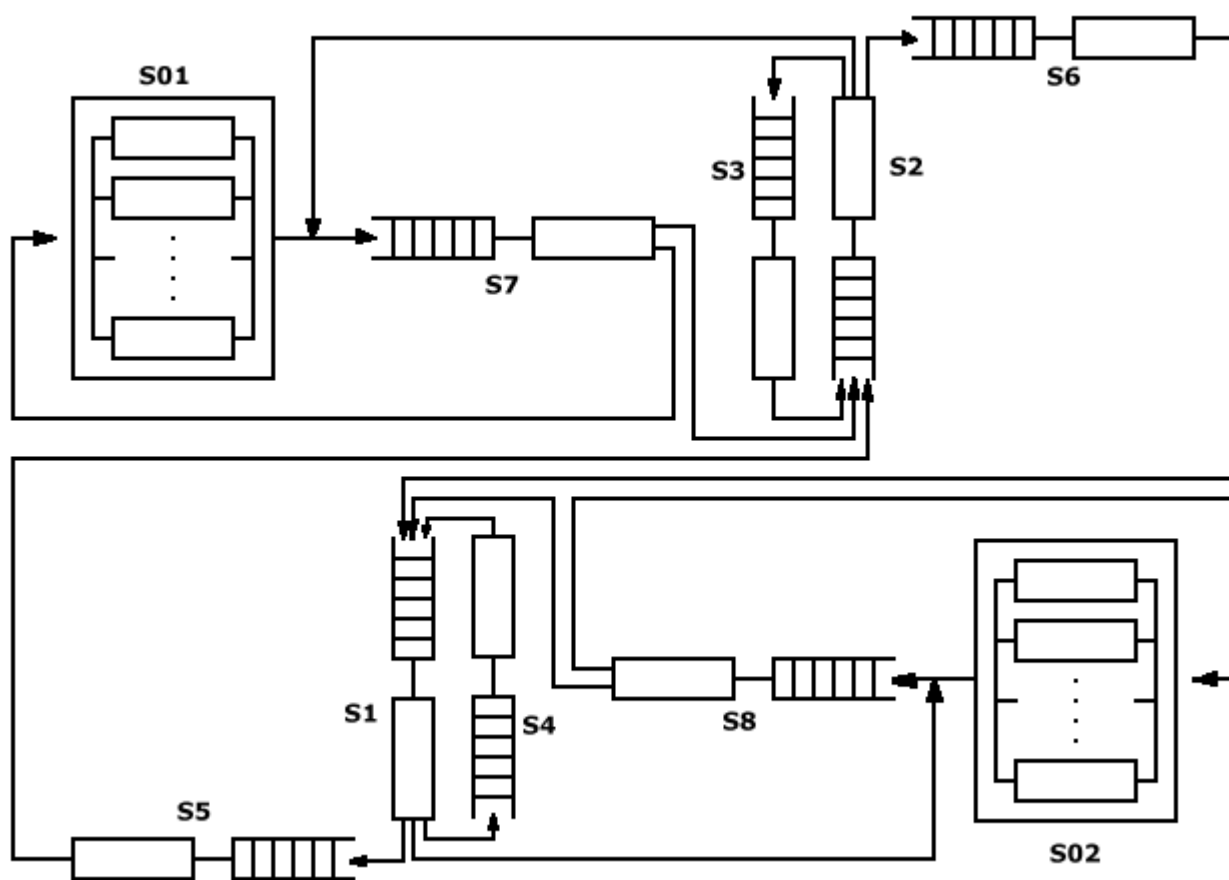
В сети циркулируют 2 потока заявок:

Каждая заявка поступает на вход соответствующего узла коммутации, где определяется место обработки. Затем заявка передается на «свой» сервер или по каналу связи на соседний, где обрабатывается, после чего возвращается к источнику и покидает сеть.

Замкнутой называется СМО с множеством узлов m без источника и стока, в которой циркулирует постоянное число заявок.

Замкнутые СМО используются для моделирования таких ВС, источниками информации для которых служит абонентский терминал, работающий в диалоговом режиме. В этом случае каждая группа абонентских терминалов представляется в виде многоканальной СМО с ожиданием и включается в состав устройств сети.

Простой режим работы диалоговых абонентов: абоненты не производят никаких действий, кроме отправки заданий в вычислительную систему и обслуживание полученного ответа.



S01, S02 – группа абонентских терминалов

S7, S8 – каналы связи с абонентами

S1, S2 – узлы коммутации

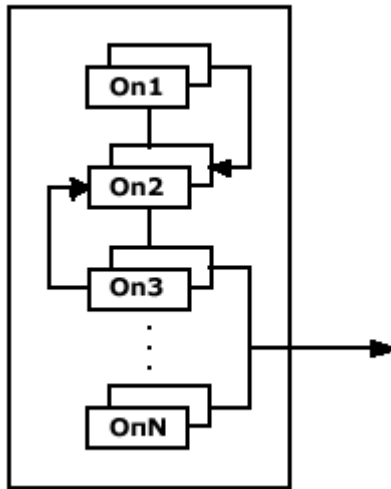
S3, S4 – серверы

S5, S6 – каналы межузловой связи

Абоненты с терминалов посылают запросы, которые по каналам связи поступают на узлы коммутатора, а оттуда на обработку на свой или соседний сервер.

При сложном режиме диалога работа абонентов представляется в виде совокупности операций некоего процесса, называемого *технологическим*. Каждая операция

технологического процесса моделируется СМО. Часть операций предусматривает обращение в ВС. Причем, часть операций может предусматривать обращение к ВС, а другая часть может замыкаться сама на себя.



Смешанной называется СМО, в которой циркулируют несколько различных типов заявок (трафика). Причем относительно одних типов заявок сеть замкнута, а относительно других – открыта.

С помощью смешанных сетей моделируются такие ВС, часть абонентов которых работает в диалоговом, а часть в неоперативном режиме. Для диалоговых абонентов также различают простой и сложный режим работы. Часто смешанные СМО моделируют ВС, в которых сервер дополнительно загружается задачей, решаемыми на фоне работы самой сети.

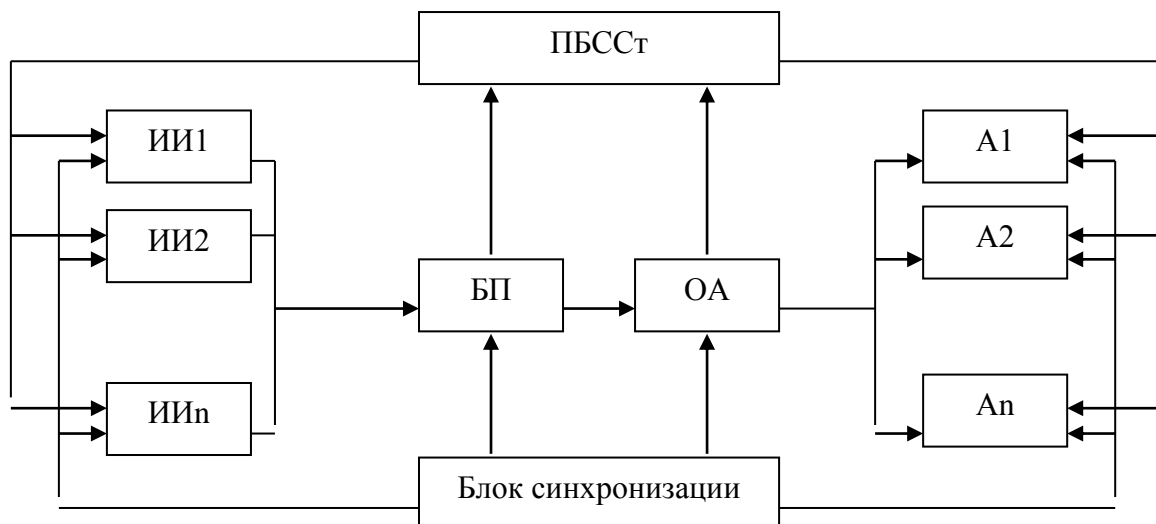
Методика построения программной модели.

Для разработки программной модели исходная система должна быть представлена как стохастическая. Это решается несколькими факторами – информация поступает от внешних признаков. Длительность обработки различных признаков информации может быть в общем случае различной. Все эти воздействия могут отображаться каким-то генератором сообщений. А весь комплекс вычислительных средств в виде обслуживающих устройств.

рис. 1.

Источники информации подают на вход буферной памяти независимо друг от друга поток сообщений. Закон появления этих сообщений произволен, но обязательно задан наперед. В буферной памяти сообщения как правило записываются при поступлении, и выбираются по одному в обслуживающий аппарат, как правило по принципу FIFO. Длительность обработки одного сообщения в обслуживающем аппарате в общем случае может быть случайной. Но закон обработки должен быть задан заранее.

Быстродействие обслуживающего аппарата ограничено, поэтому на входе как в обслуживающий аппарат, как и в буферную память (если она конечна), в общем случае возможно сложение данных.



Моделирование потока сообщений.

Поток сообщений обычно лимитируется моментами появления очередного сообщения в потоке.

Текущий момент времени появления очередного сообщения:

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

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

Процедура.

обращение к процедуре выражения случайного числа Rnd

$$T_i = -\frac{1}{\lambda} \ln(1 - R_i)$$

$$T = T + T_i$$

| Вид распределения | Выражение |
|----------------------|---|
| равномерное на [a,b] | $T_i = a + (b - a)R$ |
| нормальное | $T_i = \sigma_x \sqrt{\frac{12}{n}} (\sum_{i=1}^n R_i - \frac{n}{2}) + M_x$ |
| экспоненциальное | $T_i = -\frac{1}{\lambda} \ln(1 - R)$ |
| Эрланга | $T_i = \frac{1}{k\lambda} \sum_{i=1}^k \ln(1 - R_i)$ |

Моделирование работы Обслуживающего Аппарата.

Программа, имитирующая работу обслуживающего аппарата – набор процедур, вырабатывающих случайные отрезки времени, соответствующие длительностям обслуживания требований.

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

$$t_{iAD} = M_x + (\sum_{i=1}^{12} R_i - 6) \cdot \sigma_x$$

Схема алгоритма имитатора.

R_i – случайное число с равномерным законом распределения

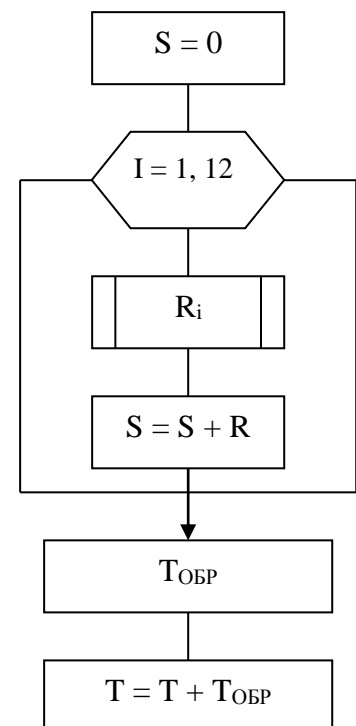
T_{OBR} – время обработки очередного сообщения

T – время освобождения ОА

XM – Мат ожидание для заданного закона обратки

DX – СКО для заданного закона обработки

$$T_{iAD} = XM + (S - 6) * DX$$



Моделирование работы абонента.

Абонента можно рассматривать как ОА, поток информации на который поступает от процессора. Для моделирования работы абонентов необходимо вырабатывать длительности обслуживания требований. Кроме того, абонент сам может быть источником заявок, претендуя на те или иные ресурсы вычислительной системы. Эти заявки могут имитироваться с помощью генератора сообщений по наперед заданному закону. Таким образом, абонент либо имитируется как ОА, либо как генератор.

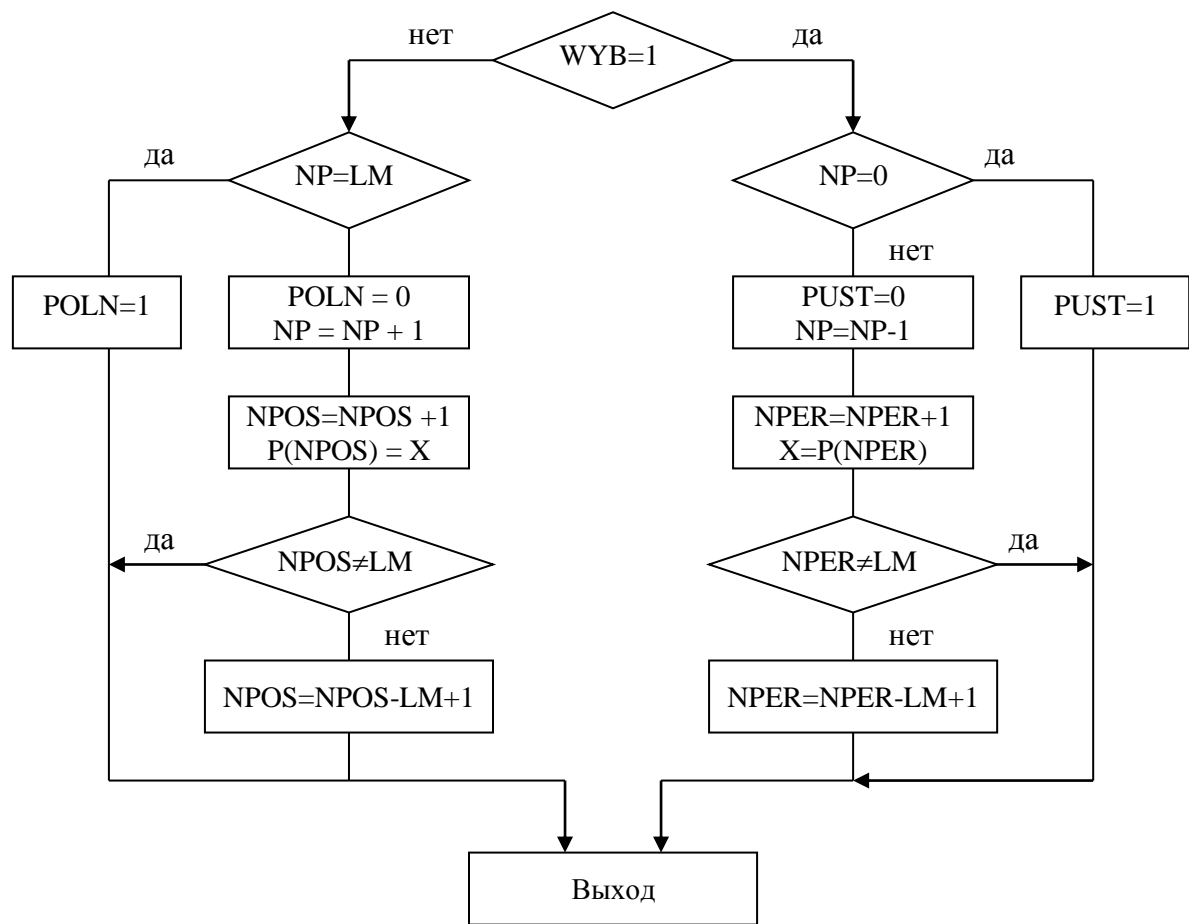
Моделирование работы буферной памяти.

Блок буферной памяти должен производить запись и считывания числа, выдавать сигналы переполнения и отсутствия данных. В любой момент времени располагать сведениями о количестве требований в блоке. Сама запоминающая среда имитируется некоторым одномерным массивом, размер которого определяет размер БП. Каждый элемент этого массива может быть либо свободен, либо занят. В качестве эквивалента требования присваивается значение времени появления этого явления.

| | |
|--|--|
| <ol style="list-style-type: none"> 1. Анализ признака режима. (запись) 2. Анализ на переполнение есть → блок статистики нет → запись информации по текущему адресу, изменение текущего адреса на 1 3. выход | <p>Чтение</p> <ol style="list-style-type: none"> 1. Анализ наличия сообщения в БП нет → запись в БС есть → чтение по текущему адресу, уменьшение текущего адреса (на 1) 2. выход |
|--|--|

Алгоритм:

Есть генератор, очередь, обслуживающий аппарат. Определяет оптимальную длину очереди. Оптимум – минимальная, при которой не теряется сообщение. Обслуживающий – по равномерному закону. Генерация – по своему закону. Параметры настраиваются.



Программа сбора статистики.

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

При моделировании работы простейшей СМО обычно 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 ОА, и моментом времени, когда оно пришло в систему от источника информации.

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

Коэффициент загрузки обслуживающего аппарата – отношение времени непосредственной работы ОА, к общему времени моделирования. Вероятность потери сообщения определяется как количество потерянных сообщений к общему числу.

Разработка управляющей программы.

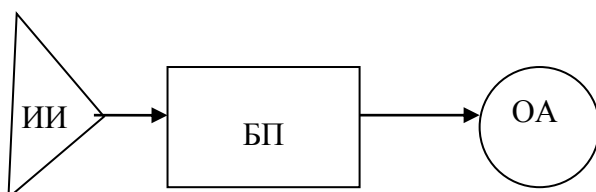
Если программа работы источника обслуживающего аппарата или памяти - имитирует работу отдельных устройств, то *управляющая программа* имитирует алгоритм взаимодействия элементов системы. Управляющая программа реализуется в основном по двум принципам:

1. принцип Δt

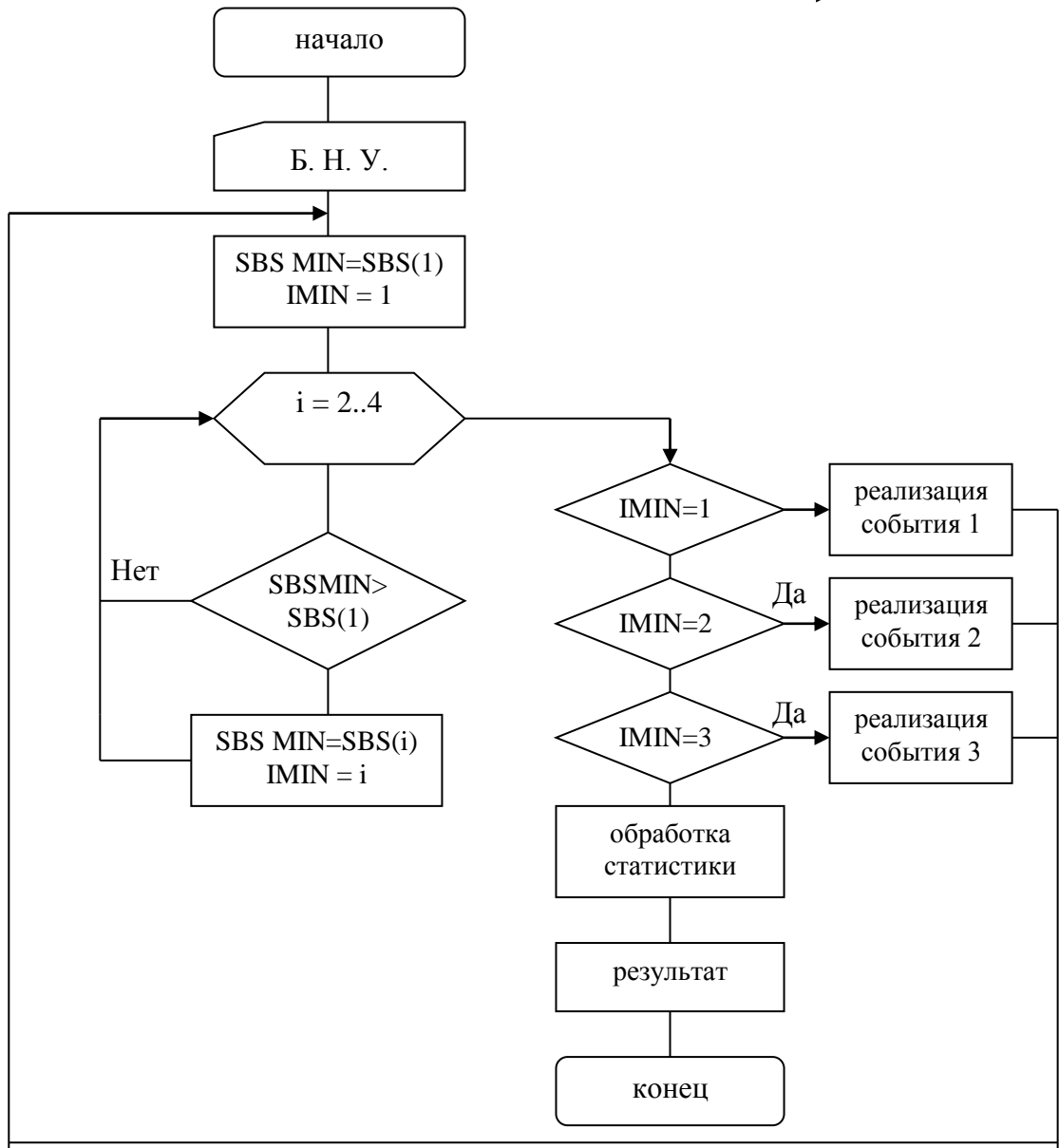
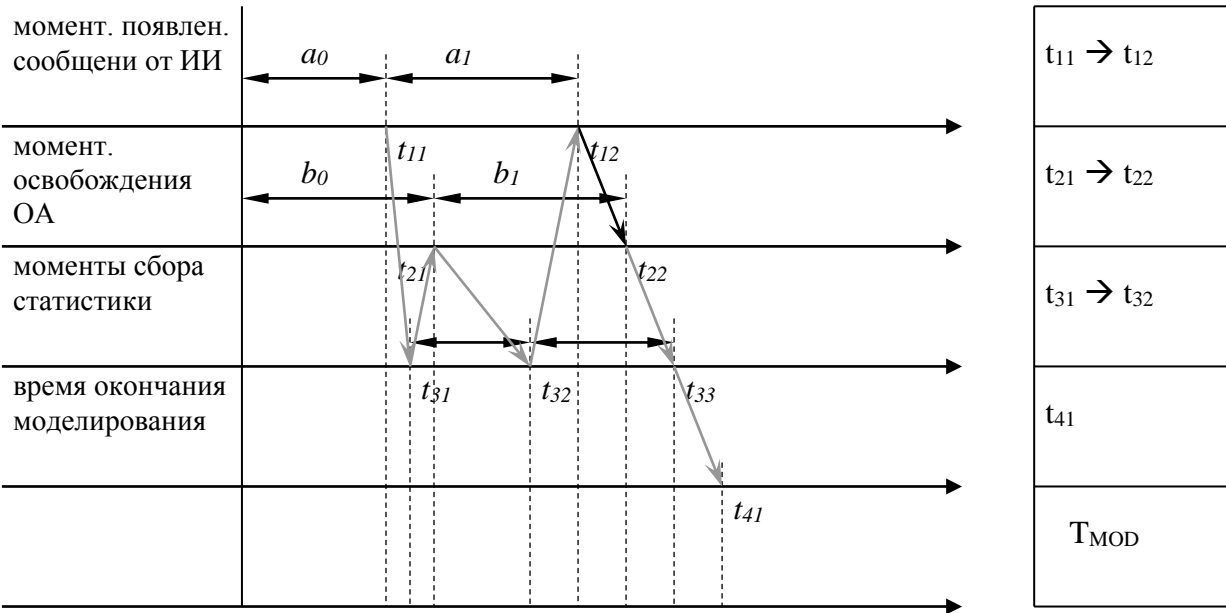
Заключается в последовательном анализе состояний всех блоков в момент $t+dt$ по заданному состоянию блоков в момент t . При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием, с учетом действующих случайных факторов, задаваемых распределением вероятности. В результате такого анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью на данный момент времени. Основной недостаток – значительные затраты машинного времени на реализацию моделирования. При недостаточно малом dt появляется опасность пропуска отдельных событий, что может привести к неверным результатам моделирования.

2. событийный принцип

Характерной свойство систем обработки информации заключается в том, что состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступлений сообщений в систему или окончанием решения задач, или возникновения аварийных ситуаций. Поэтому, моделирование и продвижение текущего времени производят, используя событийный принцип. При его использовании, состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. В момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов *ближайшего изменения состояния* каждого блока системы.



SBS



t_{11}, t_{12} – моменты появления сообщений на выходе генератора (источника информации)
 b_1 – интервал времени обслуживания первого сообщения
 t_{3n} – момент сбора статистики
 t_{41} – момент окончания моделирования
SBS – список будущих событий.

Методика реализации событийной модели.

1. Для всех активных блоков (блоков, порождающих события) заводится свой элемент в одномерном массиве – списке будущих событий.
2. В качестве подготовительной операции список в будущих событий заносится время ближайшего события от любого активного блока. Активизируя программ-имитатор, ИИ вырабатывает псевдослучайную величину a_0 , определяющую момент появления первого сообщения t_{11} . Эту величину заносят в список будущих событий. Активизируя программный имитатор ОА, вырабатывают псевдослучайную величину b_0 , определяющую момент времени t_{21} – в список будущих событий. Момент времени t_{31} (1ый сбор статистики) определяется равным стандартному шагу сбора $t_{\text{СТАТ}}$, и заносится в SBS
В SBS заносится t_{41} – время окончания моделирования.
Подготовительная часть на этом закончена и начинается протяжка модельного времени.
3. В SBS определяется минимально числовое значение и его номер.
4. Реализуется событие, порождаемое блоком с соответствующим номером, т.е. модельное время = t_{11} . Далее реализуется событие с номером 1, связанное с появлением нового сообщения в ИИ. Реализация этого события заключается в том, что само сообщение записывается в память, а с помощью имитатора ИИ, вырабатывается момент появления следующего события t_{12} . Это время помещается в соответствующую ячейку SBS место t_{11} .
Затем вновь организуется поиск минимального элемента в SBS. Для данного примера реализуется событие 3, после чего выражение момента времени t_{32} – новое время сбора статистики. Так до тех пор, пока минимально время не станет равным t_{41} .

Два описанных принципа являются универсальными алгоритмами протяжки модельного времени. Для некоторых предметных областей один принцип может работать быстро и без потерь событий, а другой будет работать при этом очень медленно. Выбор метода необходимо производить исходя из распределения событий во времени. В реальных системах распределение событий как правило неоднородно – события группируются по времени. Образованием групп связано с наступлением какого-либо значимого события, которое инициирует определенную последовательность действий соответствующими событиями, имеющими высокую плотность на определенном интервале времени. Такой интервал – *пиковый*. А распределение событий – *квазисинхронными*. Пример – цифровая сеть, в которой синхронизирующие сигналы переключают большое количество триггеров.

Дельфт.

Данный алгоритм был специально разработан для сложных дискретных систем, в которых присутствует квазисинхронное распределение событий. Особенность данного метода – автоматическая адаптация к распределению событий. Метод реализуется таким образом, что на пиковых интервалах он приближается к методу ДельтаТ, а вне них к событийному методу, с большим шагом. Алгоритм основан на использовании иерархической структуры циркулярных списков.

Рис.1.

Список уровня содержит N_1 элементов и описывает планируемые события в пиковых интервалах. Число N_1 представляет собой разбиение пикового интервала на более мелкие участки, с каждым из которых связан список событий, произошедших за этот интервал времени. Списки второго уровня и выше – являются масштабирующими списками, количество элементов которых равно константному значению N_2 , которое характеризует коэффициент масштабирования временных интервалов. Собственно, алгоритм протяжки модельного времени заключается в последовательном поиске не пустых элементов в самом верхнем циркулярном списке с большим шагом и дальнейшим поиском на более нижние уровни.

Языки имитационного моделирования.

Могут использоваться следующие языки:

1. универсальные алгоритмические языки высокого уровня.
2. функциональные языки. процессно-ориентированные языки
3. проблемно-ориентированные языки и системы моделирования

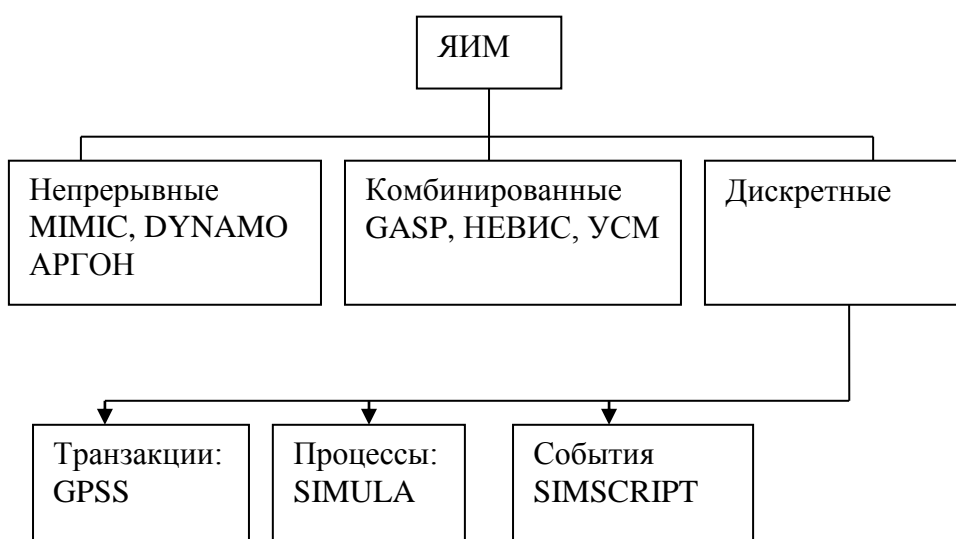
Основные методы построения языка РДО.

Качество языков характеризуется:

1. удобство описания процесса функционирования
2. удобство ввода исходных данных, варьирование структуры, алгоритмов работы и параметров модели.
3. эффективность анализа и вывод результатов моделирования
4. простотой отладки
5. доступностью восприятия и использования языка

Все современные языки моделирования определяют поведение систем во времени, с помощью событийного алгоритма или его модификации.

Классификация языков моделирования по принципу формирования системного времени.



Непрерывное представление систем сводится к дифференциальным уравнениям. Если переменные модели принимают дискретные модели, то уравнение – разностное.

GASP

События 2х типов:

1. события, зависящие от состояния
2. события, зависящие от времени

Формальное описание динамики моделируемого объекта.

Будем считать, что любая работа в системе совершается путем выполнения активности. Активность является наименьшей единицей работы. Ее рассматривают как единый дискретный шаг. Она имеет свое время выполнения. Активность является единым динамическим объектом, указывающим на совершение единицы работы. Процесс – это логически связанный набор активностей. Активности проявляются в результате свершения событий. Событие – это мгновенное изменение состояния некоторого объекта системы. Рассмотренные объекты - активности, процессы и события, являются конструктивными элементами для динамического описания поведения системы. На их основе строятся языки моделирования таких систем. В то время, когда динамическое поведение системы формируется в результате большого числа взаимодействующих процессов, сами эти процессы образуют относительно небольшое число классов, чтобы описать поведение системы, достаточно описать поведение каждого класса процессов и задать значение атрибутов для конкретных классов.

Построение модели – 2 взаимные задачи:

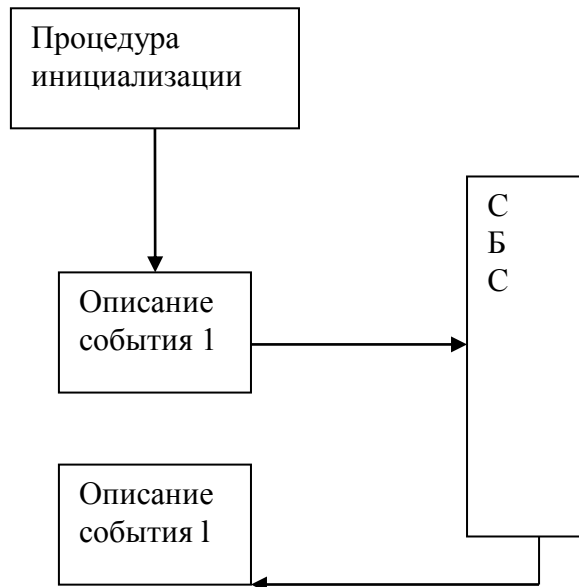
1. необходимо задать правило, определяющее виды процессов, происходящие в системе.
2. указать значение атрибутов таких процессов или задать правила генерации этих процессов. При этом, как правило, системы описываются на определенном множестве детализации, в терминах множества описания процессов, каждый из которых включает множество правил и условий возбуждения активностей. Такое описание системы может быть детализировано на более низкий иерархический уровень представления, с помощью декомпозиции процессов.

Для отображения временного поведения системы, язык моделирования дискретных систем должен обладать средствами отображения времени. В реальной системе совместно выполняется несколько активностей, принадлежащих как связанным и несвязанным процессам. Имитация их действия должна быть строго последовательной. Модель системы можно рассматривать как модель описаний активностей, событий или процессов.

Языки ориентированные на события.

Моделирующая программа организована в виде секций, они включают в себя события. Процедура событий состоит из набора операций, который в общем случае выполняется после завершения какой-либо активности. Выполнение процедуры синхронизируется во времени списком будущих событий.

Рис 2.



Языки ориентированные на процессы

Моделирующая программа в виде набора описаний процессов, каждый из которых описывает один класс процессов. Описание процесса функционирования устанавливает описание атрибутов всех процессов. Синхронизация операций во времени с помощью СБС, который содержит точку возбуждения конкретного процесса.

Описание объектов моделирования.

Инициализация

Описание процесса a1

Описание процесса a2

...

Описание процесса an

Все это программы моделирования

Результаты экспертных оценок, сравнение различных языков при моделировании широкого класса систем:

1. Возможности языка
 - SIMULA
 - SIMSCRIPT
 - GPSS
 - C
 - PASCAL
2. Простота применения
 - GPSS
 - SIMSCRIPT
 - SIMULA
 - C
 - PASCAL

3. Предпочтение пользователей
SIMSCRIPT
GPSS
SIMULA
PASCAL
C

Сравнение универсальных и специализированных языков программирования при моделировании.

| | Преимущества | Недостатки |
|--------------------|---|--|
| Универсальные | <ol style="list-style-type: none"> 1. Минимум ограничений на выходной формат 2. Широкое распространение | <ol style="list-style-type: none"> 1. значительное время затрачиваемое на программирование 2. значительное время на отладку |
| Специализированные | <ol style="list-style-type: none"> 1. меньше затрат времени на программирование 2. более эффективные методы выявления ошибок 3. краткость, точность понятий, характеризующих имитируемые процессы. 4. возможность заранее строить стандартные блоки, которые могут использоваться в любой имитационной модели 5. автоматическое формирование определенных типов данных, необходимых именно в процессе ИМ. 6. удобство накопления и представления выходной информации. 7. эффективное использование ресурсов. | <ol style="list-style-type: none"> 1. необходимость точно придерживаться определенных ограничений на форматы данных 2. меньшая гибкость модели |

РДО – ресурсы данные операции

Причина создания – требования универсальности ИМ относительно классов моделирования систем и процессов, легкости модификации модели, моделирование сложных систем управления.

Язык РДО – реализация так называемого интеллектуального подхода к ИМ. Это сравнительно новый подход, отойти от жесткого алгоритмического подхода в процессе принятия решения, и сделать процесс принятия решения максимально гибким по способам .. сложной дискретной системы.

В основе – продукционная система, состоит из 3 элементов – класса и отношение, правил управляющей структуры. Классы и отношения трактуются как БД, содержащая декларативные знания. Процедура – набор модифицированных продукционных правил типа «Если, ... то...». Управляющая структура – интерпретатор правил, управляющий выборкой правил. Условие – проверка правила, действие – изменение.

Достоинство системы – простота создания и понимания отдельных правил, простота пополнения и модификации.

Недостатки – неясность взаимных отношений правил, сложность оценки целостного образа знаний, крайне низкая эффективность обработки.

Для ИМ основным недостатком системы продукции является отсутствие времени, т.е. такие правила применимы только при моделировании статических объектов.

В РДО используют модифицированное правило:

Если <условие> то <событие 1> ждать (временной интервал) то <событие 2>

В РДО сложная дискретная система представляется в виде множества взаимодействующих между собой ресурсов. Ресурс – это элемент сложной системы, внутренней структурой которой можно пренебречь, в то время как наличие и свойство его как целого важны и существенны для описания. Каждый ресурс модели должен быть описан множеством параметров, которые могут быть следующих типов:

1. описательные, представляющие факты, внутренне присущие каждому ресурсу.
2. указывающие, используемые для имени (id)
3. вспомогательные, используемые для связи различных ресурсов, накопления статистики, графического вывода при имитации.

Лаба 5

Метро Бауманская

1. концептуальная модель – графическая схема термина СМО.
2. выделены параметры: входная информация, ограничения на параметры входного потока, внутренняя структура (количество,...), выходная информация.
3. формализация процесса в виде математической модели – выбирается либо типовая математическая модель, либо своя, с обоснованием и доказательством – почему можно промоделировать станцию именно так
4. результаты – в графическом виде.
5. текстовый отчет должен содержать информацию с рекомендациями по повышению производительности, пропускной способности

General Purpose System Simulation (GPSS)

GPSS – общецелевая система моделирования, как и любой язык, он содержит словарь и грамматику, с помощью которых легко могут быть разработаны точные модели систем определенного типа. Существуют версии 1, 2, V, PC. Любой пакет в нем построен в предположении что моделью сложной дискретной системы является описание ее элементов и логических правил, взаимодействия в процессе функционирования. Для определения класса моделирования системы можно выделить конечный набор абстрактных элементов, называемых *объектами*. Набор логических правил также ограничен и может быть описан небольшим числом стандартных операций.

Объекты языка подразделяются на 7 категория и 14 типов.

| Категории | Типы |
|----------------|---|
| Динамическая | Транзакт |
| Операционная | Блоки |
| Аппаратная | Устройства, памяти, ключи |
| Вычислительная | Переменные, арифметические, логические, функции |
| Статистическая | Очереди, таблицы |
| Запоминающие | Ячейки, матрицы ячеек |
| Группирующие | Списки, группы |

Для облегчения процесса построения модели разработан так называемый язык блок-диаграмм, который позволяет упростить переход от алгоритма, определяющего процесс функционирования, к модели.

Основой пакета является программа, описывающая функционирование выделенного ограниченного набора объектов, и специальная диспетчеризирующая программа, которая называется симулятор и выполняет следующие функции:

1. обеспечение заданных маршрутов продвижения динамических объектов
2. планирования событий, происходящих в модели путем регистрации времени каждого события и выполнение их в нарастающей временной последовательности
3. регистрация статистической информации по функционированию модели
4. продвижение модельного времени

Динамическими объектами являются *транзакты* (сообщения), которые представляют собой единицы исследуемых потоков и производят ряд определенных действий,

продвигающихся по фиксированной траектории, представляющих собой совокупность объектов других категорий.

Операционные – блоки, задающие логику работы системы и определяющие пути движения транзактов между объектами аппаратной категории.

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

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

Функции – используя их, пользователь может задавать непрерывную или дискретную последовательность между аргументом функции и ее значением. Функции задаются табличным способом с помощью оператора описания функции.

Очереди – в любой системе движение потоков транзактов может быть задержано из-за недоступности ресурсов, например, необходимое устройство уже занято. Как правило, это многоканальные устройства. В этом случае задержанные транзакты становятся в очередь. Учет этих очередей составляет одну из основных функций интерпретатора. Пользователь может специально определить точки модели, в которых необходимо собирать статистику об очередях, т.е. установить регистраторы очереди. В этом случае интерпретатор автоматически собирает статистику об очередях – длина, среднее время нахождения в очереди, и т.д. Вся эта информация является стандартным числовым атрибутом – СЧА. И она доступна пользователю в процессе моделирования. Интерпретатором также автоматически поддерживается дисциплина обслуживания FIFO. Пользователь может получить стандартную статистическую информацию только о таких очередях. Если же есть необходимость организовать очередь из транзактов другой дисциплины обслуживания, то для этого используются списки пользователя. Эти списки также помогают осуществить синхронизацию движения разных транзактов по модели.

Объект *таблица* – для сбора статистики о случайных величинах. Таблица состоит из частотных классов, в которые заносится число попаданий конкретной величины (СЧА). Вычисляется математическое ожидание и среднеквадратичное отклонение.

В процессе моделирования системы одни объекты взаимодействуют с другими, в результате чего происходят изменения атрибутов и преобразование их значений. Такие преобразования – *события*.

Транзакты моделируют прохождение по системе соответствующих единиц исследуемого потока. Такое движение может быть разбито на ряд элементарных событий, происходящих в определенные моменты времени.

Основной задачей имитатора является распределение этих событий, расположение их в правильной временной последовательности и выполнение соответствующих действий при наступлении каждого события. Все отрезки времени описываются целыми числами.

При составлении модели необходимо провести временное масштабирование, для всех временных параметров.

Особенности модели на языке GPSS.

Каждому объекту соответствуют атрибуты, описывающие объект в данный момент времени. Значения – арифметические или логические. Большая их часть недоступна пользователю. Атрибуты которые можно адресовать – стандартные логические, СЧА.

С блоками непосредственно связано - операционные блоки, изменяющие процесс моделирования, блоки вывода и печать промежуточных результатов, команды управляющие процессом моделирования, команды редактирования. Всем блокам – порядковые номера. Транзакты представляют собой описание динамических процессов реальной системы. Они могут определять реальные физические объекты и нефизические.

Канальные программы

Транзакты можно генерировать и уничтожать. Основным атрибутом транзакта является его параметры. Число которых для каждого колеблется от 0 до 1020. P_i . I – номер, x – тип. Слово, полуслово, байт, плавающая точка L . Когда два транзакта соперничают при занятии какого-либо места, первым обрабатывается тот, у которого приоритет выше. Если они одинаковые, то сначала тот, у которого время ожидания обработки больше. В одном задании может выполняться как один так и несколько прогонов модели. При этом текущем значением абсолютного времени модели будет называться суммарное время по всем реализованным прогонам – $A1$, а текущем значением относительного времени – $C1$, системное время в пределах одного прогона.

Классификация блоков GPSS.

Блоки, используются для описания функций и управляют движением транзактов. У каждого блока имеется 2 стандартных числовых атрибута.
 W_n – счетчик входа в блок. Как правило номер транзакта, входящего в данный блок.
 N_n – Общий счетчик транзактов, поступивших в блок с начального момента моделирования.

1. Блоки, осуществляющие модификацию атрибутов.
Временная задержка – ADVANCE
генерация и уничтожение – GENERATE TERMINATE SPLIT ASSEMBLE
синхронизация движения нескольких транзактов – MATCH GATHER
изменение параметров транзактов – ASSIGN INDEX MARK
изменение приоритетов - PRIORITY
2. Блоки, изменяющие последовательность передвижения транзактов (блоки передачи управления).
TRANSFER, LOOP, TEST, GATE
3. Блоки, связанные с группирующей категорией
JOIN, REMOVE, EXAMINE, SCAN, ALTER
4. Блоки, сохраняющие значения
SAVEVALUE, MSAVEVALUE
5. Блоки, организующие использование объектов аппаратного устройства.
устройство – SEIZE - RELEASE
PREEMPT - RETURN
FAVAIL – FUNAVAIL
памяти – ENTER – LEAVE
SAVAIL – SUNAVAIL
ключи – LOGIC

6. Блоки, обеспечивающие получение статистических результатов.
QUEUE, DEPART
TABULATE, TABLE
7. Специальные блоки
8. Блоки для организации цепей
LINK, UNLINK
9. Вспомогательные блоки
LOAD, SAVE

Каждый блок определяется с помощью отдельной команды. В общем случае сначала нумерация, потом поле метки, поле операции, поле операндов, если необходимо - комментарии

Блоки, связанные с динамической категорией.

Самостоятельно:

управляющая команда START со всеми ситуациями.

блок MATRIX – описание матрицы

Самостоятельно функцию типа

Относится к транзактам, которые в процессе моделирования создаются, размножаются, собираются и уничтожаются. Каждому транзакту соответствует набор параметров. Параметры транзактов: это свойства транзакта, определяемые пользователем. Множество параметров – набор стандартных числовых атрибутов, которые принадлежат именно этому данному транзакту. С точки зрения программной реализации, это локальные переменные, которые доступны именно этому транзакту. В процессе перемещения транзакта по модели, его параметры могут задаваться и модифицироваться.

Особенности параметров транзактов:

P<номер транзакта>

P22

P\$<имя>

P\$Color

Номера – с помощью целых чисел или символьных имен.

При входе транзактов в модель, начальные значения всех его параметров устанавливаются в 0. Значение параметров транзакта и их изменение определяется пользователем. Значения могут быть и отрицательные числа. Транзакт может обращаться только к своим параметрам. Если необходимо получить доступ к параметрам других транзактов, то понятие ячейки или через понятие групп транзактов.

Параметры можно использовать в качестве операндов блоков и в качестве аргументов функций. Параметры позволяют организовать косвенную адресацию.

С динамической категорией связаны следующие группы блоков:

1. *группа блоков задержки транзактов по заданному времени.* Из 4 типов событий, которые могут произойти с транзактом, простейшим является задержка транзакта в течении определенного времени. Задать это можно в блоке ADVANCE A,B. Он задает среднее время выполнения операции в моделируемой системе, а также разброс времени относительно среднего. Время может задаваться любым положительным числом, в том числе и 0. Если время равно 0, то транзакт в блоке ADVANCE не задерживается и передается в следующий блок. Для задания времени пребывания его в блоке ADVANCE, в поле A указывается среднее время, а модификатор указывается в поле B. Если время задержки постоянно, то поле B может быть пустым. Иначе, если оно 0 – то поле A пустым. Модификатор может быть 2х типов:
 1. *модификатор-интервал.* используется, когда время задержки транзакта распределено равномерно в некотором заданном диапазоне.
ADVANCE 10,5
 2. *модификатор-функция.* Он используется, если время задержки транзакта распределено более сложно. При обращении к функции определяется некоторое

число – ее значение. И время задержки транзакта в блоке определяется умножением этого числа на значение среднего. Если результат – не целое число, то берется целое.

ADVANCE 500,FN\$XPDIS

2. *Группа блоков создания и уничтожения. GENERATE, TERMINATE, SPLIT, ASSEMBLE.*

GENERATE – создание транзакта, входящего в систему.

GENERATE A,B,C,D,E,F,G,H,I

A – среднее время между поступлениями отдельных транзактов, как и в поле ADVANCE, оно может быть модифицировано – быть параметром-интервалом/функцией. Может быть 0. Если при вычислении времени появления первого транзакта оно получилось равным 0, то программа-симулятор полагает его равным 1. Среднее время принимается равным 1, если поле B пусто, а в поле A – модификатор-функции. Задаваемый модификатор-интервал не должен превосходить среднего, записанного в поле A, чтобы не получились отрицательные интервалы между появлениями транзактов. Интервал между транзактами, т.е. время появления следующего транзакта, вычисляется после того, как генерируемый транзакт покидает блок. Поэтому, если после блока GENERATE стоит блок, который по какой-либо причине может задержать сгенерированный транзакт, то время генерирования следующего транзакта будет вычислено после снятия блокирующего устройства. Следовательно, средний интервал между транзактами будет *больше* чем среднее значения поля A, что приводит к ошибке. C – записывается начальная задержка. Заданное в этом поле число (без модификации) определяет интервал времени до создания данного блока первого транзакта.

D – задает число транзактов, которое должно быть сгенерировано блоком GENERATE. Если оно пусто, то он генерирует неограниченное число транзактов.

E – задается приоритет

F – I: резервируют для транзакта необходимое число типов параметров. Слово, полслова, байт, плавающая точка.

GENERATE 10,3,100,16,5,5PB,20PH,3PL,4PF

GENERATE 10,2,1000,10,4

GEN 100,FN\$EXPON

TERMINATE A – удаляет транзакты. На блок-диаграммах для обозначения окончания. Поле указывает, изменяет ли этот блок содержимое счетчика TG1 в момент поступления транзакции. Если изменяет, то на что. Каждый раз, когда транзакт входит в этот блок, то значение счетчика меняется на это число (отнимается). Во всей модели только один блок TERMINATE, то по завершению моделирования через этот блок пройдет 500. Если поле не определено, то оно считается равным 0. И транзакты, проходящие через этот блок не уменьшают содержимое счетчика. Следовательно, в модели должен быть хотя бы один такой блок, у которого поле A не меньше 1.

3. *Группа блоков изменения параметров. Интерпретация смысла параметров транзактов – произвольна.*

ASSIGN A,B,C – основное средство для задания значений параметров транзактов.

A – какой параметр поступившего транзакта должен быть изменен. Следующий непосредственно за ним символ указывает, что надо сделать, с записанным в поле B целым числом. Варианты: прибавить, вычесть, заменить текущее значение этим числом.

Если в поле C указано какое-либо значение, оно интерпретируется как номер функции, производится определение значения функции, а результат используется

для модификации целого числа, указанного в поле В. Произведение помещается в параметр, указанный в поле А.

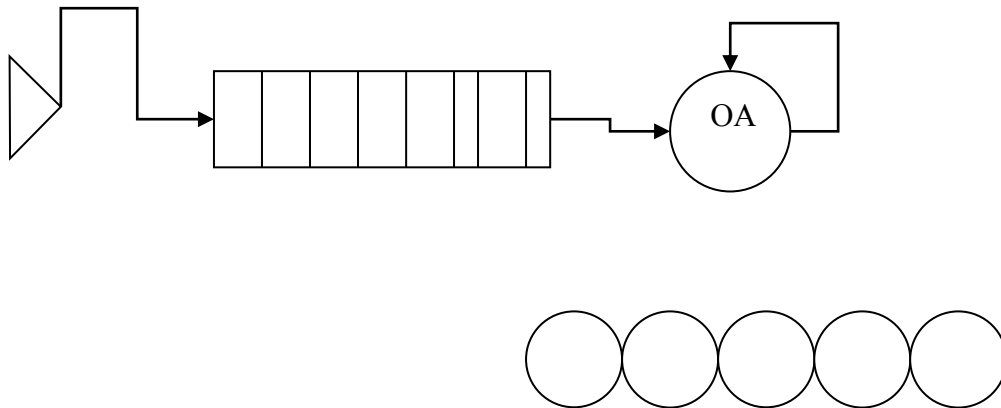
Организация циклов.

LOOP A [,B] – управляет количеством повторных проходов транзактами определенной последовательности блоков модели.

А – параметр транзакта, используемый для организации цикла – переменная цикла. Он может быть числом, именем, стандартным числовым атрибутом.

В – метка (имя) начального блока цикла. Когда транзакт входит в блок цикла, параметр поля А уменьшается на 1, а затем его значение проверяется на равенство 0. Если оно не равно 0, то транзакт переходит в блок, указанный в операнде В. Иначе он переходит в следующий блок.

Задача: построить имитационную программу модели процесса прохождения 70 деталей, поступающих с интервалом 12 ± 2 , обрабатываемых одним рабочим по 5 последовательно идущим друг за другом операциям. 2 ± 1 времена распределения. Загрузка рабочего?



```
10 GENERATE 12,2
20 ASSIGN 2,5
30 SEIZE WORKER
40 WAIT ADVANCE 2,1
50 LOOP 2, WAIT
60 RELEASE WORKER
70 TERMINATE 1
80 START 70
```

Лаба: на примере – отладка в GPSS.

Группа блоков создания копий транзактов.

SPLIT A,B,C,D,E,F,G

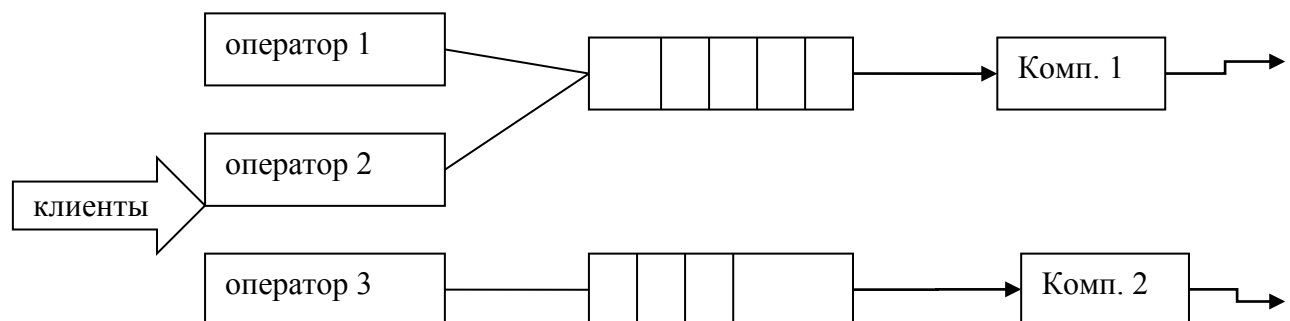
Также как блок GENERATE, предназначен для создания транзактов, но в отличие от него не создает дополнительных транзактов, а лишь генерирует заданное число копий входящего в него транзакта. Получаемые копии идентичны. Число копий задается в поле А. После прохождения блока SPLIT, исходный транзакт направляется в следующий блок, а все копии пересылаются по адресу, указанному в В. Если в поле А задано некоторое число, то $i+1$. Исходное сообщение и копия являются равноправными и могут проходить

через любое количество блоков. Все полученные копирование транзакты, а также копии копий, принадлежат к одному ансамблю. К этому ансамблю можно применять специальные блоки, осуществляющие обработку ансамблей транзакта – MATCH, GATHER, ASSEMBLE. Получаемый ансамбль может быть пронумерован, с поле С записывается номер ансамбля, в котором и будет произведена нумерация. Если в исходном транзакте значение этого параметры было равно К, то после – К+1. Первая копия – К+2. Так как копии параметров одинаковы, необходимо использование индексов для указания типа параметров, который берется при объединении серии. Полученные копии могут иметь число и типы параметров, отличные от исходных. Эти поля можно задавать в любом порядке – в каждом поле указывается индекс параметра, для определения его типа.

Лаба:

Составить имитационную модель:

в информационный центр приходят клиенты, через интервал времени 10 ± 2 мин. Если все 3 имеющихся оператора заняты, то клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание запроса пользователя за 20 ± 5 , 40 ± 10 , 40 ± 10 . Клиенты стараются занять свободного оператора с максимальной производительностью. Полученные запросы собираются в накопитель, оттуда выбираются на компьютеры. На первый – от первого и второго. На второй – от третьего. Время обработки запроса на первом и втором компьютере = 15/30 минут соответственно. Промоделировать процесс обработки 300 запросов. Определить вероятность отказа.



1. режим нормального обслуживания – клиент выбирает одного из свободных операторов, отдавая предпочтение тому, у которого меньше номер
2. режим отказа в обслуживании клиента – когда все операторы заняты

Переменные уравнения имитационной модели

Эндогенные переменные.

TR_j – время решения задачи на j-ом компьютере.

Экзогенные переменные

N_0 – число обслуженных клиентов.

Группа блоков синхронизации движения транзакции.

Блок ASSEMBLE A – используется для определения определенного числа транзактов, являющихся членами одного ансамбля. Транзакты, принадлежащие одному ансамблю, будут задерживаться в блоке ASSEMBLE до тех пор, пока не поступят заданное число транзактов этого ансамбля. В результате на выходе появляется один (первый) транзакт ансамбля, а остальные транзакты уничтожаются. В одном блоке могут накапливаться транзакты разных ансамблей. Транзакты одного ансамбля могут накапливаться. Если число разных ансамблей задается с косвенной адресацией, то для его установления используется параметр первого пришедшего транзакта.

Рассмотрим пример.

Tabulate – Запись времени между поступлениями в ящик.

Построить программу модели процесса прохождения 100 деталей, равномерный закон 8 ± 2 . Обработка параллельно 2мя рабочими, каждый из которых выполняет свою операцию независимо со временем 5 ± 3 , также распределенным равномерно. Требуется определить коэффициент занятости.

```
10    GENERATE 8, 2    // приход деталей
20    SPLIT 1, THIS    // начало обработки детали
30    SEIZE 1          // первый рабочий
40    ADVANCE 5, 3
50    RELEASE 1
60    TRANSFER , THAT

70
THIS SEIZE 2          // второй рабочий
80    ADVANCE 5, 3
90    RELEASE 2
100
THAT ASSEMBLE 2      // окончание обработки
110   TERMINATE 1
      START 100
```

самостоятельно – все окна графики

Действие блока GATHER A аналогично ASSEMBLE, отличие в том, что после накопления в блоке числа транзактов указанного в поле A, они все передаются в следующий блок. Этот блок позволяет синхронизировать в режиме транзактов одного ансамбля при их движении по одному пути.

GATHER 3 – сначала все 3 придут, и потом они же последуют дальше.

задача: на производственный участок сборки подшипников поступают обоймы и шарики с интервалом времени 25 ± 4 . На контроль обоймы затрачивается 4 ± 1 . Контроль шариков производится последовательно со временем 2 ± 1 на каждый шарик. Операция сборки требует одновременного поступления обоймы и всех шариков и производится со временем 4 ± 2 .

```
10    GENERATE 25, 4    // поступление шариков и обойм
20    SPLIT 8, THAT    // разделение обоймы и шариков
30    SEIZE 1
40    ADVANCE 4, 1
50    RELEASE 1
60    TRANSFER , FINAL
70
THAT SEIZE 2
80    ADVANCE 2, 1
90    RELEASE 2
100   GATHER 8
110   FINAL ASSEMBLE 9
120   SEIZE 3
130   ASSEMBLE 4, 2
140   RELEASE 3
```

150 TERMINATE 1
 START 80

Блок MATCH предназначен для синхронизации, продвижения транзактов ансамбля, движущихся разными путями. Для синхронизации необходимо 2 блока MATCH. Они называются сопряженными. В поле A каждого блока указывается метка сопряженного ему блока. При подходе транзакта к блоку, проверяется наличие в сопряженном ему блоке транзакта из того же ансамбля. Если в обоих блоках имеются транзакты одного ансамбля, то они одновременно пропускаются через эти блоки. Если в сопряженном блоке нет ни одного транзакта данного ансамбля, то поступивший транзакт будет ожидать поступления транзакта в сопряженный ему блок. После чего оба транзакты будут пропущены в следующие за блоками MATCH.

Одна и та же пара блоков MATCH может одновременно синхронизировать любое число пар транзактов из разных ансамблей. Транзакты из одного ансамбля могут быть также синхронизированы в любом числе пар пар блоков MATCH. Блок MATCH может быть сопряжен сам себе. При этом его действие будет аналогично блоку GATHER 2.

Пример:

AAA1 MATCH AAA1
...
BBB1 MATCH AAA1

Рис. 1

CCC1 MATCH CCC1

Здесь транзакт будет ждать прихода члена этого же ансамбля в этот же блок MATCH.

Задача:

Необходимо смоделировать процесс прохождения 500 деталей. Они поступают со временем 300+-50. Обработку производят двое рабочих, которые выполняют по 2 операции. После 1-ой операции, выполняемой первым рабочим со временем 70+-20 мин, и вторым со временем 60+-20, производится операция сверки, время выполнения которой принимается равным 0. после сверки выполняется вторая операция, первым рабочим за время 20+-10, а вторым за время 30+-20 минут. Затем 3-ий рабочий производит сборку изделия из этих деталей за время 50+-20. Все процессы подчиняются равномерному закону.

```
10    GENERATE 300,50
20 MAN_A SEIZE 1
30    ADVANCE 70,20
40    HERE MATCH THERE
50    ADVANCE 20,10
60    RELEASE 1
70    TRANSFER ,MAN_C
80 MAN_B SEIZE 2
90    ADVANCE 60,30
100   THERE MATCH HERE
110   ADVANCE 30,20
120   RELEASE 2
130 MAN_C ASSEMBLE 2
140   SEIZE 3
150   ADVANCE 50,20
160   RELEASE 3
```

170 TERMINATE 1
 START 500

Блоки, описывающие аппаратную категорию

Устройство является аналогом обслуживающего уста

В любой момент времени уст-во может быть занято только одним транзаком. Состояние уст-ва меняют 6 блоков (SEIZE...RELEASE,

 FREEMPT

 ...

 RETURN

 FAVAIL

 ...

 FUNAVAIL

В результате входа транзакта в блок SEIZE уст-ва указанные в данном блоке, будут заняты. Оно остается занятым, пока тот же транзакт не пройдет соответствующий блок RELEASE. Если какой-либо транзакт занимает уст-во, описанное в поле А блока SEIZE, то никакой другой транзакт не сможет войти в этот блок и вообще не сможет захватить это уст-во, описанное в любом другом блоке. Один транзакт может занять любое число уст-в.

Блок FREEMPT для освобождения уст-ва, которое ранее было захвачено, проходящим через блок SEIZE. При выполнении этого блока задержка возникнуть не может. Уст-во освобождается в момент входа транзакта в блок RELEASE. Освобождение происходит только тем транзаком, в котором уст-во было занято. Если перед блоком SEIZE задерживаются несколько транзактов, они обслуживаются в соответствии с режимом FIFO.

Блок FREEMPT фиксирует использование уст-ва на более высоком уровне. А также приостанавливает обслуживание транзакта, захватившего уст-ва раньше, и дает возможность прерванному транзакту захватить уст-во после того, как закончится обслуживание прервавшегося транзакта. Если при реализации блока FREEMPT, одно прерывание уже произошло, то есть уст-во обслуживает прерывание, то блок не выполняется и соответствующий транзакт задерживается до тех пор, пока не освободится уст-во. Затем обслуживается новый прерывающий транзакт, а не прерванный. Исключением из описанных выше правил является когда блок FREEMPT работает в режиме приоритета, то есть в поле В стоит PR. При этом в действие данного блока предусмотрено

Для последующей обработки прерванных транзактов существует следующая возможность: в поле С может быть определен какой-либо блок, на который будет передан прерванный транзакт. Если прерванный транзакт находится в блоке ADVANCE, то вычисляется остаток времени от момента входа, до выхода. И полученное значение помещается в параметре, описанном в поле D. В этом случае прерванный транзакт пересылается к блоку, указанному в поле С. Если в поле Е стоит мнемоническое обозначение RE, то будут производиться обычные операции, присущие данному блоку, за исключением того, что прерванный транзакт не участвует больше в конфликте из-за захвата уст-ва.