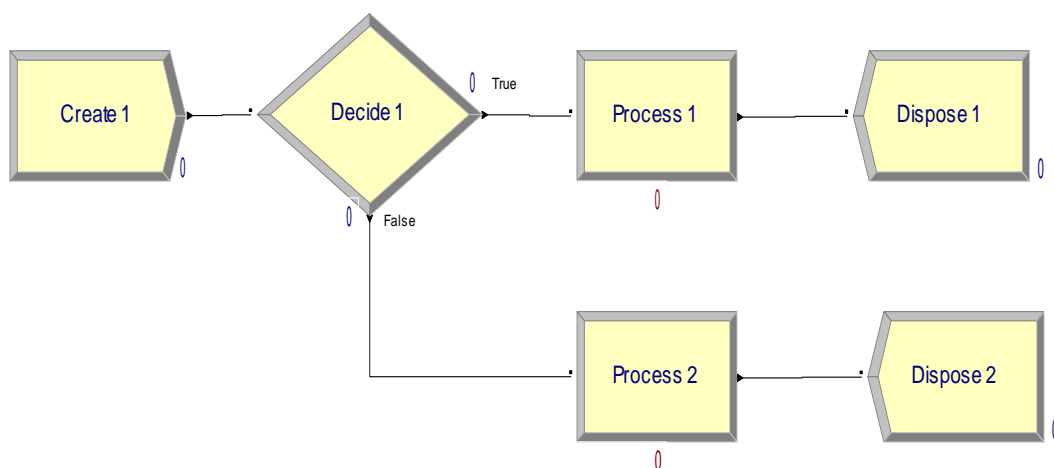


ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение
высшего профессионального образования
«Томский политехнический университет»

О. М. Замятина

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Учебное пособие



Издательство ТПУ
Томск 2007

УДК 681.3.06
ББК 32.973.2
К34

Замятина О. М.

К34 Компьютерное моделирование: Учебное пособие. – Томск: Изд-во ТПУ, 2007. – 121 с.

В учебном пособии кратко изложены основы теории моделирования систем, приведены различные типы классификации моделей, рассмотрены методологии структурного анализа и методы и средства имитационного моделирования систем.

Пособие подготовлено на кафедре автоматики и компьютерных систем Томского политехнического университета, соответствует программе дисциплины «Компьютерное моделирование» и предназначено для студентов Института дистанционного образования.

УДК 681.3.06

Рекомендовано к печати Редакционно-издательским советом
Томского политехнического университета

Рецензенты

М. П. Силич – профессор кафедры автоматизации обработки информации Томского университета систем управления и радиоэлектроники, доктор технических наук;

В. Г. Спицын – профессор кафедры вычислительной техники Томского политехнического университета, доктор технических наук.

Томский политехнический университет, 2007



Введение

Данное учебное пособие ориентировано на студентов технических и экономических специальностей, в специализацию которых входят следующие курсы: «Компьютерное моделирование», «Моделирование и анализ сложных систем», «Математическое моделирование систем», «Моделирование и анализ бизнес-процессов» и др.

Курс «Компьютерное моделирование» ориентирован на формирование у студентов навыков и знаний в теории моделирования систем и процессов различной природы с целью последующего их анализа и оптимизации.

Теоретическая часть курса дает сведения об основных понятиях моделирования, о возможных методах классификации моделей. Также рассматриваются методологии структурного анализа: IDEF0, IDEF3 и DFD и методы и средства имитационного моделирования: сети Петри, системы массового обслуживания.

Практическая часть курса позволяет студентам освоить и практически применять одно из самых современных пакетов имитационного моделирования Arena 7.0, в основу которого заложен математический аппарат раскрашенных сетей Петри и систем массового обслуживания.

Лабораторный практикум и выполнение курсовой работы позволит студентам развить системное мышление, находить различные варианты решения инженерных задач методом имитационного моделирования.

Учебный материал, ставший основой этого учебного пособия, уже в течение нескольких лет читается студентам Томского политехнического университета.

Автор выражает благодарность всем тем, кто принял участие в подготовке этого пособия, особенно Саночкиной Н. Г., совместная работа с которой принесла позитивные результаты, а также хочется выделить студентов, которые участвовали в этом: Карпову Евгению и Нгуен Минь Ки.

От автора

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты oxa@aics.ru. Я буду рада узнать Ваше мнение!

Глава 1. Основные понятия теории моделирования

1.1. Модель и моделирование

Слово «модель» (от лат. *modelium*) означает «мера», «способ», «сходство с какой-то вещью».

Термин «модель» широко используется в различных сферах человеческой деятельности и имеет множество смысловых значений. Мы под «моделью» будем понимать некий материальный или мысленно представляемый объект, который в процессе исследования замещает объект-оригинал так, что его непосредственное изучение дает новые знания об объекте-оригинале.

Модель – это объект или описание объекта, системы для замещения (при определенных условиях предложениях, гипотезах) одной системы (т. е. оригинала) другой системой для лучшего изучения оригинала или воспроизведения каких-либо его свойств [4, 23]. **Модель** – результат отображения одной структуры (изученной) на другую (малоизученную). Любая модель строится и исследуется при определенных допущениях, гипотезах. Модель должна строиться так, чтобы она наиболее полно воспроизводила те качества объекта, которые необходимо изучить в соответствии с поставленной целью [17, 27]. Во всех отношениях модель должна быть проще объекта и удобнее его для изучения. Таким образом, для одного и того же объекта могут существовать различные модели, классы моделей, соответствующие различным целям его изучения. Необходимым условием моделирования является подобие объекта и его модели. В этом случае мы должны говорить об **адекватности** модели объекту-оригиналу.

Если результаты моделирования подтверждаются и могут служить основой для прогнозирования процессов, протекающих в исследуемых объектах, то говорят, что модель адекватна объекту. При этом адекватность модели зависит от цели моделирования и принятых критериев.

Под адекватной моделью понимается модель, которая с определенной степенью приближения на уровне понимания моделируемой системы разработчиком модели отражает процесс ее функционирования во внешней среде. Под **адекватностью** (от лат. *adaequatus* – приравненный) будем понимать степень соответствия результатов, полученных по разработанной модели, данным эксперимента или тестовой задачи. Если система, для которой разрабатывается модель, существует, то сравнивают выходные данные модели и этой системы. В том случае, когда два набора данных

оказываются подобными, модель существующей системы считается адекватной. Чем больше общего между существующей системой и ее моделью, тем больше уверенность в правильности модели системы.

Проверка адекватности модели необходима для того, чтобы убедиться в справедливости совокупности гипотез, сформулированных на первом этапе разработки модели, и точности полученных результатов; соответствует точности, требуемой техническим заданием.

Для моделей, предназначенных для приблизительных расчетов, удовлетворительной считается точность 10-15 %, а для моделей, предназначенных для использования в управляющих и контролирующих системах – 1-2 % [27].

Любая модель обладает следующими свойствами:

- конечностью: модель отображает оригинал лишь в конечном числе его отношений;
- упрощенностью: модель отображает только существенные стороны объекта;
- приблизительностью: действительность отображается моделью грубо или приблизительно;
- адекватностью: модель успешно описывает моделируемую систему;
- информативностью: модель должна содержать достаточную информацию о системе в рамках гипотез, принятых при построении модели.

Процесс построения, изучения и применения моделей будем называть моделированием, т. е. можно сказать, что **моделирование** – это метод исследования объекта путем построения и исследования его модели, осуществляемое с определенной целью, и состоит в замене эксперимента с оригиналом экспериментом на модели.

Моделирование базируется на математической теории подобия, согласно которой абсолютное подобие может иметь место лишь при замене одного объекта другим, точно таким же. При моделировании большинства систем (за исключением, возможно, моделирования одних математических структур другими) абсолютное подобие невозможно, и основная цель моделирования – модель достаточно хорошо должна отображать функционирование моделируемой системы [1].

1.2 Классификация моделей

В общем случае все модели, независимо от областей и сфер их применения, бывают трех типов: познавательные, прагматические и инструментальные.

Познавательная модель – форма организации и представления знаний, средство соединения новых и старых знаний. Познавательная модель обычно подгоняется под реальность и является теоретической моделью.

Прагматическая модель – средство организации практических действий, рабочего представления целей системы для ее управления. Реальность в них подгоняется под некоторую прагматическую модель. Это, как правило, прикладные модели.

Инструментальная модель – средство построения, исследования и/или использования прагматических и/или познавательных моделей.

Познавательные отражают существующие, а прагматические – хоть и не существующие, но желаемые и, возможно, исполнимые отношения и связи.

Вся остальная классификация моделей выстраивается по отношению к объекту-оригиналу, методам изучения и т. п.

1.2.1. Классификация моделей по степени абстрагирования модели от оригинала

По степени абстрагирования от оригинала (рис. 1.1) модели могут быть разделены на материальные (физические) и идеальные. К **материальным** относятся такие способы, при которых исследование ведется на основе модели, воспроизводящей основные геометрические, физические, динамические и функциональные характеристики изучаемого объекта. Основными разновидностями физических моделей являются [17]:

- натурные;
- квазинатурные;
- масштабные;
- аналоговые.

Натурные – это реальные исследуемые системы, которые являются макетами и опытными образцами. Натурные модели имеют полную адекватность с системой-оригиналом, что обеспечивает высокую точность и достоверность результатов моделирования; другими словами, модель натурная, если она есть материальная копия объекта моделирования. Пример глобус – натурная географическая модель земного шара.

Квазинатурные (от лат. «квази» почти) – это совокупность натурных и математических моделей. Этот вид моделей используется в случаях, когда математическая модель части системы не является удовлетворительной или когда часть системы должна быть исследована

во взаимодействии с остальными частями, но их еще не существует либо их включение в модель затруднено или дорого.

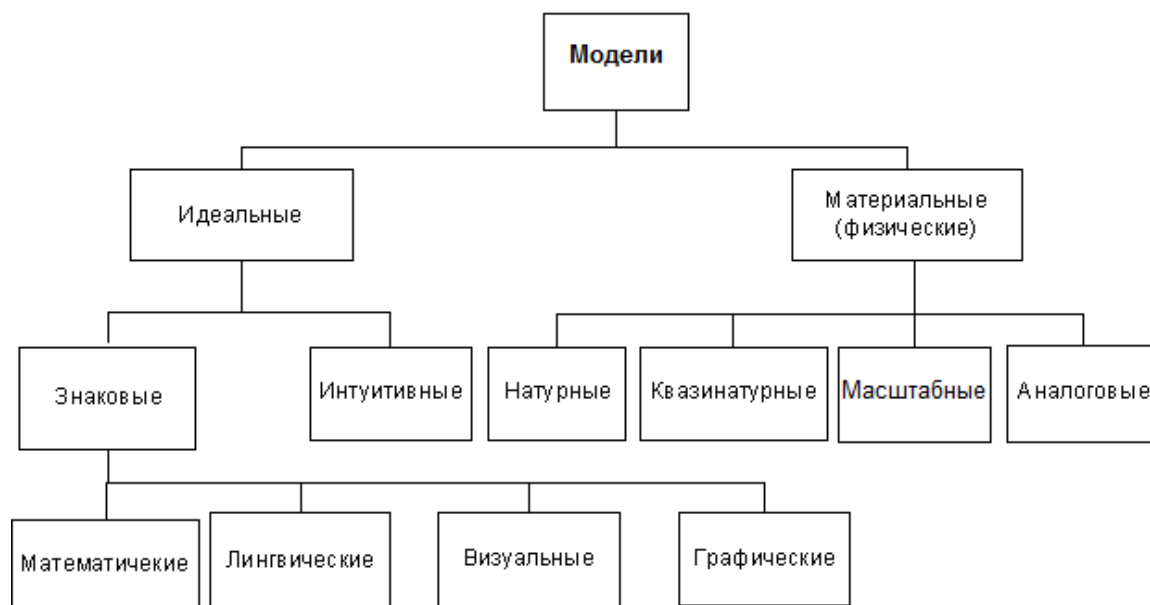


Рис. 1.1. Схема классификации моделей по степени абстрагирования от объекта-оригинала

Масштабные модели – это системы той же физической природы, что и оригинал, но отличающиеся от него размерами. В основе масштабных моделей лежит математический аппарат теории подобия, который предусматривает соблюдение геометрического подобия оригинала и модели и соответствующих масштабов для их параметров. Примером масштабного моделирования являются любые разработки макетов домов, а порой и целых районов при проведении проектных работ при строительстве. Также масштабное моделирование используется при проектировании крупных объектов в самолетостроении и кораблестроении.

Аналоговое моделирование основано на аналогии процессов и явлений, имеющих различную физическую природу, но одинаково описываемых формально (одними и теми же математическими уравнениями, логическими схемами и т. п.). В качестве аналоговых моделей используются механические, гидравлические, пневматические системы, но наиболее широкое применение получили электрические и электронные аналоговые модели, в которых сила тока или напряжение является аналогами физических величин другой природы. Например, является общеизвестным, что математическое уравнение колебания маятника имеет эквивалент при записи уравнения колебаний тока.

Идеальное моделирование носит теоретический характер. Различают два типа идеального моделирования: интуитивное и знаковое.

Под **интуитивным** будем понимать моделирование, основанное на интуитивном представлении об объекте исследования, не поддающемся формализации либо не нуждающемся в ней. В этом смысле, например, жизненный опыт каждого человека может считаться его интуитивной моделью окружающего мира.

Знаковым называется моделирование, использующее в качестве моделей знаковые преобразования различного вида: схемы, графики, чертежи, формулы, наборы символов и т. д., включающие совокупность законов, по которым можно оперировать с выбранными знаковыми элементами. Знаковая модель может делиться на лингвистическую, визуальную, графическую и математическую модели.

Модель **лингвистическая**, – если она представлена некоторым лингвистическим объектом, формализованной языковой системой или структурой. Иногда такие модели называют вербальными, например, правила дорожного движения – языковая, структурная модель движения транспорта и пешеходов на дорогах.

Модель **визуальная**, – если она позволяет визуализировать отношения и связи моделируемой системы, особенно в динамике. Например, на экране компьютера часто пользуются визуальной моделью объектов, клавиатуры в программе-тренажере по обучению работе на клавиатуре.

Модель **графическая**, – если она представима геометрическими образами и объектами, например, макет дома является натурной геометрической моделью строящегося дома.

Важнейшим видом знакового моделирования является **математическое** моделирование, классическим примером математического моделирования является описание и исследование основных законов механики И.Ньютона средствами математики.

Классификация математических моделей

Математические модели классифицируются:

– **по принадлежности к иерархическому уровню**: на модели микроуровня, макроуровня, метауровня (см. рис. 1.2).

Математические модели на **микроуровне** процесса отражают физические процессы, протекающие, например, при резании металлов. Они описывают процессы на уровне перехода (прохода).



Рис. 1.2. Схема классификации математических моделей по принадлежности к иерархическому уровню

Математические модели на **макроуровне** процесса описывают технологические процессы.

Математические модели на **метауровне** процесса описывают технологические системы (участки, цехи, предприятие в целом).

– **по характеру отображаемых свойств объекта** модели можно классифицировать на структурные и функциональные (рис. 1.3).



Рис.1.3. Схема классификации математических моделей по характеру отображаемых свойств объекта

Модель **структурная**, – если она представима структурой данных или структурами данных и отношениями между ними; например, структурной моделью может служить описание (табличное, графовое, функциональное или другое) трофической структуры экосистемы. В свою очередь, структурная модель может быть иерархической или сетевой.

Модель **иерархическая** (древовидная), – если представима некоторой иерархической структурой (деревом); например, для решения

задачи нахождения маршрута в дереве поиска можно построить древовидную модель, приведенную на рис. 1.4.

Модель **сетевая**, – если она представима некоторой сетевой структурой. Например, строительство нового дома включает операции, приведенные в нижеследующей таблице. Эти операции можно представить в виде сетевой модели, приведенной на рис. 1.5 и в табл. 1.1.

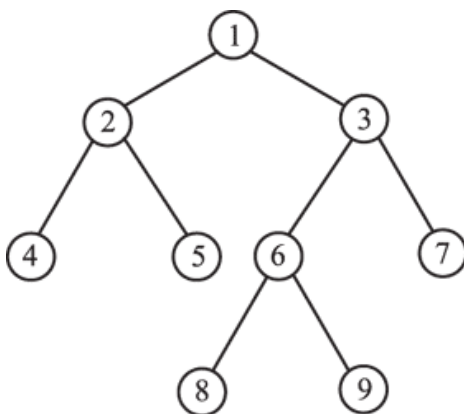


Рис. 1.4. Модель иерархической структуры

Таблица 1.1

Таблица работ при строительстве дома				
№	Операция	Время выполнения (дни)	Предшествующие операции	Дуги графа
1	Расчистка участка	1	нет	-
2	Закладка фундамента	4	Расчистка участка (1)	1-2
3	Возведение стен	4	Закладка фундамента (2)	2-3
4	Монтаж электропроводки	3	Возведение стен (3)	3-4
5	Штукатурные работы	4	Монтаж электропроводки (4)	4-5
6	Благоустройство территории	6	Возведение стен (3)	3-6
7	Отделочные работы	4	Штукатурные работы (5)	5-7
8	Настил крыши	5	Возведение стен (3)	3-8

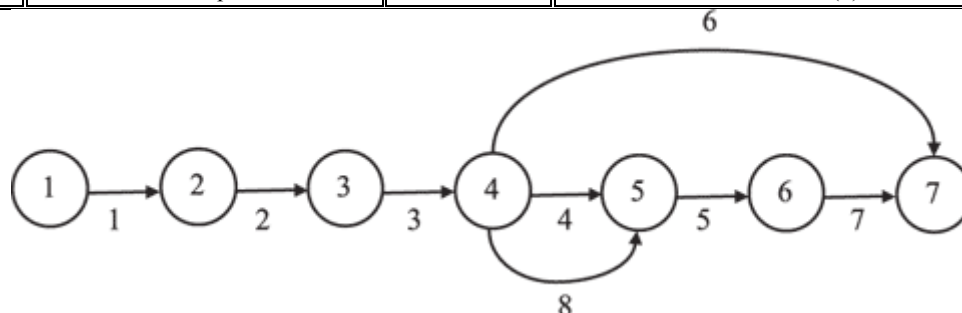


Рис. 1.5. Сетевой график строительства работ

Модель **функциональная**, – если она представима в виде системы функциональных соотношений. Например, закон Ньютона и модель производства товаров – функциональные.

– **по способу представления свойств объекта** (рис. 1.6) модели делятся на аналитические, численные, алгоритмические и имитационные [18].



Рис. 1.6. Схема классификации математических моделей по способу представления свойств объекта

Аналитические математические модели представляют собой явные математические выражения выходных параметров как функций от параметров входных и внутренних и имеют единственные решения при любых начальных условиях. Например, процесс резания (точения) с точки зрения действующих сил, представляет собой аналитическую модель. Также квадратное уравнение, имеющее одно или несколько решений, будет аналитической моделью.

Модель будет **численной**, если она имеет решения при конкретных начальных условиях (дифференциальные, интегральные уравнения).

Модель **алгоритмическая**, – если она описана некоторым алгоритмом или комплексом алгоритмов, определяющим ее функционирование и развитие. Введение данного типа моделей (действительно, кажется, что любая модель может быть представлена алгоритмом её исследования) вполне обосновано, т. к. не все модели могут быть исследованы или реализованы алгоритмически. Например, моделью вычисления суммы бесконечного убывающего ряда чисел может служить алгоритм вычисления конечной суммы ряда до некоторой заданной степени точности. Алгоритмической моделью корня квадратного из числа X может служить алгоритм вычисления его приближенного сколь угодно точного значения по известной рекуррентной формуле.

Модель **имитационная**, – если она предназначена для испытания или изучения возможных путей развития и поведения объекта путем варьирования некоторых или всех параметров модели, например модель экономической системы производства товаров двух видов. Такую модель можно использовать в качестве имитационной, с целью определения и варьирования общей стоимости в зависимости от тех или иных значений объемов производимых товаров.

– **по способу получения модели** делятся на теоретические и эмпирические (рис. 1.7).

Теоретические математические модели создаются в результате исследования объектов (процессов) на теоретическом уровне. Например, существуют выражения для сил резания, полученные на основе обобщения физических законов. Но они неприемлемы для практического использования, т. к. очень громоздки и не совсем адаптированы к реальным процессам обработки материалов.



Рис. 1.7. Схема классификации математических моделей по способу получения модели

Эмпирические математические модели создаются в результате проведения экспериментов (изучения внешних проявлений свойств объекта с помощью измерения его параметров на входе и выходе) и обработки их результатов методами математической статистики.

– **по форме представления свойств объекта** модели делятся на логические, теорико-множественные и графовые.

Модель **логическая**, если она представима предикатами, логическими функциями, например, совокупность двух логических функций может служить математической моделью одноразрядного сумматора.

Модель **теоретико-множественная**, – если она представима с помощью некоторых множеств и отношений принадлежности им и между ними.



Рис. 1.8. Схема классификации математических моделей по форме представления свойств объекта

Модель **графовая**, – если она представима графом или графами и отношениями между ними.

1.2.2. Классификация моделей по степени устойчивости

Все модели могут быть разделены на устойчивые и неустойчивые (рис. 1.9).

Устойчивой является такая система, которая, будучи выведена из своего исходного состояния, стремится к нему. Она может колебаться некоторое время около исходной точки, подобно обычному маятнику, приведенному в движение, но возмущения в ней со временем затухают и исчезают.

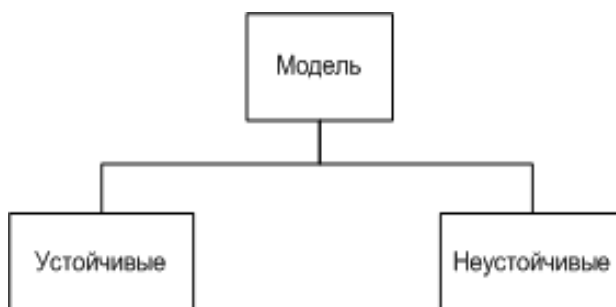


Рис. 1.9. Схема классификации математических моделей по устойчивости

В *неустойчивой* системе, находящейся первоначально в состоянии покоя, возникшее возмущение усиливается, вызывая увеличение значений соответствующих переменных или их колебания с возрастающей амплитудой.

1.2.3. Классификация моделей по отношению к внешним факторам

По отношению к внешним факторам модели могут быть разделены на открытые и замкнутые.

Замкнутой моделью является модель, которая функционирует вне связи с внешними (экзогенными) переменными. В замкнутой модели изменения значений переменных во времени определяются внутренним взаимодействием самих переменных. Замкнутая модель может выявить поведение системы без ввода внешней переменной. Пример: информационные системы с обратной связью являются замкнутыми системами. Это самонастраивающиеся системы, и их характеристики вытекают из внутренней структуры и взаимодействий, которые отражают ввод внешней информации.

Модель, связанная с внешними (экзогенными) переменными, называется *открытой*.

1.2.4. Классификация моделей по отношению ко времени

По отношению к временному фактору модели делятся на динамические и статические (см. рис. 1.10).

Модель называется *статической*, если среди параметров, участвующих в ее описании, нет временного параметра. Статическая модель в каждый момент времени дает лишь «фотографию» системы, ее срез. Одним из видов статических моделей являются структурные модели.

Динамической моделью называется модель, если среди ее параметров есть временной параметр, т. е. она отображает систему (процессы в системе) во времени.

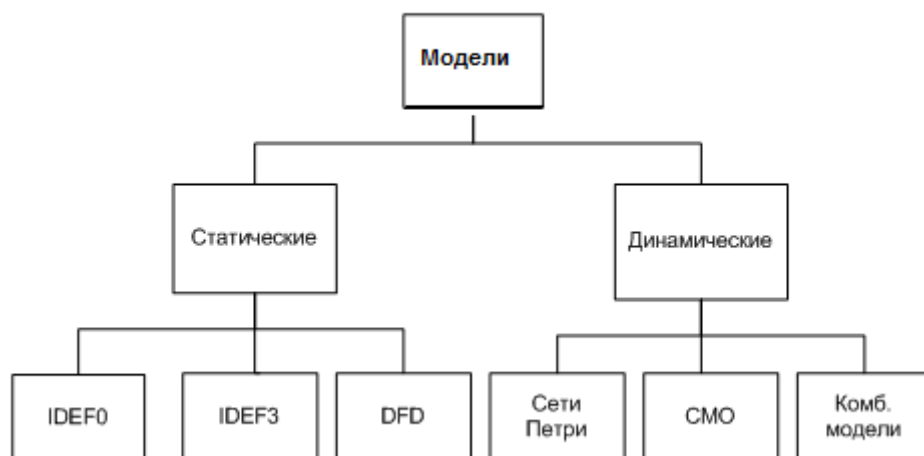


Рис. 1.10. Схема классификации математических моделей по отношению ко времени

Во второй и третьей главе этого учебного пособия будут рассматриваться методологии и средства компьютерного моделирования, позволяющие разрабатывать статические и динамические модели. Первая глава будет посвящена методологиям структурного анализа: IDEF0, IDEF3 и DFD.

Вторая глава пособия позволит изучить имитационное моделирование систем на примере современного программного пакета Arena 7.0. Эти имитационные модели, в свою очередь, являются дискретными, динамическими и стохастическими одновременно. Такой вид моделей чаще всего называют дискретно-событийным, он используется для построения моделей, отражающих развитие системы во времени, когда состояние переменных системы меняется в конкретные моменты времени. В такие моменты времени происходят события, которые изменяют состояния системы.

1.3. Этапы разработки моделей

В этой работе мы будем рассматривать процесс создания компьютерной модели. Процесс моделирования имеет итерационный характер и проводится в рамках ранее сформулированных целей и с соблюдением границ моделирования. Построение начинается с изучения (обследования) реальной системы, ее внутренней структуры и содержания взаимосвязей между ее элементами, а также внешних воздействий и завершается разработкой модели.

Моделирование – от постановки задачи до получения результатов – проходит следующие этапы:

I. Анализ требований и проектирование.

1. Постановка и анализ задачи и цели моделирования.
2. Сбор и анализ исходной информации об объекте моделирования.
3. Построение концептуальной модели.
4. Проверка достоверности концептуальной модели.

II. Разработка модели.

1. Выбор среды моделирования.
2. Составление логической модели.
3. Назначение свойств модулям модели.
4. Задание модельного времени.
5. Верификация модели.

III. Проведение эксперимента.

1. Запуск модели, прогон модели.
2. Варьирование параметров модели и сбор статистики.
3. Анализ результатов моделирования.

IV. Подведение итогов моделирования согласно поставленной цели и задачи моделирования.



Рис. 1.11. Схема создания модели

Необходимо отметить, что при разработке конкретных моделей с определенными целями и границами моделирования не обязательно все подэтапы должны выполняться. Например, при разработке статических моделей IDEF0, DFD 3 и 4 подэтапы «Разработки модели» не выполняются, т. к. эти методологии не предусматривают задание временных параметров модели.

На первом этапе моделирования – **«Анализ требований и проектирование»** – формулируется концептуальная модель, строится ее формальная схема и решается вопрос об эффективности и целесообразности моделирования системы.

Концептуальная модель (КМ) – это абстрактная модель, определяющая состав и структуру системы, свойства элементов и причинно-следственные связи, присущие анализируемой системе и существенные для достижения целей моделирования. В таких моделях обычно в словесной форме приводятся сведения о природе и параметрах (характеристиках) элементарных явлений исследуемой системы, о виде и степени взаимодействия между ними, о месте и значении каждого элементарного явления в общем процессе функционирования системы. При создании КМ практически параллельно формируется область исходных данных (информационное пространство системы) – этап подготовки исходных данных. На данном этапе выявляются количественные характеристики (параметры) функционирования системы и ее элементов, численные значения которых составят исходные данные для моделирования. Очевидно, что значительная часть параметров системы – это случайные величины. Поэтому особое значение при формировании исходных данных имеют выбор законов распределения случайных величин, аппроксимация функций и т. д. В результате выявления свойств модели и построения концептуальной модели необходимо проверить адекватность модели.

На втором этапе моделирования – **«Разработка модели»** – происходит уточнение или выбор программного пакета моделирования. Выбор средств моделирования: программные и технические средства выбираются с учетом ряда критериев. Непременное условие при этом – достаточность и полнота средств для реализации концептуальной модели. Среди других критериев можно назвать доступность, простоту и легкость освоения, скорость и корректность создания программной модели.

После выбора среды проектирования концептуальная модель, сформулированная на предыдущем этапе, воплощается в компьютерную модель, т. е. решается проблема алгоритмизации и детализации модели.

Модель системы представляется в виде совокупности частей (элементов, подсистем). В эту совокупность включаются все части, которые обеспечивают сохранение целостности системы, с одной стороны, а с другой – достижение поставленных целей моделирования (получения необходимой точности и достоверности результатов при проведении компьютерных экспериментов над моделью). В дальнейшем производится окончательная детализация, локализация (выделение системы из окружающей среды), структуризация (указание и общее описание связей между выделенными элементами системы), укрупненное описание динамики функционирования системы и ее возможных состояний.

Для того чтобы выполнить подэтап «Задание модельного времени» введем понятие модельного времени. В компьютерной модели переменная, обеспечивающая текущее значение модельного времени, называется **часами модельного времени**.

Существует два основных подхода к продвижению модельного времени: **продвижение времени от события к событию** и **продвижение времени с постоянным шагом** [14].

Подход, использующий продвижение времени в модели от события к событию, применяется всеми основными компьютерными программами и большинством разработчиков, создающих свои модели на универсальных языках (рис. 1.13) [14].

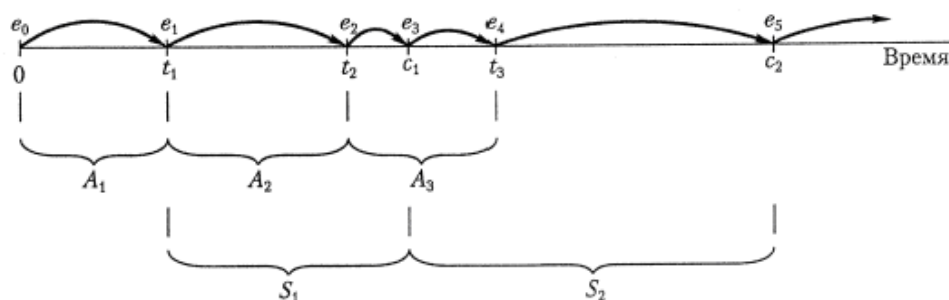


Рис. 1.13. Механизм продвижения модельного времени от события к событию

При использовании продвижения времени от события к событию часы модельного времени в исходном состоянии устанавливаются в 0, и определяется время возникновения будущих событий. После этого часы модельного времени переходят на время возникновения ближайшего события, и в этот момент обновляются состояние системы, с учетом произошедшего события, а также сведения о времени возникновения будущих событий. Затем часы модельного времени продвигаются ко времени возникновения следующего нового ближайшего события,

обновляется состояние системы и определяется время будущих событий и т. д. Процесс продвижения модельного времени от времени возникновения одного события ко времени возникновения другого продолжается до тех пор, пока не будет выполнено какое-либо условие останова, указанное заранее. Поскольку в дискретно-событийной имитационной модели все изменения происходят только во время возникновения событий, периоды бездействия системы просто пропускаются, и часы переводятся со времени возникновения одного события на время возникновения другого. При продвижении времени с постоянным шагом такие периоды бездействия не пропускаются, что приводит к большим затратам компьютерного времени. Следует отметить, что длительность интервала продвижения модельного времени от одного события к другому может быть различной [14].

При продвижении времени с постоянным шагом Δt часы модельного времени продвигаются точно на Δt единиц времени для какого-либо соответствующего выбора значения Δt . После каждого обновления часов выполняется проверка, чтобы определить, произошли какие-либо события в течение предыдущего интервала времени Δt или нет. Если на этот интервал запланированы одно или несколько событий, считается, что данные события происходят в конце интервала, после чего состояние системы и статистические счетчики соответствующим образом обновляются. Продвижение времени посредством постоянного шага показано на рис. 1.14, где изогнутые стрелки показывают продвижение часов модельного времени, а e_i ($i = 1, 2, \dots$) – это действительное время возникновения события i любого типа, а не значение часов модельного времени. На интервале $[0, \Delta t)$ событие происходит в момент времени e_1 , но оно рассматривается как произошедшее в момент времени Δt . На интервале $[\Delta t, 2\Delta t)$ события не происходят, но все же модель выполняет проверку, чтобы убедиться в этом. На интервале $[2\Delta t, 3\Delta t)$ события происходят в моменты времени e_2 и e_3 , однако считается, что они произошли в момент времени $3\Delta t$ и т. д. В ситуациях, когда принято считать, что два или несколько событий происходят в одно и то же время, необходимо применение ряда правил, позволяющих определять, в каком порядке обрабатывать события. Таким образом, продвижение времени посредством постоянного шага имеет два недостатка: возникновение ошибок, связанных с обработкой событий в конце интервала, в течение которого они происходят, а также необходимость решать, какое событие обрабатывать первым, если события, в действительности происходящие в разное время, рассматриваются как одновременные. Подобного рода проблемы можно частично решить, сделав интервалы Δt менее продолжительными, но тогда возрастает число проверок возникновения событий, что приводит к увеличению времени выполнения задачи. Принимая во внимание это

обстоятельство, продвижение времени с помощью постоянного шага не используют в дискретно-событийных имитационных моделях, когда интервалы времени между последовательными событиями могут значительно отличаться по своей продолжительности [14].

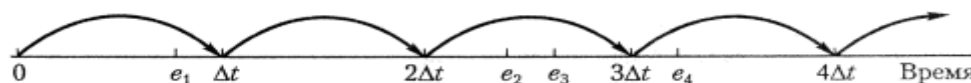


Рис. 1.14. Пример продвижения модельного времени посредством постоянного шага

В основном этот подход предназначен для систем, в которых можно допустить, что все события в действительности происходят в один из моментов n времени Δt ($n = 0, 1, 2, \dots$) для соответственно выбранного Δt . Так, в экономических системах данные часто предоставляются за годовичные промежутки времени, поэтому естественно в имитационной модели установить продвижение времени с шагом, равным одному году. Следует заметить, что продвижение времени посредством постоянного шага может быть выполнено с помощью механизма продвижения времени от события к событию, если планировать время возникновения событий через Δt единиц времени, т. е. данный подход является разновидностью механизма продвижения времени от события к событию.

Третий этап – **«Проведение эксперимента»** – является решающим, на котором, благодаря процессу имитации моделируемой системы, происходит сбор необходимой информации, ее статической обработки в интерпретации результатов моделирования, в результате чего принимается решение: либо исследование будет продолжено, либо закончено. Если известен результат, то можно сравнить его с полученным результатом моделирования. Полученные выводы часто способствуют проведению дополнительной серии экспериментов, а иногда и к изменению модели. Основой для выработки решения служат результаты тестирования и экспериментов. Если результаты не соответствуют целям моделирования (реальному объекту или процессу), значит, допущены ошибки на предыдущих этапах или входные данные не являются лучшими параметрами в изучаемой области, поэтому разработчик возвращается к одному из предыдущих этапов.

Подэтап «Анализ результатов моделирования» представляет собой всесторонний анализ полученных результатов с целью получения рекомендаций по проектированию системы или ее модификации.

На этапе «Подведение итогов моделирования согласно поставленной цели и задачи моделирования» проводят оценку проделанной работы, сопоставляют поставленные цели с полученными результатами и создают окончательный отчет по выполненной работе.

В курсе «Компьютерное моделирование» для моделирования процессов и систем используется пакет имитационного моделирования Arena 7.0. Этот программный пакет является современным средством моделирования высокого уровня, позволяющим создавать имитационные модели со сложной логикой.

Это программное средство в настоящее время только начинает использоваться в России, но его успешная апробация прошла за рубежом на ряде крупных предприятий, в таких областях, как машиностроительная отрасль, фармацевтика, авиа- и кораблестроение, промышленные производства, оборонная промышленность и т. п.

Arena имеет дружественный пользователю интерфейс, широкую панель моделирования и отчетов по результатам моделирования, специальные встроенные средства оптимизации, анализа входных и выходных данных.

Более подробно этот программный пакет будет рассмотрен в третьей главе.

1.4. Вопросы и задания к главе 1

1. Что такое модель и как Вы понимаете процесс моделирования?
2. Для чего и почему проводят моделирование реальных систем?
3. Приведите примеры различных классификаций моделей и назовите параметры этой классификации.
4. Расскажите о классификации математических моделей.
5. Перечислите и опишите основные этапы процесса моделирования.
6. Что такое «модельное время»? Какие механизмы изменения модельного времени существуют?

Глава 2. Методологии и средства структурного моделирования процессов и систем

2.1. SADT-методология

В настоящее время много написано и сказано о методологии SADT (Structured Analysis and Design Technique – методология структурного анализа и проектирования), но, несмотря на это, до сих пор существуют различные ее трактовки. Мы будем придерживаться следующей.

Методология SADT – это совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта предметной области. SADT-методология является основой семейства методологий моделирования IDEF. Семейство IDEF появилось в США в рамках правительственной программы ICAM (Integrated Computer Aid of Manufactory – интегрированная компьютерная помощь производству).

В настоящее время семейство IDEF (ICAM Definition – определение основных терминов) представляет собой IDEF0, IDEF1, IDEF2,..., IDEF16. В рамках этого учебного пособия и лекционных курсов, проводимых автором, будут рассмотрены две наиболее распространенные методологии моделирования:

1. Методология функционального моделирования IDEF0.
2. Методология событийного моделирования IDEF3.

2.1.1. Методология функционального моделирования IDEF0

За счет своей универсальности, строгости и простоты в настоящее время IDEF0-модели получили широкое распространение и используются:

1. При создании систем менеджмента качества (СМК) на предприятии. Процесс разработки СМК включает в себя разработку документированных процедур, которые представляют собой статическое описание процессов в виде IDEF0-моделей.

2. При проведении обследования деятельности предприятия. Обследование является важнейшим и определяющим этапом консалтинговых проектов, при которых осуществляется построение и анализ моделей деятельности предприятия двух типов: «как есть» и «как должно быть», отображающих текущее и целевое состояние предприятия.

3. При реинжиниринге, включающем изменение технологий целевой и текущей деятельности предприятия, операций учета, планирования, управления и контроля; построение рациональных технологий работы предприятия с учетом существующих автоматизированных систем; создание перспективной оргштатной структуры предприятия, осуществляющей реализацию рациональных технологий работы; изменение информационных потоков и документооборота, обеспечивающих реализацию рациональных технологий работы; разработку проектов схем внутреннего и внешнего документооборота, проекта положения о документообороте, проекта альбома форм входных и выходных документов.

4. При выборе критериев для внедрения корпоративных информационных систем (КИС).

5. При разработке и внедрении новых информационных систем (ИС).

6. При выборе программного обеспечения, автоматизирующего полностью или частично деятельность предприятия (например, системы электронного документооборота).

7. При стратегическом и оперативном планировании деятельности предприятия.

В основе IDEF0-методологии заложена следующая концепция [18]:

1. *Блочное моделирование и его графическое представление.* Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описывается посредством интерфейсных дуг, выражающих ограничения, которые, в свою очередь, определяют, когда и каким образом функции выполняются и управляются.

2. *Лаконичность и точность.* Выполнение правил SADT требует лаконичности и точности разрабатываемой документации и именования структурных элементов (блоков и стрелок), не накладывая в то же время чрезмерных ограничений на действия аналитика.

3. *Передача информации.* SADT-модель обычно является одной из первых стадий разработки проекта, затем модель передается для дальнейшей работы. Таким образом, модель должна быть разработана так, чтобы в дальнейшем с ней могли работать и понимать, что в нее заложено.

4. *Строгость и формализм.* Разработка моделей требует соблюдения строгих формальных правил, обеспечивающих

преимущества методологии в отношении однозначности и целостности сложных многоуровневых моделей.

5. *Итеративное моделирование*. Разработка модели представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации аналитик предлагает эксперту вариант модели, который подвергают обсуждению, рецензированию и редактированию.

6. *Отделение «организации» от «функций»*. Исключение влияния организационной структуры на функциональную модель.

2.1.1.1. Основные понятия и состав IDEF0-модели

Состав и изображение IDEF0-модели приведено на рис. 2.1.

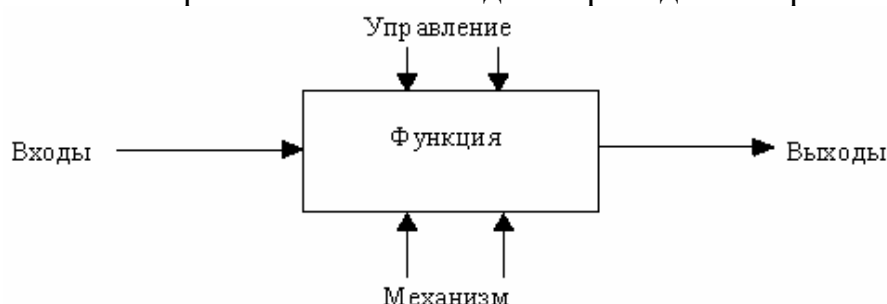


Рис. 2.1. Состав IDEF0-модели

Исходя из названия и информационного наполнения, основным структурным элементом IDEF0-методологии является **функция**, которая определяет процессы, действия, операции. Имя функции задается глаголом (например, определить стоимость, выполнить операцию). Второй структурный элемент IDEF0-методологии – это стрелка. Стрелки бывают пяти видов:

- входная стрелка, которая показывает то, что необходимо для выполнения функции (детали, заказы);
- выходная стрелка, которая является результатом выполнения функции (прибыль, готовая продукция);
- стрелка-механизм – это то, с помощью кого или чего выполняется функция (сотрудники, оборудование);
- стрелка-управление, которая регламентирует выполнение функции (устав, ГОСТы);
- стрелка-вызов представляет собой техническую стрелку, которая необходима для слияния/расщепления моделей, не несет информативной нагрузки.

Все стрелки (кроме стрелки-вызова) могут быть классифицированы на два вида: внутренние и граничные стрелки (рис. 2.2).

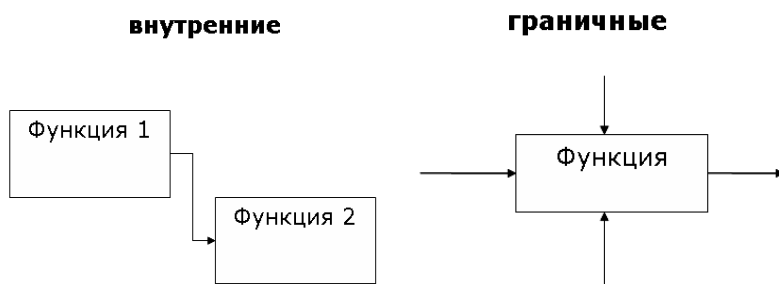


Рис. 2.2. Внутренние и граничные стрелки

В общем виде IDEF0-модель представляет собой набор согласованных диаграмм, фрагмента текста и глоссария (словаря данных). Диаграмма – часть модели, состоящая из взаимосвязанных блоков. Существует специальный вид диаграммы, который называется контекстной диаграммой. Контекстная диаграмма – это диаграмма самого верхнего уровня (уровень А-0), представляющая систему в общем, в виде «черного ящика», и связывающая ее с внешним миром с помощью интерфейсных дуг. Контекстная диаграмма состоит из одного функционального блока, любого количества стрелок, цели моделирования и точки зрения. Пример контекстной диаграммы приведен на рис. 2.3. Цель моделирования указывает, для чего разрабатывается конкретная модель. Точка зрения определяет должностное лицо или подразделение организации, с точки зрения кого разрабатывается модель.

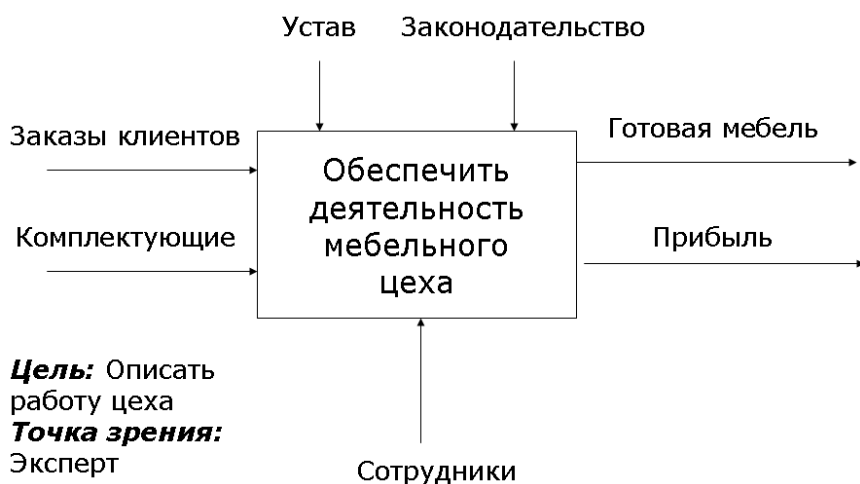


Рис. 2.3. Пример контекстной диаграммы

После разработки контекстной диаграммы проводят процесс декомпозиции. Декомпозиция – это разбиение функции на подфункции, т. е. более детальное ее представление. Говоря о декомпозиции, следует упомянуть об ICOM-кодогенерации (Input, Control, Output, Mechanism),

которая позволяет сохранить целостность модели. На практике ICOM-кодогенерация – это процесс, который автоматически переносит стрелки, присоединенные к функциональному блоку на диаграммы декомпозиции (диаграммы-потомки). Таким образом поддерживается связь между диаграммами-родителями и диаграммами-потомками, сохраняется целостность модели.

Для схожих целей в IDEF0-модели существует понятие туннелирования, или туннельной стрелки. Туннельная стрелка – это специальный вид стрелки (это может быть вход, выход, механизм или управление), которая на модели отображается в виде круглых или квадратных скобок. Квадратные скобки предупреждают разработчика, что в модели появилась ошибка. Квадратные скобки необходимо либо совсем убрать, либо заменить на круглые. Круглые скобки у блока или границы означают, что стрелка является туннельной. Туннель у границы показывает, что этой стрелки нет на диаграмме-родителе, т. е. на верхнем уровне декомпозиции эта стрелка не важна. Туннель у блока говорит о том, что эта стрелка не важна на диаграмме-потомке, и там она не отобразится. Пример туннельных стрелок приведен на рис. 2.4.

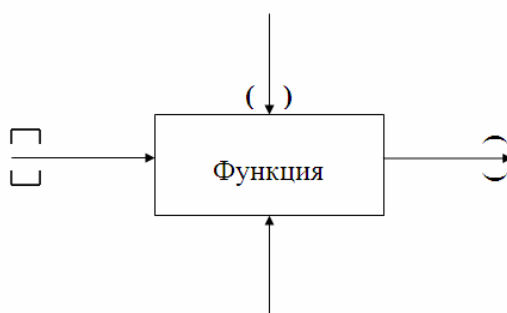


Рис. 2.4. Пример туннельных стрелок

В IDEF0-модели также могут быть стрелки ветвления и слияния, существуют правила отображения этих стрелок в модели. Пример стрелок приведен на рис. 2.5 и рис. 2.6 (*а* – неверный способ отображения, *б* – верный способ отображения).

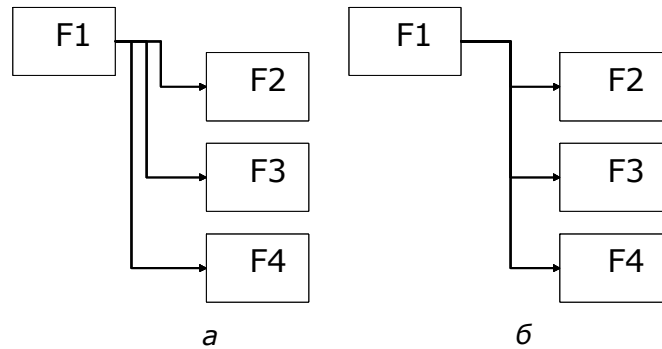


Рис. 2.5. Способ отображения стрелок ветвления:
a - неверный способ отображения; *б* - верный способ отображения

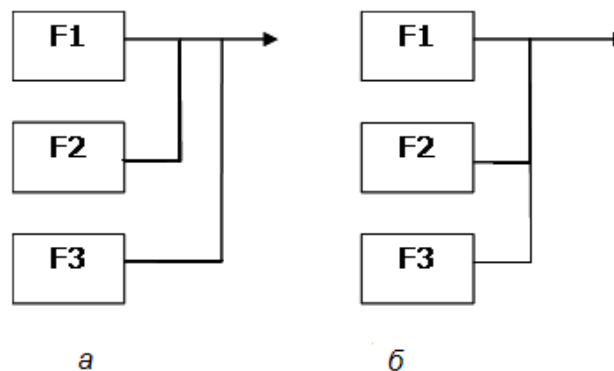


Рис. 2.6. Способ отображения стрелок слияния:
a - неверный способ отображения; *б* - верный способ отображения

Нумерация блоков в IDEF0-модели представляет собой отображение префикса (чаще всего используют А) и описание всей иерархии (рис. 2.7).

А 6.1.1

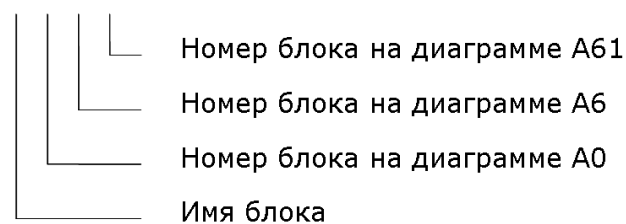


Рис. 2.7. Пример нумерации блока в модели

Между функциями в модели существуют определенные типы отношений:

– **доминирование**. Это один из распространенных типов отношений между функциями. Отношение доминирования имеет два возможных значения: во-первых, блоки, расположенные выше, более важны и доминантны в рамках рассматриваемой предметной области,

во-вторых, блоки, расположенные выше, выполняются раньше по времени, например, если рассмотреть начальную стадию работы промышленного предприятия, то первой функцией будет проведение маркетинговых исследований, затем проектные работы, после чего закупка материалов, производство и продажа готовой продукции (рис. 2.8);

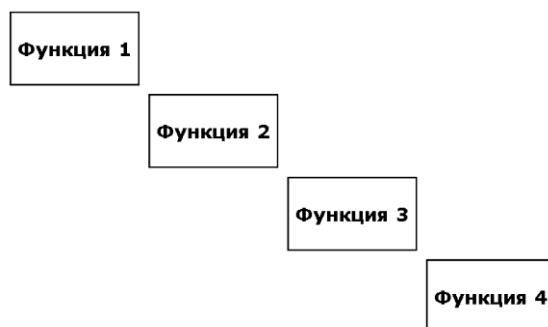


Рис. 2.8. Отношение доминирования

– **управление**. Этот тип отношения используется довольно редко, т. к. результат первой функции – это управляющее воздействие для других функций. В качестве примера можно привести следующее: первая функция «разработать учебно-методические указания», выход функции – «учебно-методические указания», тогда вторая функция – «выполнить лабораторную работу». Пример приведен на рис. 2.9;

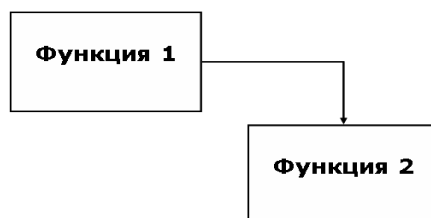


Рис. 2.9. Отношение управления

– **выход - вход**. Самый применяемый тип отношения, когда выход одной функции является входом для другой (рис. 2.10);

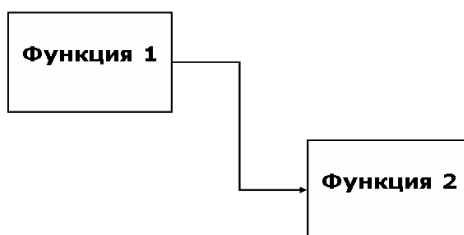


Рис. 2.10. Отношение выход-вход

– **обратная связь (ОС) по управлению**. Выход одной функции является управлением для другой, схож с отношением управления (рис. 2.11);



Рис. 2.11. Отношение *обратная связь по управлению*

– **ОС по входу**. Является аналогом отношения выход-вход (рис. 2.12);



Рис. 2.12. Отношение *обратная связь по входу*

– **выход-механизм**. Редкий тип отношения, в качестве примера можно привести следующее: предприятие занимается выпуском продукции, а потом в своей дальнейшей деятельности использует это оборудование на других этапах (рис. 2.13).

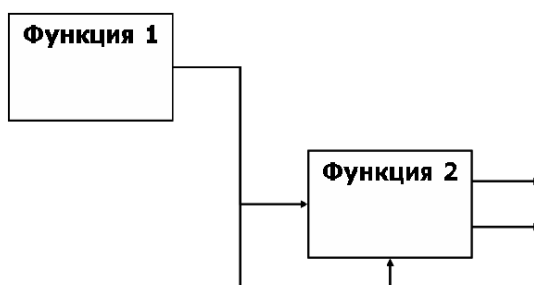


Рис. 2.13. Отношение *выход-механизм*

2.1.1.2. Правила построения диаграмм

1. В состав модели обязательно должна входить контекстная диаграмма уровня А-0.
2. Блоки на диаграмме должны располагаться по диагонали (отношение доминирования).
3. Неконтекстные диаграммы должны содержать количество блоков от 3 до 6.
4. Имена функций и стрелок должны быть уникальными. Имена функций должны быть заданы глаголом. Имена стрелок – именем существительным.
5. При разработке модели необходимо стремиться к уменьшению количества необязательных пересечений стрелок, минимизировать число петель и поворотов каждой стрелки (рис. 2.14 *а* – неверный способ отображения, *б* – верный способ отображения).

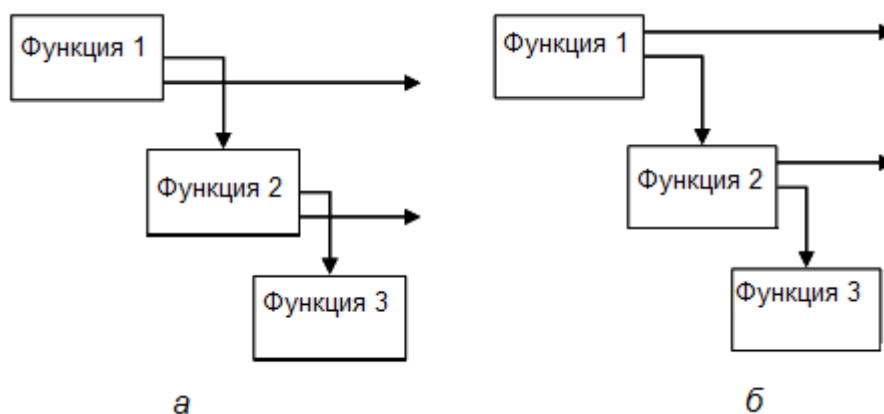


Рис. 2.14. Пример изображения стрелок в модели:
а - неверный способ отображения; *б* - верный способ отображения

6. Стрелки должны объединяться, если имеют общий источник (рис. 2.15 *а* – неверный способ отображения, *б* – верный способ отображения).

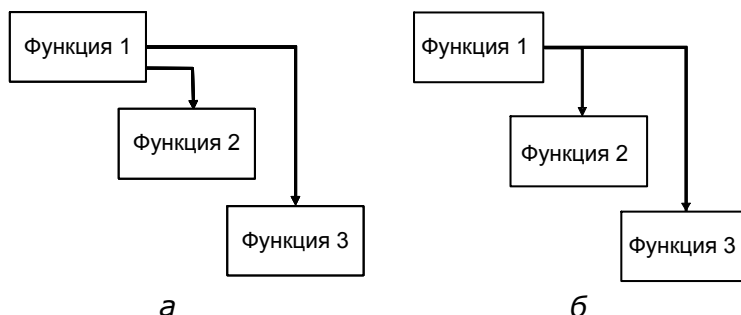


Рис. 2.15. Пример изображения ветвления стрелок в модели:
а - неверный способ отображения; *б* - верный способ отображения

На рис. 2.16 приведен пример IDEF0-модели деятельности промышленного предприятия (*а* – контекстная диаграмма, *б* – диаграмма декомпозиции).



Рис. 2.16. Пример IDEF0-модели деятельности промышленного предприятия
а - контекстная диаграмма (уровень A-0)

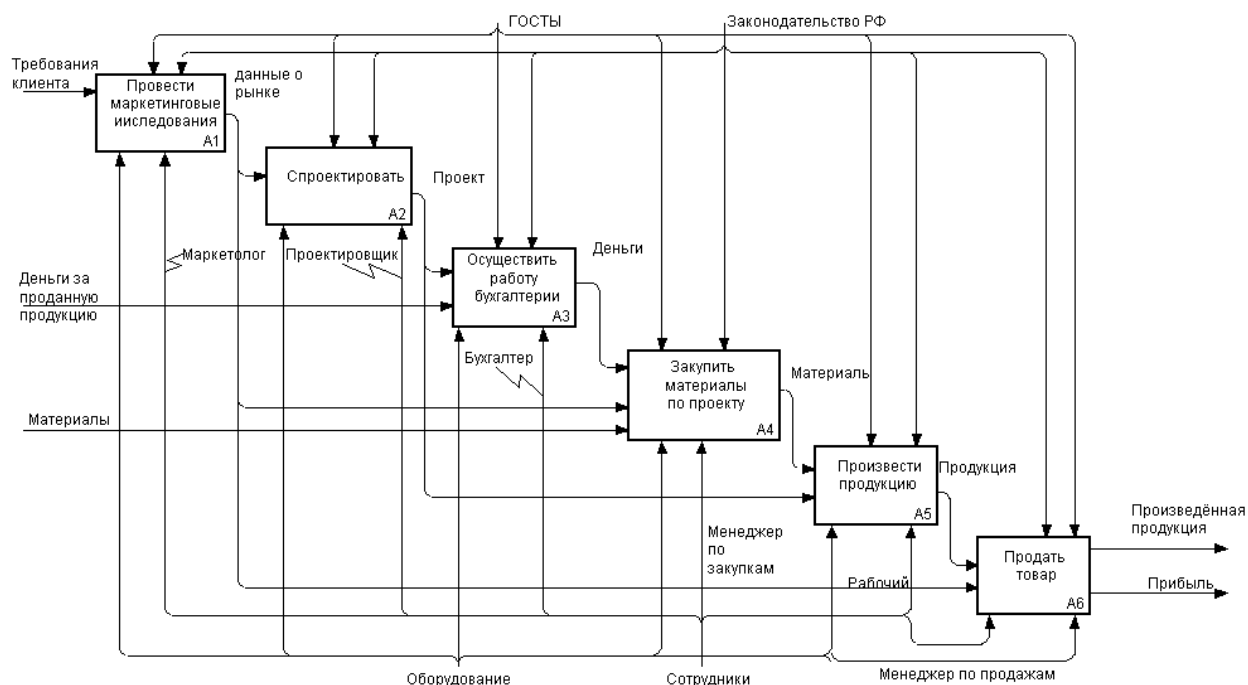


Рис. 2.16. Пример IDEF0-модели деятельности промышленного предприятия
б - диаграмма основных бизнес-процессов (уровень A0)

2.1.1.3. Глоссарий модели (словарь данных)

Словарь данных – это определенным образом организованный список всех элементов данных системы с их точными определениями, что дает возможность различным категориям пользователей однозначно понимать терминологию предметной области. Словарь данных дает возможность разработчикам, работающим с моделью, иметь общее представление об элементах системы. Также словарь данных содержит информацию, которую не удалось отобразить в модели. В словаре осуществляется определение элементов данных, к которым относятся функции, потоки слияния и ветвления, все виды стрелок и т. д.

Потоки, хранящиеся в словаре, могут быть [7]:

- простыми или групповыми;
- внутренними или внешними;
- потоками данных или потоками управления;
- непрерывными или дискретными.

Атрибуты потока данных [7]:

– имена-синонимы потока данных в соответствии с узлами изменения имени;

- БНФ-определение;
- единицы измерения потока;
- диапазон значений;
- список значений;
- список номеров диаграмм различных типов;
- список потоков;
- комментарий.

2.1.1.4. БНФ-нотация (Бэкуса-Наура форма)

БНФ-нотация позволяет формально описать слияние или ветвление потоков в виде БНФ-спецификации. В БНФ-спецификации существуют стандартные операторы, с помощью которых и происходит детализация и определение потока данных. БНФ-спецификация начинается с символа коммерческой А «@», после которой идет оператор (ИМЯ, ТИП, БНФ, ЕДИНИЦА ИЗМЕРЕНИЯ, НОРМА, КОММЕНТАРИЙ).

Синтаксис БНФ-спецификации:

@БНФ = <простой оператор> ! <БНФ-выражение>

<простой оператор> – это текстовое описание

<БНФ-выражение> – это выражение в форме Бэкуса-Наура

Между выражениями могут использоваться следующие отношения:

- = означает «композиция из»;
- + означает логическое «И»;
- ! означает логическое «ИЛИ»;
- «» означает литерал.

Примеры БНФ-спецификаций

Пример 1

@ИМЯ = ВВЕДЕННАЯ КРЕДИТНАЯ КАРТА

@ТИП = управляющий поток

@БНФ = /указывает, что кредитная карта введена/

Пример 2

@ИМЯ = ДАННЫЕ КРЕДИТНОЙ КАРТЫ

@ТИП = поток данных

@БНФ = ПАРОЛЬ + ДЕТАЛИ КЛИЕНТА+ ЛИМИТ ДЕНЕГ

Пример 3

@ИМЯ = ДАННЫЕ КЛИЕНТА

@ТИП = поток данных

@БНФ = ФИО+ адрес + телефон + ИНН

Пример 4

@ИМЯ = ДЕНЬГИ

@ТИП = дискретный поток

@БНФ = /деньги, выдаваемые клиенту/

@ЕДИНИЦА ИЗМЕРЕНИЯ = доллар

@НОРМА = 5. .1000

@КОММЕНТАРИЙ Сумма выдаваемых денег должна делиться на 5

Пример 5

@ИМЯ = СООБЩЕНИЕ

@ТИП = поток данных

@БНФ = e-mail ! факс ! письмо

2.1.2. Методология событийного моделирования IDEF3

IDEF3-методология менее популярна, чем IDEF0, но в последнее время все чаще встречаются программные продукты, ее реализующие, и сами IDEF3-модели более интересны, т. к. позволяют описать логику

процесса за счет введения ряда новых структурных элементов. Практически IDEF3-модели используются для:

1. Документирования технологических процессов, где важна последовательность выполнения процесса.
2. Описания различных ситуаций (событий) дальнейшего развития процесса с целью прогнозирования (по принципу «что будет, если...»).
3. Принятия эффективных управленческих решений при реорганизации процессов [7].

Различают два типа IDEF3-моделей: диаграммы выполнения последовательности этапов (Process Flow Description Diagram) и диаграммы изменения состояний объекта (Object State Transition Network). Отличаются эти диаграммы точкой зрения, которая рассматривается при создании модели. Диаграммы выполнения последовательности этапов разрабатываются с точки зрения стороннего наблюдателя, а диаграммы изменения состояний объекта – с точки зрения самого рассматриваемого объекта. Наиболее часто при моделировании процессов используют диаграммы выполнения последовательности этапов, именно их в дальнейшем мы и будем подразумевать, говоря о IDEF3-моделях.

Выделяют четыре элемента IDEF3-модели.

1. **Единицы работ** (Unit of work), которые отображают действия, процессы, события, этапы выполнения работ (рис. 217). Имя задается в форме глагола, указывается номер и кто исполняет данную единицу работ.

ИМЯ (глагол)	
№	Кто выполняет

Рис. 2.17. Графическое изображение единицы работ

Говоря об единицах работ, необходимо отметить, что IDEF3-модели являются **моделями «один вход – один выход»** («single input – single output»), т. е. у любой единицы работ может быть только один вход и один выход, иначе необходимо вводить дополнительные элементы – перекрестки.

2. **Ссылки** (Referents) (см. рис. 2.18) могут выполнять две роли:

- необходимые элементы для выполнения технологического процесса либо результат технологического процесса (металл, компоненты, готовое изделие и т.п.);
- активаторы процесса (клиент, поставщик и т. п.).

Имя ссылки задается именем существительным.

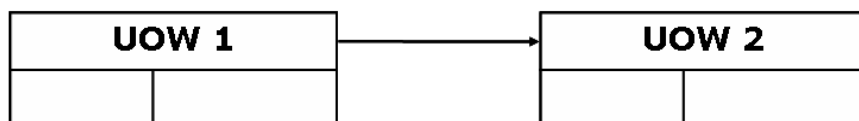


Рис. 2.18. Графическое изображение ссылки

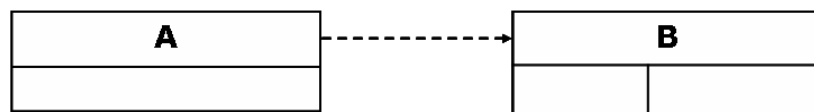
3. **Связи** (Links) отображают передачу действия от одной единицы работ к другой либо соединяют ссылку с единицей работ, т. е. активируют единицу работ.



Сплошная стрелка соединяет между собой единицы работ.



Пунктирная стрелка соединяет ссылки с единицами работ.



4. **Перекрестки** (Junctions) являются элементами модели, за счет которых описывается логика и последовательность выполнения этапов в модели. Перекресток кардинально отличает IDEF3-модель от других видов моделей, т. к. за счет него описывается событийность модели.

Перекрестки бывают двух видов: перекрестки слияния – Fan In и перекрестки ветвления – Fan Out (рис. 2.19).

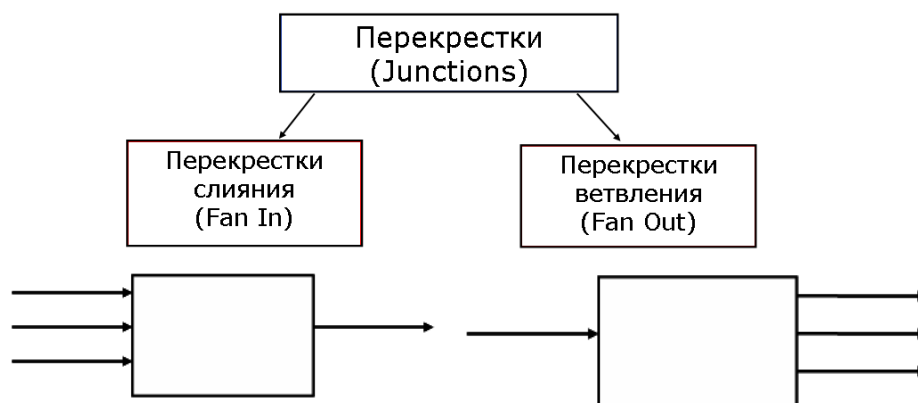


Рис. 2.19. Перекрестки слияния и ветвления

Перекресток не может быть одновременно перекрестком слияния и ветвления (рис. 2.20, *а*), т. к. в этом случае будет неясно правило его срабатывания. Эта ситуация разрешается путем введения в модель каскада перекрестков (рис. 2.20, *б*).

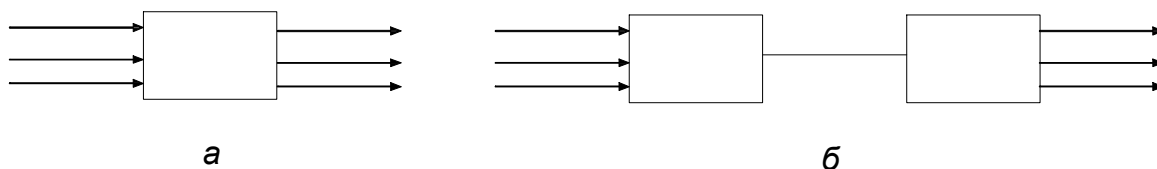







Рис. 2.20. Графическое изображение перекрестка:
а - неверный способ отображения; *б* - верный способ отображения

В свою очередь, все перекрестки могут быть пяти типов:

1.  Asynchronous AND (Асинхронное И)
2.  Synchronous AND (Синхронное И)
3.  Asynchronous OR (Асинхронное ИЛИ)
4.  Synchronous OR (Синхронное ИЛИ)
5.  XOR (Exclusive OR) (Исключающее ИЛИ)

Рассмотрим подробно правила срабатывания перекрестков.

Asynchronous AND (Асинхронное И)

Правило срабатывания перекрестка слияния (рис. 2.21, *а*): выходной процесс запустится, если завершились все входные процессы.

Вариантов срабатывания этого перекрестка – **один**.

Правило срабатывания перекрестка ветвления (рис. 2.21, *б*): после завершения входного процесса запустятся все выходные процессы.

Вариантов срабатывания этого перекрестка – **один**.

Пример: после завершения входного процесса «рассчитать клиента» запустятся все выходные процессы «пробить кассовый чек», «принять деньги» и «упаковать покупки».

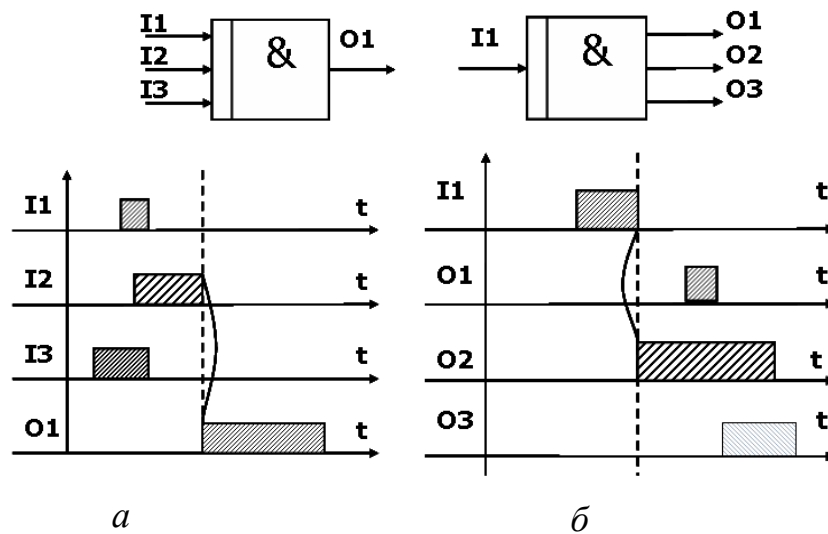


Рис. 2.21. Asynchronous AND (Асинхронное И):
a - перекресток слияния; *б* - перекресток ветвления

Synchronous AND (Синхронное И)

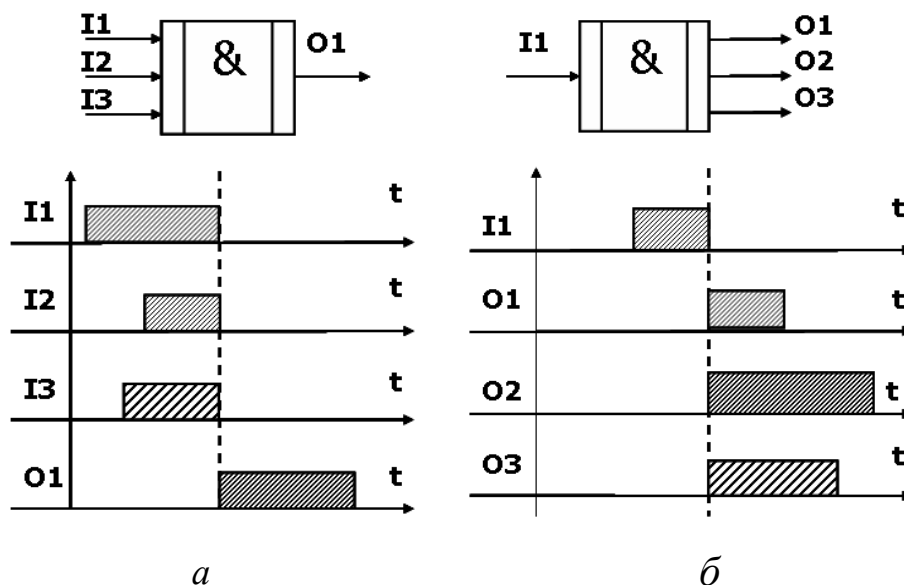


Рис. 2.22. Synchronous AND (Синхронное И):
a - перекресток слияния; *б* - перекресток ветвления

Правило срабатывания перекрестка слияния (рис. 2.22, *a*): выходной процесс запустится, если завершились одновременно все входные процессы. Одновременность не означает, что события произойдут в одну и ту же секунду, это может быть различный по длительности промежуток времени: минута, час, день (зависит от предметной области).

Вариантов срабатывания этого перекрестка – **один**.

Пример: выходной процесс «начать кирпичную кладку» начнется, если выполнены входные процессы «привезти кирпич», «приготовить раствор» и «нанять рабочих».

Правило срабатывания перекрестка ветвления (рис. 2.22, б): после завершения входного процесса запускаются все выходные процессы, причем запускаются одновременно.

Вариантов срабатывания этого перекрестка – **один**.

Asynchronous OR (Асинхронное ИЛИ)

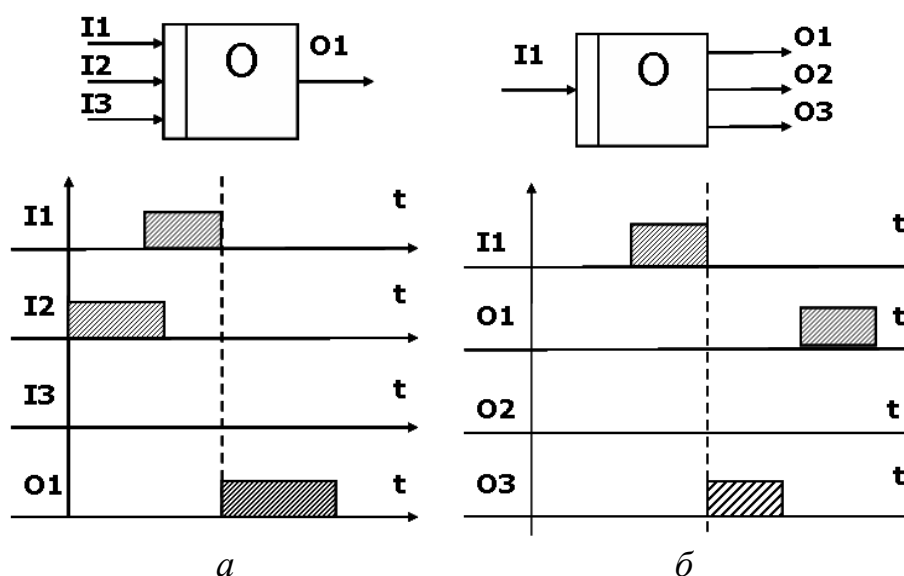


Рис. 2.23. Asynchronous OR (Асинхронное ИЛИ):
а - перекресток слияния; б - перекресток ветвления

Правило срабатывания перекрестка слияния (рис. 2.23, а): выходной процесс запустится, если завершится один или любая возможная комбинация входных процессов.

Вариантов срабатывания этого перекрестка будет $2^N - 1$, где N – количество входов перекрестка.

Правило срабатывания перекрестка ветвления (рис. 2.23, б): после завершения входного процесса запускаются один или любая возможная комбинация выходных процессов.

Вариантов срабатывания этого перекрестка будет $2^N - 1$, где N – количество выходов перекрестка.

Пример: после завершения входного процесса «написать письмо другу» запустятся первый выходной процессы «отправить обычной почтой» и третий выходной процесс «отправить электронной почтой».

Synchronous OR (Синхронное ИЛИ)

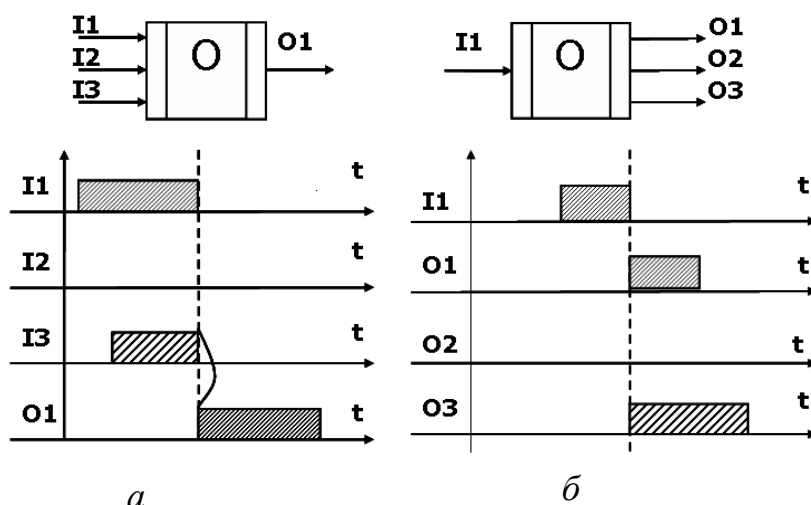


Рис. 2.24. Synchronous OR (Синхронное ИЛИ):
а - перекресток слияния; б - перекресток ветвления

Правило срабатывания перекрестка слияния (рис. 2.24, а): выходной процесс запустится, если завершится один или любая возможная комбинация входных процессов, но если сработала комбинация процессов, то тогда завершится она должна одновременно.

Вариантов срабатывания этого перекрестка будет $2^N - 1$, где N – количество входов перекрестка.

Пример: после одновременного завершения входных процессов «заштукатурить стены» и «сделать стяжку» запустится выходной процесс «вставить окна».

Правило срабатывания перекрестка ветвления (рис. 2.24, б): после завершения входного процесса запустится один или любая возможная комбинация выходных процессов, но если сработала комбинация процессов, то тогда запуститься она должна одновременно.

Вариантов срабатывания этого перекрестка будет $2^N - 1$, где N – количество выходов перекрестка.

Exclusive OR (Исключающее ИЛИ)

Правило срабатывания перекрестка слияния (рис. 2.25, а): выходной процесс запустится, если завершился только один входной процесс.

Вариантов срабатывания этого перекрестка – N , где N – количество входов перекрестка.

Правило срабатывания перекрестка ветвления (рис. 2.25, б): после завершения входного процесса запустится только один выходной процесс.

Вариантов срабатывания этого перекрестка – N , где N – количество выходов перекрестка.

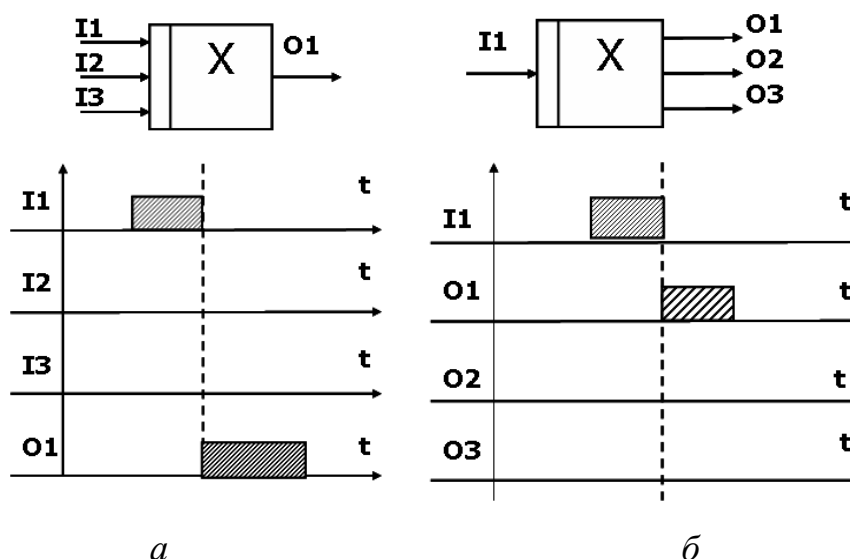


Рис. 2.25. Exclusive OR (Исключающее ИЛИ):
a - перекресток слияния; *б* - перекресток ветвления

Пример: после завершения входного процесса «завершить пошив платья» запустится только один, например первый, выходной процесс – «отдать платье заказчику». Хотя возможна и альтернатива, но тогда она исключает первый выход, например «выставить платье на продажу». Невозможно одновременно одно платье «отдать заказчику» и «выставить на продажу».

Пример IDEF3-модели разработки базы данных (БД) информационной системы приведен на рис. 2.26.

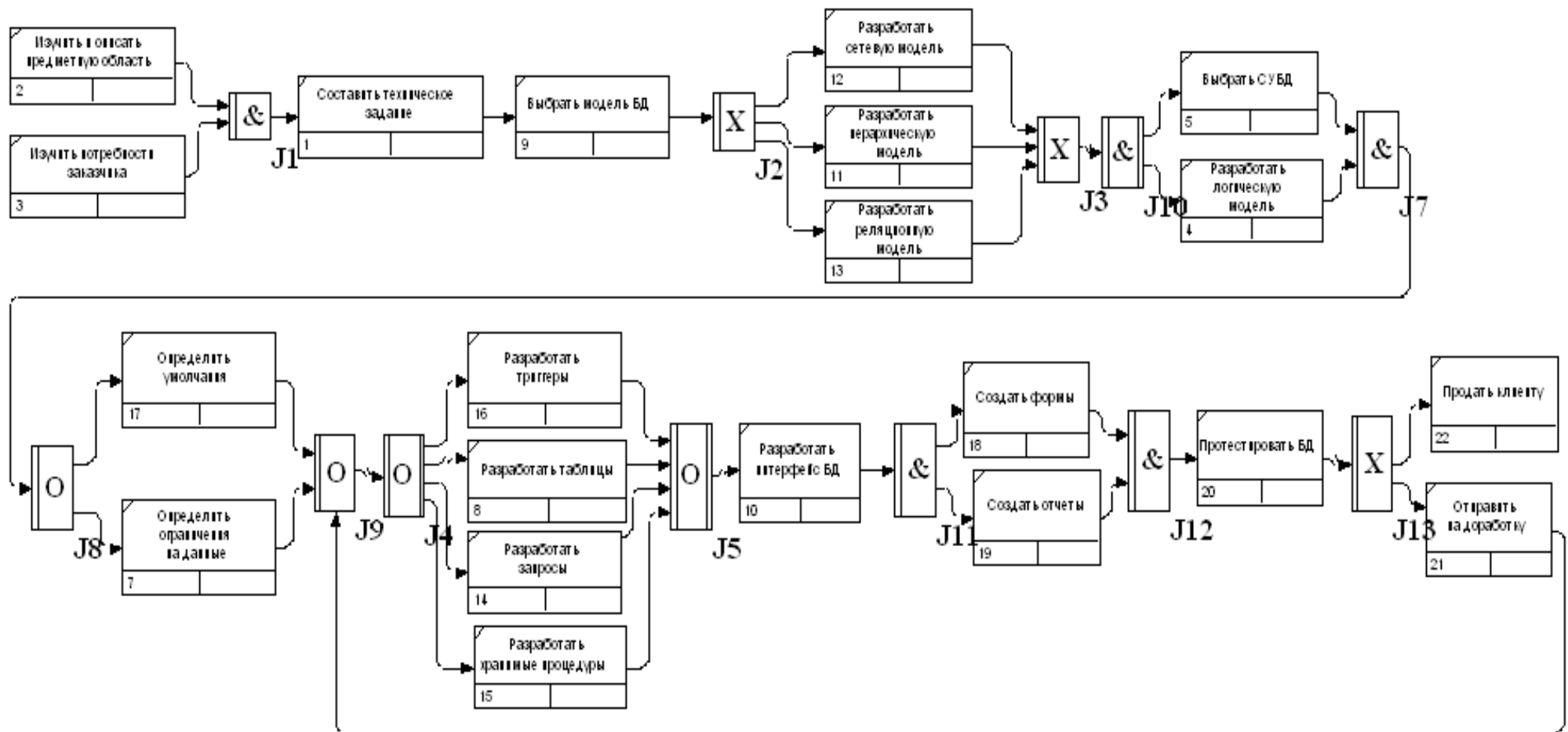


Рис. 2.26. Пример IDEF3-модели разработки базы данных

2.2. Методология моделирования потоков данных (Data Flow Diagram)

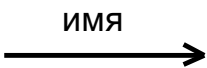
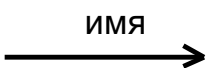






Первоначально диаграммы потоков данных разрабатывались и использовались при проектировании информационных систем. В настоящее время область применения DFD значительно расширилась и их используют:

- при проведении обследования деятельности предприятия;
- при проведении работ по реинжинирингу;
- при анализе и оптимизации бизнес-процессов предприятия;
- при внедрении систем электронного документооборота.

При построении диаграмм потоков данных наиболее часто используют две нотации: Йордана и Гейна-Сарсона [7]. Обе нотации имеют одинаковый по названиям и значению элементный состав, но имеют различное его графическое изображение (табл. 2.1).

Таблица 2.1

Графические элементы DFD

Компонента	Нотация Йордана	Нотация Гейна-Сарсона
поток данных		
процесс		
хранилище		
внешняя сущность		

Всего в DFD используется четыре структурных элемента:

1. **Процессы.** Процессы в DFD обозначают функции, операции, действия, которые обрабатывают и изменяют информацию. Процессы показывают, каким образом входные потоки данных преобразуются в выходные.

2. **Потоки данных.** Потоки данных идут от объекта-источника к объекту-приемнику, обозначая информационные потоки в системе. Взаимодействие работ с внешним миром и между собой описывается в виде стрелок (потоков данных). Поток данных соединяет выход объекта (или процесса) с входом другого объекта (или процесса).

3. **Внешние сущности.** Внешние сущности определяют элементы вне контекста системы, которые участвуют в процессе обмена информацией с системой, являясь источниками или приемниками информации. Внешние сущности изображают входы в систему и/или выходы из системы. Внешние сущности обычно изображаются на контекстной диаграмме. Внешние сущности представляют собой материальный предмет или физическое лицо, например: *ЗАКАЗЧИК, ПЕРСОНАЛ, ПОСТАВЩИК, КЛИЕНТ, СКЛАД, БАНК*.

4. **Хранилища данных.** Хранилища данных представляют собой собственно данные, к которым осуществляется доступ, эти данные также могут быть созданы или изменены процессами. Хранилище данных изображают объекты в покое и данные, которые сохраняются в памяти между последующими процессами. Информация, которую содержит хранилище данных, может использоваться в любое время после её определения. При этом данные могут выбираться в любом порядке.

В диаграммах потоков данных все используемые символы складываются в общую картину, которая дает четкое представление о том, какие данные используются и какие функции выполняются системой документооборота.

Пример DFD-модели, разработанной в нотации Гейна-Сарсона, приведен на рис. 2.27 (контекстная диаграмма) и рис. 2.28 (диаграмма основных бизнес-процессов).

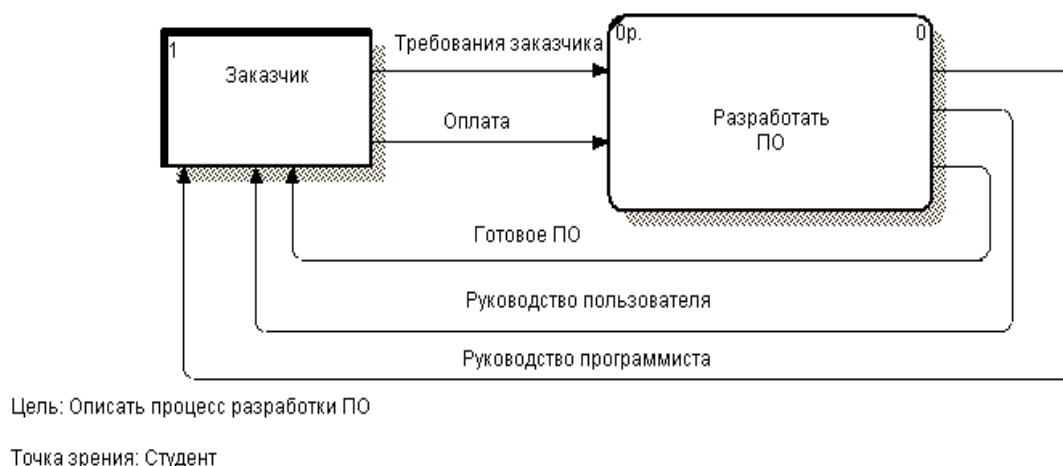


Рис. 2.27. Контекстная диаграмма DFD

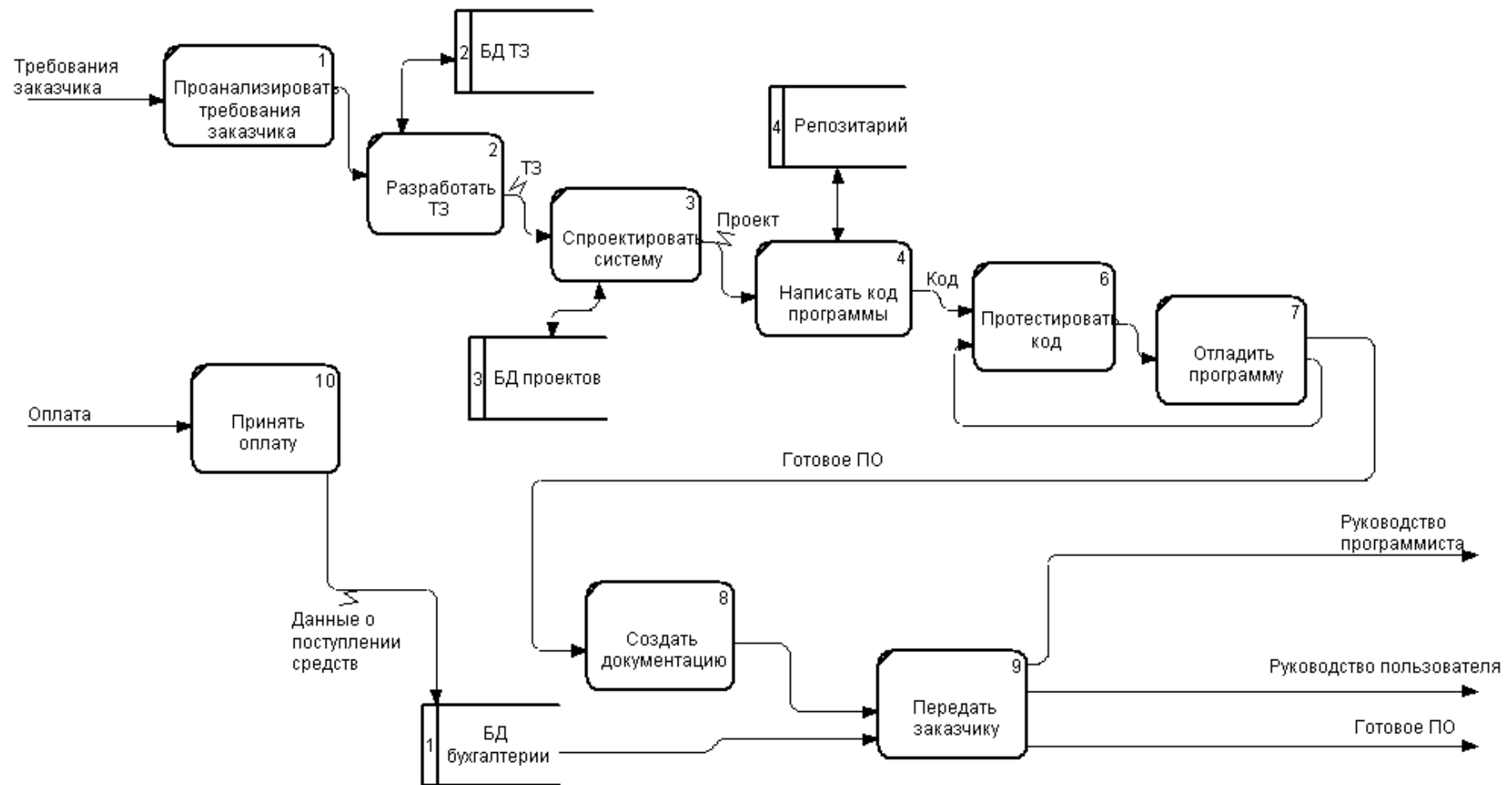


Рис. 2.28. Диаграмма основных бизнес-процессов

2.3. Вопросы и задания к главе 2

1. Что такое SADT и как SADT связана с IDEF?
2. Перечислите основные структурные элементы IDEF0-методологии.
3. Какова роль стрелки вызова и чем она отличается от других стрелок?
4. Для чего необходимы IDEF3-модели? Назовите их основное отличие от IDEF0-моделей?
5. К какому типу стрелки будут относиться *ПРИКАЗЫ РУКОВОДСТВА, АВТОТРАНСПОРТ*?
6. Чем отличаются синхронные перекрестки от асинхронных?
7. Что такое ссылка?
8. Почему перекресток «Исключающее ИЛИ» не может быть синхронным?
9. Нарисуйте временную диаграмму срабатывания перекрестка «Асинхронное И».
10. В виде какого элемента будет изображен *ЗАКАЗЧИК* в IDEF3-модели?
11. При выполнении каких проектов лучше всего использовать DFD-методологию?
12. Перечислите нотации, с использованием которых можно построить DFD-модель. В чем отличие этих нотаций?
13. Перечислите в порядке значимости элементы DFD-методологии, начиная с самого важного.
14. В виде какого элемента будет изображено *КНИГОХРАНИЛИЩЕ* на диаграмме, описывающей работу библиотеки?

Глава 3. Имитационное моделирование систем

3.1. Достоинства и недостатки имитационного моделирования систем

Как было рассмотрено в главе 1, математические модели могут быть *аналитическими, численными, алгоритмическими и имитационными*.

Когда явления в сложной системе настолько сложны и многообразны, что аналитическая модель становится слишком грубым приближением к действительности, то исследователь вынужден использовать имитационное моделирование [4, 17, 23, 27].

Имитационное моделирование – это метод исследования, заключающийся в имитации на ЭВМ (с помощью комплекса программ) процесса функционирования системы или отдельных ее частей и элементов. Сущность метода имитационного моделирования заключается в разработке таких алгоритмов и программ, которые имитируют поведение системы, ее свойства и характеристики в необходимом для исследования системы составе, объеме и области изменения ее параметров [29]. При имитационном моделировании реализующий модель алгоритм воспроизводит процесс функционирования системы во времени, причем имитируются явления, составляющие процесс, с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состояниях процесса в определенные моменты времени, дающие возможность оценить характеристики системы [14].

Имитационное моделирование позволяет осуществлять многократные испытания модели с нужными входными данными, чтобы определить их влияние на выходные критерии оценки работы системы. При таком моделировании компьютер используется для численной оценки модели, а с помощью полученных данных рассчитываются ее реальные характеристики.

Имитационное моделирование может применяться в самых различных сферах деятельности. Ниже приведен список задач, при решении которых моделирование особенно эффективно [14]:

- проектирование и анализ производственных систем;
- оценка различных систем вооружений и требований к их материально-техническому обеспечению;
- определение требований к оборудованию и протоколам сетей связи;

- определение требований к оборудованию и программному обеспечению различных компьютерных систем;
- проектирование и анализ работы транспортных систем, например: аэропортов, автомагистралей, портов и метрополитена;
- оценка проектов создания различных организаций массового обслуживания, например: центров обработки заказов, заведений быстрого питания, больниц, отделений связи;
- модернизация различных процессов в деловой сфере;
- определение политики в системах управления запасами;
- анализ финансовых и экономических систем;
- при подготовке специалистов и освоении новой техники на имитаторах (тренажёрах).

Например, имитационное моделирование может использоваться при рассмотрении производственной компанией возможности постройки больших дополнительных помещений для одного из ее подразделений, если руководство компании не уверено, что потенциальный рост производительности сможет оправдать затраты на строительство. Невозможно соорудить помещения, а затем убрать их в случае нерентабельности, в то время как моделирование работы производственной компании в ее текущем состоянии и с якобы созданными дополнительными помещениями помогает в решении этой проблемы.

В качестве второго примера можно рассмотреть случай, когда необходимо определить загруженность ресурсов (оборудование или люди) предприятия и принять управленческое решение по закупке нового оборудования или найме/увольнению сотрудников. Реальные действия могут привести к ненужным затратам: купили новое оборудование, а оно простаивает, уволили людей, а в реальности оказалось, что оставшийся персонал не справляется с объемом работы.

Имитационные модели позволяют достаточно просто учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики элементов системы, многочисленные случайные воздействия и другие, которые часто создают трудности при аналитических исследованиях. В настоящее время имитационное моделирование – наиболее эффективный метод исследования больших систем, а часто и единственный практически доступный метод получения информации о поведении системы, особенно на этапе ее проектирования [14].

В настоящее время имитационное моделирование широко применяется в мире для исследования сложных систем. Этому способствуют преимущества, присущие этому методу, а именно:

1. Большинство сложных реальных систем с вероятностными параметрами нельзя точно описать с использованием математических моделей.

2. Путем моделирования можно разработать ряд альтернативных вариантов моделей системы и затем определить, какой из них наиболее соответствует исходным требованиям.

3. Имитационное моделирование в ряде случаев гораздо менее затратное, чем проведение экспериментов с реальными системами. Тем более, что иногда эксперименты на реальных системах в принципе невозможны.

4. Моделирование позволяет изучить длительный интервал функционирования системы в сжатые сроки или, наоборот, изучить более подробно работу системы в развернутый интервал времени [14].

5. При динамическом имитационном моделировании можно получать любое количество оценок вероятностной модели, проводя ее прогоны. Подробное изучение полученных оценок приемлемо использовать при оптимизации модели.

К сожалению, несмотря на неоспоримые достоинства имитационного моделирования, в настоящее время в России этот метод исследования сложных систем используется мало, это связано с тем, что разработка таких моделей требует больших временных и стоимостных затрат. Но тенденции последнего времени вселяют надежду на то, что ситуация изменится и имитационное моделирование в России будут также широко и активно использовать, как в США, Канаде и Европе. Именно для того чтобы компенсировать этот пробел российской действительности, в курсе «Компьютерное моделирование» рассматриваются средства имитационного моделирования на примере мощного программного пакета (ПП) Arena 7.0.

3.2. Математические основы ПП Arena 7.0

В основе ПП Arena 7.0 заложен математический аппарат систем массового обслуживания (СМО) и сетей Петри. В связи с этим – для лучшего понимания процесса имитации, модулей и их параметров в ПП Arena 7.0 – рассмотрим основы этих математических аппаратов.

3.2.1. Системы массового обслуживания

Система массового обслуживания состоит из одного или более обрабатывающих устройств (сервисов), обслуживающих прибытие сущностей, ещё называемых требованиями или фишками, в систему [5]. Сущности (требования) – это индивидуальные элементы,

обрабатываемые в системе. Сущность, находящая сервис занятым, встает в очередь перед сервисом (обрабатывающим устройством). Сущности представляют собой описание динамических процессов в реальных системах. Они могут описывать как реальные физические объекты, так и нефизические объекты. Сущностями могут быть: клиенты, обслуживаемые в ресторане, больнице, аэропорту; документы, части, которые должны быть обслужены или обработаны. В бизнес-процессах – это документы или электронные отчеты (чеки, заказы, контракты). В производственных моделях, сущностями является сырье, компоненты или готовая продукция. Кроме этого, под сущностями понимают различные типы объектов, типы пакетов данных в сети, данные в программных пакетах. В табл. 3.1 приведены элементы СМО.

Таблица 3.1

Основные элементы СМО

Название элемента СМО	Назначение элемента СМО
Генераторы	Генерируют поступление сущностей в систему и временные интервалы их прибытия
Обрабатывающие устройства (сервисы)	Количество обрабатывающих устройств в системе, количество очередей, время обработки одной сущности
Очередь	Правило, по которому обрабатывающее устройство выбирает сущность для обслуживания

В зависимости от поведения сущности, поступившей в систему обслуживания в момент, когда все обрабатывающие устройства заняты, СМО делятся на три группы [1, 5]:

- системы с отказами, или системы с потерями;
- системы с ожиданием, или системы без потерь;
- системы смешанного типа.

В *системах с отказами* (системах с потерями) любая вновь поступившая сущность на обслуживание, застав все сервисы занятыми, покидает систему. Примером системы с отказами может служить работа автоматической телефонной станции: абонент получает отказ, если необходимая линия связи занята.

В *системах с ожиданием* (системах без потерь) сущность, поступившая в систему, может её покинуть только после того, как будет обслужена. В таких системах сущности, поступившие в момент, когда все сервисы заняты, образуют очередь. Примером системы обслуживания без потерь является система ремонта техники связи: неисправная техника не может быть использована без ремонта.

В *системах смешанного типа* сущность, поступившая, когда все сервисы заняты, некоторое время ожидает в очереди, и если за это время не принимается к обслуживанию, то покидает систему. Примером такой системы является обслуживание абонента в переговорном зале междугородной телефонной станции (МТС): абоненту разговор должен быть предоставлен в течение 1 часа. Если за это время разговор не состоялся, то, как правило, абонент покидает МТС.

По числу обрабатывающих устройств (сервисов) различают: *одноканальные* СМО и *многоканальные* СМО.

В свою очередь, многоканальные системы могут состоять из однотипных и разнотипных (по пропускной способности) каналов.

По числу сущностей, которые могут одновременно находиться в обслуживающей системе, различают системы с *ограниченным* и *неограниченным потоком* требований.

Существуют системы обслуживания, в которых обрабатывающие устройства расположены последовательно (пронумерованы). Очередное требование поступает сначала на первое из них и лишь в том случае, если он занят, передается второму и т. д. Такие системы называются *упорядоченными*. Все остальные системы обслуживания, в которых требования распределяются между обрабатывающими устройствами по любому другому принципу, относятся к числу *неупорядоченных* систем.

По характеру источника сущностей (генератора) различают СМО с *конечным* и *бесконечным* количеством требований на входе, соответственно различают *замкнутые* и *разомкнутые* СМО. В первом случае в системе циркулирует конечное, обычно постоянное количество требований, которые после завершения обслуживания возвращаются в генератор.

Кроме того, все СМО можно разделить по дисциплине обслуживания [27].

Дисциплина обслуживания определяется правилом, которое устройство обслуживания использует для выбора из очереди следующего требования (если таковые есть) по завершении обслуживания текущего требования. Обычно используются такие дисциплины очереди:

- FIFO (First-In, First-Out): требования обслуживаются по принципу «первым прибыл – первым обслужен»;
- LIFO (Last-In, First-Out): требования обслуживаются по принципу «последним прибыл – первым обслужен»;
- приоритет: требования обслуживаются в порядке их значимости.

3.2.2. Сети Петри

Часто аналитики в задачах анализа и синтеза сложных систем обращаются к формальным системам, основанным на использовании сетей Петри. Структура сети Петри задается ориентированным двудольным мультиграфом, в котором одно множество вершин состоит из позиций, а другое множество – из переходов [11].

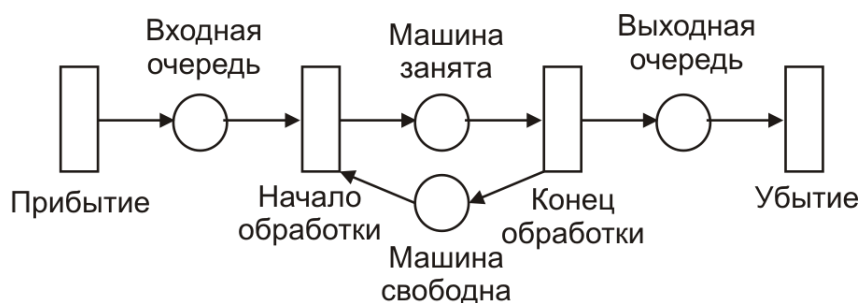
Сеть Петри – это направленный двусторонний граф, состоящий из позиций (P) и переходов (T). Основные элементы сети Петри представлены в табл. 3.2.

Таблица 3.2

Элементы сетей Петри

Название элемента	Изображение элемента
Позиция (P)	○
Переход (T)	□
Дуга	→ ←

Переходы в сети Петри являются событиями, которые изменяют состояния в реальной системе. На рис. 3.1 приведен пример интерпретации сети Петри.



Формальный аппарат сетей Петри предназначен для моделирования систем различного рода и отражает состояния исследуемой системы состоянием сети. Состояние сети Петри определяется ее маркировкой. Количество и распределение фишек сети определяют динамику исследуемой системы. Сеть Петри выполняется посредством запусков переходов в результате удаления фишек из его входных позиций и добавления их в выходные позиции перехода. Последовательность срабатываний переходов полностью определяет поведение сети. Таким образом, сеть Петри описывает структуру системы, ее состояние и поведение.

Среди достоинств аппарата сетей Петри можно указать следующие:

- позволяет моделировать асинхронность и недетерминизм параллельных независимых событий (в сети Петри могут одновременно и независимо друг от друга сработать несколько переходов), конфликтные взаимодействия между процессами;
- позволяет использовать единые методологические позиции для описания программного обеспечения, аппаратных средств и информационного обмена между системами;
- предоставляет возможность введения любой степени иерархической детализации описываемых программных и аппаратных подсистем модели;
- имеет большую анализирующую мощност, которая позволяет формальными средствами доказывать существование или отсутствие определенных состояний сети Петри.

Однако формальная модель сетей Петри, в силу своей универсальности, имеет ряд недостатков, затрудняющих практическое применение для моделирования сложных систем. К основным таким недостаткам можно отнести следующие:

- высокая трудоемкость анализа сетей большой размерности, а реальные бизнес-процессы предприятия моделируются именно сетями большой размерности;
- описательная мощност сетей Петри недостаточна для содержательного моделирования систем;
- обычные сети Петри не отражают требуемые временные характеристики моделируемой системы;
- фишка сети Петри не представляет собой никакой информации, кроме самого факта ее наличия, поэтому чрезвычайно сложно отразить преобразование информации при срабатывании переходов сети Петри;
- невозможность проведения логических преобразований и как следствие – невозможность управления продвижением фишек по сети.

Недостатки сетей Петри не позволяют описывать сложные системы и в настоящее время используются для описания простейших операций. Также эти факторы явились причиной разработки подклассов и расширений сетей Петри, в которых вводятся определенные ограничения на структуру сети, что позволяет использовать более простые алгоритмы для ее анализа либо дополнительные элементы формальной системы, призванные увеличить ее описательную мощность.

Большого внимания заслуживают сети высокого уровня, такие, как раскрашенные сети Петри (Color Petri Net), являющиеся модификацией сетей Петри и отличающиеся хорошо разработанным математическим аппаратом, широко применяемые для самых разнообразных практических целей. Основной причиной высокой эффективности этих формальных моделей является то, что они без потери возможностей формального анализа позволяют исследователю получить значительно более краткие и удобные описания, чем те, которые могут быть сделаны с помощью сетей низкого уровня. В сетях высокого уровня сложность моделей может быть разделена между структурой сети, надписями и описаниями. Это позволяет осуществлять описание значительно более сложных систем и анализировать процессы преобразования данных с помощью общепринятых математических выражений вместо сложного набора позиций, переходов и дуг. Раскрашенные сети Петри, в отличие от обычных сетей Петри, позволяют описывать структуру системы в виде иерархии диаграмм.

Но у данного аппарата моделирования также не устранен ряд недостатков, которые присущи сетям Петри. К таким недостаткам можно отнести:

- необходимость знания разработчиком специфического языка описания моделей;
- отсутствие использования принципов объектно-ориентированного подхода;
- низкая гибкость и трудоемкость описания систем в случае их декомпозиции до уровня некоторых элементарных бизнес - операций.

Раскрашенные сети Петри до сих пор применяются для моделирования сложных систем.

Все недостатки СМО и сетей Петри учтены и устранены разработчиками ПП Arena 7.0. Кроме того, этот программный пакет имеет множество необходимых операторов, законов распределения и других элементов, которые привели к его широкому распространению.

Хотелось бы добавить несколько слов о том, почему Arena 7.0 является программным пакетом. Это связано с тем, что Arena 7.0 кроме

основного модуля моделирования и анализа систем, имеет следующие встроенные программные средства:

1. ***Input Analyzer***. Это средство позволяет анализировать входные данные, определять закономерности входных данных для дальнейшего их использования при моделировании систем.

2. ***Output Analyzer***. Это средство позволяет анализировать выходные данные, полученные в результате проведенных экспериментов с моделью.

3. ***Process Analyzer***. Меняет значения параметров модели, структуру модели, занятость ресурсов, их полезность и т. д., сравнивает альтернативные сценарии и выбирает тот сценарий, который имеет наилучший результат. Сравнивая эти сценарии работы модели, можно определить лучшее решение (но не оптимальное, т. к. нельзя просмотреть все возможные решения, т.е. исследовать полностью область допустимых решений), но все-таки определить лучшее решение таким способом возможно

4. ***Генератор отчетов***. Выводит данные по результатам моделирования в виде текстовых данных, графиков, диаграмм.

5. ***Visio Process Analyzer***.

6. ***OptQuest***. Является инструментом оптимизации задач, предназначен и специально настроен для анализа результатов моделирования, выполненного с помощью пакета Arena.

Система имитационного моделирования **Arena** – основной программный продукт Systems Modeling. Корпорация Systems Modeling была основана в 1982 г. Деннисом Педгеном, автором SIMAN – первого промышленно-ориентированного общецелевого языка имитационного моделирования. В настоящее время область деятельности Systems Modeling включает в себя имитационное моделирование и разработку технологического программного обеспечения [30, 32, 34].

Система Arena позволяет моделировать виды деятельности, представленные на рис. 3.2.

С помощью Arena можно достичь основных целей моделирования сложных систем:

- понять, как устроен исследуемый объект: какова его структура, основные свойства, законы развития и взаимодействие с окружающей средой;

- выявить «узкие места» в материальных, информационных и других потоках;

– выделить переменные, наиболее важные для успешного функционирования моделируемой системы, и проанализировать имеющиеся между ними связи;

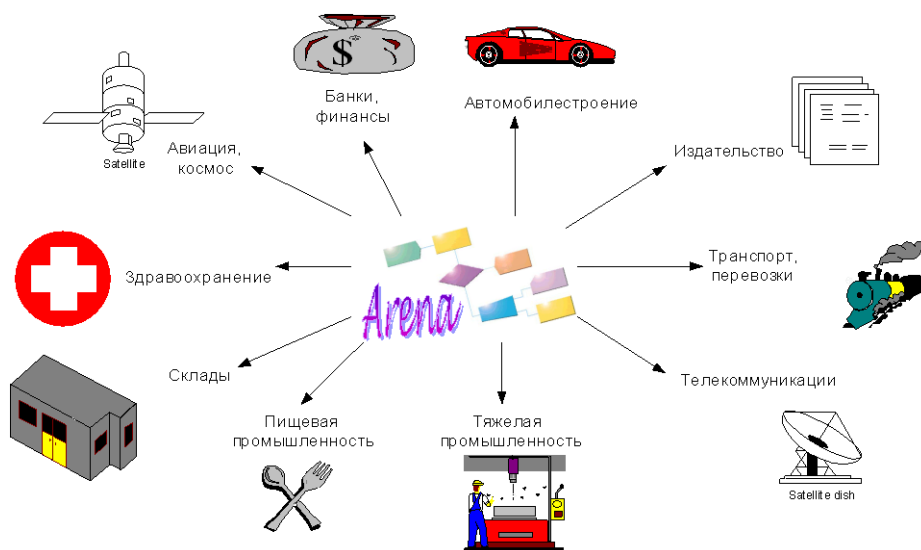



Рис. 3.2. Области применения Arena

- научиться управлять системой, определять наилучшие способы управления при заданных целях и критериях;
- прогнозировать прямые и косвенные последствия реализации заданных форм и способов воздействия на систему.

3.3. Начало работы с программным пакетом Arena 7.0

3.3.1. Создание модели с помощью ПП Arena 7.0

Для того чтобы создать новую модель, необходимо открыть ПП Arena 7.0 через Пуск → Rockwell Software → Arena7.0 → Arena7.0.1. После запуска Arena автоматически открывается новый файл. Модули помещаются на панель методом «drag & drop», соединяются с помощью коннектора . Если модуль остается «горячим» (т. е. выделенным), то при помещении нового модуля на рабочую область (окно блок-схемы) эти модули автоматически соединяются друг с другом.

Среда моделирования Arena представлена на рис 3.3.

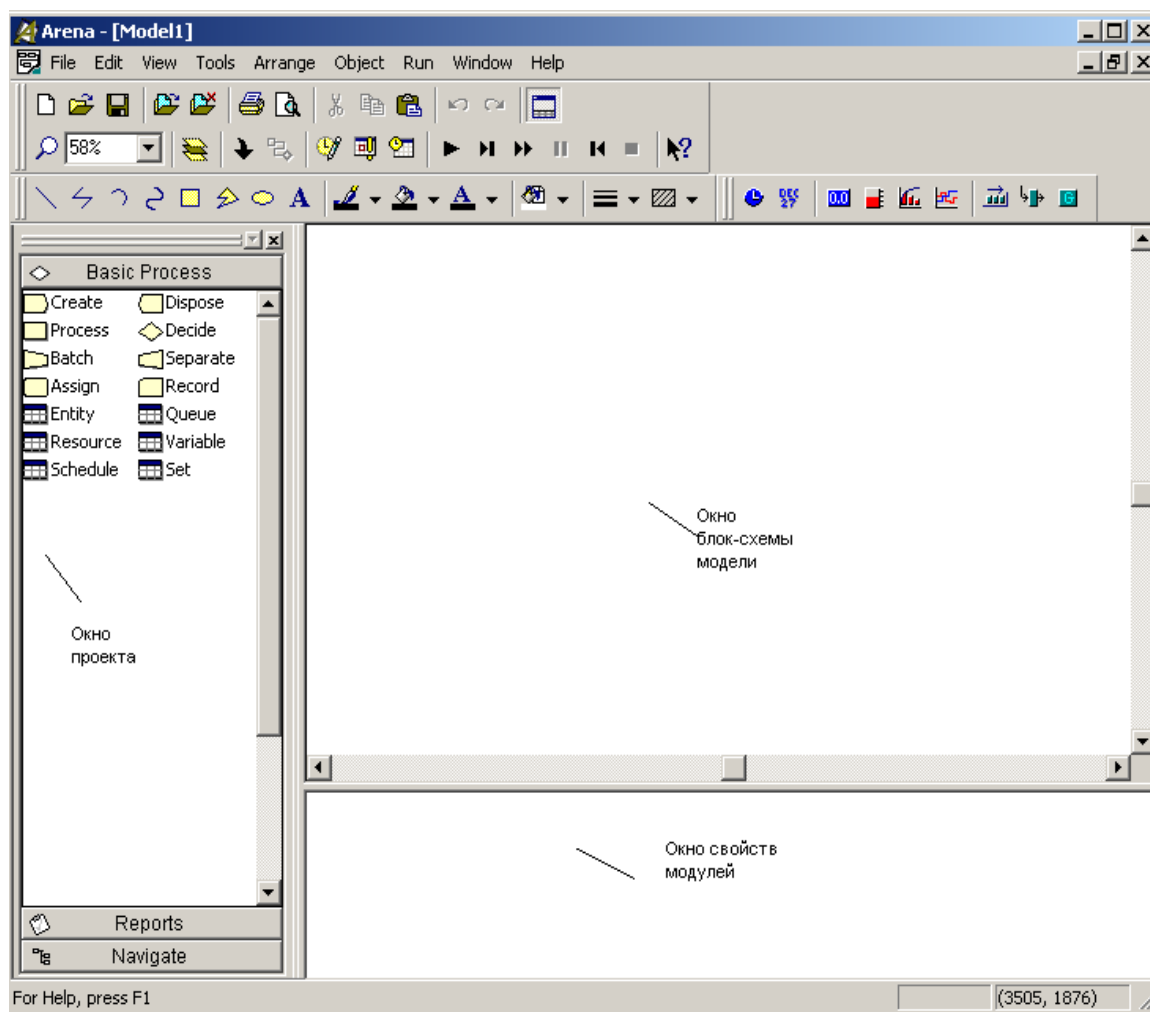


Рис. 3.3. Среда моделирования Arena

Окно приложения разделено на три области:

1. *Окно рабочего поля модели*, в котором описывается логика модели с использованием схемных (графических) модулей. Окно рабочего поля представляет графику модели, включая блок-схему процесса, анимацию и другие элементы.

2. *Окно свойств модулей*, в котором отображаются свойства всех модулей (как модулей данных, так и схемных), имеющих и используемых в модели.

3. *Окно проекта* – это навигатор системы, в котором отображается рабочая панель со всеми модулями и другие доступные и открытые панели.

Окно проекта включает в себя несколько панелей:

1. Basic Process Panel (панель основных процессов) – содержит модули, которые используются для моделирования основной логики системы.

2. Advanced Process Panel (панель усовершенствованных процессов) – содержит дополнительные модули для создания моделей со сложной логикой процесса.

3. Advanced Transfer Panel (панель перемещения) – содержит специально разработанные блоки для моделирования процесса перемещения объектов с помощью транспортера или конвейера.

4 Reports (панель отчетов) – панель сообщений: содержит сообщения, которые отображают результаты имитационного моделирования.

5. Navigate (панель навигации) – панель управления позволяет отображать все виды модели, включая управление через иерархические подмодели.

Таким образом, для того чтобы разрабатывать имитационные модели с использованием ПП Arena, необходимо изучить 3 основные панели: Basic Process Panel, Advanced Process Panel и Advanced Transfer Panel.

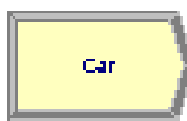
Каждая из этих панелей состоит из двух типов модулей: схемных модулей (Flowchart Modules) и модулей данных (Data Modules).

Рассмотрим более подробно состав каждой панели, свойства и назначение каждого модуля.

3.4. Basic Process Panel (панель основных процессов)

3.4.1. Схемные модули

Модуль Create



Этот модуль является отправной точкой для сущностей в имитационной модели. Сущности – это индивидуальные элементы, обрабатываемые в системе. Создание сущностей модулем происходит по расписанию или же, основываясь на значении времени между прибытиями сущности в модель. Покидая модуль, сущности начинают обрабатываться в системе. Тип создаваемых сущностей определяется в этом модуле.

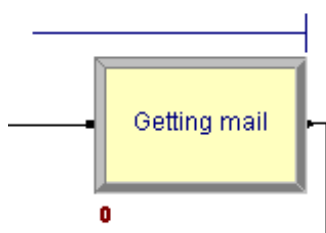
Применение: прибытие различных документов в сфере бизнеса (например: заказы, чеки, документация); прибытие клиентов в сфере обслуживания (например: в ресторан, в магазин); начало изготовления продукции на производственной линии.

Таблица 3.3

Параметры модуля Create

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Entity Type	Название типа сущности, который будет создаваться модулем
Type	Способ формирования потока прибытия. Type может иметь значения: <i>Random</i> (используется экспоненциальное распределение со средним значением, определенным пользователем), <i>Schedule</i> (определяется модулем Schedule), <i>Constant</i> (будет использоваться постоянное значение, определенное пользователем) или <i>Expression</i> (поток прибытия будет формироваться по определенному выражению)
Value	Определяет среднее значение времени между прибытиями сущностей
Schedule Name	Имя расписания, которое определяет характер прибытия сущности в систему
Expression	Этот параметр задает тип распределения или любое выражение, определяющее время между прибытиями сущностей в модель
Units	Единицы измерения времени между прибытиями (день, час, минута, секунда)
Entities per arrival	Количество сущностей, входящих в систему за одно прибытие
Max arrivals	Максимальное число сущностей, которое может создать этот модуль (ресурс генератора)
First Creation	Время, через которое прибует первая сущность в модель, от начала моделирования

Модуль Process



Этот модуль является основным модулем процесса обработки сущностей в имитационной модели. В модуле имеются опции использования ресурсов, т. е., как и при любой обработке, захватываются какие-то ресурсы. Кроме стандартного модуля Process, можно использовать подмодель, придавая

ей особую, определенную пользователем, иерархическую логическую схему. В модуле можно также задавать добавочные стоимостные и временные характеристики процесса обработки сущности.

Наиболее частое применение модуля Process: проверка документов; выполнение заказов; обслуживание клиентов; обработка деталей.

Таблица 3.4

Параметры модуля Process

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Type	Определяет логическую схему модуля. <i>Standard</i> означает, что логическая схема находится внутри модуля и зависит от параметра Action. <i>Submodel</i> показывает, что логическая схема будет находиться ниже в иерархической модели. Подмодель может содержать любое количество логических модулей
Action	Тип обработки, происходящей внутри модуля, может быть четырех типов: <i>Delay</i> просто показывает, что процесс занимает какое-то время и не отражает использование ресурсов; <i>Seize Delay</i> указывает на то, что в этом модуле были размещены ресурсы и будет происходить их захват и задержка, ресурсы будут захватываться (т. е. будут заняты обработкой сущности), а их освобождение будет происходить позднее с помощью какого-то другого модуля; <i>Seize Delay Release</i> указывает на то, что ресурсы были захвачены, а затем (через время) освободились, и <i>Delay Release</i> означает, что ресурсы до этого были захвачены сущностью, а в таком модуле сущность задержится и освободит ресурс. Все эти параметры доступны только тогда, когда Type = Standard
Priority	Значение приоритета модулей, использующих один и тот же ресурс где угодно в модели. Это свойство не доступно, если Action = Delay (или Delay Release) или когда Type = Submodel
Resources	Определяет ресурсы или группы ресурсов, которые будут обрабатывать сущности в этом модуле
Delay Type	Тип распределения или процедура, определяющая параметры задержки

Units	Единицы измерения времени задержки (день, час, минута, секунда)
Allocation	Определяет стоимостные характеристики обработки. Value Added – означает учитывать стоимостные характеристики, а Non-Value Added – не учитывать
Minimum	Поле, определяющее минимальное значение для равномерного и треугольного распределения
Maximum	Поле, определяющее максимальное значение для равномерного и треугольного распределения
Value	Поле, определяющее среднее значение для нормального и треугольного распределения или значения для постоянной временной задержки
Std Dev	Параметр, определяющий стандартное отклонение для распределения
Expression	Поле, в котором задается выражение, определяющее значение временной задержки, если Delay Type = Expression

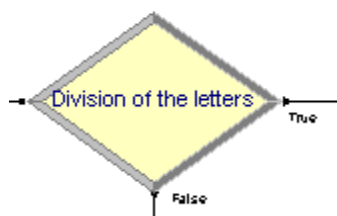
Более подробно остановимся на параметре Priority (приоритет) модуля Process. Говоря об этом параметре, мы должны ввести понятие «приоритет ресурса» и «приоритет очереди». Рассмотрим пример и объясним, что такое «приоритет ресурса».

На прием к доктору приходят пациенты двух типов: взрослые и дети. Доктор (наш ресурс) – один. Он ведет прием и детей, и взрослых, но детей доктор принимает около 30 минут, а взрослых около 20 минут, причем у детей приоритет выше, чем у взрослых.

Каким образом мы можем реализовать это с помощью модуля Process? Во-первых, параметр Action этого модуля должен быть установлен Seize Delay Release для назначения ресурса, т. е. когда сущность «пациент» зайдет в модуль, то она захватит ресурс «доктор» на определенное время. Во-вторых, у нас по условию время обслуживания пациентов различное; таким образом, мы процесс обслуживания пациентов доктором смоделируем в виде двух блоков Process с разными временными задержками (в 30 и 20 минут), но одним и тем же ресурсом «доктор». В-третьих, чтобы установить приоритет у детей выше, мы в параметре Priority в том процессе, где время обслуживания 30 минут, т. е. обслуживание детей, установим приоритет – High, а во втором процессе – Low или Medium. Таким образом, когда у нас будут приходить сущности «дети», они будут иметь наивысший приоритет в обслуживании.

Рассмотрение понятия «приоритет очереди» будет приведено ниже (см. модуль данных очередь Queue).

Модуль Decide



Этот модуль позволяет описать и задать логику модели, учитывая принятие решений. Он включает опции принятия решений, основанных на условии *By Condition* (например, если тип сущности Car) или основанных на вероятности *By Chance* (например, 75 % – true, а 25 % – false).

Условия могут быть основаны на значении атрибута *Attribute*, значении переменной *Variable*, типе сущности *Entity Type* или основанные на выражении *Expression*.

Если поставленное условие выполняется, то сущности будут покидать модуль через ветку *True*, иначе – по ветке *False*.

Данный модуль позволяет выполнять проверку не только одного условия, но и нескольких. Это достигается с помощью свойства *Type→N-way by Chance/by Condition*. В зависимости от условия сущность идет по нужной ветке. Таким образом, по ветке *True* у модуля может быть любое количество выходов (по ветке *False* – всегда один выход).

Применение: разделение дел на срочные дела и несрочные; перенаправление недоделанных или сделанных неправильно работ на доработку.

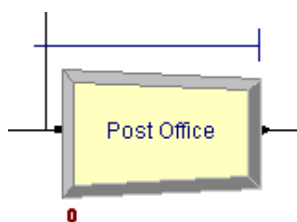
Таблица 3.5

Параметры модуля Decide

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Type	Тип принятия решения: <i>By Chance</i> – выбор направления основывается на вероятности и <i>By Condition</i> – проверка на выполнение конкретно заданного условия
Percent True	Значение, определяющее процент сущностей, который пойдет по направлению <i>True</i>
If	Тип условия, которое будет проверяться на выполнение
Named	Имя переменной, атрибута или типа сущности, который будет проверяться при входе сущности в модуль

Is	Математический знак условия, например больше, меньше, равно и т. д.
Value	Значение, с которым будет сравниваться атрибут или переменная пришедшей сущности. Если тип условия – Expression, то в выражении должен стоять знак условия, например Color<> Red

Модуль Batch



Этот модуль отвечает за механизм группировки сущностей в имитационной модели. Группировка может быть постоянной или временной. Временно сгруппированные комплекты сущностей позднее могут быть разъединены с помощью модуля Separate. Комплекты могут состоять из любого числа входящих сущностей, определенного пользователем, или же сущности могут объединяться в комплект в зависимости от атрибута сущности. Временные и стоимостные характеристики выходящей сущности, представляющей комплект, будут равны сумме характеристик вошедших в группу сущностей.

Сущности прибывают в модуль, становятся в очередь и остаются там до тех пор, пока в модуле не будет набрано заданное количество сущностей. Когда соберется нужное число сущностей, создается сущность, представляющая комплект.

Применение: собрать необходимое количество данных, прежде чем начинать их обработку; собрать ранее разделенные копии одной формы; соединить пациента и его больничную карту приема к врачу.

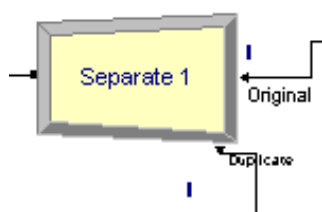
Таблица 3.6

Параметры модуля Batch

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Type	Способ группировки сущностей может быть: <i>Temporary</i> (временная) и <i>Permanent</i> (постоянная)
Batch Size	Число сущностей, образующих один комплект

Rule	Определяет, по какому признаку будут группироваться. Если Rule = Any Entity, – это значит, что первые 3 (если Batch Size = 3) сущности будут сгруппированы. Если Rule = By Attribute, то будет объединяться заданное количество сущностей с определенным атрибутом. Например, если Attribute Name = Color, то все сущности, имеющие одинаковое значение атрибута Color, будут сгруппированы
Attribute Name	Имя атрибута, по значению которого будут группироваться сущности

Модуль Separate



Этот модуль может использоваться в двух возможных вариантах:

1. Для создания копий входящих сущностей. Если модуль создает копии сущностей, то пользователь может задать количество дубликатов сущности. У дублированной сущности значения атрибута, а также анимационная картинка такие же, как и оригинала. Оригинальная сущность также покидает модуль.

2. Для разделения ранее сгруппированных сущностей. Правило для разделения стоимостных и временных характеристик копий сущностей и разделенных сущностей определяется пользователем. Когда временно сгруппированные сущности прибывают в модуль, они раскладываются на составные сущности. Сущности покидают модуль в той же последовательности, в которой они добавлялись в комплект.

Применение: разъединение ранее сгруппированных комплектов документов; для параллельной обработки счетов и документов по одному заказу.

Таблица 3.7

Параметры модуля Separate

Параметры	Описание
Name	Уникальное имя модуля
# of Duplic	Количество создаваемых копий входящей сущности
Type	Способ разделение входящей в модуль сущности. <i>Duplicate Original</i> – просто делает дубликаты входящей сущности. <i>Split Existing Batch</i> проводит разгруппировку

Allocation Rule	<p>Метод разделения стоимости и времени, если выбран Type=Split Existing Batch. Retain Original Entity Values сохраняет оригинальные значения сущностей.</p> <p>Take All Representative Values – все сущности принимают одинаковое значение.</p> <p>Take Specific Representative Values – сущности принимают специфическое значение</p>
-----------------	---

Модуль Assign



Этот модуль предназначен для задания нового значения переменной, атрибуту сущности, типу сущности, анимационной картинке сущности или другой переменной в системе.

В одном модуле можно сделать только любое количество назначений: сменить тип сущности, ее картинку, задать любое количество переменных и т. д.

Пример применения модуля Assign: установление приоритета для клиентов; присвоение номера вышедшему приказу.

Таблица 3.8

Параметры модуля Assign

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Type	Тип назначения, которое будет осуществляться. Other может включать в себя встроенные переменные, такие, как вместимость ресурса или конечное время моделирования
Variable Name	Имя переменной, которая будет изменяться в этом модуле
Attribute Name	Имя атрибута, который будет изменяться в этом модуле
Entity Type	Новый тип сущности, присваиваемый сущности в этом модуле
Entity Picture	Новая анимационная картинка для сущности, прошедшей этот модуль
New Value	Присваиваемое новое значение для атрибута, переменной

Модуль Record



Этот модуль предназначен для сбора статистики в имитационной модели. Модуль может собирать различные типы статистики, включая время между выходами сущностей из модуля, статистику сущности (время цикла, стоимость), статистику за период времени (период времени от заданной точки до текущего момента). Также доступен количественный тип статистики.

Частое применение модуля: подсчитать, какое количество заказов было выполнено с опозданием; подсчитать количество работы, совершаемое за один час.

Таблица 3.9

Параметры Модуль Record

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Type	Определяет тип статистики, которая будет собираться. <i>Count</i> будет увеличивать или уменьшать статистику на заданное значение. <i>Entity Statistics</i> будет собирать общую статистику о сущности, например: время цикла, стоимостные характеристики и т. д. <i>Time Interval</i> будет считать разницу между значением атрибута и текущим временем моделирования. <i>Time Between</i> будет отслеживать время между вхождением сущностей в модуль. <i>Expression</i> будет просто фиксировать значение, определяемое выражением
Attribute Name	Имя атрибута, значение которого будет использоваться для интервальной статистики
Value	Значение, которое будет добавляться к статистике, когда в модуль будет прибывать сущность

Модуль Dispose



Этот модуль является выходной точкой из имитационной модели. Статистика о сущности может собираться до того момента, пока она не выйдет из системы.

Применение: окончание бизнес-процесса; клиенты покидают отдел.

Таблица 3.10

Параметры модуля Dispose

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Record Entity Statistics	Определяет, будет ли вестись статистика о выходе сущности из системы

3.3.2. Модули данных

Все модули данных в навигаторе панелей имеют одинаковый вид, т. к. они не отображаются физически в блок-схеме модели, в связи с этим их изображение не приводится. Также мы не будем рассматривать стоимостные параметры модулей, т. к. они не влияют на логику модели.

Модуль Entity

Этот модуль определяет тип сущности и ее анимационную картинку в имитационном процессе, также определяет стоимостную информацию. Для каждого источника должен быть определен тип сущности, который он генерирует.

Применение модуля Entity: документы (факсы, письма, отчеты и т. д.); люди в моделях больницы или магазина.

Таблица 3.11

Параметры модуля Entity

Параметры	Описание
Entity Type	Название типа сущности
Initial Picture	Графическое представление сущности в начале имитационного процесса. Это значение может быть впоследствии изменено с помощью модуля Assign. Просмотреть анимационные картинки можно так: Edit/ Entity picture

Модуль Queue

Этот модуль данных предназначен для изменения правила расстановки сущностей в очереди, т. е. задается правило обслуживания сущности в процессе. По умолчанию тип очереди First in First out.

Применение: стопка документов, ожидающих освобождения ресурса; место для собирания частей, ожидающих упаковки (группировки).

Таблица 3.12

Параметры модуля Queue

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Attribute Name	Имя атрибута, значение которого будет учитываться, если тип = Lowest Attribute Value или Highest Attribute Value
Type	Правило расстановки сущностей в очереди: <i>First in First out</i> – первый вошел, первый вышел; <i>Last in first out</i> – последний пришел, первый вышел; <i>Lowest Attribute Value</i> – первый выйдет из очереди тот, значение атрибута у которого низшее; <i>Highest Attribute Value</i> – первый выйдет из очереди тот, значение атрибута у которого наивысшее

Более подробно хотелось бы остановиться на параметре Type, т. к. именно с помощью него можно определить, что такое «приоритет очереди» и как его необходимо задавать. Рассмотрим несколько измененный наш пример.

На прием к доктору приходят пациенты двух типов: взрослые и дети. Доктор (наш ресурс) – один. Он ведет прием и детей, и взрослых, причем время приема одинаково (около 30 минут), но у детей приоритет при обслуживании выше, чем у взрослых.

Каким образом мы это можем реализовать? Во-первых, в модуле Process задается ресурс «доктор»; с помощью параметра Action, который устанавливаем Seize Delay Release для назначения ресурса. Таким образом, когда сущность «пациент» зайдет в модуль процесс, то она захватит ресурс «доктор» на определенное время (около 30 минут). Во-вторых, у нас по условию время обслуживания пациентов одинаковое, таким образом, мы процесс обслуживания пациентов доктором смоделируем в виде одного блока Process, с временной задержкой в 30 минут. Но здесь возникает вопрос: каким образом задать приоритет? В данном случае, мы рассматриваем ситуацию, когда ресурс задан в одном блоке, т. е. нет смысла менять параметр Priority модуля Process. В этом случае, возникает ситуация, когда приоритет не ресурса, а приоритет очереди. И задается он в модуле Queue. Необходимо выбрать, у какого типа сущности он выше. Это

производится с помощью параметра Type: Lowest Attribute Value – первый выйдет из очереди тот, значение атрибута у которого низшее или Highest Attribute Value – первый выйдет из очереди тот, значение атрибута у которого наивысшее. Таким образом, когда у нас будут приходить сущности «дети», они будут иметь наивысший приоритет в обслуживании.

Модуль Resource

Этот модуль предназначен для определения ресурсов и их свойств в имитационном процессе; кроме того, модуль включает в себя стоимостную информацию о ресурсах и вместимость ресурсов. Ресурсы могут иметь фиксированную вместимость или же основанную на расписании. У ресурсов с фиксированной вместимостью в течение имитационного процесса вместимость изменяться не может. Ресурс должен быть связан с каким-либо процессом.

Применение: люди (клерки, продавцы, бухгалтеры, рабочие и т. д.); оборудование (телефонная линия, станок, компьютер).

Таблица 3.13

Параметры модуля Resource

Параметры	Описание
Name	Имя ресурса
Type	Метод, определяющий вместимость ресурса. <i>Fixed Capacity</i> – фиксированная вместимость ресурса. <i>Based on Schedule</i> – вместимость ресурса определяется модулем Schedule
Capacity	Число ресурсов, находящихся в системе
Schedule Name	Имя Schedule модуля, который определяет вместимость ресурса, если Type = Based on Schedule
Busy / Hour	Почасовая стоимость обработки сущности ресурсом. Время учитывается только тогда, когда ресурс занят обработкой и прекращает учитываться, когда ресурс освобождается
Idle / Hour	Стоимость ресурса, когда он не занят
Per Use	Стоимость обработки ресурсом одной сущности (не зависит от времени)

Модуль Schedule

Этот модуль может использоваться вместе с модулем Resource для определения вместимости ресурса и с модулем Create – для задания расписания прибытия сущностей.

Применение: расписание работы персонала с перерывами на обед; значение покупателей, прибывающих в супермаркет.

Таблица 3.14

Параметры модуля Schedule

Параметры	Описание
Name	Название расписания
Type	Тип расписания, который может быть <i>Capacity</i> (расписание для ресурсов), <i>Arrival</i> (для модуля Create) или <i>Other</i> (разнообразные временные задержки или факторы)
Time Units	Масштаб оси времени в графике расписания

Модуль Set

Этот модуль данных, который описывает группу ресурсов, использующихся в модуле Process. В группе могут находиться несколько ресурсов. Модуль Set автоматически создает ресурсы, вместимость которых по умолчанию равна 1, и без всякой стоимостной информации. Следовательно, если для ресурсов, входящих в группу, не нужно стоимостной информации и вместимость более 1, то можно обойтись созданием только модуля Set.

Возможно применение модуля для организации работы группы работников, например по очереди.

Таблица 3.15

Параметры модуля Set

Параметры	Описание
Name	Название группы
Members	Перечисляет ресурсы, входящие в группу. Порядок перечисления ресурсов важен, когда в модуле Process используется правило выбора Cyclical или Preferred Order
Resource Name	Названия ресурсов, входящих в группу

Модуль Variable

Этот модуль данных определяет значение переменных. Переменные, относящиеся к модулю Decide или Assign, могут использоваться в выражениях. Если переменная не описана в этом модуле, то ее первоначальное значение равно 0.

Применение: число документов обрабатываемых в час; присвоение серийного номера для идентификации продукции.

Таблица 3.16

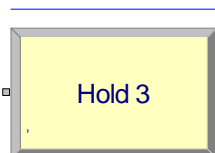
Параметры модуля Variable

Параметры	Описание
Name	Имя переменной
Initial Value	Первоначальное значение переменной. Это значение в последствии может меняться модулем Assign
Rows	Число строк в размерной переменной
Columns	Число столбцов в размерной переменной
Clear Option	Определяет время, когда значение переменной сбрасывается в начальное значение. <i>Statistics</i> – сбрасывает переменную в начальное значение в любой момент, когда статистика была расчищена. <i>System</i> – сбрасывает переменную в начальное значение в любой момент, когда система была расчищена. <i>None</i> – никогда не сбрасывает переменную в начальное значение, исключая предшествующую первой репликации
Statistics	Определяет, будет ли вестись статистика по этой переменной

3.5. Advanced Process Panel (панель усовершенствованных процессов)

3.5.1. Схемные модули

Модуль Hold



Модуль Hold удерживает (захватывает) сущности. Процесс удержания может продолжаться до бесконечности или до выполнения условия.

Применение модуля: складироваться детали; пассажиры ожидают транспорт на остановке.

Таблица 3.17

Параметры модуля Hold

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Type	Тип удержания сущности: <i>Infinite Hold</i> (удерживает до бесконечности, в этом случае у блока нет выхода), <i>Scan of Condition</i> (ожидает срабатывания определенного условия), <i>Wait of Signal</i> (ожидает сигнала, который вырабатывается только модулем Signal)

Если у модуля тип *Infinite Hold*, то забрать сущность из блока можно другими специальными модулями: *Remove*, *Signal* или *Pickup*. Соответственно, сущность выйдет по ветке именно из этих модулей, а не из *Hold*.

Поля *Queue Type* и *Queue name* присутствуют среди параметров модуля *Hold* всегда, задаются чаще всего автоматически (менять не рекомендуется).

Если тип имеет значение *Wait for signal*, то появляются поля *Wait for value* и *Limit* (ожидание конкретного значения сигнала и предел количества сущностей для освобождения из модуля *Hold*).

Если тип принимает значение *Scan of Condition*, то в этом случае становится доступным поле *Condition*, т. е. задержка напрямую зависит от выражения, заданного в этом поле.

Модуль Signal



Этот модуль посылает значение сигнала каждому модулю Hold в модели, в котором установлен тип Wait for signal , и освобождает заданное число сущностей.

Когда сущность прибывает в модуль Signal, то вырабатывается сигнал и посылается код сигнала в систему. В это время сущности в модуле Hold, который ожидает этого же сигнала, удаляются из очереди Hold и выходят из модуля.

Применение: прием преподавателем экзамена у определенного количества студентов; ожидание людьми определенного автобуса.

Таблица 3.18

Параметры модуля Signal

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Signal value	Значение посылаемого сигнала для модуля Hold
Limit	Число сущностей, которые будут освобождены из модуля Hold, когда сигнал будет получен

Модуль Pickup



Этот модуль предназначен для удаления определенного количества последовательно стоящих сущностей из определенной очереди. Сущности, которые удаляются из очереди, добавляются в конец сущности, вошедшей в блок Pickup. Чаще всего используется для удаления сущностей из модуля Hold, при условии, что тип Infinity Hold (без выхода). В модуле Pickup задается имя очереди, из которой будут забираться сущности, и определяется количество забираемых сущностей. Все сущности (вместе с исходной) выйдут из модуля Pickup в виде временной группировки.

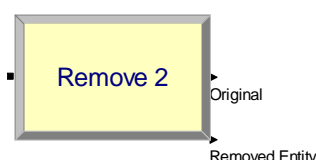
Применение: развоз товаров по магазинам со склада; посадка пассажиров в автобус на автобусной остановке.

Таблица 3.19

Параметры модуля Pickup

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Quantity	Количество сущностей, которые должны быть удалены из очереди
Queue Name	Имя очереди, из которой будут удаляться сущности
Starting Rank	Позиция сущностей в очереди, с которой начинается удаление

Модуль Remove



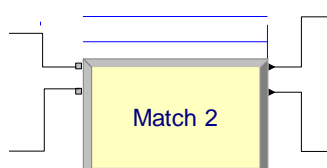
Модуль предназначен для удаления сущностей из любой очереди при условии, что эти сущности задерживаются бесконечно (Infinity). Отличие этого модуля от других заключается в том, что он может забрать только одну сущность из очереди. И у этого модуля 2 выхода: original и removed entity. По ветке original выходит та сущность, которая зашла (активировала) в этот модуль, а по ветке removed entity выходит та сущность, которая была забрана из очереди другого модуля (чаще всего модуля Hold).

Таблица 3.20

Параметры модуля Remove

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Queue name	Название очереди, из которой будет произведено удаление
Rank of entity	Глубина удаления (количество сущностей для удаления)

Модуль Match



Этот модуль предназначен для синхронизации движения двух или более сущностей, расположенных в различных, несвязанных очередях. Количество очередей может

варьироваться от 2 до 5. Сущность ждет в очереди до тех пор, пока в остальных очередях не появятся другие сущности либо с таким же значением атрибута, как и у исходной сущности.

Применение: сборка частей детали для дальнейшей обработки; собирание различных, но строго определенных продуктов по заказу клиента; синхронизация выхода покупателя с выходом заполненного заказа

Таблица 3.21

Параметры модуля Match

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Number to Match	Количество очередей для синхронизации сущностей
Type	Метод сравнения входящих сущностей для синхронизации. Значения: <i>Any Entities</i> – в каждой очереди должно быть по одной любой сущности, для того чтобы они вышли. <i>Based on Attribute</i> – в каждой очереди должна быть хотя бы одна сущность с таким же атрибутом для выхода
Attribute Name	Название атрибута, по которому сущности должны сравниваться. Используется, только если установлен тип Based on Attribute

Модуль Dropoff



Модуль Dropoff перемещает определенный набор сущностей из группы сущностей и посылает их в другой модуль, связанный с ним графическим соединением.

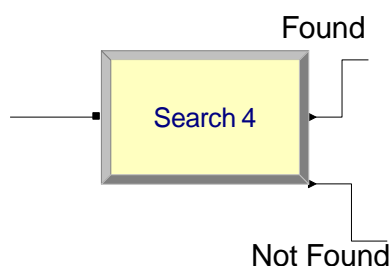
В этот модуль приходит временная группировка, из которой мы можем выделить требуемое количество сущностей, они пойдут по ветке Members, оставшаяся группа (в виде одной сущности) пойдет по ветке Original.

Таблица 3.22

Параметры модуля Dropoff

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Quantity	Число сущностей, которое будет выбрано из всех пришедших в группу сущностей
Starting Rank	Начальное значение выбрасываемой сущности
Member Attributes	Метод определения того, как назначить значение атрибута представленной сущности (такие как стоимость, время) для выброса оригинальных сущностей)
Attribute Name	Название атрибутов сущности, которые обозначены для выброса оригинальной сущности из группы

Модуль Search



Этот модуль необходим для поиска определенного элемента в очереди, в пакете, либо в каком-то выражении. Он имеет два выхода: True, если элемент найден, и False, если элемент не найден.

Применение: поиск среди коробок самой легкой.

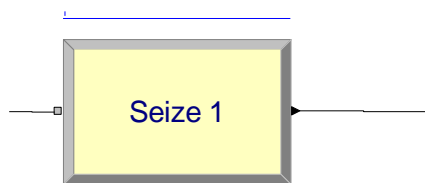
Таблица 3.23

Параметры модуля Search

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Type	Тип поиска: среди сущностей, объединенных в очередь, сущности, объединенные в пакет или поиск выражения
Queue Name	Имя очереди, в которой будет осуществляться поиск
Starting Value	Начальный класс в очереди или в пакете или начальное значение для переменной в выражении

Ending Value	Конечный класс в очереди или в пакете или конечное значение для переменной в выражении
Search condition	Условия, включающие индекс поиска выражений, или содержащие атрибут при поиске пакетов или сущностей в очереди

Модуль Seize



Модуль Seize предоставляет сущности один или несколько ресурсов. Он может быть использован для того, чтобы захватывать отдельный ресурс, ресурс из набора ресурсов или ресурс, определённый альтернативным методом, таким как атрибут или выражение.

Когда сущность поступает в этот модуль, она ждёт в очереди, пока определённые в этом модуле ресурсы не будут доступны. Также здесь определяется тип распределения ресурсов для поступивших сущностей.

Замечания

1. Сущности, которые захватываются с более высокой величиной приоритета, имеют более высокий приоритет, чем сущности, которые захватываются с более низкой величиной. Приоритетные выражения, оцененные как отрицательные величины, рассматриваются как нулевой приоритет. Если несколько сущностей с равными приоритетами пытаются захватить один и тот же ресурс, то его получает сущность с наибольшим временем ожидания.

2. Возможно определить набор состояний (State set) для ресурса и назначить состояние ресурса в определённых ситуациях, используя область состояния ресурса (Resource State Field). Затем можно собрать статистику:— сколько времени приходится на каждое состояние ресурса.

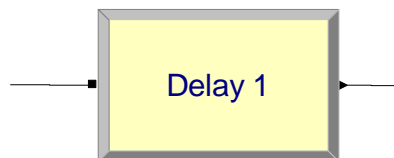
Таблица 3.24

Параметры модуля Seize

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Allocation	Определяет категорию, по которой будет распределена стоимость использования ресурса для сущности, проходящей через модуль Seize

Priority	Приоритет сущности, ожидающей в этом модуле ресурс. Определяется в случае, когда 1 или несколько сущностей из других модулей ожидают тот же ресурс (1 – высокий, 2 – средний, 3 – низкий, др.)
Type	Тип ресурса, который должен быть захвачен. Определяет конкретный ресурс или выбирает набор ресурсов. Имя ресурса также может быть определено атрибутом или выражением (Resource, Set, Attribute, Expression)
Resource name	Имя ресурса, который должен быть захвачен
Selection rule	Метод выбора среди доступных ресурсов в наборе

Модуль Delay



Модуль Delay задерживает сущность на определённое количество времени. По прибытии сущности в модуль выражение времени задержки оценивается и сущность остаётся в модуле на результирующее время. Затем время выделяется и, в зависимости от Allocation, либо добавляется к значению сущности, либо не добавляется, либо передаётся, либо ждет другое время. Также стоимости складываются, вычисляются и выделяются.

Таблица 3.25

Параметры модуля Delay

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Allocation	Тип категории, в которой сущности могут быть подвергнуты задержке времени и добавлению стоимости
Delay Time	Определяет значение задержки времени для сущности
Units	Указывает единицу измерения задержки времени

Модуль Release



Модуль Release используется для того, чтобы освобождать ресурсы, которые прежде были захвачены сущностью. Этот модуль может быть использован для того, чтобы освобождать индивидуальные ресурсы или ресурсы в пределах набора. Для каждого ресурса, который нужно освободить, определяется имя и количество. Когда сущность поступает в модуль, она теряет управление определённым ресурсом. Любые сущности, ожидающие в очередях этот ресурс, получают его немедленно.

Замечания:

1. Если есть сущность, ожидающая в очередях для захвата определённого ресурса, то, когда ресурс освобождается, он автоматически распределяется в ждущую сущность. Эта ждущая сущность будет обработана, как только сущность, которая освободила ресурс, переместится.

2. Системная переменная NR (имя ресурса) возвращает номер последнего занятого ресурса. Когда сущность поступает в модуль Release, NR уменьшается на количество освобождённых ресурсов, если ресурс не будет немедленно захвачен другой сущностью.

3. Если освобождается больше количество ресурсов, чем было ранее захвачено, то происходит ошибка.

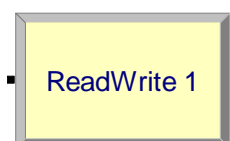
4. Освобождение множества ресурсов выполняется в порядке их появления в модуле Release.

Таблица 3.26

Параметры модуля Release

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Resources	Имя освобождаемых модулем ресурсов

Модуль ReadWrite



Модуль ReadWrite используется для чтения данных из входного файла или с клавиатуры и задания данных в список переменных или атрибутов (или других выражений). Этот модуль также используется, чтобы

записать данные на выходное устройство, например на экран или в файл.

Когда объект приходит в модуль ReadWrite, то файл изучается для того, чтобы увидеть, открыт он или нет. Если нет, файл автоматически открывается. Величины атрибутов, переменные или выражения будут прочитаны или записаны в зависимости от того, какой формат определен.

Таблица 3.27

Параметры модуля ReadWrite

Параметры	Описание
Name	Уникальный модульный идентификатор. Это имя отображается в модульной форме
Type	Метод использования (чтение или запись). Данные могут быть записаны в файл или на экран. Данные могут быть считаны из файла или с клавиатуры
Arena File Format	Имя файла, чтобы идентифицировать файл в пределах модуля File
Overriding File Format	Формат для записи или чтения данных. Этот формат аннулирует любой формат, определенный в структурной области модуля File. FORTRAN или C может использоваться, чтобы описать тип и позицию каждой области
Variable Type	Тип информации, что будет прочитана или записана
Attribute Name	Определяет символьное имя атрибута для записи или чтения
Variable Name	Определяет символьное имя переменной для записи или чтения
Other	Определяет выражение для чтения или записи других типов информации

3.5.2. Модули данных

Модуль Advanced Set

Этот модуль определяет наборы (очереди, хранилищ или другие наборы) с соответствующими его составляющими. Набор определяет группу схожих элементов, к которым можно обращаться через имя и индекс. К элементам, входящим в набор, можно обращаться как к членам этого набора.

Наборы очередей могут быть определены при помощи модуля Seize.

Таблица 3.28

Параметры модуля Advanced Set

Параметры	Описание
Name	Уникальный идентификатор
Set Type	Тип набора. Может быть Queue, Store, Other (другой)
Members	Задаются конкретные составляющие (очереди, хранилища), входящие в набор

Модуль Expression

Модуль Expression позволяет определять выражения и задавать им значения. К выражению обращаются при помощи имени. Выражения могут быть заданы как одномерный или двумерный массив.

Таблица 3.29

Параметры модуля Expression

Параметры	Описание
Name	Уникальное имя выражения
Row	Максимальное количество строк в определяемом выражении
Column	Максимальное количество столбцов в определяемом выражении. Данное свойство задается, только когда задано свойство Row
Expression Value	Значение, которое соответствует выражению

Этот модуль необходим для того, чтобы задавать какие-то часто используемые выражения, чтобы разгрузить модель, например в модулях Decide, Hold, Pickup.

Модуль Statistic

Модуль Statistic используется для того, чтобы определить дополнительную статистику, которая должна собираться в течение времени моделирования, а также чтобы определить файлы выходных данных.

Таблица 3.30

Параметры модуля Statistic

Параметры	Описание
Name	Уникальное имя модуля
Type	Тип статистики. Тип может быть time-persistent, tallies (observational data), count-based, outputs, and frequency-based

В зависимости от выбранного типа статистики появляются дополнительные поля.

1. Если выбран тип Tally: Tally Name – определяется символьное имя для типа статистики Tally, Tally Output File – имя выходного файла.

2. Если выбран тип Counter: Counter Name – определяется символьное имя для типа статистики Counter; Limit определяет лимит счетчика; Counter Output File – имя выходного файла.

Модуль Storage

Модуль Storage определяет имя Хранилища. Хранилище автоматически создается любым модулем, который на него ссылается.

Модуль File

Модуль File должен быть включен всякий раз, когда обращаются к внешнему файлу, используя ReadWrite модуль. Этот модуль выделяет системный файл, называет и определяет метод доступа, форматирование и эксплуатационные характеристики файла.

Таблица 3.31

Параметры модуля File

Параметры	Описание
Operating System File Name	Операционное системное имя, путь к файлу, откуда читаем или записываем. Символьная строка
Structure	Тип файловой структуры. Неформатированный, свободный формат, WorksSheet, специфические C-или FORTRAN-форматы
End of File Action	Тип действия, которое произойдет, когда будет достигнут конец файла. Ошибка, выход, на начало, игнорировать
Comment Character	Символ, указывающий отображение комментирующей записи. Одиночный символ

Модуль StateSet

Модуль используется для того, чтобы определить состояние ресурса или набора ресурсов. Состояния могут быть связаны с автосостоянием или могут быть заданы новые состояния для ресурса. Модуль Resource в базовой панели Process ссылается на StateSet, который данный ресурс будет использовать.

Таблица 3.32

Параметры модуля StateSet

Параметры	Описание
StateSet Name	Название набора состояний, которые могут быть назначены ресурсу в течение модельного времени
State Name	Имя пользователя определившего состояние
Auto State or Failure	Используется, чтобы связать State Name с автосостоянием или с заданным пользователем, именем отказа

Модуль Failure

Модуль Failure разработан для использования с ресурсами, а именно для имитации отказов ресурса. Может использоваться для ресурсов с однократной способностью или для ресурсов многократной способности, когда индивидуальные единицы ресурса заняты в одно и то же время.

Таблица 3.33

Параметры модуля Failure

Параметры	Описание
Name-	Имя отказа
Count	Определяет число ресурсов, реализуемых для отказов
Time	Определяет время для отказов
Up Time	Определяет время между отказами (число)
Up Time Units	Задаем формат времени (секунда, минута, час, день)
Down Time	Определяем продолжительность отказа (число)
Down Time Units	Задаем формат времени (секунда, минута, час, день)

3.6. Advanced Transfer Panel (панель перемещения)

3.6.1. Схемные модули

Модуль Station



Модуль Station определяет станцию или набор станций для физической или логической обработки, некая логическая («отправная») точка в модели.

Таблица 3.34

Параметры модуля Station

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Station Type	Тип станции
Station Name	Имя станции
Set Name	Уникальное имя набора станций
Save Attribute	Название атрибута, куда будут сохраняться значения атрибутов сущностей
Station Set Members	Перечисляется набор станций

Модуль Route



Модуль Route позволяет принять указанную сущность на заданную станцию, при этом позволяет имитировать время, которое будет затрачено сущностью на дистанцию к заданной станции.

Таблица 3.35

Параметры модуля Route

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Route Time	Время прохода через этот модуль
Units	Единицы измерения времени задержки (день, час, минута, секунда)
Destination Type	Тип станции назначения, на которую должна прибыть сущность (Station, Sequential, Attribute, Expression)

Модуль PickStation



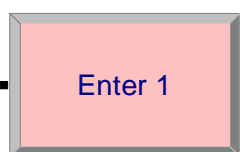
Модуль PickStation позволяет сущностям выбирать определенную станцию из множества существующих (маршрутизатор).

Таблица 3.36

Параметры модуля PickStation

Параметры	Описание
Name	Уникальное имя блока
Test Condition	Определяется тип выбора станции (минимум или максимум по полям): <i>Number In Queue</i> (количество в очереди); <i>Number En Route to Station</i> (количество маршрутизированных станций); <i>Number of Resources Busy</i> (количество занятых ресурсов) и <i>Expression</i> (выражение)
Route Time	Время в пути (до станции)
Units	Единицы измерения времени пути (день, час, минута, секунда)
Save Attribute	Имя атрибута, который хранит имя станции
Transfer Type	Определяет, каким образом сущности будут транспортироваться до следующей станции (Route, Transport, Convey or Connect)

Модуль Enter



Модуль Enter определяет станцию (или станции), соответствующую физическим или логическим позициям, где происходит обработка. Если модуль Enter определяет конкретную станцию, он эффективно определяет многочисленные обработки позиций.

Станция (или каждая станция в пределах решаемого комплекта) соотносится к области деятельности, которая используется, чтобы сообщить о времени и издержках, повышенных сущностями, на этих станциях. Эта сущность имени AreaTs также называется станцией.

Сущность может переместиться из предыдущего модуля в модуль Enter, причем двумя способами: отправление на станцию, связанную с модулем дистанционно или через реальное графическое соединение.

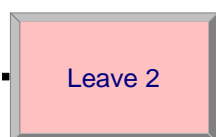
Когда сущность прибывает в модуль Enter, «разгружая», может произойти задержка и любое действие с передачей.

Таблица 3.37

Параметры модуля Enter

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Station Type	Определяет индивидуальную станцию или комплект станций, чтобы определить точку входа в этот модуль. Если выбран комплект (set), это указывает, что этот модуль входит в подмодель станции
Station Name	Имя станции активно в том случае, когда выбран тип Type Station
Parent Activity Area	Имя места отправления
Delay	Время задержки сущности по прибытии на данную станцию
Allocation	Тип категории, к которому будет добавляться время сущности и цена
Transfer In	Если выбран ресурс (транспортер или конвейер), чтобы доставить сущность к станции, используется для «отпускания», «освобождения», или «выхода»

Модуль Leave



Этот модуль используется для передачи сущности к станции или другому модулю.

Когда сущность прибывает в модуль Leave, она ожидает прибытия транспорта, когда прибывает транспорт, тратится время на загрузку и в конечном итоге сущность отправляется в пункт модуля назначения.

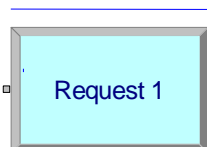
Таблица 3.38

Параметры модуля Leave

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Allocation	Тип категории, к которому будет добавляться время сущности
Delay	Время задержки сущности по прибытии на данную станцию
Unit	Величина задержки: день, час, минута, секунда
Transfer Out	Тип, содержащий запрос на транспорт

Далее будут подробно рассмотрены модули транспортера.

Модуль Request



Модуль Request вызывает (запрашивает) транспортер по прибытии в него сущности. Когда сущность достигает модуля Request, она размещается на транспортере, когда он доступен. Сущность остается в модуле Request, пока транспортер не достиг станции. Только тогда сущность перемещается из модуля Request для дальнейшего движения по модели.

Таблица 3.39

Параметры модуля Request

Параметры	Описание
Name	Уникальное имя модуля
Transporter Name	Название (имя) транспортера
Velocity	Скорость, с которой транспортер перемещает (единица длины в единицу времени). Единица времени определена в поле Units
Units	Определяет единицы времени для Velocity (т. е. в минуту, в час и т.д.)
Queue Type	Определяет тип очереди при загрузке транспортера
Queue Name	Эта область видима, только если тип очереди – очередь, и это определяет имя символа очереди

Модуль Activate

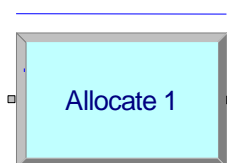


Модуль Activate активизирует или увеличивает вместимость предварительно приостановленного транспортера или транспортера, который был первоначально бездействующим (как определено в модуле Transporter).

Параметры модуля Activate

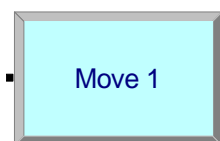
Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Transporter Name	Имя транспортера, с которым работает модуль
Unit Number	Определяет, насколько увеличится вместимость

Модуль Allocate



Модуль Allocate аналогичен модулю Request. Различие только в том, что модуль Allocate не позволяет задавать скорость и единицы измерения скорости транспортера.

Модуль Move



Модуль Move продвигает транспортер от одной станции к другой, которая является пунктом назначения. Контролируемая сущность ожидает в текущем модуле, пока транспортер прибудет в назначенный пункт. После этого сущность может перемещаться в другой модуль модели.

Время задержки перемещения транспортера из одного пункта (модуля Station) в другой основано на скорости транспортера, которая определяется в модуле Transporter, и расстоянии между пунктами, определенном в модуле Distance.

Сущность не может быть перемещена транспортером, если он не вызван с помощью модулей Request или Allocate. Сущность будет оставаться в модуле Move, пока транспортер не достигнет своего пункта назначения. Если определена скорость движения, это изменение временно и утилизируется только для определенного транспортера, который перемещается.

Таблица 3.41

Параметры модуля Move

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Transporter Name	Имя транспортера для перемещения
Unit Number	Определяет транспортер из множества транспортеров
Destination Type	Тип места назначения транспортера
Station Name	Имя места назначения (станции), в которое транспортер переместится
Velocity	Скорость, с которой транспортер переместится в пункт назначения, в единицах времени. Единицы времени определяются в поле Units
Units	Определяет единицы времени (секунды, минуты, часы, дни)

Модуль Transport

Модуль Transport по прибытии в него сущности запускает транспортер и перемещает его от одной станции к другой. Время задержки на перемещение и передачу сущности от одной станции к другой основывается на скорости транспортера и расстоянии между станциями.

Когда сущность входит в модуль Transport, то атрибут станции (Entity.Station) подставляется в станцию назначения, затем сущность передается в станцию назначения. Если станция назначения входит как Sequential, то следующая станция определяется посредством «Запроса сущности» и Jobstep с множеством (специально определенных атрибутов Entity.Sequence and Entity.Jobstep, respectively).

Модуль Transport является эквивалентом модуля Move, с той разницей, что Transport передает сущности дистанционно.

Таблица 3.42

Параметры модуля Transport

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Transporter Name	Определяет имя транспортера для передачи
Unit Number	Определяет, какой из транспортеров из множества транспортеров подлежит перемещению
Destination Type	Определяет тип места назначения сущности
Station Name	Определяет имя места назначения (станции), в которое сущность будет перемещаться
Velocity	Скорость, с которой транспортер перемещается к станции назначения
Units	Это поле определяет единицы измерения времени для скорости

Модуль Free



Модуль Free освобождает транспортер для дальнейшего его использования.

Таблица 3.43

Параметры модуля Free

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Transporter Name	Имя транспортера, который освободится

Модуль Halt



Модуль Halt изменяет состояние (статус) транспортера на неактивное. Если транспортер занят, в то время как сущность вошла в модуль Halt, то его статус определяется как занят и неактивен до тех пор, пока сущность, которая управляет транспортером, не освободится. Если во время вхождения сущности в модуль Halt транспортер является свободным, то статус транспортера изменяется на неактивный.

немедленно. Никакая сущность не может получить управление над остановленным транспортером, пока он снова не будет активизирован.

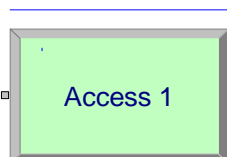
Таблица 3.44

Параметры модуля Halt

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Transporter Name	Имя транспортера, который требуется остановить
Unit Number	Определяет, какие из модулей транспортера из набора транспортера следует останавливать

Далее будут подробно рассмотрены модули конвейера.

Модуль Access



Этот модуль вызывает конвейер, распределяет ячейки конвейера для перемещения сущности от станции к станции. Получив контроль над ячейками конвейера, сущность может переместиться к другой станции конвейера. Этот модуль является эквивалентом модуля Request.

Таблица 3.45

Параметры модуля Access

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Cell	Количество ячеек, необходимых для перемещения конвейера
Conveyor Name	Имя конвейера-исполнителя
Queue Name	Имя очереди, в которую поступают сущности конвейера, если конвейер занят

Модуль Convey



Модуль Convey перемещает сущности по конвейеру от одной станции к другой. Время задержки сущности в пути определяется полем Velocity модуля Conveyor и расстоянием между станциями, определенным в модуле Segment. Этот модуль является эквивалентом модуля Transport.

Таблица 3.46

Параметры модуля Convey

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Cell	Количество ячеек, необходимых для перемещения конвейера
Conveyor Name	Имя конвейера, который будет использоваться
Destination Type	Определяет метод для определения пункта назначения сущности: <i>Station Name</i> – имя станции; <i>Attribute Name</i> – имя атрибута, который хранит имя станции; <i>Sequential</i> – следующая станция, которая определяется атрибутами сущности <i>Entity.Sequence</i> и <i>Entity.JobStep</i> , и <i>Expression</i> – выражение, которое определяет станцию

Модуль Start



Модуль Start изменяет статус конвейера от бездействующего до активного, т. е. активизирует (вызывает) конвейер. Конвейер может быть остановлен или от модуля Stop, или окончания создания сущности в начале моделирования. Скорость конвейера может изменяться постоянно после начала работы конвейера. Является эквивалентом модуля Move.

Таблица 3.47

Параметры модуля Start

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Conveyor Name	Имя конвейера, который требуется активировать
Velocity	Скорость, с которой конвейер переместится в пункт назначения, в единицах времени. Единицы времени определяются в поле Units
Units	Определяет единицы времени (секунды, минуты, часы, дни)

Модуль Stop



Модуль Stop устанавливает действующий статус конвейера в неактивный. Конвейер может быть активирован для любого модуля Start или по причине

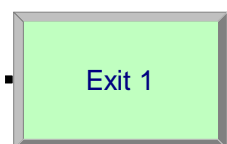
активации в начале моделирования. Когда сущность входит в модуль Stop, конвейер мгновенно останавливается, принимая во внимание тип конвейера или номер сущности, вошедшей в конвейер. Является эквивалентом модуля Halt для транспортера.

Таблица 3.48

Параметры модуля Stop

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Conveyor Name	Имя конвейера для остановки

Модуль Exit



Модуль Exit выпускает сущности из определенного конвейера и освобождает его для дальнейшей перевозки сущностей. Является эквивалентом модуля Free транспортера.

Таблица 3.49

Параметры модуля Exit

Параметры	Описание
Name	Уникальное имя модуля, которое будет отражено в блок-схеме
Conveyor Name	Имя конвейера, который освободится
# of Cells	Число последовательных сущностей для выпуска

3.6.2. Модули данных

Модуль Transporter

Модуль Transporter предназначен для определения транспортера в модели. Чаще всего модуль связан со схемным модулем Request, который вызывает транспортер, и модулем Move, который передвигает транспортер по схеме.

Таблица 3.50

Параметры модуля Transporter

Параметры	Описание
Name	Уникальное имя транспорта
Capacity	Количество транспортеров в наборе
Distance set	Определяет имя дистанции (пути), по которому будет двигаться транспортер
Velocity	Определяет начальную скорость транспорта
Units	Единицы измерения скорости
Initial Position	Определяет начальную станцию, с которой транспортер начнет свое движение

Модуль Distance

Модуль Distance предназначен для определения пути, по которому будет двигаться транспортер.

Таблица 3.51

Параметры модуля Distance

Параметры	Описание
Name	Уникальное имя дистанции
Beginning Station	Начальная станция дистанции
Ending Station	Конечная станция дистанции
Distance	Длина дистанции

Модуль Sequence

Этот модуль используется для определения последовательности для сущностей в модели при их движении. Последовательность состоит из списка станций, которые сущность должна посетить.

Таблица 3.52

Параметры модуля Sequence

Параметры	Описание
Name	Название последовательности
Station Name	Название станции
Step Name	Название станции, которая может быть в последовательности
Next Step	Шаг последовательности

Модуль Conveyor

Модуль Conveyor позволяет перемещать сущности между станциями, является аналогом модуля Transporter.

Таблица 3.53

Параметры модуля Conveyor

Параметры	Описание
Name	Название конвейера
Segment Name	Имя сегмента, по которому будет двигаться конвейер
Type	Существует 2 типа конвейера: накапливающий и не накапливающий
Velocity	Определяет начальную скорость транспортера
Units	Единицы измерения скорости

Модуль Segment

Модуль Segment определяет путь, по которому будет двигаться конвейер.

Таблица 3.54

Параметры модуля Segment

Параметры	Описание
Name	Имя сегмента
Beginning Station	Начальная станция
Next Station	Следующая станция в сегменте (может задаваться набором)
Length	Расстояние до предыдущей станции

3.7. Панель отчетов

С помощью панели отчетов можно просмотреть результаты имитации. На панели отчетов представлены несколько видов отчетов: Отчет «Краткий обзор категорий» и отчеты по четырем категориям, такие, как Сущности, Процессы, Очереди и Ресурсы.

1. *Отчет Category Overview категорий (Краткий обзор категорий)*

Отчет Category Overview отражает итоговую информацию о сущностях, процессах, очередях и ресурсах. Также показывает информацию о заданных пользователем переменных и информацию, собранную модулем Record.

2. Отчет о сущностях

Отчет о сущностях разделен на несколько частей.

2.1. Cycle Time: в этой части отчета показано среднее, максимальное и минимальное время существования сущности. Время существования сущности считается с момента её прибытия в систему и до того момента, когда сущность попадает в модуль Dispose. Ниже представляется гистограмма среднего времени цикла для каждого типа сущности.

2.2. NVA Cost: в этой части показано среднее, максимальное и минимальное значение недобавочной стоимости сущностей по каждому типу. Недобавочная стоимость рассчитывается на основании значения NVA Time.

2.3. Total Cost: в этой части показано среднее, максимальное и минимальное значение общей стоимости сущностей по каждому типу. Общая стоимость вычисляется путем сложения стоимости ожидания, добавочной стоимости и недобавочной стоимости для каждой сущности.

2.4. VA Cost: в этой части показано среднее, максимальное и минимальное значение добавочной стоимости сущностей по каждому типу. Добавочная стоимость рассчитывается на основании VA Time.

2.5. Wait Cost: в этой части показано среднее, максимальное и минимальное значение стоимости ожидания сущностей по каждому типу. Стоимость ожидания подсчитывается, исходя из времени ожидания, стоимости ресурса и стоимости нахождения сущности в системе.

2.6. Wait Time: в этой части показано среднее, максимальное и минимальное значение времени ожидания сущностей по каждому типу. Время ожидания – это период времени с момента поступления сущности в очередь (либо в модуле Process ожидает ресурс, либо в модуле Batch ожидает группировки) и до момента выхода из нее (начнет обрабатываться либо будет сгруппирована).

2.7. WIP (Work In Process): в этой части показано среднее, максимальное и минимальное значение времени ожидания сущностей в процессах.

3. Отчет о процессах

Отчет о процессах разделен на такие же части, как и отчет по сущностям, только с уклоном на процессы.

4. *Отчет о ресурсах* содержит информацию о загрузенности и простое ресурсов.

5. *Отчет по очередям* содержит информацию о среднем, минимальном и максимальном времени нахождения сущности в очереди и максимальных, средних и минимальных очередях.

3.8. Панель навигации

С помощью панели навигации можно быстро передвигаться по различным уровням модели, быстро менять виды. Можно задать быстрые клавиши для изменения вида. Виды подмоделей создаются автоматически, но также возможно добавить новые виды с помощью команды Add View. Можно передвигаться не только по различным уровням модели, но также быстро получать нужный масштаб какой-либо части модели.

3.9. Построитель выражений

ПП Arena позволяет строить сложные выражения. Это достигается с помощью Expression Builder. Построитель выражений имеет внешний вид, показанный на рис. 3.4.

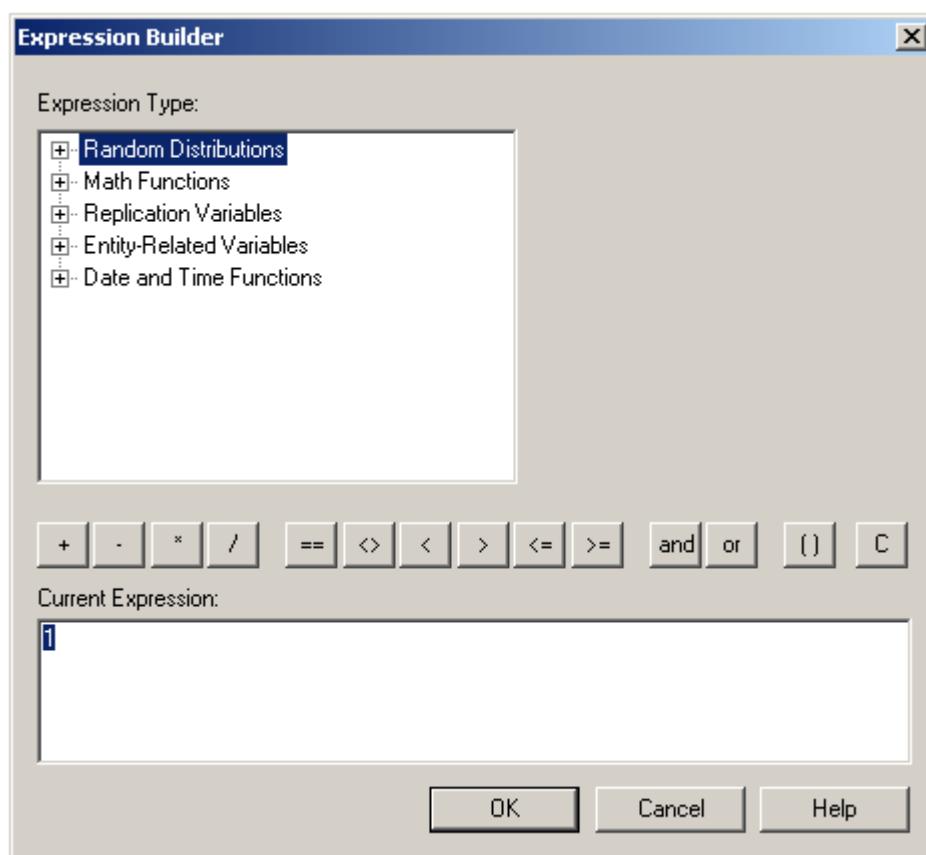


Рис. 3.4. Внешний вид построителя выражений

Построитель выражений имеет 3 секции:

1. Окно типов выражений. Рассмотрим более подробно *окно типов выражений*, которое состоит из четырех разделов:

1.1. Random Distributions (Вероятностные распределения).

В ПП Arena 7.0 заложены 13 типов стандартных распределений:

- normal (нормальное): *Mean, StdDev*;
- exponential (экспоненциальное): *Mean*;
- uniform (равномерное): *Min, Max*;
- poisson (пуассоновское): *Mean*;
- gamma (гамма): *Beta, Alpha*;
- beta (бета): *Beta, Alpha*;
- triangular (треугольное): *Min, Mode, Max*;
- continious (непрерывное): *CumP₁, Val₁, CumP_n, Val_n*;
- discrete (дискретное): *CumP₁, Val₁, CumP_n, Val_n*
- erlang (распределение Эрланга): *ExpoMean, k*;
- johnson (распределение Джонсона): *Gamma, Delta, Lambda, Xi*;
- lognormal (логнормальное): *LogMean, LogStd*;
- weibull (распределение Вейбулла): *Beta, Alpha*.

Остановимся более подробно на двух видах распределений, которые наиболее часто используются при моделировании сложных систем. Это равномерное (UNIF или Uniform) и треугольное распределения, приведенные на рис. 3.5, *а* и *б* соответственно. Равномерное распределение показывает, что вероятность возникновения события P_1 одинакова (равновероятна) на интервале от *Min* до *Max*, например клиенты приходят раз в 5-9 минут. Треугольное распределение показывает, что наиболее вероятно (*Most Likely*) появление события в какое-то определенное время, например клиенты приходят раз в 5-9 минут, но чаще всего раз в 7 минут.

1.2. Math Functions (Математические функции), к которым относятся 11 алгебраических операторов:

- абсолютное значение;
- округление до ближайшего целого;
- целая часть от нецелочисленного значения;
- минимальное значение;
- максимальное значение;
- натуральный логарифм;
- корень квадратный и т. д.

и 9 геометрических функций:

- синус;
- косинус;
- тангенс;

- арксинус и т.д.
- Replication Variables (Переменные, связанные с репликациями модели);
- Maximum Replications (максимальное количество повторений);
- Current Replication Number (текущее количество повторений).

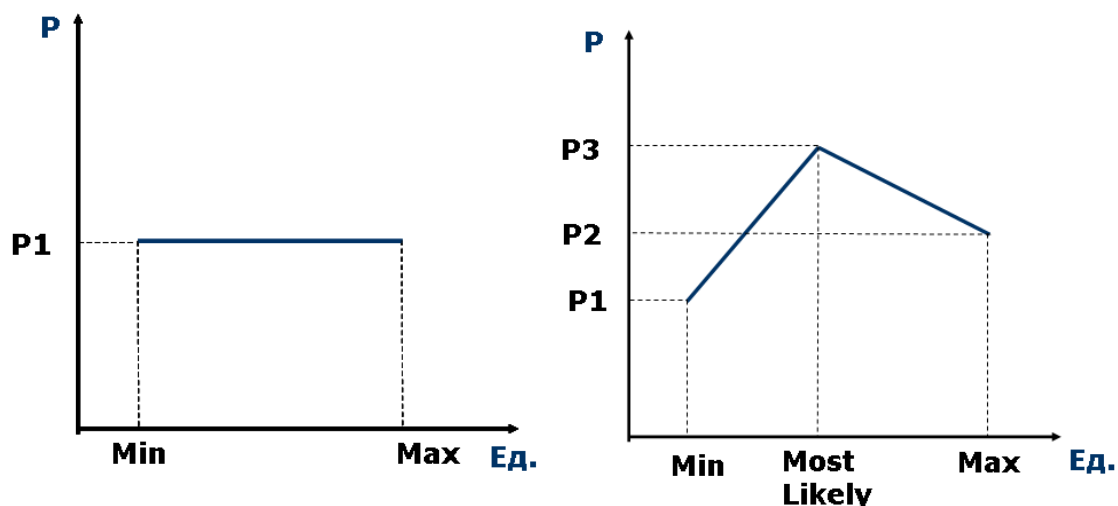


Рис. 3.5. Графики зависимостей распределений:
 а - равномерное распределение; б - треугольное распределение

1.3. Entity-Related Variables (переменные, связанные с сущностью):

- Attributes (Атрибуты). К наиболее интересным атрибутам следует отнести: Entity.Type (тип сущности), Entity.SerialNumber (серийный номер сущности), Entity.Picture (анимационная картинка сущности), Entity.CreateTime (Время создания сущности), User-Defined Attribute Value (атрибуты, заданные пользователем);
- Group Member Variables (Групповые переменные).

1.4. Date and Time Functions (временные функции).

Наиболее интересный и часто используемый оператор из этой группы – это TNOW (Current Simulation Time – текущее время моделирования).

2. Панель операторов, используемых в выражениях (сложение, вычитание, и т. д.; элементы сравнения, логические операторы и т. д.)

3. Окно записи выражения

3.10. Вопросы и задания к главе 3

1. Перечислите основные достоинства ПП имитационного моделирования Arena 7.0.
2. Какие основные панели используются в ПП Arena 7.0 для моделирования процессов и систем?
3. На какие типы подразделяются модули в строительных панелях?
4. Перечислите, с помощью каких модулей можно забрать (освободить) сущности из модуля Hold, если тип задержания в модуле Infinity Hold?
5. Для чего необходим модуль Match и в связке с каким модулем он обычно работает?
6. Приведите пример использования модуля Dropoff.
7. Какие параметры необходимо задать модулю Process, чтобы появилась очередь?
8. Поясните, каким образом можно смоделировать, чтобы модуль Process мог обрабатывать по 5 сущностей, а только шестая и последующие становились в очередь?
9. Объясните, в чем разница типов Split existing batch и Duplicate Original модуля Separate?
10. Что такое Resource и что значит его параметр Capacity?

Глава 4. Лабораторный практикум

4.1. Задания к лабораторным работам

Целью выполнения лабораторных работ является:

1. Знакомство с ПП Arena 7.0, изучение основных строительных панелей, модулей и свойств модулей.
2. Моделирование тестовых примеров согласно заданиям.
3. Подготовка студентов к выполнению курсовой работы.

Задание 1. Модель 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 интервалов 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.

Доходы от работы парикмахерской определяются количеством клиентов, обслуженных в течение рабочего дня (стоимость стрижки – 100 рублей, бритья – 20 рублей), убытки определяются временем простоев парикмахера (в отсутствие клиентов) и количеством необслуженных клиентов в очереди.

Моделирование проведите для рабочей недели (6 дней по 8 часов).

После разработки модели, согласно заданию, внесите в нее следующие дополнения и/или изменения:

1. Клиенты первого типа имеют анимационную картинку «Woman» (в виде женщины), а клиенты второго типа – «Man».
2. Задайте анимацию ресурсу «Парикмахер», когда он свободен (Idle), рис. 4.1, а, и когда он занят (Busy), рис. 4.2, б.
3. Измените правило обслуживания: приоритет в обслуживании имеют женщины (клиенты первого типа).
4. Рассмотрите возможность ввода в модель второго парикмахера. Как измениться доход парикмахерской?

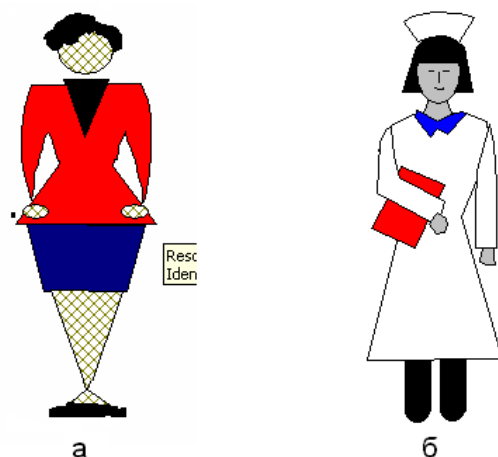


Рис. 4.1. Анимационная картинка ресурса «Парикмахер»:
а - ресурс свободен; *б* - ресурс занят

Задание 2. Работа сборочного цеха

В сборочный цех поступают детали трех видов. Детали первого типа (Д1) поступают 20 ± 3 мин (наиболее часто 20 мин). Детали второго типа (Д2) – 16 ± 5 мин. Детали третьего типа (Д3) – 20 мин. Как только сборщику поступают три детали (любые), он производит монтаж готового изделия за 5 мин. Из собранных изделий 15 % бракованные. Если изделие бракуется в первый раз, то оно поступает на повторный монтаж к сборщику. Если изделия бракуются 2 раза, то они идут в отходы (10 мин). Не бракованные изделия упаковываются по 5 штук за 3 минуты упаковщиком.

Смоделировать 8-часовой рабочий день.

Построить модель согласно заданию и выполнить следующие задания:

1. Определить каждому типу деталей свою анимационную картинку.
2. Определить анимационную картинку готовому изделию и упакованному изделию.
3. Задать анимационную картинку ресурсам «Сборщик» и «Упаковщик», когда они свободны и заняты.
4. Собрать статистику по бракованным изделиям (отходы и один раз бракованные), количеству упаковок, по загрузженности ресурсов «Сборщик» и «Упаковщик».
5. Изменить модель следующим образом: сборщик собирает изделие из деталей разного типа, и готовые не бракованные изделия

складируются. Один раз в 10 часов из гаража выезжает грузовик и забирает со склада все упаковки.

Задание 3. Работа системы сбора информации

Распределенный банк данных системы сбора информации организован на базе ЭВМ, соединенных дуплексным каналом связи. Поступающий запрос обрабатывается на первой ЭВМ, и с вероятностью 50 % необходимая информация обнаруживается на месте. В противном случае необходима посылка запроса во вторую ЭВМ.

Запросы поступают через 10 ± 3 с, первичная обработка запроса занимает 2 с, выдача ответа требует 18 ± 2 с, передача по каналу связи занимает 3 с. Временные характеристики второй ЭВМ аналогичны первой.

Смоделировать прохождение 400 запросов.

Определить необходимую емкость накопителей перед ЭВМ, обеспечивающую безотказную работу системы, и функцию распределения времени обслуживания заявки.

Построить модель согласно заданию и выполнить следующие задания:

1. В систему первоначально поступают сущности в виде дискет, а затем преобразовываются в самолеты и лодки.
2. Первые 200 запросов идут по ветке True, остальные – по False
3. Первые 30 мин. все запросы шли по True, остальные – по False
4. Первые 200 запросов проходили первичную обработку 2 с, остальные – 4 с.
5. Запросы моделируются с разными приоритетами: в модуле условия с большим приоритетом – по True, с меньшим – по False

Задание 4

В компанию поступают запросы (20 ± 4 мин.). Поступающий запрос обрабатывается двумя сотрудниками, причем первый сотрудник обрабатывает 75 % запросов, второй обрабатывает остальные запросы. Первичная обработка запроса занимает 23 минуты, выдача ответа требует 18 ± 5 мин., как у первого, так и у второго сотрудника.

Смоделировать прохождение 350 запросов.

Построить модель согласно заданию и выполнить следующие пункты:

1. Определить количество запросов, обработанных каждым сотрудником за 24 часа.
2. В систему первоначально поступают сущности в виде телефонных звонков, а затем к первому сотруднику приходят в виде отчетов, а ко второму сотруднику – в виде дискет.
3. Измените модель: первые 50 запросов идут к первому сотруднику на обработку, остальные – ко второму.
4. Измените модель: первые 4 часа все запросы идут ко второму сотруднику на обработку, остальные – к первому.
5. Измените модель: первые 150 запросов проходят первичную обработку 23 мин., остальные – 30 минут.
6. Измените модель: на обработку поступают 2 вида запросов (телефонные звонки и письма). Причем при первичной обработке у телефонных звонков приоритет выше, чем у писем.
7. Создайте анимационные картинки ресурсам, когда они свободны и заняты. Первый сотрудник должен быть изображен также, как на рис. 4.2 (*а* – свободен, *б* – занят). Второй сотрудник должен быть изображен также как на рис. 4.3 (*а* – свободен, *б* – занят).

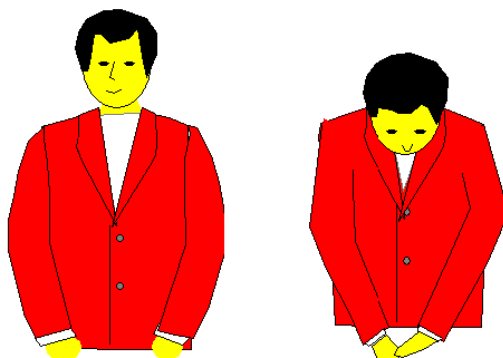


Рис. 4.2. Анимационная картинка ресурса «Сотрудник 1»:
а - ресурс свободен; *б* - ресурс занят

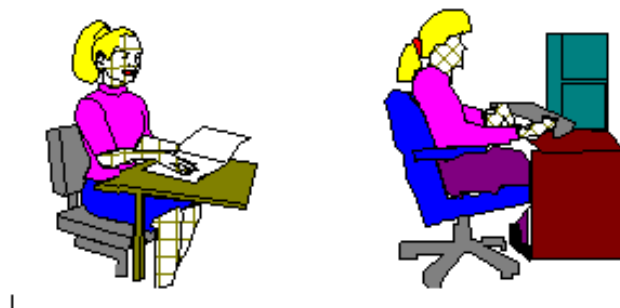


Рис. 4.3. Анимационная картинка ресурса «Сотрудник 2»:
а - ресурс свободен; *б* - ресурс занят

Задание 5

В слот-бар приходят клиенты. В игровой автомат, типа «однорукий бандит», каждые 5–10 минут опускается монета номиналом 5 рублей. Автомат случайным образом в течение 10 секунд выдает три цифры от 0 до 9.

В случае совпадения всех трех цифр, игрок выигрывает 50 монет (по 5 рублей), в случае выпадения любой другой комбинации – монета игрока уходит в доход казино. Принятые монеты автомат упаковывает в пачки по 10 штук в каждой.

В случае выигрыша игрок опускает в автомат дополнительно от 10 до 15 монет, на каждую монету он тратит 20 секунд.

Смоделировать работу автомата в течение 24 часов. Определить сумму денег, выигранных игроками, и чистую прибыль казино в рублях.

Задание 6

Создать модель полета рейсовых самолетов.

Клиенты, желающие приобрести билет на самолет, приходят в кассу аэропорта в среднем через 20 ± 5 , чаще 10 минут, причем 25 % из них приобретают билеты в первый класс, 70 % – во второй класс, а остальные вообще отказываются приобретать билеты и уходят.

Время вылета самолета определяется его полной загрузкой, т. е. самолет вылетит только при наличии 10 пассажиров первого класса и 20 пассажиров второго класса.

Самолеты прибывают в аэропорт в среднем раз в 6–12 часов, максимальное количество самолетов – 20.

Время полета занимает в среднем (5 ± 3) часов, чаще 6 часов. По прилету пассажиров отвозят в здание аэропорта, а самолет – на техническое обслуживание.

Задание 7

Участок ремонта кузовов автомобилей состоит из двух рабочих мест: первое рабочее место – это кузовной ремонт автомобиля, второе рабочее место – окраска кузова. После восстановления кузова автомобили поступают в окрасочную камеру.

Время поступления на ремонт поврежденных автомобилей первой модели – случайная величина, равномерно распределенная на интервале от 1 до 6 часов, второй модели – от 1 до 2 часов.

На кузовной ремонт автомобилей первой модели тратится от 1 до 3 часов, второй модели – от 2 до 5 часов.

Время окраски любого автомобиля равномерно распределено на интервале (15 – 20) минут.

Модели первого типа при обслуживании имеют более высокий приоритет.

В случае если ремонтная мастерская и покрасочная камера заняты, автомобили дожидаются обслуживания в очередях, длины которых не ограничена.

За 12 часов оценить отдельно для первой и второй модели:

- среднее время, которое тратится на ремонт автомобилей;
- среднее время ожидания в очередях;
- количество отремонтированных автомобилей;
- максимальный размер очереди «ожидания» начала

обслуживания и очереди перед операцией окраски.

Проанализировать зависимость приведенных выше характеристик при изменении их числовых значений.

Сделать анимацию в модели:

1. Модели первого типа – грузовики (Entity Picture = Track); модели второго типа – минивэны (Entity Picture = Van).

2. Первое рабочее место имеет анимационную картинку, приведенную на рис. 4.4, второе рабочее место на рис. 4.5.

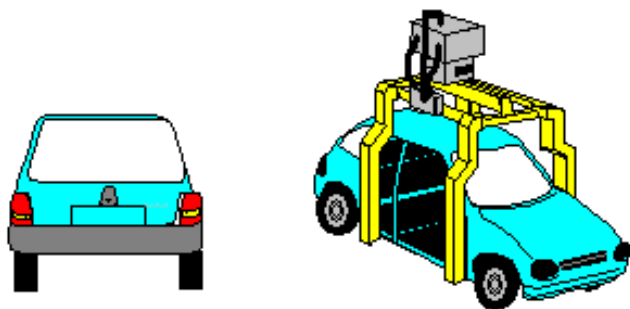


Рис. 4.4. Анимационная картинка ресурса «Первое рабочее место»:

а - ресурс свободен;

б - ресурс занят

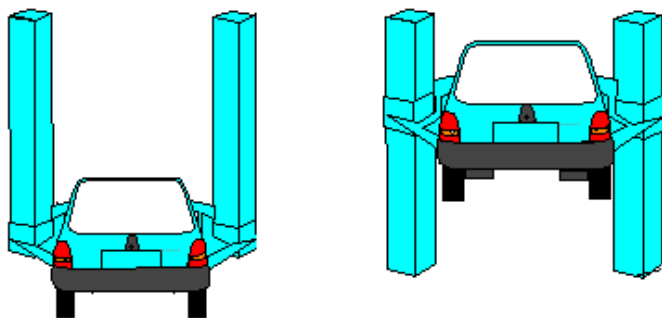


Рис. 4.5. Анимационная картинка ресурса «Второе рабочее место»:

а - ресурс свободен;

б - ресурс занят

Задание 8

В магазин за покупками приходят клиенты. Для работников магазина клиенты классифицируются на постоянных и обычных. Продавцы (менеджеры) затрачивают в среднем 2 минуты на человека для разъяснения информации по товарам и ответа на вопросы. Приоритетное право на обслуживание без очереди имеют постоянные клиенты; 25 % посетителей уходят без покупок, а остальные встают в очередь в кассу. Кассир один, он обслуживает из очереди постоянных клиентов, а потом обычных посетителей, причем как только кассир рассчитал одного клиента, он сразу же обслуживает следующего, время обслуживания клиента занимает 5 минут.

Постоянные клиенты в основном приходят утром (с 9 до 11 часов) и в конце рабочего дня (с 16 до 18 часов), а обычные посетители – в основном в середине дня.

Построить график, отображающий уровень посещаемости магазина покупателями, и график загруженности кассира.

Задание 9

Люди приносят на почту письма, которые могут быть двух видов: заказные и обычные. Затем почтовые работники их обрабатывают.

Заказные письма поступают круглосуточно раз в 5–20 минут, а простые письма принимаются только с 8.00 до 20.00 (8.00–12.00 их количество увеличивается от 50 до 120, наибольшее их количество (260) поступает между 12.00 и 14.00, а затем их количество плавно убывает от 180 до 70).

Оба вида писем обрабатываются одним работником почтовой службы, причем заказные письма обрабатываются вне очереди, т. к. они важнее. Время обработки заказных писем – 1–3 минуты, а время обработки простых писем – 3–5, чаще 4 минуты.

Затем все эти письма поступают в отдел подготовки к отправлению, где заказные письма обслуживаются также вне очереди. Время подготовки писем к отправке – 10–15 минут.

Создать анимацию работы сотрудников почты и отразить процесс обработки простых писем на гистограмме.

Задание 10

На участке термической обработки выполняются цементация и закаливание шестерен, поступающих через 10 ± 5 мин.

Цементация занимает 10 ± 7 мин., а закаливание - 10 ± 6 мин. Качество определяется суммарным временем обработки.

Шестерни со временем обработки:

- больше 25 мин. – покидают участок;
- от 20 до 25 мин. – передаются на повторную закалку;
- меньше 20 мин. – должны пройти повторную полную обработку.

Детали с суммарным временем обработки меньше 20 мин. считаются вторым сортом.

Смоделировать процесс обработки на участке 400 шестерен.

Определить:

- количество обработанных деталей;
- число повторений полной и частичной обработки.

В результате выполнения лабораторной работы (задание выдается преподавателем) студентом сдается **отчет**, который должен содержать:

1. Титульный лист согласно стандарту ТПУ (пример можно найти на сайте ТПУ <http://standard.tpu.ru/stdpredp/stp42i.doc>).
2. Цель работы и задание.
3. Схему модели и текстовое описание логики модели.
4. Описание каждого модуля (print screen каждого модуля и текстовое описание).
5. Эксперименты с моделью.
6. Выводы по работе.

4.2. Пример выполнения задания

Задание: Самолеты прибывают для посадки в район крупного аэропорта каждые 10 ± 5 мин. Если взлетно-посадочная полоса свободна, прибывший самолет получает разрешение на посадку. Если полоса занята, самолет выполняет полет по кругу и возвращается к аэропорту через каждые 4 мин. Если после пятого круга самолет не получает разрешения на посадку, он отправляется на запасной аэродром.

В аэропорту через каждые 10 ± 2 мин к взлетно-посадочной полосе выруливают готовые к взлету машины и получают разрешение на взлет, если полоса свободна. Для взлета и посадки самолеты занимают полосу ровно на 2 мин. Если при свободной полосе одновременно один самолет прибывает для посадки, а другой – для взлета, полоса предоставляется взлетающей машине.

Смоделировать работу аэропорта в течение суток. Подсчитать количество самолетов, которые взлетели, сели и были направлены на запасной аэродром. Определить коэффициент загрузки взлетно-посадочной полосы.

Рассмотрим подробно логику реализованной на рис. 4.6 модели.

1. Прибытие самолетов для взлета имитируется модулем Create «Take off». Этот модуль генерирует сущности Entity 1 в виде самолетов.

2. Главным условием взлета этих самолетов является то, что взлетно-посадочная полоса должна быть свободна. В нашей модели взлетно-посадочная полоса моделируется модулем Process 1, которому соответствует Resource 1. После того как появляется самолет, желающий взлететь, он попадает в модуль Hold 2, который выпустит этот самолет при условии, что полоса освободилась. Взлетевший самолет, т. е. обработанный модулем Process 1, уходит из системы через модуль Dispose 2.

3. Прибытие самолетов для посадки имитируется модулем Create «Landing». Этот модуль генерирует сущности Entity 2 в виде самолетов. Модуль Assign 2 задает значение Attribute 1, равное 1; это необходимо далее для подсчета кругов.

4. При посадке по заданию должны выполняться следующие условия: полоса должна быть свободна и не должно быть самолетов, идущих на взлет, т. к. у них приоритет выше.

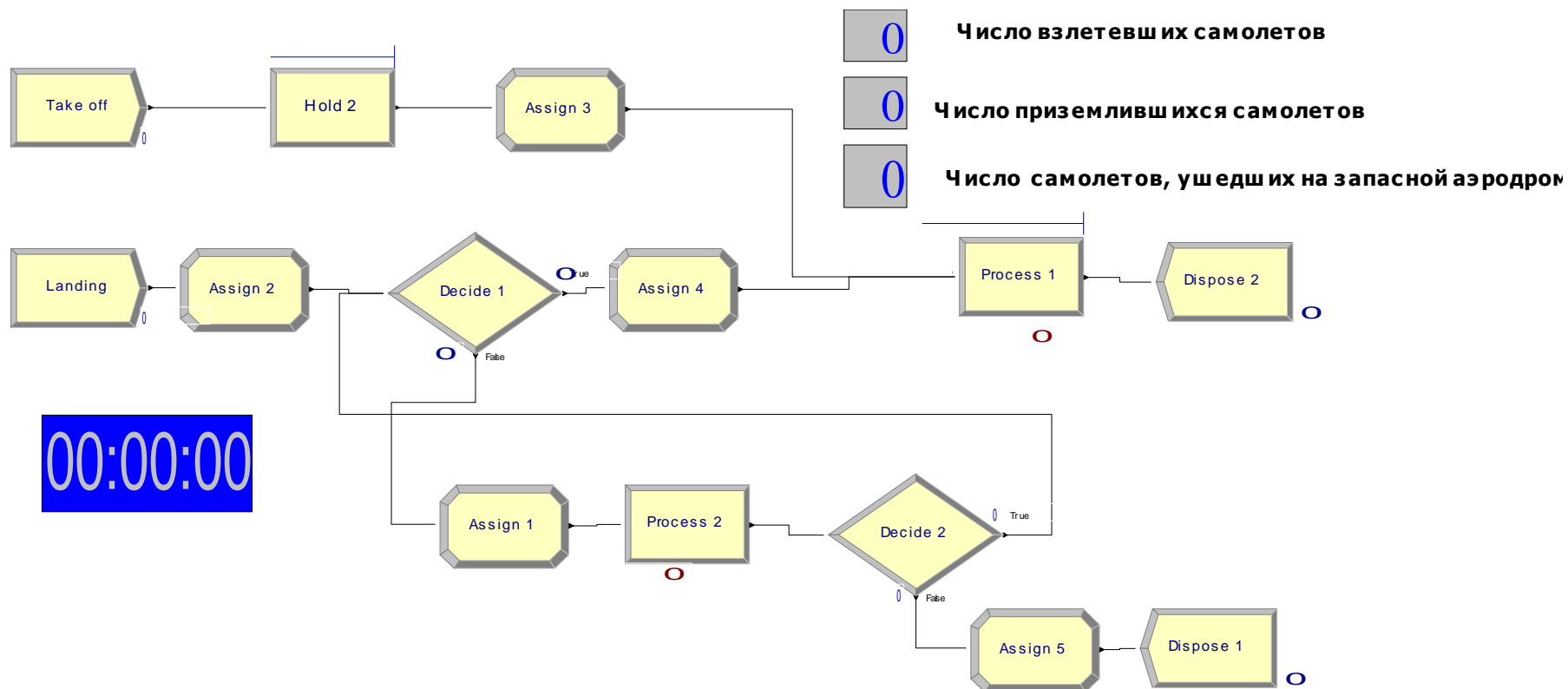


Рис. 4.6. Модель функционирования взлетно-посадочной полосы аэропорта в ПП Arena 7.0

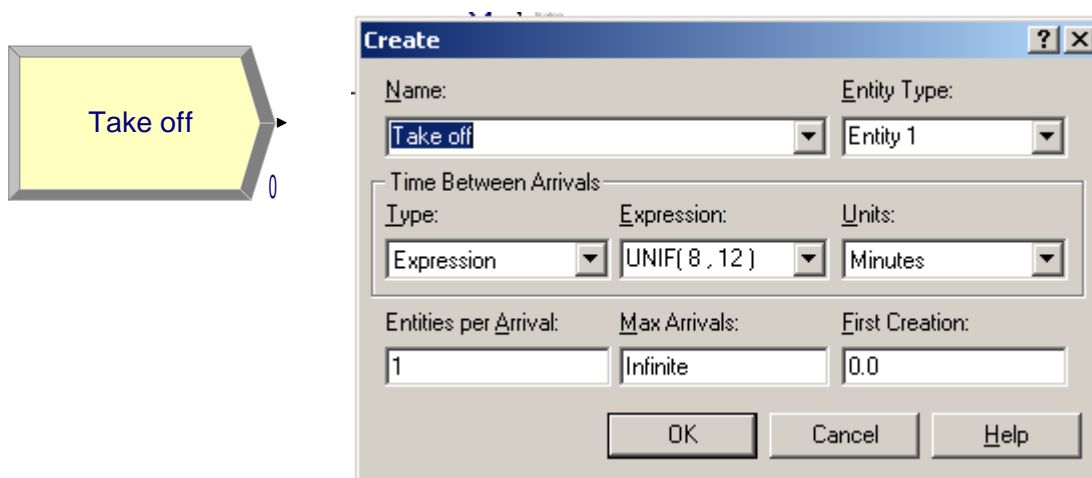
Это мы будем реализовывать через модуль Decide 1, в котором мы будем проверять занятость Recourse 1 в Process 1, и проверять очередь в Hold 2. Приземлившийся самолет, т. е. обработанный модулем Process 1, уходит из системы через модуль Dispose 2.

5. В Decide 2 будет проверяться следующее: если по прибытии самолета для посадки полоса (Recourse 1) будет занята и /или будут присутствовать самолеты на взлет в Hold 2, то этот самолет пойдет не по ветке True на полосу, а по ветке False.

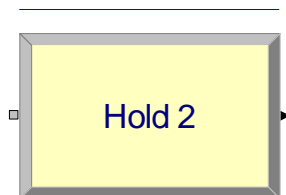
6. В ветке False первым стоит модуль Assign 1, который увеличивает Attribute 1 на единицу каждый раз, когда он проходит по этой ветке. Затем модуль Process 2 имитирует круг над аэропортом, после чего в модуле Decide 2 проверяется, сколько уже кругов сделал этот самолет, если меньше 5, то он опять возвращается к аэропорту для проверки условий, а если уже 5, то летит на запасной аэропорт.

7. Модули Assign 3, Assign 4 и Assign 5 необходимы для сбора статистики по взлетевшим, севшим самолетам и самолетам, ушедшим на запасной аэропорт.

Рассмотрим более подробно наиболее интересные модули.



В аэропорту через каждые 10 ± 2 мин к взлетно-посадочной полосе вырливают готовые к взлету машины.



Hold [?] [X]

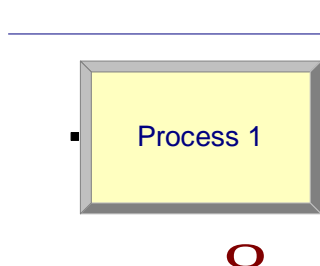
Name: Type:

Condition:

Queue Type:

Queue Name:

Готовые к взлету машины и получают разрешение на взлет, если полоса свободна: $STATE(Resource\ 1) == IDLE_RES$.



Process [?] [X]

Name: Type:

Logic:

Action: Priority:

Resources:

Resource, Resource 1, 1	<input type="button" value="Add..."/>
<End of list>	<input type="button" value="Edit..."/>
	<input type="button" value="Delete"/>

Delay Type: Units: Allocation:

Value:

☒ Report Statistics

Для взлета и посадки самолеты занимают полосу ровно на 2 мин., Process 1 имитирует взлетно-посадочную полосу.



Create

Name: Entity Type:

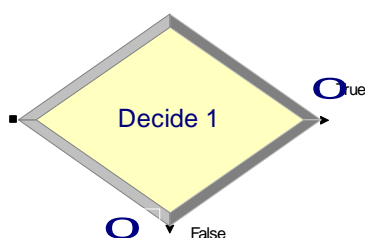
Time Between Arrivals:

Type: Expression: Units:

Entities per Arrival: Max Arrivals: First Creation:

OK Cancel Help

Самолеты прибывают для посадки в район крупного аэропорта каждые 10 ± 5 мин.



Decide

Name: Type:

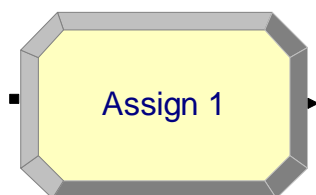
If:

Value:

OK Cancel Help

Если взлетно-посадочная полоса свободна, прибывший самолет получает разрешение на посадку и у них приоритет ниже, т. е. очередь в Hold 2 равна 0:

$STATE(Resource\ 1) == IDLE_RES \ \&\& \ NQ(Hold\ 2.Queue) == 0.$



Assign

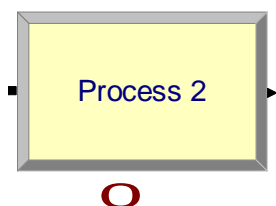
Name: Assign 1

Assignments:

- Attribute, Attribute 1, Attribute 1+1
- <End of list>

Buttons: Add..., Edit..., Delete, OK, Cancel, Help

Этот модуль увеличивает Attribute 1+1, который моделирует количество кругов.



Process

Name: Process 2 Type: Standard

Logic:

Action: Delay

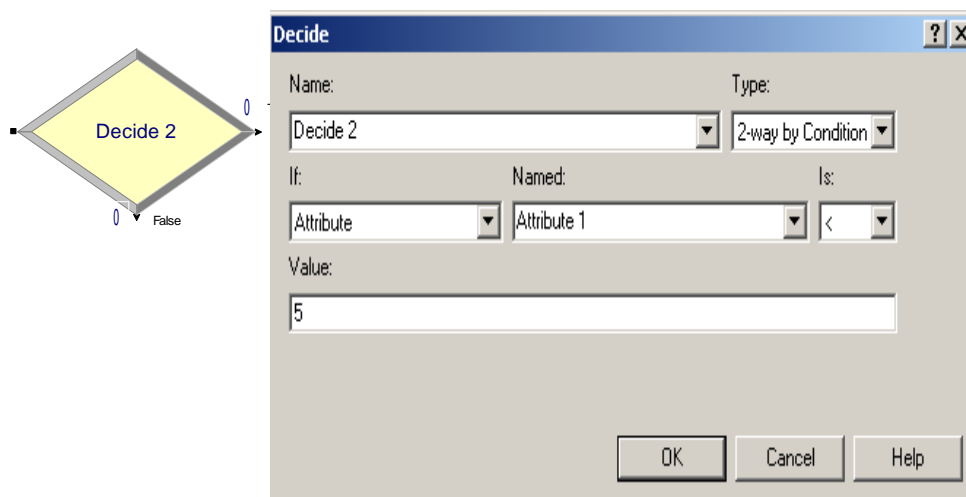
Delay Type: Constant Units: Minutes Allocation: Value Added

Value: 4

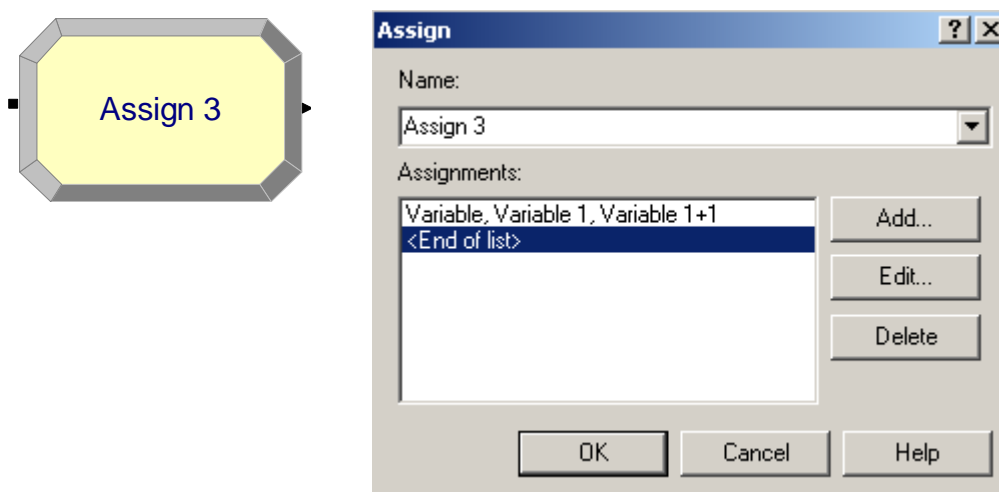
☒ Report Statistics

Buttons: OK, Cancel, Help

Если полоса занята, самолет выполняет полет по кругу и возвращается к аэропорту через каждые четыре минуты. Process 2 моделирует процесс совершения по кругу.



Этот модуль проверяет, сколько кругов сделал самолет: если 5, то он уходит на запасной аэропорт в Dispose 1.



Модули Assign 3, Assign 4 и Assign 5 аналогичны и необходимы для сбора статистики по взлетевшим, севшим самолетам и самолетам, ушедшим на запасной аэропорт:

- Variable 1 подсчитывает взлетевшие самолеты;
- Variable 2 подсчитывает севшие самолеты;
- Variable 3 подсчитывает самолеты, ушедшие на запасной аэродром.

Просмотреть значения переменных, полученных в результате моделирования, можно в стандартных отчетах, которые формируются в результате каждого прогона модели.

Time Persistent

Variable	Average	Half Width	Minimum	Maximum
Variable 1	72.3690	(Insufficient)	0	144.00
Variable 2	69.3066	(Insufficient)	0	141.00
Variable 3	0.9783	(Insufficient)	0	1.0000

Таким образом, из отчета видно, что значение переменных следующее:

- Variable 1 = 144;
- Variable 2 = 141;
- Variable 3 = 1.

Также в отчетах мы можем просмотреть загруженность полосы, которая у нас задана Resource 1.

Resource 1				
Usage	Average	Half Width	Minimum	Maximum
Instantaneous Utilization	0.3958	0,011194829	0	1.0000
Number Busy	0.3958	0,011194829	0	1.0000
Number Scheduled	1.0000	(Insufficient)	1.0000	1.0000
Scheduled Utilization	0.3958			
Total Number Seized	285.00			

Загруженность определяется параметром Number Busy, и в нашем случае равна 39,58 % от общего времени моделирования.

В этом примере, согласно заданию, необходимо было смоделировать работу аэропорта в течение 24 часов. Эта настройка задается в модели в окне Run/Setup/Replication Parameters.

В этих настройках мы длину репликации с бесконечности заменили на 24 часа.

Заключение

В настоящее время компьютерное моделирование и анализ данных являются широко используемыми инструментами, применяющимися в науке, но хотелось, чтобы компьютерное моделирование более активно внедрялось на реальных предприятиях и производствах. Общеизвестно, что компьютерные эксперименты гораздо дешевле, чем реальные действия с людьми и оборудованием. В связи с этим дисциплина «Компьютерное моделирование» должна входить не только в рабочие программы специальностей, связанных с информационными технологиями, а также для студентов других технических специальностей.

В этом учебном пособии были изложены основы теории моделирования систем, рассмотрены основные понятия, приведены различные классификации систем. Вторая глава пособия посвящена наиболее часто использующимся при моделировании бизнес-процессов предприятия структурным моделям и методологиям, позволяющим разрабатывать структурные модели IDEF0, IDEF3 и DFD. Во второй главе раскрыты понятия имитационного моделирования процессов и систем. Рассмотрено современное программное средство имитационного моделирования, в основы которого заложены два наиболее распространенных математических аппарата сети Петри и системы массового обслуживания.

Автор понимает, что в настоящее время в высших учебных заведениях при обучении студентов используют различные программные средства моделирования. Но использование пакета Arena является перспективным и апробировано на ряде зарубежных предприятий различных отраслей экономики. Тем более, что основа всех средств моделирования одна (сети Петри и СМО).

Список использованных источников

1. Бахвалов Л.А. Компьютерное моделирование: долгий путь к сияющим вершинам [Электронный ресурс]. – Режим доступа: <http://www.gpss-forum.narod.ru/GPSSmodeling.html>, свободный.
2. Бешенков С. А. Моделирование и формализация: методическое пособие. – М.: Лаборатория базовых знаний, 2002.
3. Большаков А. С. Моделирование в менеджменте: учеб. пособие. – М.: Филинь, 2000.
4. Бусленко Н. П. Моделирование сложных систем. – М.: Наука, 1978.
5. Бычков С. П., Храмов А. А. Разработка моделей в системе моделирования GPSS: учеб. пособие. – М.: МИФИ, 1997.
6. Введение в математическое моделирование: учеб. пособие / под ред. П. В. Трусова. – М.: Интермет инжиниринг, 2000.
7. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998.
8. Волчков С., Балахонова И. Бизнес-моделирование для совершенствования деятельности промышленного предприятия // ЦИТ «Платон» "КомпьютерПресс". 2001. №11.
9. Докукин В. П. Основы математического моделирования: Конспект лекций. Санкт-Петербургский ГГИ. – М.: Дело, 2000.
10. Имитационное моделирование производственных систем / Под ред. А. А. Вавилова. – М.: Машиностроение, 1983.
11. Калашников В. В., Рачев С. Т. Математические методы построения стохастических моделей обслуживания. – М.: Наука, 1988.
12. Киндлер Е. Языки моделирования. – М.: Энергия, 1985.
13. Клейнен Дж. Статистические методы в имитационном моделировании. – М.: Статистика, 1978.
14. Лоу А. М., Кельтон В. Д. Имитационное моделирование. Классика CS. – 3-е изд. – СПб.: Питер; Киев: Издательская группа BNV, 2004. – 847 с.: ил.
15. Лукасевич И. Я. Имитационное моделирование инвестиционных рисков [Электронный ресурс]. – Режим доступа: http://www.cfin.ru/finanalysis/imitation_model-2-1.shtml, свободный.
16. Марков А. А. Моделирование информационно-вычислительных процессов: учеб. пособие. – М.: Изд-во МГТУ им. Н. Э.Баумана, 1999.
17. Максимей И. В. Имитационное моделирование на ЭВМ. – М.: Радио и связь, 1988.

- 18.Марка Д. А., МакГоуэн К. Методология структурного анализа и проектирования. – М.: МетаТехнология, 1993.
- 19.Математическое моделирование: Методы, описания и исследования сложных систем / под ред. А. А. Самарского. – М.: Наука, 1989.
- 20.Перегудов Ф. И. Основы системного анализа: учебник. –2-е изд., доп. – Томск: Изд-во НТЛ, 1997.
- 21.Питерсон Дж. Теория сетей Петри и моделирование систем: Пер. с англ. – М.: Мир, 1984.
- 22.Рапопорт Б. М. Инжиниринг и моделирование бизнеса. – М: Тандем, 2001.
- 23.Романовский И. В. Исследование операций и статистическое моделирование. – С.-Пб.: С.-Пб.гос.ун-т, 1994.
- 24.Советов Б. Я. Моделирование систем: Учебник для вузов. – 3-е изд., перераб. и доп. – М.: Высш. Шк., 2001.
- 25.Суворова Н. Информационное моделирование: Величины, объекты, алгоритмы. – М.: Лаборатория базовых знаний, 2002.
- 26.Федотова Д. Э., Семенов Ю. Д., Чижик К. Н. CASE-технологии. – М.: Горячая линия – Телеком, 2003.
- 27.Шебеко Ю. А. Имитационное моделирование и ситуационный анализ бизнес-процессов принятия управленческих решений (учебное и практическое пособие). – М.: Диаграмма, 1999.
- 28.Шеннон Р. Ю. Имитационное моделирование систем – искусство и наука: Пер. с англ. – М: Мир, 1978.
- 29.Щепетова С. Е. Динамическое моделирование функционирования предприятия и формирование стратегии его поведения в конкурентной среде: автореф. дис. на соискание ученой степени к.э.н. – М.: Финансовая академия при Правительстве РФ, 2001.
- 30.ARENA Users Guide, Sewickley: Systems Modeling Co., 1996.
- 31.Barjis J., Shishkov B. UML based business systems modeling and simulation. – Proceedings of EuroSim 2001, 2001.
- 32.Giaglis G. M., Paul R.G., Okeefe R. M. Discrete simulation for business simulation. – Berlin: Springer – Verlag, 2003.
- 33.Goldsman D. A Whirlwind Tour of Computer Simulation Techniques. – www.hyperionics.com , 2003.
- 34.Hlupic V., Robinson S. Business Process Modeling and Analysis using discrete-event simulation. – Proceedings of the 1998 Winter Simulation Conference, pp.1363–1369.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
ГЛАВА 1.ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ МОДЕЛИРОВАНИЯ	4
1.1. Модель и моделирование	4
1.2 КЛАССИФИКАЦИЯ МОДЕЛЕЙ	5
1.2.1. Классификация моделей по степени абстрагирования модели от оригинала	6
1.2.2. Классификация моделей по степени устойчивости	13
1.2.3. Классификация моделей по отношению к внешним факторам	14
1.2.4. Классификация моделей по отношению ко времени	14
1.3. ЭТАПЫ РАЗРАБОТКИ МОДЕЛЕЙ	15
1.4. ВОПРОСЫ И ЗАДАНИЯ К ГЛАВЕ 1	21
ГЛАВА 2.....МЕТОДОЛОГИИ И СРЕДСТВА СТРУКТУРНОГО МОДЕЛИРОВАНИЯ ПРОЦЕССОВ И СИСТЕМ	22
2.1. SADT-МЕТОДОЛОГИЯ	22
2.1.1. Методология функционального моделирования IDEF0.....	22
2.1.2. Методология событийного моделирования IDEF3	33
2.2. МЕТОДОЛОГИЯ МОДЕЛИРОВАНИЯ ПОТОКОВ ДАННЫХ (DATA FLOW DIAGRAM).....	42
2.3. ВОПРОСЫ И ЗАДАНИЯ К ГЛАВЕ 2	45
ГЛАВА 3. ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СИСТЕМ	46
3.1. Достоинства и недостатки имитационного моделирования СИСТЕМ	46
3.2. МАТЕМАТИЧЕСКИЕ ОСНОВЫ ПП ARENA 7.0	48
3.2.1. Системы массового обслуживания	48
3.2.2. Сети Петри.....	51
3.3. НАЧАЛО РАБОТЫ С ПРОГРАММНЫМ ПАКЕТОМ ARENA 7.0	55
3.3.1. Создание модели с помощью ПП Arena 7.0	55
3.4. BASIC PROCESS PANEL (ПАНЕЛЬ ОСНОВНЫХ ПРОЦЕССОВ).....	57
3.4.1. Схемные модули.....	57
3.3.2. Модули данных	66
3.5. ADVANCED PROCESS PANEL (ПАНЕЛЬ УСОВЕРШЕНСТВОВАННЫХ ПРОЦЕССОВ)	71
3.5.1. Схемные модули.....	71
3.5.2. Модули данных	79
3.6. ADVANCED TRANSFER PANEL (ПАНЕЛЬ ПЕРЕМЕЩЕНИЯ).....	83
3.6.1. Схемные модули.....	83

3.6.2. Модули данных	92
3.7. ПАНЕЛЬ ОТЧЕТОВ.....	94
3.8. ПАНЕЛЬ НАВИГАЦИИ.....	96
3.9. ПОСТРОИТЕЛЬ ВЫРАЖЕНИЙ	96
3.10. ВОПРОСЫ И ЗАДАНИЯ К ГЛАВЕ 3.....	99
ГЛАВА 4. ЛАБОРАТОРНЫЙ ПРАКТИКУМ.....	100
4.1. ЗАДАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ	100
4.2. ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ.....	108
ЗАКЛЮЧЕНИЕ	116
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	117

Замятина Оксана Михайловна

кандидат технических наук, доцент кафедры автоматики
и компьютерных систем Томского политехнического
университета

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Учебное пособие

Научный редактор
доктор технических наук, профессор М. П. Силич

Редактор Н. Т. Синельникова

Подписано к печати

Формат 60x84/16, бумага офсетная.

Плоская печать. Усл. печ. л. . Уч. Изд. л.

Тираж экз. Заказ № . Цена свободная.

Издательство ТПУ. 634050, г. Томск, пр. Ленина, 30.