МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования «Казанский (Приволжский)
федеральный университет»

Набережночелнинский институт (филиал)

Кафедра Бизнес-информатики и математических методов в экономике

Имитационное моделирование экономических процессов

Учебно-методическое пособие

Набережные Челны
2019 г.

УДК 510.636
ББК 22.18

Печатается по решению учебно-методической комиссии экономического отделения Набережночелнинского института (филиала) федерального государственного автономного образовательного учреждения высшего образования «Казанский (Приволжский) федеральный университет», от «24» января 2019г. (протокол №5)

Рецензенты:
Доктор физ.-мат. наук, профессор А.Г. Исавнин
Доктор экономических наук, профессор А.Н. Макаров

Розенцвайг А.К., Шарипов Р.Ш. Имитационное моделирование экономических процессов: учебно-методическое пособие / А.К. Розенцвайг, Р.Ш. Шарипов – Набережные Челны: Изд-во Набережночелнинского института КФУ, 2019. – 51 с.

Учебно-методическое пособие содержит последовательное изложение базовых понятий теории имитационного моделирования экономических процессов. Подробно изложены: введение в систему Micro Saint; развитие концепций: дополнительные методы и средства имитации. Функции, динамика моделирования, разработка многоуровневых структур программы.

Учебно-методическое пособие предназначено для использования в учебном процессе студентами технических направлений в экономике и экономического отделения дневной, заочной и дистанционной форм обучения.

Содержание

Введение

Одним из основных направлений, определяющих методологию, концептуальные и реализационные основы информационной технологии поддержки принятия управленческих решений, является имитационное моделирование. Методология имитационного моделирования основана на воспроизведении реальных или гипотетических бизнес-процессов в специальной компьютерной среде, образующей виртуальный мир предприятия, организации, производства и любого другого объекта управления.

Эта технология появилась в 60-х г. XX в., и на протяжении многих лет она не только остается одной из основных в исследовании операций, но и бурно развивается в области реинжиниринга бизнес-процессов и новых направлений искусственного интеллекта.

Основу этой технологии составляет компьютерный имитационный эксперимент, связанный с воспроизведением динамических процессов функционирования исследуемой системы. В процессе такого воспроизведения осуществляется наблюдение за функционированием модели и выявление «узких мест» в организации деятельности. Основными достоинствами этого метода являются:

возможность воспроизведения реальной системы с практически любым уровнем детальности;

повторяемость эксперимента;

возможность произвольной фрагментации и структуризации системы.

Технология имитационного моделирования излагается на основе использования системы Micro Saint, одной из наиболее простых и адекватных задачам исследования систем операционного и производственного менеджмента. Отличительными особенностями ее являются концептуальная завершенность, целостность и простота использования даже для исследования весьма сложных систем, является прекрасным инструментом для изучения концепций и технологии имитационного моделирования систем управления в различных областях производственного и операционного менеджмента.

Введение в систему Micro Saint

Этап конструирования и описания модели

Сетевая структура модели

Модель исследуемой системы должна быть представлена в виде сетевой структуры. Пример такой структуры приведен на рис. 1.

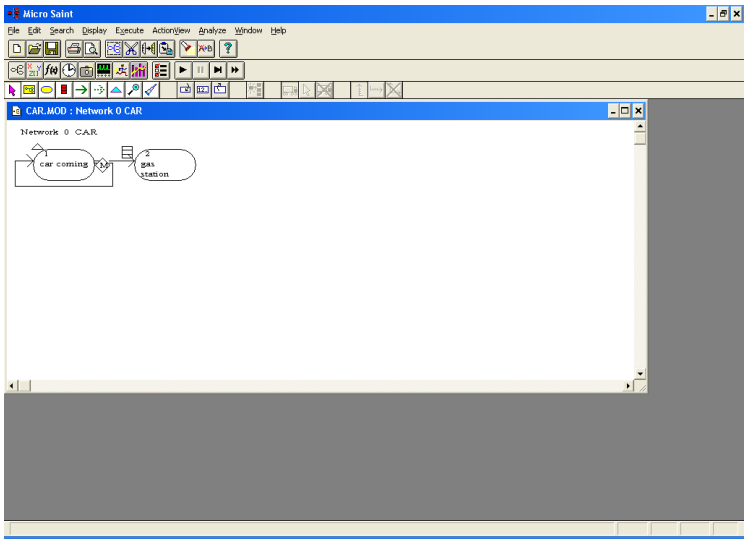


Рис.1. Сетевая структура модели

Овалы изображают блоки действий, стрелки - пути перемещения динамических объектов (тэгов), ромбы - разветвления таких путей, таблички изображают очереди тэгов, а маленький треугольник рядом с первым блоком определяет точку ввода тэгов в модель. Конструирование структуры модели связано с использованием «подсвеченных» кнопок нижней панели инструментов (см. рис. 2).

Рис. 2.

Пример 1: модель обслуживания автомобилей на заправочной станции

Станция оснащена одной бензоколонкой, перед которой может образовываться очередь. Тэг в этой модели исполняет роль автомобиля, а разветвитель используется для организации потока автомобилей. Буква M, стоящая внутри ромбика (первая буква слова Multiple - множественный),

означает, что любой тэг-автомобиль, приехавший на заправку, попадая в разветвитель, раздваивается (порождает копию). При этом основной тэг направляется в блок 2 (на обслуживание) а копия возвращается в первый блок, имитируя приезд другого автомобиля.

Разветвитель появляется в структуре модели автоматически при создании в блоке нескольких выходных стрелок связи с помощью инструментов панели (рис. 2).

Блок «Car coming» определяет приезд автомобилей на заправку, а блок «Gas station» - обслуживание автомобиля у бензоколонки. Любой тэг, «входящий» в блок, может задерживаться в этом блоке на определенное время. Такое время имитирует интервалы между приходами автомобилей на заправочную станцию и длительность процедуры заправки.

Описание элементов модели

Описание элементов модели связано с понятием переменной. Любая переменная используется для описания какой-либо характеристики системы, например, бензоколонка может находиться в одном из двух состояний:

• занята (идет заправка автомобиля),

• свободна (нет заправки - колонка простаивает).

Мы можем имитировать эти состояния с помощью переменной Status, которая будет принимать два значения:

Status:= 1; (колонка переходит в состояние «Занята»),

Status:= 0; (колонка переходит в состояние «Свободна»).

Оператор «:=» называется оператором присваивания, он назначает переменной Status то значение, которое записано справа от оператора присваивания. В этом примере Status - имя переменной, а 0 и 1 - возможные значения переменной. Во многих случаях перечислить все значения переменной трудно (или невозможно), поэтому в таких ситуациях переменную характеризуют типом, определяющим ее возможные значения.

Например, переменную N - количество автомобилей, обслуженных на

автозаправке, характеризуют типом Integer (целое число), а переменную V - количество заливаемого бензина - типом Real (действительное число).

В процессе имитации исследуемой системы переменные модели будут изменять свои значения, поэтому перед запуском модели всем переменным необходимо назначить исходные начальные значения (Initial Value), которые будут определять исходное состояние системы.

Все переменные модели должны быть внесены в список переменных.

Открытие такого списка реализуется нажатием кнопки  . При этом откроется окно списка переменных (рис. 3).

В этом списке представлены системные переменные. Такие переменные используются в любой модели, важнейшими из них для нас являются переменные clock - модельное время и tag (тэг) - индивидуальный номер динамического объекта.

Одновременно в модели может присутствовать много тэгов, все они совершают передвижения по блокам структурной схемы. Однако реализация таких передвижений происходит в определенной последовательности, при этом тэг, который в текущий момент времени стоит в голове такой последовательности, считается активным.

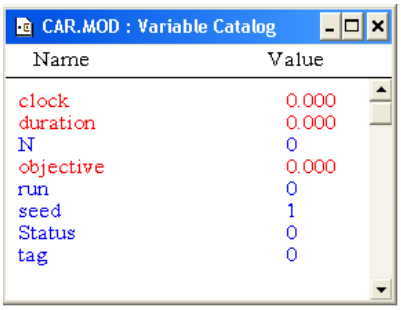


Рис. 3. Окно списка переменных

(Name - имя, Value - значение переменной)

Кроме системных в модели присутствуют переменные, которые определяются непосредственно пользователем применительно к его модели. В нашем случае это переменные Status - состояние бензоколонки и N - количество автомобилей, обслуженных на автозаправке. Для того чтобы ввести эти

переменные в список, необходимо нажать кнопку  (нижний ряд панели инструментов рис. 1, справа). При этом откроется окно рис. 4, в котором задается имя переменной, ее смысловое содержание, тип и начальное значение.

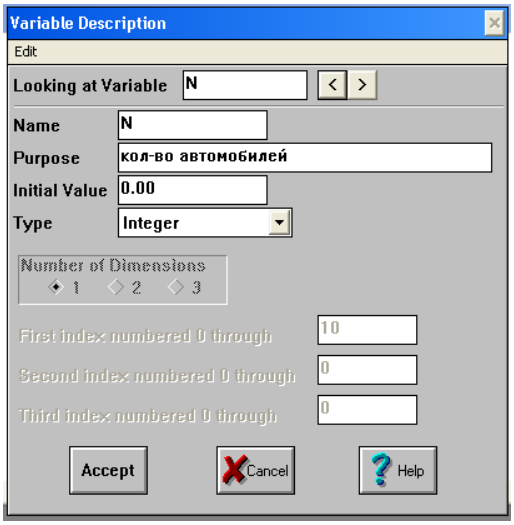


Рис. 4. Окно описания переменной

Нажатие кнопки Accept вводит переменную в список, только после этого она становится полноправным участником процесса моделирования.

Описание задачи

Окно описания задачи открывается после двойного клика на соответствующем блоке сетевой структуры модели.

Поля Time Distribution, Mean Time и Standard Deviation используются для задания интервала времени между приходами автомобилей на заправочную станцию. В списке Time Distribution содержатся различные законы распределения вероятностей. Для нашего примера мы использовали равномерное распределение (Rectungular) со средним значением 10 (мин) и отклонением от среднего, равным 3 (мин). (В поле Standard Deviation в этом случае записывается 10 - 3 = 7 (мин), т. е. нижняя граница равномерного распределения.) Такое задание приведет к тому, что в нашу модель будут поступать автомобили (тэги) через интервал времени (10±3) мин. Выбор в качестве единицы времени 1 мин. обусловлен только исследователем. В общем

случае это может быть произвольная единица (день, час и т. п.).

Поле Release Condition в общем случае содержит условие возможности входа тэга в соответствующий блок. Если в этом поле присутствует любое число, большее 0, вход в блок открыт (в нашем примере для блока «Car coming» любой автомобиль может въехать на заправочную станцию). Для блока «Gas station» в поле Release Condition должно быть размещено условие входа в этот блок: Status= =0, т.е. вход автомобиля в блок возможен только тогда, когда бензоколонка свободна. В противном случае автомобиль не сможет подъехать к бензоколонке и будет находиться в очереди (символ «= =» используется как отношение равенства).

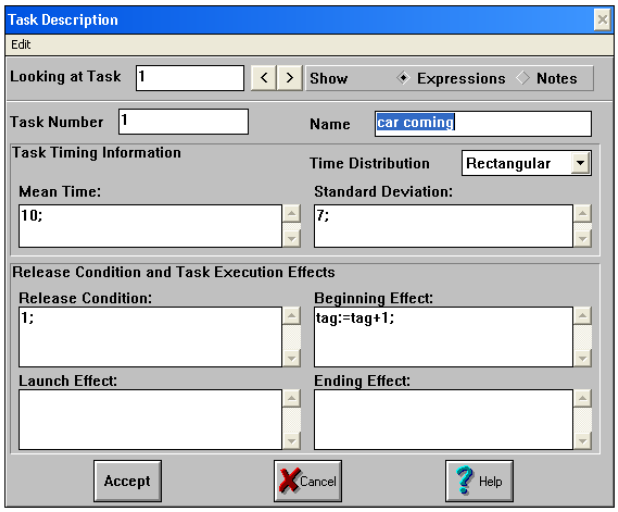


Рис. 5. Окно описания задачи

Поля Beginning Effect, Launch Effect и Ending Effect предназначены для размещения в них описаний действий, которые необходимо выполнить соответственно при входе тэга в блок, при прерывании его обслуживания в блоке и при выходе тэга из блока (поле Launch Effect в дальнейшем не используется).

В нашем примере для блока «Car coming» (рис. 5) мы использовали только поле Beginning Effect, в котором разместили оператор: tag:=tag+1. Этот оператор присваивания (см. выше) реализует алгоритм накопления: каждый раз при входе вновь приехавшего автомобиля в блок «Car coming» входной эффект будет увеличивать индивидуальный порядковый номер автомобиля на 1. Таким

образом, через структуру модели будут последовательно проводиться тэги-автомобили с номерами 0, 1, 2 и т. д.

В аналогичном окне для блока «Gas station» в полях входного и выходного эффектов будут размещены действия по изменению состояния бензоколонки. В поле входного эффекта определяется действие по занятию бензоколонки: Status:=1; а в поле выходного - по ее освобождению: Status:=0. Кроме того, в этом поле определяется действие, связанное с подсчетом количества автомобилей, «прошедших» через автозаправочную станцию: N:=N+1.

В полях Time Distribution, Mean Time и Standard Deviation блока «Gas station» следует указать характеристики времени заправки автомобиля (например, равномерное распределение в интервале 12 ± 6 мин.).

Исполнительная система, реализующая проводку тэгов через структуру модели, вычисляет описанные эффекты в следующем порядке:

• условие входа в блок (Release Condition);

• Beginning Effect;

• определение времени задержки в блоке (поля Time Distribution, Mean Time и Standard Deviation);

• Ending Effect

Описание регистратора очереди

Окно регистратора очереди вызывается двойным кликом на значке соответствующей очереди (табличка) в структуре модели.

Поле Sorting Order определяет порядок элементов (тэгов) в очереди:

• FIFO (First In – First Out, первым вошел - первым вышел);

• LIFO (Last In – First Out, последним вошел - первым вышел);

• Sorted (очередь упорядочена по значениям выражения, которое размещено в поле Priority).

В последнем случае из очереди для входа в блок задачи выводится тот тэг, для которого значение выражения в поле Priority максимально. Обычно это

выражение использует индивидуальный номер тэга.

Использование полей Entering Effect и Departing Effect аналогично использованию полей входного и выходного эффекта в окне описания задачи.

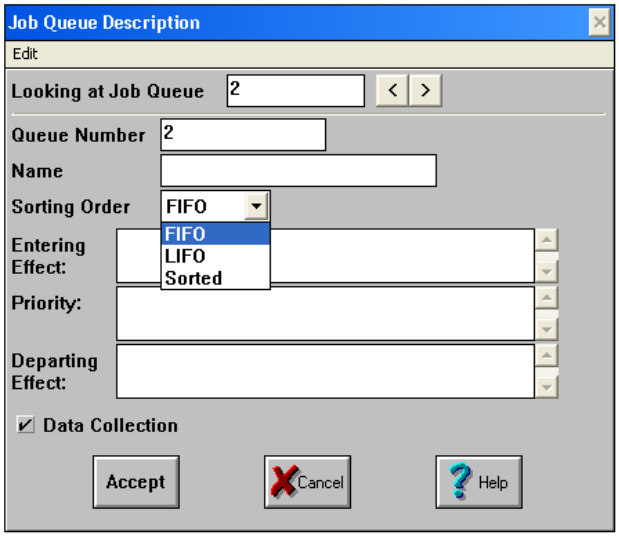


Рис. 6. Окно описания очереди

Использование полей Entering Effect и Departing Effect аналогично использованию полей входного и выходного эффекта в окне описания задачи.

Описание разветвителя

Разветвитель может использовать один из трех типов разветвления (Decision Type): Multiple, Probabilistic и Tactical. Первый тип разветвления уже обсуждался выше - он создает множество копий тэга, вошедшего в разветвитель, и отправляет их на все выходы разветвителя. Такой режим позволяет моделировать параллельные (одновременно протекающие) процессы в исследуемой системе.

Единицы в полях Routing Condition интерпретируются так же, как и в полях Release Condition описания задачи (см. рис. 5).

Тип Probabilistic определяет стохастический (случайный) механизм выбора тэгом направления дальнейшего движения. Для такого выбора в полях Routing Condition задаются вероятности переходов по выбранному направлению (их сумма должна быть равна 1).

Тип Tactical использует поля Routing Condition для записи выражений, значения которых определяют направление дальнейшего движения тэга. Тэг,

проходящий через разветвитель типа Tactical, будет направлен в ту задачу, для которой выражение, записанное в соответствующем поле Routing Condition, примет (в момент перехода) максимальное значение.

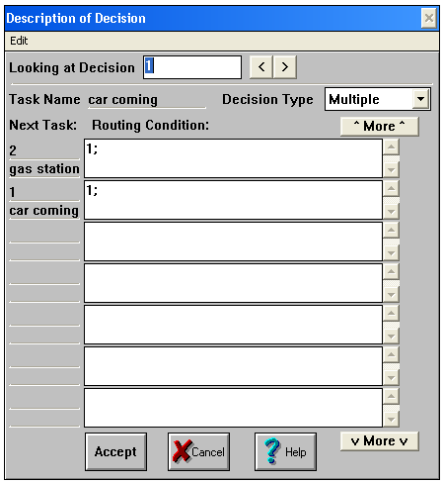


Рис. 7. Описание разветвителя

Этап проведения компьютерного эксперимента с моделью

Запуск модели

После составления описаний всех элементов модели она может быть запущена для выполнения задач. Для запуска и управления процессом выполнения модели используется правая часть средней панели инструментов (рис. 8).

Кроме кнопок этой панели можно также обратиться к разделу Execute главного меню системы.

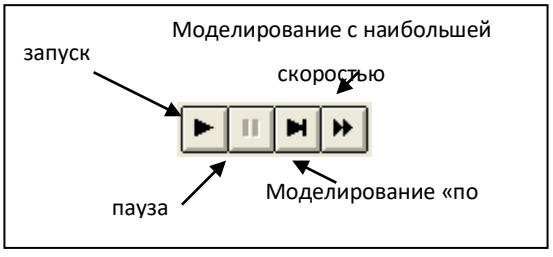


Рис. 8. Панель управления моделированием

Остановка модели

Существует два основных способа остановки модели: «вручную» (меню Execute, оператор Halt) и по определенному условию (достижение в модели определенного состояния).

Второй способ связан с размещением оператора остановки Halt() в определенных полях описания модели. Например, размещение оператора Halt() в поле Entering Effect очереди (см. рис. 6) приведет к тому, что, когда первый тэг войдет в очередь и произойдет вычисление этого эффекта, модель будет остановлена.

Сбор результатов компьютерного эксперимента

Целью любого компьютерного эксперимента является сбор информации о значениях переменных модели, наблюдаемых в процессе проведения эксперимента, и состояниях очередей, возникающих в процессе моделирования. Переменные, которые могут наблюдаться в эксперименте, составляют коллекцию переменных. Для одной и той же модели могут быть определены несколько разных коллекций, отличающихся друг от друга составом переменных и условиями, при которых осуществляется регистрация их значений.

Список коллекций открывается нажатием кнопки  . Если для разработанной модели не определено ни одной коллекции, этот список пуст (рис. 9).

Рис. 9. Список коллекций переменных

Для определения коллекции следует нажать кнопку  при открытом списке коллекций.

В поле Document Name указывается имя коллекции (Gas), в поле Variables to Store - имена переменных, которые мы включаем в коллекцию (в нашей коллекции три переменных: closk, N и Status).

Для коллекции Gas значения всех переменных будут регистрироваться каждые 10 единиц модельного времени, начиная с 0 (запуск модели) и кончая 1440 единицами модельного времени (остановка модели). Эти

данные размещены в полях: Trigger Type, Trigger at Time, Repeat Interval и Stop Time.

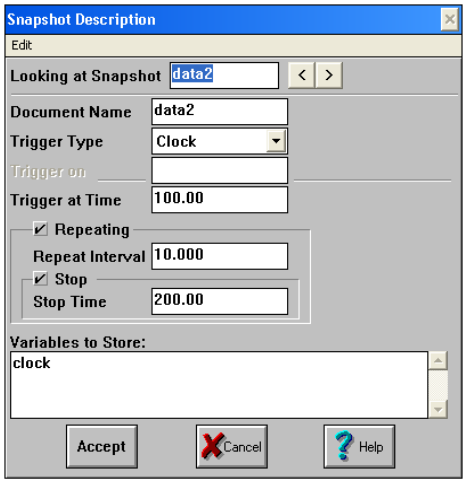


Рис. 10. Описание коллекции переменных

В общем случае условия сбора значений переменных коллекции могут быть и другими - эти возможности определяются списком Trigger Type (см. рис. 11 с раскрытым ниспадающим меню).

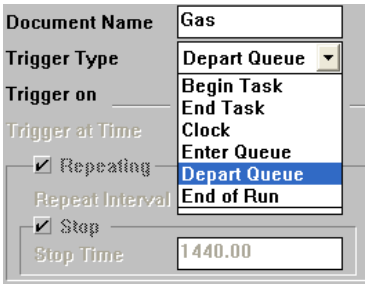


Рис.11. Меню условий сбора переменных в эксперименте

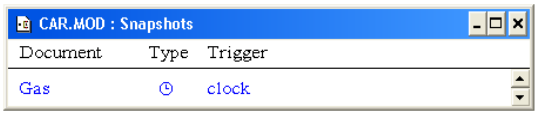
Варианты Begin Task и End Task определяют в качестве условий сбора значений переменных соответственно ситуации, когда тэг входит в блок задачи и выходит из блока задачи. При этом поле Trigger on будет определять номер этой задачи.

Варианты Enter Queue и Depart Queue определяют в качестве условий сбора ситуации, когда тэг входит в очередь и соответственно выходит из нее. При этом поле Trigger on будет определять номер очереди.

Вариант End of Run будет связан со сбором результатов только в момент окончания моделирования.

Выбор одного из этих вариантов будет определять условия сбора значений всех переменных коллекции.

После того как мы определили состав переменных коллекции, условия сбора результатов и закрыли окно рис. 10, в окне рис. 9 появится запись



Теперь при необходимости внесения изменений в коллекцию (добавление или удаление переменных) можно открыть окно рис. 10 двойным кликом на строке коллекции в списке Snapshots.

Определение параметров прогона модели

Перед запуском модели необходимо определить, в каком режиме будет проходить моделирование (выполняться прогон модели). Для ввода такой информации следует нажать кнопку определения параметров прогона  , при этом на экране монитора появляется окно рис. 12.

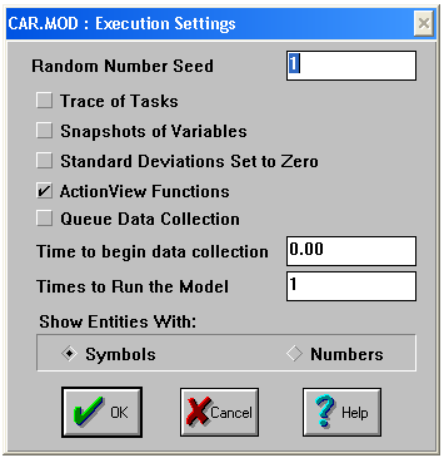


Рис. 12. Установка параметров прогона

В этом окне отмечаются функции, которые будут реализованы в процессе прогона модели.

1. Trace of Tasks (трассировка задач).
2. Snapshots of Variables (сбор значений переменных).
3. Standart Deviations Set to Zero (не используется для простых задач).
4. Action View Functions (анимация).
5. Queue Data Collection (сбор данных о состояниях очередей).

6. Time to begin Data Collection (модельное время начала сбора данных).

7. Times to Run the Model (номер прогона модели).

8. Show Entities With (форма показа динамических объектов - тэгов, символами или их количеством, обычно символами).

Второй и пятый из этих пунктов необходимы для сбора результатов, остальные реализуют вспомогательные функции.

После установки этих позиций при запуске модели система запросит вас о файлах, в которых вы предполагаете собирать информацию. Обычно имена таких файлов совпадают либо с именем модели (для нашего примера CAR), либо с именем коллекции переменных (Gas).

Структуры файлов результатов

На рис. 13 представлен фрагмент файла GAS.RES, в котором собрана коллекция Gas (рис. 10), а на рис. 14 - фрагмент файла CAR.QUE, в котором собрана информация о состоянии очередей в течение прогона программы модели.

Файл рис. 13 содержит 4 поля, из них 3 хранят значения переменных коллекции, собранных через интервал в 10 ед. времени, а поле Trigger информирует нас об условии сбора данных (см. рис. 10, 11). Файл в целом содержит 144 записи о значениях переменных коллекции Gas. (Термин Snapshop определяет как бы моментальный снимок, фотографию, на которой запечатлено состояние модели, зафиксированное в определенный момент времени.)

Файл рис. 14 содержит 6 полей:

1. Clock (момент времени, в который изменилось состояние очереди);

2. Tag (номер объекта-тэга, который был активен в этот момент времени). Напомним, что активный тэг - это тэг, который в текущий момент времени передвигается по структуре модели и меняет ее состояние;

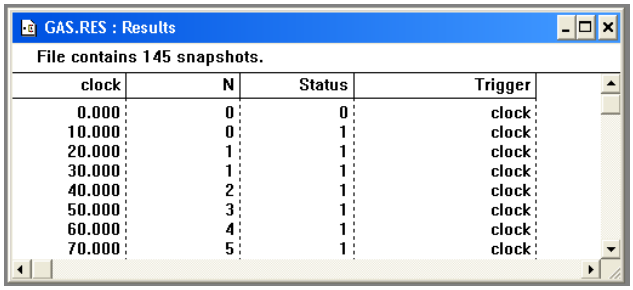
3. Run (номер прогона программы модели, он для нас не

информативен);

4. Length (длина очереди тэгов в момент Clock);

5. Wait (время ожидания в очереди активного тэга);

6. Trigger (характер изменения состояния очереди).



GAS.RES : Results

File contains 145 snapshots.

clock	N	Status	Trigger
0.000	0	0	clock
10.000	0	1	clock
20.000	1	1	clock
30.000	1	1	clock
40.000	2	1	clock
50.000	3	1	clock
60.000	4	1	clock
70.000	5	1	clock

Рис. 13. Фрагмент коллекции Gas

Поскольку в нашем примере всего одна очередь с номером 2 (см. рис. 6), все записи файла рис. 14 свидетельствуют об изменении состояния очереди с номером

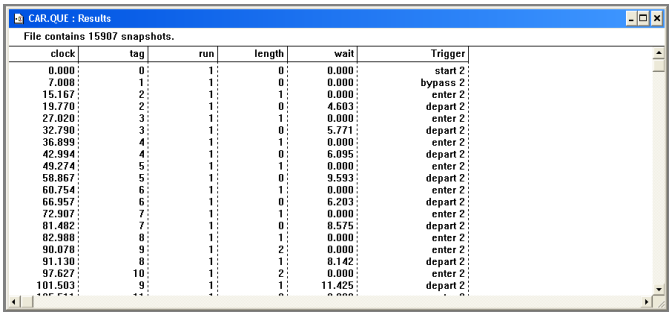
В поле Trigger встречаются записи четырех видов:

1) Start (старт для работы с очередью);

2) Enter (вход тэга в очередь);

3) Depart (выход тэга из очереди);

4) Bypass (проход тэга через очередь без задержки, т. е. через пустую очередь).



CAR.QUE : Results

File contains 15907 snapshots.

clock	tag	run	length	wait	Trigger
0.000	0	1	0	0.000	start 2
7.008	1	1	0	0.000	bypass 2
15.167	2	1	1	0.000	enter 2
19.770	2	1	0	4.603	depart 2
27.020	3	1	1	0.000	enter 2
32.790	3	1	0	5.771	depart 2
36.899	4	1	1	0.000	enter 2
42.994	4	1	0	6.095	depart 2
49.274	5	1	1	0.000	enter 2
58.867	5	1	0	9.593	depart 2
60.754	6	1	1	0.000	enter 2
66.957	6	1	0	6.203	depart 2
72.907	7	1	1	0.000	enter 2
81.482	7	1	0	8.575	depart 2
82.988	8	1	1	0.000	enter 2
90.078	9	1	2	0.000	enter 2
91.130	8	1	1	0.142	depart 2
97.627	10	1	2	0.000	enter 2
101.503	9	1	1	11.425	depart 2

Рис. 14. Фрагмент файла CAR.QUE

Первая запись файла CAR.QUE говорит о том, что в начальный момент времени тэг с номером 0 вошел в очередь нулевой длины и тут же вышел из нее (в блок 2), вторая свидетельствует о проходе тэга 1 через пустую очередь. Тэг с номером 2 вошел в пустую очередь и простоял в ней (в одиночестве) 3.476 ед. времени в связи с занятостью блока 2 (заправка

автомобиля - тэга с номером 1). Аналогично поведение тэгов с номерами 3 и 4. Всего за весь сеанс моделирования система зарегистрировала 259 изменений в очереди.

Анализ результатов эксперимента

Результаты эксперимента могут быть проанализированы с использованием методов, инкапсулированных в меню Анализ (Analyze). Эти методы позволяют представить результаты прогона модели в наглядной лаконичной форме графика или таблицы. Использование методов анализа призвано способствовать выявлению «узких мест» в организации исследуемой системы и принятию управленческих решений по реорганизации бизнес-процессов.

Статистика очереди

Файлы результатов с расширением QUE анализируются методами «Статистика очереди» (Queue Statistics) и «Графы очереди» (Queue Graphs). Для вызова этих методов необходимо открыть соответствующий файл (сделать окно с файлом активным) и выбрать в меню нужный вам метод.

На этом рисунке 15 приведены 4 вида графиков: зависимость длины очереди от времени, гистограмма длины очереди, зависимость времени ожидания в очереди от времени, гистограмма времени ожидания в очереди.

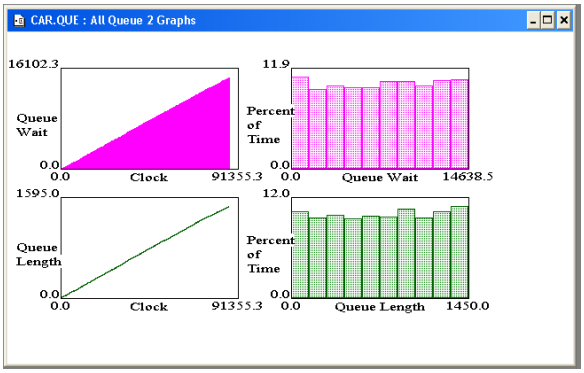


Рис. 15. Графики характеристик очереди

Зависимости

Методы анализа файла с расширением RES можно разделить на две группы: построение зависимостей и построение гистограмм.

Построение зависимостей выполняется в три этапа:

- разметка координатных осей будущего графика зависимости;
- оформление графика (название зависимости, названия осей, диапазоны и т. п.);
- выбор формы представления графика зависимости.

Разметка координатных осей графика связана с использованием правой части нижнего ряда панели инструментов (рис. 16). Для разметки необходимо «перетащить» кнопку горизонтальной оси с панели инструментов в тот столбец таблицы файла, который будет определять значения горизонтальной оси.

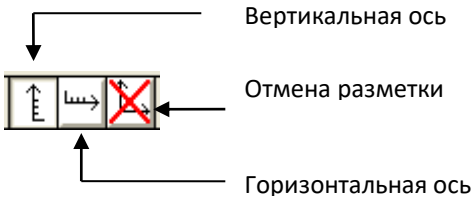


Рис. 16. Инструменты разметки

Аналогично перетаскивается кнопка вертикальной оси (оси ординат), но в отличие от горизонтальной она может быть перетащена в несколько столбцов таблицы файла. В этом случае в рамках одного графика будет построено несколько зависимостей. Кнопка удаления разметки «стирает» элементы сделанной разметки также путем ее «перетаскивания» в соответствующие столбцы файла.

Рис. 17 иллюстрирует разметку файла GAS.RES для построения зависимости переменной N от модельного времени clock, а рис. 18 - график этой зависимости, построенный с помощью метода «Линейчатый граф» (Bar Graph) меню Анализ.

GAS.RES : Results

File contains 145 snapshots.

clock	N	Status	Trigger
0.000	0	0	clock
10.000	0	1	clock
20.000	1	1	clock
30.000	1	1	clock

Рис. 17. Размеченный файл

Оформление графика реализуется методами «Шкала графика» и «Заголовки графика» (Graph Scale, Graph Titles) меню Анализ.

Гистограммы

Гистограмма (Frequency Distribution) - это особый вид графика, для разметки которого указывается только одна горизонтальная ось (ось абсцисс). Столбец файла, используемый для определения такой оси, сортируется:

• определяется максимальное значение, содержащееся в этом столбце (Max);

• определяется минимальное значение, содержащееся в этом столбце (Min);

• диапазон (Max - Min) делится на заданное число интервалов (Int) одинаковой ширины: w = (Max - Min)/Int.

После этого в процессе сортировки подсчитывается количество значений Li, I = 1 ÷ Int, попадающих в каждый из таких интервалов. Подсчитанные таким образом величины Li выводятся на вертикальную ось графика гистограммы. Гистограмма интерпретируется обычно как показатель эффективности использования того или иного оборудования в течение определенного интервала времени. Например, рис. 20 представляет гистограмму переменной Status, определяющей состояние бензоколонки в течение всего периода моделирования (1440 мин. = 1сутки). Эта переменная может принимать в модели только два значения: 1 (занята) и 0 (свободна). Гистограмма представлена двумя столбиками: высота первого определяется количеством нулей в столбце Status файла GAS.RES, а высота второго - количеством единиц в том же столбце. Анализ этого графика наглядно показывает, что бензоколонка в течение суток простаивает менее 4% времени, что свидетельствует о ее высокой загрузке.

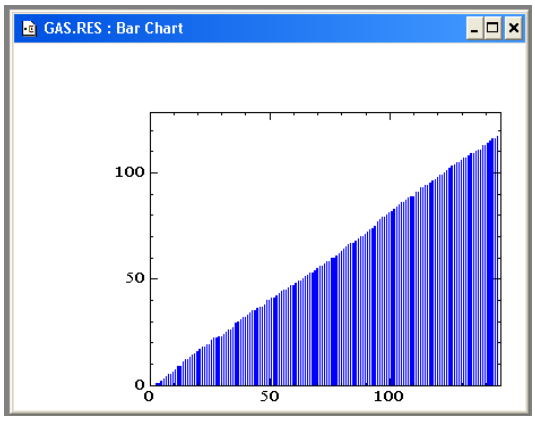


Рис. 18. Пример графика зависимости

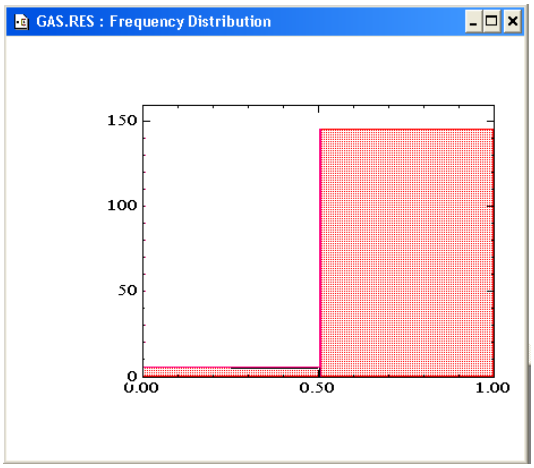


Рис. 19 Пример графика гистограммы

Оформление графика гистограммы выполняется теми же методами, что и для обычных зависимостей.

Развитие концепций: дополнительные методы и средства имитации

Модели потоков в сетевых структурах

В предыдущем разделе мы рассматривали модель потока автомобилей, приезжающих на автозаправочную станцию. При этом в качестве элемента такого потока рассматривался тэг - автомобиль (пример 1).

В общем случае элементом потока может являться любой объект, перемещающийся, перемещаемый или возникающий во времени или в пространстве. Объектом информационного потока может быть накладная (например, на получение товара), документ, распоряжение, сообщение и т.п. Финансовый поток формируется последовательностью трансфертов, денежных

поступлений, вложений и т. д.

Любое изменение состояния исследуемой системы (изменение значений переменных модели и/или перемещение тэга) рассматривается как событие. Поэтому любой информационный поток может рассматриваться как поток событий. В формальное понятие события можно вложить любое смысловое содержание, например, начало рабочей смены, возникновение аварии, приезд автомобиля на заправочную станцию и т. п. В качестве элементов событийных потоков могут выступать любые объекты реального (или виртуального) мира.

Виды потоков

Наиболее простой является модель однородного потока. Элементы такого потока однотипны, они либо не отличаются один от другого, либо такие отличия несущественны для решения проблемы. Например, в примере 1 мы рассматривали поток автомобилей как однородный, поскольку нас не интересовали ни марка автомобиля, ни объем его бензобака, - эти свойства мы считали несущественными для построения нашей простой модели. Поток описывался только тремя характеристиками:

1) средним временем между приходами автомобилей;

2) среднеквадратичным отклонением;

3) законом распределения вероятностей времени между поступлениями автомобилей на заправку.

Во многих задачах поток однородных событий характеризуется интенсивностью - величиной, обратной среднему времени между событиями в потоке.

Однородный - поток, в котором интервалы времени между событиями распределены по экспоненциальному закону, называется простейшим. Такой поток имеет единственную числовую характеристику - среднее время между событиями (поле Standard Deviation при задании такого потока не используется).

Модели неоднородных потоков характеризуются наличием индивидуальных особенностей у тэгов - элементов потока. Например, грузовик и автобус - две разновидности транспортных средств, которые могут являться элементами одного транспортного потока. Моделирование неоднородных потоков всегда связано с привнесением в модель индивидуальных особенностей тэгов. Техника такого моделирования поясняется в примере

Пример 2: модель обслуживания клиентов в парикмахерской

В парикмахерскую могут приходить клиенты двух типов. Клиенты первого типа желают только стричься. Распределение интервалов их прихода 35±10 мин. Клиенты второго типа желают постричься и побриться. Распределение интернатов их прихода 60±20 мин. Парикмахер обслуживает клиентов в порядке «первым пришел – первым обслужен». На стрижку уходит 18±6 мин., а на бритье 10±2 мин.

В парикмахерской оборудовано только одно место для обслуживания клиентов. Определите, насколько целесообразно оборудование второго места и прием на работу второго парикмахера.

Индивидуальные свойства тэгов

В этой задаче необходимо учитывать индивидуальные особенности клиентов, т. е. не только знать индивидуальный номер тэга, но и тип клиента, которого он представляет в модели. Для того чтобы реализовать такую возможность, необходимо «повесить на грудь» каждого тэга – клиента, входящего в модель, «визитную карточку», на которой должно быть написано, к какому типу он принадлежит:

• 1 – клиенту нужна только стрижка;

• 2 – клиенту нужна стрижка и бритье.

Массив таких «визитных карточек» должен быть описан в списке переменных. На рис. 20 приведено окно описания переменной client_type, имитирующей визитные карточки клиентов парикмахерской.

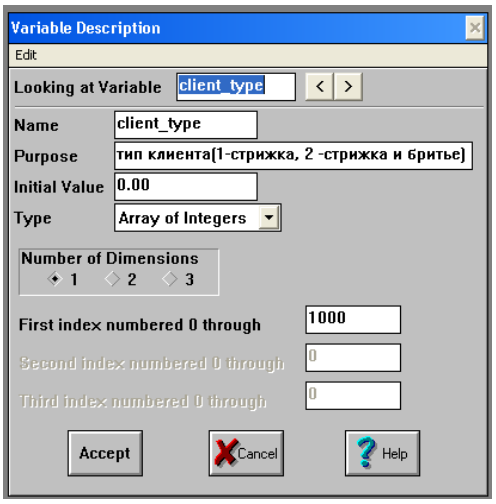


Рис. 20. Описание переменной client_type

Эта переменная характеризуется размерностью и типом. Размерность массива определяет резервируемое количество визитных карточек, – в нашем случае мы выбрали 1000. Каждая из карточек является элементом массива client_type и идентифицируется индексом, например client_type [1] – визитная карточка первого визитера, client_type [2] – второго и т. д. В общем случае, если tag – номер тэга, то client_type [tag] – визитная карточка этого тэга. Переменная или число в квадратных скобках определяют индекс элемента массива (обычно это целое число или целочисленная переменная). Тип переменной client_type определен как Array of Integers – массив целых чисел, это означает, что каждый из элементов массива может иметь в качестве значения только целые числа (в нашем примере 1 или 2).

Описание модели

На рис. 21 приведена структура модели примера 2. Она призвана проиллюстрировать только принцип моделирования неоднородных потоков и потому описана здесь не полностью.

В этой структуре первый блок генерирует единственный тэг, который запускает на вход парикмахерской два потока клиентов: первого типа (client I stream) и второго (client 2 stream). В этих блоках отрабатываются задержки между приходами клиентов и «навешиваются визитные карточки». Для этого в поле Beginning Effect блока 2 (client 1 stream) размещаются операторы: tag: = tag + l; client_type [tag]: = l; а в блоке 3 (client 2 stream) – операторы: tag: = tag + l;

client_type [tag]: = 2;. Перед блоком 4 образуется общая очередь клиентов разных типов, поскольку вход в этот блок ограничивается условием: St = = 0; где St – переменная, имитирующая состояние парикмахера (0 – свободен, 1 – занят). Это условие выносится в поле Release Condition блока 4.

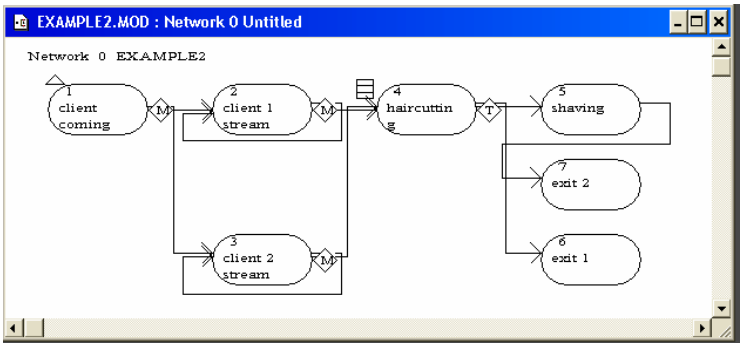


Рис 21. Структура модели примера 2

Задержка на время обслуживания в этом блоке определяется величиной 18±6 ед. времени. В поле Beginning Effect размещается оператор St: = l, который имитирует переход парикмахера в состояние занятости, а в поле Ending Effect размещается оператор освобождения парикмахера. В нашем примере он может быть освобожден, если завершилось обслуживание клиента первого типа, если же стригся клиент второго типа, то его обслуживание должно быть продолжено (в блоке 5 – shaving), и следовательно парикмахер не может быть освобожден. В соответствии с этим содержанием в поле Ending Effect блока 4 размещается оператор: if client_type [tag] = = 1 then St: = 0;.

Клиент, выходящий из четвертого блока, направляется в пятый при выполнении условия: client_type [tag] = = 2; и направляется в шестой при выполнении условия client_type [tag] = = 1. Эти условия размещаются в разветвителе, стоящем на выходе пятого блока.

Соответственно в поле Ending Effect блока 5 размещается оператор освобождения парикмахера: St: = 0;. В остальном эта модель мало чем отличается от модели примера 1.

Задача как накопитель тэгов

В общем случае в блоке задачи могут одновременно находиться несколько тэгов. Как правило, это происходит тогда, когда задача имитирует механизм обслуживания нескольких клиентов. В таких случаях задача превращается в своеобразный накопитель тэгов.

Пример 3: модель автозаправки с несколькими колонками

Эта модель отличается от модели примера 1 наличием нескольких бензоколонок.

В этом случае целесообразно, сохраняя структуру модели примера 1 (см. рис. 1), ввести переменную Nb – число свободных бензоколонок и переопределить блок 2 описанием, приведенным на рис. 22 (переменная Status при этом оказывается ненужной).

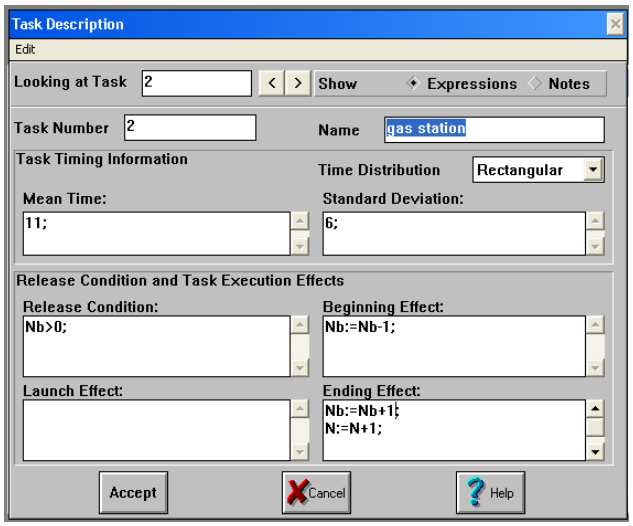


Рис. 22. Пример описания задачи-накопителя тэгов

В этом случае блок 2 примера 1 превращается в накопитель, в котором одновременно могут находиться Nb тэгов, имитирующих автомобили.

Замкнутые системы

Замкнутыми будем называть системы, в которых отсутствуют входные потоки тэгов. Все изменения в таких системах происходят за счет внутренних преобразований, внутренних процессов, происходящих в системе.

Деление систем на открытые и замкнутые достаточно условно. В

любой реальной системе обычно можно выделить компоненты того и другого вида, однако понятие замкнутой системы полезно для освоения приемов имитации сложных систем.

Пример 4: модель использования общего оборудования

Производство изделий определенного вида включает в себя длительный процесс индивидуального изготовления, заканчивающийся коротким периодом обжига изделия в печи. Поскольку содержание печи обходится довольно дорого, несколько рабочих, каждый из которых изготовляет «свое» изделие, используют одну печь, в которой одновременно можно обжигать только одно изделие. Рабочий не может начать новую работу, пока не вытащит из печи законченное изделие.

Таким образом, рабочий трудится в следующем режиме:

1) изготавливает изделие;

2) ожидает возможности использования печи по принципу «первым пришел – первым обслужен»;

3) использует печь;

4) переходит к изготовлению нового изделия.

На операцию изготовления изделия требуется 30±5 мин., на операцию обжига 8±2 мин. Требуется построить имитационную модель для определения такого количества рабочих, при котором с одной стороны очередь минимальна, с другой – простои печи минимальны.

Решение этой задачи связано с разработкой модели, в которой используется переменная Nw – количество используемых рабочих, значение которой подбирается в процессе моделирования таким образом, чтобы обеспечить наилучший баланс между длиной очереди и временем занятости печи. Структура модели иллюстрируется схемой рис. 23.

Блок 1 (Begin) создает тэги в количестве, определяемом величиной Nw. Каждый тэг имитирует рабочего, рее созданные тэги – рабочие становятся в очередь перед вторым блоком (Operation 1) и ждут «начала работы». Работа начинается с появлением в модели $(Nw)^{ого}$ тэга, который «разрешает» тэгам –

рабочим войти во 2ой блок. После входа каждый из рабочих начинает «ходить по кругу»: Блок2 – Очередь – Блок3. Подобное «хождение» имитирует рабочий цикл, выполняемый каждым рабочим в течение рабочего дня.

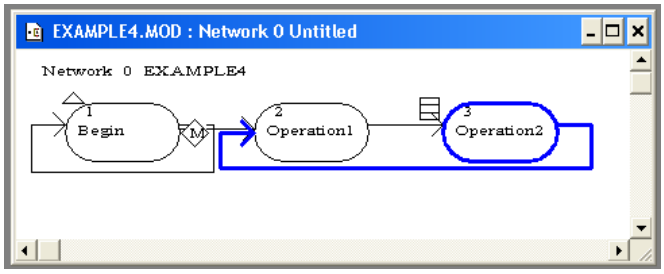


Рис. 23. Структура модели примера 4

Язык описания эффектов

Для описания модели используется язык описания эффектов. Основными элементами этого языка являются переменные, операторы и функции. Переменные были кратко описаны в разделе Описание элементов модели, там же был описан и оператор присваивания. Здесь мы излагаем дополнительные сведения о средствах языка описания эффектов, необходимые для разработки имитационных моделей.

Любое текстовое описание представляется последовательностью операторов языка, разделенных знаком «;». Кроме того, в любое место такого описания может быть вставлен комментарий – произвольный текст, заключенный в фигурные скобки, например {Это текст комментария}.

К операторам, которые используются наиболее часто, относятся операторы присваивания, уточняющие (adjustment) и логические операторы. Два последних описываются ниже. Более подробные сведения по составу операторов и особенностям их использования содержатся в справочном разделе системы.

Уточняющий оператор определяет лаконичную запись соответствующего оператора присваивания. Например, оператор присваивания X: = X + 1 эквивалентен уточняющему оператору X + = 1. Аналогично оператор X: = X – 6 эквивалентен оператору X – = 6, оператор X: = X * n оператору X * = n, а X: = X / Nn оператору X / = Nn. Здесь X, n, Nn – имена переменных.

Логические операторы сравнивают два числовых значения или логических аргумента. Результат равен 1, если сравнение является истинным, или 0, если сравнение ложно. Соответственно 1 рассматривается как значение ИСТИНА, а 0 – как значение ЛОЖЬ. Например, если известно, что a равно 0,01 a b равно 3, то следующие выражения истинны: a < = b; a < b; a <> b; a следующие ложны: a = = b; a > = b; a > b. (Здесь запись < = означает «меньше или равно», < означает «меньше», <> – «не равно», = = – «равно», > = – «больше или равно», > – «больше».)

Не путайте записи « = =» и «: =»: логический оператор «= =» сравнивает значения двух переменных, а оператор присваивания «: =» назначает значение переменной, стоящей слева от оператора.

К логическим операторам относятся также операторы «&» (логическое И) и «|» (логическое ИЛИ). Оператор «&» проверяет истинность двух логических выражений (одновременно) и возвращает значение 1, они оба истинны, в противном случае он возвращает значение 0. Например, если a равно 0,01 a b равно 3, то следующие выражения истинны:

(a = = 0,01) & (b = = 3); (a < 2) & (b > 2); (a < b) & (b <> 0).

Заметим, что выражение (a & b) также истинно, поскольку оба аргумента больше нуля, соответственно истинным будет и выражение (a * 100) & (b / 3). В этом смысле любое число, не равное нулю в логическом операторе, интерпретируется как ИСТИНА, а ноль – как ЛОЖЬ.

Логический оператор (|) проверяет, есть ли из двух значений хотя бы одно, не нулевое, и возвращает в этом случае значение ИСТИНА, а если нет, то возвращает значение ЛОЖЬ. В нашем примере логические операторы (a = = 0,01) | (b>4), (a>0) | (b >0) истинны, а (a = = 0) (a = = b), (a – 0,01) | (b – 3) – ложны.

If – then – else оператор обеспечивает выполнение действий, необходимых при заданных условиях.

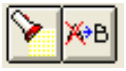
Например, оператор: if a + 3 = 5 then b: = I, c: = 1; в случае, когда (a + 3 = 5), т. е. переменная a имеет значение 2, запишет в переменные b и c значения

1, а в противном случае он ничего не сделает.

Оператор: if a then b + = 1 else b – = 1; в случае, когда а не равно нулю, увеличит значение переменной b на 1, а в противном случае (а равно нулю) уменьшит b на единицу.

Оператор: if a < b then b else a; при условии, что а меньше b, вернет значение переменной b, а в противном случае – значение переменной а.

Использование редактора

При заполнении полей для описания элементов модели целесообразно воспользоваться текстовым редактором, который доступен в любом окне описания элемента (кнопка Edit в левом верхнем углу окна). Редактор обеспечивает функции вырезания, копирования/вставки и удаления выделенных фрагментов текстового описания. Кроме того, при редактировании текстовых фрагментов целесообразно использовать специальные инструменты поиска и замены фрагментов текста, доступные через кнопки  верхней панели инструментов.

Датчики случайных чисел

Датчики случайных чисел реализуют механизмы имитации стохастических факторов. Значения таких факторов характеризуются распределениями вероятностей. Например, когда время между приходами автомобилей на заправочную станцию задается величиной 10±3 ед. времени, подразумевается, что такое время является случайным фактором, значения которого равномерно распределены в интервале [7, 13] ед. времени.

Равномерное распределение вероятностей (Rectangular Distribution, Uniform Distribution) продуцируется функцией random(), которая выдает действительные случайные числа в диапазоне 0.0 ÷ 1.0, и функцией randomInt (min, max), которая выдает целые случайные числа в диапазоне от min до max.

Кроме равномерного распределения вероятностей в прикладных задачах широко используются также экспоненциальное распределение и

распределение Пуассона.

Экспоненциальное распределение (Exponential Distribution) связано с моделированием простейших потоков. В таких потоках время между событиями распределено по экспоненциальному закону. Это распределение характеризуется единственным параметром – средним значением. Вызов функции expon (Mean) вернет в качестве результата значение случайного числа, выбранного из экспоненциального распределения со средним mean. Если в задаче задана интенсивность простейшего потока Int, то среднее время между событиями будет определяться как mean = l / Int. Поэтому для имитации задержек между появлениями событий следует воспользоваться вызовом функции expon (l / Int).

Распределение Пуассона (Poisson Distribution) тесно связано с экспоненциальным распределением: оно характеризует количество событий в простейшем потоке, наблюдаемое за определенный интервал времени. Если задать величину этого интервала (T) и интенсивность потока (Int), то произведение Mean~ = (Int * T) будет определять среднее количество событий за интервал времени T. Эта характеристика является единственным параметром функции poisson(Mean), которая используется как датчик пуассоновских случайных чисел. Использование функции poisson(Mean) возможно и для других задач, например для имитации количества записей в инвентаризационной ведомости, объемов производства деталей в течение рабочего дня и т. п. Функция poisson(Mean) всегда выдает случайные числа, которые являются положительными и целыми.

Для более подробного знакомства с использованием других распределений вероятностей следует обратиться к справочной информации системы или специальной литературе.

Функции

Функции языка разделяются на две категории: встроенные и определяемые пользователем Встроенные функции нам уже неоднократно встречались, примерами таких функций являются halt() (функция, реализующая

оператор остановки модели), Poisson (Mean) (датчик случайных чисел) и т. п. Эти функции не могут быть изменены пользователем – они не доступны для изменений.

Вторая категория функций создается пользователем для описания тех или иных эффектов моделирования. Создание такой функции связано с определением функции и включением ее в библиотеку функций. Для выполнения этих действий следует нажать кнопку  , которая откроет окно библиотеки функций.

В этой библиотеке хранятся только функции, созданные пользователем. Для внесения изменений в уже определенную функцию достаточно сделать двойной клик на соответствующей записи в окне библиотеки функций. Для определения новой функции следует при открытом окне библиотеки функций нажать кнопку  и заполнить поля открывающегося при этом окна определения функции (рис. 24).

В поле Name задается имя функции. Имена функций не должны совпадать с именами переменных, поэтому для именования функций целесообразно использовать заглавные буквы. В поле Purpose размещается краткое описание семантики функции, а в поле Expressions – алгоритмическое описание эффекта, связанного с вызовом функции. В дальнейшем вы получаете возможность использовать созданную функцию в любом текстовом фрагменте любого поля описания вашей модели. Для этого необходимо просто вставить имя функции в соответствующее место описания. В нашем примере на рис. 25 определена функция, имитирующая интервалы времени между прибытиями автомобилей на заправочную станцию. Поэтому для имитации потока автомобилей достаточно вставить в поле Mean Time соответствующего блока вызов функции INT.

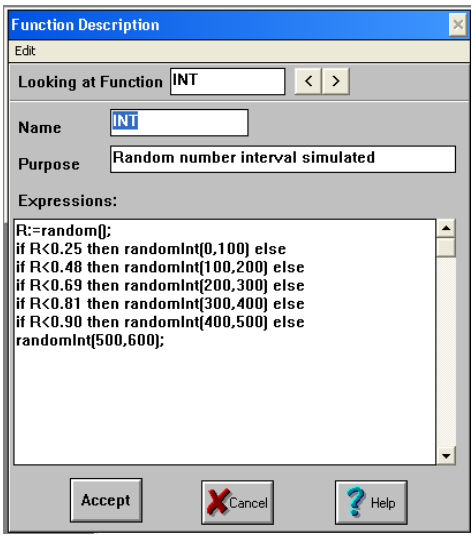


Рис. 24. Окно определения функции

Пример 5. Имитация случайных величин, заданных эмпирическим распределением (таблицей)

В качестве примера описания в окне рис. 24 мы использовали функцию датчика случайных чисел, определенных эмпирическим законом распределения вероятностей. Такой закон обычно задается таблицей. Здесь приведен пример такой таблицы и краткий комментарий к составлению алгоритма вычисления функции INT.

Интервалы времени между прибытиями автомобилей (сек)	Суммарная частота	Интервалы времени между прибытиями автомобилей (сек)	Суммарная частота
Меньше 0	0	400	0,81
100	0,25	500	0,9
200	0,48	600	1.0
300	0,69		

Из данных, приведенных в таблице, следует, что в 25% наблюдений интервалы времени между прибытиями автомобилей на автостанцию оказались меньше 100 сек, в 23% (0,48 – 0,25) – от 100 до 200 сек, в 21% (0,69 – 0,48) – от 200 до 300 сек и т. д. Датчик случайных чисел, имитирующий интервалы времени между приходами автомобилей, строится по простой схеме: сначала разыгрывается интервал (строка) таблицы, а затем число внутри этого интервала.

Динамика моделирования

Запуск и выполнение модели определяют динамический процесс моделирования. Этот процесс визуально отображается в двух формах: в виде процессов прохождения тэгов через структуру модели и в виде анимационных картин.

Первая форма отображения связана с закрашиванием тех блоков модели, в которых в текущий момент времени находятся тэги, и с изображением тэгов, скапливающихся в определенных «узких» местах (в очередях или перед блоками с ограниченным доступом). Очередь, в которой находятся тэги, закрашивается так же, как и занятые блоки, а сами тэги изображаются символами. Вместо индивидуального изображения тэгов на диаграмме исполняемой модели может указываться общее количество тэгов в блоке, перед блоком или в очереди. Установки отображения тэгов в динамике работы модели назначаются в окне рис. 12 (Show Entities With). Выбор «Symbols» определяет индивидуальное отображение тэгов символами, выбор «Numbers» – отображение общего количества тэгов.

Рисунок 25 иллюстрирует две формы отображения одного и того же состояния, наблюдаемого в динамике выполнения модели примера 1. Отображаемое состояние характеризуется тем, что в каждом из двух блоков модели находится по одному тэгу, а в очереди стоят 18 тэгов, ожидающих освобождения бензоколонки.

Форма анимационных картин используется главным образом для

презентации моделей. Средства анимации Micro Saint рассматриваются ниже.

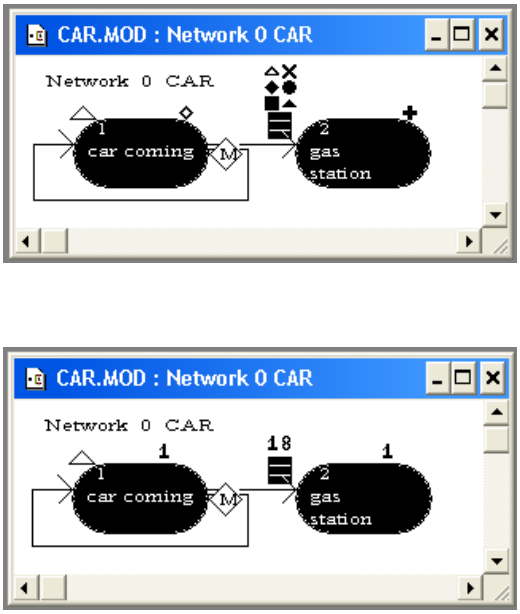


Рис.25. Две формы отображения динамики перемещения

Отладка моделей

При описании моделей в текстовых фрагментах могут быть допущены различные ошибки. Обнаружение и идентификация таких ошибок обычно проводятся в динамике выполнения модели, при этом пользователю выдается сообщение об ошибке. Получив такое сообщение, следует найти ошибку, исправить ее и продолжить выполнение модели или заново запустить модель. Ниже приводятся некоторые типичные примеры сообщений об ошибках.

• Message Syntax ERROR Job 1 tag: = tag + l

Semicolon expected at end. (в конце ожидается точка с запятой)

• Message s: = l;

Unrecognized word (неизвестное слово)

• Message

Matherror couldn't create the dialog box (Математическая ошибка не дает создать область диалога)

Любое сообщение об ошибке начинается со слова Message. Первый тип сообщения – наиболее полный. В нем содержится тип ошибки (синтаксическая), место локализации ошибки (блок 1), выражение, в котором

зафиксирована ошибка (tag:= tag + l) и собственно вид ошибки (в конце ожидается точка с запятой). При получении такого сообщения ошибку исправить легче всего для этого надо открыть окно описания блока 1, найти нужное выражение и вставить в конец его знак «;». (Попутно заметим, что этот знак должен «закрывать» любой оператор модели.)

Второй тип сообщения не содержит места локализации ошибки. Здесь просто приводится выражение (s:= l;) и констатируется, что система – не может распознать переменную s. Как правило, это связано с тем, что такую переменную забыли включить в список переменных. Устранение ошибки связано, таким образом, с включением переменной (или функции) в соответствующий список.

Третий тип ошибки обычно связан с использованием переменной типа Array of Integers или Array of Reals. Ошибка заключается в том, что размерность массива недостаточна и ее следует увеличить. Обычно для этого нужно в окне описания соответствующей переменной в строке «first (second, third) index numbered 0 through 10» увеличить последнее число: «through 1000». Такая ошибка обычно возникает при использовании в качестве индекса массива переменной tag (например, tag: = tag + l; f[tag]:= l, и f описана как Array of Integers или Array of Reals).

Все другие виды сообщений об ошибках по форме похожи на приведенные примеры.

Однако наряду с рассмотренными видами ошибок, которые могут быть обнаружены исполнительной системой, в модели могут присутствовать семантические ошибки, которые может распознать только человек – разработчик и пользователь модели. Например, по замыслу разработчика должно использоваться выражение tag:= tag + l;, а в модель введено выражение tag:= tag – 1;. Формально (синтаксически) оно правильно, но результаты, полученные на такой модели, будут неадекватны исследуемой системе. Такие ошибки обнаружить достаточно трудно. В этом смысле может оказаться полезным использование исполнительного монитора (execution monitor), окно

которого открывается кнопкой  .

В этом окне (рис. 26) отображаются значения переменных, которые изменяются в процессе моделирования и характеризуют с точки зрения пользователя адекватность имитационной модели исследуемой системы.

Любая модификация записи в окне исполнительного монитора реализуется двойным кликом на соответствующей строке, а добавление новой записи производится уже известной нам кнопкой. И в том и в другом случае открывается окно для ввода выражений, которые используются исполнительным монитором (рис. 27).

В поле expression to be evaluated вводятся выражения, значения которых могут характеризовать адекватность модели. Пользователь, наблюдая динамику изменений этих значений, может оценить, насколько правдоподобно поведение отлаживаемой модели.

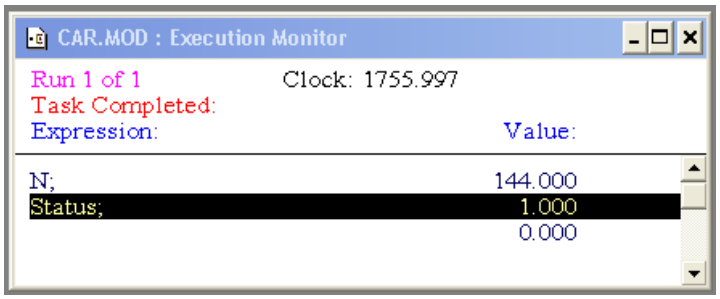


Рис. 26. Окно исполнительного монитора

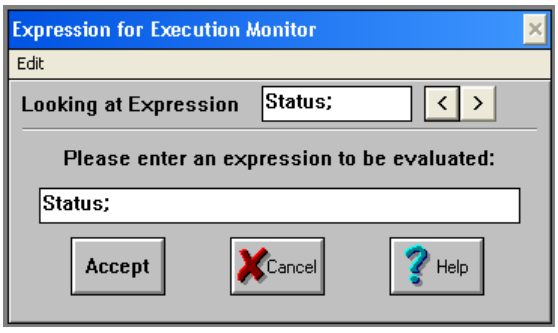


Рис.27. Окно ввода выражений для исполнительного монитора

Календарь событий

Модельное время – это системная переменная, имитирующая ход часов реального времени, в котором «живет» и развивается исследуемая система. Имитация хода реального времени основывается на концепции

событий, которые связаны с изменениями состояния модели. Такие события упорядочены по времени их возникновения в специальной структуре – календаре (расписании) событий. Пересчет модельного времени связан с выбором ближайшего по времени события из календаря и «переводом стрелок часов» модельного времени на момент возникновения этого события. Такая схема предполагает, что события могут следовать одно за другим через интервалы времени разной величины, включая и ноль (одновременно происходящие события).

Таким образом, календарь событий представляет собой своеобразный сценарий моделирования. Каждое событие такого сценария связано с выполнением определенного набора действий, которые должны произойти в тот или иной момент времени, по тому или иному условию. Сценарий содержит события двух типов:

1) события, связанные с изменением текущего состояния системы (например, выход активного тэга из очереди, вход тэга в тот или иной блок и т. п.);

2) запланированные события, специально введенные в календарь пользователем для управления процессом выполнения модели.

События первого типа полностью определяются структурой модели и описанием ее элементов, события второго типа управляют компьютерным экспериментом. Такое управление может быть связано с остановкой модели, установкой новых значений переменных в процессе моделирования, выполнением дополнительных вычислений по окончании этапа моделирования и т. п.

Планирование событий в календаре обычно используется для обработки промежуточных результатов компьютерного эксперимента и внесения необходимых изменений в динамике интерпретации модели.

В качестве примера использования календаря событий для управления экспериментом приведем планирование события остановки модели. Допустим, что нам необходимо остановить модель примера 1 через 1 сутки работы

заправочной станции:

1 сутки = 24 (час) * 60 (мин / час) = 1440 (мин).

Для планирования такого события кнопкой  открываем окно календаря событий (рис. 28), затем открываем окно описания события (для этого используется уже знакомая нам общая кнопка добавления объектов) и в этом окне (рис. 29) заполняем соответствующие поля.

В поле Perform at Time назначается время наступления события (1440 единиц модельного времени (EMB.), 1 EMB. = 1 мин.), в поле Expression определяется оператор, который должен быть выполнен в назначенное время (оператор остановки halt()). Нажатие кнопки Accept приводит к появлению в календаре событий соответствующей строки – уведомления о событии (см. рис. 28). Внесение в календарь такого уведомления приведет к остановке модели в момент времени clock = 1440, т. е. ровно через сутки функционирования исследуемой системы (бензозаправочной станции) в реальном времени.

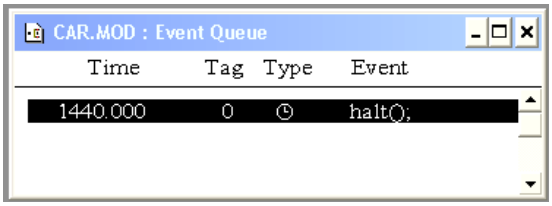


Рис.28. Пример записи в календаре событий

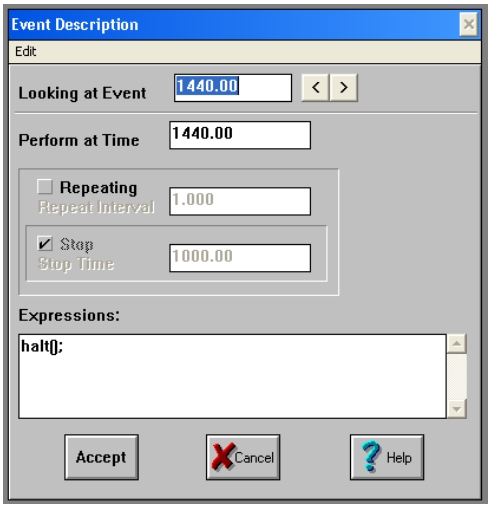


Рис.29. Окно описания события

Использование полей Repeating и Stop связано с перепланированием

событий через определенный интервал времени (Repeat Interval) до момента окончания процесса перепланирования (Stop Time). Перепланирование событий используется для внесения управляющих воздействий в динамике развития модели через определенные интервалы времени. Например, перепланирование события «Пауза» (pause ()) приведет к периодическим остановкам в процессе компьютерного моделирования.

Единица модельного времени и ее связь с реальным временем

Модельное (или системное) время – одно из основных понятий имитационного моделирования. Модельное время всегда связано с определенной системной переменной, которая должна копировать ход часов реального времени. Работа исследуемой системы на протяжении длительных периодов реального времени (сутки, месяцы, годы) воспроизводится в компьютерной имитационной модели за секунды или минуты с сохранением всех хронологических особенностей исследуемой системы (всей хронологии причинно – следственных связей, действующих в исследуемой системе). Такая хронологическая адекватность модели обусловлена тем, что единица модельного времени (EMB) определяет своеобразный масштаб, соотносящий размерности (реального времени и машинного таймера, определяющего скорость интерпретации модели.

Выбор единицы модельного времени (EMB) полностью определяется условиями задачи. Если все хронологические характеристики исследуемой системы заданы в одних и тех же временных единицах (например, секундах), то определение EMB тривиально: EMB = 1сек. Если же для описания отдельных компонент системы используются разные временные единицы (например, месяцы и дни), то выбор EMB определяется из субъективных соображений представления о точности и адекватности модели. Выбор EMB = 1 (день) приведет к более точной модели, но ее использование может потребовать больших затрат машинного времени, а выбор EMB = 1 (месяц) – к менее точной, но более быстродействующей модели.

Системная переменная clock, определяющая модельное время, имеет тип Real (действительное число). В этом смысле такой формат представления времени позволяет выбирать в качестве EMB любую единицу реального времени с минимальной потерей точности. Тем не менее, если в задаче используются сильно отличающиеся по протяженности временные отрезки, то целесообразно в качестве EMB выбирать наименьший из них, соответствующий минимальной единице реального времени.

Разработка многоуровневых структурных моделей

До сих пор мы рассматривали модели только одного (нулевого, верхнего) уровня. В общем случае сетевая структура модели может быть представлена на нескольких уровнях. При этом нулевой уровень определяет основную модель, составленную из компонент, каждая из которых в свою очередь может рассматриваться как модель первого, второго и т.д. уровня. Концепция многоуровневой модели позволяет уточнять структуру составляющих ее компонент на нижних уровнях, оставляя на верхних только общие «архитектурные контуры» исследуемой системы. Для конструирования многоуровневой модели используются дополнительные средства панели инструментов конструирования сетевой структуры (рис. 30). Эти средства включают в себя:

• создание подуровня сетевой структуры;

• средства навигации (перемещения) по иерархической структуре модели;

• дополнительные средства вставки / вырезки структурных объектов.

Создание подуровня **Вырезка/вставка структурных фрагментов**

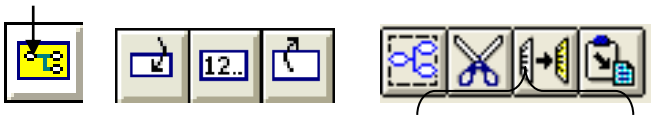


Рис.30. Средства конструирования многоуровневых моделей

Допустим, что на автозаправочной станции имеется магазин, и каждый водитель после заправки автомобиля может подъехать к этому магазину и сделать несколько покупок. Такая обобщенная модель в дополнение к введенным ранее блокам будет содержать еще модель обслуживания покупателей в магазине. Не определяя детали этой вновь вводимой компоненты, выделим для нее специальный подуровень модели. Для введения этого подуровня в структуру модели следует нажать кнопку и связать появившийся на экране монитора прямоугольник с ранее определенными блоками.

Затем для определения структуры модели обслуживания покупателей в магазине следует перейти в окно подуровня и определить эту модель обычными средствами. В результате мы получим двухуровневую структуру, приведенную на рис. 31

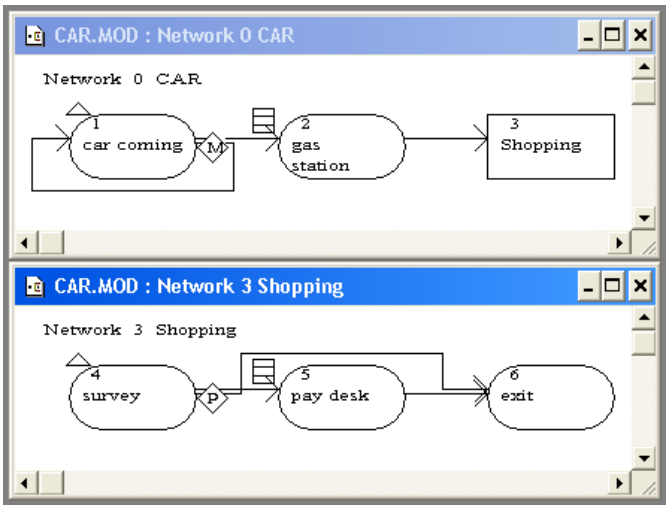


Рис.31.Пример многоуровневой модели

В этой модели окно CAR.MOD: Network 0 CAR определяет структуру модели примера 1 с добавленным фрагментом модели магазина, которую мы

назвали «Shopping». Прямоугольная рамка блока с номером 3 определяет, что модель, заключенная в этом блоке, обладает собственной сетевой структурой, которую мы определили в окне CAR.MOD: Network 3 Shopping как состоящую из трех блоков задач: осмотр магазина (survey), оплату покупок у кассы (pay desk) и выход (exit).

Любой блок–прямоугольник, используемый в процессе разработки модели, определяет собственную сетевую диаграмму. Таким образом, в рамках одной общей модели можно определить множество вложенных сетевых структур. Заметим, что нумерация блоков при этом остается сквозной, проходящей через все сетевые диаграммы, составляющие модель системы.

Отметим, что в тех случаях, когда сетевой блок–прямоугольник имеет выходящие стрелки, в его сетевой структуре в качестве завершающего элемента используется псевдоблок в виде окружности. Этот псевдоблок не имеет своего номера, он фактически является двойником блока, следующего за сетевым в диаграмме верхнего уровня. Сказанное иллюстрируется рис. 32.

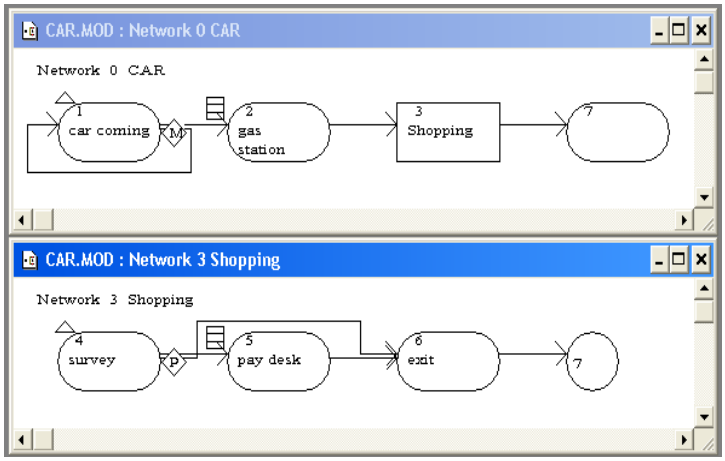


Рис.32. Иллюстрация к использованию псевдоблока

При моделировании сложных систем использование концепции многоуровневых моделей и «вложенных» сетей делает структуру модели значительно более наглядной и легко понимаемой.

Анимация имитационных моделей

Для презентации имитационных моделей и наглядного представления

процессов функционирования исследуемой системы используются специальные средства анимации модели. С помощью этих средств в отдельном окне Action View, предназначенном для просмотра анимационных картин, строится сцена, на которой в процессе имитации будут демонстрироваться процессы перемещения объектов (тэгов), определяющие динамику развития исследуемой системы.

Для того чтобы реализовать анимацию созданной модели, необходимо:

• определить фон, на котором будут развиваться анимационные процессы;

• связать динамику изменения сцены с программой модели.

Окно сцены (Action View) открывается нажатием кнопки (средний ряд панели инструментов).

Проиллюстрируем анимацию имитационной модели на примере1 «Модель обслуживания автомобилей на заправочной станции» (см. рис. 1).

Определение фона

Этот этап обычно связан с использованием готового рисунка или его созданием с использованием графического редактора. В качестве такого редактора проще всего использовать Paint, который включен в группу «Стандартные» общего списка программ Windows. Рисунок сохраняется в файле с расширением BMP.

Для вставки фонового рисунка в окно Action View необходимо:

• открыть вставляемый рисунок в графическом редакторе,

• скопировать его (в буферную область памяти),

• открыть окно Action View,

• вставить рисунок в окно Action View.

Последнее действие реализуется с использованием меню Micro Saint (раздел Edit – Paste) или кнопкой (верхний ряд панели инструментов).

Ниже на рис.33 приведен вариант фонового рисунка, созданного для

рассматриваемого примера в редакторе Paint.

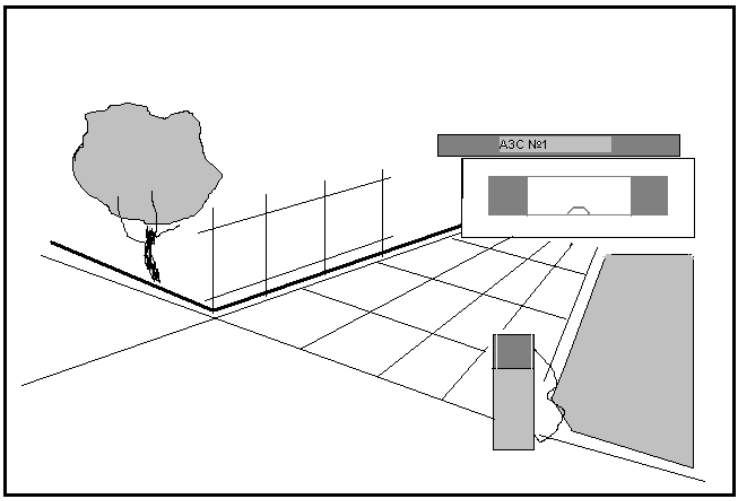


Рис.33. Пример фона для конструирования анимационной сцены

Динамика изменения сцены

Для реализации анимационной динамики необходимо выбрать изображения динамических объектов (иконки), которые в процессе моделирования будут перемещаться по сцене на подготовленном фоне. Иконки выбираются в меню Action View, раздел View Icons. Поскольку для рассматриваемого примера движущимся объектом является автомобиль, выберем иконку с изображением автомобиля.

Все процессы имитации движения автомобилей в программе модели будут иллюстрироваться передвижением соответствующих иконок на фоновом рисунке. Траектории такого передвижения определяются отрезками прямых с заданными начальной и конечной точками. Указание этих точек проводится непосредственно на фоновом рисунке в окне Action View с помощью специальной панели инструментов (нижний ряд справа).

Панель инструментов для размещения иконки объекта на сцене содержит три кнопки:

• левая кнопка с изображением трейлера (курсор – трейлер) для размещения иконки на сцене в точке начала траектории (в этой точке иконка появляется на сцене);

• средняя кнопка с изображением стрелки (курсор – стрелка) для

перемещения иконки по сцене в соответствии с планируемым отрезком траектории от точки начала отрезка до точки его конца;

• правая кнопка с изображением перечеркнутого трейлера (курсор – крест) для удаления иконки со сцены в точке окончания траектории

После выполнения первого действия окно Action View будет выглядеть, как показано на рис. 34.

В левом нижнем углу окна установлена точка появления иконки автомобиля на сцене – в это место фонового рисунка устанавливается курсор – трейлер и делается щелчок левой кнопкой мыши. При этом в строке под заголовком Action View (строке оператора) размещается оператор создания изображения объекта (автомобиля) на сцене create. Этот оператор фиксирует, что мы создали объект (tag), с координатами места на сцене, где разместилась иконка объекта (X = 121, Y = 158)

Для того чтобы оператор создания изображения объекта на сцене (create) выполнялся синхронно с оператором создания объекта в модели, его (оператор create) необходимо скопировать из окна Action View и вставить в соответствующее место программы. Копирование оператора в окне Action View связано с использованием команды Copy редактора Edit, а вставка в окно задачи – команды Paste. Для рассматриваемого примера такую вставку следует сделать в раздел Beginning Effect блока car coming (см рис 5) При этом каждый раз, когда в модели появляется новый тэг – автомобиль, приехавший на заправочную станцию, в окне Action View на сцене появляется его иконка, иллюстрирующая событие появления автомобиля.

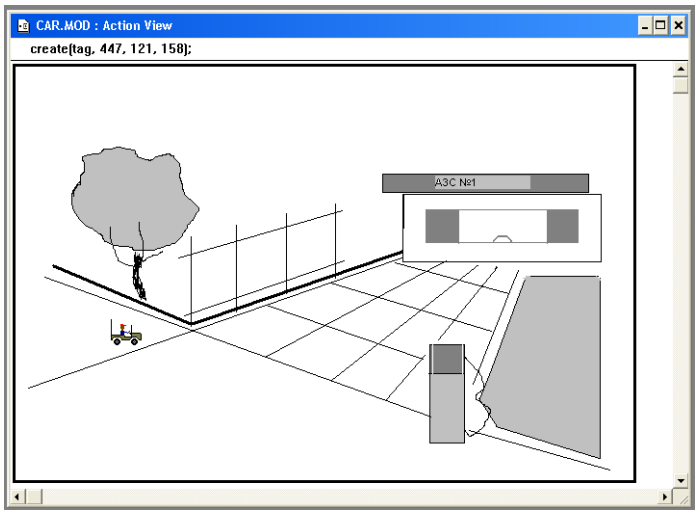


Рис.34. Конструирование анимационной сцены – 1

Анимация передвижения автомобиля от точки начала траектории до бензоколонки связана с «перетаскиванием» иконки объекта от точки его появления на сцене до изображения бензоколонки. Для этого необходимо курсором – стрелкой (средняя кнопка панели инструментов анимации) перетащить иконку в нужное место экрана.

При этом в левом верхнем углу окна Action View появится оператор перемещения объекта move, который определяет новые координаты положения иконки объекта на сцене и время, за которое произошло перемещение (duration) Этот оператор должен быть скопирован и перенесен в поле Launch Effect блока car coming (см рис 5)

Удаление иконки объекта со сцены выполняется с помощью правой кнопки панели инструментов анимации. Для этого курсор – крест, связанный с этой кнопкой, устанавливается на удаляемой иконке объекта и делается щелчок кнопкой мыши. Оператор удаления Dispose появляется в строке операторов окна Action View и переносится в соответствующее место программы модели аналогично. (Если мы ограничимся только анимацией приезда автомобилей на заправку, этот оператор должен быть размещен в поле Ending Effect блока car coming)

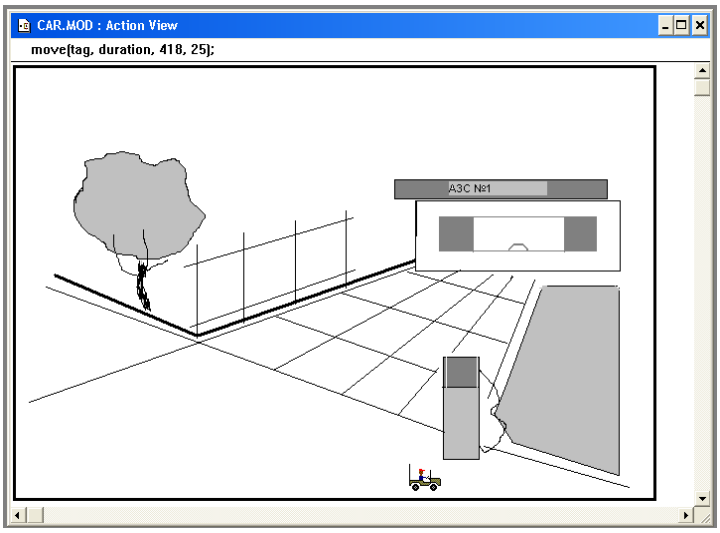


Рис.35. Конструирование анимационной сцены – 2

Таким образом, сделанные анимационные добавления в программу модели приводят к описанию блока car coming, приведенному на рис 36.

При запуске этой программы в окне Action View можно наблюдать динамический анимационный процесс приезда автомобилей на заправочную станцию.

Для того чтобы траектория движения автомобиля «оставляла след» на сцене, необходимо использовать разделы Dot, Line и Refresh меню Action View. Они связаны с видом представления траектории (точки, линии, стирание траекторий).

Основным вопросом анимации является правильная и корректная вставка операторов анимации в текст программы модели. Операторы create и dispose связаны с созданием и удалением иконки объекта (тэга), поэтому они должны синхронизироваться с созданием и удалением самого объекта. Оператор create обычно вставляется в поле Beginning Effect, a dispose – в поле Ending Effect. Оператор перемещения move связан с перемещением иконки тэга из точки начала отрезка траектории в точку конца за время duration. Duration – это то время, которое тэг проводит в блоке, поэтому оператор move всегда размещается в поле Launch Effect.

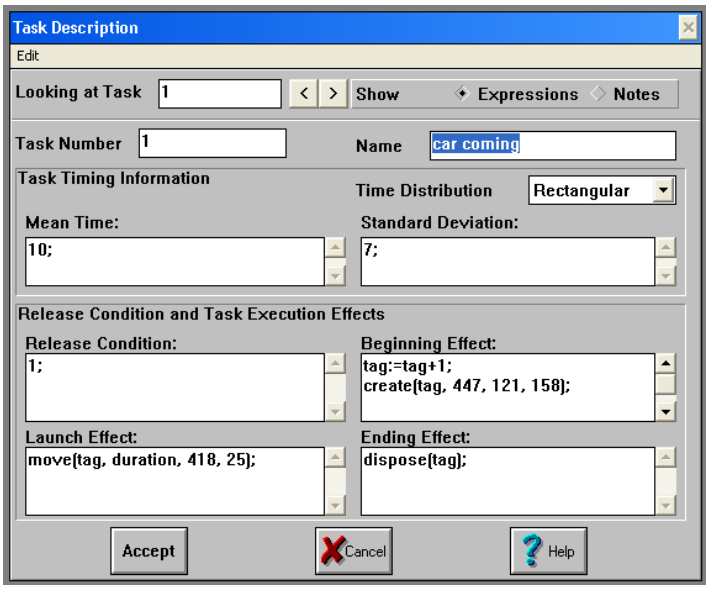


Рис.36. Окно описания задачи car coming с элементами анимации

Попутно заметим, что duration является системной переменной, связанной с активным тэгом, т.е. тэгом, который обрабатывается в модели в текущий момент времени.

В сложных случаях для получения наглядной анимационной картины может оказаться необходимым добавление в модель специальных блоков, которые служат только целям создания анимации. Например, для рассматриваемого примера мы ограничились анимацией только приезда автомобилей на заправочную станцию. Фаза ожидания освобождения бензоколонки и фаза собственно заправки не анимируются, поскольку автомобиль на этих фазах находится в неподвижном состоянии. Анимация фазы отъезда связана с уточнением дополнительных обстоятельств – как автомобиль уезжает? Просто его иконка исчезает с экрана монитора или движется по некоторой траектории до точки исчезновения? Для анимации этой фазы необходимо получить ответы на подобные вопросы и только после этого модифицировать модель. Если отъезд автомобиля будет связан с созданием траектории отъезда, в модель придется добавить еще один блок.

Существует и другая, не менее важная проблема, связанная с анимацией. В модели (и на сцене) могут одновременно присутствовать несколько динамических объектов. Если такие объекты движутся по одной и той же траектории, то в одном месте траектории могут находиться несколько

объектов одновременно. При этом иконка одного из объектов может заслонять другую, что лишает анимационную картину наглядности. Для преодоления этого эффекта можно использовать, например, метод смещения на сцене иконок разных объектов относительно друг друга путем введения случайной аддитивной составляющей в их координаты. Например, вместо оператора create (tag,447,121,158) использовать оператор create (tag,447,121 + randomInt (0,20),158 + randomInt (0,20)). Такой оператор каждый раз при появлении нового объекта в модели будет выводить на сцену иконку, координаты которой будут случайным образом выбираться из интервалов (121÷141) для X и (158÷178) для Y. На экране это выглядит как некоторая «туча» иконок, связанных с приезжающими и отъезжающими автомобилями.

Для более наглядной анимации динамики очереди используются более сложные механизмы, требующие специальных расчетов координат иконок на сцене. Однако все трудности составления программ анимации с лихвой окупаются наглядной иллюстрацией динамических взаимодействий в системе.

Розенцвайг А.К., Шарипов Р.Ш.

Имитационное моделирование экономических процессов

Учебно-методическое пособие

Подписано в печать 22.04.2019.
Формат 60х84/16. Печать ризографическая.
Бумага офсетная. Гарнитура «Times New Roman».
Усл.п.л. 3.25 Уч.-изд. л. 3.12
Тираж 100 экз. Заказ № 1249

Отпечатано в Издательско-полиграфическом центре
Набережночелнинского института
Казанского (Приволжского) федерального университета

423810, г. Набережные Челны, Новый город, пр.Мира, 68/19
тел./факс (8552) 39-65-99 e-mail: ic-nchi-kpfu@mail.ru