Министерство образования и науки Российской Федерации
Архангельский государственный технический университет

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ЛЕСНОГО КОМПЛЕКСА

Рекомендовано УМО вузов по университетскому образованию в качестве учебного пособия при подготовке дипломированных специалистов по специальности 071900 «Информационные системы и технологии» направления 6547 «Информационные системы»

Архангельск

2004

Рассмотрено и рекомендовано к изданию советом УМО вузов
по университетскому политехническому образованию
17 декабря 2003 года

Рецензенты:

кафедра мебели и дизайна Архангельского государственного
технического университета;

Д.В. ТРУБИН, консультант Управления
анализа и развития департамента ЛПК
администрации Архангельской области, канд. с.-х. наук

Гурьев А.Т., Блок А.А. Имитационное моделирование процессов лесного комплекса: Учебное пособие. – Архангельск: Изд-во Арханг. гос. техн. ун-та, 2003. – 188 с.

Рассмотрены технологии анализа процессов лесного комплекса на основе имитационного моделирования. Рассмотрены теоретические основы имитационного моделирования. Приведены сведения о системе имитационного моделирования GPSS World (язык GPSS). Рассмотрены основы работы с системой визуального проектирования имитационных моделей Arena. Приведены примеры применения имитационного моделирования для анализа процессов лесного комплекса.

Предназначено для студентов специальности 071900 “Информационные системы и технологии” очной и заочной форм обучения, а также может быть использовано для специальностей лесного профиля. Может быть использовано научными работниками и аспирантами для исследования в области автоматизации и совершенствования действующих технологических процессов лесного комплекса.

Ил. 72. Табл. 8. Библиогр. 14 назв.

ISBN 5–261–00154–4

©Архангельский государственный
технический университет, 2003

ВВЕДЕНИЕ

Вопросы моделирования процессов и машин приобрели в настоящее время первостепенное значение.

Моделирование, как замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели, позволяет уменьшить ресурсы, необходимые для создания изучения, выбора и оптимизации технологических процессов.

Пособие является практическим руководством по моделированию процессов лесопромышленного комплекса с использованием систем моделирования GPSS World и Arena.

Методическое пособие состоит из пяти глав.

Первая глава посвящена изложению основ моделирования процессов лесного комплекса. Дано описание основных подходов к построению математических моделей систем: непрерывно-стохастических и сетевых моделей.

Во второй главе приведены принципы разработки современных средств имитационного моделирования. Описаны основные технологии моделирования систем. Приведен пример разработки специализированного вычислительного комплекса.

Третья глава содержит описание системы имитационного моделирования GPSS World.

В четвертой главе содержатся основные системы имитационного моделирования Arena.

Моделирование процессов лесного комплекса реализовано на примере лесосечных машин.

В пятой главе на конкретных примерах реализована методика формализации и моделирования комплексов лесосечных машин.

Учебное пособие снабжено необходимыми указаниями для работы с инструментальными средствами, реализующими GPSS World и Arena, а также необходимыми примерами для лучшего понимания этих методологий.

1. ОСНОВЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ПРОЦЕССОВ ЛЕСНОГО КОМПЛЕКСА

1.1. ПРОЦЕССЫ ЛЕСНОГО КОМПЛЕКСА

Лесной комплекс представляет собой сложенную систему, насчитывающую десятки тысяч предприятий различного профиля. В состав лесного комплекса входят следующие отрасли: лесное хозяйство, лесозаготовительная промышленность, деревообрабатывающая промышленность.

Рассмотрим особенности лесного комплекса как объекта моделирования на примере лесозаготовительной промышленности. Этой отрасли, как и другим, характерны:

- географическая разобщенность предприятий;
- расположение предприятий в отдаленных и труднодоступных районах;
- отсутствие средств связи и телекоммуникаций;
- сезонный характер лесозаготовок и производства продукции.

Современное предприятие лесного комплекса отличает разветвленная структура производства, связанная с необходимостью выполнения различных функций. По своей структуре оно относится к классу больших, сложных систем. Весь этот объем работ не остается постоянным, интенсивность их выполнения изменяется в течение суток, месяца, года. Причем это изменение происходит как планомерно, так и непредвиденно, случайно.

Лесное производство подвержено воздействию природных факторов. Это воздействие носит хаотичный, случайный характер и приводит к соответствующим колебаниям как производительности отдельных машин, так и всего процесса. Лесным процессам свойственны два вида причин неравномерности: систематические и случайные. Особенность систематических причин заключается в возможности их предварительного учета при планировании объемного выполнения работ. Заранее известны продолжительность и степень воздействия этих причин на планируемую операцию. Систематические причины полностью контролируются. К систематическим причинам, например, относятся: изменение объемов заготовки, вывозки и обработки лесоматериалов в зависимости от времени года и суток; ввод

дополнительных мощностей, новой технологии и оборудования; изменения организационного характера, например изменение сменности работы.

Наиболее существенно влияние систематических факторов сказывается на лесосечных работах. Процесс освоения любой лесосеки происходит циклично, с переменной интенсивностью: в начале ежедневные объемы рубок возрастают по мере завершения подготовительных работ и ввода в действие всех лесосечных машин, затем процесс стабилизируется и завершение освоения лесосеки происходит с убывающей интенсивностью в связи с окончанием работ на отдельных операциях и переводом высвобождаемой техники и людей на новую лесосеку.

Особенность функционирования систем лесосечных машин заключается в необходимости их перебазировки к новой лесосеке по мере завершения работ на предыдущей. При малых размерах лесосек время перебазировки существенно снижает производительность. Поэтому при расчете производительности системы лесосечных машин, оптимизации выбора систем машин и технологий приходится учитывать не только способ рубок, рельеф, почвенно-грунтовые условия и таксационные характеристики, но и нестационарность процесса, его цикличный характер к воздействию случайных факторов.

Специфика лесозаготовок заключается в необходимости одновременного и последовательного выполнения на лесосеке различных по назначению работ: рубки леса, отгрузки заготовленного древесного сырья, уборки лесосек и подготовки почвы к лесовосстановлению. Это накладывает определенные трудности в организации процесса.

Полностью влияние случайных причин не устраняется с вводом нового оборудования, технологии и организации производства. В результате этого воздействие некоторых из них может ослабевать, но будет возрастать воздействие других или будут возникать новые случайные причины. В любом случае останется случайная изменчивость параметров предмета труда – деревьев и хлыстов как продуктов природы. Не может быть ежедневно постоянной работоспособность рабочих, обслуживающих машины, как нельзя достичь постоянной стопроцентной надежности орудий производства, какими бы совершенными они ни были. Все эти случайные факторы воздействуют на любую операцию. Используя терминологию теории массового обслуживания, можно сказать, что одной из особенностей потоков древесины лесных процессов является их стохастический (вероятностный) характер. Детерминированных (регулярных) потоков древесины, таких, чтобы предметы труда (в данном случае деревья, хлы-

сты, бревна) поступали равномерно через строго определенные промежутки времени, не встречается.

Нерегулярность потоков приводит к образованию «узких мест» в производственном процессе, вызывает необходимость «сглаживания» отрицательного воздействия смежных машин друг на друга за счет размещения между ними буферных устройств, создания запасов сырья и продукции, а также приводит к снижению загрузки оборудования и удорожанию себестоимости продукции.

Не меньшие сложности возникают при проектировании машин вследствие непостоянства предмета труда и изменчивости природных условий. В отличие, например, от металлообрабатывающего станка лесная машина при каждом рабочем цикле должна функционировать в отличном от предыдущих циклов режиме: меняются усилие обработки, динамические нагрузки, качество обработки. Причем эти показатели существенно меняются в течение цикла. Характер такого изменения нестабильный, во многом случайный.

Система лесных машин, оптимально подобранная по своему составу в схеме применения для одних природно-производственных условий, может оказаться совершенно непригодной для работы в других условиях: варьируется режим работы машин (валочных, валочно-пакетирующих, валочно-трелевочных машин, трелевочных тракторов, лесовозных автомобилей, лесопильных рам и деревообрабатывающих станков) вследствие случайного колебания диаметра спиливаемого дерева, его породы, высоты, момента инерции, кривизны ствола, формы и размеров кроны, силы и направления ветра, массы, рейсовой нагрузки, изменения состояния проезжей части (волока, усов и веток) из-за погодных колебаний, сезона года и многих других причин. Таким образом, в качестве исходных данных для моделирования необходимо иметь количественные значения характеристик природно-производственных условий и их влияния на показатели технологических процессов и работу машин.

1.2. Основные понятия и принципы имитационного моделирования

1.2.1. Основные понятия

Моделирование представляет собой один из методов познания мира. Под моделью следует понимать любое, будь то мысленное, формальное, физическое или какое-либо другое представление объекта окружающего мира, обеспечивающее изучение некоторых свойств данного объекта. Необходимо отметить, что в общем смысле модель является также объектом. Этот объект замещает объект-оригинал и создается с целью исследования объекта-оригинала. В свою очередь, моделирование - это процесс создания модели.

Под системой понимают группу или совокупность объектов, объединенных какой-либо формой регулярного взаимодействия или взаимозависимости с целью выполнения определенной функции. Термин динамические системы применяется к системам, свойства которых изменяются с течением времени. В производственной деятельности с помощью методов моделирования решаются в основном вопросы исследования функциональных характеристик систем, их взаимодействия между собой, а также прогнозирования результатов функционирования систем.

Необходимость изучения количественных и качественных изменений исследуемых систем обусловило разработку и применение мощного математического аппарата для целей моделирования. На основе математического аппарата был создан целый класс методов моделирования, называемых аналитическими. Аналитические методы позволяют получить характеристики системы как некоторые функции параметров ее функционирования. Таким образом, аналитическая модель представляет собой систему уравнений, при решении которой получают параметры, необходимые для оценки системы (время ответа, пропускную способность и т.д.). Использование аналитических методов дает достаточно точную оценку, которая, зачастую, хорошо соответствует действительности. Смена состояний реальной системы происходит под воздействием множества как внеш-

них, так и внутренних факторов, подавляющее большинство из которых носят стохастический характер. Вследствие этого, а также большой сложности большинства реальных систем, основным недостатком аналитических методов является то, что при выводе формул, на которых они основываются и которые используются для расчета интересующих параметров, необходимо принять определенные допущения. Тем не менее, нередко оказывается, что эти допущения вполне оправданы [4].

С развитием вычислительной техники широкое применение получили имитационные методы моделирования для анализа систем, преобладающими в которых являются стохастические воздействия. В настоящее время дано много различных определений понятия "имитационного моделирования". Приведем два из них. Известный американский ученый Роберт Шеннон дает следующее определение: "Имитационное моделирование есть процесс конструирования модели реальной системы и постановки экспериментов на этой модели с целью либо понять поведение системы, либо оценить (в рамках ограничений, накладываемых некоторым критерием или совокупностью критериев) различные стратегии, обеспечивающие функционирование данной системы."[14] В свою очередь, один из ведущих теоретиков имитационного моделирования Алан Прицкер считает, что имитационное моделирование - это представление динамического поведения системы посредством продвижения ее от одного состояния к другому в соответствии с хорошо определенными операционными правилами [9].

Все имитационные модели используют принцип черного ящика. Это означает, что они выдают выходной сигнал системы при поступлении в нее некоторого входного сигнала. Поэтому в отличие от аналитических моделей для получения необходимой информации или результатов необходимо осуществлять "прогон" имитационных моделей, т. е. подачу некоторой последовательности сигналов, объектов или данных на вход модели и фиксацию выходной информации, а не "решать" их. Происходит своего рода "выборка" состояний объекта моделирования (состояния - свойства системы в конкретные моменты времени) из пространства (множества) состояний (совокупность всех возможных значений состояний). Насколько репрезентативной окажется эта выборка, настолько результаты моделирования будут соответствовать действительности. Этот вывод показывает важность статистических методов оценки результатов имитации. Таким образом, имитационные модели не формируют свое собственное решение в том виде, в каком это имеет место в аналитических моделях, а могут

лишь служить в качестве средства для анализа поведения системы в условиях, которые определяются экспериментатором.

1.2.2. Критерии применения имитационного моделирования

Применение имитационного моделирования целесообразно при наличии определенного условия. Эти условия определяет Р. Шеннон:

1. Не существует законченной математической постановки данной задачи, либо еще не разработаны аналитические методы решения сформулированной математической модели. К этой категории относятся многие модели массового обслуживания, связанные с рассмотрением очередей.
2. Аналитические методы имеются, но математические процедуры столь сложны и трудоемки, что имитационное моделирование дает более простой способ решения задачи.
3. Кроме оценки определенных параметров, желательно осуществить на имитационной модели наблюдение за ходом процесса в течение определенного периода [14].

Дополнительным преимуществом имитационного моделирования можно считать широчайшие возможности его применения в сфере образования и профессиональной подготовки. Разработка и использование имитационной модели позволяет экспериментатору видеть и "разыгрывать" на модели реальные процессы и ситуации.

1.2.3. Проблемы применения имитационного моделирования

Необходимо обозначить ряд проблем, возникающих в процессе моделирования систем. Исследователь должен акцентировать на них внимание и попытаться их разрешить, дабы избежать получения недостоверных сведений об изучаемой системе.

Первая проблема, которая касается и аналитических методов моделирования, состоит в нахождении "золотой середины" между упрощением и сложностью системы. По мнению Шеннона искусство моделирования в основном состоит в умении находить и отбрасывать факторы, не влияющие или незначительно влияющие на исследуемые характеристики системы. Нахождение этого "компромисса" во многом зависит от опыта, квалификации и интуиции исследователя. Если модель слишком упрощена и в

ней не учтены некоторые существенные факторы, то высока вероятность получить по этой модели ошибочные данные, с другой стороны, если модель сложная и в нее включены факторы, имеющие незначительное влияние на изучаемую систему, то резко повышаются затраты на создание такой модели и возрастает риск ошибки в логической структуре модели. Поэтому перед созданием модели необходимо проделать большой объем работы по анализу структуры системы и взаимосвязей между ее элементами, изучению совокупности входных воздействий, тщательной обработке имеющихся статистических данных об исследуемой системе.

Вторая проблема заключается в искусственном воспроизводстве случайных воздействий окружающей среды. Этот вопрос очень важен, так как большинство динамических производственных систем являются стохастическими, и при их моделировании необходимо качественное несмещенное воспроизведение случайности, в противном случае, результаты, полученные на модели, могут быть смещенными и не соответствовать действительности. Существует два основных направления разрешения этой проблемы: аппаратная и программная (псевдослучайная) генерация случайных последовательностей. При аппаратном способе генерации случайные числа вырабатываются специальным устройством. В качестве физического эффекта, лежащего в основе таких генераторов чисел, чаще всего используются шумы в электронных и полупроводниковых приборах, явления распада радиоактивных элементов и т.д. Недостатками аппаратного способа получения случайных чисел является отсутствие возможности проверки (а значит гарантии) качества последовательности во время моделирования, а также невозможности получения одинаковых последовательностей случайных чисел. Программный способ основан на формировании случайных чисел с помощью специальных алгоритмов. Этот способ наиболее распространен, так как не требует специальных устройств и дает возможность многократного воспроизведения одинаковых последовательностей. Его недостатками являются погрешность в моделировании распределений случайных чисел, вносимая по причине того, что ЭВМ оперирует с n-разрядными числами (т.е. дискретными), и периодичность последовательностей, возникающая в силу их алгоритмического получения. Таким образом, необходима разработка методов улучшения и критериев проверки качества генераторов псевдослучайных последовательностей.

Третьей наиболее сложной проблемой является оценка качества модели и полученных с ее помощью результатов (эта проблема актуальна и для аналитических методов). Адекватность моделей может быть оценена

методом экспертных оценок, сравнением с другими моделями (уже подтвердившими свою достоверность), по полученным результатам. В свою очередь, для проверки полученных результатов часть из них сравнивается с уже имеющимися данными.

1.2.4. Процесс имитационного моделирования

Процесс последовательной разработки имитационной модели начинается с создания простой модели, которая затем постепенно усложняется в соответствии с предъявляемыми решаемой проблемой требованиями. В каждом цикле создания программной модели можно выделить следующие этапы:

1. Формулирование проблемы: описание исследуемой проблемы, установление границ и ограничений моделируемой системы, определение целей исследования.
2. Разработка модели: переход от реальной системы к некоторой логической схеме (абстрагирование).
3. Подготовка данных: отбор данных, необходимых для построения модели, и представление их в соответствующей форме.
4. Трансляция модели: описание модели на языке имитационного моделирования.
5. Оценка адекватности: повышение до приемлемого уровня степени уверенности, с которой можно судить относительно корректности выводов о реальной системе, полученных на основании обращения к модели.
6. Планирование: определение условий проведения машинного эксперимента с имитационной моделью.
7. Экспериментирование: многократный прогон имитационной модели на компьютере для получения требуемой информации.
8. Анализ результатов: изучение результатов имитационного эксперимента для подготовки выводов и рекомендаций по решению проблемы.
9. Реализация и документирование: реализация рекомендаций, полученных на основе имитации, и составление документации по модели и ее использованию.[1,9,11]

1.2.5. Основные подходы к построению математических моделей систем

Важным этапом моделирования является создание математической модели исследуемой системы. На базе математической модели происходит анализ характеристик системы. При компьютерном моделирования на основе математической модели создается алгоритм программ для получения информации о поведении системы. Формальное описание объекта исследование необходимо также для взаимопонимания между специалистами разных областей, объединенных для решения какой-либо задачи.

В общем случае математическую модель любой динамической системы можно представить в следующем виде:

$$\vec{y}(t)=F(\vec{x},\vec{h},t),$$

где $\vec{x}$ - совокупность входных воздействий на систему,

$\vec{h}$ - совокупность внутренних параметров системы,

$\vec{y}$ - совокупность выходных характеристик системы,

$\boldsymbol{F}$ - закон функционирования системы.

Процесс функционирования системы можно рассматривать как последовательную смену состояний $z_1(t),z_2(t),\ldots,z_k(t)$:

$$\vec{z}(t)=G(\vec{z}^0,\vec{x},\vec{h},t),$$

где G - закон функционирования системы.

$\vec{z}^0$ - совокупность начальных состояний.

Таким образом, общую математическую модель системы можно также представить следующим образом:

$$\vec{y}(t)=F(\vec{z}(t)).$$

При построении математических моделей процессов функционирования систем можно выделить следующие основные подходы: непрерывно-детерминированный, дискретно-детерминированный, дискретно-стохастический, непрерывно-стохастический, сетевой, агрегативный. [11]

Для задач имитационного моделирования используются в основном непрерывно-стохастический и сетевой подходы к созданию математических моделей.

1.2.6 Непрерывно-стохастические модели (Q-схемы)

Непрерывно-стохастический подход применяется для формализации процессов обслуживания. Этот подход наиболее известен ввиду того, что

большинство производственных, а также экономических и технических систем по своей сути являются системами массового обслуживания.

В качестве процесса обслуживания могут быть представлены различные процессы функционирования экономических, производственных, технических и других систем.

Типовой математической схемой моделирования таких систем являются Q-схемы. В обслуживании можно выделить две элементарные составляющие: ожидание обслуживания и собственно обслуживание, а в любой системе массового обслуживания можно выделить элементарный прибор. Соответственно в этом приборе выделяют накопитель (Н) заявок, ожидающих обслуживания, некоторой емкостью; канал обслуживания (К); потоки событий (последовательность событий, происходящих одно за другим в какие-то случайные моменты времени): поток заявок на обслуживание w_i, характеризующийся моментами времени поступления и атрибутами (признаками) заявок (например, приоритетами), и поток обслуживания u_i, характеризующийся моментами начала и окончания обслуживания заявок (рис 1.1) .

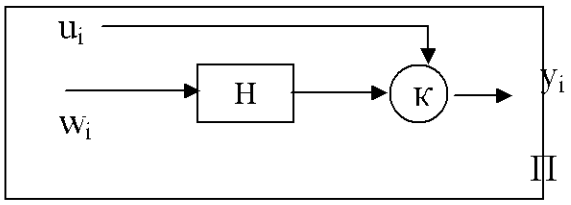


Рис. 1.1. Прибор обслуживания заявок

Различают потоки однородных и неоднородных событий. Поток событий называется однородным, если он характеризуется только моментами поступления этих событий и задается последовательностью $\{t_n\}=\{0<= t_1<= t_2 \ldots <= t_n <=\ldots\}$, где t_n - момент наступления n-го события (неотрицательное вещественное число).

Потоком неоднородных событий называется последовательность $\{t_n,f_n\}$, где f_n - набор признаков события (приоритет, принадлежность к какому-либо источнику и т.д.)

Рассмотрим поток, в котором события разделены интервалами времени, являющимися случайными величинами. Пусть интервалы независимы между собой. Тогда поток событий называется потоком без последействия.

Поток событий называется ординарным, если вероятность того, что на малый интервал времени Δt, примыкающий к моменту времени t, попа-

дает больше одного события $P_{>1}(t, \Delta t)$ пренебрежительно мала по сравнению с вероятностью того, что на этот же интервал времени Δt попадает ровно одно событие $P_1(t, \Delta t)$. Для любого интервала Δt верно следующее:

$$P_0(t, \Delta t)+ P_1(t, \Delta t)+ P_{>1}(t, \Delta t)=1$$

Как сумма вероятностей событий, образующих полную группу и несовместных. Тогда для ординарного потока событий справедливо следующее:

$$P_0(t, \Delta t)+ P_1(t, \Delta t)=1,\ P_{>1}(t, \Delta t)=0(\Delta t).$$

Стационарным потоком событий называется поток, для которого вероятность появления того или иного числа событий на интервале времени зависит лишь от длины этого интервала и не зависит от того, где на оси времени взят этот интервал.

Рассмотрим на оси времени ординарный поток событий и найдем среднее число событий, наступающих на интервале времени Δt:

$$0\cdot P_0(t, \Delta t)+1\cdot P_1(t, \Delta t)= P_1(t, \Delta t)$$

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

$$\lim_{\Delta t \to 0}[P_1(t,\Delta t)/\Delta t] = \lambda(t)$$

Для стационарного потока интенсивность постоянна.[1,11]

Процесс функционирования прибора П можно представить как процесс изменения состояний его элементов во времени z(t). Переход в новое состояние для прибора означает изменение количества заявок, которые в нем находятся. Таким образом, вектор состояний для прибора имеет вид $\vec{z} = (z^{н}, z^{к})$, где $z^{н}$ - состояние накопителя, $z^{к}$ - состояние канала.

Если каналы различных приборов обслуживания соединены параллельно, то имеет место многоканальное обслуживание (многоканальное Q-схема), а если приборы и их параллельные композиции соединены последовательно, то имеет место многофазное обслуживание (многофазная Q-схема). В зависимости от емкости накопителя разделяют системы с потерями (емкость накопителя равна нулю), системы с ожиданием (емкость накопителя неограниченна) и системы смешанного типа (емкость накопителя ограничена).

Неоднородность заявок, отражающая процесс в той или иной реальной системе, учитывается с помощью введения классов приоритетов. Различают статические и динамические приоритеты. Статические приоритеты назначаются заранее и не зависят от состояний Q-схемы. Динамические

приоритеты возникают при моделировании в зависимости от возникающих ситуаций. Исходя из правил выбора заявок из накопителя на обслуживание каналом можно выделить относительные и абсолютные приоритеты. Относительный приоритет означает, что заявка с более высоким приоритетом, поступившая в накопитель, ожидает окончания обслуживания предшествующей заявки каналом и только после этого занимает канал. Абсолютный приоритет означает, что заявка с более высоким приоритетом, поступившая в накопитель, прерывает обслуживание каналом заявки с более низким приоритетом и сама занимает канал.

При рассмотрении алгоритмов функционирования приборов обслуживания необходимо задать также набор правил, по которым заявки покидают накопители и каналы, например: блокировки по входу и выходу, маршрутизация заявок.

Аналитически модели систем массового обслуживания исследуются с помощью систем дифференциальных уравнений. Переменными в этих уравнениях являются вероятности переходов между состояниями (состояния, в свою очередь, определяются количеством и местонахождением заявок в системе).

Формализованное описание системы массового обслуживания для имитационного моделирования представляет собой графическое отображение (с соответствующими комментариями) элементарных приборов, клапанов, отражающих наличие управляющих воздействий, и связей между ними. На данный момент существует много программных средств, специализирующихся на создание имитационных моделей на основе Q-схем. Примерами для прикладного применения непрерывно-стохастического подхода могут служить исследования потоков поставок продукции некоторому предприятию, потоков деталей и комплектующих изделий на сборочном конвейере цеха и т.п.[11]

1.3. Сетевой подход к построению математических моделей систем

1.3.1. Общие понятия сетей Петри

Сетевые модели (сети Петри) используется для анализа причинно-следственных связей в сложных системах. Аппарат теории сетей Петри позволяет описывать структуру и взаимодействие параллельных систем и процессов. Сеть Петри (N-схема) задается четырьмя элементами:

$$N = \langle B,D,I,O \rangle,$$

где B - конечное множество позиций; D - конечное множество переходов; I - входная функция (прямая функция инцидентности), $I:BXD \rightarrow \{0,1\}$; O - выходная функция (обратная функция инцидентности), $O:DXB \rightarrow \{0,1\}$.

I отображает переход d_j в множество входных позиций $b_i \in I(d_j)$, а выходная функция O отображает переход d_j в множество выходных позиций $b_i \in D(d_j)$. Для каждого $d_j \in D$ можно определить множество входных позиций перехода $I(d_j)$ и выходных позиций перехода $O(d_j)$ как

$$I(d_j) = \{b_i \in B \mid I(b_i, d_j) = 1\}$$

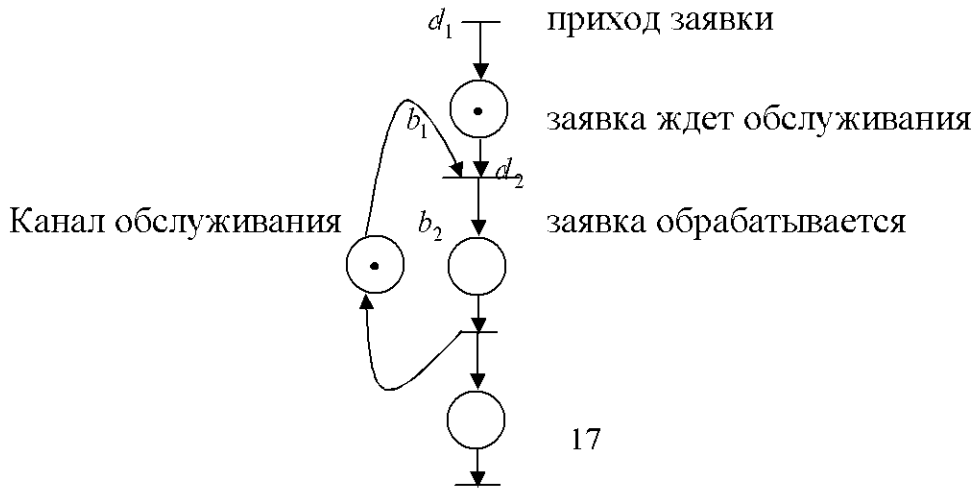
$$O(d_j) = \{b_i \in B \mid I(d_j, b_i) = 1\}$$

Аналогично, для каждого $b_i \in B$ можно определить множество входных переходов позиции $I(b_i)$ и выходных переходов позиции $O(b_i)$:

$$I(b_i) = \{d_j \in D \mid I(d_j, b_i) = 1\}$$

$$O(b_i) = \{d_j \in D \mid O(b_i, d_j) = 1\}$$

Графически сеть Петри изображается в виде двудольного ориентированного мультиграфа, представляющего собой совокупность позиций и переходов (рис. 1.2).



свободен b_3 (канал занят)

d_3 конец обслуживания

b_4

заявка ждет вывода

d_4

выход заявки

Рис. 1.2. Структура N-схемы

Для представления динамических свойств объекта вводится функция маркировки (разметки) $M:B\rightarrow\{0,1,2,\ldots\}$. Маркировка - присвоение абстрактных объектов, называемых метками (фишками), позициям:

$$N_M = \langle B,D,I,O,M \rangle.$$

Функционирование сети Петри отражается путем перехода от разметки к разметке. M_0 - начальная разметка. Смена разметок - срабатывание одного из переходов сети. Необходимое условие срабатывания перехода d_j:

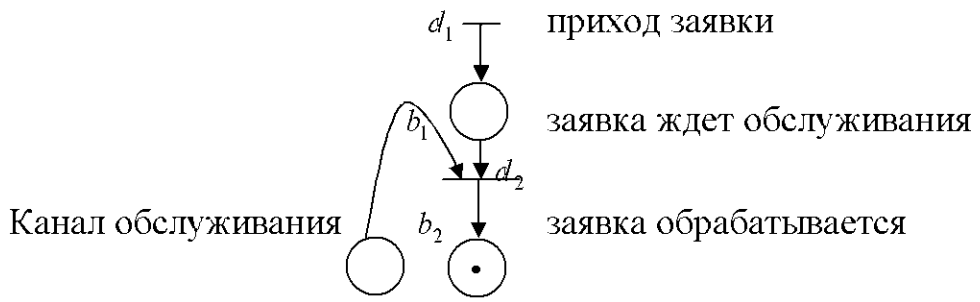
$$b_i \in I(d_j)\ \{M(b_i)\geq 1\},$$

где $M(b_i)$ - разметка позиции b_i.

Срабатывание перехода d_j изменяет разметку сети $M(b)$ на $M'(b)$ по следующему правилу:

$$M'(b)=M(b)-I(d_j)+O(d_j),$$

т.е. переход d_j изымает по одной метке из каждой своей входной позиции и добавляет по одной метке в каждую из выходных позиций. Таким образом, схема на рис. 1.2 при срабатывании перехода d_2 сменится на следующую схему (рис. 1.3):



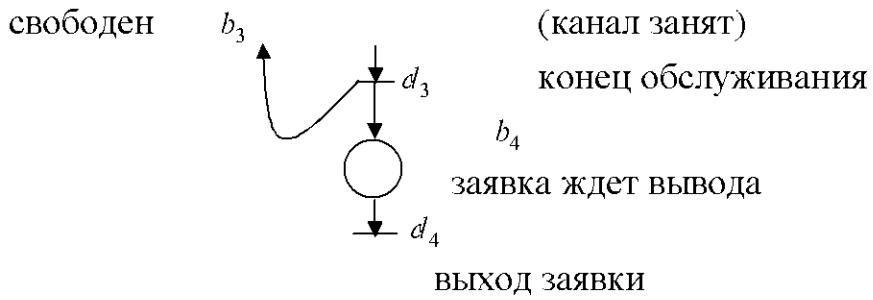


Рис. 1.3. Изменение разметки

Для отражения временных параметров процесса функционирования моделируемой системы на базе N-схем используется расширение аппарата сетей Петри - временные сети. [7,11]

Сети Петри представляют удобный математический аппарат для моделирования параллельных технологических процессов с разделяемыми ресурсами. Преимуществом сетей Петри также является легкость построения иерархических конструкций, что позволяет сначала исследовать отдельные подсистемы, а затем, объединяя уже созданные модели, всю систему в целом. Модели, построенные на основе сетей Петри, предназначены для анализа с помощью имитации на компьютере. Такие модели довольно легко реализуются программно даже с помощью универсальных языков программирования. Необходимо также отметить, что на сегодняшний день практически все компиляторы и операционные системы оптимизируются с помощью методов анализа сетей Петри. Приведем примеры описания реальных систем сетями Петри.

1.3.2. Разработка моделей вычислительных сетей предприятий лесного комплекса

Создание единой информационной среды взаимодействующих участников процессов лесного комплекса предполагает организацию распределенных вычислительных сетей. Проектирование оптимальной структуры таких сетей представляет достаточно сложную задачу. В качестве инструмента для проектирования распределенных вычислительных сетей видится целесообразным использование аппарата сетей Петри.

Упрощенная модель сегмента вычислительной сети с общей средой передачи (с общей шиной - коаксиальное соединение, или на основе витой пары с разветвителем (HUBом)) на языке сетей Петри представлена на рис. 1.4.

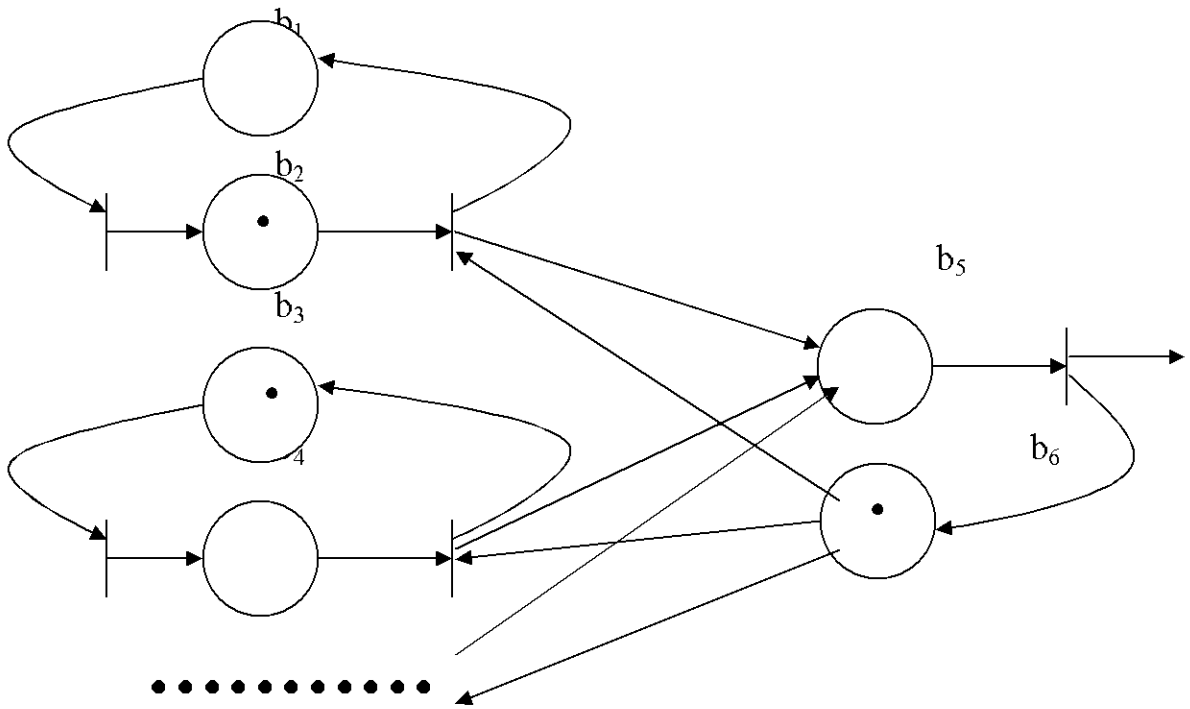
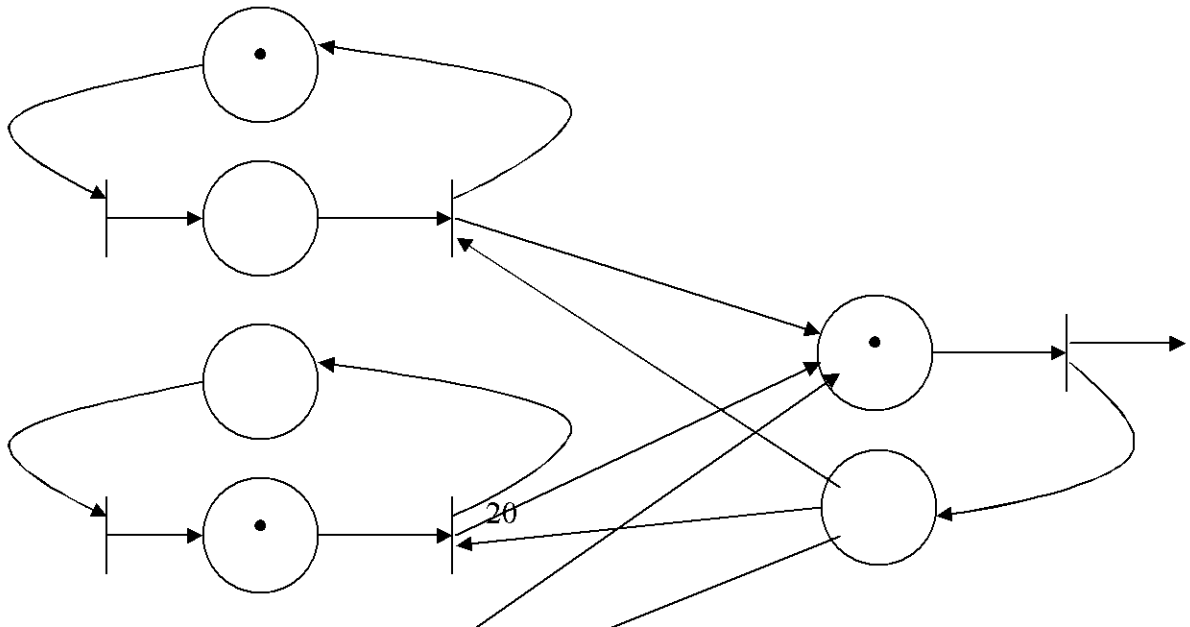


Рис. 1.4. Модель сегмента вычислительной сети.

Маркировка позиции b_1 означает, что у первого узла сети нет данных для передачи по сети, b_2 - есть данные для передачи по сети, соответственно позиции b_3 , b_4 - для второго узла и т.д. Позиция b_5 - передача данных по общей среде передаче данных (СПД), b_6 - СПД свободна. Данная маркировка (рис. 1.4) означает готовность к передаче данных первого узла и незанятость СПД. Возможна следующая смена маркировки (рис. 1.5). Данные по СПД передаются от первого узла, также к передаче данных готов второй узел, но он не сможет приступить к передаче, пока занята СПД. Общая модель всей вычислительной сети будет представлять композицию моделей ее сегментов. Такая схема пригодна только для анализа информационных потоков в сети на макроуровне. Она не отражает особенности сетевых технологий, используемых в сегментах вычислительных сетей. Поэтому с помощью этой модели нельзя оценить с большой точностью, например, потенциальную загрузку проектируемой сети.



• • • • • • • • • • •

Рис. 1.5. Модель сегмента вычислительной сети.

Большое влияние на характеристики производительности сети при использовании стандарта Ethernet оказывают коллизии.

Модель механизма обнаружения коллизий для двух узлов сети может быть представлена следующим образом:

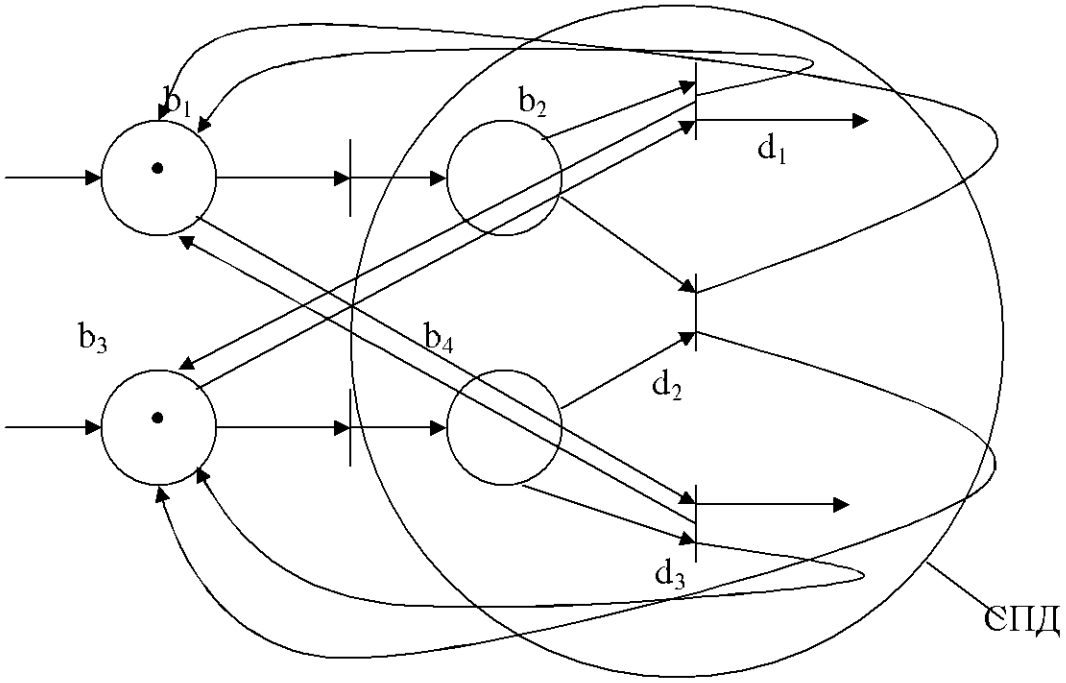


Рис. 1.6. Модель механизма коллизий для двух узлов.

Срабатывание перехода d_1 означает, что передача пакета данных от первого узла нормально завершена. Срабатывание перехода d_2 - произошла коллизия и попытка передачи будет повторена. Перехода d_3 сработает, если пакет данных от второго узла нормально передан. В случае модели механизма коллизий для большего количества узлов - сложность модели возрастает по экспоненте. Этот пример также иллюстрирует тот факт, что при большей детализации технологических процессов сложность моделей стремительно возрастает, что значительно увеличивает временные затраты

на их разработку. В связи с этим необходимо ранжировать детализацию моделей в зависимости от целей исследования.

1.3.3. Моделирование лесозаготовительных процессов

Рассмотрим использование аппарата сетей Петри для моделирования лесозаготовительных процессов. Лесозаготовительный пункт можно представить следующей структурной схемой:

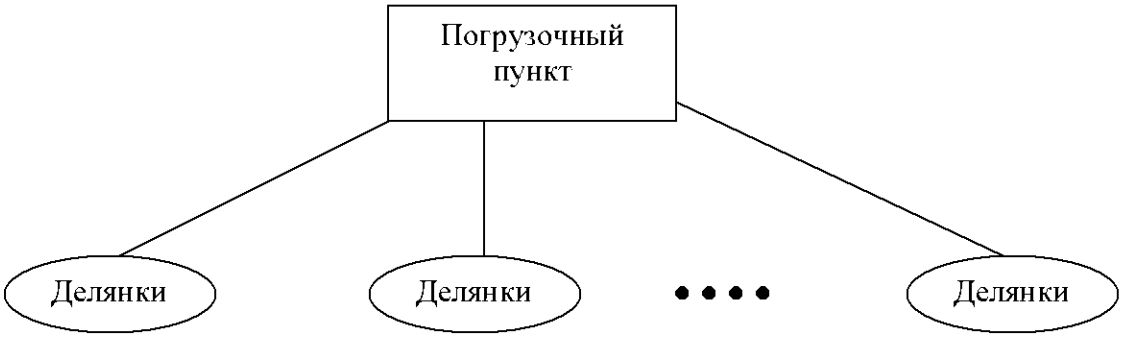


Рис. 1.7. Структурная схема лесозаготовительного пункта

Тогда модель работы лесозаготовительного пункта в виде сети Петри примет следующий вид:

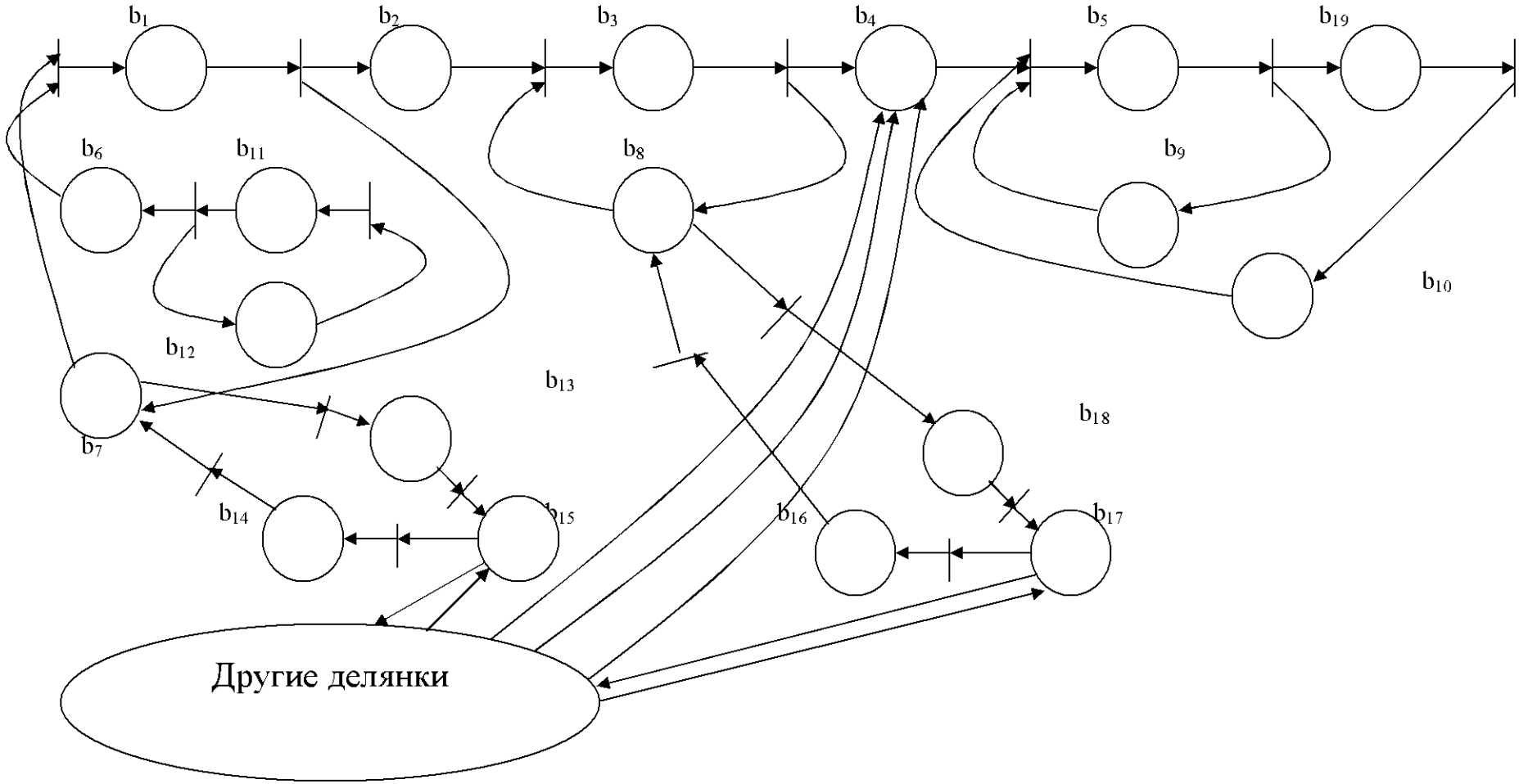


Рис. 1.8. Модель лесозаготовительных процессов

Значение позиций: b_1 - валка и пакетирование, b_2 - ожидание трелевки, b_3 - трелевка, b_4 - ожидание погрузки, b_5 - погрузка, b_6 - запас древесины на делянке, b_7 - ресурсы для валки, b_8 - ресурсы для трелевки, b_9 - ресурсы для погрузки, b_{10} - наличие лесовоза, b_{11} , b_{12} - лесовосстановительные ресурсы, b_{15} - общие для лесозаготовительного пункта ресурсы для валки и пакетирования, b_{17} - общие для лесозаготовительного пункта ресурсы для трелевки, b_{13} , b_{14} , b_{16} , b_{18} - транспортировка ресурсов.

1.3.4. Моделирование процессов кредитования предприятий лесного комплекса

Финансовое обеспечение предприятий лесного комплекса зависит во многом от стабильной и эффективной работы банковской инфраструктуры. При решении задач оптимизации финансовой деятельности кредитных организаций могут быть использованы сети Петри. Упрощенно процесс кредитования можно представить следующей N-схемой (рис. 1.9).

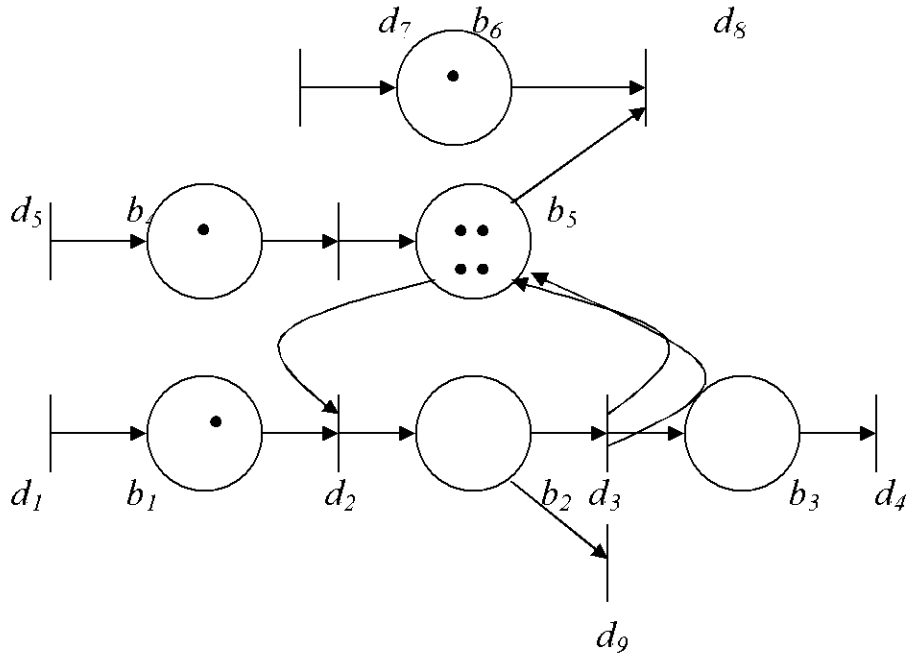


Рис. 1.9. N-схема кредитования

Разметка позиции b_5 задает объем кредитных ресурсов, которые состоят из собственных средств банка (определяются начальной разметкой) и привлеченных ресурсов. Привлечение ресурсов происходит по переходу d_5 , позиции b_4 и срабатыванию перехода d_6 ; соответственно, изъятие (возвращение привлеченных средств - в основном депозитов) происходит по

переходу d_7, позиции b_6 и срабатыванию перехода d_8. Приход заявки на кредит происходит по переходу d_1 , позиции b_1. Выдача ссуды означает срабатывание перехода d_2 , которое возможно только при наличии маркировки позиции b_1 (т.е. наличия заявки на ссуду) и маркировки позиции b_5 (наличия кредитных ресурсов). Позиция b_2 означает использование ссуды, срабатывание перехода d_3 - означает возвращение ссуды с процентами и конец обслуживания заявки на кредит (b_3, d_4). Невозвращение ссуды интерпретируется как срабатывание перехода d_9.

1.3.5. Анализ сетей Петри

Существует два подхода к анализу свойств сетей Петри - это статистическое моделирование на компьютере и аналитические методы. Большое практическое значение имеет изучение следующих свойств сетей Петри: ограниченность, сохранение, активность, достижимость и покрываемость. Необходимо отметить, что задачи нахождения этих свойств разрешимы для классических сетей Петри и неразрешимы для расширений сетей Петри (например, Е-сетей).

Рассмотрим свойство ограниченности. Позиция является к-ограниченной, если количество фишек в ней не может превышать целое число k: $M(b_i) \leq k$, для всех $M \in R(N,M)$ (множество достижимости). 1-ограниченная позиция называется безопасной. Сеть Петри ограниченна, если все ее позиции ограничены. Изучение данного свойства может иметь важное практическое значение при проектировании технических систем, в которых должно быть ограниченно присутствие одновременное присутствие активных процессов, например моделирование транспортных потоков или систем управления трафиком в глобальных вычислительных сетях. Такие системы должны быть обеспечены механизмами автоматической разгрузки или распределения.

Важным свойством для сетей Петри является сохранение. Сеть Петри является с начальной маркировкой M называется сохраняющей по отношению к вектору взвешивания $W=(w_1, w_2,..., w_n)$, $n=|B|$, $w_i > 0$, если для всех возможных маркировок выполняется следующее условие: $\sum w_i \cdot M'(b_i) = \sum w_i \cdot M(b_i)$.

Строго сохраняющая сеть Петри является сохраняющей по отношению к вектору взвешивания (1,1,...,1). Важное значение это свойство может иметь для моделирования систем распределения ресурсов.

При распределении ресурсов между параллельными процессами системы могут возникать тупики. Особенно остро эта проблема проявляется в области вычислительной техники. Тупик в сети Петри - это множество переходов, которые не могут быть запущены. Выявление таких тупиков можно назвать исследованием сети Петри на активность.

Очевидными являются задачи достижимости и покрываемости. Задача достижимости: можно ли из данной маркировки M достичь маркировки M'?. Задача покрываемости: для данной маркировки M существует ли достижимая маркировка M', такая что M'>=M.

В некоторых возможных задачах типа достижимости могло бы игнорироваться содержимое некоторых позиций и приниматься во внимание сравнение или покрытие содержимого несколько важных позиций. Таким образом, мы можем рассматривать достижимость и покрываемость "по модулю" множества позиций. Эти задачи называются задачами достижимости подмаркировки и покрываемости подмаркировки. [7]

Для задач имитационного моделирования наибольшие возможности имеют расширения сетей Петри, в особенности аппарат Е-сетей.

1.3.6. Е-сети

Оценочные или *Е-сети* были предложены Г. Натом как расширение сетей Петри и средство описания моделей функционирования вычислительных систем. Это наиболее мощное расширение сетей Петри. Е-сеть задается совокупностью следующих множеств:

$$N=\langle B, B_P, B_R, D, M_0 \rangle,$$

где B - конечное множество позиций, $B_P \subset B$ - конечное множество периферийных позиций, $B_R \subset B$ - конечное множество решающих позиций, D - множество описаний переходов $d_i=(s, t(d_i), \rho)$ (s - тип перехода, $t(d_i)$ - время перехода, ρ - процедура перехода).

Е-сети являются безопасными, т.е. для всех позиций M(b)<=1, однако выполнение этого условия поддерживается в сети искусственно, за счет изменения логики работы перехода.

Важной особенностью Е-сети является детализация представления метки. С каждой меткой в Е-сети связаны n описателей. Значение i-го описателя в позиции b_k обозначается как $M(b_k(i))$. Каждый из описателей метки несет в себе определенную количественную информацию о моделируемом объекте. Переход Е-сети моделирует некоторое событие не только на уровне выполнения всех необходимых условий, но и отражает также ряд

операций, связанных с данным событием, посредством модификации описателей меток. Набор операций и условия их выполнения задаются *процедурой перехода p*.

Для Е-сетей определены пять основных типов переходов (рис. 1.10). Логика работы переходов задается указанием разрешенных смен разметок. Срабатывание перехода типа T_E (рис. 1.10,*а)* происходит при наличии метки во входной позиции b_1 и отсутствии метки в выходной позиции b_2, т.е.

$$(1,0) \xrightarrow{T_E} (0,1)$$

Для перехода F_E (рис. 1.10, б)

$$(1,0,0) \xrightarrow{F_E} (0,1,1).$$

Переход объединения типа J_E (рис. 1.10, в) описывается как

$$(1,1,0) \xrightarrow{J_E} (0,0,1).$$

Управляемый переход разветвления X_E (рис. 1.10, г) с управляющей позицией $b_1 \in B_R$ задается соотношениями:

$$(0,1,0,0) \xrightarrow{X_E} (0,0,1,0);$$

$$(0,1,0,1) \xrightarrow{X_E} (0,0,1,1);$$

$$(1,1,0,0) \xrightarrow{X_E} (0,0,0,1);$$

$$(1,1,1,0) \xrightarrow{X_E} (0,0,1,1).$$

Приоритетный переход Y_E (рис. 1.10, д), где $b_1 \in B_R$, задается следующим набором:

$$(0,1,1,0) \xrightarrow{Y_E} (0,0,1,1);$$

$$(0,1,0,0) \xrightarrow{Y_E} (0,0,0,1);$$

$$(0,0,1,0) \xrightarrow{Y_E} (0,0,0,1);$$

$$(1,1,1,0) \xrightarrow{Y_E} (0,1,0,1);$$

$$(1,1,0,0) \rightarrow (0,0,0,1);$$
$$Y_E$$
$$(1,0,1,0) \rightarrow (0,0,0,1).$$

Приведенные выше пять основных типов переходов позволяют моделировать различные ситуации, встречающиеся в реальных системах. Переход T_E моделирует событие, наступающее при выполнении одного условия. В случае необходимости двух условий используется переход J_E. Разветвление потока информации отображается в переходе F_E. При необходимости изменения направления потока информации по некоторому условию используется переход типа X_E. Переход типа Y_E отражает приоритетность, устанавливаемую для одних потоков информации по отношению к другим. При этом в зависимости от внутренней логики позиции возможно представление различных приоритетов для потоков меток: фиксированное, в виде функции от описателей меток или от внешних переменных системы.

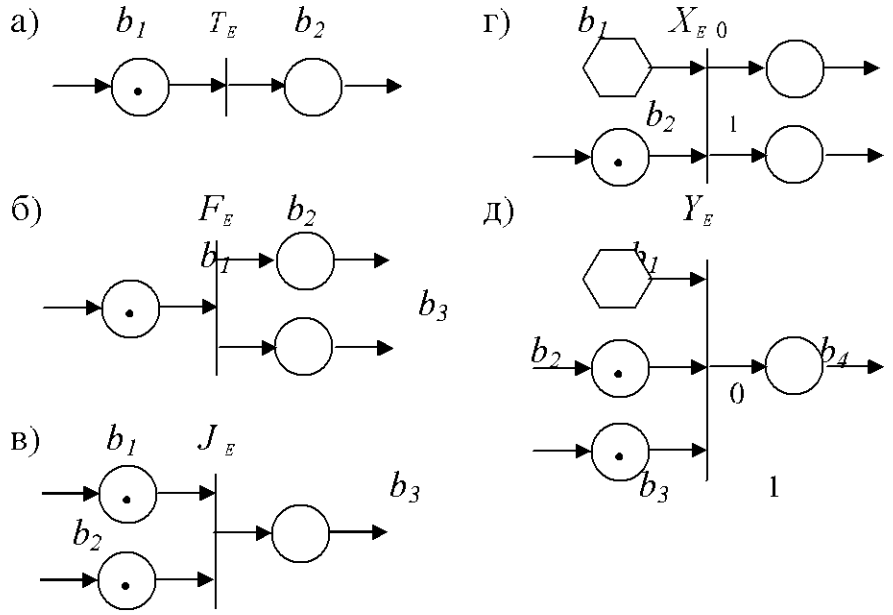


Рис. 1.10. Базовый выбор Е-сетевых переходов.

Позиции $b_i \in B_R$ могут являться в сети как внутренними, так и периферийными. Если $b_i \in B_P$, то состояние позиции может быть нулевым $M(b_i)=0$, единичным $M(b_i)=1$ или же неопределенным $M(b_i)=\varnothing$. Для перевода позиции из неопределенного состояния в нулевое или единичное используется *решающая процедура* перехода

$$r:[P_1 \rightarrow M(b_i):=\alpha,\ P_2 \rightarrow M(b_i):=1-\alpha],$$

где P_1, P_2 - предикаты, принимающие два значения 1 или 0 ($\alpha=\{0, 1\}$).

Выполнение решающей процедуры начинается с вычисления P_1. Если P_1 истинно, то $M(b_i):=\alpha$ и далее оценка не производится. Если же P_1 ложно, то вычисляется P_2. В том случае, если и P_2 оказывается ложным, состояние b_i остается неопределенным и повторное выполнение решающей процедуры производится только после изменения хотя бы одного из аргументов P_1 или P_2.

Если решающая позиция $b_i \in (B/B_P)$, то ее состояние устанавливается обычным для сетей Петри образом в результате срабатывания переходов сети.

Функционирование перехода Е-сети определяется как последовательность четырех фаз.

1) Фаза псевдоготовности присутствует для тех переходов d_j, для которых все входные нерешающие позиции промаркированы (все позиции маркировка которых является необходимым условием срабатывания перехода). В течении этой фазы выполняется решающая процедура перехода.

2) фаза готовности: вычисляется $t(d_i)$ и устанавливается активная фаза.

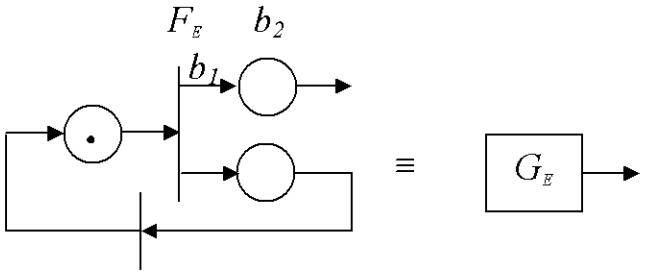
3) активная фаза: длительность фазы определяется вычисленным значением $t(d_i)$. В течение этой фазы производятся операции над описателями меток.

4) Заключительная фаза - изменение разметки в соответствии с уравнениями перехода.

Составление Е-сетевой модели системы значительно упрощается при использовании иерархического подхода к структуре модели, когда ряд наиболее часто встречающихся подсетей выделяется в качестве макроэлементов - макропозиций и макропереходов. Необходимыми элементами Е-сетевой модели являются обычно макропозиции генератора G_E и поглощающая метки макропозиция A_E. На рис. 1.11, а, б показаны подсети, образующие данные Е-сетевые макроэлементы.[8]

Примеры Е-сетевых моделей приведены в главе 5.

а)



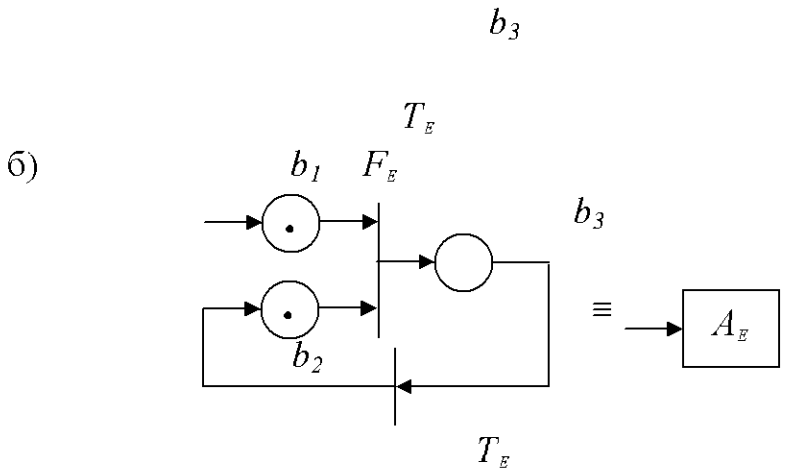


Рис 1.11. Макропозиции генерации и поглощения меток

1.4. Статистическое моделирование

1.4.1. Общая характеристика метода статистического моделирования

При создании программных имитационных моделей стохастических систем используется метод статистических испытаний Монте-Карло, который базируется на использовании случайных чисел, то есть возможных значений некоторой случайной величины с заданным распределением вероятностей. Статистическое моделирование представляет собой метод получения с помощью ЭВМ статистических данных о процессах, происходящих в моделируемой системе.

Сущность метода статистического моделирования заключается в построении для процесса функционирования исследуемой системы S некоторого моделирующего алгоритма, имитирующего поведение и взаимодействие элементов системы с учетом случайных входных воздействий и воздействий внешней среды и реализации этого алгоритма с использованием вычислительной техники.

Этот метод применим как для изучения стохастических систем; так и решения детерминированных задач.

Результатом статистического моделирования является серия частных значений искомых величин или функций, их статистическая обработка. Если количество реализаций (испытаний) $N \to \infty$, результаты устойчивы и достаточно точны.

Теоретическая основа метода статистического моделирования являются предельные теоремы теории вероятностей. Их значение - гарантиру-

ют высокое качество статистических оценок при числе испытаний $N \to \infty$. Часто приемлемые результаты могут быть получены при достаточно небольших N.

Неравенство Чебышева. Для неотрицательной случайной величины X и любого K>0 выполняется неравенство:

$$P(X>=K)=<M(X)/K$$

Теорема Бернулли. Если проводится N независимых испытаний, в каждом из которых некоторое событие A осуществляется с вероятностью p, то относительная частота появления события m/N при $N \to \infty$ сходится по вероятности к p, т.е. при любом $\varepsilon>0$

$\lim\limits_{N\to\infty} P(|m/N - p| \geq \varepsilon) = 0$, где m - число положительных исходов испытания.

Теорема Пуассона.

$\lim\limits_{N\to\infty} P(|m/N - \frac{1}{N}\sum_{i=1}^{N} p_i| \geq \varepsilon) = 0$, где p_i - вероятность осуществления события A в i-м испытании.

Обобщенная теорема Чебышева.

$\lim\limits_{N\to\infty} P(|\frac{1}{N}\sum_{i=1}^{N} X_i - \frac{1}{N}\sum_{i=1}^{N} M(X_i)| \geq \varepsilon) = 0$, X_i - i-ая случайная величина

Теорема Маркова. Обобщенная теорема Чебышева справедлива и для зависимых случайных величин, если

$$\lim_{N\to\infty} P(|\frac{1}{N^2} D[\sum_{i=1}^{N} X_i] = 0$$

Центральная предельная теорема. Если X_1, X_2,..., X_n - независимые одинаково распределенные случайные величины, имеющие математическое ожидание $M(X_i)=a$ и дисперсию σ^2, то при $N \to \infty$ закон распределения суммы $\sum_{i=1}^{N} X_i$ неограниченно приближается к нормальному:

$$\lim P(\alpha < (\sum_{i=1}^{N} X_i - Na)/\sqrt{N}\sigma < \beta) = \frac{1}{\sqrt{2\pi}}\int_{\alpha}^{\beta} e^{-t^2/2} dt = \Phi_0(\beta) - \Phi_0(\alpha).$$

Приведем примеры применения методов статистического моделирования.

Пример 4.1. Входное воздействие - $x = 1 - e^{-\lambda}$, воздействие внешней среды - $\vartheta = 1 - e^{-\varphi}$, λ и φ - случайные величины с известными функциями распределения.

$y = \sqrt{x^2 + \vartheta^2}$. Необходимо определить математическое ожидание величины y:

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Схема алгоритма (ген. – генерация):

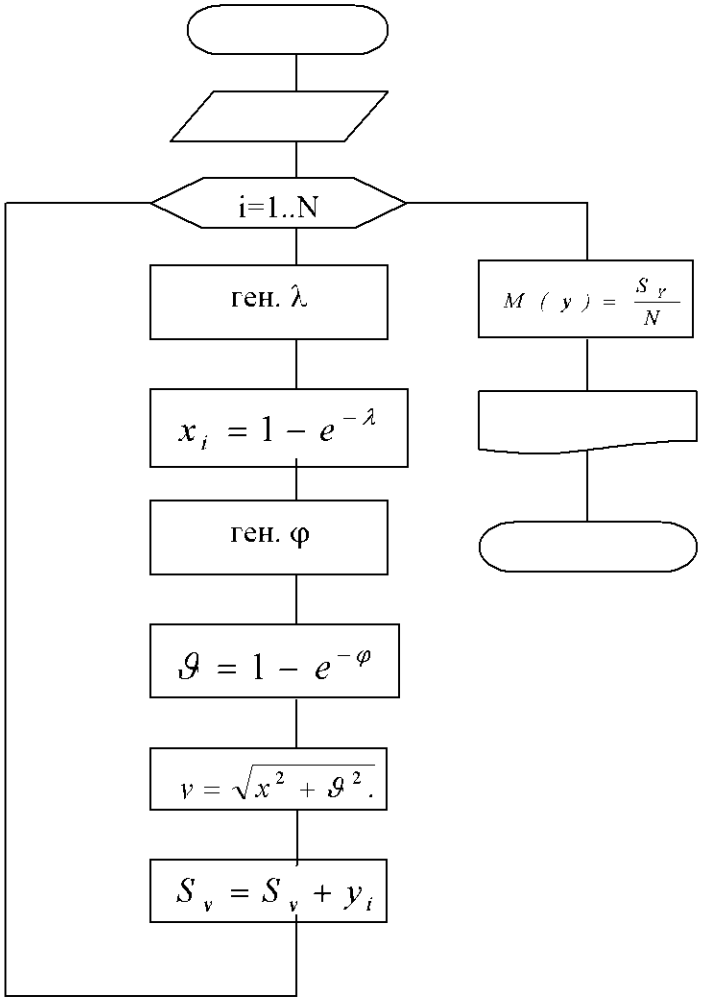


Рис. 1.12. Алгоритм определения математического ожидания

Следующий пример иллюстрирует применения метода статистического моделирования для решения детерминированных задач. [1,11]

Пример. Необходимо найти площадь фигуры:

$$\begin{cases} \alpha = 1, \\ \gamma = f(\alpha), \\ 0 \le f(\alpha) \le 1, \\ 0 \le \alpha \le 1 \end{cases}$$

$$S = \int_0^{\alpha} f(\alpha) d\alpha$$

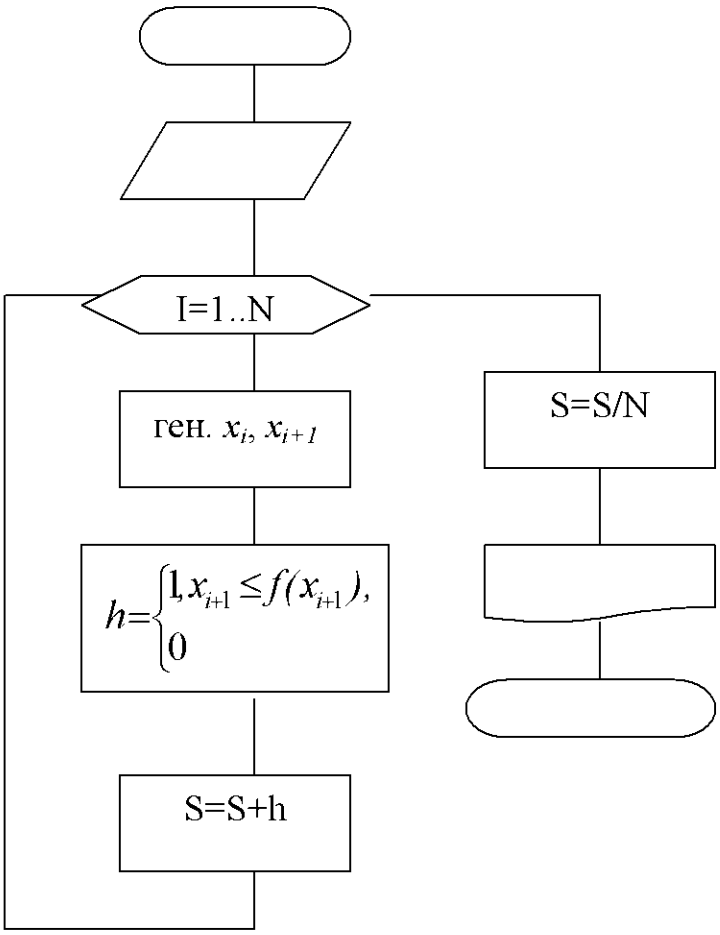


Рис. 1.13. Алгоритм нахождения площади фигуры

Пример. Проводится s=10 независимых выстрелов по мишени, причем вероятность попадания при одном выстреле задана и равна p. Требуется оценить вероятность того, что число попаданий в мишень будет четным.

Аналитическое решение этой задачи:

$$P = \sum_{k=0}^{5} C_{10}^{2k} p^{2k} (1-p)^{10-2k}$$

Схема алгоритма (статистическое моделирование):

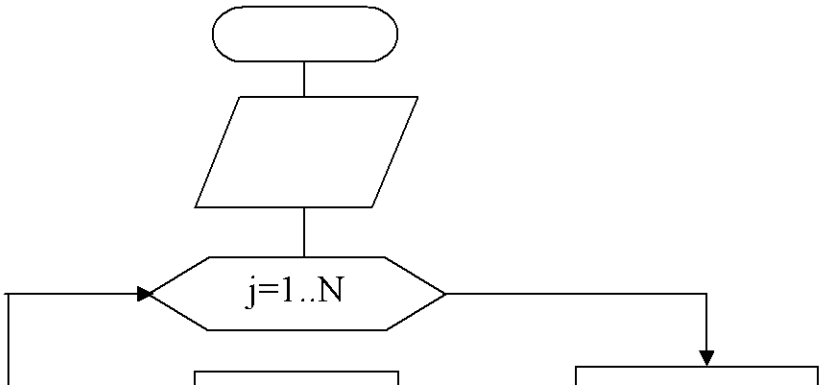


Рис. 1.14. Алгоритм нахождения вероятности

1.4.2. Генерация последовательностей случайных чисел

Существует три способа генерации случайных чисел:

1. Аппаратный - в основе лежит какой-либо физический эффект (например, шумы в электронных устройствах), случайные числа вырабатываются с помощью специального датчика. Этот способ не гарантирует

качество последовательности случайных чисел непосредственно во время моделирования. С помощью этого способа нельзя получать одинаковые последовательности. Используется редко.

2. Табличные - случайные числа оформлены в виде таблицы в оперативной памяти или на внешнем носителе. При этом способе запас чисел ограничен, вычислительные ресурсы используются неэффективно. Используется редко.

3. Программный (алгоритмический) - случайные числа формируются с помощью специальных программ. Каждое случайное число вычисляется с помощью соответствующей программы по мере возникновения потребностей при моделировании системы на ЭВМ. Этот способ наиболее распространен.

Программная имитация случайных воздействий сводится к генерированию некоторых стандартных (базовых) процессов и к их последующему функциональному преобразованию. Чаще всего в качестве базовой последовательности используют независимые случайные величины, равномерно распределенные на интервале (0,1).

Непрерывная случайная величина имеет равномерное распределение в интервале (a,b), если ее функции плотности и распределения соответственно примут вид

$$f(x)=\begin{cases}1/(b-a), a\le x\le b,\\ 0,\ x<a, x>b;\end{cases}$$

$$F(x)=\begin{cases}0, x<a,\\ (x-a)/(b-a), a\le x\le b,\\ 1, x>b.\end{cases}$$

$$M(X)=\int_a^b xf(x)dx=\int_a^b xdx/(b-a)=(a+b)/2;$$

$$D(X)=\int_a^b (x-M(x))^2 f(x)dx=(b-a)^2/12.$$

Для получения случайных чисел на ЭВМ используются алгоритмы, поэтому такие последовательности, являющиеся по сути детерминированными, называются псевдослучайными. ЭВМ оперирует n-разрядными числами, поэтому на ЭВМ вместо непрерывной совокупности равномерных случайных чисел интервала (0,1) используют дискретную последовательность 2^n случайных чисел того же интервала - закон распределения такой

дискретной последовательности называется квазиравномерным распределением.

Требования к идеальному генератору случайных чисел:

1. Последовательность должна состоять из квазиравномерно распределенных чисел.
2. Числа должны быть независимыми.
3. Последовательности случайных чисел должны быть воспроизводимыми.
4. Последовательности должны иметь неповторяющиеся числа.
5. Последовательности должны получаться с минимальными затратами вычислительных ресурсов.

Наибольшее применение в практике моделирования на ЭВМ для генерации последовательностей псевдослучайных числе находят алгоритмы вида:

$$x_{i+1}=\Phi(x_i),$$

представляющие собой реккурентные соотношения первого порядка.

Пример. $x_0 = 0{,}2152$, $(x_0)^2=0{,}04631104$, $x_1 = 0{,}6311$, $(x_1)^2=0{,}39828721$, $x_2=0{,}8287$ и т.д.

Недостаток подобных методов - наличие корреляции между числами последовательности, а иногда случайность вообще отсутствует, например:

$x_0 = 0{,}4500$, $(x_0)^2=0{,}20250000$, $x_1 = 0{,}2500$, $(x_1)^2=0{,}06250000$, $x_2=0{,}2500$ и т.д.

Широкое применение получили конгруэнтные процедуры генерации псевдослучайных последовательностей.

Два целых числа α и β конгруэнтны (сравнимы) по модулю m, где m - целое число, тогда и только тогда, когда существует такое целое число k, что $\alpha-\beta=km$.

$$1984\equiv4 \pmod{10},\ 5008\equiv8 \pmod{10^3}.$$

Большинство конгруэнтных процедур генерации случайных чисел основаны на следующей формуле:

$$x_{i+1} = \lambda x_i + \mu \,(mod\, M\,)$$

где λ, x_i, μ, M - неотрицательные целые числа.

По целым числам последовательности $\{X_i\}$ можно построить последовательность $\{x_i\}=\{X_i/M\}$ рациональных чисел из единичного интервала (0,1).

Применяемые генераторы случайных чисел перед моделированием должны пройти тщательное предварительное тестирование на равномерность, стохастичность и независимость получаемых последовательностей случайных чисел.[1,11,9]

Методы улучшения качества последовательностей случайных чисел:

1. Использование рекуррентных формул порядка r:

$$x_{i+1} = \Phi(x_i, x_{i-1}, ..., x_{i-r+1}).$$

Но применение этого способа приводит к увеличению затрат вычислительных ресурсов на получение чисел.

2. Метод возмущений:

$$x_{i+1} = \begin{cases} \Phi(x_i), \text{если } i \neq 0 (mod M) \\ \Psi(x_i), \text{если } i = 0 (\text{mod } M), \end{cases}$$

1.4.3. Моделирование случайных событий

1. Необходимо реализовать случайное событие А, наступающее с заданной вероятностью р. Определим А как событие, состоящее в том, что выбранное значение x_i равномерно распределенной на интервале (0,1) случайной величины удовлетворяет неравенству:

$$x_i =< p.$$

Тогда вероятность события А будет $P(A) = \int_0^p dx = p$. Противоположное событие состоит в том, что $x_i > p$, его вероятность равна 1-р.

2. Рассмотрим группу событий. Пусть $A_1, A_2, ..., A_s$ - полная группа событий, наступающих с вероятностями $p_1, p_2, ..., p_s$ соответственно. Определим событие A_m как событие, состоящее в том, что выбранное значение x_i случайной величины удовлетворяет неравенству

$$l_{m-1} < x_i \leq l_m,$$

где $l_r = \sum_{i=1}^{r} p_i$.

Процедура моделирования испытаний в этом случае состоит в последовательном сравнении случайных чисел x_i со значениями l_r. Если условие выполняется, исходом испытания оказывается событие A_m.

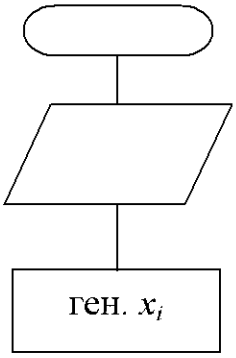
3. Рассмотрим независимые события А и В с вероятностями наступления p_A и p_B. Возможными исходами совместных испытаний в этом случае будут события АВ, $\overline{A}B, A\overline{B}, \overline{A}\overline{B}$ с вероятностями $p_A p_B$, $(1-p_A)p_B$, $p_A(1-$

p_B), $(1-p_A)(1-p_B)$. Для моделирования совместных испытаний можно использовать два варианта процедуры:

- Последовательное выполнение процедуры, рассмотренной в п.1.
- Определение одного из исходов AB, $\overline{A}B, A\overline{B}, \overline{A}\overline{B}$ по жребию с соответствующими вероятностями, т.е. процедура, рассмотренная в п.2.

Первый вариант потребует двух чисел x_i и двух сравнений. При втором варианте можно обойтись одним числом x_i , но сравнений может потребоваться больше. С точки зрения удобства построения моделирующего алгоритма и экономии количества операций и памяти ЭВМ более предпочтителен первый вариант.

4. События A и B являются зависимыми и наступают с вероятностями p_A и p_B . Обозначим через $p_A(B)$ условную вероятность наступления события B при условии, что событие A произошло. Алгоритм модели подобного случая может быть следующим [1,11]:



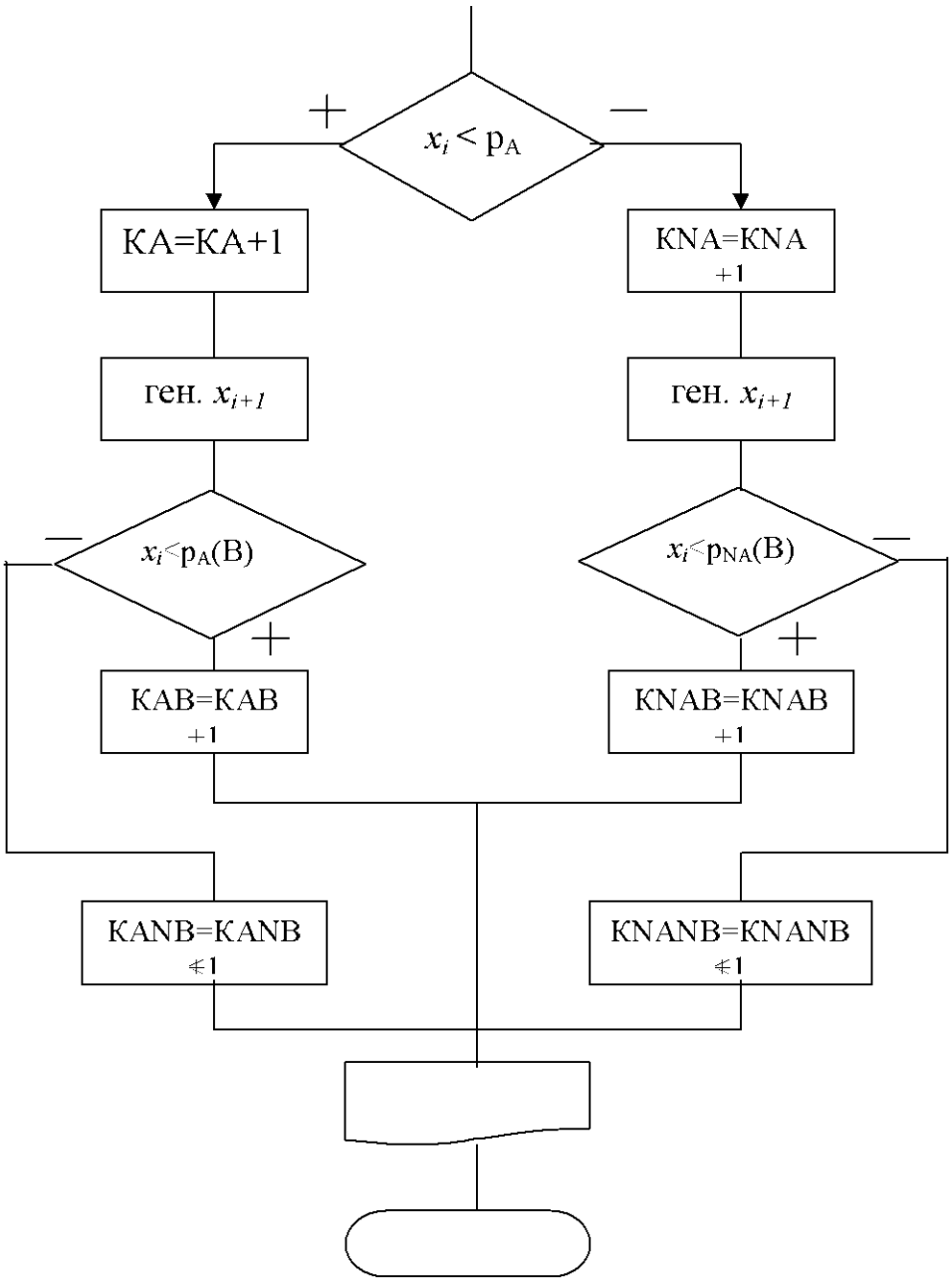


Рис. 1.15 Алгоритм генерации зависимых случайных событий

1.4.4. Моделирование дискретных случайных величин

Дискретная случайная величина Y принимает значения $y_1 <= y_2 <= \ldots <= y_j <= \ldots$ с вероятностями $p_1, p_2, \ldots, p_j, \ldots$ составляющими дифференциальное распределение вероятностей.

Таблица 1.1

Y	y_1	y_2	…	y_j	…
P	p_1	p_2	…	p_j	…

При этом интегральная функция распределения

$$F=\sum_{j=1}^{m} p_j;\ y_m=<y=<y_{m+1};\ m=1,2,\ldots;$$

$$F=0;\ y<y_1$$

Для получения дискретных случайных величин можно воспользоваться методом обратных функций: если X – равномерно распределенная на интервале (0,1) случайная величина, то искомую случайную величину получают при выполнении следующих действий:

Если $x_1<p_1$, то Y= y_1, иначе,

Если $x_1<p_1+p_2$, то Y= y_2, иначе,

……

Если $x_1<\sum_{j=1}^{m} p_j$, то Y= y_m, иначе,

……

Предположим, что известны частоты выбора из *N* объектов на определенном интервале времени, i=*1,..., N*. Пример таких частот для N=7 представлен в табл. 1.2. Первая строка таблицы - это номер объекта, а вторая - частота его выбора. Требуется разработать программную функцию, которая должна возвращать значение номера объекта в соответствии с заданным законом распределения.[3,9,11]

Таблица 1.2

Значения обратных функций для получения дискретного распределения

I	1	2	3	4	5	6	7
x_i	150	43	6	25	87	3	56

β_i	0,406	0,116	0,016	0,068	0,235	0,008	0,151
γ_i	0,406	0,522	0,538	0,606	0,841	0,849	1,000

Воспользуемся методом обратных функций. Сначала найдем сумму всех частот – получим S=370. После этого построим таблицу нормированных значений $\beta_i=x_i/S$ (третья строка табл. 1.2). Далее рассчитаем значения дискретной функции по формуле: $\gamma_r = \sum_{i=1}^{r} \beta_i$.

Полученные значения находятся в четвертой строке табл. 4.1.. Далее воспользуемся программой получения случайных величин, распределенных равномерно на отрезке (0,1), и каждый раз будем получать случайную величину p_t, После этого выбор объекта с номером i осуществляется при выполнении соотношения

$$\gamma_{i-1} < p_t <= \gamma_i$$

1.4.5. Моделирование непрерывной случайной величины

Для получения непрерывных случайных величин с заданным законом распределения, как и для дискретных величин, можно воспользоваться методом обратной функции. Если случайная величина Y имеет плотность распределения f(y), то распределение случайной величины

$$F(y) = \int_0^y f(y)dy$$

является равномерным на интервале (0,1). Чтобы получить число, принадлежащее последовательности случайных чисел $\{y_i\}$, имеющих функцию плотности f(y), необходимо разрешить относительно y_i уравнение

$$x_i = \int_0^y f(y)dy$$

где x_i - число, принадлежащее последовательности случайных чисел равномерно распределенных на интервале от (0,1).

Пример. Необходимо получить случайные числа с показательным законом распределения (например, интервалов времени между поступлениями заявок на обслуживание):

$$f(t) = \lambda e^{-\lambda t}.$$

$$x_i = \int_0^t \lambda e^{-\lambda t} dt$$

$$x_i = 1 - e^{-\lambda t}$$

$$ln(1 - x_i) = ln\, e^{-\lambda t}$$

$$t = -\frac{1}{\lambda} ln(1 - x_i)$$

$1 - x_i$ - случайное число, имеющее равномерное распределение на интервале (0,1). Тогда

$$t = -\frac{1}{\lambda} ln\, x_i$$

Этот способ получения случайных чисел с заданным законом распределения имеет ограниченную сферу применения, так как для многих законов распределения, встречающихся в практических задачах моделирования, интеграл не берется, т.е. приходится прибегать к численным методам решения, что увеличивает затраты вычислительных ресурсов на получение каждого числа; даже для случаев, когда интеграл берется в конечном виде получаются формулы, содержащие действия логарифмирования, извлечения корня и т.д., что также резко увеличивает затраты машинного времени на получение каждого случайного числа. Поэтому на практике часто пользуются приближенными способами преобразования случайных чисел, которые можно классифицировать следующим образом:

а) универсальные способы, с помощью которых можно получать случайные числа с законом распределения любого вида; б) неуниверсальные способы, пригодные для получения случайных чисел с конкретным законом распределения.

Рассмотрим приближенный универсальный способ получения случайных чисел, основанный на кусочной аппроксимации функции плотности.

Пусть требуется получить последовательность случайных чисел $\{y_j\}$ с функцией плотности $f_\eta(y)$, значения которой лежат в интервале (a,b). Разобъем интервал (a,b) на m интервалов (рис.1.16), и будем считать $f_\eta(y)$ на каждом интервале постоянной. Разбивать необходимо так, чтобы вероятность попадания случайной величины в любой интервал (a_k, a_{k+1}) была постоянной, т.е.:

$$\int_{a_k}^{a_{k+1}} f_\eta(y)dy = 1/m$$

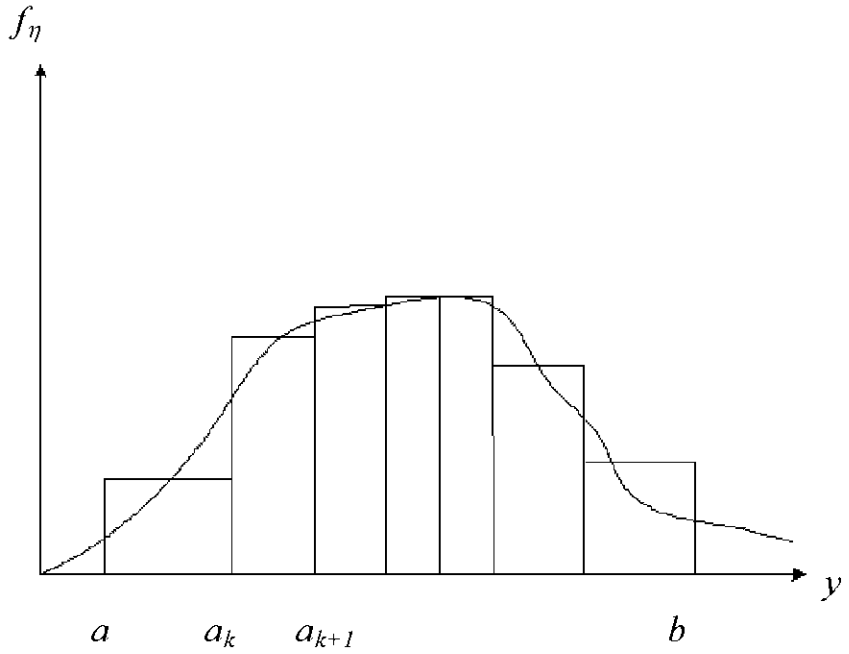


Рис. 1.16. Кусочная аппроксимация функции плотности

В таком случае, алгоритм этого способа получения случайных чисел сводится к выполнению следующих действий:

1) Генерируется случайное равномерно распределенной число x_i из интервала (0,1);

2) с помощью этого числа выбирается интервал (a_k, a_{k+1});

3) генерируется число x_{i+1};

4) вычисляется случайное число $y_j = a_k + x_{i+1}(a_{k+1} - a_k)$ с требуемым законом распределения.

Рассмотрим пример применения способа преобразования последовательности равномерно распределенных случайных чисел $\{x_i\}$ в последовательность с заданным законом распределения $\{y_i\}$ на основе предельных теорем теории вероятностей. Такие способы ориентированы на получение последовательностей чисел с конкретным законом распределения, т.е. не являются универсальными. [1,11]

Пример. Пусть требуется получить последовательность случайных чисел $\{y_i\}$, имеющих нормальное распределение с математическим ожиданием m и средним квадратическим отклонением σ:

$$p(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-m)^2}{2\sigma^2}}$$

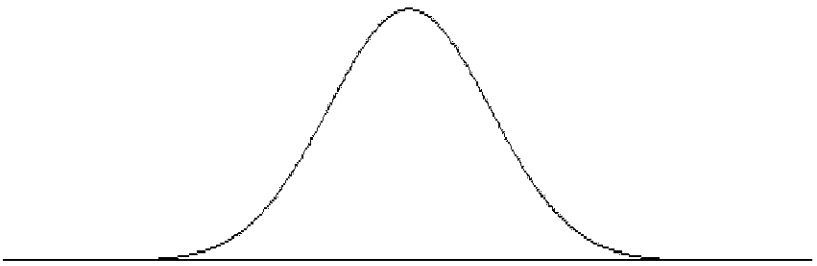
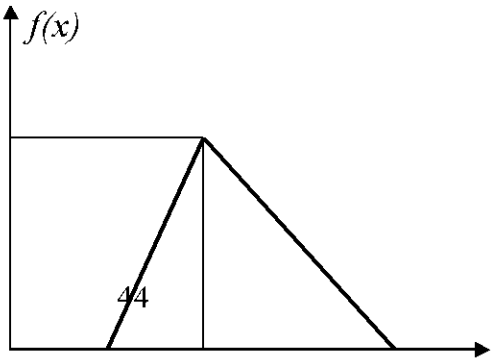


Рис. 1.17. Вид нормального распределения

Будем формировать случайные числа t_j в виде сумм последовательностей случайных чисел $\{x_i\}$, равномерно распределенных на интервале от (0,1). Можно воспользоваться центральной предельной теоремой: Если X_1, X_2,..., X_n - независимые одинаково распределенные случайные величины, имеющие математическое ожидание $M(X_i)=a$ и дисперсию σ^2, то при $N \to \infty$ сумма $\sum_{i=1}^{N} X_i$ асимптотически нормальна с математическим ожиданием Na и средним квадратическим отклонением $\sigma\sqrt{N}$. Практически достаточно N=8÷12, а в простейших случаях - 4÷5.

Преимущество этого способа - высокое быстродействие. Недостатком является игнорирование «хвостов» нормального распределения. Поэтому при проведении особо точных экспериментов применяются другие - более точные (но более медленные) способы. В современных системах имитационного моделирования обычно используются не менее двух программных датчиков случайных величин, распределенных по нормальному закону (их выбор осуществляется автоматически управляющей программой).[3,11]

Треугольное распределение применяется в тех случаях, когда о случайной величине ничего неизвестно, кроме наиболее вероятного значения и диапазона возможных значений этой случайной величины (рис. 1.18).[1,3,9,11]



f(C)

0 *A* *C* *B* *x*

Рис. 1.18. Общий случай треугольного распределения вероятностей

Функция для получения последовательностей случайных чисел, подчиненных треугольному распределению находится с помощью метода обратных преобразований:

$$y = \begin{cases} a + \sqrt{x(m-a)(b-a)}, \; x \le (m-a)/(b-a) \\ b - \sqrt{(1-x)(b-m)(b-a)}, \; x > (m-a)/(b-a) \end{cases},$$

где x – случайная величина, равномерно распределенная на интервале (0,1);

a – минимально возможное значение случайной величины;

b – максимально возможное значение случайной величины;

m – наиболее вероятное значение случайной величины.

1.5. Инструментальные средства моделирования

Универсальным инструментальным средством создания моделей являются языки программирования общего пользования (Pascal, C/C++ и др.). На основе этих языков в настоящее время бурное развитие получили средства визуального проектирования программ (Delphi, Visual C++), облегчающие выполнение некоторых трудоемких операций, например, создание интерфейса программы. Наряду с этим существует множество специализированных средств моделирования, позволяющих быстрее и с меньшими затратами (по сравнению с универсальными языками программирования) создавать и исследовать модели. В развитии специализированных средств можно выделить два направления:

1. Средства моделирования для анализа достаточно широкого класса систем. К ним относятся языки имитационного моделирования (GPSS, SIMSCRIPT и др.), пакеты прикладных программ, использующих для моделирования аналитические методы, такие как MathCad, MathLab, SAS и др. Основным недостатком этих средств является то, что их применение требует от исследователя специальной подготовки.

2. Программные комплексы, специализирующиеся на моделировании узкого круга систем одной конкретной предметной области. Недостаток, заключающийся в ограниченности применения таких программ одной предметной областью, с лихвой покрывается такими преимуществами, как легкость их освоения специалистами в данной предметной области, и эффективность применения, являющаяся следствием узкой специализации.

Подробнее рассмотрим специфику инструментальных средств имитационного моделирования. Как было ранее отмечено, существует два направления их развития. Первое из них представляют языки имитационного моделирования. Эти языки по сравнению с универсальными языками программирования снижают трудоемкость написания моделирующих программ, включают специализированные процедуры, которые могут применяться в любой имитационной модели, и отличаются точностью выражения понятий, характеризующих имитируемые процессы, и автоматическим формированием определенных типов данных, необходимых в процессе имитационного моделирования.

Таким образом, языки имитационного моделирования позволяют за сравнительно короткий срок составлять программные модели довольно сложных систем. К сожалению, такие модели обладают низкой способностью отвечать на вопросы типа "а что, если ...", поскольку это именно те вопросы, которые наиболее полезны, так как они способствуют более глубокому пониманию проблемы и поиску лучших способов оценки возможных действий. Для ответа на подобные вопросы часто приходится непосредственно изменять программный код модели, что повышает затраты времени на анализ системы. При использовании языков имитационного моделирования возникает также другая проблема: затраты на изучение и освоение языка, тем более, что эти языки оперируют абстрактными понятиями, в то время как экспериментатор часто является специалистом в той области, которой принадлежит моделируемая система, и применяет специфическую терминологию, что может значительно осложнить освоение языка исследователем.

В настоящее время языки имитационного моделирования получили дальнейшее развитие в виде визуальных средств моделирования, где исследователь оперирует не командами и операторами языка, а объектами, представляемыми в графическом виде (Arena, Scitor Process и др.). Визуальные средства моделирования частично снимают проблемы языков имитационного моделирования, описанные чуть выше, но в то же время ос-

новные из них остаются, например, освоение исследователем абстрактных терминов, используемых в этих средствах.

Вторым направлением развития инструментальных средств имитационного моделирования являются узкоспециализированные моделирующие программные комплексы. Преимуществами таких комплексов является быстрая реализация модели, использование терминологии, понятной исследователю, применяющему это средство, позволяют быстро отвечать на вопросы типа "а что, если ...".

2. Принципы разработки современных средств имитационного моделирования

2.1. Основные понятия и объекты системы моделирования

Основой любой моделирующей системы является язык имитационного моделирования, даже в системах визуального моделирования графические объекты функционально являются аналогами языка имитационного моделирования. Большинство языков имитационного моделирования имеют общую архитектуру, которую можно представить следующим образом:

1. Объекты моделирования описываются с помощью некоторых атрибутов языка;
2. Атрибуты взаимодействуют с процессами, адекватным реально протекающим явлениям в моделируемой системе;
3. Процессы требуют конкретных условий, определяющих логическую основу и последовательность взаимодействия этих процессов во времени;
4. Условия влияют на события, имеющие место внутри объекта моделирования и при взаимодействии с внешней средой;

События изменяют состояния модели системы в пространстве и времени.[11]

В большинстве случаев с помощью имитационных моделей исследуются характеристики и поведение системы на определенном отрезке времени, поэтому для моделирующей системы необходимым является реализация двух функций:

1) предоставление средств для формализованного описания системных компонентов, дисциплин выполнения различных работ, для задания структуры модели, привязки объектов модели к временной и пространственной координате;
2) осуществление координации событий, определение путей прохождения транзактов, изменение состояний узлов и передачу управления между блоками модели.

Существует несколько основных понятий, являющихся общими для большинства современных моделирующих систем:

1. Граф модели. Все процессы независимо от количества уровней структурного анализа, объединяются в виде направленного графа (многослойный иерархический).
2. Транзакт – это формальный запрос на какое-либо обслуживание. Транзакт в отличие от обычных заявок, которые рассматриваются при анализе моделей массового обслуживания, имеет набор динамически изменяющихся особых свойств и параметров. Пути

движения транзактов по графу модели определяются логикой функционирования компонентов модели.

Транзакт является динамической единицей любой модели, работающей под управлением имитатора.

Транзакт может выполнять следующие действия:

- Порождать группы (семейства) других транзактов;
- Поглощать другие транзакты конкретного семейства;
- Захватывать ресурсы и использовать их некоторое время, а затем освобождать;
- Определять времена обслуживания, накапливать информацию о пройденном пути и иметь информацию о своем дальнейшем пути и о путях других транзактов.

Основные параметры транзактов:

- Уникальный идентификатор транзакта;
- Идентификатор (номер) семейства, к которому принадлежит транзакт;
- Наборы различных ресурсов, которые транзакт может захватывать и использовать какое-то время;
- Время жизни транзакта;
- Приоритет – неотрицательное число;
- Параметры обслуживания в каком-либо обслуживающем устройстве (включая вероятностные характеристики).

Примеры транзактов:

- телекоммуникационный пакет;
- покупатель;
- заявка на товар;
- автомобиль;
- обрабатываемая деталь;
- работник и т.д.

3. Узлы графа модели представляют собой центры обслуживания транзактов. В узлах транзакты могут задерживаться, обслуживаться, порождать семейства новых транзактов, уничтожать другие транзакты. Транзакт всегда принадлежит одному из узлов графа модели и независимо от этого относится к определенной точке простарнства или местности, координаты которой могут изменяться.
4. Событие – факт входа из узла одного транзакта. Разработчик модели практически не может управлять событиями вручную.

Функция управления событиями отдана специально управляющей программе – координатору, автоматически внедряемому в состав модели.

5. Ресурс независимо от его природы в процессе моделирования может характеризоваться тремя общими параметрами: мощностью, остатком и дефицитом. Мощность – это макс. число ресурсных единиц. Остаток ресурса – число незанятых на данный момент единиц. Дефицит ресурса – количество единиц ресурса в суммарном запросе транзактов, стоящих в очереди к данному ресурсу.
6. Пространство – географическое, декартова плоскость. Узлы, транзакты и ресурсы могут быть привязаны к точкам пространства и мигрировать в нем.

Программная реализация моделирующих систем использует объектно-ориентированный способ представления процессов. Транзакты, узлы, события и ресурсы являются основными объектами имитационной модели.[9]

2.2. Подходы к разработке программных моделей

Существуют три альтернативных методологических подхода к построению имитационных моделей: событийный подход, процессно-ориентированный подход и подход сканирования активностей. В основе этих подходов лежат соответственно понятия события, действия и процесса.

2.2.1 Событийный подход

Событием называется последовательность логически связанных действий, происходящих в некоторый фиксированный момент модельного времени. Появление события связано с изменением состояния модельной среды. Примером события может служить генерация сообщений, передача сообщения по каналу связи, обработка сообщения узлом и т.п.

Событийный подход основан на формировании потока событий (рис. 2.1). Такой поток образует сгруппированные последовательности действий:

$$\{E_1, E_2, ..., E_k\}, \{E_{k+1}, E_{k+2}, ..., E_m\}, ..., \{E_n, E_{n+1}, ..., E_\upsilon\}...$$

Группы событий последовательно, происходящих в один и тот же момент времени, условно заключены в скобки. События $\{E_1, E_2, ..., E_k\}$ образуют первую группу, события $\{E_{k+1}, E_{k+2}, ..., E_m\}$ - вторую, а $\{E_n, E_{n+1}, ..., E_\upsilon\}$ - группу N. Обозначим $T_i, i = 1,2,...,N...$, модельное время осуществления событий тогда для последовательности времен справедливо упорядочение

$$T_1 < T_2 < ...T_N...$$

Если выполняемые внутри группы действия, связанные с одиночным событием, назвать процессом, то можно сказать, что часть процессов, определенных в группе, выполняются квазипараллельно. Соответствующее этим определениям понятие события было раскрыто выше. Собственно события являются элементами, с которыми оперирует управляющий алгоритм системы моделирования. С одной стороны, появление событий связано с изменениями в модельной среде. С другой, исполнение действий, связанных с событием, приводит к изменениям модельной среды. Действия, связанные с каждым событием, должны быть описаны в модели, а также должны быть составлены условия возникновения события.

Как правило, условия описываются отношениями связанных с событием переменных. В результате выполнения события значения некоторых модельных переменных могут измениться; это приводит к тому, что возникают условия, необходимые для выполнения другого события. При выполнении последовательности действий, связанной с выполнением некоторого события, можно изменить условия, связанные с его последующим вызовом. Таким образом, в результате выполнения потока событий изменяется среда моделирования; это приводит к возникновению новых условий, новые условия приводят к выполнению новых событий. Моделирование развивается по такой схеме сколь угодно долго.[5,9]

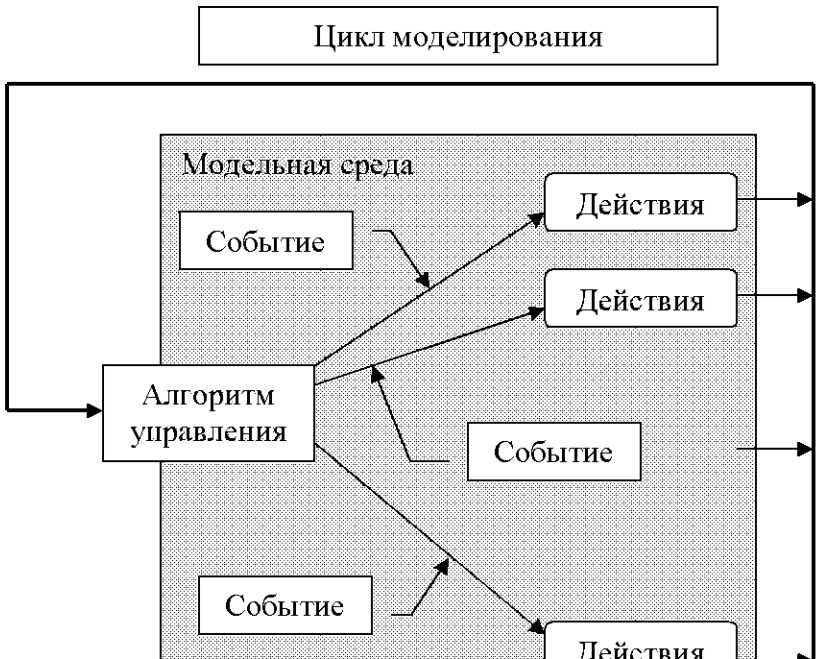


Рис. 2.1. Событийный подход

Событийный подход лежит в основе многих систем моделирования, например, GASP, SIMSCRIPT и т.д.

Событийные модели обладают большей гибкостью, но они уступают процессно-ориентированным системам в простоте и наглядности составления моделей.

2.2.2. Процессно-ориентированный подход

Языки, реализующие процессно-ориентированный подход, имеют в своем составе блоки или операторы, позволяющие описать процесс продвижения компонентов через систему (рис. 2.2). В моделях, использующих подобную схему, описываются не события и условия их возникновения, а процесс, развивающийся в ней.

Для этого используются средства языка системы моделирования. Естественно, эти средства должны быть достаточно представительными для того, чтобы описывать широкий круг процессов. В частности, для процессного подхода необходимыми являются средства явного (или неявного) определения участков модели, на которых необходимо представление квазипараллельных процессов. К достоинствам процессно-ориентированного представления моделей следует отнести компактность и наглядность. Процессно-ориентированный подход реализован в системе моделирования GPSS. [5,9]

Рис. 2.2. Графическое представление примера процесса продвижения транзактов

2.2.3. Подход сканирования активностей

При использовании подхода сканирования активностей разработчик описывает действия, в которых принимают участие элементы системы, и задает условия, определяющие начало и окончание этих действий. События, которые начинают или завершают действие, не планируются разработчиком модели, а инициируются по условиям, определенным для данного действия. Условия начала или окончания действия проверяются после очередного продвижения имитационного времени. Если заданные условия удовлетворяются, происходит соответствующее действие. Для того, чтобы было выполнено каждое действие в модели, сканирование условий производится для всего множества действий при каждом продвижении имитационного времени.

Подход сканирования активностей эффективен для ситуаций, в которых продолжительность действия определяется в зависимости от того, насколько состояние системы удовлетворяет заданным условиям. Тем не менее, так как необходимо сканировать условия для каждого действия, под-

ход сканирования активностей менее эффективен по сравнению с событийным подходом. [9]

В большинстве современных систем моделирования (Arena, Scitor Process и т.п.) используется обычно комбинация перечисленных методологических подходов к построению моделей.

2.3. Управление модельным временем

Одной из наиболее важных задач при создании модели является реализация механизма корректировки временной координаты состояния системы ("продвижение" времени, организация "часов") и обеспечение согласованности различных блоков и событий в системе (синхронизация во времени, координация с другими блоками).

Таким образом, функционирование модели должно протекать в искусственном (не в реальном и не в машинном) времени, обеспечивая появление событий в требуемом логикой работы исследуемой системы порядке и с надлежащими временными интервалами между ними. При этом надо учитывать, что элементы реальной системы функционируют одновременно (параллельно), а компоненты программной модели действуют последовательно, так как реализуются с помощью ЭВМ последовательного действия. Поскольку в различных частях объекта моделирования события могут возникать одновременно, то для сохранения адекватности причинно-следственных временных связей необходимо в языке имитационного моделирования создать «механизм» задания времени для синхронизации действий элементов модели системы.[3,5,11]

Существует два основных подхода к реализации механизма модельного времени.

Рассмотрим первый подход, называемый *принципом* Δt. Организуем счетчик системного времени, который в начальный момент показывает время t_0. Прибавим интервал времени Δt, тогда счетчик будет показывать $t_1 = t_0 + \Delta t$. Определим вектор состояний системы $Z(t_0 + \Delta t)$, затем перейдем к моменту времени $t_2 = t_1 + \Delta t$ и т.д. Если шаг Δt достаточно мал, то таким путем можно получать приближенные значения Z. Этот принцип позволяет определить последовательные состояния процесса функционирования системы через заданные интервалы времени Δt. Но с точки зрения затрат машинного времени он иногда оказывается неэкономичным.

Второй подход называется принципом особых состояний. При рас-

смотрении процессов функционирования некоторых систем можно обнаружить, что для них характерны два типа состояний: 1) особые, присущие процессу функционирования системы только в некоторые моменты времени (моменты поступления входных или управляющих воздействий, возмущений внешней среды и т. п.); 2) неособые, в которых процесс находится все остальное время. Особые состояния характерны еще и тем обстоятельством, что функции состояний Z(t) в эти моменты времени изменяются скачком, а между особыми состояниями изменение координат Z(t) происходит плавно и непрерывно или не происходит совсем. Таким образом, следя при моделировании системы только за ее особыми состояниями в те моменты времени, когда эти состояния имеют место, можно получить информацию, необходимую для построения функций Z(t). Обозначим скачкообразное (релейное) изменение состояния z как δz, а «принцип особых состояний» — как *принцип* δz.

Например, для системы массового обслуживания (Q-схемы) в качестве особых состояний могут быть выбраны состояния в моменты поступления заявок на обслуживание в прибор *П* и в моменты окончания обслуживания заявок каналами *К*, когда состояние системы, оцениваемое числом находящихся в ней заявок, меняется скачком.

Отметим, что характеристики процесса функционирования таких систем с особыми состояниями оцениваются по информации об особых состояниях, а неособые состояния при моделировании не рассматриваются. «Принцип δz» дает возможность для ряда систем существенно уменьшить затраты машинного времени на реализацию моделирующих алгоритмов по сравнению с «принципом Δt». Логика построения моделирующего алгоритма, реализующего «принцип δz», отличается от рассмотренной для «принципа Δt» только тем, что включает в себя процедуру определения момента времени t_δ, соответствующего следующему особому состоянию системы. Для исследования процесса функционирования больших систем рационально использование комбинированного принципа построения моделирующих алгоритмов, сочетающего в себе преимущества каждого из рассмотренных принципов.

Рассмотрим соответствующие способы управления временем в модели системы на примере, показанном на рис. 2.3, где по оси реального времени отложена последовательность событий в системе $\{s_i\}$ во времени (рис. 2.3, а). Под действием событий s_i, изменяются состояния модели z_i, в момент времени t_{Si}, причем такое изменение происходит скачком δz.

В модели, построенной по принципу Δt (рис. 2.3, *б),* моменты системного времени будут последовательно принимать значения $t'_1=\Delta t$, $t'_2=2\Delta t$, $t'_3=3\Delta t$. Эти моменты системного времени $t'_j(\Delta t)$ никак не связаны с моментами появления событий, которые имитируются в модели системы. Системное время при этом получает постоянное приращение, выбираемое и задаваемое перед началом имитационного эксперимента.

В модели, построенной по «принципу δz» (рис. 2.3, в), изменение времени наступает в момент смены состояния системы, и последовательность моментов системного времени имеет вид $t''_1=t_{S1}$, $t''_2= t_{S2}$, $t''_3= t_{S3}$, $t''_4= t_{S4}$, $t''_5= t_{S5}$, т.е. моменты системного времени $t''_k(\delta z)$ непосредственно связаны с моментами появления событий в системе s_i.

У каждого из этих методов есть свои преимущества с точки зрения адекватного отражения реальных событий в системе и затрат машинных ресурсов на моделирование. При использовании «принципа δz» события обрабатываются последовательно и время смещается каждый раз вперед до начала следующего события. В модели, построенной по «принципу Δt», обработка событий происходит по группам, пакетам или множествам событий. При этом выбор Δt оказывает существенное влияние на ход процесса и результаты моделирования, и если Δt задана неправильно, то результаты могут получиться недостоверными, так как все события появляются в точке, соответствующей верхней границе каждого интервала моделирования. При применении «принципа δz» одновременная обработка событий в модели имеет место только тогда, когда эти события появляются одновременно и в реальной системе. Это позволяет избежать необходимости искусственного введения ранжирования событий при их обработке в конце интервала Δt.

При моделировании по «принципу Δt» можно добиться хорошей аппроксимации: для этого Δt должно быть малым, чтобы два неодновременных события не попали в один и тот же временной интервал. Но уменьшение Δt приводит к увеличению затрат машинного времени на моделирование, так как значительная часть тратится на корректировку «часов» и отслеживание событий, которых в большинстве интервалов может и не быть. При этом даже при сильном «сжатии» Δt два неодновременных события могут попасть в один и тот же временной интервал Δt, что создает ложное представление об их одновременности.

Современные моделирующие системы в основном для реализации

механизма управления модельным временем используют принцип особых состояний. Принцип Δt в основном используется при создании небольших имитационных моделей с помощью универсальных языков программирования. [3,5,11]

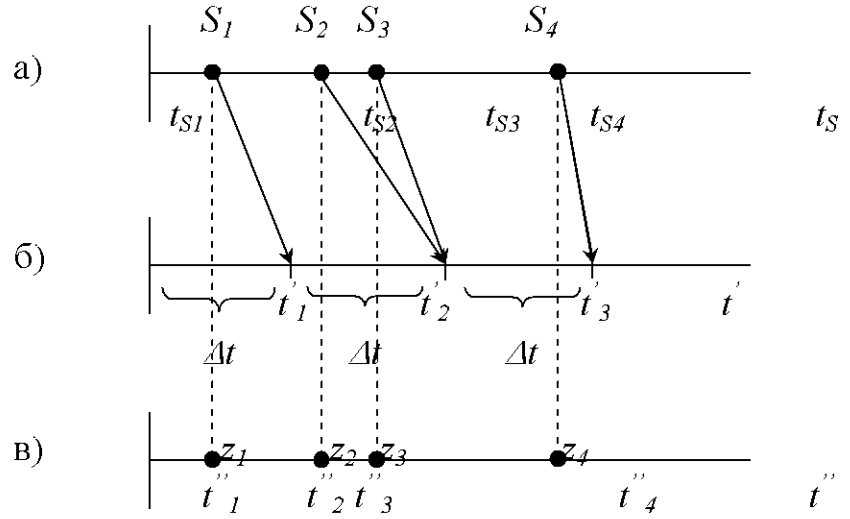


Рис. 2.3. Способы управления временем в модели системы

2.4. Механизм управления процессом моделирования

Рассмотрим общие принципы управления процессами для большинства систем моделирования. При моделировании реальных систем необходимо имитировать параллельные процессы. Под параллельностью процессов понимают, что одновременно могут происходить множество событий. При этом интерпретатор моделирующей системы выполняет команды последовательно. Таким образом, необходим программный механизм для виртуального представления параллельных процессов в моделирующей системе. Смысл этого механизма заключается в том, что пока не обработаны все события, которые должны произойти в данный момент времени, модельное время не меняется. Под обработкой события понимают изменение состояния элементов модели, вследствие наступления данного события и генерация следующих событий, как независимых, так и вызванных последним событием, учет статистики и т.п..

Обычно в моделирующих системах выделяют два основных списка событий: список текущих событий (СТС) и список будущих событий (СБС). Каждое событие ассоциируется с динамическим объектом - транзактом. В список текущих событий входят все события, запланированные на текущий момент модельного времени. Программа управления модели-

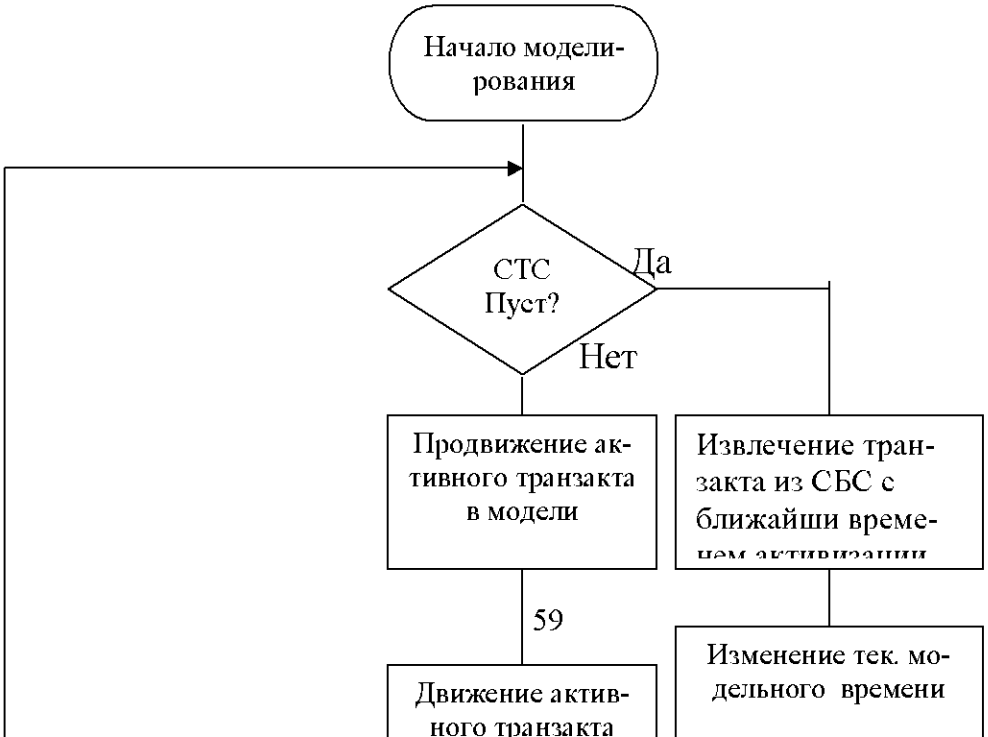
рованием просматривает в первую очередь этот список и пытается переместить по модели те транзакты, для которых выполнены условия. Если в этом списке таких транзактов нет, процесс управления моделированием обращается к другому списку - списку будущих событий. Она переносит все события, которые запланированы на ближайший момент модельного времени, из этого списка в список текущих событий и повторяет его просмотр.

При моделировании модельное время может меняться быстрее или медленнее, чем в реальной системе. Это зависит от степени детализации модели и сложности описания изучаемого процесса. В любом случае модельное время изменяется при выполнении некоторых событий, а события в системе моделирования возникают в результате перемещения транзактов. Причиной изменения модельного времени может послужить явная задержка транзакта на некоторый отрезок модельного времени.

Рассмотрим транзакт, проходящий некоторую цепочку событий. Речь идет об активном транзакте, таким может быть только один транзакт в модели, а именно тот, который продвигается в настоящий момент времени по модели. Активный транзакт может перемещаться от события к событию, вызывая выполнение связанных с событиями действий (рис. 2.4). Значение модельного времени изменяться не будет до тех пор, пока транзакт остается активным. Прекратить активность транзакта может некоторое событие. Таким событием, например, может быть явная задержка транзакта. В этом случае активный транзакт помещается в список будущих событий, где он занимает место среди транзактов, ожидающих своего времени продвижения. Другим событием, прекращающим активность транзакта, может быть попытка поступить в занятый прибор. В этом случае транзакт должен быть помещен в список ожидающих освобождения прибора. Некоторые другие события могут также прекращать активность транзакта. После того, как такое событие произошло, активным должен стать другой транзакт.

Если в системе имеются другие транзакты, способные к продвижению в данный момент времени, то они находятся в списке текущих событий; если этот список не пуст, то из него извлекается первый находящийся в нем транзакт и он становится активным, после чего продвигается по модели. Наконец наступит такой момент, что ни одного транзакта в списке текущих событий не останется; тогда и просматривается список будущих событий. Из него выбирается транзакт с минимальным "будущим време-

нем продвижения", модельное время устанавливается равным времени продвижения этого транзакта, а сам транзакт перемещается в список текущих событий. Вместе с ним в список текущих событий перемещаются все транзакты с таким же временем продвижения. После этого первый транзакт из списка текущих событий становится активным, и процесс моделирования продолжается. [5,12]



Нет Да

Рис. 2.4. Упрощенная схема протяжки модельного времени

2.5. Обработка и анализ результатов имитационного моделирования

В имитационных исследованиях необходимо на основе результатов имитационных экспериментов делать некоторые выводы или прогнозы относительно поведения моделируемой системы. Поскольку имитационная модель содержит вероятностные элементы, результаты имитации являются наблюдениями случайных величин. Как следствие, любая интерпретация поведения системы на основе анализа полученных результатов имитации

должна учитывать их разброс. Таким образом, для анализа выходных данных используют методы математической статистики.

Относительно выходных данных имитационной модели можно сформулировать два типа вопросов:

1. Насколько адекватна построенная имитационная модель и каков разброс данных, обусловленных имитационной моделью.

2. Какие выводы можно сделать о характеристиках реальной системы на основе использования имитационной модели?

Первый вопрос связан с выяснением того, работает ли она в соответствии с задуманной схемой. Необходимо также выявление чувствительности выходных данных модели к изменению ее входных параметров. Разброс выходных данных зависит прежде всего от точности подбора заложенных в модель вероятностных распределений.

Второй вопрос связан с конечной целью построения имитационной модели. Ответ на него включает описание характеристик системы и их статистическую обработку.

Искомыми величинами обычно являются следующие оценки:

- вероятность наступления некоторого события;
- математическое ожидание случайной величины;
- дисперсия случайной величины;
- коэффициенты ковариации или корреляции случайной величины.

В большинстве систем моделирования существуют встроенные функции обработки и анализа результатов имитационных экспериментов. [9,11,12]

2.6. Технология визуального проектирования имитационных моделей

В последние годы все большее распространение получили визуальные средства имитационного моделирования (Arena, Scitor Process и т.д.) позволяющие автоматизировать процессы проектирования и анализа моделей. Подобные средства активно используют методологию структурного анализа, предусматривающую наглядное и эффективное проектирование

системы путем выделения ее составляющих и их последовательного рассмотрения. Описание системы начинается с общего обзора и выделения основных ее компонентов или процессов. Для визуального представления создается первый уровень или слой, на котором отображаются выделенные процессы и их взаимосвязи. Далее для ряда процессов может быть проведена детализация, в свою очередь выделяющая новые процессы в их структуре. Так, последовательным усложнением описания объекта и его процессов разработчик достигает необходимой детализации. Глубина детализации определяется как необходимой точностью, так и набором исходных данных. В процессе структурного анализа выявляется иерархическая структура модели. Пакеты визуального проектирования имитационных моделей предоставляют пользователю инструментарий работы с проектом, так называемый графический конструктор, опирающийся на мощные современные графические средства отображения информации в виде графов, диаграмм, схем и таблиц. Если раньше для создания имитационной модели разработчику приходилось писать программный код, использующий языковые средства системы моделирования, то сейчас визуальные средства моделирования позволяют автоматизировать процесс создания структуры модели и автоматически генерируют код моделирующей программы.

Таким образом, визуальные средства моделирования разработчику имитационных моделей предоставляют следующие преимущества:

- автоматическая генерация программного кода позволяет пользователю не задумываться о структуре и синтаксисе программы, уделяя все внимание структуре и параметрам самой модели и ее узлов;
- генерация функций описания узлов графическим конструктором исключает ошибки, связанные с неправильной последовательностью указания позиционных параметров или пропуском некоторых из них;
- анализируя модель, конструктор не позволяет пользователю выполнять заведомо неверные действия, а также предупреждает о возможных ошибках;
- возможность создания иерархических моделей, что может быть очень удобно при выполнении моделей с большим количеством узлов.[3,12]

Еще одним важным преимуществом подобных систем является наличие в них специальных интерфейсов для интеграции с другими программными продуктами, используемыми в процессе разработки имитационной модели. Например, в системе имитационного моделирования Arena существует специальный интерфейс для интеграции с системой функционального моделирования BPWin. Это дает возможность автоматизировать преобразование функциональных моделей, построенных по технологии SADT (стандарт IDEF), в имитационную модель в системе Arena. Также существуют интерфейсы с базами данных (например, ODBC - Open Data Base Connectivity), что позволяет получать входные данные для имитационной модели в структурированном виде из баз данных систем сбора и учета информации или передавать результаты моделирования в другие программы для дальнейшего анализа.

В следующих главах приведены пример разработки специализированного моделирующего программного комплекса для анализа вычислительных сетей и краткие описания систем моделирования GPSS World и Arena.

2.7. Пример разработки специализированного моделирующего комплекса для анализа вычислительных сетей

В настоящее время стремительно растет объем информационного обмена, вследствие этого в современных сетях часто возникает необходимость прогнозирования последствий изменений в сети, смены топологии, смены сетевой технологии и т.д. Последствия могут оцениваться с точки зрения влияния на производительность, время ответа сети, доступность тех или иных ресурсов. При проектировании сетей приходится решать такие вопросы, как, например, выбор структуры связи между узлами сети, характеристик каналов связи и т.д. Важно исследовать, как изменяются основные показатели эффективности функционирования сети в зависимости от нагрузки на нее, и определить "узкие" места в сети, т. е. узлы, в которых образуются относительно длинные очереди пакетов. Поэтому необходимо при создании новых сетей предварительно оценивать эффективность их функционирования с учетом различных вариантов структурной организации, это дает возможность на стадии проектирования сети выбрать наиболее подходящий из альтернативных вариантов решения и доказать заказ-

чику экономическую и техническую целесообразность выбранного решения.

Одним из способов прогнозирования рабочих параметров будущей сети является имитационное моделирование информационных потоков.

2.7.1. Предпосылки построения модели

Во-первых, все корпоративные сети обладают общим принципом взаимодействия между узлами сети (рис. 2.5): логически можно выделить передающее устройство, генерирующее поток сообщений; среда передачи (канал связи), по которому передается этот поток, принимающее устройство, обрабатывающее этот поток сообщений. Причем, обычно передающее и принимающее устройства конструктивно объединены в одном узле, т. е. узел может как передавать, так и принимать информацию, но в любой момент времени можно логически выделить данную схему работы сети. Таким образом, если между двумя узлами имеем симплексный канал связи, допускающий передачу информации только в одном направлении, то схема передачи остается неизменной; если между двумя узлами полудуплексная связь, которая допускает передачу информации в обоих направлениях, но в разные моменты времени, то данная схема в разные моменты времени будет менять только направление передачи в противоположную сторону; в случае же дуплексной связи (одновременная передача информации в обоих направлениях) можно выделить две такие логические схемы, работающие параллельно и имеющие противоположные направления передачи.

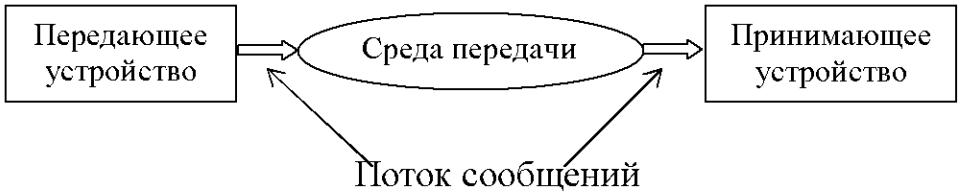


Рис. 2.5. Общая схема взаимодействия между элементами корпоративной сети

Во-вторых, корпоративные сети состоят из устройств одного и того же типа: рабочие станции, серверы, сетевые принтеры, концентраторы (hub), повторители (repeater), мосты, коммутаторы (switch), маршрутизаторы (router), каналы связи (коаксиальный кабель, витая пара, оптоволоконный кабель, также могут использоваться беспроводные системы передачи на основе радиосигналов, лазерных и микроволновых технологий). В-третьих, основной задачей для всех корпоративных сетей является передача информации между узлами сети. В-четвертых, любую корпоративную сеть можно разбить на элементарные подсистемы, описанные по единой логической схеме (рис. 2.6).

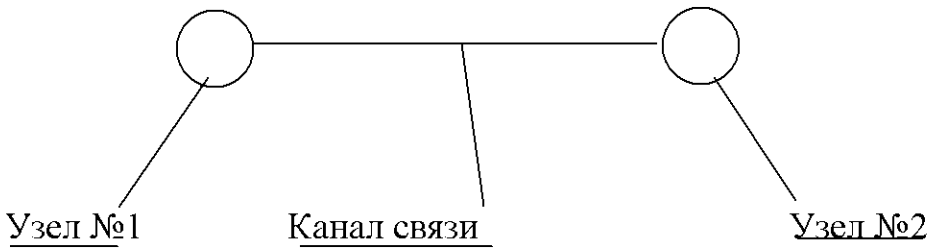


Рис. 2.6. Общий вид элементарной подсистемы корпоративной сети

Каждый узел может входить одновременно в несколько таких элементарных подсистем; число подсистем, в которые входит данный узел равно числу узлов, с которыми этот узел напрямую связан каналами связи. Канал связи также может входить одновременно в несколько элементарных подсистем, это зависит от числа узлов подключенных к этому каналу связи, т.е. число элементарных подсистем, в которые входит данный канал связи равно:

$$C_n^2 = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2},$$

где n - это количество узлов связанных непосредственно этим каналом связи. Таким образом, общее количество элементарных подсистем равно количеству пар узлов, связанных между собой напрямую каналами связи. В любой момент времени работу корпоративной сети можно представить как совокупность работающих элементарных подсистем, т.е. тех подсистем, в которых в данный момент происходит передача данных между узлами. Следовательно, составив общую модель элементарных подсистем, можно представить работу корпоративной сети в каждый момент времени как работу одной или нескольких, работающих параллельно, таких моделей, отличающихся друг от друга только параметрами узлов и каналов связи, уча-

ствующих в передаче. Если узлы и каналы связи корпоративной сети определенным образом проиндексировать, то параметры общей модели можно будет изменять с помощью системы индексов в зависимости от активности той или иной элементарной подсистемы, т.е. от наличия факта передачи в этой подсистеме в данный момент времени.

2.7.2. Обобщенная схема вычислительной сети

Логически любую корпоративную сеть можно представить в виде следующей обобщенной схемы:

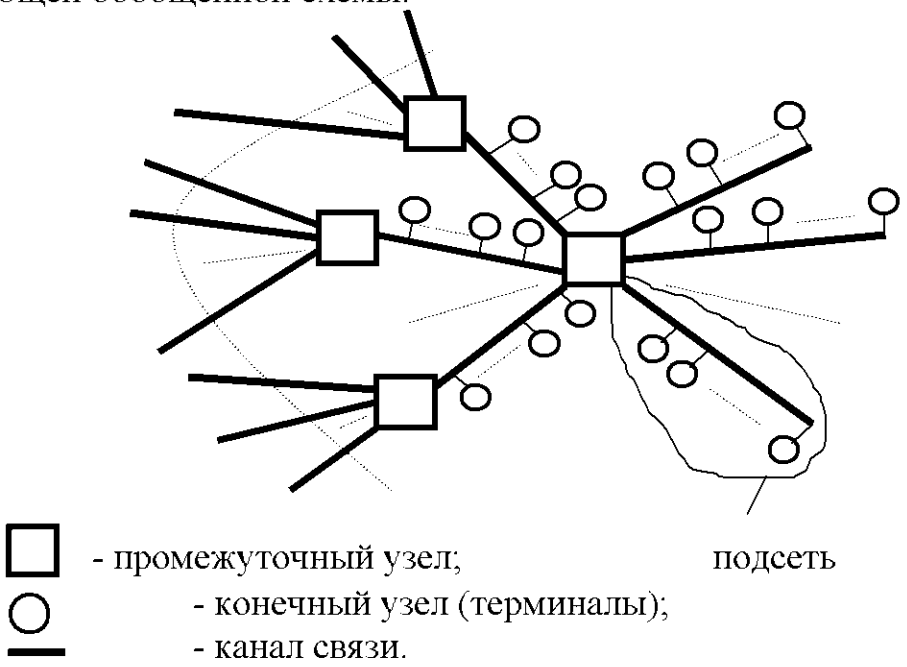


Рис. 2.7. Обобщенная схема корпоративной сети

2.7.3 Упрощения, принятые при построении модели

Ранее было указано, что процесс последовательной разработки имитационной модели начинается с создания простой модели, которая затем постепенно усложняется в соответствии с предъявляемыми решаемой проблемой требованиями. Поэтому сделаем некоторые упрощения, т.е. избавимся от избыточных компонентов, которые незначительно влияют на общую схему функционирования корпоративной сети, либо в настоящий момент не столь актуальны (в дальнейшем эти детали при необходимости можно ввести в модель).

1. Объединим логически такие сетевые устройства, как повторитель и концентратор, с каналом связи. Эти устройства используются для объединения сегментов сети (концентратор также используется для создания топологии "пассивная звезда) и имеют скорее конструктивное назначение; они не влияют на логику передачи данных и вносят лишь незначительные временные задержки при передаче информации.

2. Следующее упрощение состоит в предположении, что в нашей абстрактной корпоративной сети отсутствуют маршрутизаторы. Действительно, в большинстве реальных корпоративных сетей маршрутизаторы выполняют не задачу связи удаленных административных офисов, а задачу интеграции данной корпоративной сети с глобальной сетью Internet. Поэтому для моделирования работы этой корпоративной сети необходимо ее "замкнуть", т.е. представить маршрутизаторы в виде конечных узлов типа рабочей станции или сервера с соответствующими производительностью и объемами передаваемого и принимаемого трафика (в данном случае опускается возможность виртуальных корпоративных сетей).

3. Не будем учитывать задержки, связанные с искажением передаваемой информации и повторной передачей данных.

4. Из рассмотрения исключим возможность создания виртуальных частных сетей внутри корпоративной сети.

После сделанных упрощений любую корпоративную сеть можно представить сетью, состоящую из трех типов элементов: промежуточные узлы (коммутаторы, мосты; серверы, выполняющие функции моста), конечные узлы (рабочие станции, серверы, сетевые принтеры) и каналы связи (рис. 2.7). Промежуточные узлы служат для объединения каналов связи, конечные узлы подключаются к каналам связи.

2.7.4. Понятие подсети

Введем понятие "подсеть". Подсеть включает канал связи и все узлы, подключенные к этому каналу связи (рис. 2.7), причем, в подсети в любой момент времени может быть активна только одна элементарная подсистема, т.е. в один момент времени передача данных может осуществляться только между двумя узлами и только в одном направлении. В случае дуплексного канала связи его необходимо логически разбить на два противоположных симплексных канала, а подсеть соответственно на две подсети. Между собой подсети взаимодействуют через промежуточные узлы, т.е.

промежуточные узлы входят в несколько подсетей. От количества подсетей зависит максимальное число параллельно работающих процессов передачи.

Важным свойством подсети является то, что она функционирует на основе одной сетевой технологии (Ethernet, Token Ring, ATM, Frame Relay, X.25, ISDN и др.). При моделировании необходимо учитывать особенности сетевых технологий, такие, как максимальная пропускная способность, обеспечиваемая данной технологией, минимальный и максимальный размер передаваемого сообщения, метод доступа к передающей среде (например, метод обнаружения коллизий в Ethernet, метод маркера в Token Ring и т.п.) и др. Для этого необходимо создать библиотеку процедур, моделирующих особенности широко известных сетевых технологий в рамках общей модели. В этом случае исследователю необходимо только ввести вид технологии, на основе которой функционирует данная подсеть. При "прогоне" программной модели будут вызываться из этой библиотеки процедуры, соответствующие типу технологии, используемой в подсети, в которой происходит передача информации в данный момент времени.

2.7.5. Входные данные модели

На основе приведенных выше рассуждений определим необходимые входные данные для имитации суточной работы конкретной корпоративной сети с помощью общего моделирующего алгоритма:

Количество узлов, их производительности, тип каждого узла. Тип узла имеет значение при расчете различных технологических пауз при передаче, например, в сети Ethernet при возникновении коллизии коммутатор и рабочая станция делают различные по времени паузы.

Количество подсетей; для каждой подсети должен указываться тип сетевой технологии, на основе которой она функционирует.

Среднесуточный объем трафика между каждой парой узлов. Следовательно, для каждого узла должно указываться, какой объем информации он в среднем за сутки передает каждому узлу, входящему в сеть. Для коммутаторов и мостов эти данные должны быть равны нулю (если не учитывать служебный трафик), т.к. они являются при передаче лишь транзитными узлами.

Структура сети, для каждой подсети указываются номера входящих в нее узлов (промежуточные узлы входят в несколько подсетей).

Почасовая нагрузка всей сети.

2.7.6. Алгоритм движения сообщения

При создании моделирующего алгоритма необходимо обратить внимание на следующие моменты:

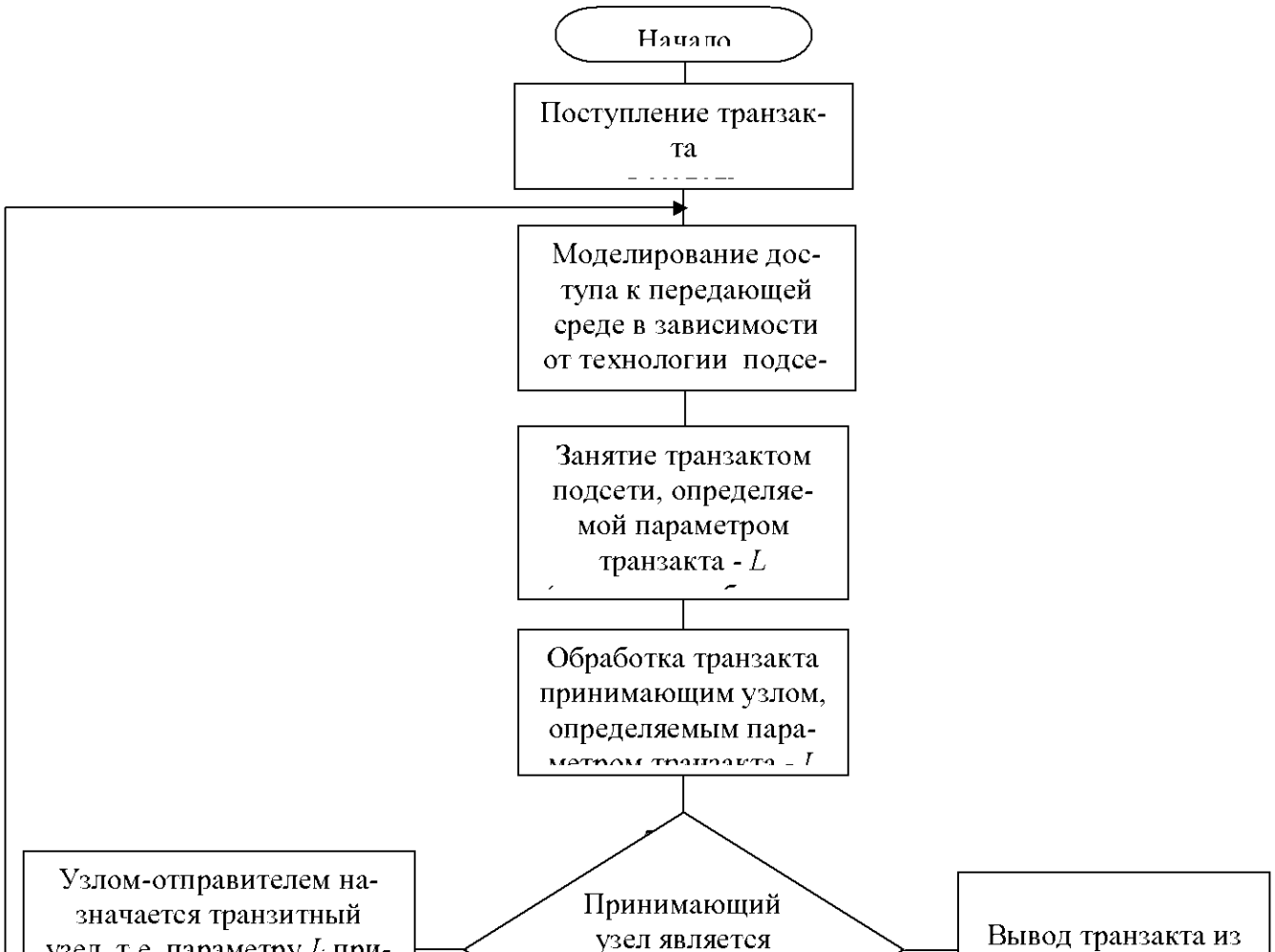
В сеансе связи между двумя узлами, находящихся в разных подсетях, могут участвовать несколько промежуточных узлов.

Информационный поток непостоянен, и при его возрастании принимающий узел может не успевать обрабатывать приходящие сообщения, а передающий узел может не успевать передавать генерируемые сообщения из-за постоянной занятости канала связи. Поэтому принимающие узлы для накопления поступающих сообщений, передающие узлы для накопления передаваемых сообщений должны иметь буфер. Буфер можно представить в виде очереди сообщений на обработку или передачу. В нашей модели сделаем некоторое упрощение и примем очередь бесконечной. Очереди организуем по методу FIFO.

Подсети могут работать по разным технологиям. При стыке разнородных подсетей передаваемые пакеты или сообщения преобразуются (например, разбиваются на несколько пакетов). Возникает проблема моделирования этого преобразования, а также смена модели дальнейшего продвижения пакета по сети, обусловленная сменой технологии на данном участке движения пакета. Для решения этой проблемы создается библиотека процедур, моделирующие особенности различных технологий. В этой библиотеке также должны содержаться процедуры для моделирования преобразований пакетов при взаимодействии разнородных подсетей. В качестве модели передаваемого сообщения введем абстрактный объект, который назовем транзактом. Определим основные параметры этого объекта:

1. V - объем сообщения;
2. I_i - индекс узла-отправителя;
3. I_o - индекс узла-адресата;
4. I_t - индекс транзитного узла;
5. L - индекс подсети, в которой происходит передача сообщения;
6. T - временной параметр (например, время генерации сообщения или время окончания передачи по каналу связи т.д.).

Приведем описание модели процесса движения одного транзакта (т.е. процесса передачи одного сообщения) в виде схемы:



Нет Да

Рис. 2.8. Схема модели движения одного транзакта

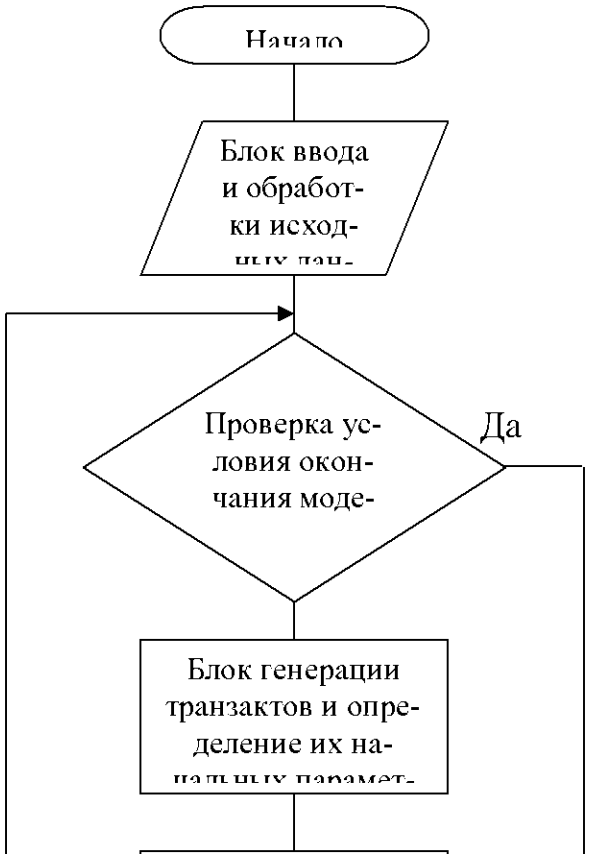
2.7.7. Моделирование доступа к передающей среде

Для блока моделирования доступа к передающей среде необходимы подробные пояснения. В этом блоке вызывается процедура, реализующая метод доступа сетевой технологии подсети, которую стремиться занять данный транзакт. Эта процедура, исходя из общей структуры программного комплекса, должна вызываться из библиотеки подпрограмм, учитывающих особенности сетевых технологий. Метод доступа - это способ, с помощью которого сетевое устройство получает доступ к сети (или управление сетью) для передачи данных. Существуют различные методы доступа к передающей среде, например, в технологии Ethernet используется метод обнаружения коллизий, в технологии Token Ring - маркерный метод доступа.

2.7.8 Разработка программной модели

В корпоративной сети может параллельно функционировать несколько процессов передачи сообщений (это зависит от количества подсетей), но моделирующая программа при выполнении ее на однопроцессорной вычислительной системе выполняется последовательно (в случае многопроцессорной системы количество параллельно обрабатываемых процессов зависит от количества процессоров), поэтому для создания моделирующего алгоритма необходим механизм, синхронизирующий работу нескольких моделей движения транзактов (рис. 2.8).

Приведем общую схему программного комплекса, имитирующего работу корпоративной сети.



Нет

Рис. 2.9. Укрупненная схема универсальной программной модели корпоративной сети

Блок генерации транзактов требует подробных пояснений. В процессе функционирования реальной сети время инициализации и объем передачи для каждого узла, а также количество узлов, одновременно начинающих передачу, случайны. В некоторой степени эти параметры в реальной сети зависят от распределения объема трафика между узлами, от количества активных узлов в сети (т.е. узлов, генерирующих трафик) и от функции почасовой нагрузки сети и других факторов. Для генерации транзактов и определения их параметров необходимо вычислить следующие данные: время отправки транзактов (сообщений); количество узлов, одновременно начинающих передачу; объем информации, отправляемой в этот момент; индексы передающих и принимающих узлов; распределение общего объема информации, отправляемого в это время, между парами передающих и принимающих узлов.

Перед созданием транзактов для одной пары передающий узел - принимающий узел необходимо определить подсеть, по которой начнут

передаваться создаваемые транзакты, и транзитный узел (если передающий и принимающий узлы находятся в одной подсети, то транзитный узел совпадет с узлом-адресатом). Для этого надо знать путь движения транзактов, т.е. последовательность подсетей, по которым необходимо двигаться транзактам, чтобы достичь узла-адресата. Он определятся с помощью рекурсивного алгоритма, который перебирает все возможные пути до тех пор, пока не найдет путь, последней в котором является подсеть, содержащая узел-отправитель. Этот алгоритм можно выразить следующей рекурсивной функцией:

$$S(o,i) = a(o,k) + S(k,i),$$

где o - номер (индекс) узла-отправителя; i - номер (индекс) узла-адресата; k - индекс первого транзитного узла; S - путь от узла отправителя до узла-адресата, представляющий собой последовательность номеров (индексов) подсетей маршрута движения:; a - номер (индекс) подсети, по которой начнет передаваться транзакт. Результатом функции будет последовательность $S = \{a,...,n\}$, где n - последняя подсеть маршрута.

Известен путь - значит, можно определить текущую подсеть, по которой начнут передаваться транзакты, и транзитный узел, связывающий текущую подсеть пути со следующей. Эти же процедуры используются затем в модели процесса движения транзакта (рис. 2.8). После нахождения текущей подсети и транзитного узла создаются транзакты и определяются их параметры.

Объем одного отдельного сообщения случаен в пределах размеров сообщений, принятых в той сетевой технологии, на основе которой работает текущая подсеть. Параметр транзакта "Объем сообщения" находится с помощью функции из библиотеки процедур, учитывающих особенности конкретных сетевых технологий.

Блок имитации процессов передачи (рис. 2.8) представляет собой модель движения транзакта (рис. 2.9), характеристики и компоненты которой меняются в зависимости от параметров транзактов, а также механизм синхронизации движения транзактов. В общей модели сети может функционировать (передаваться) параллельно несколько транзактов, но движение каждого из них имитируется одним и тем же алгоритмом (рис. 2.9), т.е. реально в модели в один момент времени может функционировать только один транзакт. Таким образом, функционирование общей модели сети должно протекать в искусственном времени, изменяя которое определенным образом, можно обеспечивать появление и движение транзактов в

надлежащем порядке и с надлежащими временными интервалами между ними. Искусственное время также называют модельным временем.

Рассмотрим следующий частный случай. Допустим, в один момент два узла из разных подсетей сгенерировали по транзакту, и пока они не будут переданы других транзактов не возникнет, в этот момент начало времени передачи транзактов совпадает с модельным временем. В реальной сети они могут передаваться параллельно, но в модели двигается сначала один транзакт. При занятии подсети этот транзакт задерживается на время передачи по данному каналу связи, в этот момент начинает двигаться второй транзакт, который занимает другую подсеть и также задерживается на время передачи по этой подсети. Во время движения по модели каждый транзакт вносит соответствующие изменения в некоторые статистические показатели системы. Первый освободившийся транзакт, т.е. транзакт с меньшей задержкой (временные координаты транзактов сохраняются в их временных параметрах), устанавливает модельное время равным времени окончания передачи транзакта по каналу связи. Этот транзакта входит в принимающий или транзитный узел и задерживается там на время обработки, а модельное время становится равным времени окончания передачи второго транзакта, т.е. по модели двигается уже другой транзакт и т.д. Параллельно функционирующих транзактов может быть намного больше, причем во время задержек транзактов в подсетях и узлах могут появляться другие транзакты. При переключении между этими транзактами происходят соответствующие изменения модельного времени. В реальный момент времени в модели функционировать может только один транзакт, назовем его активным. В любой момент модельное время соответствует реальным временным координатам активного транзакта. Таким образом, синхронизация параллельных процессов движения транзактов происходит за счет изменения модельного времени, так называемой "протяжки модельного времени".

На основе описанных принципов была создана программа, моделирующая корпоративные вычислительные сети.

В качестве инструментальных средств для создания программного комплекса были выбраны Delphi 4.0 и система моделирования СИМПАС.

3. Система имитационного моделирования GPSS World

3.1. Общие сведения

GPSS World основан на оригинальном языке имитационного моделирования GPSS (General Purpose Simulation System - общецелевая система моделирования. Этот язык был разработан IBM в 1960 г. и в настоящее время входит в число наиболее распространенных языков имитационного моделирования. GPSS World - это комплексный моделирующий инструмент, охватывающий области как дискретного, так и непрерывного компьютерного моделирования, обладающий достаточно высоким уровнем интерактивности и визуального представления информации.

GPSS World является объектно-ориентированным языком моделирования, улучшенный встроенным процедурным языком программирования

PLUS. В системе GPSS World существует 4 основных вида объектов: объекты "Модель", "Процесс моделирования", "Отчет" и текстовые объекты.

Объект "Модель" содержит последовательность операторов модели (операторы языка GPSS или определения PLUS-процедур), а также набор встроенных величин, называемых настройками. Таким образом, объект "Модель" описывается с помощью объектов языка GPSS. Кроме того, частью объекта "Модель" является совокупность закладок и циркулярный результирующий список синтаксических ошибок. После трансляции созданный объект "Процесс моделирования" наследует все настройки породившего объекта "Модель". Операторы модели определяют объект "Процесс моделирования" при первой трансляции, а также могут быть введены в интерактивном режиме во время выполнения процесса моделирования. Объект "Процесс моделирования" выполняет следующие основные функции:

- обеспечение заданных в объекте "Модель" маршрутов продвижения динамических объектов - транзактов;
- планирование событий, происходящих в модели, путем регистрации времени наступления каждого события и выполнения их в нарастающей временной последовательности;
- регистрация статистической информации о функционировании модели;
- продвижение модельного времени в процессе моделирования системы;
- обеспечение интерактивной настройки параметров прогона модели.

Объекты типа "Отчет" используются для получения стандартных отчетов GPSS World. Текстовый объект - это способ представления простого текстового файла. Эти объекты используются для получения доступа к разделяемому набору операторов модели из объекта "Модель", для считывания и записи данных в текстовые файлы, а также для создания файлов результатов моделирования для последующего анализа.

На уровне интерфейса GPSS World представляет собой реализацию архитектуры "документ-вид", общей для всех приложений Windows.

3.2. Интерфейс GPSS World

Главное окно имеет вид, показанный на рис. 3.1.

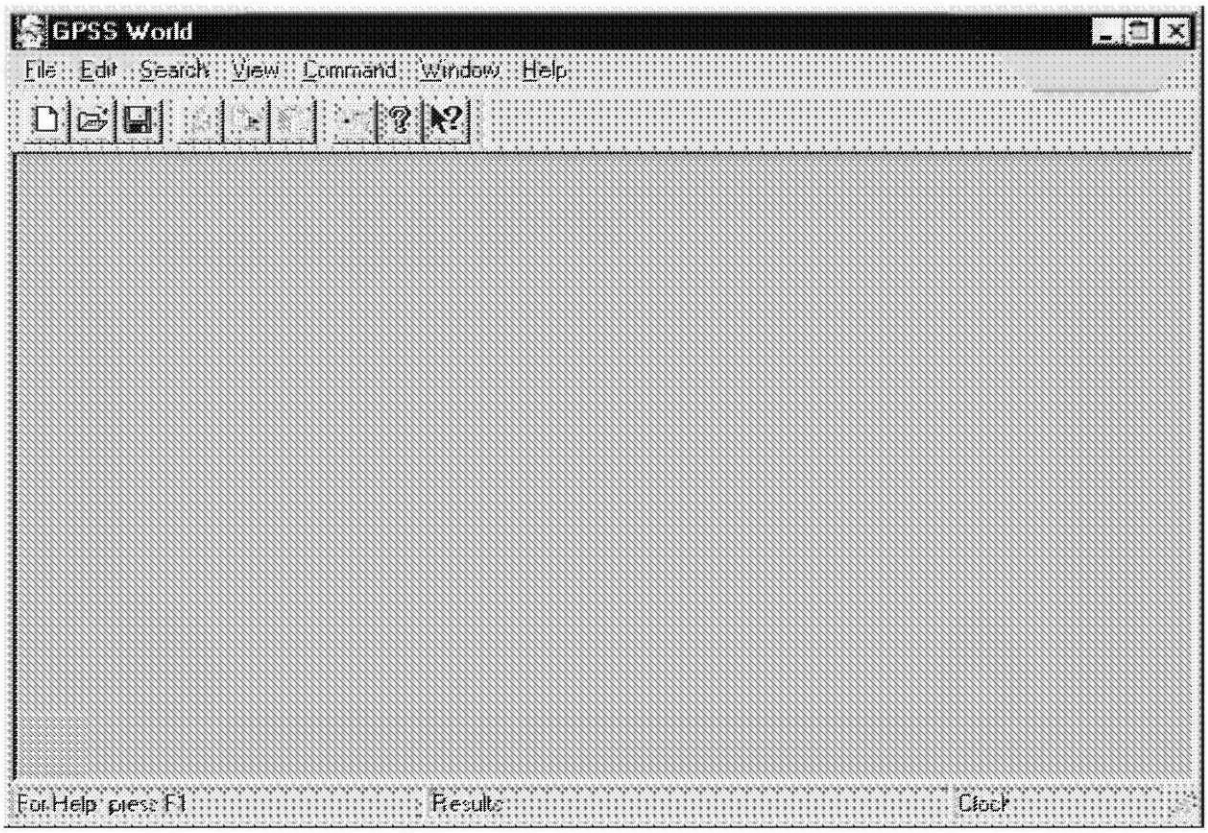


Рис. 3.1. Главное окно GPSS World

Основное меню содержит следующие опции:

File – работа с файлами;

Edit – редактирование исходной программы и общие настройки системы моделирования;

Search – инструменты поиска, замены и закладки;

View – выбор элементов для отображения;

Command – команды управления процессом моделирования;

Window – команды управления режимом отображения окон документов и отображение информации об объектах модели;

Help – информационно-справочная система.

Рассмотрим подробнее опцию меню Command:

Create Simulation – трансляция операторов модели в исполняемый объект “Процесс моделирования;

Retranslate – ретрансляция объекта “Процесс моделирования”;

Repeat Last Command – повторить последнюю команду;

Оставшиеся команды представляют формальные команды GPSS. Последняя команда меню Custom… позволяет ввести собственный оператор модели, не перечисленный в меню Command.

3.3. Объекты GPSS

GPSS построена на допущении, что любую сложную систему можно представить на основе нескольких элементарных абстрактных понятий, называемых объектами. Основные типы объектов - это транзакты и блоки, а моделирование в основном заключается в продвижении множества транзактов от одного блока модели к другому. Объекты GPSS нумеруются. При использовании имени для обращения к объекту в модели оно автоматически преобразуется в целое число, которое используется для поиска соответствующего объекта. Большинство объектов GPSS создаются автоматически по мере необходимости. Но некоторые объекты перед их использованием должны быть объявлены заранее. Как правило, они имеют такой атрибут, как размер, который должен быть известен объекту "Процесс моделирования":

— памяти должны объявляться операторами STORAGE;
— арифметические переменные должны объявляться операторами VARIABLE;
— переменные с плавающей точкой должны объявляться операторами FVARIABLE;
— булевы переменные должны объявляться операторами BVARIABLE;
— матрицы должны объявляться операторами MATRIX;
— таблицы должны объявляться операторами TABLE и QTABLE;
— функции должны объявляться операторами FUNCTION;
— параметры транзактов должны объявляться в блоках ASSIGN, MARK, SELECT, READ, SPLIT, COUNT или TRANSFER SBR перед обращением к ним.

3.3.1. Транзакты

В процессе моделирования транзакты двигаются от блока к блоку, имитируя функционирование реальной системы. Как только транзакт начал движение в модели, он продолжает двигаться до тех пор, пока это возможно. В процессе моделирования транзакт, пытающийся войти из одного блока в другой, называется активным. Если для транзакта выполняется условия для входа в блок, то он задерживается. Тогда по модели начинает двигаться другой транзакт, это происходит до тех пор, пока он в свою оче-

редь не перейдет в состояние ожидания. Поведение транзакта как объекта определяется его атрибутами или свойствами. Основными атрибутами транзакта являются:

- Номер транзакта.
- Время появления транзакта в системе.
- Номер блока, в котором находится транзакт.
- Номер следующего блока, в который будет пытаться перейти данный транзакт.
- Приоритет транзакта.
- Указатель семейства. Семейства используются для обеспечения синхронизации движения транзактов.
- Состояние транзакта:

ACTIVE - транзакт имеет наивысший приоритет и находится в списке будущих событий;

SUSPENDED - транзакт находится в списке будущих событий или в списке текущих событий и ожидает возможности стать активным;

PASSIVE - транзакт перешел в состояние ожидания и находится в списке пользователя, списке задержки или списке отложенных прерываний;

TERMINATED - транзакт удаляется и больше не существует в процессе моделирования;

PREEMPTED - обслуживание транзакта в устройстве прервано, и он находится одном и более списке прерываний, это состояние не является взаимоисключаемым по отношению к указанным выше.

- Параметры транзактов и др.

Параметры транзактов отображают свойства моделируемого динамического объекта. Например, если моделируется движение автомобилей на участке дороги, то параметрами транзакта, имитирующего автомобиль, в зависимости от целей моделирования могут быть скорость, грузоподъемность, вес перевозимого груза и т.п.

3.3.2. Блоки

Блок GPSS является основным структурным элементом процесса моделирования. Последовательность блоков GPSS-модели показывает направления, в которых перемещаются транзакты. Блоки представляют собой подпрограммы, написанные на макроассемблере или на языке Си, и

содержат набор параметров (операндов) для обращения к ним. Передача управления от блока к блоку реализуется с помощью движения транзактов в модельном времени. Обращение к подпрограммам блоков происходит через движение транзактов.

3.3.3. Устройства

Устройства (приборы) - это объекты, которые имеют несколько атрибутов, наиболее важным из которых является право на его использование. Право на использование устройства в любой момент времени может иметь только один транзакт. Таким образом устройство может быть занято или свободно. Существует возможность замены транзакта, в данный момент занимающего устройство, активным транзактом - реализация абсолютного приоритета. В случае, если транзакт не может занять устройство, он задерживается и дожидается его освобождения. Устройство имеет несколько списков для транзактов, которые ожидают наступления события, связанного с устройством. Каждое устройство имеет список задержки для транзактов, ожидающих освобождения устройства; список отложенных прерываний для транзактов, претендующих на прерывание обслуживания других транзактов; список прерываний, в который входят транзакты, обслуживание которых было прервано. Когда устройство освобождается, следующий владелец устройства выбирается из списков транзатов, ожидающих занятие устройства: первыми выбираются транзакты из списка отложенных прерываний, затем выбираются ранее прерванные транзакты и только потом выбираются транзакты, ожидающие освобождения устройства в обычной очереди в соответствии с приоритетами.

3.3.4. Памяти (многоканальные устройства)

Память (многоканальное устройство) представляет собой объект, состоящий из элементов памяти, которые занимаются и освобождаются транзактами. Он является аналогом таких объектов реальных систем, как склад, память компьютера или параллельно обслуживающих приборов. Когда транзакт входит в память, он может занимать один или более элементов памяти. Транзакт не может входить в память если его потребность в ячейках памяти не может быть удовлетворена, поэтому он должен дождаться, пока другие транзакты не освободят достаточное количество памя-

ти. Емкость памяти может освобождаться любым транзактом, даже если он перед этим не входил в память.

3.3.5. Логические ключи

Некоторые события в системе могут заблокировать или изменить движение транзактов. Например, переключение светофора на перекрестке меняет направление движения автотранспорта. Для моделирования подобных ситуаций введены логические ключи. Транзакт может устанавливать эти ключи положение "Включено" (1) или "Выключено" (0). Через некоторое время состояние ключа может быть использовано другими транзактами для выбора одного из двух возможных путей движения или ожидания момента изменения состояния ключа.

3.3.6. Переменные

Арифметические переменные позволяют вычислять арифметические выражения, которые состоят из операций над атрибутами объектов. В выражених могут быть использованы функции (библиотечные и пользовательские). Булевы переменные позволяют пользователю одновременно проверять несколько условий, исходя из атрибутов объектов.

3.3.7. Функции

Используя функции, пользователь может задавать непрерывную или дискретную функциональную зависимость между аргументом функции и ее значением. Функции в GPSS задаются табличным способом с помощью операторов описания функций.

3.3.8. Ячейки и матрицы

Ячейки сохраняемых величин и матрицы используются для хранения некоторой пользовательской числовой информации, запись в эти объекты и считывание из них выполняют транзакты. Матрицы - это массив элементов, каждый из которых может иметь какое-то значение. Матрицы могут иметь размерность до 6, и могут быть глобальными и постоянными, или локальными и временными. Временные матрицы определяются в соответ-

ствии с объявлениями временных матриц в PLUS-процедурах. Они создаются во время вызова процедуры и позже удаляются. Они имеют локальную область видимости, и к ним можно обращаться только из процедуры, в которой они объявлены.

3.3.9. Очереди

В любой системе движение потока транзактов может быть задержано из-за недоступности ресурсов (например, необходимые устройства или памяти уже заняты). В этом случае задержанные транзакты становятся в очереди. Для сбора статистики и регистрации количества входов, максимальной длины очереди и среднего времени ожидания в GPSS введены специальные объекты - очереди.

3.3.10. Таблицы

Объект "таблица" предназначен для сбора статистики о случайных величинах, заданных пользователем. Таблица состоит из частотных классов, в которые заносится число попаданий конкретной величины. Для каждой таблицы вычисляется математическое ожидание и среднеквадратическое отклонение. На основе таблиц строятся гистограммы.

3.3.11. Списки пользователя

Список пользователя содержит некоторую совокупность транзактов. Списки пользователей удобны для моделирования сложных алгоритмов планирования и формирования очередей. Они обеспечивают динамичное управление постановкой транзактов в очередь.

3.3.12. Числовые группы

Числовая группа - это набор числовых величин. Числовые группы удобны для записи событий или описания состояния процессов, которые вы моделируете.

3.3.13. Группы транзактов

Группы транзактов - это набор транзактов. Ограничения на количество транзактов в группе транзактов и количество групп транзактов, к которым может принадлежать отдельный транзакт не существует. Группы транзактов удобны для классификации транзактов и доступа к ним. Активный транзакт может проверять параметры членов группы транзактов.

3.3.14. Потоки данных

Поток данных это последовательность текстовых строк, используемых процессом моделирования. Каждый поток данных идентифицируется уникальным номером, с тем, чтобы сразу несколько из них могли использоваться в одно и то же время внутри одного процесса моделирования. С помощью потока данных можно считывать и записывать данные в файл или поддерживать в памяти компьютера набор данных с прямым доступом.

3.3.15. Стандартные числовые атрибуты

В процессе моделирования GPSS регистрирует и корректирует определенную информацию различных объектов в модели. Каждому объекту соответствует некоторый набор свойств, описывающий его состояние в данный момент времени, которые называются стандартными числовыми атрибутами (СЧА).

Приведем полный список СЧА, доступных в GPSS World:

• A1 - Семейство активного транзакта. Целочисленное значение.

• AC1 - Значение абсолютного модельного времени. (Модельное время с момента последней команды CLEAR). Вещественное значение.

• BV*Entnum* - Значение булевой переменной *Entnum*. Вещественное значение.

• C1 - Значение относительного модельного времени. (Модельное время с момента последней команды RESET). Вещественное значение.

• CA*Entnum*-Среднее содержимое списка пользователя. Взвешенное по времени среднее количество транзактов, помещенных в список пользователя. Вещественное значение.

• CC*Entnum* - Общее число входов транзактов в список пользователя. Число транзактов, находившиеся в списке пользователя *Entnum*. Целочисленное значение.

• CH*Entnum* - Текущее содержимое списка пользователя. Текущее количество транзактов, находящихся в списке пользователя *Entnum*. Целочисленное значение.

• CM*Entnum* - Максимальное содержимое списка пользователя. Максимальное количество транзактов, одновременно находившихся в списке пользователя *Entnum*. «Верхняя отметка уровня». Целочисленное значение.

• CT*Entnum* - Среднее время пребывания одного транзакта в списке пользователя. Средняя продолжительность пребывания транзактов в списке пользователя *Entnum*. Вещественное значение.

• F*Entnum* - Состояние устройства. Если в данный момент устройство *Entnum* занято F*Entnum* возвращает 1. Иначе F*Entnum* возвращает 0. Целочисленное значение.

• FC*Entnum* - Счетчик использования устройства. Количество транзактов, занимавших устройство *Entnum* с помощью блоков SEIZE и PREEMPT. Целочисленное значение

• FI*Entnum* - Устройство прервано. Если устройство *Entnum* в данный момент было занято более приоритетным транзактом, FI*Entnum* возвращает 1. Иначе FI*Entnum* возвращает 0. Целочисленное значение.

• FU*Entnum*- Функция. Результат вычисления значения функции *Entnum*. Вещественное значение.

• FR*Entnum*- Коэффициент использования устройства. Отношение времени, в течение которого устройство *Entnum* было занято, к общему времени моделирования, FR*Entnum* выражается в долях от тысячи и возвращает значения, изменяющиеся от 0 до 1000 включительно. Может быть нецелочисленным. Вещественное значение.

• FT*Entnum* - Среднее время удержания устройства. Среднее время удержания устройства *Entnum* одним транзактом. Вещественное значение.

• FV*Entnum* - Устройство доступно. FV*Entnum* возвращает 1, если устройство *Entnum* в состоянии «доступно», и 0, если устройство в состоянии «не доступно». Целочисленное значение.

• GN*Entum* - Счетчик числовой группы. GN*Entnum* возвращает количество элементов числовой группы *Entnum*. Целочисленное значение.

• GT*Entnum*- Счетчик группы транзактов. *GTEntnum* возвращает количество элементов группы транзактов *Entnum*. Целочисленное значение.

• LS*Entnum* - Состояние логического ключа. *LSEntnum* возвращает 1, если логический ключ *Entnum* в установленном состоянии, и 0, если логический ключ в сброшенном состоянии. Целочисленное значение.

• MB*Entnum* - Соответствие в блоке MATCH. MB*Entnum* возвращает 1, если в сопряженном блоке MATCH присутствует транзакт, принадлежащий тому же семейству что и активный транзакт. В противном случае MB*Entnum* возвращает 0. Целочисленное значение.

• MP*Parameter*- Транзитное время параметра. Текущее абсолютное модельное время за вычетом значения, содержащегося в параметре *Parameter*. Вещественное значение.

• MX*Entnum(m,n)* - Ячейка матрицы. Возвращает значение ячейки в строке m, столбце m матрицы *Entnum*. Для матриц, имеющих более 2 измерений, все остальные индексы считаются равными 1. В отличие от СЧА класса MX, PLUS-выражения могут обращаться к любому элементу многомерных матриц.

• M1 - Транзитное время, M1 возвращает абсолютное модельное время за вычетом времени входа транзакта в систему. Вещественное значение.

• N*Entnum* – Счетчик входов в блок. Возвращает общее количество транзактов, которые входили в блок *Entnum* . Целочисленное значение.

• P*Parameter* или **Parameter* – Значение параметра. Возвращает значение параметра *Parameter* активного транзакта. Целочисленное, вещественное или строковое значение. Для косвенной адресации используется вариант **Parameter*.

• PR - Приоритет транзакта. Значение приоритета активного транзакта. Целочисленное значение.

• Q*Entnum*- Текущее содержимое очереди. Текущее значение количества транзактов в очереди *Entnum*. Целочисленное значение.

• QA*Entnum* - Среднее значение длины очереди. Взвешенное по времени количество транзактов в очереди *Entnum*. Вещественное значение.

• QC*Entnum* - Общее количество входов в очередь. Сумма всех входов транзактов в

очередь *Entnum*. Целочисленное значение.

• QM*Entnum* - Максимальное содержимое очереди. Максимальное количество транзактов, находившихся в очереди *Entnum*. «Верхняя отметка уровня». Целочисленное значение.

• QT*Entnum* - Среднее время пребывания в очереди. Взвешенное среднее время пребывания транзактов в очереди *Entnum*. Вещественное значение.

• QX*Entnum* - Среднее время пребывания в очереди за исключением

нулевых входов. Взвешенное среднее время пребывания транзактов в очереди *Entnum*, не считая входов со временем пребывания, равным 0. Вещественное значение.

• QZ*Entnum* - Количество входов с нулевым временем пребывания в очереди. Количество входов транзактов, время пребывания в очереди *Entnum* которых равно нулю. Целочисленное значение

• R*Entnum* - Объем свободной памяти. Содержимое памяти *Entnum*, доступное для входов транзактов. Целочисленное значение.

• RN*Entnum* - Случайное число. RN*Entnum* возвращает случайное целое число от 0 до 999 из генератора случайных чисел *Entnum*. Целочисленное значение.

• S*Entnum*- Объем занятой памяти, S*Entnum* возвращает количество элементов памяти *Entnum*, занятых транзактами. Целочисленное значение.

• QA*Entnum* - Среднее значение занятой памяти. QA*Entnum* возвращает взвешенное по времени среднее количество занятых элементов памяти *Entnum*. Вещественное значение.

• SC*Entnum* - Счетчик использования памяти. Общее количество использовавшихся элементов памяти *Entnum*. Целочисленное значение.

• SE*Entnum* - Память пуста. SE*Entnum* возвращает 1, если все элементы ячейки памяти *Entnum* пусты, и 0, если нет. Целочисленное значение.

• SF*Entnum* - Память заполнена, SF*Entnum* возвращает 1, если память *Entnum* полностью заполнена, и 0, если нет. Целочисленное значение.

• SR*Entnum* - Коэффициент использования памяти. Доля общего использования памяти *Entnum*, представленная отношением среднего количества используемых элементов памяти *Entnum к* общему количеству элементов памяти *Entnum*. SR*Entnum* выражается в долях от тысячи и возвращает значения, изменяющиеся от 0 до 1000 включительно. Может быть нецелочисленным. Вещественное значение.

• SM*Entnum* - Максимальное количество занятых элементов памяти *Entnum*. «Верхняя отметка уровня». Целочисленное значение.

• ST*Entnum* - Среднее время использования одного элемента памяти *Entnum*. Вещественное значение.

• SV*Entnum* - Память в доступном состоянии. SV*Entnum* возвращает 1, если память *Entnum* в состоянии “доступно”, и 0, если память *Entnum* в состоянии “не доступно”. Целочисленное значение.

• TB*Entnum* - Среднее значение невзвешенных аргументом таблицы

Entnum. Вещественное течение.

• TC*Entnum* – Количество невзвешенных аргументов таблицы *Entnum*. Целочисленное значение.

• TD*Entnum* - Стандартное отклонение невзвешенных аргументов таблицы *Entnum* Вещественное значение.

• TG1 - Текущее значение счетчика завершения. TG1 возвращает значение уменьшаемое блоками TERMINATE с положительным значением операнда A. значение задается оператором START и указывает на завершение процесса моделирования, когда становится равным нулю. Целочисленное значение.

• V*Entnum* - Результат вычисления целочисленной переменной *Entnum* или переменной с плавающей точкой *Entnum*. Вещественное значение.

• W*Entnum* - Текущий счетчик блока. Текущее количество транзактов в блоке *Entnum*. Целочисленное значение.

• X*Entnum* - Ячейка. Возвращает значение ячейки *Entnum*. Целочисленное, вещественное или строковое значение.

• XN1 - Номер активного транзакта. Возвращает номер активного транзакта. Целочисленное значение.

• Z1 - Свободная системная память. Величина, возвращаемая операционной системой. Целочисленное значение.

Во всех случаях *Entnum* необходимо заменять одним из спецификаторов объектов. Это может быть имя (перед именем должен стоять символ “$”) или номер.

3.4. Таймер модельного времени

Транзакты моделируют прохождение по системе соответствующих единиц исследуемого потока. Такое движение может быть разбито на цепь элементарных событий, происходящих в определенные моменты времени. Основной задачей интерпретатора GPSS является определение моментов наступления этих событий, расположение их в правильной временной последовательности и выполнение соответствующих действий при наступлении каждого события. Чтобы обеспечить правильную последовательность обработки событий во времени, в GPSS имеются "системные часы", хранящие значение абсолютного модельного времени. Абсолютное модельное время обнулить нельзя.

Относительное время - время, наращиваемое в модели по мере поступления транзактов до тех пор, пока не произойдет сброс статистики. При отсутствии в модели режима сброса значения относительного и абсолютного модельного времени совпадают.

Единица модельного времени определяется разработчиком. Эту единицу времени интерпретатору не сообщают. Значение принятой единицы модельного времени выражают в неявном виде в форме временных данных модели. Единица модельного времени - это абстрактная временная величина, связанная с некоторым реальным временным отрезком, причем за правильность этой связи отвечает разработчик модели. Разработчик исходя из собственных соображений должен выбрать единицу модельного времени и провести масштабирование всех временных величин, вводимых в модель, относительно выбранной единицы модельного времени.

Если за единицу модельного времени разработчиком принимается отрезок в одну минуту, то все временные промежутки необходимо масштабировать относительного временного интервала, равного одной минуте. То есть временной промежуток в 5 минут составит 5 единиц модельного времени, а временной промежуток в 1 час - 60 единиц модельного времени. Разработчик может задавать такую единицу модельного времени, которая ему удобна, для того чтобы правильно отразить события реальной системы в своей модели. При этом необходимо строго следить за тем, чтобы все данные, связанные со временем, были выражены через заданную единицу модельного времени.

3.5. Стандартные отчеты

3.5.1. Общая информация

Строка заголовка стандартного отчета содержит имя файла модели, который создал отчет. Также в нее включена дата и время прогона модели.

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	5187.692	33	3	1

- START TIME - Начальное время. Абсолютное модельное время в начале периода измерения. Коэффициенты использования и пространственно-временные результаты основываются на начальном времени. Начальное время устанавливается равным абсолютному модельному времени с помощью операторов RESET или CLEAR,

• END TIME - Конечное время. Абсолютное модельное время, когда счетчик завершения становится равным 0.

• BLOCKS - Количество блоков в модели на момент завершения процесса моделирования.

• FACILITIES - Количество устройств в модели на момент завершения процесса моделирования.

• STORAGES - Количество памятей в модели на момент завершения процесса моделирования.

3.5.2. Имена

NAME	VALUE
ADDTO	10007.000
CHAIN 1	10012.000
COLLECT	10017.000

• NAME – Имя. Заданные пользователем имена, использованные в вашей модели GPSS World со времени последней трансляции.

• VALUE - Значение. Числовое значение, присвоенное имени. Числа заданные системой, начинаются с 10000.

3.5.3. Блоки

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	325	0	0
	2	ASSIGN	325	0	0
	3	TEST	325	0	0
	4	LEAVE	325	0	0
	5	QUEUE	325	0	0

• LABEL - Метка. Алфавитно-числовое имя данного блока (если оно задано).

• LOC - Числовой номер позиции данного блока в модели. "Местоположение».

• BLOCK TYPE - Тип блока. Имя блока GPSS.

• ENTRY COUNT - Количество транзактов, вошедших в данный блок с момента выполнения последнего оператора RESET или CLEAR или с момента последней трансляции

• CURRENT COUNT - Количество транзактов в данном блоке к моменту завершения процесса моделирования.

• RETRY – количество транзактов, ожидающих выполнения специфического условия , зависящего от состояния данного блока.

3.5.4. Устройства

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
MACHINE	325	0.354	0.087	1	0	0	0	0	0

• FACILITY - Имя или номер устройства.

• ENTRIES - Количество раз, которое устройство было занято или занято с прерыванием с момента выполнения последней команды RESET или CLEAR или с момента последней трансляции модели.

• UTIL. - Коэффициент использования. Доля времени моделирования за последний период измерения, в течение которого устройство было занято. Период измерения начинается с момента трансляции модели или выполнения команды RESET или CLEAR.

• AVE. TIME - Среднее время занятия устройства одним транзактом в течение периода измерения. Период измерения начинается с момента трансляции модели или выполнения команды RESET или CLEAR.

• AVAIL. - Состояние устройства к моменту завершения процесса моделирования. 1 означает, что устройство доступно, 0 - не доступно.

• OWNER - Номер транзакта, который занимает устройство. 0 означает, что устройство не занято.

• PEND. - Количество транзактов, ожидающих выполнения с прерыванием других транзактов (т.е. вошедших в блоки PREEMPT в «прерывающем режиме»).

• INTER. - Количество транзактов, на данный момент вытесненных из устройства (прерванных). Количество транзактов в списке прерываний.

• RETRY - Количество транзактов, ожидающих выполнения специфического условия, зависящего от состояния данного устройства.

• DELAY. Количество транзактов, ожидающих занятия устройства. Этот список также содержит транзакты, ожидающие занятия устройства с

помощью блоков PREEMPT в “режиме приоритетов”.

3.5.5. Очереди

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
PACKING	4	0	325	208	0.104	0.026	0.071	0

• QUEUE - Имя или номер очереди.

• MAX. - Максимальное содержимое очереди в течение периода измерения. Период измерения начинается с момента трансляции модели или выполнения команды RESET или CLEAR.

• CONT. - Текущее содержимое очереди к моменту завершения процесса моделирования.

• ENTRY - Общее количество входов транзактов в очередь в течение периода измерения.

• ENTRY(0) - Общее количество входов в очередь с нулевым временем пребывания в очереди.

• AVE.CONT. - Взвешенное по времени среднее содержимое очереди в течение периода измерения. Пространственно-временной результат, поделенный на продолжительность времени периода измерения.

• AVE.TIME. - Среднее время пребывания в очереди одного транзакта в течение периода измерения. Пространственно-временной результат, поделенный на общую продолжительность времени периода измерения.

• AVE.(-0) - Среднее время пребывания транзакта в очереди одного транзакта в течении периода измерения без учета “нулевых входов”. Пространственно-временной результат, поделенный на общее количество входов без учета нулевых входов.

• RETRY – Количество транзактов, ожидающих выполнения специфического условия, зависящего от состояния очереди.

3.5.6. Памяти

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
STOCK	4000	2928	0	1378	7300	1	1132.973	0.283	0	0

• STORAGE - Имя или номер памяти.

• CAP. - Емкость памяти, определенная оператором STORAGE.

• REM. - Количество неиспользуемых элементов памяти на момент завершения процесса моделирования.

• MIN. - Минимальное количество элементов памяти, использовавшихся в течении периода измерения. Период измерений начинается с момента трансляции модели или выполнения команды RESET или CLEAR.

• MAX. – Максимальное количество элементов памяти, использо-

вавшихся в течение периода измерения.

• ENTRIES - Количество “входов” в память в течение периода измерения. Общая сумма значений операндов B операторов ENTER..

• AVL. - Состояние памяти к моменту завершения процесса моделирования. 1 означает доступное состояние, 0 - не доступное.

• AVE.C. - Взвешенное по времени среднее содержимое памяти в течение периода измерения. Пространственно-временной результат, поделенный на продолжительность периода измерения.

• UTIL - Коэффициент использования памяти.

• RETRY - Количество транзактов, ожидающих выполнения специфического условия, зависящего от состояния данной памяти.

• DELAY- Количество транзактов, ожидающих в блоках ENTER, связанных с данной памятью, на момент завершения процесса моделирования.

3.5.7. Таблицы

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
TRANSIT	0.113	0.055				0		
			0.045	-	0.060		22	6.77
			0.060	-	0.075		52	22.77
			0.075	-	0.090		66	43.08
			0.090	-	0.105		51	58.77
			0.105	-	0.120		43	72.00
			0.120	-	0.135		20	78.15
			0.135	-	0.150		17	83.38
			0.150	-	0.165		7	85.54
			0.165	-	0.180		17	90.77
			0.180	-	0.195		8	93.23
			0.195	-	0.210		2	93.85
			0.210	-	0.225		3	94.77
			0.225	-	0.240		3	95.69
			0.240	-	0.255		2	96.31
			0.255	-	0.270		1	96.62
			0.270	-	0.285		5	98.15
			0.285	-			6	100.00

• TABLE - Имя или номер таблицы;

• MEAN - Взвешенное арифметическое среднее табличных значений.

• STD.DEV. - Взвешенная выборка среднеквадратического отклонения табличных значений.

• RANGE - Нижний и верхний пределы частотного класса. Значения табличного аргумента, которые больше нижнего и меньше или равны верхнему пределу, приводят к обновлению частотного класса. Операнд B оператора TABULATE может использоваться для обеспечения весового

коэффициента. Частотные классы, не содержащие данных, в отчет не включаются.

• RETRY - Количество транзактов, ожидающих выполнения специфического условия, зависящего от состояния данной таблицы.

• FREQUENCY — Общее взвешенное количество табличных элементов, попавших в данный частотный класс. Общее количество операндов В оператора TABULATE.

• CUM.% - Накопленная частота, выраженная в процентах от общего количества.

3.5.8. Списки пользователя

USER CHAIN	SIZE	RETRY	AVE.CONT	ENTRIES	MAX	AVE.TIME
CHAIN1	0	0	0.218	8	1	27.650

• USER CHAIN - Имя или номер списка пользователя.

• SIZE - Количество транзактов, находящихся в списке пользователя, к моменту завершения процесса моделирования.

• RETRY - Количество транзактов, ожидающих выполнения специфического условия, зависящего от состояния данного списка пользователя.

• AVE.CONT. - Взвешенное по времени среднее количество транзактов в списке в течение периода измерения. Пространственно-временной результат, поделенный на продолжительность периода измерения.

• ENTRIES - Общее количество транзактов, входивших в список пользователя в течение периода измерения.

• MAX - Максимальное количество транзактов в списке пользователя в течение периода измерения.

• AVE.TIME - Среднее время пребывания транзакта в списке пользователя в течение периода измерения. Пространственно-временной продукт, поделенный на общее количество входов.

3.5.9. Группы транзактов

XACT GROUP	GROUP SIZE	RETRY
MAINGRP	4	0

• XACT GROUP - Имя или номер группы транзактов.

• GROUP SIZE - Количество транзактов, входящих в группу к мо-

менту завершения процесса моделирования.

• RETRY - Количество транзактов, ожидающих выполнения специфического условия, зависящего от состояния данной группы транзактов.

3.5.10. Числовые группы

NUMERIC GROUP	GROUP SIZE	RETRY
NUMGRP	2	0

• NUMERIC GROUP - Имя или номер числовой группы.

• GROUP SIZE - Количество транзактов, входящих в группу к моменту завершения процесса моделирования.

• RETRY - Количество транзактов, ожидающих выполнения специфического условия, зависящего от состояния данной числовой группы.

3.5.11. Логические ключи

LOGICSWITCH	VALUE	RETRY
SWITCH_1	0	0

• LOGICSWITCH - Имя или номер логического ключа.

• VALUE - Значение логического ключа к моменту завершения процесса моделирования. 1 означает “установлен” или “истина”, 0 – “сброшен” или “ложь”.

• RETRY - Количество транзактов, ожидающих выполнения специфического условии, зависящего от состояния данного логического ключа.

3.5.12. Ячейки

SAVEVALUE	RETRY	VALUE
ADDUP	0	14.000
COLLECT	0	-10

• SAVEVALUE - Имя или номер ячейки.

• VALUE — Значение ячейки к моменту завершения процесса моделирования.

• RETRY - Количество транзактов, ожидающих выполнения специ-

фического условия, зависящего от состояния данной ячейки.

3.5.13. Матрицы

```
MATRIX      RETRY       INDICES     NUMERIC VALUE MA-
TRIX1                        0
                  Intermediate Values are Zero.
                        (2,2)          6.113
                  Intermediate Values are Zero.
```

• MATRIX - Имя или номер матрицы.

• RETRY - Количество транзактов, ожидающих выполнения специфического условия зависящего от состояния данной матрицы.

• INDICES - До 6 целых чисел, определяющих элемент матрицы.

• NUMERIC VALUE - Значение данного элемента матрицы к моменту завершения процесса моделирования. Элементы с нулевыми значениями выводятся в отчете группами.

3.5.14. Список текущих событий

```
CEC XN   PRI      M1     ASSEM  CURRENT  NEXT  PARAMETER   VALUE
  13      0       3987.350  13                3       4
```

• XN - Номер каждого транзакта, находящегося в списке текущих событий.

• PRI - Приоритет транзакта.

• M1 - Время входа транзакта в систему. Время создания транзакта или его самого раннего предка (породившего транзакта) или время входа транзакта в блок MARK без операндов.

• ASSEM - Номер семейства данного транзакта.

• CURRENT - Номер блока, а котором находился транзакт к моменту завершения процесса моделирования или к моменту создания данного отчета.

• NEXT - Номер следующего блока, запланированного для данного транзакта.

• PARAMETER - Имя или номер параметра транзакта.

• VALUE - значение параметра.

3.5.15. Список будущих событий

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
20	0	5949.774	20	0	1		

• XN - Номер каждого транзакта, находящегося в списке будущих событий.

• PRI - Приоритет транзакта.

• BDT - Время выхода из блока. Значение абсолютного модельного времени, когда планируется вывести транзакт из списка будущих событий.

• ASSEM - Номер семейства данного транзакта.

• CURRENT - Номер блока, в котором находился транзакт к моменту завершения процесса моделирования или к моменту создания данного отчета.

• NEXT - Номер следующего блока, запланированного для данного транзакта.

• PARAMETER - Имя или номер параметра транзакта.

• VALUE-Значение параметра.

3.6. Формат операторов GPSS

Модель в системе GPSS World представляет собой последовательность операторов. Операторами модели могут быть операторы языка GPSS или определения PLUS-процедур. Под оператором языка GPSS понимается команда или блок. Операторы GPSS должны представляться одной текстовой строкой не более, чем из 250 символов. Оператор состоит из следующих полей:

Метка Операция Операнды ;Комментарии

Поле метки дает возможность пользователю присваивать собственные имена объектам (устройствам, переменным, таблицам и т.п) и обращаться к ним с помощью имен. Наименование метки не должно совпадать со служебными словами GPSS, поэтому рекомендуется использовать в названии метки и не меньше трех символов или использовать знак подчеркивания.

Поле операции содержит название блок или команды. GPSS-интерпретатор распознает команд по первым четырем символам в их напи-

сании, и поэтому достаточно вместо полного имени команды указать лишь четыре первые ее буквы.

Большинство блоков и команд могут иметь один или несколько полей операндов. Эти поля зависят от конкретного ключевого слова, некоторые из них строго обязательны, другие нет. Допустимый синтаксис того или иного ключевого слова можно найти в справочной системе. Как правило, операнды отделяются друг от друга запятыми.

Язык GPSS предоставляет ряд средств для внесения комментариев в программу. Во-первых, наличие звездочки "*" в первой позиции строки указывает на то, что строка содержит только комментарии и не просматривается транслятором. Во-вторых, появление точки с запятой ";" в строке означает начало комментариев.

3.7. Язык PLUS

Для увеличения алгоритмических возможностей в среду моделирования GPSS World встроен язык низкого уровня PLUS. Рассмотрим основные элементы языка PLUS.

3.7.1. Алфавит

Язык PLUS содержит алфавитно-цифровые (прописные буквы A-Z, строчные буквы a-z, цифры 0-9 и знак подчеркивания “_”) и специальные символы (“#”,“*”,”&”,”+”,”-”,”/”,”\”,”,”,”;”). В комментария допускается применение символов русского алфавита “А-Я”.

3.7.2. Имена

Имена – это созданные пользователем последовательности символов, служащие для обозначения объектов, переменных и программ. Имя должно начинаться с буквы, в нем можно использовать от 1 до 250 алфавитно-цифровых символов. Имена не должны совпадать с ключевыми словами GPSS и с СЧА. Регистр алфавита GPSS World не различает.

3.7.3. Выражения

PLUS-выражение – это комбинация одного или нескольких элементов, называемых факторами. Выражения строятся с использованием опера-

торов и вызовов процедур, обрабатывающих факторы. Выражения применяются в PLUS-процедурах и в операндах операторов GPSS. Если выражение используется в операндах GPSS-блоков, то оно должно записываться в круглых скобках.

3.7.4. Типы данных

Переменные пользователя, элементы матриц, ячейки, параметры транзактов могут иметь значения различных типов данных, причем сначала каждый из них имеет неопределенную форму, которая исключает их использование до того, как им будет присвоено первое значение. В GPSS World используется три основных типа данных:

— Целочисленный (Integer) – это 32-х битные двоичные числа. Если при выполнении арифметической операции происходит переполнение целочисленного значения, производится автоматическое его преобразование в вещественное;

— Вещественный (Real) – это числа с плавающей точкой двойной точности. Для представления мантиссы используется 15 десятичных разрядов, а для порядка – значения в интервале от –306 до 306.

— Строковый тип - это последовательность символов ASCII . Строки не ограничены в размерах; они ограничиваются лишь максимальным запросом памяти, заданным для данного процесса на странице “Simulation” в журнале настроек модели.

3.7.5. Факторы

Факторы – основные элементы выражений. Факторами выражений GPSS World могут быть:

1) Строковые константы;
2) Вещественные константы;
3) Целочисленные константы;
4) Имена;
5) Элементы PLUS-матрицы;
6) Обращения к процедурам;
7) Стандартные числовые атрибуты.

Для объединения факторов в выражение используются операторы и вызовы процедур, это производится в соответствии с правилами, связанными с конкретным оператором или процедурой.

3.7.6. Операторы

В табл. перечислены арифметические операторы, используемые в выражениях GPSS World, в порядке убывания их приоритетов.

Таблица 3.1

Оператор	Действие	Результат	Арность	Ассоциативность
-	Отрицание	Аддитивная инверсия	Унарный	Право
^	Возведение в степень	Арифметический показатель степени	Бинарный	Право
NOT или ~	Инверсия	1 (TRUE) или 0 (FALSE)	Унарный	Право
AND	Логическое И	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
OR	Логическое ИЛИ	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
G или >	Больше	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
L или <	Меньше	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
E или =	Равно	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
NE или /=	Не равно	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
LE или <=	Меньше или равно	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
GE или <=	Больше или равно	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
# (или *)	Умножение	Арифметическое произведение	Бинарный	Лево
/	Деление	Арифметическое частное	Бинарный	Лево
\	Целочисленное деление	Целочисленное частное	Бинарный	Лево
@	Деление по модулю	Целочисленный остаток	Бинарный	Лево
+	Сложение	Арифметическая сумма	Бинарный	Лево
-	Вычитание	Арифметическая разность	Бинарный	Лево

Все арифметические операторы автоматически преобразуют строковые операнды в числовые значения.

К операторам языка PLUS также относятся:

— PROCEDURE – определяет PLUS-процедуру;

— EXPERIMENT – определяет PLUS-эксперимент;

— TEMPORARY – определяет и ограничивает область действия локальных переменных пользователя и локальных матриц;

— BEGIN/END – составной оператор (создает блок PLUS-операторов);

— Присваивание – устанавливает значение именованного значения элемента матрицы (Lvalue=Expression);

— Вызов процедуры – вызывает библиотечную процедуру;

— Оператор с меткой - суперкласс операторов, начинающихся с метки;

— IF/THEN – проверяет выражение, и, если результат равен “TRUE” (истина), выполняет действие;

— IF/THEN/ELSE – проверяет выражение и в зависимости от результат производит то или иное действие;

— WHILE/DO – несколько раз выполняет действие;

— GOTO – передает управление к метке внутри процедуры;

— RETURN – останавливает выполнение процедуры и может возвращать результат ее выполнения.

Каждый PLUS-оператор должен заканчиваться разделителем “;”.

Если PLUS-процедура используется более, чем в одной модели, то ее можно сохранить в исходном файле, называемом библиотекой процедур пользователя. Командой INCLUDE можно включать эту библиотеку в каждую модель, в которой будет использоваться данная процедура.

Оператор EXPERIMENT предназначен для определения специальной процедуры пользователя, которая применяется для управления несколькими повторяющимися процессами моделирования. Эксперименты обычно используются совместно с библиотечной процедурой DoCommand() для управления процессом моделирования, а также с библиотечной процедурой дисперсионного анализа ANOVA для анализа результатов моделирования. Обычно эксперимент заполняет результатами выполнения глобальную матрицу, которая передается процедуре ANOVA. Существует возможность автоматически создавать отсеивающие и оптимизирующие эксперименты при помощи генераторов экспериментов GPSS World. Эксперимент вызывается только командой CONDUCT.

3.7.7. Встроенная библиотека процедур

Встроенная библиотека содержит процедуры, которые могут быть разбиты на следующие группы: обслуживающие процедуры, файловые процедуры, процедуры динамического вызова, математические процедуры, вероятностные распределения, строковые процедуры, процедуры запроса.

Обслуживающие процедуры (процедуры управления прогонами имитации и анализа экспериментов):

— DoCommand – транслирует строку аргумента и передает результат трансляции выполняющейся модели. Эта процедура может вызываться толь ко из экспериментов и процедур, вызванных из экспериментов;

— ANOVA – производит многофакторный дисперсионный анализ;

— Exit – завершает сеанс работы с GPSS World (останавливает процесс моделирования и закрывает все окна GPSS World).

Файловые процедуры (процедуры потоков данных) – процедуры управления потоками данных внутри PLUS-процедуры. Потоки данных используются для чтения и записи файлов или для хранения и доступа больших объемов данных в памяти. Файловые процедуры выполняют следующие операции:

Open(DataStream, FileNameString) – инициализирует поток данных;

Close(DataStream) – закрывает поток данных и извлекает код ошибки;

Read(DataStream) – считывает текстовую строку из потока данных;

Write(DataStream) – передает потоку данных текстовую строку;

Seek(DataStream, NewLinePosition) – устанавливает позицию текущей строки потока данных и извлекает предыдущую позицию строки.

Процедуры динамического вызова используются для вызова функций, хранящихся во внешних исполняемых файлах, включая динамически подключаемые библиотеки DLL. Соответствующие процедуры можно использовать для запуска функций, поддерживающих протокол CDECL. Процедурами динамического вызова являются:

— ReturnCode = Call(ExecutableFileName, FunctionName) – вызывает внешнюю CDECL-функцию без аргументов (т.е. с пустым аргументом);

— ReturnCode = Call_Integer(ExecutableFileName, FunctionName,Argument) – вызывает внешнюю CDECL-функцию с одним целочисленным аргументом;

— ReturnCode = Call_String(ExecutableFileName, FunctionName,Argument) – вызывает внешнюю CDECL-функцию с одним символьным аргументом (char *);

— ReturnCode = Call_Real(ExecutableFileName, FunctionName,Argument) – вызывает внешнюю CDECL-функцию с одним вещественным аргументом (double).

Первый аргумент процедуры динамического вызова является строкой, указывающей имя исполняемого модуля (DLL или EXE-файла), содержащего вызываемую функцию. При задании пути доступа используют обратную косую черту “\”. По умолчанию расширение DLL; если последний символ – “.”, то имя модуля не имеет расширения.

Математические процедуры:

— ABS(Expression) – абсолютное значение;

— ATN(Expression) – арктангенс (в радианах);

— COS(Expression) – косинус;

— EXP(Expression) – экспонента, возведенная в степень;

— INT(Expression) – выделение целой части с отбрасывание дробной части;

— LOG(Expression) – натуральный логарифм;

— SIN(Expression) – синус;

— SQR(Expression) – квадратный корень;

— TAN(Expression) – тангенс.

Строковые процедуры выполняют операции со строками:

— Align(InsertString, SourceString, Offset) - вставляет одну строку в другу с выравнивание по правому краю;

— Catenate(String1, String2) – производит объединение строк;

— Copies(SourceString, Count) – создает одну строку из нескольких копий исходной;

— DataType(Datum) – возвращает строку, содержащей тип данных аргумента;

— Find(TestString, SourceString) – вычисляет смещение одной строки содержащейся в другой строке;

— Left(SourceString,MaxCount) – возвращает левую часть заданной строки;

— Length(SourceString) – возвращает количество символов в строке;

— LowerCase(SourceString) – переводит все прописные буквы в строчные;

— Place(InsertString, SourceString, Offset) – вставляет одну строку в другую с выравниванием по левому краю;

— PolyCatenate(String1, String2, …) – производит объединение двух и более строк;

— Right(SourceString, MaxCount) - возвращает правую часть заданной строки;

— String(Datum) – преобразует данные в их строковый эквивалент;

— StringCompare(String1,String2) – сравнивает две строки;

— SubString(SourceString, Offset, MaxCount) – возвращает подстроку заданной строки;

— Trim(SourceString) – удаляет начальные и конечные пробелы;

— UpperCase(SourceString) –переводит все строчные буквы в строке в прописные;

— Value(Datum) – возвращает числовой эквивалент строки;

— Word(SourceString, WordNumber) – возвращает заданное слово строки.

Процедуры запроса возвращают информацию о состоянии транзакта:

— QueryXNExist(TransactionNumber) - возвращает 1, если транзакт существует в модели, иначе – 0;

— QueryXNParameter(TransactionNumber,Parameter) – возвращает значение параметра транзакта. Если искомого параметра не существует, происходит останов по ошибке;

— QueryXNAssemblySet(TransactionNumber) – возвращает номер семейства, к которому принадлежит транзакт;

— QueryXNPriority(TransactionNumber) – возвращает приоритет транзакта в виде целого числа;

— QueryXNM1(TransactionNumber) – возвращает время входа транзакта в систему.

3.8. Управление моделированием

Команда START управляет процессом моделирования. Эта команда задает условие счета по модели. Так операнд A задает начальное значение счетчика завершений. Для уменьшения счетчика завершений используются блоки TERMINATE. При достижении счетчиком завершений нуля или отрицательного значения процесс моделирования завершается.

Команда START также показывает транслятору условия вывода окончательных значений счета по модели.

Команда START записывается в следующей форме:

START A, B, C, D

где A - число, заносимое в счетчик числа завершений;

B - подавление вывода статистики моделирования в стандартного форме (подавление вывода - NP (No Printing);

C - не используется, поддерживается для совместимости с предыдущими версиями;

D - вывод списков, если D=1, то отчет будет содержать списки текущих и будущих событий, а также списки пользователя.

При выполнении команды start устанавливается счетчик завершений и во все блоки GENERATE, помеченные как "не запущенные", вводится один транзакт.

Например, в счетчик числа завершений необходимо занести число 2 - соответствующая команда имеет вид:

START 2

3.9. Моделирование входящих и выходящих потоков. Блоки создания и уничтожения транзактов

Каждый блок имеет входы и выходы, с помощью которых осуществляется их связь в модели. Существует два особых блока: GENERATE, имеющий только выход, и TERMINATE, имеющий только вход. Через блок GENERATE транзакты вводятся в модель. Блок TERMINATE удаляет транзакты из модели.

Когда транзакт входит через блок GENERATE, транслятор планирует время прихода следующего транзакта разыгрыванием случайного числа в соответствии с распределением интервала поступления и прибавляет его к текущему значению таймера. При достижении этого времени следующий транзакт вводится в модель и т.д. Транзакты не могут входить в блок GENERATE, так как он сам их генерирует. Блок имеет следующие операнды:

GENERATE A, B, C, D, E

где A - среднее значение интервала времени между приходом предыдущего и последующего

транзакта;

B - модификатор, может быть двух типов: модификатор-интервал и модификатор-

функция. Модификатор-интервал используется, когда интервалы времени между поступлениями транзактов распределено равномерно в некотором заданном диапазоне. Указывается половина поля допуска равно-

мерно распределенного интервала (транзакты генерируются с периодичностью A-B...A+B единиц модельного времени). Модификатор-функция используется, если время задержки транзактов распределено неравномерно (более сложным образом). Интервал времени генерации транзакта определяется произведением значения операнда A на значение функции, заданной операндом B;

C - время появление первого транзакта, т.е. смещение интервалов генерации транзактов

(по умолчанию смещение отсутствует). Наличие этого операнда позволяет ввести транзакт в модель в любой необходимый разработчику момент модельного времени;

D - общее число генерируемых транзактов (по умолчанию - бесконечность). Позволяет

ограничить число транзактов, генерируемых блоком;

E - Уровень приоритета транзакта. При соперничестве между транзактами в занятии

блока преимущество получает транзакт, первым подошедшим к данному блоку (FIFO

- первым пришел - первым обслужен), в определенных случаях очередность обработки транзактов может быть изменена в зависимости от уровня приоритета (первым обслуживается транзакт, имеющий более высокий приоритет, среди транзактов с равным приоритетом первым обслуживается тот, который пришел первым).

Например, необходимо сымитировать входящий поток из 30 транзактов так, чтобы первый пришел спустя 32 единицы модельного времени после начала моделирования, а последующие равномерно с интервалом от 7 до 13 единиц модельного времени (10-3...10+3). Блок GENERATE будет выглядеть следующим образом:

GENERATE 10,3,32,30

Следующий пример создает три транзакта в момент времени 0:

GENERATE ,,,3

Для удаления транзактов из модели используется блок TERMINATE. Этот блок удаляет активный транзакт из процесса моделирования и может уменьшать счетчик завершений. Вид блока:

TERMINATE A

где A - величина уменьшения счетчика завершений (по умолчанию A=0).

В процессе моделирования транзакты попадают в блоки TERMINATE, и содержимое счетчика завершений уменьшается на значение операнда А. При достижении счетчиком завершений нулевого или отрицательного значения процесс моделирования завершается.

Например, необходимо окончить моделирование по прохождении 10 транзактов через блок TERMINATE:

```
TERMINATE 1
START 10
```

Данный процесс можно смоделировать и так:

```
TERMINATE 5
START 50
```

Например, необходимо построить модель таки образом, чтобы процесс моделирования завершился после прохождения через блок TERMINATE с меткой MET3 десяти транзактов. При возникновении особо оговоренной ситуации (транзакты поступают на блок с меткой MET2) моделирование завершить сразу же. Транзакты, поступившие на блок с меткой MET1, выводить из модели без изменения счетчика завершения.

```
            ...
MET1 TERMINATE
            ...
MET2 TERMINATE 10
            ...
MET3 TERMINATE 1
            ...
START 10
```

3.10. Сегментация модели. Сегмент ограничения модельного времени

Программа GPSS состоит из одного или нескольких сегментов, каждый из которых имеет свою функцию. Сегменты осуществляют свою работу параллельно, т.е. транзакты по своим сегментам продвигаются вне зависимости от движения транзактов по другим сегментам.

Пример. Моделируется процесс валки деревьев. Стоит задача завершить моделирование по завершении валки 1000 деревьев. Такая модель будет состоять из одного сегмента:

```
...
TERMINATE 1
START 1000
```

Если же поставлена задача завершить моделирование через какой-то период времени, то модель будет состоять из двух сегментов: сегмента имитации процесса валки и сегмента ограничения модельного времени. Более сложные процессы моделируются большим число сегментов.

Пример. За единицу модельного времени принята 1 минута. Организовать модель так, чтобы моделирование завершилось по истечении 7 часов (7*60=420 минут). Один из вариантов ограничения модельного времени имеет вид:

```
...
* Сегмент ограничения модельного времени
GENERATE 420
TERMINATE 1
START 1
```

Через 420 единиц модельного времени генерируется транзакт, который, пройдя в блок TERMINATE за значением операнда A, равным единице, завершит моделирование (при значении операнда A команды START, равном единице).

Эту задачу можно решить и иначе, сгенерировав 420 транзактов, входящих через 1 единицу модельного времени:

```
        ...
        GENERATE 1
        TERMINATE 1
    START 420
```

При этом блок GENERATE генерирует последовательность транзактов с периодичностью 1 единица модельного времени. Каждый из этих транзактов уменьшает на единицу счетчик числа завершений, и при достижении им нуля, т.е. спустя 420 единиц модельного времени процесс моделирования завершается. Следует отметить, что второй способ менее эффективен с точки зрения производительности, так как создается большое количество объектов.

3.11. Моделирование приборов

Статические элементы системы массового обслуживания моделируются специальными объектами GPSS - устройствами (приборами). Попытка занятия устройства моделируется блоком SEIZE. Если устройство занято, то транзакт задерживается в предыдущем блоке; если прибор свободен, то транзакт занимает его. Устройство находится в занятом состоянии в течении времени обслуживания. По окончании обслуживания происходит освобождение прибора блоком RELEASE. Транзакт может занять любое число приборов.

SEIZE A

RELEASE A

где A - номер или имя устройства.

Освобождать устройство должен тот же транзакт, который его занимал.

3.12. Имитация обработки транзакта

Блок ADVANCE моделирует задержку транзакта в течение определенного периода времени. Этот блок задает среднее время выполнения оперрации в моделируемой системем, а также разброс времени относительно среднего:

ADVANCE A, B

где A - среднее время задержки;

B - временной полуинтервал или модификатор функции.

Блок ADVANCE не препятствует входу транзактов, и любое их количество может находиться в нем одновременно. Вновь поступающий транзакт никак не влияет на уже находящиеся в блоке.

Пример 1. Смоделировать задержку всех проходящих через блок транзактов на 20.5±10.3 единиц модельного времени.

ADVANCE 20.5,10.3

Пример 2. Предположим, что прибор обрабатывает транзакт в течение 10...90 единиц модельного времени.

...

SEIZE Pribor

ADVANCE 50,40

RELEASE Pribor

...

Пример 3. Предположим, что челюстному погрузчику для погрузи одного автолесовоза требуется от 10 до 16 минут. Смоделируем процесс погрузки автолесовозов, если известно, что лесовозы прибывают под погрузку через промежутки времени, равномерно рапределенные в интервале 16...24 минут, и если погрузчик занят, то лесовоз ожидает его освобождения. В модели за транзакт принимаем автолесовоз, работу погрузчика будем моделировать устройством POGR, в качестве единицы модельного времени примем одну минуту. Рассмотрим работу погрузчика за 10 часов работы. Данный процесс целесообразно моделировать двумя сегментами: сегментом имитации работы погрузчика и сегментом ограничения модельного времени:

```
* Сегмент имитации работы погрузчика
GENERATE 20,4
SEIZE POGR
ADVANCE 13,3
RELEASE POGR
TERMINATE
* Сегмент ограничения модельного времени
GENERATE 600
TERMINATE 1
START 1
```

Результаты моделирования можно рассматривать следующим образом: число нагруженных автолесовозов - число обслуживаний прибора; коэффициент использования погрузчика - коэффициент использования прибора:

```
FACILITY      ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER
RETRY DELAY
POGR                 29   0.619      12.800      1          0      0
0      0      0
```

3.13. Моделирование отказа устройства

Чтобы занять устройство, транзакты, как правило, ждут своей очереди. Однако в некоторых системах вновь прибывающему транзакту разрешается заместить транзакт на занятом устройстве. В этом случае устройство захватывается у еще не закончившего обслуживание транзакта, а вновь

приходящему разрешается немедленно занять устройство. Замещенный транзакт ждет освобождения устройства. Ситуация прерывания работы устройства часто встречается в системах массового обслуживания с ненадежными каналами и заданным временем наработки на отказ.

Захват и возвращение устройства моделируется блока PREEMPT и RETURN соответственно.

PREEMPT A, B, C, D, E

где A - имя или номер устройства;

B - приоритетный режим. PR - приоритетный режим (более приоритетный транзакт может вытеснить транзакт, занимающий устройство, если претендент имеет недостаточный приоритет, он помещается в список задержки в приоритетном порядке), если операнд опущен, то будет реализован режим прерывания;

C - имя или номер блока - нового места назначения для транзакта, в данный момент занимающего устройство;

D - номер параметра вытесненного транзакта, в который записывается оставшееся время обслуживания;

E - Режим удаления. RE - удаляет вытесненный транзакт из состязания за устройство.

RETURN A

где A - номер или имя устройства.

Пример 1. Смоделировать возможность поломки прибора. Время наработки на отказ равномерно распределяется на интервале от 100 до 200 единиц модельного времени, время ремонта - 10 единиц модельного времени.

```
...
GENERATE 150,50
PREEMPT Pribor
ADVANCE 10
RETURN Pribor
TERMINATE
...
```

Пример 2. Смоделируем работу окорочного станка, если известно, что на окорку одного хлыста требуется от 4 до 6 минут. Наработка на отказ составляет 10...30 часов, время ремонта - 20...120 минут. Моделирование произведем исходя из 1000 часов времени работы. Примем также условие, что входной буфер (пачка хлыстов перед станком) бесконечен, т.е. нет ограничения по входному параметру.

При моделировании принимаем за единицу времени 1 минуту.

Модель целесообразно представить из трех сегментов:

- сегмента имитации работы окорочного станка;
- сегмента имитации отказа в работе;
- сегмента ограничения модельного времени.

```
*Сегмент имитации работы окорочного станка
GENERATE 5
SEIZE stanok
ADVANCE 5,1    ; окорка хлыста
RELEASE stanok
TERMINATE
*Сегмент имитации отказа в работе
GENERATE 1200,600   ; наработка на отказ
PREEMPT stanok
ADVANCE 70,50    ; ремонт
RETURN stanok
TERMINATE
*Сегмент ограничения модельного времени
GENERATE 60000
TERMINATE 1
START 1
```

Число обслуженных станком транзактов есть число прошедших через окорочный станок хлыстов (11303), число прошедших через блок RETURN транзактов (52)- число остановок в работе:

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	11999	695	0
	2	SEIZE	11304	0	0
	3	ADVANCE	11304	1	0
	4	RELEASE	11303	0	0
	5	TERMINATE	11303	0	0
	6	GENERATE	52	0	0
	7	PREEMPT	52	0	0
	8	ADVANCE	52	0	0
	9	RETURN	52	0	0
	10	TERMINATE	52	0	0
	11	GENERATE	1	0	0
	12	TERMINATE	1	0	0

3.14. Моделирование многоканального устройства (памяти)

Часто необходимо смоделировать работу нескольких однородных устройств, работающих параллельно. Для этого язык GPSS предоставляет специальный объект - память. Память служит для моделирования объектов, обладающих определенной емкостью. Например, если в салоне работают три парикмахера - то салон можно рассматривать как память емкостью 3.

Емкость памяти задают с помощью оператора STORAGE:

Имя_памяти STORAGE A

где A - общая емкость памяти.

Занятие памяти транзактом происходит в блоке ENTER, а освобождение в блоке LEAVE:

ENTER A, B

LEAVE A, B

где A - имя или номер памяти;

B - число занимаемых единиц памяти (по умолчанию B=1).

Если емкости памяти не хватает для удовлетворения потребности транзакта, то транзакт помещается в очередь в порядке приоритета.

Пример 1. Смоделировать работу памяти емкостью 130, время обработки транзакта в памяти - 100 единиц модельного времени. Значение емкости памяти, необходимой для обработки, находится в первом параметре транзакта.

```
...
SKLAD STORAGE 130
...
ENTER SKLAD,P1
ADVANCE 100
LEAVE SKLAD,P1
...
```

При этом до 130 транзактов единовременно могут находиться в памяти, при условии что значение первого параметра транзакта P1 равно 1, каждый из них задерживается на 100 единиц модельного времени.

3.15. Моделирование очередей

Для сбора статистики об очередях (общее и среднее число транзактов, находящихся в очереди, среднее время ожидания и пр.) в местах задержки транзактов ставятся блоки QUEUE и DEPART, имитирующих события "присоединение к очереди" и "уход из очереди" соответственно.

QUEUE A, B

DEPART A, B

где A - имя или номер очереди;

B - число элементов, на которое изменяется значение счетчика содержимого очереди (по умолчанию - B=1).

Эти блоки сами по себе не создают очередь, а лишь являются средством ее регистрации.

Например, необходимо собрать статистику по ожиданию транзактов в очереди перед прибором:

```
...
QUEUE OCHERED
SEIZE PRIBOR
DEPART OCHERED
...
RELEASE PRIBOR
...
```

3.16. Параметры транзактов

При входе транзакта в модель все его параметры равны нулю. Для назначения и изменения параметров транзактов используется блок ASSIGN.

ASSIGN A, B, C

где A - имя или номер изменяемого параметра активного транзакта; если за операндом A следует знак "+" или "-" то значение операнда B соответственно добавляется или вычитается из A. Иначе значение B становится текущим значением параметра транзакта;

B - значение;

C - модификатор функции.

Приведем примеры. Третьему параметру активного транзакта присвоить число 5:

ASSIGN 3,5

Увеличивать на единицу второй параметр активного транзакта:

ASSIGN 2+,1

Присвоить параметру активного транзакта с именем TEXT строку:

ASSIGN TEXT, "Hello, World"

При необходимости получить значение параметра активного транзакта в других командах и блоках GPSS (если значение операнда не определяется именем или номером параметра) можно с помощью СЧА P*parameter* (*Parameter* - номер параметра транзакта) или P$*Name* (Name - имя)

3.17. Работа с ячейками и матрицами

Для обращения в процессе моделирования к сохраняемым значениям используют ячейки и матрицы. Занесение информации в ячейку осуществляется блоком SAVEVALUE:

SAVEVALUE A, B

где A - номер или имя ячейки. Если за операндом A стоит знак "+" или "-", значение

операнда B соответственно прибавляется или вычитается из текущего содержимого

ячейки. Если знак не указан, то значение операнда B записывается в ячейку;

B - значение.

SAVEVALUE Account, 10.5

В этом примере в ячейку с именем Account будет записано значение 10.5.

SAVEVALUE TEXT,"Hello, World!"

В этом примере в ячейку с именем TEXT записывается строка.

Для получения значения ячейки в других командах и блоках GPSS используют СЧА X*Entnum* (*Entnum* - номер ячейки) или X$*Name* (Name - имя).

Матрица определяется командой MATRIX.

Имя MATRIX A, B, C, D, E, F, G

где A - неиспользуемое поле (для совместимости с ранними версиями GPSS);

B - Количество элементов в первом измерении (количество строк);

C - Количество элементов во втором измерении (количество столбцов);

D - Количество элементов в третьем измерении;

E - Количество элементов в четвертом измерении;

F - Количество элементов в пятом измерении;

G - Количество элементов в шестом измерении.

Massiv MATRIX ,5,100

Это команда определяет матрицу с именем Massiv с 5 строками и 100 столбцами.

Изменение элементов матрицы происходит в блоке MSAVEVALUE.

MSAVEVALUE A, B, C, D

где A - имя или номер матрицы. если за операндом A следует знак "+" или "-" то значение

операнда D соответственно добавляется или вычитается из ячейки матрицы A. Иначе значение D становится текущим значением элемента матрицы;

B - номер строки;

C - номер столбца;

D - значение.

MSAVEVALUE Massiv+,3,70,6.2

В этом примере, когда транзакт входит в блок MSAVEVALUE, элемент матрицы с именем Massiv, находящийся на пересечении 3 строки и 70 столбца, увеличивается на 6.2.

Доступ к третьему и более высокому измерению получают через PLUS-операторы.

Для получения значения элемента матрицы в других командах и блоках GPSS используют СЧА MX*Entnum(m,n)* (*Entnum(m,n)* - ячейка матрицы) или MX$*Name(m,n)* (Name - имя).

При моделировании начальные значения ячеек и элементов матриц устанавливаются равными нулю. Оператором INITIAL можно задать желаемые значения.

INITIAL A, B

где A - логический ключ (LS*Ключ* или LS$*Name* (Name - имя)), ячейка (X*Entnum* или X$*Name*

(Name - имя)) или элемент матрицы (MX*Entnum(m,n)* или X$*Name(m,n)*), определенные как СЧА, или имя матрицы. Если операнд A задает имя матрицы, то все ее элемнтам присваивается значение указанное операндом B;

B - значение или "unspecified" (не определено). По умолчанию равно 1.

3.18. Переменные

В GPSS при моделировании допустимо использование арифметических и логических переменных.

В выражениях арифметические переменные используют такие арифметические операции:

+ - алгебраическое сложение;

- - алгебраическое вычитание;

- алгебраическое умножение;

/ - деление;

\ - целочисленное деление;

@ - деление по модулю;

^ - возведение в степень.

При использовании в переменной правил булевой алгебры могут использовать следующие обозначения:

= 'E' - равно;

< 'L' - меньше;

> 'G' - больше;

>= 'GE' - больше или равно;

<= 'LE' - меньше или равно;

!= 'NE' - не равно;

& 'AND' - логическое И;

| 'OR' - логическое ИЛИ.

Арифметическая переменная задается оператором VARIABLE:

Имя VARIABLE X

X - Выражение.

Var1 VARIABLE P1#X$Cell

Эта команда определяет переменную, которая вычисляется, когда встречается V$Var1.

Команда BVARIABLE определяет булевы переменную

Имя BVARIABLE X

где X - выражение.

При обращении к булевой переменной используют СЧА BV$Имя_переменной.

3.19. Генераторы случайных чисел

Алгоритм генерации псевдослучайных чисел в GPSS World основан на мультипликативно-конгруэнтном алгоритме Лемера с максимальным периодом. Алгоритм генерирует псевдослучайное число в открытом интервале от 0 до 2147483647 и до самоповторения генерирует 2147483646 уникальных псевдослучайных чисел. Дополнительно используется шаг перемешивания. СЧА класса RN возвращает число от 0 до 999 включительно, а при вычислении случайных функций используется случайное число на интервал (0,1). Начальное число равно номеру объекта генератора случайных чисел, пока не будет изменено оператором RMULT. Например, RN2 запускается с начальным числом 2.

3.20. Функции

Для получения чисел с распределением, отличным от равномерного используются специальные объекты GPSS- функции. Функцию задают набором пар точек - координат.

Имя FUNCTION A, Bn

Где A - аргумент функции;

Bn - тип функции, n - число точек.

Далее, в следующей строке, начиная с первой позиции поля экрана, записываются координаты точек (x_i,y_i), причем обязательно в последовательности $x_1<x_2<...<x_n$:

$x_1,y_1/x_2,y_2/.../x_i,y_i$

$x_{i+1},y_{i+1}/x_{i+2},y_{i+2}/.../x_k,y_k$

...

$x_{m+1},y_{m+1}/x_{m+2},y_{m+2}/.../x_n,y_n$

Существует 5 различных типов функций.

Тип C - непрерывная функция. Воспроизводится в виде ломаной кривой, отрезки которой соединяют соседние точечные значения. Получая значение X, после линейной интерполяции функция возвращает значение Y.

Output FUNCTION V$Input, C3

1.1,10.1/20.5,98.7/33.3,889.2

Если V$Input равно 1 или меньше, то FN$Output возвращает 10.1, если V$Input равно 25,то после линейной интерполяции FN$Output возвращает 376,6101563.

Тип D - дискретные функции. Дискретная функция имеет вид ступенчатой кривой. Каждому значению аргумента присваивается числовое значение. Дискретная функция со случайным аргументом используется для определения дискретного вероятностного распределения.

Пример 1. Смоделировать входящий поток транзактов, распределение интервалов времени появления которых приведено в таблице:

Таблица 3.2

Значения случайной переменной X	Относительная частота	Диапазон	Значения функции распределения Y
2	0.15	0.00...0.15	0.15
5	0.20	0.15...0.35	0.35
8	0.25	0.35...0.60	0.60
9	0.22	0.60...0.82	0.82
12	0.18	0.82...1.00	1.00

```
DRAS FUNCTION RN3,D5
.15,2/.35,5/.60,8/.82,9/1,12
GENERATE FN$DRAS
...
```

Тип E - дискретные атрибутивно-значимые функции. Для вычисления каждому значению аргумента назначается определенный СЧА. Функции типа E вычисляются тем же способом, что и функции типа D.

```
Edisc FUNCTION X$QRA, E5
1,S$Stor1/3,S$Stor2/5,s$Stor3/9,S$Stor5/10,S$Stor6
```

Тип L - списковая функция. Значение аргумента используется для определения позиции возвращаемого значения.

```
Listtype FUNCTION Q$Barber,L5
1,PAR1/2,PAR2/3,PAR3/4,PAR4/5,PAR5/
```

Тип M - списковая атрибутивно-значимая функция. Значение аргумента используется для определения положения в списке СЧА, который должен быть вычислен и возвращен как результат функции.

```
Mlist FUNCTION X$Name1,M5
1,Q$Nam1/2,Q$NamX/3,Q$Nam4/4,Q$Nam6/5,F$Tan1/
```

Пример 2. Модель сортировочного транспортера. Известно, что сортименты на сортировочный транспортер поступают с периодичностью 5±3 минуты. В зависимости от породы и длины они сортируются по накопительным карманам. Условно примем, что сортировка производится исходя из трех видов пород (порода номер 1, порода номер 2, порода номер 3) и трех видов длин (длина 1, длина 2, длина 3) в соответствии с табл. 3.2.

Таблица 3.3

Номер накопителя	Порода (П)	Длина (Д)	Формула (Д*3+П)
1	1	1	4
1	2	1	5
2	3	1	6
3	1	2	7
4	2	2	8
5	3	2	9
6	1	3	10
6	2	3	11
7	3	3	12

Процентные соотношения встречающихся сортиментов представлены в табл. 3.4 и табл. 3.5.

Таблица 3.4

Порода	Процент	Значение функции распределения
1	20	0,2
2	50	0,7
3	30	1,0

Таблица 3.5

Длина	Процент	Значение функции распределения
1	40	0,4
2	40	0,8
3	20	1,0

Время движения сортиментов по транспортеру до своего накопителя:

Таблица 3.6

Накопитель	1	2	3	4	5	6	7
Время движения, мин	2	3	5	6	8	10	12

Смоделировать процесс сортировки сортиментов в течение 100 рабочих часов.

Программа на GPSS будет иметь следующий вид:

```
*Определение функций
POR FUNCTION RN1,D3        ;задание породы
.2,1/.7,2/1,3
DL FUNCTION RN2,D3         ;задание длины
.4,1/.8,2/1,3
NAK FUNCTION V$FORM,D7     ;задание накопителя
5,1/6,2/7,3/8,4/9,5/11,6/12,7
T_MOV FUNCTION P$NAK,D7    ;время движения
1,2/2,3/3,4/4,5/5,6/6,7/7,8
FORM VARIABLE FN$DL#3+FN$POR  ;формула порода-длина
*Сегмент имитации работы сортировочного транспортера
  GENERATE 5,3                ;поступление сортиментов
  ASSIGN NAK,FN$NAK           ;определение накопителя
  ADVANCE FN$T_MOV            ;движение по транспортеру
  SAVEVALUE COU+,1            ;сортимент в накопителе
  TERMINATE
*Сегмент ограничения модельного времени
  GENERATE 6000
  TERMINATE 1
   START 1
```

Основные результаты прогона модели представлены следующим листингом:

```
LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE          1228            0       0
                     2    ASSIGN            1228            0       0
                     3    ADVANCE           1228            1       0
                     4    SAVEVALUE         1227            0       0
                     5    TERMINATE         1227            0       0
                     6    GENERATE             1            0       0
                     7    TERMINATE            1            0       0


SAVEVALUE                RETRY       VALUE
 COU                       0       1227.000
```

3.21. Переадресация потока транзактов

Поток транзактов обычно проходит блоки последовательно, поэтому предполагается, что, выполнив блок с номером n, транзакт переходит к выполнению блока с номером n+1. Однако в моделируемой системе в зависимости от создавшихся условий поток моделируемых объектов может быть задержан или направление его движения изменено. Для моделирования таких ситуаций используется блок TRANSFER, который обеспечивает переход активного транзакта к новому блоку в зависимости от создавшихся условий.

TRANSFER A, B, C, D

где A - режим передачи;

B - номер или метка блока;

C - номер или местоположение блока;

D - приращение номера блока для режима ALL (по умолчанию - 1).

Режим безусловной передачи. Когда операнд A опущен, транзакт безусловно передается в блок определяемый операндом B.

TRANSFER ,MET1

Режим статистической передачи. Когда операнд A задан десятичной дробью, блок TRANSFER функционирует в режиме статистической передачи. В этом режиме активный транзакт переходит к местоположению, заданному в операнде C, с вероятностью заданной в операнде A. Альтернативное местоположение задается в операнде B, если операнд B пропущен, транзакт переходит к следующему по порядку блоку.

TRANSFER .4,MET1,MET2

Когда транзакт входит в данный блок, он переходит к метке MET2 с вероятностью 0.4 и к метке MET1 с вероятностью 0.6.

Режим BOTH. Когда в операнде A указано BOTH, транзакт последовательно пытается войти в блоки, номера (метки) которых указаны операндами B и C, до тех пор, пока один из них не станет доступным.

Режим ALL. Когда в операнде A указано ALL, транзакт последовательно пытается войти в блоки, определяемые значениями B, B+D, B+2D, B+3D,...,C.

Режим PICK (выборочный). Когда в операнде А указано PICK, местоположение нового блока выбирается случайно с числовом интервале между В и С.

Режим FN (функциональный). Когда в операнде А указано FN, место назначение активного транзакта выбирается путем вычисления функции, заданной в операнде В, с прибавлением к этому значению необязательного приращения, заданного в операнде С.

TRANSFER FN,Func1,5

Режим P (параметрический). Когда в операнде А указано P, активный транзакт переходит к месту назначения, вычисленному, как сумма значения параметра транзакта, указанного в операнде В, и операнда С.

TRANSFER P,Placemarker,1

Режим SBR (подпрограммный). Когда в операнде А указано SBR, активный транзакт переходит к месту назначения, заданному в операнде В, а местоположение блока TRANSFER записывается в параметр транзакта, указанный в операнде С.

TRANSFER SBR,MET1,Placemarker

Для возвращения из подпрограммы, используется блок TRANSFER в параметрическом режиме (см. выше).

Режим SIM (одновременный). Когда в операнде А указано SIM, активный транзакт переходит к одному из двух мест назначения в зависимости от индикатора задержки транзакта.

Пример 1. Модель работы бригады вальщиков леса. Известно, что на спиливание и валку одного дерева вальщику требуется от 3 до 7 минут, на переход от дерева к дереву и на подготовку рабочего места (подготовка путей отхода, срезание кустарника и подобные работы) – от 8 до 16 минут.

Необходимо определить число сваленных бригадой вальщиков (6 человек) деревьев в течение 1 смены (чистое время работы – 5 часов).

Программа на GPSS будет иметь следующий вид:

```
    *Сегмент имитации работы бригады вальщиков
    GENERATE ,,,6     ;ввод в модель 6 транзактов
MET1 ADVANCE 5,2      ;спиливание и валка
    SAVEVALUE COU+,1  ;счетчик числа сваленных деревьев
    ADVANCE 12,4      ;переход от дерева к дереву
    TRANSFER ,MET1
    *Сегмент ограничения модельного времени
    GENERATE 300
```

TERMINATE 1
START 1

В данной модели за единицу модельного времени принимаем одну минуту.

Число транзактов, обращающихся в сегменте имитации работы бригады вальщиков леса, соответствует числу человек в бригаде. Принятие транзакта в качестве моделируемой единицы (вальщика) предполагает некоторую условность: транзакт одновременно имитирует как собственно работающего, так и предмет труда – дерево, т.е., проходя по модели, транзакт испытывает задержку на 5±2 единицы модельного времени, отождествляемую с валкой дерева, а задержка на 12±4 единицы модельного времени отождествляется с характером работы самого вальщика – переход от дерева к дереву, подготовка рабочего места.

3.22. Моделирование с использованием блоков условного перехода

Блок TEST описывает условие, которое проверяется при входе в него транзакта и определяет направление дальнейшего движения транзакта в зависимости от этого условия. При выполнении условия транзакт пропускается в следующий за блоком TEST блок. В случае невыполнения условия транзакт направляется в блок, метка которого указана операндом C, либо при отсутствии операнда C происходит задержка транзакта в блоке TEST до тех пор, пока условие не будет выполнено. Блок TEST записывается в форме:

TEST r A,B,C

где r - оператор отношения, которым задается операция сравнения (E,G,L,GE,LE,NE);

A - проверяемое значение;

B - контрольное значение;

C - номер или метка блока назначения транзакта, если условие сравнения не выполняется.

Пример 1. Если значение первого параметра проходящего через блок TEST транзакта равно 5, то впустить данный транзакт в блок, иначе направить его на блок с меткой MET0.

TEST E P1,5,MET0

Пример 2. Пока значение ячейки Cell меньше 15, задерживать все транзакты.

TEST GE X$Cell,3,15

Для определения состояния объектов в модели используется блок GATE. Блок работает в двух режимах:

– отказа или условного входа;
– перехода или безусловного входа.

При работе в режиме отказа блок GATE не пропускает транзакты, если соответствующий объект не находится в требуемом состоянии. Если же поставленное в блоке условие удовлетворяется, блок разрешает вход транзактов.

Если операндом B указан номер блока, то вместо отказа блок GATE будет посылать транзакт в указанный блок.

GATE r A,B

где r - внутренний операнд, определяет проверяемое условие:

FV - устройство свободно;

I - устройство прервано;

LS - логический ключ включен;

LR - логический ключ выключен;

M - блок MATCH, явно заданный в операнде A, содержит транзакт (связанный с активным транзактом), ожидающий условия синхронизации;

NI - устройство не прервано;

NM - блок MATCH, явно заданный в операнде A, не содержит транзакт (связанный с активным транзактом), ожидающий условия синхронизации;

NU - устройство не используется;

SE - память пуста;

SF - память заполнена;

SNE - память не пуста;

SNF - память не заполнена;

SNV - память не доступна;

SV - память доступна;

U - устройство используется;

A - номер или имя объекта;

B - метка или номер блока, по которой передается транзакт, если условие сравнения не выполняется.

Пример 1. Если устройство Pribor занят, то впустить транзакт в блок GATE, в противном случае транзакт не впускать.

GATE U Pribor

Пример 2. Если память Stor заполнена, то пропустить транзакт через блок GATE, иначе - направить транзакт на блок с меткой MET15.

GATE SF Stor,MET15

3.23. Организация таблиц

Для сбора статистических данных о различных объектах модели и их представления в стандартной табличной форме используются таблицы. Занесение информации в таблицу осуществляется блоком TABULATE в момент входа очередного транзакта в этот блок. При входе транзакта в блок TABULATE интерпретатор записывает в соответствующую таблицу определенную статистическую информацию:

TABULATE A,B

где A - номер или имя таблицы, в которую заносится табулируемая величина в момент входа транзакта в данный блок;

B - весовой коэффициент. Задает число раз, которое табулируемая величина должна быть занесена в таблицу (по умолчанию - 1).

Пример 1. Занести значение в таблицу, номер или имя которой определено значением параметра 4 проходящего через блок TABULATE транзакта.

TABULATE P4

Если есть в модели организованная таблица, GPSS позволяет увидеть ее гистограмму. Для этого надо обратиться к главному меню, его опции Window - Simulation Window - Table Window, затем имя таблицы, гистограмму которой мы желаем увидеть.

Командой TABLE описывается таблицу, которую необходимо получить в процессе моделирования:

Имя TABLE A,B,C,D

где A - аргумент таблицы (имя переменной, значение которой заносится в таблицу);

B - верхняя граница нижнего интервала (первого частотного класса);

C - ширина интервалов таблицы (размер частотных классов);

D - число интервалов (количество частотных классов).

Пример 1. Когда транзакт входит в блок TABULATE, ссылающийся на таблицу TAB1, занести в таблицу значение первого параметра данного транзакта. Таблица строится на следующих интервалах: [-∞...10],[11..30],[31...+ ∞].

Tab1 TABLE P1,10,20,3

Команда QTABLE инициализирует таблицу плотности распределения времени пребывания в очереди.

Имя QTABLE A,B,C,D

где A -имя очереди;

B - верхняя граница нижнего интервала (первого частотного класса);

C - ширина интервалов таблицы (размер частотных классов);

D - число интервалов (количество частотных классов).

Использование команды QTABLE приводит к автоматической регистрации статистики для транзактов, входящих в блоки QUEUE и DEPART.

3.24. Логические ключи

Для имитации управления в GPSS есть специальные двухпозиционные объекты - логические ключи, которые имеют два положения: "включено" и "выключено". Установка и проверка состояния логических ключей осуществляется в процессе моделирования. По умолчанию логический ключ находится в состоянии "выключено". Состояние логического ключа может измениться, если транзакт пройдет через блок LOGIC:

LOGIC r A

где r - логический оператор:

I - переключить;

R - выключить;

S - включить;

A - номер или имя логического ключа.

Пример 1. Сбросить логический ключ Switch, если транзакт пройдет по блоку с меткой MET2.

MET2 LOGIC R Switch

3.25. Моделирование ансамблей событий

В модели может быть несколько семейств, одному из которых обязательно принадлежит транзакт. Семейства используются для обеспечения синхронизации транзактов. Когда транзакт создается блоком GENERATE, он автоматически входит в одно из семейств. При входе транзакта в блок SPLIT формируется множество дополнительных транзактов. При этом первый становится родителем, а дополнительные - его потомками.

SPLIT A,B,C

где A – число создаваемых копий;

B – следующий блок для копий;

C – параметр транзакта для хранения порядкового номера копий.

Нумерация копий осуществляется следующим образом:

— порядковый номер исходного транзакта увеличивается первым;

— порядковый номер первой копии станет на единицу больше, чем у исходного транзакта;

— номера последующих копий также увеличиваются на единицу.

Блок SPLIT создает новые транзакты, имеющие те же значения атрибутов, что и у порождающих транзактов.

Пример 1. Создать 10 копий вошедшего в блок SPLIT транзакт, копии направить в блок с меткой MET1.

SPLIT 10,MET1

Вывод из модели одного или нескольких членов семейства (объединение транзактов, принадлежащих одному семейству) осуществляется в блоке ASSEMBLE:

ASSEMBLE A

где A - число объединяемых транзактов.

Первый транзакт семейства, достигнув блока ASSEMBLE, задерживается в нем до тех пор, пока остальные члены ансамбля не поступят в этот блок. Когда транзакты, число которых указано в операнде A, поступят в этот блок, они будут удалены из модели, а первый прибывший продолжит движение.

Пример 2. После того, как три транзакта одного семейства войдут в блок ASSEMBLE, один (первый) пропустить, остальные (два) вывести из модели.

ASSEMBLE 3

Синхронизация движения двух транзактов осуществляется в блоке MATCH:

MATCH A

где A – имя или номер сопряженного блока.

Первый транзакт, достигнув блока MATCH, задерживается в нем до тех пор, пока другой транзакт одного и того же семейства не достигнет сопряженного блока MATCH, указанного в операнде A.

Пример 3. Смоделировать условие синхронизации двух транзактов.

…

AAA MATCH BBB

…

BBB MATCH AAA

…

Накопление транзактов, являющихся членами одного семейства:

GATHER A

где A – число транзактов, которые должны быть накоплены.

Транзакты одного семейства задерживаются в блоке GATHER до тех пор, пока их число не станет равным значению операнда A. Когда последний транзакт войдет в блок GATHER, все они одновременно выходят из него в том порядке, в котором поступили.

Пример 4. Собрать в блоке GATHER шесть транзактов, после этого данные транзакты пропустить дальше.

GATHER 6

Для изменения семейства активного транзакта используется блок ADOPT:

ADOPT A

где A – семейство.

Пример 5.

ADOPT 2000

Идентификатору семейства активного транзакта присваивается значение 2000. В результате активный транзакт становится членом семейства номер 2000.

3.26. Организация списков пользователя

Размещением активного транзакта в списке пользователей управляет блок LINK:

LINK A,B,C

где A – номер списка пользователя;

B – упорядочивание списка (FIFO – очередь – “первым пришел – первым обслужен”; LIFO – стек – “последним пришел – первым обслужен”; любое другое значение – транзакты помещаются в список пользователя по убыванию этого значения – транзакты с меньшими значениями являются последними в списке, если значение – значение параметра транзакта, то, наоборот, транзакты в списке пользователя устанавливаются в очередь в порядке возрастания значения параметра);

C – местоположение следующего блока. Блок назначения для транзактов, входящих в блок, когда индикатор компоновки списка пользователя находится в выключенном состоянии (индикатор компоновки полезен при использовании списка пользователя , для управления формированием очереди транзактов на использование ресурса. Он включен, когда гипотетический ресурс занят и выключен, когда гипотетический ресурс свободен).

Пример 1. Собрать транзакты в список пользователя List с дисциплиной обслуживания FIFO:

LINK List,FIFO

Возврат транзактов из списка пользователя осуществляется блоком UNLINK:

UNLINK R A,B,C,D,E,F

где R – оператор отношения (по умолчанию – E):

E – чтобы транзакт был исключен из списка, его атрибут, определенный операндом D, должен быть равен контрольному значению, определенному операндом E;

G – чтобы транзакт был исключен из списка, его атрибут, определенный операндом D, должен быть больше контрольного значения, определенного операндом E;

GE – чтобы транзакт был исключен из списка, его атрибут, определенный операндом D, должен быть больше или равен контрольному значению, определенному операндом E;

L – чтобы транзакт был исключен из списка, его атрибут, определенный операндом D, должен быть меньше контрольного значения, определенного операндом E;

LE – чтобы транзакт был исключен из списка, его атрибут, определенный операндом D, должен быть меньше или равен контрольному значению, определенному операндом E;

NE – чтобы транзакт был исключен из списка, его атрибут, определенный операндом D, должен быть не равен контрольному значению, определенному операндом E;

A – номер списка пользователя;

B – номер блока, в который осуществляется вывод транзактов из списка пользователя;

C – максимальное число выводимых транзактов (по умолчанию – все);

D – значение, с которым ведется сравнение, из списка выводятся только те транзакты, у которых соответствующая величина совпадает с заданной величиной (операнд D) или BACK – транзакты выводятся из конца списка;

E – контрольное значение, с которым сравнивается операнд D; если опущен – то сравниваются значения параметров в списке пользователя с тем же параметром проходящего через блок UNLINK транзакта;

F – альтернативный выход, по которому происходит переход проходящего через блок UNLINK транзакта, если вывод заданного числа транзактов невозможен.

Пример 2. При прохождении транзакта через блок UNLINK вывести один транзакт из конца списка пользователя List и направить данный транзакт в блок с меткой MET1.

UNLINK List,MET1,1,BACK

Пример 2. Модель работы группы валочно-пакетирующих машин.

На лесосеке работают ВПМ манипуляторного типа. Смоделировать работу пяти ВПМ за 10 рабочих смен (время смены – 7 часов), если известно, что:

— площадь развертки (площадь леса, доступная манипулятору с одной рабочей позиции) – 63 м^2;

— запас леса на 1 га – 160 м^3;

— средний объем хлыста – 0.25 м^3;

— время на валку-пакетирование одного дерева - экспоненциально распределенная зависимость со средним значением 1 минута;

— время на переезд между рабочими позициями – значение равномерно распределенное в интервале от 3 до 5 минут;

— время работы без поломок – экспоненциально распределенная зависимость со средним значением 10 часов;

— время ремонта – экспоненциально распределенная зависимость со средним значением 25 минут.

Число деревьев, приходящееся на одну рабочую позицию составляет

63*160/0.25/10000=4.032 шт.

В целях повышения производительности все числовые параметры модели приведем к целочисленному типу. При принятии за единицу модельного времени одной секунды общее время моделирования составит

10*7*3600=252000 единиц модельного времени.

Программа на GPSS будет иметь следующий вид:

```
*Сегмент имитации работы ВПМ
     GENERATE ,,,5
     ASSIGN VOLUME,4032   ;среднее число деревьев на сто-
янке
MET1 TEST GE X$POL,1,MET2 ;вероятность поломки
     SAVEVALUE POL-,1
     LINK REMONT,FIFO     ;ВПМ выходит из работы
MET2 ADVANCE (EXPONENTIAL(1,0,60)) ;валка-пакетирование
     SAVEVALUE SUMMA+,1   ;общая сумма деревьев
     ASSIGN VOLUME-,1000  ;число деревьев на стоянке - 1
     TEST L P$VOLUME,1000,MET1     ;проверка выработки
стоянки
     ASSIGN VOLUME+,4032  ;число деревьев на стоянке плюс
                          *остаток от предыдущего
     SAVEVALUE PACHKI+,1      ;число сформированных пачек
     ADVANCE 240,60           ;переезд между стоянками
     TRANSFER ,MET1
*Сегмент имитации отказов в работе
     GENERATE (EXPONENTIAL(1,0,36000)) ;наступление полом-
ки
     SAVEVALUE POL+,1
     ADVANCE (EXPONENTIAL(1,0,1500))      ;время ремонта
     UNLINK REMONT,MET2,1,BACK       ;возврат ВПМ к работе
     TERMINATE
*Сегмент ограничения модельного времени
     GENERATE 252000
     TERMINATE 1
     START 1
```

Результаты работы модели анализируются следующим образом: общее число сваленных деревьев – содержание ячейки (SUMMA) – равно 10420, общее число сформированных пачек – содержание ячейки (PACHKI) – равно 2584.

3.27. Использование приоритета транзакта

Блок BUFFER останавливает процесс движения транзактов и заново начинает просмотр списка текущих событий. При этом очередность транзактов на входе в блок перестраивается с учетом приоритета транзакта (первым обслуживается транзакт, имеющий более высокий приоиртет, среди транзактов с равным приоритетом первым обслуживается тот, который пришел первым). Далее продолжает движение транзакт, оказавшийся первым в перестроившейся очереди. Блок не связан с задержкой транзакта. Операндов не имеет:

BUFFER

Первоначально приоиртет транзакту назначается блоком GENERATE (операндом E). Изменить уровень приоритета транзакта в ходе продвижения его по модели можно блоком PRIORITY:

PRIORITY A,B

где A – новое значение приоритета;

B – буферизующая опция. Может быть опущен или равен BU (помещает активный транзакт в список текущих событий позади равных ему по приоритету транзактов.

3.28. Переустановка значений генераторов случайных чисел

Команда RMULT устанавливает начальное число для генераторов случайных чисел.

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

где A – начальное число генератора 1;

B – начальное число генератора 2;

C – начальное число генератора 3;

D – начальное число генератора 4;

E – начальное число генератора 5;

F – начальное число генератора 6;

G – начальное число генератора 7;

RMULT ,,111

В этом примере генератор случайных чисел номер 3 инициализируется начальным числом 111.

4. Система имитационного моделирования Arena

Система Arena разработана компанией *Systems Modeling Corporation*. Система Arena предоставляет пользователю широкие интерфейсные воз-

можности среды Windows. В Arene реализована идеология иерархического построения модели от общей схемы последовательно приближаясь к реальному объекту.

4.1. Основные модули для построения простейших моделей

4.1.1. Архитектура Arena

Основа технологий Arena - язык моделирования SIMAN и система Cinema Animation. Для отображения результатов моделирования используется анимационная система Cinema Animation. Процесс моделирования организован следующим образом. Сначала пользователь шаг за шагом строит в визуальном редакторе системы Arena модель. Затем система генерирует по ней соответствующий код на SIMAN, после чего автоматически запускается Cinema Animation. Интерфейс Arena включает в себя всевозможные средства для работы с данными, в том числе электронные таблицы, базы данных, ODBC, OLE, поддержку формата DXF.

4.1.2. Построение моделей в Arena

Имитационная модель в Arena включает следующие основные элементы: источники и стоки (Create и Dispose), процессы (Process) и очереди (Queue).

Источники — это элементы, от которых в модель поступает информация или объекты. Скорость поступления данных или объектов от источника обычно задается статистической функцией.

Сток — это устройство для приема информации или объектов. Время обработки объектов (производительность) в разных процессах могут быть разными. В результате перед некоторыми процессами могут накапливаться объекты, ожидающие своей очереди. Часто целью имитационного моделирования является минимизация количества объектов в очередях.

Процессы — это аналог работ в функциональной модели. В имитационной модели может быть задана производительность процессов.

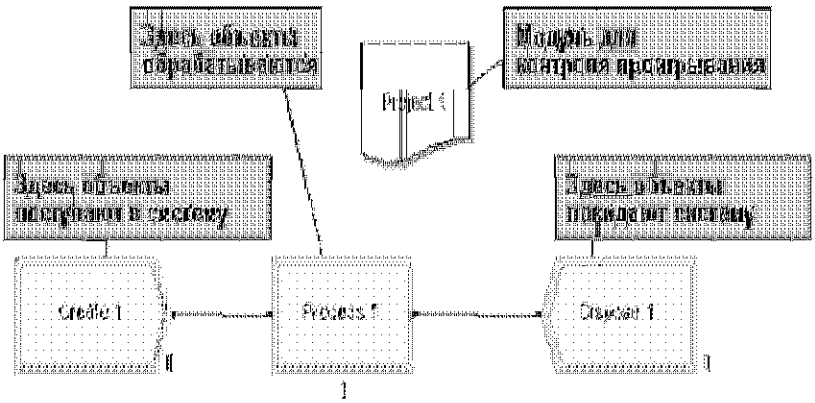
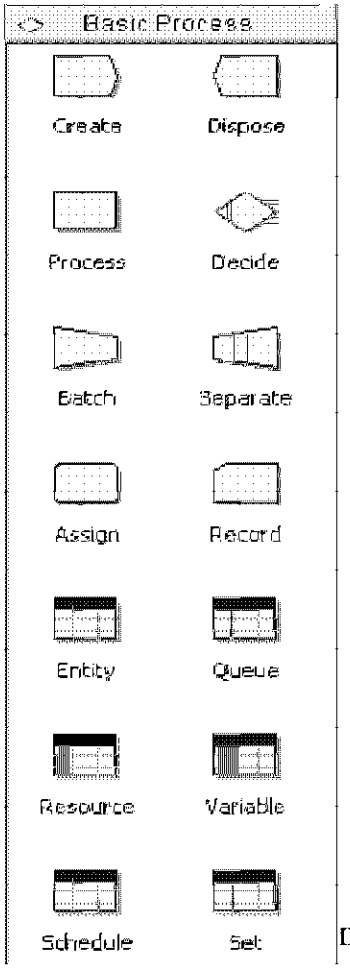


Рис. 4.1. Простейшая имитационная модель, созданная в Arena

Для построения моделей **Arena** имеет набор средств, которые включают палитру инструментов, набор гидов и др. Для создания модели сначала нужно щелкнуть по кнопке New на панели инструментов. Слева появляется палитра инструментов (рис. 2) , которая содержит два типа модулей.



Модули типа Flowchart (в том числе Create, Dispose и Process) служат для отображения потоков объектов и могут быть перенесены на рабочее пространство модели drag&drop. Модули типа Data (например Queue) не могут быть размещены в рабочее пространство модели и служат для настройки параметров модели. Окно редактирования параметров появляется в нижней части модели, когда фокус установлен на модуле типа Data. Подробнее о Flowchart и Data модулях будет написано ниже.

Из панели инструментов в рабочее пространство модели можно перенести по одному модулю Create, Dispose и Process. Связи между модулями устанавливаются автоматически (хотя могут быть переопределены вручную). Модуль Create является источником сущностей (аналогия транзактам в системе GPSS World) в системе. Так, например, если описывается изготовление изделий, то модуль Create может описывать поступление заготовок на конвейер. Модуль Process отвечает за обработку сущностей. Например, он может имитировать станок, об-

інструментов

рабатывающий заготовки. Модуль Dispose является стоком сущностей из системы. Он может моделировать снятие готовых изделий с конвейера.

Для проигрывания модели (имитации реального процесса) необходимо перейти в меню Run/Go. После проигрывания модели автоматически генерируются отчеты.

Модель в Arena может быть гораздо более сложной, чем представленная на рисунке 1. Она может включать сотни модулей различных типов. Модули, обрабатывающие сущности, могут иметь различные состояния, например "ожидание" или "работа". Каждому состоянию можно поставить в соответствие определенное изображение и, тем самым, анимировать имитационную модель.

4.2. Интерфейс и основные модули среды разработки Arena Model

4.2.1. Model Window

В Model Window (Окно Модели), есть две основных области верхняя и нижняя. Верхняя область называется *рабочей областью модели* и содержит всю графику вашей модели, включая блок-схему процесса, анимацию, и другие рисованные элементы. Нижняя область – это *электронная таблица,* в ней отображаются данные модели, как например время, издержки, и другие параметры.

Щелчок на иконке модуля в Project Bar (Панель Проекта, располагается в левой части экрана) отобразит в электронной таблице все модули этого типа.

Как уже говорилось выше, первоначально Model Window разделено на две области, на два подокна (верхнее подокно используется для размещения модулей, изображающих потоки в системе, нижнее подокно является электронной таблицей используемой для ввода и редактирования данных.)

Arena отобразит во весь экран только подокно для модуля, который Вы выбираете в Project Bar. Если Вы щелкните на иконку flowchart(см. ниже) модуля, отобразится подокно рабочей области модели; если же Вы щелкните на иконку Data модуля, отобразится подокно электронной таблицы. Здесь прошу заметить то, что при отображении во весь экран одного из подокон Вы будете не в состоянии отредактировать flowchart модули

через электронную таблицу поскольку всякий раз, когда Вы щёлкаете на иконку flowchart модуля, активизируется рабочая область модели.

Когда Вы запускать модель на выполнение, Арена автоматически скрывает подокно электронной таблицы модели. Вы можете отобразить электронную таблицу во время выполнения, также используя опцию View/Split Screen, или перемещая разделитель на желаемую позицию. Когда моделирование завершится, расположение подокон возвратится к предшествующим установкам.

4.2.2. Понятие “модуль”

Для того, чтобы строить модели в Арене, используются формы моделирования, называемые *модулями*. Модули находятся на панели Basic Process (рис 2.) (и других моделепостроительных панелях отображаемых в Project Bar).

Есть два типа модулей на панели: модули Блок-схемы (Flowchart) и модулей Данных (Data).

Flowchart модули помещаются в рабочую область модели посредством Drag&Drop и связываются, чтобы сформировать блок-схему, описывающую логику вашего процесса.

Для того, чтобы построить блок-схему, перетащите flowchart модули из панели модулей (например Basic Process) в вашу модель и соедините их вместе. Редактировать свойства flowchart модуля, можно либо щелкнув два раза на нём и заполнив открывшуюся форму, либо с помощью электронной таблицы в нижнем подокне model window.

Data модули не помещаются в рабочую область модели. Они редактируются через интерфейс электронной таблицы. Чтобы определить Data модуль, щёлкните на модульной иконке, на панели модулей, активизировав при этом свою электронную таблицу. Двойной щелчок на надписи “Double-click here to add a new row”, в активизированной электронной таблице, добавит новую строку тем самым, создав новый Data модуль. Каждая строка в электронной таблице представляет отдельный модуль. Данные в строке можно редактировать.

4.2.3. Автоматически создаваемые Data модули

Некоторые Data модули автоматически создаются другими модулями. Например, модуль Create требует, чтобы был определён тип создаваемого им объекта (*Entity Type*). А этот тип автоматически сгенерирует свой Data модуль Entity. Кроме того, Арена постоянно отслеживает ссылки на Data модули, которые созданы автоматически и автоматически удаляет их, если они больше не используются моделью. Это означает, что если удалить модуль Create, то его *Entity Type* "Documents" больше не будет ссылаться на Data модуль Entity, таким образом Arena автоматически удалит Entity модуль "Documents" из модели.

Data модули, которые создаются вручную или редактируются, автоматически не удаляются. Например, если отредактировать свойство *Initial Waiting Cost* в Entity модуле "Documents" и затем удалить модуль Create, Entity модуль "Documents" не будет удален.

Следовательно, необходимо удалять вручную Data модули, которые редактировались или добавлялись вручную, что они больше не используются в вашей модели. Для того, чтобы удалить Data модуль, необходимо щелкнуть правой кнопкой мыши на соответствующей строке в электронной таблице и выбрать опцию *Delete Row* (удалить строку).

4.2.4. Различие Data и Flowchart модулей

Data и Flowchart модули имеют несколько различий. Во-первых, Data модуль существует только в форме электронной таблицы, тогда как Flowchart модуль существует как объект в рабочей области модели, так и как строка в электронной таблице. Во-вторых, Data модули могут быть добавлены или удалены через электронную таблицу, тогда как flowchat модули могут быть добавлены или удалены только размещением или удалением объекта из рабочей области модели.

4.2.5. Использование панели Basic Process

Flowchart модули.

Модуль Assign.

Этот модуль используется для присвоения новых значений переменным, атрибутам сущности, типам сущности, изображениям сущности и другим системным переменным. Одним модулем Assign может быть сделано множество назначений.

Переменным статуса модели, таким как, entity cost и временные атрибуты или process cost и время, не может быть присвоена величина в модуле Assign.

Множество назначений будут сделаны в том порядке, в котором они заданы в модуле Assign.

При переназначении типа сущности, новый тип сущности автоматически сгенерируется в электронной таблице Entity, используя значения по умолчанию для изображения сущности и cost информации. Значение по умолчанию может быть изменено щелчком в модуле Entity и модификацией определенного типа сущности.

Переменная, при присвоении значения в модуле Assign, автоматически будет добавлена в электронную таблицу Variable, если она ещё там не определена.

Тип Variable может быть назначен только простой переменной. Переменные-массивы (одно- или двухмерные) должны быть определены при помощи специфического типа Other. Переменные-массивы, назначенные таким образом, автоматически не добавляются в электронную таблицу Variable.

Модуль Batch.

Этот модуль работает как группирующий механизм в пределах имитационной модели. Пакеты сущностей могут быть навсегда или временно сгруппированы. Временные пакеты позже могут быть разделены с помощью модуля Separate.

Группировка может производиться со всеми входящими сущностями(Any Entitis), или основываясь на атрибутах(By Attribute).

Сущности, поступающие в модуль Batch, помещаются в очередь до тех пор, пока необходимое количество сущностей не накопится. Как только необходимое количество сущностей накоплено, модуль выпускает одну сущность.

Модуль Create.

Этот модуль работает как отправной пункт сущностей в имитационной модели. Сущности создаются по расписанию или, основываясь на времени между отправками. После создания, сущности покидают модуль, и участвуют в моделировании. В этом модуле задаётся тип сущности.

Тип сущности автоматически будет добавлен в электронную таблицу Entity, используя значения по умолчанию для её основных свойств. Значе-

ния по умолчанию могут быть изменены щелчком в модуле Entity и модификацией определенного типа объекта.

Если сущности создаются по расписанию, то оно автоматически будет добавлено в электронную таблицу Schedule.

Если вы используете детерминированный подход, тогда для задания интервала времени между отправками сущностей должна использоваться константа (*Constant*). Если вы используете стохастический подход, наиболее подходящее распределение времени между отправлениями – экспоненциальное (*Exponential*). Экспоненциал используется как для Расписания (*Schedule*) так и для Произвольного(Random) типа. В случае произвольного (*Random*) отправления, свойство *Value* – это значение экспоненциального распределения.

Модуль Decide.

Этот модуль работает как условный переход в системе. Он содержит опции для принятия решений основанных на одном или нескольких условиях (например, если тип сущности является *Gold Card*) или основанных на одном или нескольких вероятностях (например, 75% истина (*true*); 25% ложь(*false*)). Условия могут быть основаны на проверке значений атрибутов (*Attribute*) (например, приоритет(Priority)), переменных (*Value*) (например, Number), тип сущности (*Entity Type*), или выражение(*Expression*) (например, NQ(ProcessA.Queue) - количество в очереди).

Модуль Dispose.

Этот модуль предназначен для выхода сущностей из имитационной модели. Статистика сущности может быть записана прежде, чем сущность будет выведена из модели.

Модуль Process.

Этот модуль – основной обрабатывающий метод в моделировании. В нём доступны опции для захвата и освобождения ресурса. К тому же, в этом модуле есть опция, для того чтобы использовать “подмодель” т.е. процесс будет содержать в себе набор модулей выполняющих какую-либо определенную задачу в рамках общей модели.

Модуль Record.

Этот модуль используется, для сбора статистики в имитационной модели. В нем доступны различные типы статистического наблюдения,

включая время между прохождениями сущностей (*Time Between*), статистику объекта (Entity Statistics) (time, costing, и т.п.), общие наблюдения, и статистика интервалов (с некоторого момента времени до текущего имитационного времени). Также доступен тип счета статистики. Могут быть определены Итог (*Tally*) и установки Счетчика (Counter).

Модуль Separate.

Этот модуль используется, для того чтобы копировать поступающие сущности и выдавать на выходе несколько сущностей, а также, для того чтобы разделять прежде объединённые в модуле Batch сущности.

При разбиении временно-объединённых сущностей, оригинальные сущности, которые были до объединения, будут восстановлены. Сущности последовательно выпускаются из модуля в той же последовательности, в которой они первоначально были объединены.

При дублировании сущностей, делается и посылается из модуля определенное количество копий. Оригинальная сущность также покидает модуль.

Data модули.

Модуль Entity.

Этот Data модуль определяет различные типы сущностей и их свойства. Среди свойств такие как *initial picture* (начальное изображение в модели), информация об *initial costing* и *holding costs*.

Модуль Queue.

Этот Data модуль может быть использован, чтобы изменять свойства определенной очереди. По умолчанию все очереди работают по правилу “первый пришёл, первый ушёл” (FIFO), если в этом модуле не указано другое.

Модуль Resource.

Этот Data модуль определяет ресурсы в имитационной системе, включая costing information и доступность ресурса. Ресурсы могут иметь фиксированные возможности, которые не изменить после в ходе моделирования или могут обслужить, базируясь на расписании.

Модуль Schedule.

Этот Data модуль может быть связан с модулем Resource, чтобы определять расписание работы для ресурса или с модулем Create, чтобы определять расписание выпуска сущностей. К тому же, расписание может быть использовано в распределении задержек времени, основанном на имитационном времени.

Модуль Set.

Этот Data модуль определяет различные типы наборов, включая Resource, Counter, Tally, Entity Type и Entity Picture. Наборы Ресурса могут быть использованы в модуле Process. Наборы Counter и Tally могут быть использованы в модуле Record.

Модуль Variable.

Этот Data модуль используется для того, чтобы определять измерения и начальные значения переменных. Переменным может быть присвоена новая величина в модуле Assign, и может быть использована в любом выражении (Expression).

Submodel (Подмодель).

В модуле Process, есть два типа обработки. *Standart* (стандартная), означает, что сущность просто будет обработана в пределах модуля, основываясь на свойстве Action (Действие) которое может иметь следующие значения: Delay (задержка), Seize Delay (задержка с захватом), Seize Delay Release (задержка с захватом и выпуск) или Delay Release (задержка и выпуск).

Другой тип обработки - это *Submodel*(подмодель). Обработка в подмодели позволяет пользователю, определять больший комплекс и иерархическую логику для обработки в пределах данного модуля Process.

Когда тип процесса определен как *submodel*, все области для задания свойств модуля Процесса исчезают и не используются. В место этого отображается текст, который описывает как, входить и выходить из подмодели. Для того, чтобы войти в подмодель, просто щёлкните правой кнопкой мыши на модуле Process, затем в открывшемся меню выберите опцию Edit Submodel (редактировать подмодель). Появится окно, в котором можно задать логику подмодели подобно основной модели. Окно содержит слева - входную точку и справа – выходную. Любая логика, определённая в подмодели, должна быть подключена к входной и выходной точкам. Выйти из подмодели, можно щёлкнув правой кнопкой мыши и выбрав вкладку Close Submodel.

Подмодель можно создать и другим способом: меню Object - Submodel - Add Submodel. Полученная таким образом подмодель внешне бу-

дет выглядеть несколько иначе, чем подмодель модуля Process, но тем не менее это то же самое, за исключением некоторых отличий. Например, вход в данную подмодель можно осуществить двойным щелчком левой кнопки мыши на подмодели.

Ещё одним отличием полученной подмодели является то, что она может иметь несколько точек входа и выхода. Указать требуемое вам количество точек можно в свойствах подмодели, которые можно открыть, щелкнув на подмодели правой кнопкой мыши и выбрав вкладку Properties. Number of entry points – это количество входных точек, number of exit points – соответственно выходных.

4.3. Практическое применение основных элементов среды имитационного моделирования Arena

4.3.1. Краткие сведения об объекте моделирования

Объектом исследования и моделирования в данной работе является участок транспортного цеха, обеспечивающий процесс отгрузки готовой продукции (автомобилей), производимой автомобильным заводом (рис. 4.3). Со сборочного конвейера предприятия каждые 6 мин сходит готовый автомобиль, который за 8..12 мин перегоняется на склад готовой продукции. Склад является промежуточным звеном между сборкой и отгрузкой в транспортном цехе предприятия. На погрузочную эстакаду транспортного цеха один раз в сутки подается состав из 10 платформ, каждая из которых вмещает по 24 автомобиля. Погрузка автомобилей может производиться на эстакаде транспортного цеха на две платформы одновременно. Для погрузки одного автомобиля на платформу в среднем требуется 5 минут. Причем время погрузки автомобиля является случайным и интервалы времени погрузки имеют экспоненциальное распределение. Подача каждой пустой платформы на эстакаду под погрузку занимает 5 мин. Сборочный конвейер завода работает круглосуточно, а состав подается под погрузку ежедневно в 6 часов утра. Для избежания простоя подвижного состава из-за отсутствия в момент погрузки готовых к отправке автомобилей на предприятии (на складе готовой продукции) вводится страховой запас, который должен быть определен экспериментально на базе имитационной модели.

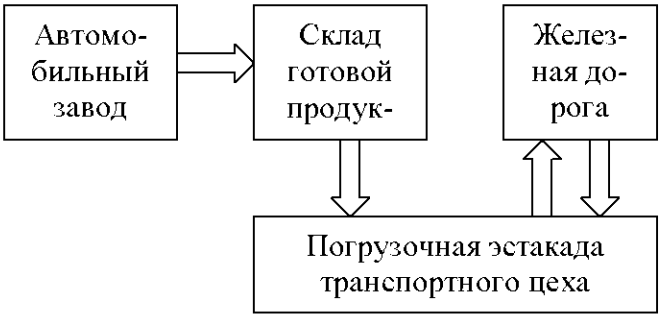
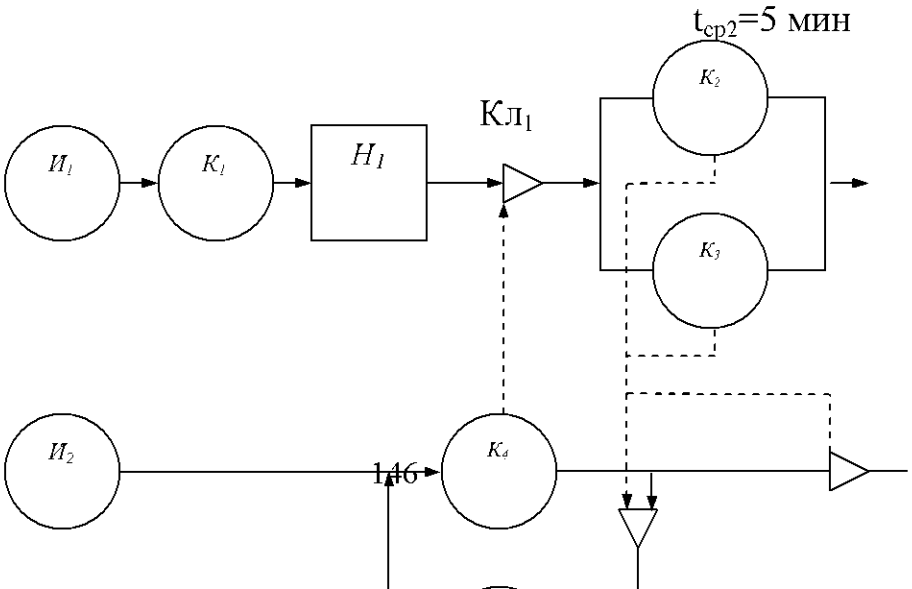


Рис. 4.3. Участок транспортного цеха

Необходимо смоделировать работу транспортного цеха в течении 7 суток.

Функциональная схема работы цеха приведена на рис. 4.4, где И1, И2 - соответственно источники, имитирующие поступление автомобилей и составов; К1 - имитатор перегона автомобилей, К2 и К3 - имитаторы погрузки автомобилей на платформу, К4 и К5 - имитаторы простоя под погрузкой и перегона платформ. Пунктиром обозначены направления управляющих воздействий на клапаны Кл, сплошными линиями - направления материальных потоков.



Δt=6 мин $t_{ср1}$=[8,12] L=300

$t_{ср3}$=5 мин

$Кл_3$

$Кл_2$

Δt=360 мин

t_5=5 мин

Рис. 4.4. Q-схема работы транспортного цеха

4.3.2. Выполнение модели в среде Arena

Для начала определим логику модели. В данной работе мы имеем два потока сущностей, первый – это автомобили из источника И1, второй – составы из источника И2. Т.е. в нашей модели будет два модуля Create (рис. 4.5). Назовём их соответственно Sborochniy Sklad и RWDepo.

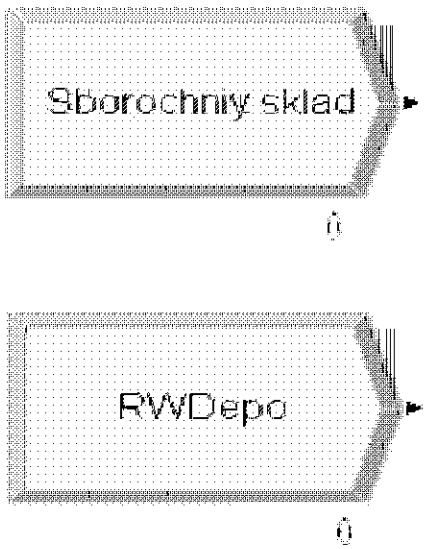


Рис 4.5. Источники Create.

Теперь нужно отредактировать свойства источников в соответствии с поставленной задачей. Для редактирования свойств будем пользо-

ваться формой заполнения свойств, которая открывается двойным щелчком на соответствующем модуле.

Итак, рассмотрим сначала модуль Sborochniy Sklad. Со сборочного склада по заданию каждые 6 мин (редактируем группу свойств Time Between Arrivals: свойству Type назначим Constant, свойству Value – 6, свойству Units - Minutes) сходят готовые автомобили (свойство Entity Type назовём Cars). За 6 минут выходит один автомобиль, поэтому свойство Entities per Arrivals равно 1. Т.к. общее модельное время 7 суток, то за это время со сборочного склада сойдёт 1680 автомобилей, следовательно свойству Max Arrivals присвоим значение 1680. И свойству First Creation(время выпуска первого автомобиля) присвоим значение 0.0(значение по умолчанию).Таким образом форма свойств модуля Sborochniy Sklad будет выглядеть так:

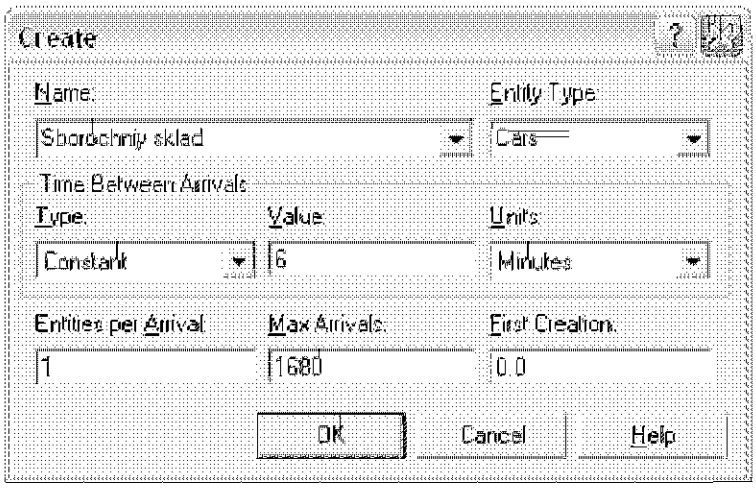


Рис. 4.6. Свойства модуля Sborochniy Sklad

Теперь рассмотрим свойства модуля RWDepo. По заданию состав(свойство Entity Type - Sostav) подается один раз в сутки(свойство Type – Constant; Value – 24; Units – Hours; Entities per Arrival - 1) под погрузку ежедневно в 6 часов утра(First Creation - 6). Всего за 7 суток подаётся 7 составов(Max Arrivals - 7). Форма свойств модуля RWDepo будет выглядеть так:

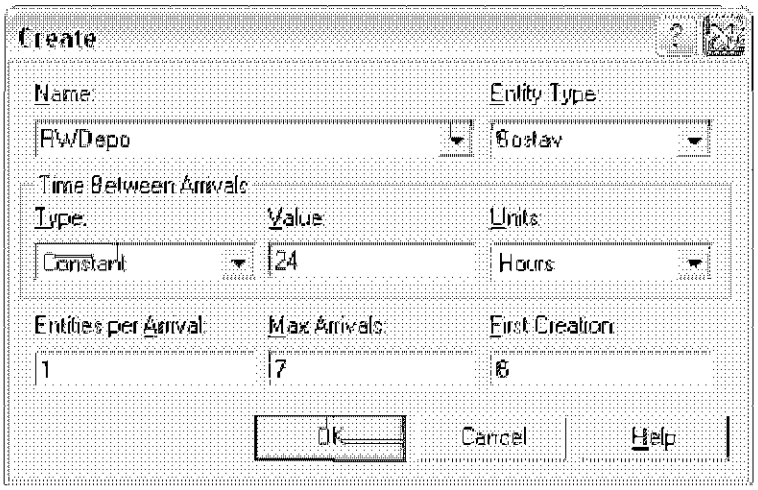


Рис. 4.7. Свойства модуля RWDepo

Все свойства этих двух модулей можно также посмотреть и отредактировать в подокне электронной таблицы Create – Basic Process в нижней части экрана.

Рассмотрим сначала ветку движения автомобилей. Со сборочного склада автомобили перегоняются до склада готовой продукции. Перегонка по условию занимает от 8 до 12 минут. Создадим подмодель “Peregonka” которая будет содержать в себе набор модулей реализующих процесс перегонки. Здесь нам потребуются модули Process для реализации собственно перегонки, Decide для проверки заполненности склада готовой продукции (если склад заполнен, автомобили теряются; если нет – автомобили поступают на склад) и Dispose для уничтожения автомобилей, не попавших на склад.

Рассмотрим модуль Process. Назовём его Peregonka(свойство Name), свойства Type и Action не меняем, оставляем соответственно Standart и Delay. Delay Type (тип задержки) выберем Uniform, свойству Units присвоим Minutes, Minimum и Maximum (соответственно минимальное и максимальное значение задержки) – 8 и 12. Таким образом, форма свойств модуля Peregonka будет выглядеть так:

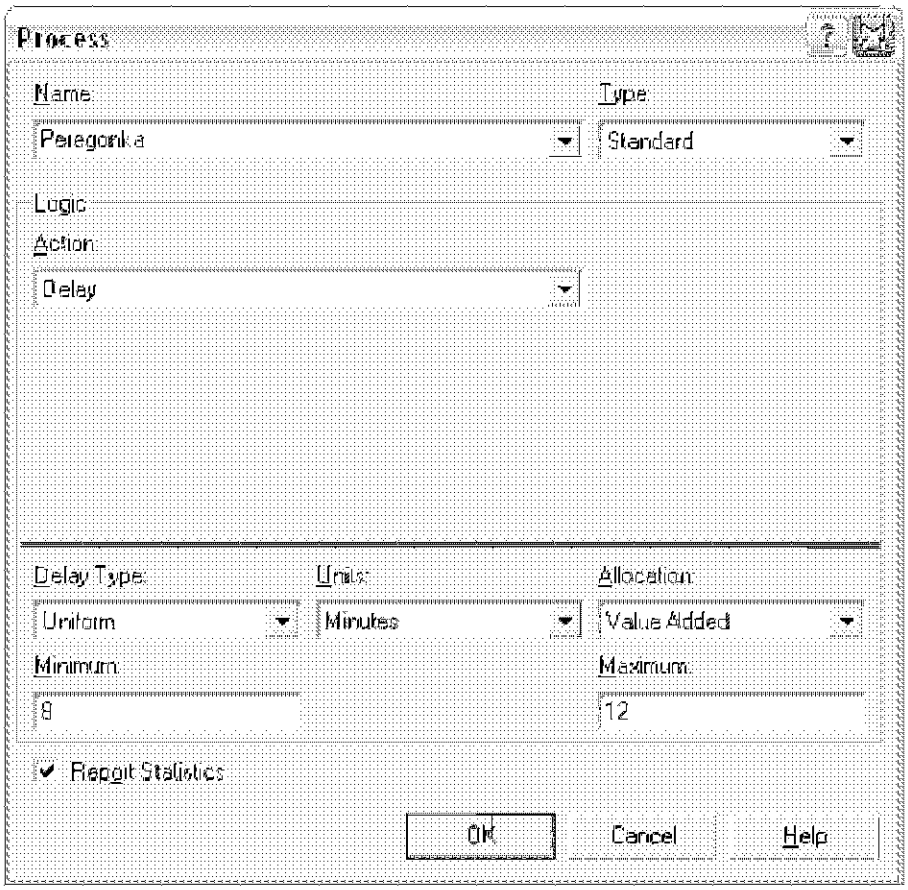


Рис. 4.8. Свойства модуля Peregonka

Рассмотрим модуль Decide. Назовём его вопросом Sklad Zapolnen?. По условию ёмкость склада равна 300. Т.к. склад – это накопитель, который помещает автомобили в очередь(Queue), то в модуле Sklad Zapolnen? Нужно проверять количество автомобилей в очереди. Для этого свойству Type присвоим значение “2-way by Condition” (два пути по условию), свойству if присвоим значение “Expression” и в свойстве “Value” напишем выражение NQ(Sklad.Queue) == 300, где NQ – Number of Queue(количество в очереди) и Sklad.Queue это название проверяемой очереди. В результате мы получаем модуль, который проверяет заполненность склада и если количество автомобилей на складе равно 300, то автомобили уничтожаются, уходя через выход true, а если количество автомобилей на складе не равно 300 (т.е. меньше 300), автомобили поступают на склад. Форма заполнения свойств модуля Sklad Zapolnen? выглядит так:

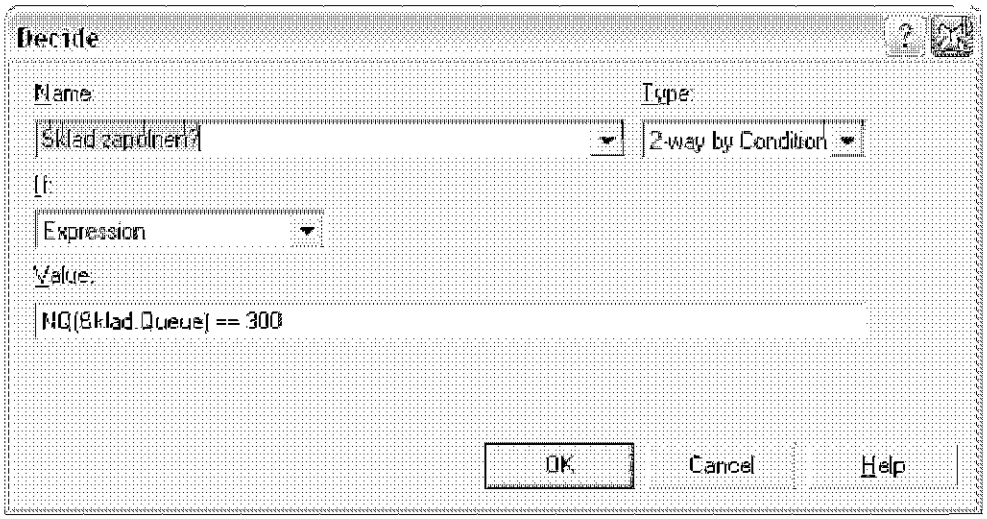


Рис. 4.9. Свойства модуля Sklad Zapolnen?

Следующий модуль - это модуль Dispose, который просто уничтожает автомобили, не попавшие на склад назовём его Destroy.

В итоге в подмодели Peregonka у нас получится следующее:

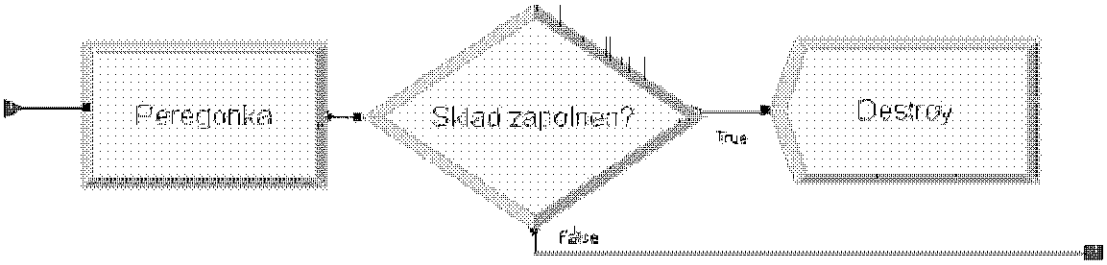


Рис. 4.10 Свойства модуля Sklad Zapolnen?

Следующим этапом у нас будет склад готовой продукции, куда поступают автомобили посредством перегонки. Для реализации склада нам потребуется модуль **Hold**. Этого модуля нет на панели Basic Process, он находится на другой панели, которая называется **Advanced Process.** Эта панель открывается следующим образом: щелчком правой кнопки мыши на панели Basic Process откроем меню, в котором выберем вкладку **Attach…,** перед нами откроется окно **Attach Template Panel** содержащее файлы панелей модулей с расширением ***tpo.** Далее выбираем файл с названием **Advanced Process.tpo** и щелкаем Открыть. В результате в левой части окна (там же где и панель Basic Process) появится панель Advanced Process, которая содержит нужный нам модуль Hold.

Модуль Hold.

Этот flowchart модуль будет держать сущность в очереди либо до сигнала, либо пока условие не станет истинным, либо бесконечно.

Если сущность ожидает выполнения условия, то до тех пор, пока условие не станет истинным, сущность будет оставаться в очереди.

Для реализации склада также создадим подмодель Sklad в которую поместим один модуль Hold. На складе автомобили ожидают до тех пор пока какая либо из двух платформ не будет готова к погрузке. Создадим две переменные Sostoyanie1p и Sostoyanie2p (соответственно состояние 1-й платформы и состояние 2-й платформы).

Для того чтобы создать переменную необходимо выбрать на панели Basic Process модуль Variable. В подокне электронной таблицы откроется таблица Variable - Basic Process, в которой двойным щелчком по надписи “Double-click here to add a new row”, либо, щёлкнув правой кнопки мыши на таблице открыть выпадающее меню, в котором, выбрав вкладку Insert Row (вставить строку), добавить переменную. Добавленная переменная имеет несколько свойств (Name, Row, Columns, Initial Values и др.), которые можно редактировать прямо в электронной таблице.

Переменные Sostoyanie1p и Sostoyanie2p могут иметь одно из двух значений: 0- соответствующая платформа не готова; 1- соответствующая платформа готова. Платформы по условию готовы тогда, когда состав пришёл на погрузку. Поэтому в начале моделирования переменные Sostoyanie1p и Sostoyanie2p должны быть равны 0, т.к. первый состав подходит только в 6 часов утра, а процесс моделирования начинается с 0 часов.

Добавленный модуль Hold назовём Sklad и для того, чтобы он работал по условию, сменим свойство Type на “Scan for Condition”. А в поле Condition впишем следующее выражение “(Sostoyanie1pl==1) || (Sostoyanie2pl==1)”. Т.о. если хотя бы одна платформа свободна, то автомобили выпускаются со склада. Форма заполнения свойств модуля Sklad выглядит так:

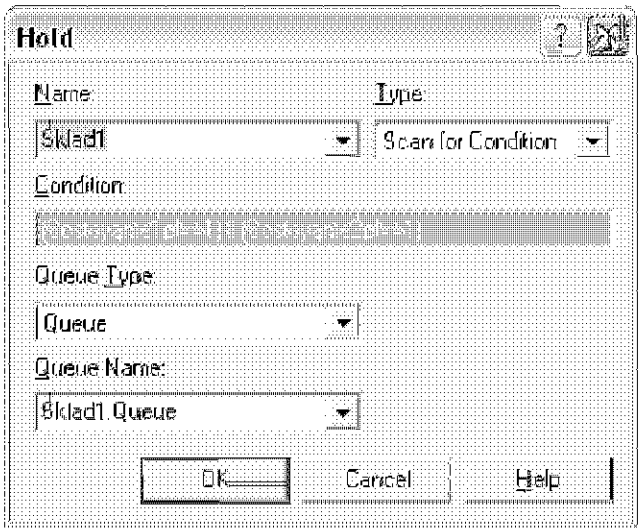


Рис. 4.11. Форма заполнения свойств модуля Sklad

В целом подмодель Sklad будет выглядеть так:

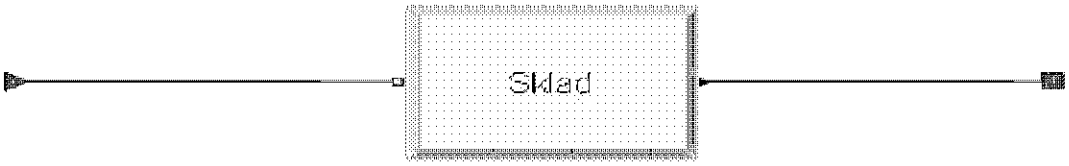


Рис. 4.12. подмодель Sklad

Далее необходимо распределить потоки автомобилей между двумя платформами. Если первая платформа не готова, нужно посылать автомобили на вторую платформу. Если же не готова вторая платформа, нужно посылать автомобили на первую платформу, но какая-то платформа всё равно должна быть готова, потому что условие в модуле Sklad было выполнено. Другими словами данное распределение автомобилей- это группа условий, которые мы объединим в подмодели Klapan 1. Эта подмодель будет содержать два модуля Decide. Первый из них будет иметь три выхода (два true и один false), для этого свойству модуля Type необходимо присвоить значение N-way by Condition (N путей по условию). Первое условие будет следующим (Sostoyanie1pl==1) && (Sostoyanie2pl==1) (если обе платформы готовы); второе условие (Sostoyanie1pl==1) && (Sostoyanie2pl==0) (если готова только первая платформа); и оставшееся состояние - готова только вторая платформа. Форма заполнения свойств первого модуля Decide выглядит так:

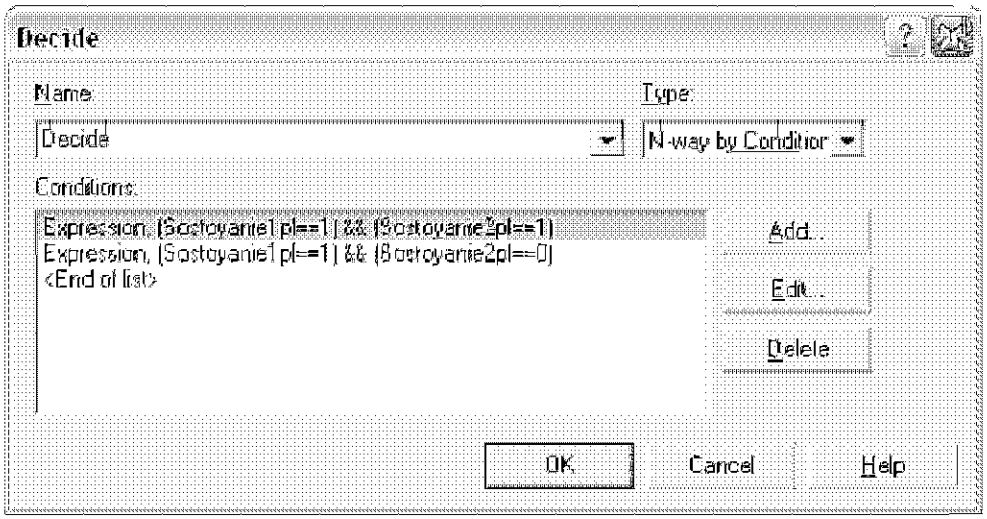


Рис. 4.13. Свойства модуля Decide

Если выполнится первое условие, т.е. первая и вторая платформы готовы, то автомобили распределим между ними с равной вероятностью (50% на 50%). Для этого автомобиль, вышедший из первого выхода, далее входит во второй блок Decide, который назовём "50% NA 50%", свойство Type оставим 2-way by Chance, свойство Percent True (0 - 100) – 50. Форма заполнения свойств первого модуля "50% NA 50%" выглядит так:

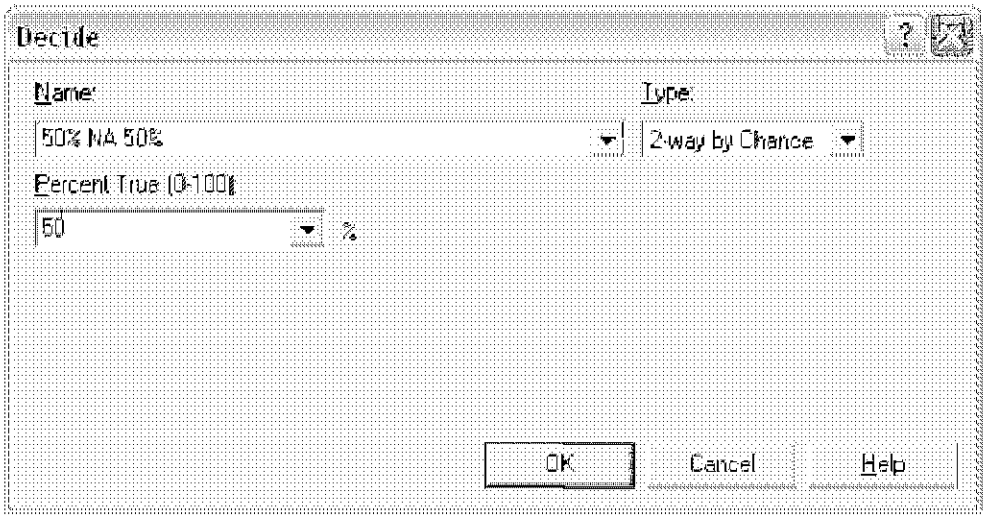


Рис. 4.14. Свойства модуля 50% NA 50%

Если выполнится второе условие, т.е. будет готова к погрузке только первая платформа, то автомобиль сразу направляется на первую платформу. Если же ни одно из первых двух условий не выполнится, автомобиль направится на вторую платформу. В целом подмодель Klapan будет выглядеть так:

Таким образом подмодель Klapan 1 имеет два выхода, один на

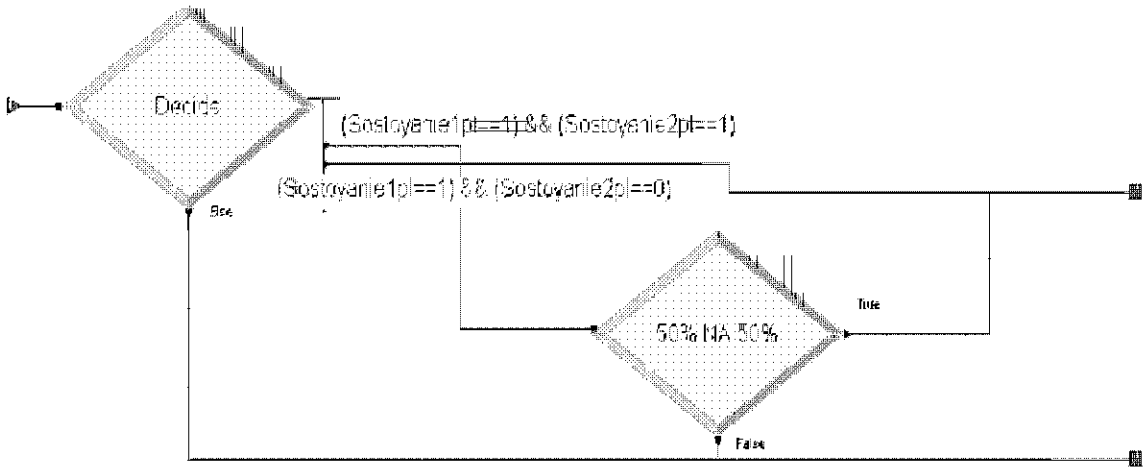


Рис. 4.15. Подмодель Klapan

первую платформу, другой - на вторую. Обе платформы реализуем в виде подмоделей, которые назовём соответственно Platforma 1 и Platforma 2. Рассмотрим логику работы обеих платформ. По условию, на погрузочную эстакаду транспортного цеха один раз в сутки подается состав из 10 платформ, каждая из которых вмещает по 24 автомобиля. Погрузка автомобилей может производиться на эстакаде транспортного цеха на две платформы одновременно. Для погрузки одного автомобиля на платформу в среднем требуется 5 минут. Причем время погрузки автомобиля является случайным, и интервалы времени погрузки имеют экспоненциальное распределение. Для отслеживания состояния погрузки (платформа должна сменяться после погрузки на неё 24-х автомобилей) введём две переменные - счётчика p1 и p2 соответственно для первой и второй платформы. С поступлением каждого нового автомобиля переменная – счётчик увеличивает своё значение на единицу. При достижении переменной - счётчиком значения 24, она обнуляется, а платформа становится не готовой к дальнейшей погрузке до тех пор, пока платформа не поменяется или не подойдёт новый состав. Для реализации рассмотренной логики в среде Arena используем следующие блоки: Process, два блока Assign и блок Decide. Рассмотрим сначала первую платформу. В блоке Process, назовём его Pogruzka 1 (для второй платформы Pogruzka 2), идет погрузка, которая занимает в среднем 5 мин. по экспоненциальному закону распределения (свойство Action имеет значение Delay, Delay Type – Expression, Expression – EXPO(5)). Таким образом, форма заполнения свойств модуля Pogruzka 1 имеет вид:

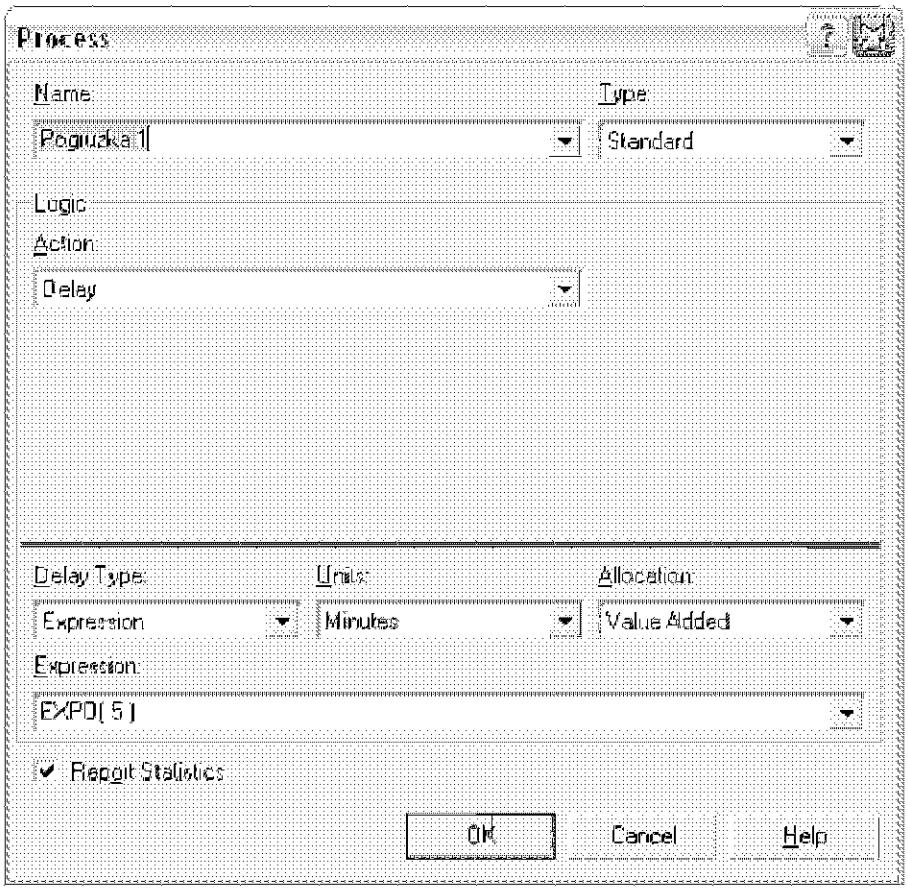


Рис. 4.16. Подмодель Pogruzka 1

Следующий модуль – модуль Assign. Назовём его p1Plus1 (для второй платформы p2Plus1). В этом модуле переменная - счётчик p1 (для второй платформы p2) увеличивается на единицу. Форма заполнения свойств модуля p1Plus1 имеет вид:

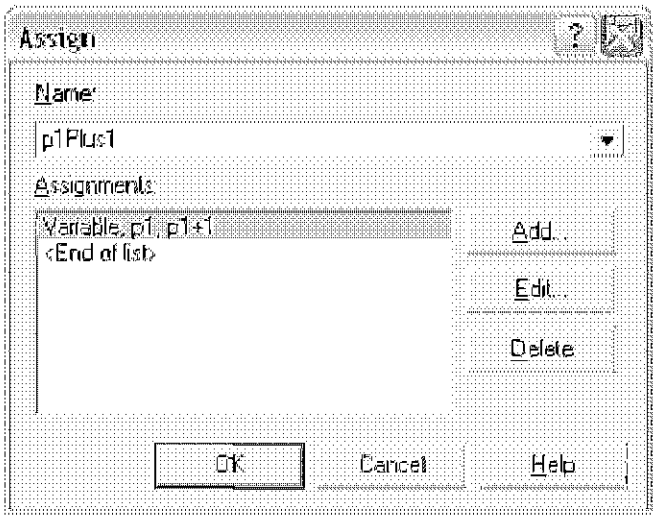


Рис. 4.17. Модуль p1Plus1

Следующий модуль – модуль Decide. Назовём его p1EQ24?(p2EQ24?). В этом модуле проверяется значение переменной p1(p2), если оно равно 24 – выход true, иначе – выход false. Форма заполнения свойств модуля p1EQ24? имеет вид:

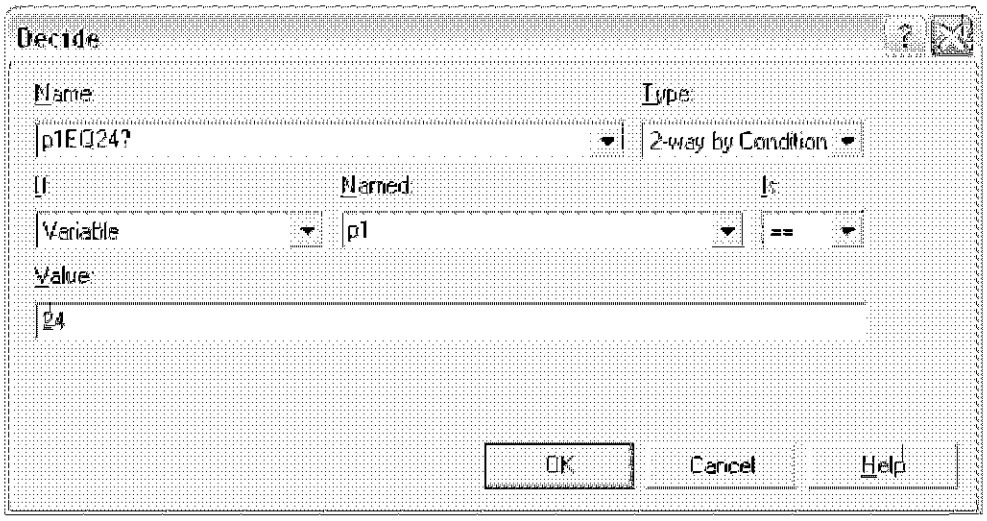


Рис. 4.18 Модуль p1EQ24?

Если сработал выход true, то следующий модуль – модуль Assign, назовём его p1EQ0(p2Q0). В этом модуле переменная p1(p2) обнуляется и переменной Sostoyanie1pl(Sostoyanie2pl) присваивается значение 0. Т.о. форма заполнения свойств модуля p1EQ0 имеет вид:

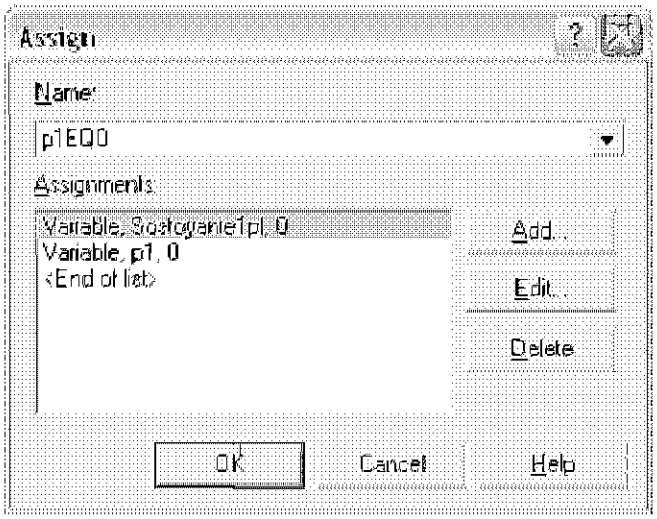


Рис. 4.19. Модуль p1EQ0

Если же у модуля p1EQ24? сработал выход false, то модуль p1EQ0(p2Q0) пропускается.

В целом подмодель Platforma 1 выглядит следующим образом:

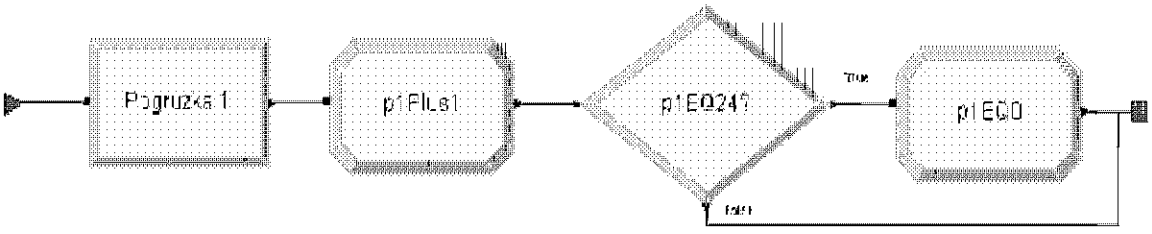


Рис. 4.20. Подмодель Platforma 1

Подмодель Platforma 2 аналогична подмодели Platforma 1 и выглядит так:

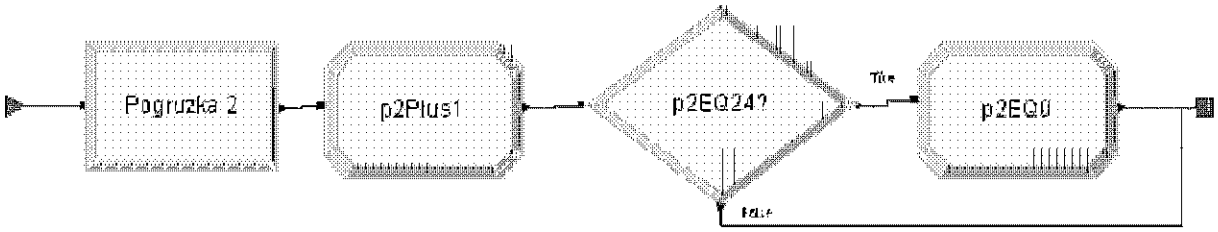


Рис. 4.21. Подмодель Platforma 2

После подмоделей погрузочных платформ ветка моделирования движения автомобилей завершается модулем Dispose. Назовём его Realise.

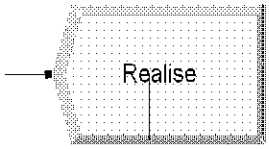


Рис. 4.22. Модуль Realise

Параллельно (одновременно) с движением автомобилей, от сборочного склада до погрузочных платформ, нужно смоделировать процесс подхода состава с платформами. Модуль выпускающий составы – модуль RWDepo, мы разобрали выше, когда разбирали модуль Sborochniy Sklad. Из модуля RWDepo состав попадает в модуль Assign, в котором переменным Sostoyanie1pl и Sostoyanie2pl присваивается значение 1, также присваивается 1 новой переменной root(переменная, которая может принимать значения 0 или 1, когда она равна 0 – все платформы загружены или догружается последняя платформа; 1 – есть свободные платформы, ожидающие погрузки). Форма заполнения свойств модуля p1EQ0 имеет вид:

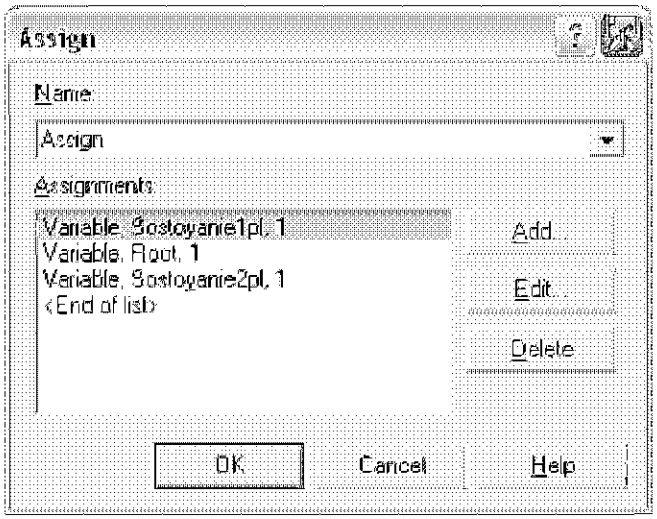


Рис. 4.23. Модуль p1EQ0

После модуля Assign поместим подмодель Platforma gotova. В этой подмодели будет проверяться готовы платформы к погрузке или нет, если какая-либо платформа не готова, значит происходит её замена, которая по условию занимает 5 минут. Подмодель будет содержать следующие модули: два модуля Assign, модуль Hold и два модуля Decide. Рассмотрим первый модуль – модуль Hold . В этом модуле платформы состава ожидают условия, пока какая-либо из платформ будет не готова (загружена) или количество автомобилей, вышедших из модуля Sborochniy Sklad, будет равняться количеству уничтоженных автомобилей в модуле Realise (т.е. процесс моделирования завершен). Таким образом, в строке Condition(условие) модуля Hold, назовём его Platforma gotova, будет строка следующего содержания: (Sostoyanie1pl==0) || (Sostoyanie2pl==0) || (Sborochniy sklad.NumberOut == Realise.Numberout). Форма заполнения свойств модуля Platforma gotova имеет вид:

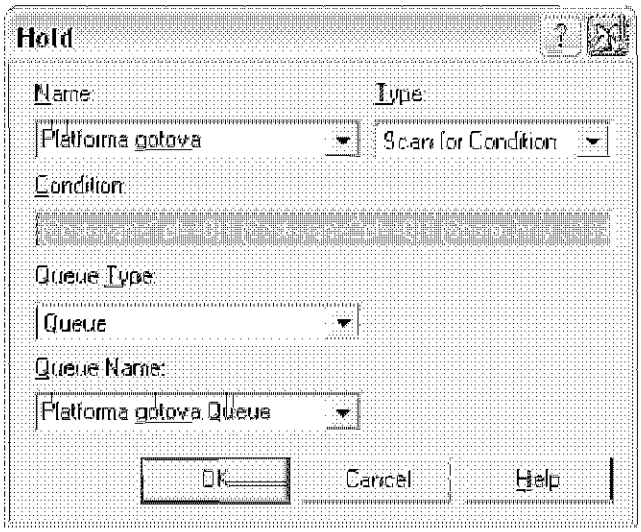


Рис. 4.24. Модуль Platforma gotova

После модуля Hold, идёт модуль Decide, который проверяет, закончено моделирование или нет, назовём этот модуль соответственно вопросом Simulate End?. Если моделирование закончено, т.е. выполняется условие (Sborochniy sklad.NumberOut == Realise.Numberout), то состав направляется в модуль Dispose. Если моделирование не закончено, т.е. выполняется условие (Sborochniy sklad.NumberOut <> Realise.Numberout). Форма заполнения свойств модуля Simulate End? имеет вид:

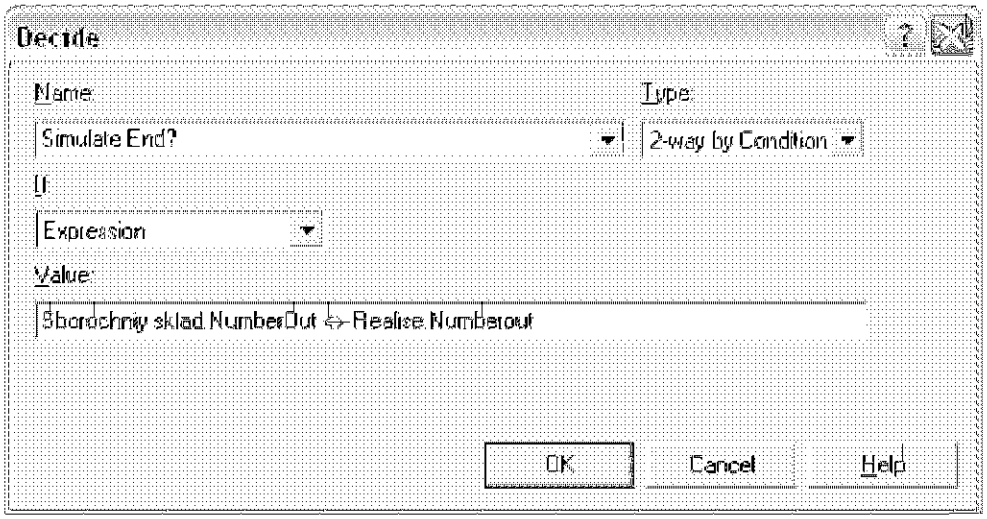


Рис. 4.25. Модуль Simulate End?

Следующий модуль – опять модуль Decide. В этом модуле нам необходимо выяснить какая именно из платформ (первая (1) или вторая(2)) заполнена. Для этого будем проверять условие Sostoyanie1pl == 0. Если условие истинно, следовательно, первая платформа не готова к дальнейшей погрузке и следующий модуль Assign присвоит новой переменной Variable 1 значение 1. Если условие ложно, следовательно, вторая платформа не го-

това к дальнейшей погрузке и следующий модуль Assign присвоит новой переменной Variable 1 значение 2. По условию состав состоит из 10-ти платформ, каждая из которых вмещает 24 автомобиля, значит всего на состав можно погрузить 240 автомобилей. Введём новую переменную CurPogr, которая с каждой погруженной платформой будет увеличиваться на значение 24. И при достижении значения 216, платформа, на которой произошла загрузка, более не участвует в погрузке текущего состава (переменная root = 0). Когда догрузится вторая платформа, значение переменной CurPogr становится равным 240 и состав "уходит" в модуль Dispose. Таким образом, в модулях Assign (где переменная Variable 1 принимает значения либо 1, либо 2) переменная CurPogr будет увеличиваться на 24. Форма заполнения свойств одного из модулей Assign будет иметь вид:

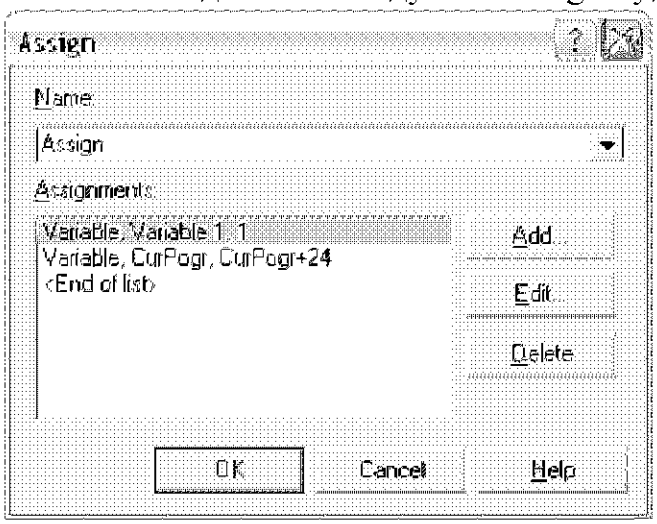


Рис. 4.26. Модуль Assign

Форма заполнения свойств второго модулей Assign будет иметь аналогичный вид, за исключением первой строки в поле Assignments, которая будет выглядеть так Variable, Variable 1, 2. В целом подмодель Platforma gotova будет иметь вид:

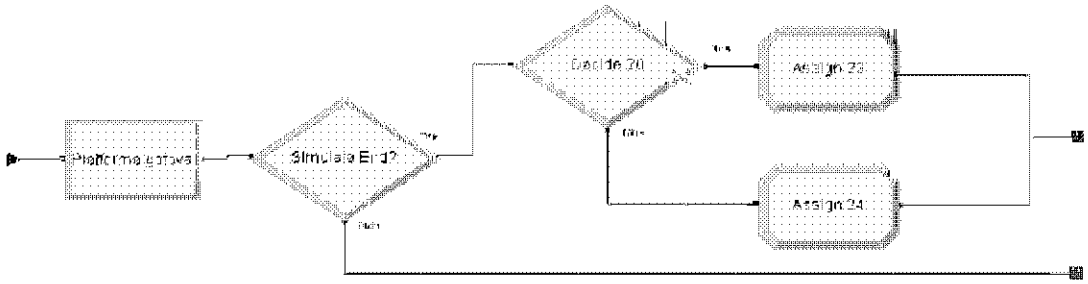


Рис. 4.27. Подмодель Platforma gotova

Далее в нашей модели идёт подмодель, Klapan 2. Эта подмодель содержит два модуля Decide и модуль Assign. Рассмотрим первый модуль – это модуль Decide. В этом модуле проверяется условие CurPogr <> 216 (переменная CurPogr не равна 216). Форма заполнения свойств данного модуля Decide будет иметь вид:

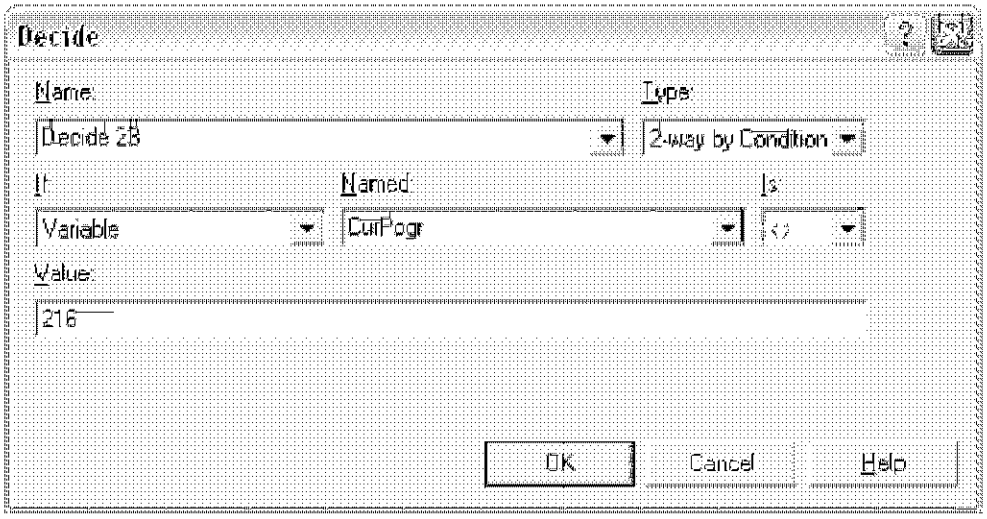


Рис. 4.28. Модуль Decide 23

Если условие модуля Decide истинно, следовательно, далее управление переходит ещё одному модулю Decide, в котором проверяется условие CurPogr ==240. Форма заполнения свойств данного модуля Decide будет иметь в общем такой же вид как и у предыдущего модуля Decide, за исключением поля is и поля Value.

Если же условие первого модуля Decide ложно, то это значит что переменная CurPogr имеет значение 216. В этом случае, нагруженная платформа более не участвует в погрузке текущего состава, а платформа, находящаяся в данный момент под погрузкой, завершит погрузку состава, поэтому далее следует модуль Assign, в котором переменной root, учитывающей этот момент, присваивается значение 0. Таким образом, подмодель Klapan 2 будет иметь следующий вид:

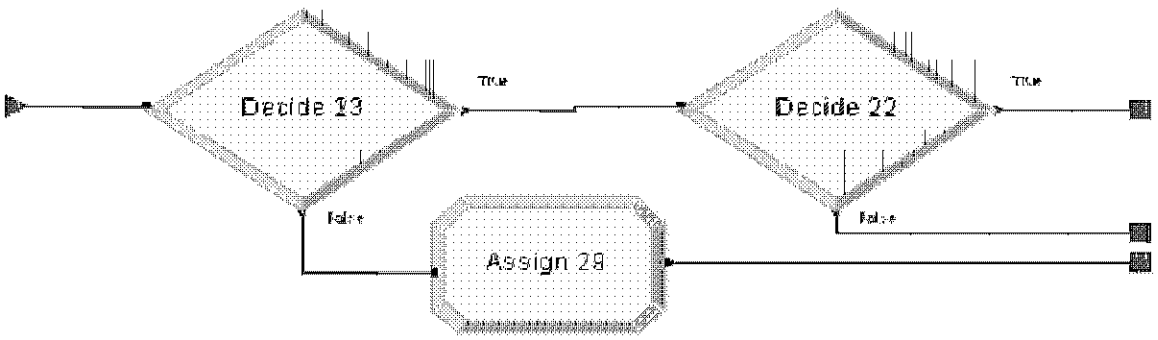


Рис. 4.29. Подмодель Klapan 2

У нас получилось, что подмодель Klapan 2 имеет 3 выхода. 1-й – CurPogr==240; 2-й – CurPogr<>240 и CurPogr<>216; 3-й - CurPogr==216 и root==0. Первый выход будет означать, что состав полностью загружен и должен уйти, т.о. следующий по этому ответвлению модуль – это модуль Assign, назовём его Klapan 3. В этом модуле переменная CurPogr обнуляется. Далее следует модуль Dispose, назовём его RWDepoOut, через него загруженные составы выводятся из системы.

Второй и третий выходы подмодели Klapan 2 будут входить в новую подмодель, которую назовём Smena platformi. Эта платформа будет иметь два входа: 1-й - CurPogr<>240 и CurPogr<>216; 2-й - CurPogr==216 и root==0. В данной подмодели будет осуществляться смена платформы, и приниматься решение о готовности платформы. Данная платформа будет содержать следующие модули: Process, Decide и четыре модуля Assign. Первый вход будет передавать управление модулю Process. Этот модуль и есть собственно смена платформы, так и назовём его Smena platformi. По условию смена платформы занимает 5 минут, таким образом, форма заполнения свойств этого модуля будет выглядеть так:

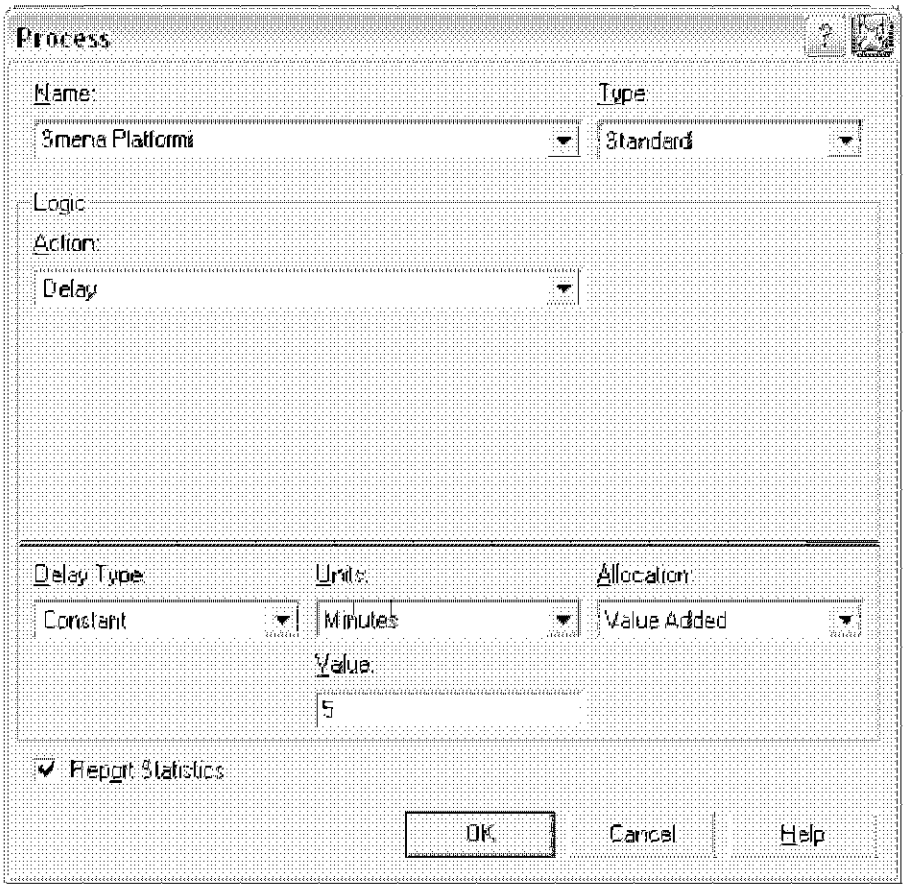


Рис. 4.30. Модуль Smena platformi

Далее идет объединение двух входов подмодели в модуле Decide. Этот блок будет иметь четыре выхода для четырёх вариаций переменных Variable 1 и Root. Рассмотрим эти вариации. Первая: Variable 1 ==1 и Root == 1(сменилась первая платформа и нужно сделать её активной). Вторая: Variable 1 ==2 и Root == 1(сменилась втрая платформа и нужно сделать её активной). Третья: Variable 1 ==1 и Root == 0(первая платформа более не участвует в погрузке текущего состава). Четвёртая: Variable 1 ==2 и Root == 0(вторая платформа более не участвует в погрузке текущего состава). Форма свойств данного блока выглядит так:

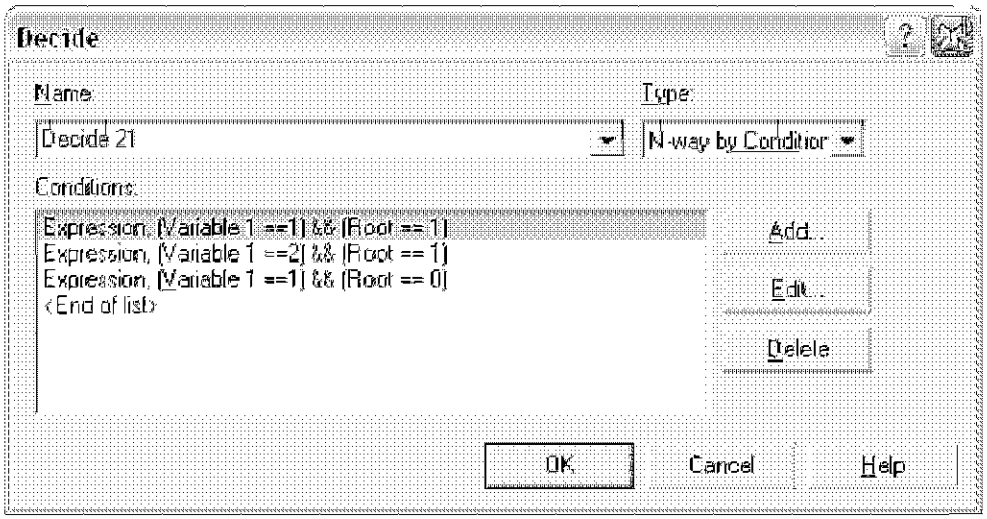


Рис. 4.31. Модуль Decide 21

Далее каждому выходу соответствует свой модуль Assign. Назовём их соответственно Assign 1, Assign 2, Assign 3, Assign 4. В модуле Assign 1 переменной Sostoyanie1pl присваивается значение 1. В модуле Assign 2 переменной Sostoyanie2pl присваивается значение 1. В модуле Assign 1 переменной Sostoyanie1pl присваивается значение 2 (в этом случае первая платформа более не будет участвовать в погрузке текущего состава). В модуле Assign 1 переменной Sostoyanie2pl присваивается значение 2 (в этом случае вторая платформа более не будет участвовать в погрузке текущего состава). В целом подмодель Smena platformi будет выглядеть так:

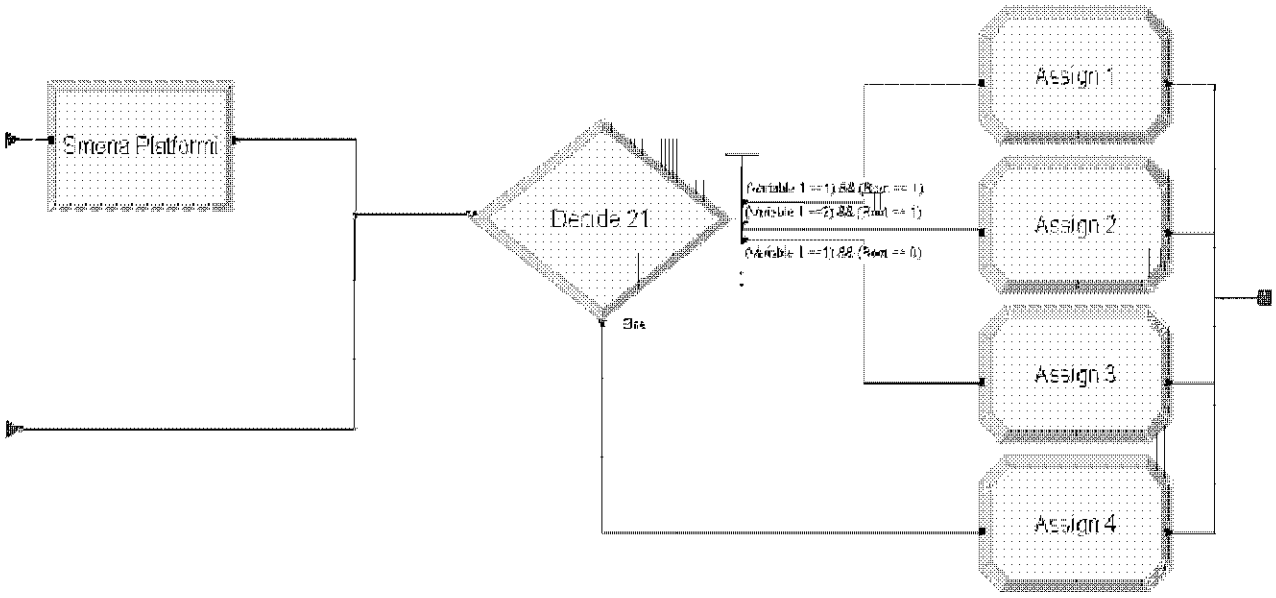


Рис. 4.32. Подмодель Smena platformi

Выход данной подмодели будет соединён с входом подмодели Platforma gotova т.е. создали цикл, в который входят подмодели Platforma gotova, Klapan 2 и Smena platformi и в котором происходит смена платформ до тех пор пока не будет полностью загружен весь состав.

Итак, мы полностью разобрали модель транспортного цеха в среде Arena. В общем виде она выглядит как на рисунке 4.

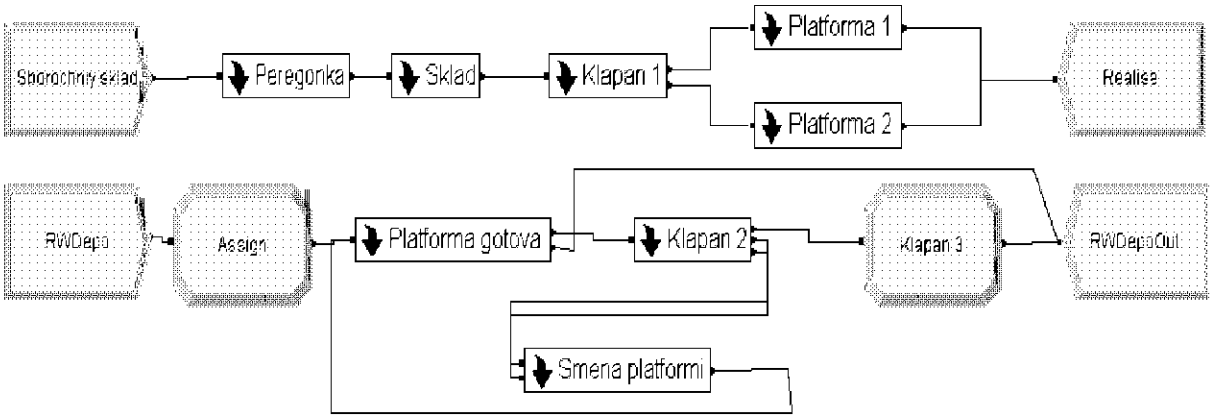


Рис. 4.33. Модель транспортного цеха в среде Arena.

4.4. Создание отчётов

Для того чтобы посмотреть отчет нужно (если модель еще не запускалась) в меню программы выбрать вкладку “Run” на ней команду “Go” или нажать на клавиатуре клавишу “F5”, если модель уже была запущена то для того чтобы просмотреть отчет нужно на панели “Project Bar “ выбрать вкладку “Reports”, а в ней соответствующее подменю рис. 4.34. Также отчёт может выводиться сразу после моделирования для того что бы это настроить нужно в меню программы выбрать вкладку “Run” на ней команду “Setup…” в открывшемся диалоговом окне выбрать вкладку “Reports”. Если на этой вкладке выбрать “Always” то отчёт будет выводиться без выдачи каких либо сообщений сразу после процесса моделирования и будет выведен тот отчет который будет указан в выпадающем списке “Default Report”, если выбрать пункт “Never” то отчёт будет создан но не выведется если в этом пункте поставить галочку напротив сообщения “Disable generation of report database” то отчёт не будет создан, пункт “Prompt me”такой же как “Always” только после процесса моделирования выдается сообщение о том выводить отчет или нет.

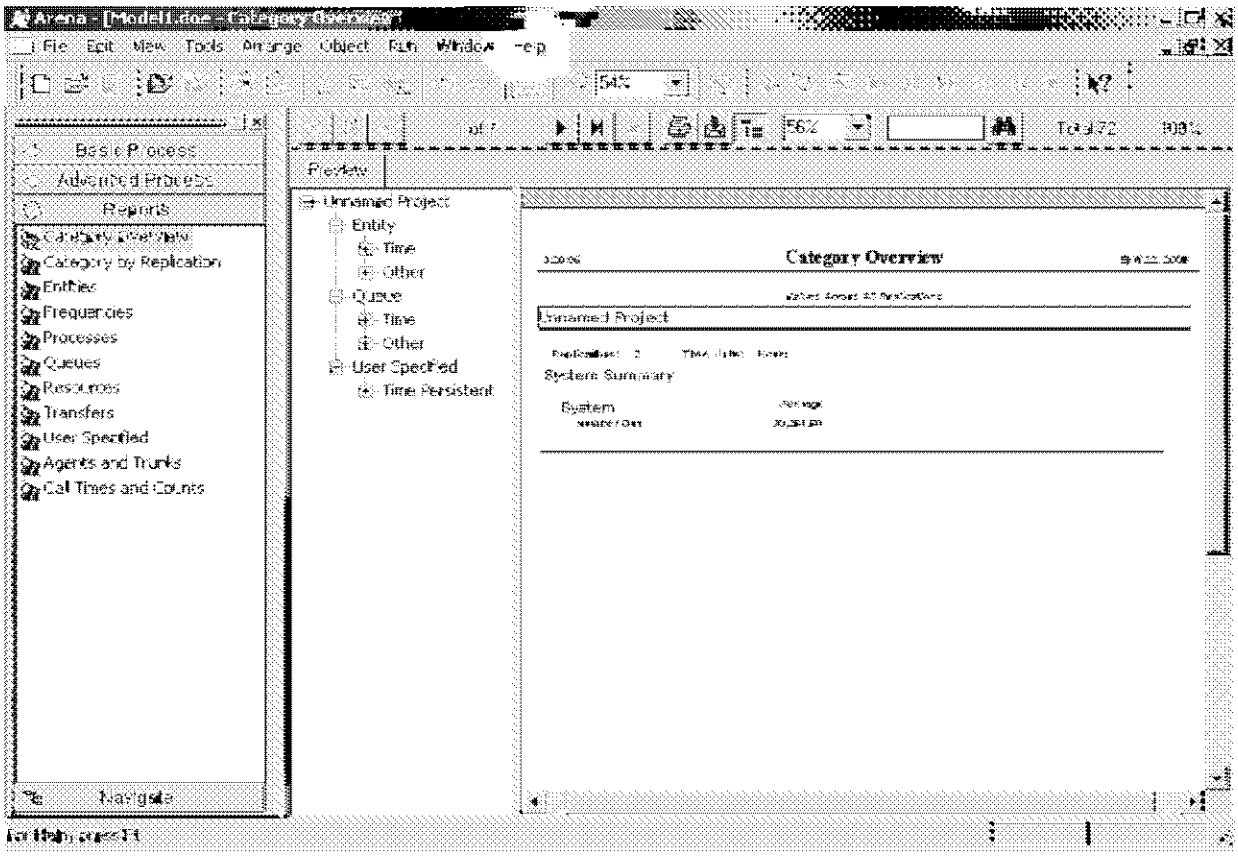


Рис. 4.34. Отчет о процессе моделирования

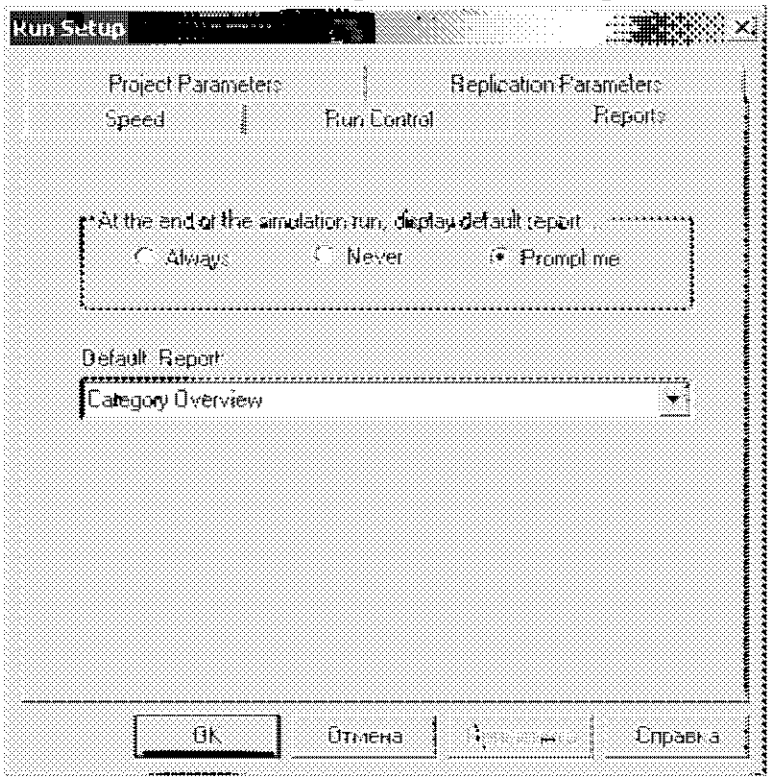


Рис. 4.35. Настройка вывода отчётов

Эти настройки будут действовать только для данной модели и не повлияют на созданные вновь модели. Для того чтобы настройки сохранились по умолчанию нужно в меню программы выбрать вкладку “Tools” на ней команду “Options…” в открывшемся диалоговом окне выбрать вкладку “Reports”. Изменения сделанные на этой вкладке повлияют на вновь создаваемые модели.

Также в этом меню делаться настройки того, что будет отображаться в отчете.

4.5. Основные преимущества системы моделирования Arena

Естественная и целостная методика моделирования.

Многие пакеты моделирующего ПО требуют от пользователей изучения и применения различных непростых методик, зависящих от сложности конкретного приложения. Arena использует для построения модели знакомую технологию блок-схем вне зависимости от сложности моделируемого приложения.

Возможности расширения и настройки.

Стандартная открытая архитектура Arena позволяет создавать собственные интерфейсы и приложения. По мере развития использования моделирования на предприятии профессиональный выпуск Arena способен обеспечить ему платформу разработки для создания собственных комплектов адаптированных инструментов моделирования.

Масштабируемая архитектура.

Arena предлагает мощные средства моделирования с естественным графическим интерфейсом. Спроектированная под потребности начинающих, но достаточно мощная, чтобы удовлетворить требования наиболее опытных пользователей, система Arena адаптируется под моделируемое приложение и уровень опытности пользователя.

Открытая архитектура.

Для усиления значимости моделирования и использования в нем существующих корпоративных информационных активов продукты Arena предоставляют встроенные гибкие интерфейсы с распространенными приложениями, а также встроенные средства Microsoft Visual Basic for Applications (VBA) для создания собственных интерфейсов. Например, с помощью встроенного в Arena шаблона Visio можно перенести логику и данные модели из рисунков Visio. Кроме того, данные в модели можно

внести непосредственно из Microsoft Excel или с помощью удобного интерфейса данных в виде электронной таблицы.

Полная поддержка проекта моделирования.

Arena предоставляет полный набор инструментов для поддержки операций проекта моделирования на протяжении всего его жизненного цикла, в числе которых входят:

- статистический анализ входных данных;
- обширная библиотека шаблонов и конструкций моделирования;
- средства анимации для визуализации работы процесса и обмена информацией о поведении системы с другими сотрудниками;
- инструменты анализа выходных данных для статистического сравнения результатов прогонов модели и принятия решения;
- средства оптимизации для автоматического определения оптимальных решений по выбранным сценариям;
- возможности распространения модели в процессе выполнения для ее просмотра или исследования другими сотрудниками.

5. Моделирование работы комплекса лесных машин

5.1. Постановка задачи

Рассматриваемый участок лесозаготовки имеет следующую схему работы. Харвестер (тимбержек-618) срезает до 250 кубометров за смену и складывает их в пачки, которые транспортируются тремя трелевочниками (ЛТ-154). Затем выполняется обрезка сучьев и погрузка хлыстов на лесовозы.

Необходимо построить имитационную модель работающего участка лесозаготовки с целью анализа эффективности данной схемы работы и изучение вариантов повышения производительности за счет имеющейся техники.

Формализованное описание системы

На первом этапе необходимо построение формальной модели с целью более глубокого понимания взаимосвязей между элементами модели и их однозначного отражения с помощью стандартизованных нотаций. В качестве математического аппарата для формализованного описания данной системы используем Е-сети (расширение сетей Петри).

Общая схема Е-сети для данной модели участка лесозаготовки приведена на рис. 5.1.

G_{E1} - складирование хлыстов харверстером в пачки, b_{3-5} - погрузка пачек на трелевочники, b_{6-8} - транспортировка, b_{9-11} - разгрузка, b_{16} - работа сучкорезных машин; b_{18-20} - холостой ход; R_{E1} - общий объем срезанной древесины; R_{E2} - общий объем пачек, поступивших к сучкорезной машине; R_{E3} - объем заготовленной древесины; A_{E1}, A_{E2}, A_{E3} - терминаторы; описатели меток: $M(b_i(1))$ - объем перевозимой трелевочником пачки, $M(b_i(2))$ - номер трелевочника.

Как видно из схемы с помощью аппарата Е-сетей довольно наглядно можно отобразить динамику параллельных процессов в моделируемой системе.

Для более подробного пояснения данной схемы рассмотрим некоторые отдельные ее элементы.

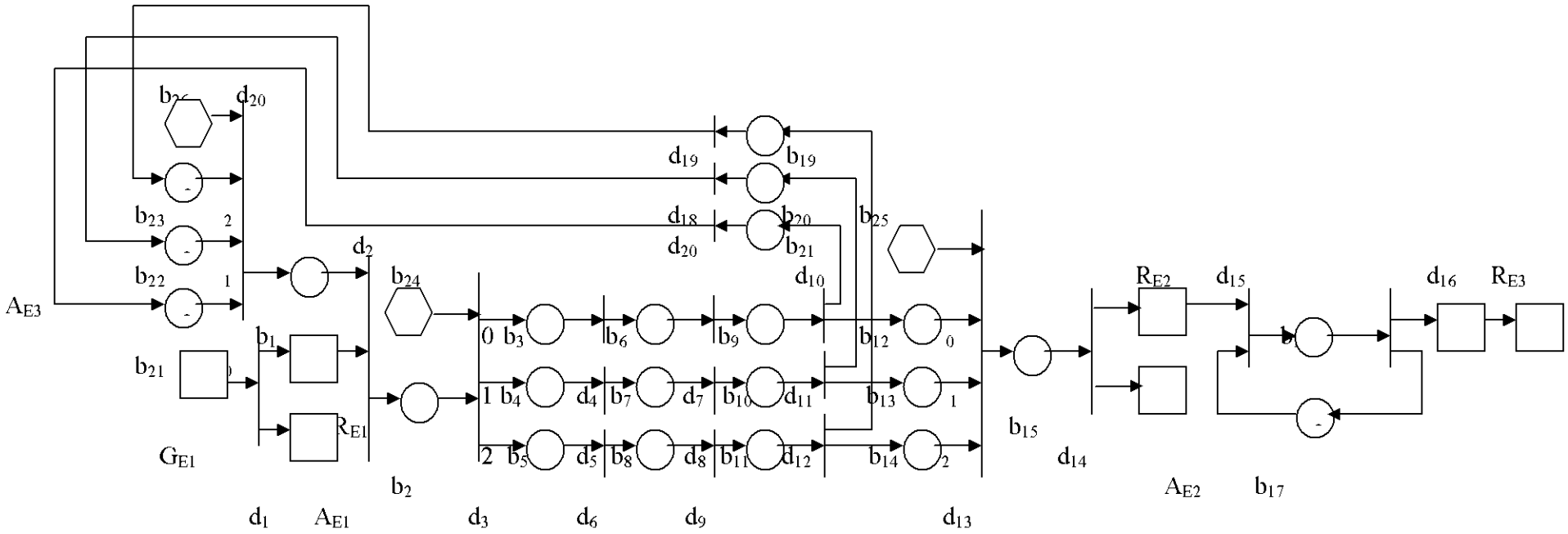


Рис. 5.1. Е-сеть, моделирующая участок лесозаготовки

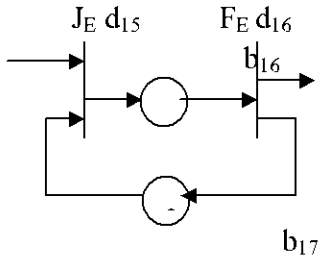


Рис. 5.2. Модель сучкорезной машины

На рис. 5.2 представлена модель сучкорезной машины. Маркировка позиции показывает, что машина свободна. Переход d_{15} (типовой переход объединения J_E) может сработать, когда есть сырье и свободна машина (позиция b_{17}), в этом случае будут метка будет помещена в позицию b_{16} - из всех входных позиций метки будут удалены, т.е. это имитирует обработку древесины сучкорезной машиной. По окончании обработки через срабатывание перехода d_{16} (типовой переход разветвления F_E) метка изымается из позиции b_{16} и размещается по двум направлениям: одно направление - перемещение сырья по этапам обработки, второе - помещение в позицию b_{17} (машина свободна).

Второй элемент модели, имитирующий погрузку трелевочника, представлен на рис. 5.3.

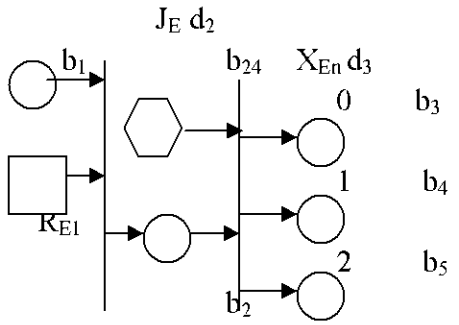


Рис. 5.3. Модель погрузки трелевочника

В данном элементе схемы маркировка позиции b_1 означает готовность трелевочника к погрузке. В случае, если набралось достаточное количество хлыстов для погрузки и трелевочник готов к работе, срабатывает переход d_2 (типовой переход объединения J_E), после чего метка помещается в позицию b_2 (погрузка трелевочника) - из всех входных позиций изымается по метке. После окончания погрузки в зависимости от маркировки позиции b_{24}, значение которой зависит от номера трелевочника, метка через макропереход X_{En} (создан на основе типового управляемого перехода разветвления X_E) перемещается в одну из позиций $b_{3\text{-}5}$, т.е. начало перевозки трелевочником древесины.

Ниже приведен символьный вариант представления схемы, приведенной на рис. 5.1.

$d_1=(F_E(G_{E1}, R_{E1}, A_{E1}); 0; |R_{E1}|:=|R_{E1}|+M(G_{E1}(2)))$

$d_2=(J_E(b_1, R_{E1}, b_2); 0; |R_{E1}|:=|R_{E1}|- f_1(M(b_1(2)), M(b_2(1)):= f_1(M(b_1(2)))$

$d_3=(X_E(b_{24}, b_2, b_3, b_4, b_5); 0; -)$

$d_4=(T_E(b_3, b_6); f_2(M(b_3(2)), M(b_3(1))); -)$

$d_5=(T_E(b_4, b_7); f_2(M(b_4(2)), M(b_4(1))); -)$

$d_6=(T_E(b_5, b_8); f_2(M(b_5(2)), M(b_5(1))); -)$

$d_7=(T_E(b_6, b_9); f_3(M(b_6(2))); -)$

$d_8=(T_E(b_7, b_{10}); f_3(M(b_7(2))); -)$

$d_9=(T_E(b_8, b_{11}); f_3(M(b_8(2))); -)$

$d_{10}=(F_E(b_9, b_{12}, b_{18}); f_4(M(b_9(1)), M(b_9(2)))); -)$

$d_{11}=(F_E(b_{10}, b_{13}, b_{19}); f_4(M(b_{10}(1)), M(b_{10}(2))); -)$

$d_{12}=(F_E(b_{11}, b_{14}, b_{20}); f_4(M(b_{11}(1)), M(b_{11}(2))); -)$

$d_{13}=(Y_{E3}(b_{25}, b_{12}, b_{13}, b_{14}, b_{15}); 0; -)$

$d_{14}=(F_E(b_{15}, R_{E2}, A_{E2}); 0; |R_{E2}|:=|R_{E2}|+M(b_{15}(1)))$

$d_{15}=(J_E(R_{E2}, b_{17}, b_{16}); 0; M(b_{16}(1)):= f_5, |R_{E2}|:=|R_{E2}|- M(b_{16}(1)))$

$d_{16}=(F_E(b_{16}, R_{E3}, b_{17}); f_6(M(b_{16}(1))); |R_{E3}|:=|R_{E2}|- f_6(M(b_{16}(1))))$

$d_{17}=(T_E(b_{18}, b_{21}); f_8(M(b_{18}(2))); -)$

$d_{18}=(T_E(b_{19}, b_{22}); f_8(M(b_{19}(2))); -)$

$d_{19}=(T_E(b_{20}, b_{23}); f_8(M(b_{20}(2))); -)$

Детализация G_{E1} (рис. 5.4):

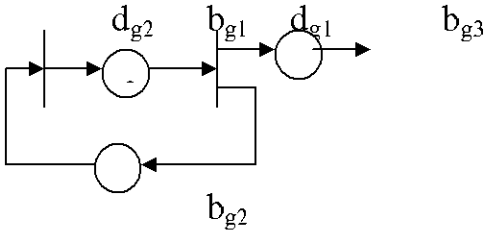


Рис. 5.4. Модель формирования пачек

$d_{g1}=(F_E(b_{g1}, b_{g2}, b_{g3}); f_9; M(b_{g3}(1)):= f_{10})$

$d_{g2}=(T_E(b_{g2}, b_{g1}); 0; -)$

Решающие процедуры:

$r(b_{24})=[M(b_2(2))=0\rightarrow M(b_{24}):=0; M(b_2(2))=1\rightarrow M(b_{24}):=1; M(b_2(2))=2\rightarrow M(b_{24}):=2]$

$r(b_{25})=[M(b_{25}):=0]$

$r(b_{26})=[M(b_{26}):=0]$

$f_1(x_1)$ - функция объема древесины, перевозимой трелевочником (x_1 - тип трелевочника);

$f_2(x_1, x_2)$ - функция времени погрузки на трелевочник (x_1 - тип трелевочника, x_2 - объем погружаемой древесины);

$f_3(x_1)$ - функция времени транспортировки;

$f_4(x_1, x_2)$ - функция времени разгрузки;

f_5 - функция объема древесины, загружаемой в окорочную машину;

$f_6(x_2)$ - функция времени обработки окорочным станком;

$f_7(x_2)$ - объем выхода из окорочного станка;

$f_8(x_1)$ - функция времени на обратную дорогу транспорта;

f_9 - функция времени поступления древесины на погрузку;

f_{10} - функция объема поступаемой на погрузку древесины.

5.3. Реализация программной модели

Полученная формальная модель может быть реализована с помощью специализированных средств моделирования GPSS и Arena (рис. 5.5).

Приведем анализ соответствия между элементами Е-сети и объектов GPSS:

1. Метка Е-сети может быть представлена динамическим объектом GPSS, т.е. транзактом.

2. Описатели меток аналогичны параметрам транзактов GPSS.

3. Позиция Е-сети идентична накопителю единичной емкости или каналу.

4. Решающие позиции $b_i \in B_R$ в зависимости от принадлежности к множеству B_P реализуются двумя способами: а) если $b_i \in B_P$, b_i эквивалентна набору объектов типа булевых переменных; б) если $b_i \in B/B_P$, то b_i может быть представлена накопителем единичной емкости или каналом.

5. Временные параметры переходов $t(d_i)$ реализуются ADVANCE.

6. Операции вычисления предикатов соответствует применение блока TEST, изменяющего маршруты транзактов в сочетании с булевыми переменными.

7. Операции процедуры перехода над описателями меток выполняются с помощью блоков ASSIGN в сочетании с арифметическими переменными.

8. Хранение значений описателей меток и ресурсов можно имитировать путем записи значений параметров транзактов в ячейки хранимых значений с помощью блока SAVEVALUE.

9. Процессы синхронизации движения меток через переход и удаления меток из решающей позиции $b_i \in B_R$ могут быть обеспечены с помощью логических переключателей (LOGIC S и LOGIC R).

10. Макропозиция генератора аналогична блоку GENERATE.

11. Макропозиция поглощения функционально идентична блоку TERMINATE.

12. Макропозиция очереди может интерпретироваться в GPSS записью транзакта в цепь пользователя.

В процессе реализации модели потребовалось уточнение следующих характеристик:

Средний объем одного дерева - 0,35 кбм;

Средний объем древесины, перевозимый одним трелевочником - 10 кбм;

Суммарное время погрузки на трелевочник, транспортировки к сучкорезным машинам и разгрузки составляет в среднем 65 мин;

Время обратной дороги трелевочника в порожнем состоянии составляет в среднем 15 мин;

Одна сучкорезная машина обрабатывает в среднем 0,6 кбм в минуту.

Параметры модели были получены в ходе статистического анализа результатов работы реальной системы. Было выявлено, что колебания объема древесины наиболее точно отражаются нормальным законом распределения. Объем древесины, перевозимый одним трелевочником, также является случайной величиной, имеющей почти нормальное распределение. Интервалы времени погрузки на трелевочник, транспортировки к сучкорезным машинам и разгрузки является случайной величиной, которую наиболее точно описывает показательный закон распределения. Случайные интервалы времени обратной дороги также распределены по показательному закону. Экспоненциальный закон также наиболее точно описывает случайные интервалы времени обработки на сучкорезной машине. За интервал времени моделирования в ходе эксперимента на имитационной модели участка лесозаготовки был выбран 1 год.

Приведем листинг модели на GPSS:

```
simulate

rez_tab table x$rez_sklad,5,10,10
zapas_tab table x$zapas,5,20,10

stanok storage 2
```

```
generate ,,,1
assign 1,1
transfer ,met1

generate ,,,1
assign 1,2
transfer ,met1

generate ,,,1
assign 1,3
transfer ,met1

generate 1
savevalue zapas+,(normal(3,0.35,0.2))
tabulate zapas_tab
terminate

met1 assign 2,(normal(1,10,2))
   queue harvest
   test ge x$zapas,p2
   depart harvest
   savevalue zapas-,p2
   advance (exponential(1,0,65))
   savevalue rez_sklad+,p2
   tabulate rez_tab
   assign 2,0
   advance (exponential(1,0,15))
   transfer ,met1

  generate ,,,2
met2  assign 2,(normal(1,0.35,0.2))
  queue suchkorez
  test ge x$rez_sklad,p2
  depart suchkorez
  savevalue rez_sklad-,p2
  enter stanok
  advance (exponential(1,0,1.6))
```

```
  leave stanok
  savevalue treat_sclad+,p2
  assign 2,0
  transfer ,met2

out generate 129600
   terminate 1
```

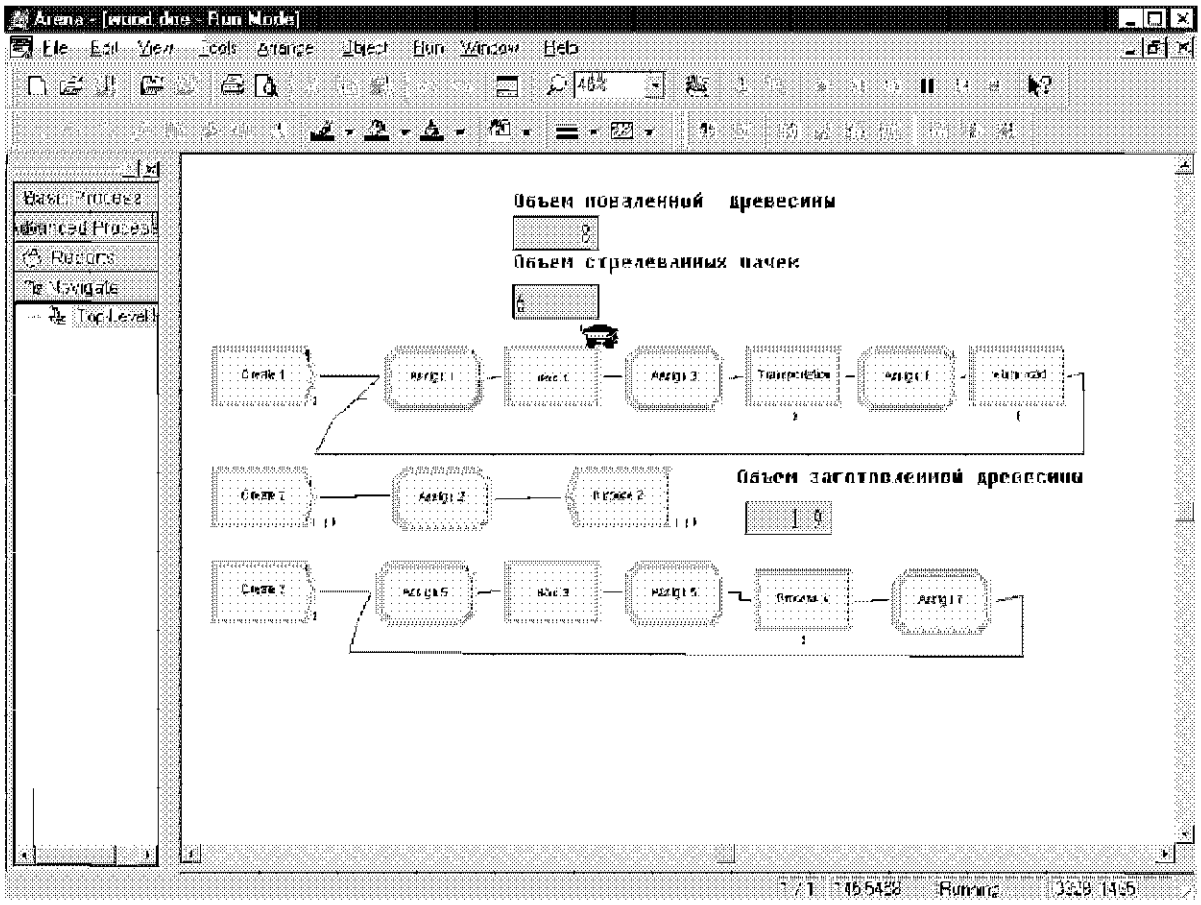


Рис. 5.5 Модель в системе Arena

5.4. Анализ результатов

В результате многократного прогона программной модели были получены следующие статистические данные:

1. Распределение объема межоперационного запаса деревьев на лесосеке показано на рис. 5.6. По оси X отражен объем, по оси Y - частоты попадания в тот или иной диапазон. Как видно из распределения в основ-

ном объем межоперационного запаса деревьев на лесосеке составлял от 0 до 65 кбм. Средний объем древесины межоперационного запаса деревьев на лесосеке составил приблизительно 28 кбм.

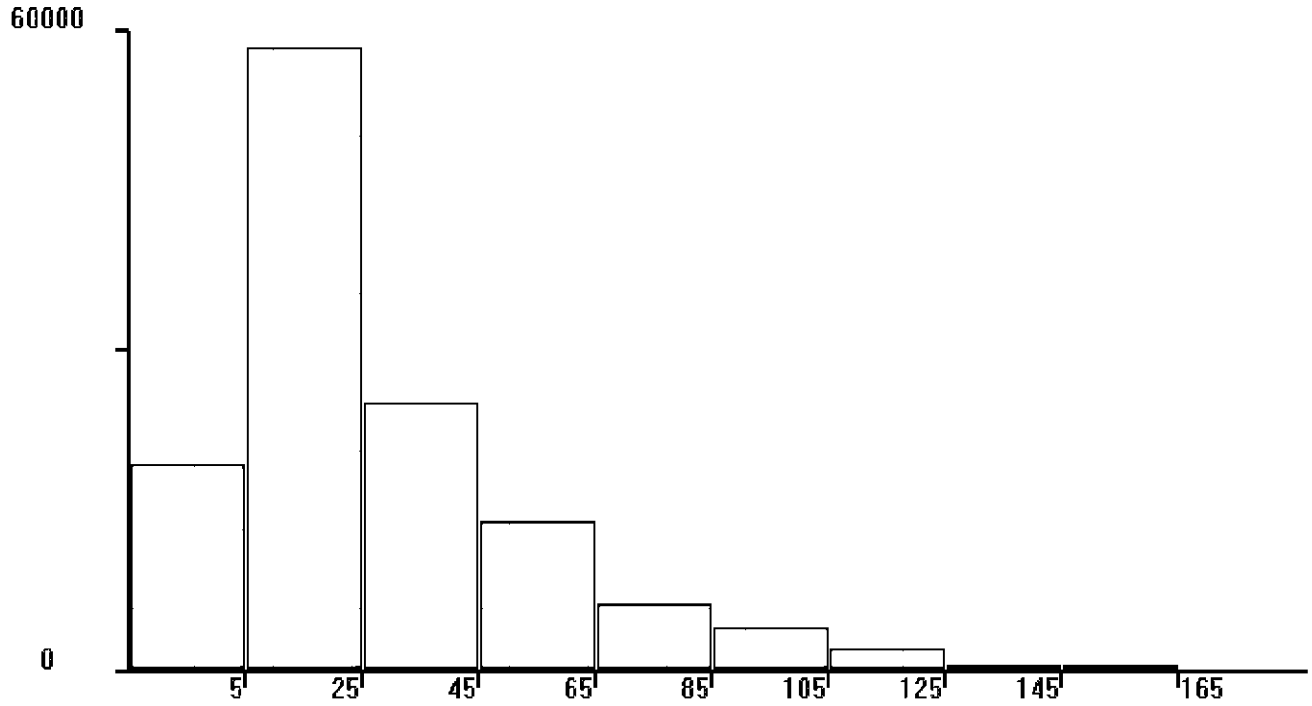


Рис. 5.6. Распределение объема межоперационного запаса сучкорезной машины

2. Максимальная длина очереди трелевочников, ожидающих погрузки, равняется 3, в то же время средняя длина меньше 1. В среднем трелевочники ожидали погрузки порядка шести минут.

3. Распределение объема межоперационного запаса сучкорезной машины показано на рис. 5.7. По оси X отражен объем, по оси Y - частоты попадания в тот или иной диапазон. Как видно из распределения в основном объем межоперационного запаса составлял от 5 до 35 кбм. Средний объем межоперационного запаса сучкорезной машины составил приблизительно 22 кбм.

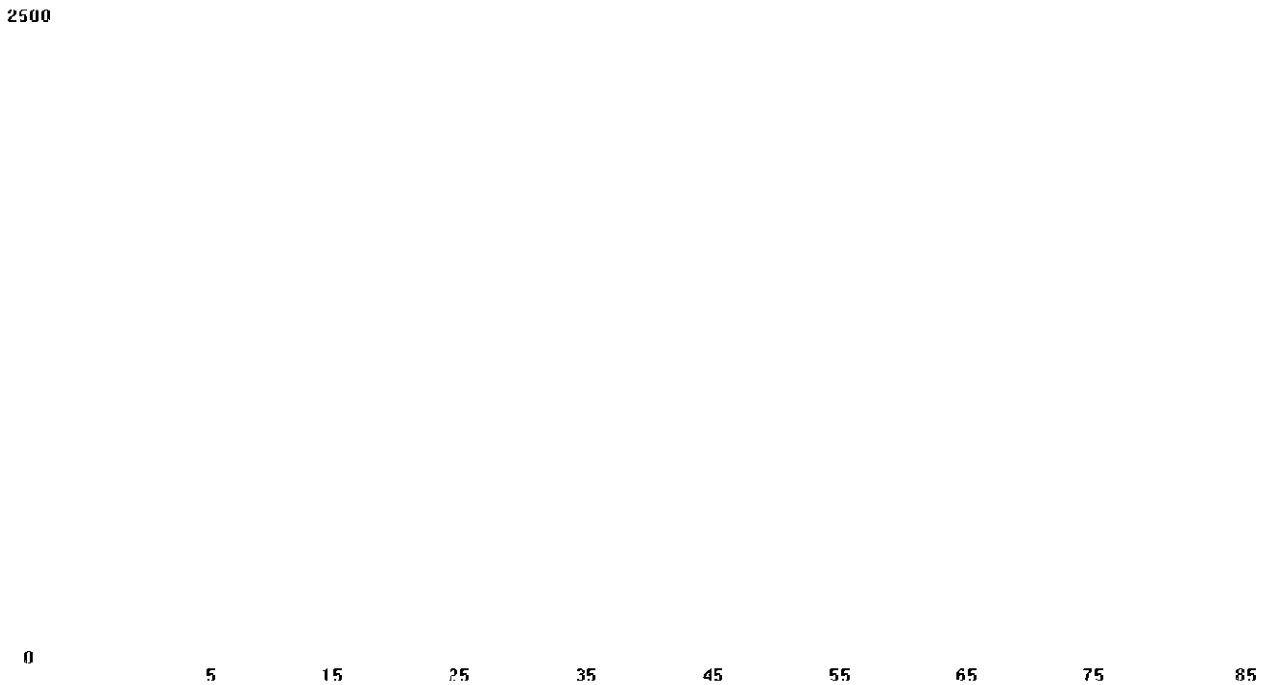


Рис. 5.7. Распределение объема межоперационного запаса сучкорезной машины

4. Суммарный коэффициент загрузки двух сучкорезных станков составил 80%.

В целом можно сделать следующие выводы: построенная модель работает стабильно, вероятность переполнения на каком-либо участке практически равна нулю (в модели не учитывались возможные поломки оборудования), объем складируемой древесины не превышает обычных норм технологических запасов. Необходимо отметить, что данные результаты получены на модели, не учитывающей случаи выхода оборудования из строя, что значительно бы ухудшило настоящие результаты. В целом данные, полученные на модели, соответствуют реальным показателям, что подтверждает адекватность созданной программной модели. Данная модель позволяет при каких-либо изменений внешних условий в реальной системе проанализировать появившуюся ситуацию и получить варианты адекватных изменений производственной схемы. Например, с помощью данной системы можно проанализировать целесообразность замены двух старых трелевочников одним новым с улучшенными производственными характеристиками. Модель позволяет также проанализировать возможность увеличения объемов переработки при сохранении существующей производственной схемы.

5.5. Варианты модернизации

На данном участке лесозаготовки могут быть реализованы такие технологические схемы, как харвестер+форвардер или харвардер. Рассмотрим формальные модели данных технологических схем.

1. Комплекс машин с технологической схемой харвестер+форвардер (рис. 5.8)

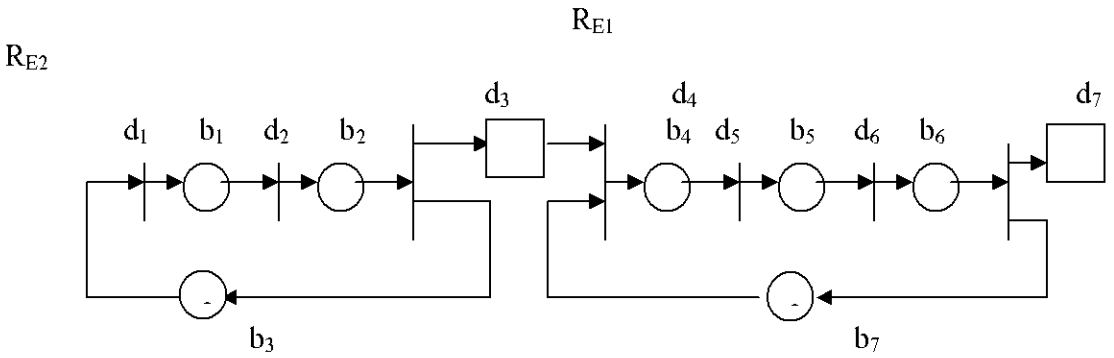


Рис. 5.8. Модель комплекса машин с технологической схемой харвестер+форвардер

b_3 – валка дерева, b_1 – обрезка сучьев, b_2 – заготовка сортиментов; R_{E1} - общий объем заготовленных сортиментов; b_4 – загрузка сортиментов на форвардер, b_5 – транспортировка сортиментов, b_6 – разгрузка сортиментов, b_7 – обратная дорога форвардера, R_{E2} - общий объем сортиментов на погрузочном пункте; описатели меток: $M(b_i(1))$ - объем древесины на соответствующей операции.

$d_1 = (T_E(b_3, b_1); f_1(M(b_3(1))); -)$

$d_2 = (T_E(b_1, b_2); f_2(M(b_1(1))); M(b_2(1)) := f_3(M(b_1(1)))$

$d_3 = (F_E(b_2, R_{E1}, b_3); f_4(M(b_2(1))); |R_{E1}| := |R_{E1}| + f_5(M(b_2(1))), M(b_3(1)) := f_6)$

$d_4 = (J_E(R_{E1}, b_7, b_4); f_{13}; M(b_4(1)) := f_7, |R_{E1}| := |R_{E1}| - M(b_4(1)))$

$d_5 = (T_E(b_4, b_5); f_8(M(b_4(1))); M(b_5(1)) := f_9(M(b_4(1))))$

$d_6 = (T_E(b_5, b_6); f_{10}; -)$

$d_7 = (F_E(b_6, R_{E2}, b_7); f_{11}(M(b_6(1))); |R_{E2}| := |R_{E2}| + f_{12}(M(b_6(1))), M(b_7(1)) := 0)$

$f_1(x_1)$ - функция времени валки дерева (x_1 - объем древесины);

$f_2(x_1)$ - функция времени обрезки сучьев;

$f_3(x_1)$ - функция потерь при обрезке сучьев;

$f_4(x_1)$ - функция времени заготовки сортиментов;

$f_5(x_1)$ - функция потерь при заготовке сортиментов;

f_6 - функция объема дерева к валке;

f_7 - функция объема сортиментов, погружаемых на форвардер;

$f_8(x_1)$ - функция времени погрузки сортиментов на форвардер;

$f_9(x_1)$ - функция потерь при погрузке сортиментов;

f_{10} – функция времени транспортировки сортиментов;

$f_{11}(x_1)$ - функция времени разгрузки сортиментов;

$f_{12}(x_1)$ - функция потерь при разгрузке сортиментов;

f_{13} – функция времени холостого хода форвардера.

2. Лесозаготовки с использованием харвардера (рис. 6).

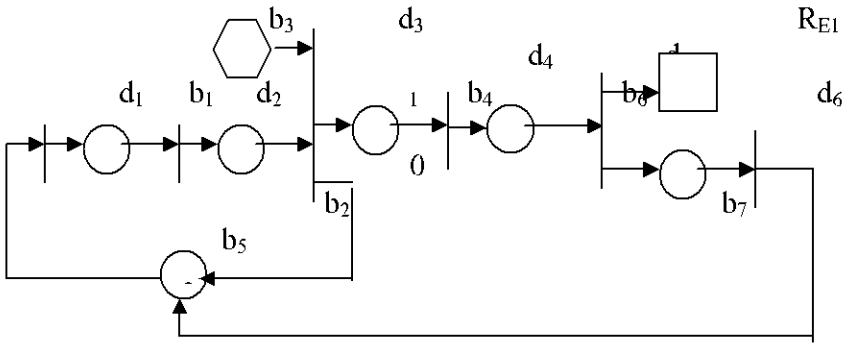


Рис. 5.9. Модель лесозаготовки с использованием харвардера

b_5 – валка дерева, b_1 – обрезка сучьев, b_2 – заготовка сортиментов; b_4 – транспортировка сортиментов; b_6 – разгрузка сортиментов; b_7 – обратная дорога; R_{E1} - общий объем сортиментов на погрузочном пункте; описатели меток: $M(b_i(1))$ - объем древесины на соответствующей операции; $M(b_i(2))$ – объем древесины на харвардере.

Решающая процедура:

$r(b_3)=[M(b_2(2))<f_1 \rightarrow M(b_3):=0;\ M(b_2(2))>=f_1 \rightarrow M(b_3):=1]$

$d_1=(T_E(b_5, b_1);\ f_1(M(b_5(1)));\ -)$

$d_2=(T_E(b_1, b_2);\ f_2(M(b_1(1)));\ M(b_2(1)):= f_3(M(b_1(1)))$

$d_3=(X_E(b_3,b_2, b_4, b_5);\ f_4(M(b_2(1)));\ M(b_2(2)):=M(b_2(2))+ f_5(M(b_2(1))),$ $M(b_5(1)):= f_{10})$

$d_4=(T_E(b_4, b_6);\ f_6;\ -)$

$d_5=(F_E(b_6, R_{E1}, b_7);\ f_7(M(b_6(2)));\ |R_{E2}|:=|R_{E2}|+f_8(M(b_6(2))),\ M(b_7(2)):=0)$

$d_6=(T_E(b_7, b_5);\ f_9;\ M(b_5(1)):= f_{10})$

$f_1(x_1)$ - функция времени валки дерева (x_1 - объем древесины на соответствующей операции);

$f_2(x_1)$ - функция времени обрезки сучьев;

$f_3(x_1)$ - функция потерь при обрезке сучьев;

$f_4(x_1)$ - функция времени заготовки и загрузки сортиментов;

$f_5(x_1)$ - функция потерь при заготовке сортиментов;

f_6 - функция времени транспортировки;

$f_7(x_1)$ – функция времени разгрузки сортиментов;

$f_8(x_1)$ - функция потерь при разгрузке сортиментов;

f_9 - функция времени на обратную дорогу;

f_{10} – функция объема срезаемого дерева;

Таким образом, с помощью аппарата Е-сетей наглядно и однозначно отображается динамика параллельных процессов функционирования лесных машин.

Список литературы

1. Бусленко Н.П. Моделирование сложных систем. - М.: Наука, 1978. - 400 с.
2. Дягтерев Ю.И. Исследование операций. - М.: Высш. шк., 1986. - 320 с.
3. Емельянов А.А., Власова Е.А., Дума Р.В. и др. Имитационное моделирование экономических процессов. - М.: Финансы и статистика, 2002. - 368 с.
4. Кульгин М. Технологии корпоративных сетей. Энциклопедия. - СПб.: Питер, 1999. - 704 с.
5. Марков А.А. Моделирование информационно-вычислительных процессов. Учеб. пособие для вузов. - М.: Изд-во МГТУ им. Н.Э. Баумана, 1999. - 360 с.
6. Моделирование лесозаготовительных процессов на GPSS: Методические указания к выполнению лабораторных работ по имитационному моделированию / Гурьев А.Т. Сизов Д.Л., Остапенко Е.А., Остапенко А.А. - Архангельск: РИО АГТУ, 1995. - 64 с.
7. Питерсон Дж. Теория сетей Петри и моделирование систем: Пер. с англ. - М.: Мир, 1984. - 264 с., ил.
8. Применение микропроцессорных средств в системах передачи информации: Учеб. пособие для вузов/Б.Я. Советов, О.И. Кутузов, Ю.А. Головин, Ю.В. Аветов. - М.: Высш. шк., 1987. - 256 с.: ил.
9. Прицкер А. Введение в имитационное моделирование и язык СЛАМ II. - М.: Мир, 1987. - 646 с.
10. Руководство пользователя GPSS. - Казань: Изд-во "Мастер Лайн", 2002.
11. Советов Б.Я., Яковлев С.А. Моделирование систем. - М.: Высш. школа, 1998.-319 с.
12. Томашевский В.Н., Жданова Е.Г. Имитационное моделирование в среде GPSS. - М.: Бестселлер, 2003. - 416 с.

13. Учебное пособие по GPSS World. - Казань: Изд-во "Мастер Лайн", 2002. - 272 с.
14. Шеннон Р. Имитационное моделирование систем: искусство и наука. - М.: Мир, 1978. - 419 с.

Введение 3
1. Основы имитационного моделирования процессов лесного комплекса 4
1.1. Процессы лесного комплекса 4
1.2. Основные понятия и принципы имитационного моделирования 8
1.2.1. Основные понятия 8
1.2.2. Критерии применения имитационного моделирования 10
1.2.3. Проблемы применения имитационного моделирования 10
1.2.4. Процесс имитационного моделирования 12
1.2.5. Основные подходы к построению математических моделей систем 13
1.2.6. Непрерывно-стохастические модели (Q-схемы) 14
1.3. Сетевой подход к построению математических моделей систем 17
1.3.1 Общие понятия сетей Петри 17
1.3.2. Разработка моделей вычислительных сетей предприятий лесного комплекса 19
1.3.3. Моделирование лесозаготовительных процессов 22
1.3.4. Моделирование процессов кредитования предприятий лесного комплекса 24
1.3.5. Анализ сетей Петри 25
1.3.6. Е-сети 26
1.4. Статистическое моделирование 30
1.4.1. Общая характеристика метода статистического моделирования 30
1.4.2. Генерация последовательностей случайных чисел 34
1.4.3. Моделирование случайных событий 37
1.4.4. Моделирование дискретных случайных величин 40
1.4.5. Моделирование непрерывной случайной величины 41
1.5. Инструментальные средства моделирования 45

2. Принципы разработки современных средств имитационного моде-

лирования 48
2.1. Основные понятия и объекты системы моделирования 48
2.2. Подходы к разработке программных моделей 50
2.2.1. Событийный подход 50
2.2.2. Процессно-ориентированный подход 52
2.2.3. Подход сканирования активностей 53
2.3. Управление модельным временем 54
2.4. Механизм управления процессом моделирования 57
2.5. Обработка и анализ результатов имитационного моделирования 61
2.6. Технология визуального проектирования имитационных моделей 62
2.7. Пример разработки специализированного моделирующего комплекса для анализа вычислительных сетей 63
2.7.1. Предпосылки построения модели 64
2.7.2. Обобщенная схема вычислительной сети 66
2.7.3. Упрощения, принятые при построении модели 67
2.7.4. Понятие подсети 68
2.7.5. Входные данные модели 68
2.7.6. Алгоритм движения сообщения 69
2.7.7. Моделирование доступа к передающей среде 72
2.7.8. Разработка программной модели 72

3. Система имитационного моделирования GPSS World 77
3.1. Общие сведения 77
3.2. Интерфейс GPSS World 78
3.3. Объекты GPSS 79
3.3.1. Транзакты 80
3.3.2. Блоки 81
3.3.3. Устройства 81
3.3.4. Памяти (многоканальные устройства) 82
3.3.5. Логические ключи 82
3.3.6. Переменные 83
3.3.7. Функции 83
3.3.8. Ячейки и матрицы 83
3.3.9. Очереди 83

3.3.10. Таблицы 84
3.3.11. Списки пользователя 84
3.3.12. Числовые группы 84
3.3.13. Группы транзактов 84
3.3.14. Потоки данных 84
3.3.15. Стандартные числовые атрибуты 85
3.4. Таймер модельного времени 89
3.5. Стандартные отчеты 90
3.5.1. Общая информация 90
3.5.2. Имена 91
3.5.3. Блоки 91
3.5.4. Устройства 92
3.5.5. Очереди 92
3.5.6. Памяти 93
3.5.7. Таблицы 94
3.5.8. Списки пользователя 95
3.5.9. Группы транзактов 95
3.5.10. Числовые группы 96
3.5.11. Логические ключи 96
3.5.12. Ячейки 96
3.5.13. Матрицы 97
3.5.14. Список текущих событий 97
3.5.15. Список будущих событий 98
3.6. Формат операторов GPSS 98
3.7. Язык PLUS 99
3.7.1. Алфавит 99
3.7.2. Имена 99
3.7.3. Выражения 100
3.7.4. Типы данных 100
3.7.5. Факторы 100
3.7.6. Операторы 101
3.7.7. Встроенная библиотека процедур 102
3.8. Управление моделированием 105
3.9. Моделирование входящих и выходящих потоков. Блоки создания и уничтожения транзактов 106
3.10. Сегментация модели. Сегмент ограничения модельного времени 108
3.11. Моделирование приборов 110

3.12. Имитация обработки транзакта 110
3.13. Моделирование отказа устройства 111
3.14. Моделирование многоканального устройства (памяти) 114
3.15. Моделирование очередей 115
3.16. Параметры транзактов 115
3.17. Работа с ячейками и матрицами 116
3.18. Переменные 118
3.19. Генераторы случайных чисел 119
3.20. Функции 119
3.21. Переадресация потока транзактов 123
3.22. Моделирование с использованием блоков условного перехода 125
3.23. Организация таблиц 127
3.24. Логические ключи 128
3.25. Моделирование ансамблей событий 129
3.26. Организация списков пользователя 131
3.27. Использование приоритета транзакта 134
3.28. Переустановка значений генераторов случайных чисел 134

4. Система имитационного моделирования Arena 136
4.1. Основные модули для построения простейших моделей 136
4.1.1. Архитектура Arena 136
4.1.2. Построение моделей в Arena 136
4.2. Интерфейс и основные модули среды разработки Arena Model 138
4.2.1. Model Window 138
4.2.2. Понятие “модуль” 139
4.2.3. Автоматически создаваемые Data модули 140
4.2.4. Различие Data и Flowchart модулей 140
4.2.5. Использование панели Basic Process 140
4.3. Практическое применение основных элементов среды имитационного моделирования Arena 145
4.3.1. Краткие сведения об объекте моделирования 145
4.3.2. Выполнение модели в среде Arena 147
4.4. Создание отчётов 166
4.5. Основные преимущества системы моделирования

Arena 168

5. Моделирование работы комплекса лесных машин 170
5.1. Постановка задачи 170
5.2. Формализованное описание системы 170
5.3. Реализация программной модели 174
5.4. Анализ результатов 178
5.5. Варианты модернизации 180
Список литературы 183