Признаки событийности являются наиболее существенными для адекватного составления математических моделей. Признаки событийности и построенные на них методы моделирования существенно отличают исследуемый подход от методов моделирования, общепринятых, например, в теории систем и теории автоматического управления, основными инструментами которых являются интегро-дифференциальные уравнения, методы теории вероятностей, математической статистики и случайных процессов. Результатами моделирования являются восстановленное фазовое пространство моделируемых систем и характеристики звеньев управления и регулирования.

Другим подходом является имитационное моделирование [4], использующее методы теорий систем и сетей массового обслуживания (СМО и СеМО соответственно), в которых рассматриваются различные модели входных, выходных потоков и правил обслуживания, построенных на базе соответствующих законов распределения случайных величин и процессов. в. В данном случае методами моделирования являются генераторы некоторых типов случайных процессов (имитирующих моменты времени поступления заявок на обслуживание), а результатами моделирования (часто усредненными) – времена пребывания заявки в очереди, системе, времена обслуживания, вероятности пребывания в очереди, вероятности обслуживания (за некоторый период) и другие вероятностные и статистические характеристики.

Заметим, что и в общей теории систем, и в теориях СМО и СеМО неявно предполагается использование процессного времени. Это означает, что в первом случае принимается, что процессы в системах протекают по возможности мгновенно (хотя в некоторых случаях допустима их инерционность), а во втором случае принимается, что процессы подчинены некоторому вероятностному закону. В целом процессное время означает, что изменение состояний системы, а также ее модели можно отметить на некоторой временной шкале; если шкала является непрерывной, то естественным будет непрерывное моделирование состояний системы, иначе – дискретное, а также соответствующий им математический аппарат.

В основу дискретно-событийного моделирования положена концепция - состояния системы изменяются под воздействием некоторых событий, в общем случае безотносительно к их точной привязке к временной шкале. Существенными являются лишь факты возникновения этих событий и взаимодействие их между собой, то есть синхронизация (некоторое событие предшествует другому, некоторое событие вызывает возникновение другого либо других событий и так далее).

**Подходы и алгоритмы дискретно-событийного моделирования**

Проведем грань между дискретно-событийным моделированием и другими методами моделирования более четко.

Большинство систем моделируется по принципу «вход–состояние–выход». Принимая общеизвестные обозначения векторов: u(t) – вход, x(t) – состояние, y(t) – выход, динамика моделируемой системы описывается уравнениями с начальными условиями t > 0:

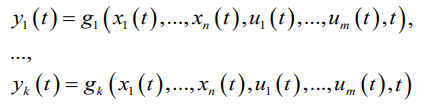
(1)

Уравнение (1) означает составление множества состояний моделируемой системы, а если принять, что множество таких состояний равно n, а множество входных сигналов равно m, то получается, что необходимо моделировать n уравнений состояний:

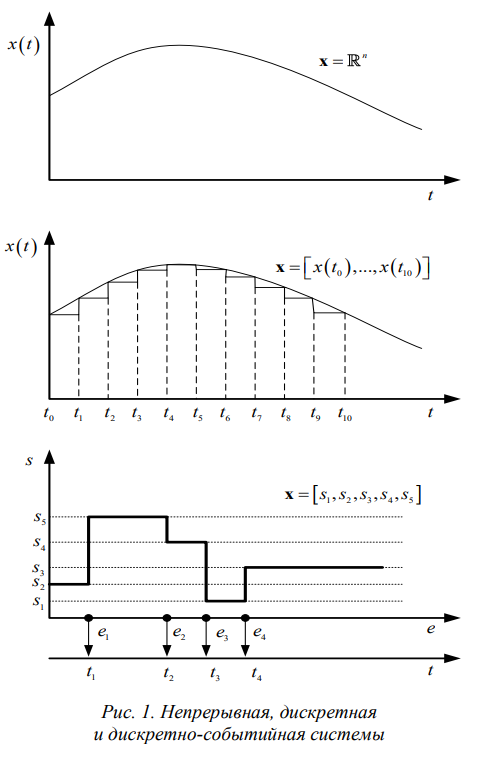




и k выходных уравнений системы:



Еще раз следует заметить, что изменение состояний динамической системы в таких случаях моделирования всегда привязано ко времени, какими бы его измерениями мы ни пользовались, непрерывными или дискретными. В дискретно-событийном моделировании важен факт фиксации события (или группы событий), а в какое время, либо интервал времени, либо через какой промежуток времени эти события фиксируются, уже не столь важно. Получается, что «событие первично, а время вторично».



В верхней части показано пространство состояний непрерывной системы, в средней – дискретной системы, а в нижней – дискретно-событийной системы. Ясно, что пространство состояний дискретно-событийной системы является дискретным и составляет события s1, …,s5, а переключение между этими состояниями происходит в соответствии с наступлением некоторых событий e1, …, e4. Естественно, при динамической смене состояний системы может происходить возврат к предыдущим состояниям, поэтому моделирование пространства состояний будет составлять при упорядочении по хронологии цепочек событий и совпадении момента времени и события (или группы событий) последовательность пар («время», «состояние»). В данном случае (рис. 1) это

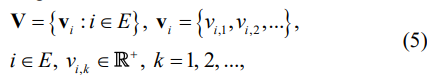
 (3)

В формальном виде **дискретно-событийная система** – это некая разновидность временнόго автомата [11], который представляется в следующем виде

**Определение 1. Модель дискретно-событийной системы** представляет собой кортеж:

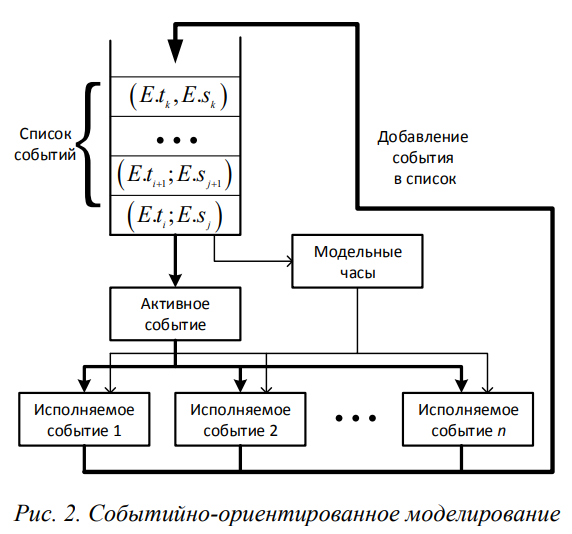
(4)

где X – конечное множество, пространство состояний системы; E – конечное множество событий; f – функция смены состояний, f: X × E → X; Г – конечное множество активных (и исполняемых в текущий момент) событий; x0 – начальное состояние. В связи с тем, что время как таковое не присутствует в модели (4), но при имитационном моделировании все же необходимо воспроизводить хронологию событий по мере упорядоченности их между собой, (4) дополняется модельными часами, связанными с множеством событий. Такие модельные часы представляют собой конечное множество:



где vi,k – время жизни (продолжительность) события.

Для последовательности событий {e1, e2, …, ek, ek+1, …} можно «включить модельные часы» (5), получить V = {vi : i = 1, …, m} и генерировать события ek+1 = h(xk, v1, …, vm). Динамика состояний дискретно-событийной системы при этом определяется уравнением xk+1 = f (xk, v1, …, vm). Таким образом, очень существенной задачей является формирование (генерация) списков событий, то есть пар вида (3), в зависимости от которой можно выделить событийно-ориентированное и процессно-ориентированное исполнение событий.



Алгоритм 1. Событийно-ориентированное моделирование.

1. Установить модельные часы в 0. Инициализировать начальный список событий, расположив их в хронологическом порядке следования. Элемент списка событий имеет структуру (E.ti; E.sj) и характеризуется временем и типом состояния (ti, sj).

2. Выбрать событие E из начала списка. Если список пуст, завершить моделирование.

3. Установить модельные часы в E.ti. Проверить длительность события и при превышении времени, отведенного на моделирование, завершить его.

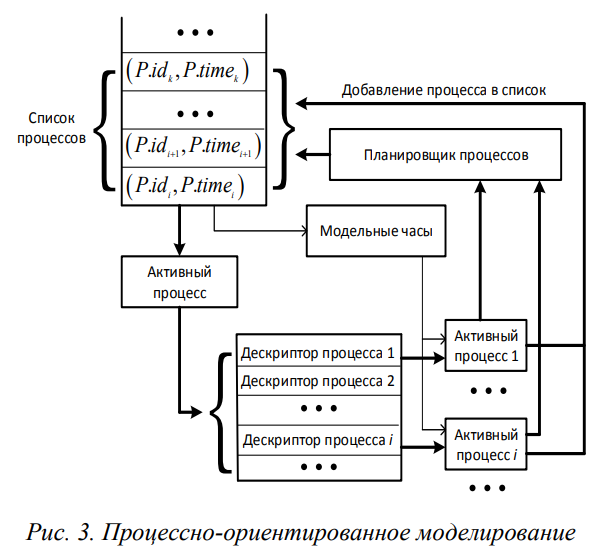
4. В соответствии с типом события и состояния E.sj исполнить подпрограмму-обработчик события.

5. Обновить список событий, системные переменные и структуры, поместить новое событие в список событий.

6. Продолжить моделирование, перейдя к п. 2. Элементы возможной программной реализации алгоритма 1 показаны в примере 1.

Обработчик устанавливает начальное время, инициализирует список событий и в цикле, выбирая следующее событие, переустанавливает модельные часы, проверяя, чтобы не было превышения максимально допустимого времени моделирования, вызывает подпрограмму exec\_event(status). При ее реализации имеет смысл запрограммировать хотя бы простейшее планирование списка событий в виде динамической очереди FIFO (First In First Out).

Во втором подходе, процессно-ориентированном исполнении событий, есть возможность исполнения группы событий при их логическом объединении в процессы,



Основным отличием данного вида от предыдущего является не только возможность объединять процессы в группы, но и планировать переключение между процессами, разделяя при этом совместные ресурсы моделируемой системы.

Алгоритм 2. Процессно-ориентированное моделирование.

1. Установить модельные часы в 0. Инициализировать начальный список событий, расположив их в хронологическом порядке следования. Элементы списков событий группируются в процессы, имеющие структуру (P.idi; P.timei), и характеризуются идентификатором и временем (id; time). Запустить основной цикл моделирования.

2. Создать активный процесс P из списка планируемых к исполнению событий. Если список пустой, завершить моделирование.

3. Проверить превышение максимального времени, отведенного на моделирование. Если есть превышение, моделирование завершить.

4. Присвоить процессу дескриптор исполнения и передать его планировщику процессов.

5. Выполнить планирование процессов: исполнение, ожидание, переключение, завершение.

6. В рамках процесса требуется исполнить обработчики событий, генерировать новые события.

7. Перейти к п. 2.

**ПРАКТИКА**

Удобным инструментом имитационного моделирования в подобных случаях являются системы дискретно-событийного моделирования. За сложным математическим термином скрываются вполне простые концепции:

* в соответствии с расписанием прилёта и вылета самолётов, в пространство модели вводятся "процессы" (в нашем случае, машины с водителями и пассажирами)
* эти машины являются "клиентами" трёх типов разделяемых, ограниченных по мощности "серверов": устройств выдачи парковочных талонов на въезде, устройства приёма парковочных талонов на выезде и автоматов оплаты
* при въезде, ожидании выезда и во время оплаты, машины и водители образуют очереди FIFO по принципу первый вошёл, первый вышел
* в качестве параметров модели задаётся расписание прихода машин, количество доступных серверов, время обслуживания и время ожидания пассажиров на территории аэропорта

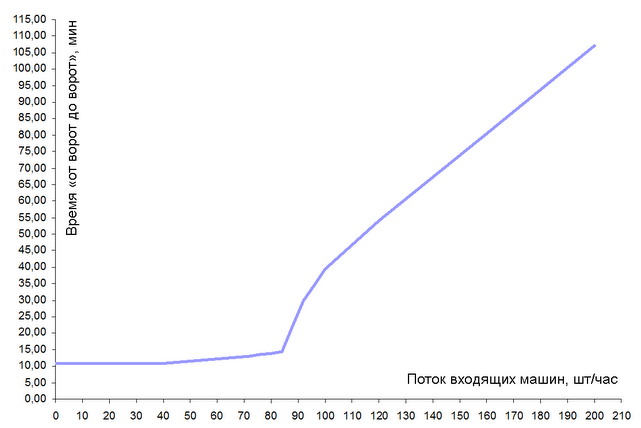
В соответствии с этими соображениями, дискретно-событийная модель будет выглядеть как полигон, на который в дискретные моменты времени прибывают независимые виртуальные машинки, действующие по заданной программе (встать в очередь на въезд - получить талон - дождаться пассажиров - встать в очередь на выезд - оплатить услуги) и реагирующие на события (освобождение или захват ресурса).

Для придания большей реалистичности, часть процессов можно описать с помощью случайных величин (например, с помощью распределения Пуассона, время прихода машин) - этот вариант имитационных моделей называется моделированием методом Монте-Карло (по аналогии с вероятностной природой казино). В нашем случае, чтобы не усложнять модель, я решил оставить всем параметрам детерминированные значения.

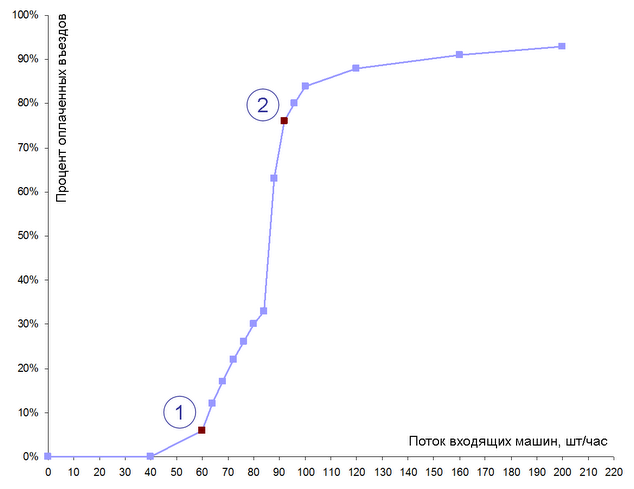
Первый вопрос, на который хотелось получить ответ: действительно ли, при ухудшении уровня сервиса будет расти выручка? Для ответа на этот вопрос был проведён ряд вычислительных экспериментов с разным числом входящих машин в течение часа (можно назвать это нагрузочным тестированием).

Под уровнем сервиса мы считаем время "от ворот до ворот", т.е. среднее время пребывания машин на территории, хотя правильнее было бы суммировать время нахождения в очереди на въезд и на выезд. На графике видно, что наблюдаемый мною эффект резкого увеличения времени ожидания в очереди, действительно, имеет место, когда среднее время пребывания становится равным 15 минутам и половина водителей начинает выходить из машин для оплаты. Очевидно, что интенсивность этого эффекта можно усугубить, уменьшив число работающих автоматов и увеличив время ожидания.

Машины поступали в систему в течение часа "партиями" каждые 15 минут. В результате, получился следующий график зависимости среднего времени пребывания машин на территории от потока машин:



Что происходит с выручкой? Лучшим показателем для отражения ситуации является процент машин, не попавших в 15-минутный бесплатный коридор от общего числа вошедшего транспорта. График числа оплаченных въездов приведён ниже:



Наша теория подтвердилась. Из графика видно, что низкая пропускная способность выгодна парковщикам.

 Из первого графика видно, что минимальное увеличение "бесплатного" интервала положительно влияет на пропускную способность (например, с 15 до 20 минут). Понятно, что обратной стороной этого улучшения является сокращение свободного пространства в зоне посадки и высадки пассажиров, что в какой-то момент может стать лимитирующим фактором (не смоделированного в нашем сценарии). Очевидно, что и здесь можно найти выход: нужно стимулировать водителей к короткому времени нахождения на территории аэровокзала. Как этого добиться? Во-первых, предоставить водителям актуальную информацию о прилёте самолётов.

schedule0.txt (в том же каталоге, что и сама программа). Пример записей файла:

0   20  
15   20  
30   20  
45   20

Каждая строка означает количество поступающих машин в определённый момент времени по часам модели (20 машин с 0-й минуты, 20 машин с 15-й и т.д.). Столбцы должны быть разделены символом табуляции.