МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО АВТОНОМНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

И.В. МАКАРОВА, В.Г. МАВРИН, Г.В. МАВРИН

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ

Учебное пособие для выполнения лабораторных работ и курсового проекта

для студентов направления подготовки 09.03.02 Информационные системы и технологии

Набережные Челны

2018

УДК 519.876.5
ББК 22.1

Моделирование процессов и систем: учебное пособие для выполнения лабораторных работ и курсового проекта для студентов направления подготовки 09.03.02 Информационные системы и технологии / Макарова И.В., Маврин В.Г., Маврин Г.В. – Набережные Челны: изд-во НЧИ КФУ, 2018. - 130 с.

Учебное пособие предназначено для использования в учебном процессе студентами направления подготовки 09.03.02 Информационные системы и технологии.

Рецензенты:
д.ф.-м.н., профессор, заведующий кафедрой бизнес информатики и математических методов в экономике Набережночелнинского института КФУ Исавнин Алексей Геннадьевич;
д.т.н., заместитель директора по развитию ПАО «КАМАЗ» Хисамутдинов Равиль Миргалимович

Печатается по решению методической комиссии Набережночелнинского института (филиала) ФГАОУ ВО «Казанский (Приволжский) федеральный университет»

Оглавление

ВВЕДЕНИЕ

Учебное пособие предназначено для выполнения лабораторных работ и курсового проекта по дисциплине «Моделирование процессов и систем»

Целью выполнения задания студентами по лабораторным работам и курсового проектирования является освоение практических приемов имитационного моделирования, планирования, проведения и обработки данных компьютерного эксперимента.

В настоящее время в имитационном моделировании выделяют три подхода: системной динамики, дискретно-событийный и агентный. Из этих подходов в рамках многих дисциплин изучается дискретно-событийный подход, обеспечивающий универсальность и эффективность имитационного моделирования. Он ориентирован на исследование широкого класса сложных систем, представимых в виде систем массового обслуживания. Развитию и широкому распространению данного подхода в значительной степени способствовало наличие у разработчиков имитационных моделей специализированных систем имитационного моделирования.

Система AnyLogic разработана компанией XJTechnologies (сейчас The AnyLogic Company) в 1991 г. Это объектно-ориентированная система. Она для построения моделей позволяет использовать системную динамику, агентный и дискретно-событийный подходы. Также, наряду с GPSS World, получила признание и широкое распространение.

1 Общие вопросы имитационного моделирования

Моделирование состоит из трех этапов, на которых от разработчика модели требуются как формальные, так и неформальные умения. Первый этап — анализ реального явления и построение его упрощенной модели, второй этап — анализ построенной модели формальными средствами (например, с помощью компьютера), и, наконец, на третьем этапе выполняется интерпретация результатов, полученных на модели, в терминах реального явления. Первый и третий этапы не могут быть формализованы, их выполнение требует интуиции, творческого воображения и понимания сути изучаемого явления, т.е. качеств, присущих работникам искусства.

1.1 Модели процессов и систем

Современная парадигма научного исследования состоит в том, что реальные объекты заменяются их упрощенными представлениями, абстракциями, выбираемыми таким образом, чтобы в них была отражена суть явления, те свойства исходных объектов, которые существенны для решения поставленной проблемы. Построенный в результате упрощения объект называется моделью. Модель — это упрощенный аналог реального объекта или явления, представляющий законы поведения входящих в объект частей и их связи. Построение модели и ее анализ называется моделированием. В научной работе моделирование является одним из главных элементов научного познания.

В практической деятельности цель построения модели — решение некоторой проблемы реального мира, которую дорого либо невозможно

решать, экспериментируя с реальным объектом. На рисунке 1 схематично представлены эти два пути: прямой путь решения проблемы, основанный на экспериментах с реальным объектом, может быть заменен "окольным" путем, на котором эксперименты проводятся с абстрактной моделью.

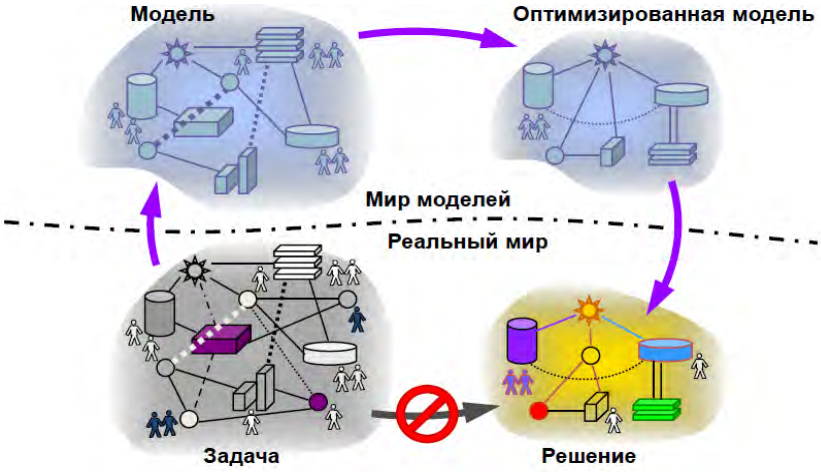


Рис. 1.1. Два пути решения проблемы реального мира

Практика показывает, что схема на рисунке 1 коммутативна при правильно выбранном уровне абстракции. Иными словами, оба пути могут привести к решению проблемы, но с помощью моделирования такое решение находится значительно проще и дешевле.

Обычно исходная проблема состоит в анализе существующего или предполагаемого объекта для принятия решения по его управлению. Например, таким объектом может быть географически распределенная система поставщиков сырья, заводов, складов готовой продукции и их транспортные связи. Другой пример — сервисный центр, осуществляющий услуги по техническому обслуживанию и ремонту автомобилей.

При построении модели как заменителя реальной системы выделяются те аспекты, которые существенны для решения проблемы, и игнорируются те аспекты, которые усложняют проблему, делают анализ очень сложным или вообще невозможным. Проблема анализа всегда ставится в мире реальных объектов. В примере с сервисным центром это может быть проблема оптимального использования существующих ресурсов (организация движения автомобилей по сервисному центру и использования постов ТО и ремонта) дня организации обслуживания клиентов.

Принимать решения по управлению ресурсами, перестраивая реальную систему, экономически нецелесообразно. Другой путь решения — сформулировать эту проблему для модели, которую составят схема сервисного центра, количество постов обслуживания, средняя интенсивность прибытия автомобилей на обслуживание, среднее время ремонта автомобиля и т. п.

Реальные объекты и ситуации обычно сложны, и модели нужны для того, чтобы ограничить эту сложность, дать возможность понять ситуацию, понять тенденции изменения ситуации (спрогнозировать будущее поведение анализируемой системы), принять решение по изменению будущего поведения системы и проверить его. Если модель отражает свойства системы, существенные для решения конкретной проблемы, то анализ модели позволяет вывести характеристики, которые объяснят известные и предскажут новые свойства исследуемой реальной системы без экспериментов с самой системой. С помощью моделирования

получено множество впечатляющих результатов в науке, технике и на производстве.

1.2 Моделирование для поддержки принятия управленческих решений

Принятие разумных решений по рациональной организации и управлению современными системами становится невозможным на основе обычного здравого смысла или интуиции из-за возрастающей сложности систем. Еще в 1969 г. известный ученый, родоначальник системной динамики Джей Форрестер отмечал, что на основе интуиции для управления сложными системами чаще выбираются неверные решения, чем верные, и это происходит потому, что в сложной системе причинно-следственные отношения ее параметров не являются простыми и ясными. В литературе имеется большое число примеров, показывающих, что люди неспособны предвидеть результат их воздействий в сложных системах. Примером может служить каскадное развитие аварий в энергосистемах Северо-Запада США 16 августа 2003 г. и в Московском регионе 25 мая 2005 г., приведших к миллиардным потерям и затронувшим миллионы людей.

Повышение производительности и надежности, уменьшение стоимости и рисков, оценка чувствительности системы к изменениям параметров, оптимизация структуры — все эти проблемы встают как при эксплуатации существующих, так и при проектировании новых технических и организационных систем. Трудность понимания причинно-следственных зависимостей в сложной системе приводит к неэффективной организации систем, ошибкам в их проектировании, большим затратам на

устранение ошибок. Сегодня моделирование становится единственным практическим эффективным средством нахождения путей оптимального (либо приемлемого) решения проблем в сложных системах, средством поддержки принятия ответственных решений.

Моделирование особенно важно именно тогда, когда система состоит из многих параллельно функционирующих во времени и взаимодействующих подсистем. Такие системы наиболее часто встречаются в жизни. Каждый человек мыслит последовательно, даже очень умный человек в конкретный момент времени обычно может думать только об одном деле. Поэтому понимание одновременного развития во времени многих влияющих друг на друга процессов является для человека трудной задачей. Имитационная модель помогает понять сложные системы, предсказать их поведение и развитие процессов в различных ситуациях и, наконец, дает возможность изменять параметры и даже структуру модели, чтобы направить эти процессы в желаемое русло. Модели позволяют оценить эффект планируемых изменений, выполнить сравнительный анализ качества возможных вариантов решений. Такое моделирование может осуществляться в реальном времени, что позволяет использовать его результаты в различных технологиях (от оперативного управления до тренинга персонала).

1.3 Уровни абстракции и адекватность модели

Основной парадокс моделирования состоит в том, что изучается упрощенная модель системы, а полученные выводы применяются к исходной реальной системе со всеми ее сложностями. Является ли такая подмена правомерной?

При изучении естественных объектов исследователь абстрагируется от несущественных, случайных деталей, которые не просто усложняют, но могут и затемнить само явление. Например, при анализе нефтеналивного порта удобно говорить о танкерах как о емкостях, из которых производится перекачка определенного объема нефти с некоторой скоростью, а не как о кораблях с каютами, определенной численностью экипажа и т. п. Поскольку все абстракции неполны и неточны, можно говорить только о приближенном соответствии реальности тех результатов, которые получены исследованием моделей. Соответствие модели моделируемому объекту или явлению при решении конкретной проблемы называется адекватностью. Адекватность определяет возможность использования приближенных результатов, полученных на модели, для решения практической проблемы реального мира. Часто адекватность модели определяется рядом условий и ограничений на сущности реального мира, и для того чтобы использовать результаты анализа, полученные на модели, необходимо тщательно проверять (или даже обеспечивать) эти ограничения и условия при функционировании реальной системы (например, чтобы сделать процессы в обществе управляемыми, создается вертикаль власти). Поскольку адекватность модели определяется только возможностью использования модели для решения конкретной проблемы, адекватная модель не обязательно должна досконально отображать процессы, происходящие в моделируемой системе (или, что то же самое, модель не обязательно должна отображать "физически правильную" картину мира).

На рисунке 2 представлена шкала уровней абстракции и примеры проблем моделирования в конкретных областях, приблизительно расставленные на этой шкале.

Рис1.2. Уровни абстракции при решении проблемы

На нижнем уровне абстракции решаются проблемы, в которых важны отдельные физические объекты, их индивидуальное поведение и физические связи, точные размеры, расстояния, времена. Примерами моделей, относящихся к этому уровню абстракции, являются модели движения пешеходов, модели движения механических систем и их систем управления. На среднем уровне обычно решаются проблемы массового производства и обслуживания, здесь представляются отдельные объекты, но их физическими размерами пренебрегают; значения скоростей и времен усредняются или используются их стохастические значения. Примерами моделей на этом уровне абстракции являются модели массового обслуживания, модели движения транспорта, модели управления ресурсами. Высокий уровень абстракции используется при разработке моделей сложных систем, в которых исследователь абстрагируется от индивидуальных объектов и их поведений, рассматривая только

совокупности объектов и их интегральные, агрегированные характеристики, тенденции изменения значений, влияние на динамику системы причинных обратных связей. Модели рынка и динамики народонаселения, экологические модели и классические модели распространения эпидемий построены на этом уровне абстракции.

Для каждой цели исследования даже одного и того же объекта реального мира должна быть построена своя модель, которая соответствует этой цели. Для решения конкретной задачи будет удобна модель, адекватно отражающая структуру объекта и законы, по которым он функционирует на выбранном уровне абстракции. Например, очевидно, что планеты не материальные точки, но при такой абстракции в рамках ньютоновской теории тяготения можно достаточно точно предсказать характеристики движения планет. Однако эта модель требует уточнения для расчета траекторий спутников и ракет. Для решения проблемы оптимального использования транспорта в нашем примере и эта модель бесполезна, здесь необходимы подробные карты, расстояния и времена. То, что Земля на карте представляется плоской, не существенно для решения транспортных проблем.

Хотя существуют устоявшиеся подходы к выбору уровня абстракции и разумные объяснения данного выбора для построения достаточно адекватных моделей при решении многих типов проблем, все же общей методики построения модели с требуемым уровнем адекватности не существует. В качестве рекомендации по выбору уровня абстракции можно сказать лишь следующее.

Нужно начать с наиболее простой модели, отражающей только самые существенные (с точки зрения исследователя) аспекты моделируемой системы. После обнаружения неадекватности модели, т.е. неприменимости ее к решению поставленной проблемы, отдельные подструктуры и процессы модели следует реализовать более детально, на более низком уровне абстракции. Можно быть уверенным, что разработка последовательности усложняющихся все более подробных моделей может привести к обеспечению приемлемой адекватности при решении любой конкретной задачи.

Очевидно, что никакая модель никогда не дает полных знаний о моделируемом объекте. Например, никакая модель не может представить все характеристики планеты Земля. С другой стороны, очевидно также, что любая конкретная задача не потребует для своего решения знания всех этих характеристик.

Конечно, можно построить модели, которые абстрагируются от существенных аспектов реальности. Такие модели будут неадекватными, а выводы, полученные на основе этих моделей, будут неверными.

1.4 Виды моделей

1.4.1 Статические и динамические модели

Статические модели оперируют характеристиками и объектами, не изменяющимися во времени. В динамических моделях, которые обычно более сложны, изменение параметров во времени является существенным. Модель сервисного центра является динамической: в ней моделируется поведение во времени отдельных объектов системы: движение

автомобилей в сервисной зоне, обслуживание автомобилей, передвижение рабочих в сервисной зоне и т.д.

Статические модели обычно имеют дело с установившимися процессами, уравнениями балансового типа, с предельными стационарными характеристиками. Моделирование динамических систем состоит в имитации правил перехода системы из одного состояния в другое с течением времени. Под состоянием системы понимается набор значений ее существенных параметров и переменных. Изменение состояния системы во времени в динамических системах — это изменение значений переменных системы в соответствии с законами, определяющими связи переменных и их зависимости друг от друга во времени.

1.4.2 Непрерывные, дискретные и гибридные модели

Реальные физические объекты функционируют в непрерывном времени, и для изучения многих проблем физических систем их модели должны быть непрерывными. Состояние таких моделей изменяется непрерывно во времени. Это модели движения в реальных координатах, модели химического производства и т. п. Процессы движения объектов и процессы обслуживания автомобилей являются непрерывными.

На более высоком уровне абстракции для многих систем адекватными являются модели, в которых переходы системы из одного состояния в другое можно считать мгновенными, происходящими в дискретные моменты времени. Такие системы называются дискретными. Примером мгновенного перехода является изменение числа клиентов банка или количества покупателей в магазине. Очевидно, что дискретные

системы — это абстракция, процессы в природе не происходят мгновенно. В реальный магазин реальный покупатель входит в течение некоторого времени, он может застрять в дверях, колеблясь, войти или нет, и всегда существует непрерывная последовательность его положения во время прохождения дверей магазина. Однако при построении модели магазина для оценки, например, средней длины очереди в кассу при заданном потоке покупателей и известных характеристиках обслуживания кассиром клиентов можно абстрагироваться от этих второстепенных явлений и считать систему дискретной: результаты анализа полученной дискретной модели обычно достаточно точны для принятия обоснованных управленческих решений для подобных систем. В модели нефтеналивного порта мгновенными можно считать, например, переходы светофоров на входе в гавань из состояния "запрещено" в состояние "разрешено".

На еще более высоком уровне абстракции при анализе систем также используются непрерывные модели, что характерно для системной динамики. Потоки машин на автострадах, потребительский спрос, распространение инфекции среди населения часто удобно описывать с помощью взаимозависимостей непрерывных переменных, описывающих количества, интенсивности изменения этих количеств, степени влияния одних количеств на другие. Соотношения таких переменных выражаются обычно дифференциальными уравнениями.

Во многих случаях в реальных системах присутствуют оба типа процессов, и если оба они являются существенными для анализа системы, то и в модели одни процессы должны представляться как непрерывные, другие — как Дискретные. Такие модели со смешанным типом процессов

называются гибридными. Например, если при анализе функционирования магазина существенным является не только количество покупателей, но и пространственное их положение и перемещение покупателей, то модель в этом случае должна представлять смесь непрерывных и дискретных процессов, т. е. это гибридная модель. Другим примером может служить модель функционирования крупного банка. Поток инвестиций, получение и выдача кредитов в нормальном режиме описывается набором дифференциальных и алгебраических уравнений, т. е. модель является непрерывной. Однако существуют ситуации, например дефолт (дискретное событие), в результате чего возникает паника у населения, и с этого момента система описывается совершенно другой непрерывной моделью. Модель данного процесса на том уровне абстракции, на котором мы хотим адекватно описать оба режима работы банка и переход между режимами, должна включать как описание непрерывных процессов, так и дискретные события, а также их взаимозависимости.

1.4.3 Детерминированные и стохастические модели

При моделировании сложных реальных систем исследователь часто сталкивается с ситуациями, в которых случайные воздействия играют существенную роль. Стохастические модели, в отличие от детерминированных, учитывают вероятностный характер параметров моделируемого объекта. Например, в модели сервисного центра не могут быть определены точно моменты заезда автомобилей на ремонт. Данные моменты являются случайными величинами, потому модель эта является стохастической: значения переменных величин модели, которые зависят от реализаций случайных величин, сами становятся случайными величинами.

Анализ подобных моделей выполняется на компьютере на основе статистики, набираемой в ходе имитационных экспериментов при многократном прогоне модели для различных значений исходных случайных величин, выбранных в соответствии с их статистическими характеристиками.

1.4.4 Аналитические и имитационные модели

Использование абстракций при решении проблем с помощью моделей часто состоит в применении того или иного математического аппарата. Простейшими математическими моделями являются алгебраические соотношения, и анализ модели часто сводится к аналитическому решению этих уравнений. Некоторые динамические системы можно описать в замкнутой форме, например, в виде систем линейных дифференциальных и алгебраических уравнений и получить решение аналитически. Такое моделирование называется аналитическим. При аналитическом моделировании процессы функционирования исследуемой системы записываются в виде алгебраических, интегральных, дифференциальных уравнений и логических соотношений, и в некоторых случаях анализ этих соотношений можно выполнить с помощью аналитических преобразований. Современным средством поддержки аналитического моделирования являются электронные таблицы типа MS Excel.

Однако использование чисто аналитических методов при моделировании реальных систем сталкивается с серьезными трудностями: классические математические модели, допускающие аналитическое решение, в большинстве случаев к реальным задачам неприменимы.

Например, в модели сервисного центра построить аналитическую формулу для оценки коэффициента использования оборудования невозможно хотя бы потому, что в системе существуют стохастические процессы, есть приоритеты обработки заявок на использование ресурсов, внутренний параллелизм в обрабатывающих подсистемах, прерывания работы и т. п. Даже если аналитические модели удается построить, для реальных систем они часто являются существенно нелинейными, и чисто математические соотношения в них обычно дополняются логико-семантическими операциями, а для них аналитического решения не существует. Поэтому при анализе систем часто стоит выбор между моделью, которая является реалистическим аналогом реальной ситуации, но не разрешимой аналитически, и более простой, но неадекватной моделью, математический анализ которой возможен.

При имитационном моделировании структура моделируемой системы — ее подсистемы и связи — непосредственно представлена структурой модели, а процесс функционирования подсистем, выраженный в виде правил и уравнений, связывающих переменные, имитируется на компьютере.

2 Парадигмы имитационного моделирования

Имитационное моделирование — это разработка и выполнение на компьютере программной системы, отражающей структуру и функционирование (поведение) моделируемого объекта или явления во времени. Такую программную систему называют имитационной моделью этого объекта или явления. Объекты и сущности имитационной модели представляют объекты и сущности реального мира, а связи структурных

единиц объекта моделирования отражаются в интерфейсных связях соответствующих объектов модели. Таким образом, имитационная модель — это упрощенное подобие реальной системы, либо существующей, либо той, которую предполагается создать в будущем. Имитационная модель обычно представляется компьютерной программой, выполнение программы можно считать имитацией поведения исходной системы во времени.

В русскоязычной литературе термин "моделирование" соответствует американскому "modeling" и имеет смысл создание модели и ее анализ, причем под термином "модель" понимается объект любой природы, упрощенно представляющий исследуемую систему. Слова "имитационное моделирование" и "вычислительный (компьютерный) эксперимент" соответствуют англоязычному термину "simulation". Эти термины подразумевают разработку модели именно как компьютерной программы и исполнение этой программы на компьютере.

Итак, имитационное моделирование — это деятельность по разработке программных моделей реальных или гипотетических систем, выполнение этих программ на компьютере и анализ результатов компьютерных экспериментов по исследованию поведения моделей. Имитационное моделирование имеет существенные преимущества перед аналитическим моделированием в тех случаях, когда:

- ✓ отношения между переменными в модели нелинейны, и поэтому аналитические модели трудно или невозможно построить;
- ✓ модель содержит стохастические компоненты;

- ✓ для понимания поведения системы требуется визуализация динамики происходящих в ней процессов;
- ✓ модель содержит много параллельно функционирующих взаимодействующих компонентов.

Во многих случаях имитационное моделирование — это единственный способ получить представление о поведении сложной системы и провести ее анализ.

Имитационное моделирование — очень обширная область. Можно по- разному подходить к классификации решаемых в ней задач. В соответствии с одной из классификаций эта область насчитывает в настоящее время четыре основных направления: моделирование динамических систем, дискретно- событийное моделирование, системная динамика и агентное моделирование. В каждом из этих направлений развиваются свои инструментальные средства, упрощающие разработку моделей и их анализ. Данные направления (кроме агентного моделирования) базируются на концепциях и парадигмах, которые появились и были зафиксированы в инструментальных пакетах моделирования несколько десятилетий назад и с тех пор не менялись.

Например, моделирование динамических систем направлено на исследование сложных объектов, поведение которых описывается системами алгебро- дифференциальных уравнений. Инженерным подходом к моделированию таких объектов 40 лет назад была сборка блок-схем из решающих блоков аналоговых компьютеров: интеграторов, усилителей и сумматоров, токи и напряжения в которых представляли переменные и параметры моделируемой системы. Этот подход и сейчас

является основным в моделировании динамических систем, только решающие блоки являются не аппаратными, а программными. Он реализован, например, в инструментальной среде Simulink. В другой области идея моделирования систем с дискретными событиями, в которых потоки пассивных заявок обрабатываются активными приборами, была сформулирована более 40 лет назад и реализована в среде моделирования GPSS, которая с некоторыми модификациями до сих пор используется для обучения имитационному моделированию. Еще одним примером является системная динамика. Идея потоковых диаграмм, отражающих причинно-следственные связи в сложной системе, была предложена Дж. Форрестером еще в 1958 году, она реализована в существующих до сих пор на рынке системах моделирования.

2.1 Моделирование динамических систем

Динамические системы – это сложные объекты, поведение которых описывается системами алгебраических и дифференциальных уравнений, а также событиями, меняющими либо среду, либо модель, либо даже саму структуру модели. К этому классу относятся системы управления, физические и механические объекты, объекты химической технологии, системы обработки сигналов и т.д.

Моделирование динамических систем состоит в имитации правил перехода системы из одного состояния в другое с течением времени. Под состоянием системы при этом понимается набор значений ее существенных параметров и переменных. Изменение состояния системы во времени в динамических системах – это изменение значений

переменных системы в соответствии с законами, определяющими связи переменных и их зависимости друг от друга во времени.

Моделирование динамических систем, по сути, является прародителем системно-динамического подхода. Моделирование с помощью данного подхода используется в мехатронике, электрической, химической и других инженерных областях в качестве стандартного этапа процесса разработки. С математической точки зрения динамическая система представляет собой набор переменных состояния и алгебраических дифференциальных уравнений различного вида, заданных для этих переменных и описывающих их изменение с течением времени. В отличие от системной динамики, переменные здесь несут некоторый "физический" смысл: координаты местоположения, скорость, ускорение, сила, концентрация и.т.д., они, как это следует из их смысла, непрерывны и не являются агрегированными величинами, отражающими, например, общее количество или среднее значение нескольких сущностей.

Классическая динамическая система состоит из двух связанных подсистем: объекта управления и регулятора.

Пусть объектом управления является автомобиль, который разгоняется до скорости T. Величину скорости T необходимо поддерживать на заданном уровне Tz. Скорость автомобиля зависит от входного параметра – в данном случае, от силы нажатия на педаль газа U. Объект управления подвергается внешнему возмущению F (F может характеризовать, к примеру, силу трения автомобиля об асфальт или воздух), вследствие чего значение выходного параметра T может меняться. Поддержание значения T на заданном уровне Tz есть задача

регулятора. Регулятор по разнице заданного и текущего значений входного параметра ($Tz - T$) формирует величину входного параметра объекта управления – в данном случае регулируется сила нажатия на педаль газа водителем, который и играет роль регулятора.

Для моделирования динамических систем уже с 50-х годов XX века стал использоваться метод построения структурных схем из решающих блоков аналоговых компьютеров: источников тока, интеграторов, усилителей, сумматоров, умножителей. Токи и напряжения в блоках представляли константы, переменные и параметры моделируемой системы, а сами блоки реализовывали те преобразования, которые удобно выполнить аппаратно с электрическими токами: интегрирование, сложение и умножение.

Начиная с тех давних пор и по сегодняшний день блочный подход является основным в моделировании динамических систем, только сейчас решающие блоки являются не аппаратными, а реализуются программно.

В качестве простейшего примера применения блоковых диаграмм (стейтчартов) рассмотрим модель регулируемого пешеходного перехода со светофором, разрешающим или запрещающим движение транспорта.

2.2 Дискретно-событийное моделирование

Системы называются *дискретно-событийными* (или просто дискретными), если изменения переменных состояния в них происходят только в явно определенные моменты времени или под влиянием явно определенных событий. Находясь в некотором состоянии, дискретная система сохраняет его (не изменяет своих характеристик) до наступления очередного события, под воздействием которого переменные системы (и,

следовательно, ее состояние) изменяются скачком. Например, при построении модели банка состояние системы может быть представлено количеством клиентов в помещении банка и числом занятых кассиров. Состояние системы изменяется, если только новый клиент входит в банк или когда освобождается кассир, а это на некотором уровне абстракции можно считать мгновенными событиями, сопровождаемыми изменением состояния системы.

Термин "дискретно-событийное моделирование" исторически закрепился за моделированием систем обслуживания потоков объектов некоторой природы: клиентов банка, автомобилей на заправочной станции, телефонных вызовов, пациентов в поликлиниках и т. п. Такие системы названы *системами массового обслуживания (СМО)*. В моделях таких систем типичными объектами являются заявки и обслуживающие их приборы. *Заявки* моделируют клиентов, заказы на выполнение работ, телефонные вызовы, покупателей, автомашины и т.п. *Приборы обслуживания* моделируют кассиров, продавцов в магазине, лифты, линии передачи данных и т. п. Сам процесс *обслуживания* — с точки зрения модели — это просто временная задержка. Каждая СМО состоит из какого-то числа приборов обслуживания и имеет на входе потоки заявок, поступающих обычно в случайные моменты времени. Обслуживание заявки каждым прибором СМО происходит обычно также в течение случайного времени. Различные СМО отличаются характеристиками случайных входных потоков заявок, числом обслуживающих приборов и дисциплиной обслуживания. *Дисциплиной обслуживания* называют порядок прохождения заявками приборов в системе, правила ухода заявок

из системы, характеристики времени обслуживания заявок приборами, правила организации очередей к приборам.

Системы массового обслуживания являются типичными системами событийно- дискретного типа. Событием в СМО является появление в системе очередной заявки, постановка заявки в очередь на обслуживание, начало и окончание обслуживания и т. п.

Из-за однотипности, похожести задач, решаемых моделями систем массового обслуживания, удобно отдельные блоки (генераторы заявок, обслуживающие приборы, очереди и т. п.) реализовать как набор (библиотеку) объектов, из которых может собираться структура конкретной модели и параметры которых можно настраивать в зависимости от характеристик моделируемой системы. Именно для этих целей в AnyLogic создана библиотека Enterprise Library. Она предоставляет высокоуровневый интерфейс для быстрого создания дискретно-событийных моделей с помощью блок-схем.

В AnyLogic, как и в большинстве других инструментов для моделирования дискретных систем, ключевой сущностью является *транзакция (заявка)*. Заявки обрабатываются в системе, захватывают и освобождают ресурсы, тем самым влияя на те характеристики производительности системы, которые нас интересуют. В модели сервисного заявки моделируют автомобили клиентов автосервисов — они создаются, двигаются по системе и, в конце концов, выходят из нее, имитируя то, как автомобили перемещаются в реальном сервисном центре. В модели одновременно перемещается множество независимых заявок.

Для того чтобы придать заявкам индивидуальность, нужно наделить их атрибутами или свойствами. Например, заявке можно приписать вид обслуживания (ТО, ремонт, диагностика и т.д.). Дополнительно мы можем отслеживать время входа заявки в систему, времена нахождения заявок в очередях и т.д.

Заявки в модели будут использовать ресурсы. Ресурсы представляют рабочих и оборудование автосервиса, которые используются автомобилями. Посты ожидания и обслуживания также можно представить как ресурсы. Ресурсов одного типа может быть несколько, но количество ресурсов всегда ограничено, поэтому заявки конкурируют за ресурсы. Они получают ресурсы в соответствии с некоторой дисциплиной обслуживания. Заявки для своего обслуживания могут потребовать наличия нескольких ресурсов одновременно. Если заявка не находит свободного ресурса, она ожидает в очереди.

Модели дискретных событийных систем, как было сказано ранее, строятся как имитация прохождения заявок по сети, состоящей из блоков обработки и направленных связей, соединяющих блоки.

Особый класс блоков библиотеки Enterprise Library необходим для построения моделей систем, которые имеют существенные отличия от простых СМО. *Первое отличие* состоит в том, что в этих системах для обслуживания потока заявок используются ресурсы. Например, в модели оптового магазина, который, конечно, является системой обслуживания, может несколько типов ресурсов: погрузчики на складе, продавцы и т.д. Наличие некоторого набора свободных ресурсов обязательно для выполнения процесса обслуживания клиента: очередная заявка,

моделирующая клиента, может быть принята к обслуживанию, если свободны ресурсы определенных типов, например.

Второй отличительной особенностью этих систем является то, что время обслуживания заявок в системе зависит от геометрических характеристик системы. Например, время, затраченное на движение погрузчика от зоны выгрузки товара до ячеек склада, зависит от места расположения свободной ячейки, поэтому и время наполнения склада до следующего заказа зависит от деталей архитектурного плана здания: длины склада, расположения ячеек, их назначения. Таким образом, время обслуживания заявок в подобной системе существенно зависит от реальных расстояний на плане. Подобными особенностями обладают также производственные системы с конвейерами и средствами внутрицеховой доставки. Другую группу обслуживающих систем, в которых архитектурные детали и расположение на плане пунктов обслуживания играет существенную роль при их анализе, составляют крупные магазины, аэропорты и им подобные системы. Имитационные модели всех таких систем являются, конечно, дискретно-событийными, но их построение требует специальных средств.

Для построения моделей таких систем обслуживания в библиотеку включена группа блоков "Транспортировка по сети".

Транспортная сеть состоит из узлов и их связей. Узлами сети являются все те пункты, где заявка может находиться некоторое время, а именно складские помещения, зона приемки товара, зона отгрузки товара, вход и выход. Ребра сети — это пути, по которым в процессе моделирования будут перемещаться заявки.

Узлы сети задаются в анимации с помощью прямоугольников, а ребра сети — с помощью линий и ломаных. Когда заявка переместится в один из узлов сети, визуально она будет отображаться в случайно выбранной точке внутри узла-прямоугольника. Кроме узлов, помешенных в очевидные места плана (складские помещение, зона приемки), необходимо задать промежуточные узлы сети, например, на пересечении коридоров. Для того чтобы соединить два узла сети, нужно нарисовать ломаную, конечные точки которой будут лежать в соединяемых узлах (например, прямоугольниках). Во время моделирования эти прямоугольники (и только они) будут связаны в единую транспортную сеть. Прямоугольники, лежащие не под конечными точками ломаной, не будут связаны в сеть и не будут рассматриваться как ресурсы модели.

2.3 Моделирование многоагентных систем

Существует множество определений понятия агента. Общим во всех этих определениях является то, что агент — это некоторая сущность, которая обладает активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, может взаимодействовать с окружением и другими агентами, а также может изменяться (эволюционировать).

Многоагентные (или просто агентные) модели используются для исследования децентрализованных систем, динамика функционирования которых определяется не глобальными правилами и законами, а наоборот, эти глобальные правила и законы являются результатом индивидуальной активности членов группы. Цель агентных моделей — получить

представление об этих глобальных правилах, общем поведении системы, исходя из предположений об индивидуальном, частном поведении ее отдельных активных объектов и взаимодействии этих объектов в системе. Рост производительности компьютеров и достижения в информационных технологиях, использованные в AnyLogic, сделали возможным реализацию агентных моделей, содержащих десятки и даже сотни тысяч активных агентов.

Моделирование агентов и многоагентных систем не представляет сложностей в AnyLogic ни в концептуальном, ни в техническом аспекте: все указанные ранее свойства агентов легко реализуются в этой системе. Основной концепцией AnyLogic является та, что модель состоит из активных объектов, имеющих каждый свои правила поведения и взаимодействующих через явно определенные интерфейсы. Поэтому агентный подход к построению моделей является в AnyLogic совершенно естественным: в этой среде можно быстро создавать модели с агентами, взаимодействующими как друг с другом, так и со средой.

При создании агентной модели логика поведения агентов и их взаимодействие не всегда могут быть выражены чисто графическими средствами, здесь часто приходится использовать программный код. Вообще, AnyLogic предоставляет для разработки агентных моделей всю мощь визуальной графической разработки: стейтчарты, события, таймеры, синхронное и асинхронное планирование событий, библиотеки ранее определенных активных объектов — все это оказывается удобным и естественным при разработке агентных моделей. Однако разработчик

обычно должен включать в агентную модель фрагменты кода на языке Java.

Агент естественно реализовать с помощью базового объекта AnyLogic — активного объекта. В модели на AnyLogic можно создавать классы активных объектов и далее использовать в модели любое число экземпляров этих классов. Активный объект имеет параметры, которые можно менять извне, переменные, которые можно считать памятью агента, а также поведение.

Параметры могут указывать пол агента, дату рождения и т. п. Переменными можно выразить, например, возраст агента, его координаты в пространстве, социальные свойства. Стейтчарты могут выражать поведение: состояния агента и изменение состояний под воздействием событий и условий. Кроме того, агент может иметь интерфейс для взаимодействия с окружением.

Поведение агента может выражать, например, правила действий агента или законы перемещения агента в пространстве, изменения его социального статуса, переходы в разные возрастные или социальные группы, изменения образования и дохода, семейного положения и т.п. Для представления дискретного поведения естественно использовать стейтчарты. Разные роли агента могут выражаться разными стейтчартами.

Типичная архитектура любой агентной модели включает в себя интерфейсные переменные переменные и методы (функции), карты состояний и события (стейтчарты), а также анимацию агента.

Множество агентов в модели реплицировано (размножено), их количество может быть установлено статически (заданием параметра реп-

ликации), но может и изменяться динамически. Если **people** — имя класса агентов, а Person — имя реплицированного множества их экземпляров, то для создания нового агента нужно вызвать функцию **add_people()**, а для динамического удаления агента с номером p (например, в случае его смерти), следует вызвать функцию **remove_people (p)**.

Очевидно, что в среде может быть создано несколько групп агентов, функционирующих в одной среде, например два враждебных клана. Реплицированные агенты сами могут иметь вложенные группы агентов, например, компании могут включать своих работников.

Агенты обычно функционируют в некоторой среде, и взаимодействие со средой является важной задачей агента. Роль среды для агентов играет либо активный объект, в который вложены агенты, либо в качестве среды может быть использован другой активный объект. Среда может быть пассивной либо иметь свое поведение: это тоже активный объект. Динамика среды может задаваться уравнениями, стейтчартом, потоковой диаграммой, использованием таймеров и т.д. Например, у среды может быть тактовый таймер, который циклически запускает вызов функции обращения к агентам для их продвижения или выполнения ими собственных операций. Функционирование агентов в такой среде можно назвать синхронным в отличие от асинхронного их функционирования, когда каждый агент имеет свои собственные средства продвижения времени.

Агенты могут обитать в различных типах пространств. AnyLogic поддерживает два типа пространств: двумерное непрерывное и двумерное

дискретное. AnyLogic Professional также поддерживает геопространственный тип ГИС.

В непрерывном пространстве есть возможность изменять местоположение агента и получать информацию о его текущем местоположении, перемещать агента с заданной скоростью из одного места в другое, выполнять действия по его прибытии в место назначения, рисовать анимацию (статического или движущегося) агента, устанавливать соединения согласно выбранному шаблону расположения агентов, и многие другие возможности. Часть функциональности непрерывного пространства даже не требует того, чтобы агенты принадлежали явно заданной среде - если среда не указана, то по умолчанию будет приниматься, что пространство именно непрерывное (но если агенты принадлежат среде, то тип пространства должен быть задан явно).

Двумерное дискретное пространство представляет собой прямоугольный массив ячеек, полностью или частично занятых агентами. В одной ячейке может находиться не больше одного агента. Поддержка этого типа пространства в AnyLogic включает в себя возможности по распределению агентов по ячейкам, их перемещению в соседние или любые другие ячейки, определению того, какие агенты являются соседями (согласно выбранной модели соседства), нахождению свободных ячеек и т.д.

Взаимодействие агентов в агентных моделях может быть реализовано различными способами. Если связи между агентами достаточно постоянны, то агенту нужно запоминать тех агентов, которые

состоят с ним в какой-то связи. Смысл таких связей может быть, например, следующим: друг, коллега, родитель, ребенок и т.д. Одним из способов хранения связей агента является хранение их в простых переменных или коллекциях. Эти связи можно устанавливать при создании агентов или динамически во время выполнения модели.

AnyLogic предоставляет встроенный механизм поддержки "плоских" связей, таких, как друзья, социальные контакты и т.д. Любой агент может иметь определенное число связей - ссылок на других агентов, обитающих в той же среде. Эти связи всегда двунаправленны: если у агента A в списке соединений значится B, то и у агента B, в свою очередь, в его списке связей будет значиться A. Связями можно управлять самостоятельно с помощью программного интерфейса агента и/или автоматически с помощью среды. AnyLogic поддерживает несколько типов сетей агентов:

Случайное – агенты соединяются случайно, у каждого агента устанавливается заданное количество связей.

Согласно расстоянию – друг с другом соединяются те агенты, расстояние между которыми не больше заданного радиуса соединения (только в непрерывном пространстве).

Решеточно упорядоченное кольцо – связи агентов образуют кольцо, в котором каждый агент соединяется с заданным количеством ближайших агентов.

Малый мир – представляет собой решеточно упорядоченное кольцо, где некоторые связи были разорваны и установлены с удаленными агентами.

Безразмерная – некоторые агенты являются "хабами" (или концентраторами) с множествами соединений, а некоторые - "отшельниками" с небольшим числом соединений.

Таким образом, агентный подход применяется для решения проблем во всех тех случаях, когда именно индивидуальное поведение объектов является существенным в системе, а интегральные характеристики и динамика всей системы выводятся из этих индивидуальных поведений. С помощью агентов можно моделировать рынки (агент представляет потенциального покупателя со своей историей, возрастом и родом занятий), конкуренцию компаний на рынке (агент — компания со своим капиталом, стратегией и бизнес-процессами), динамику населения (агент — семья, житель или избиратель со своими политическими предпочтениями, уровнем образования, местом проживания) и многое другое.

2.4 Системная динамика

Идея моделирования сложных систем на самом верхнем уровне абстракции, когда исследователь абстрагируется от индивидуальных объектов системы (сотрудников, машин, документов, товаров) и рассматривает только агрегированные количественные характеристики потоков таких объектов, взаимовлияния и взаимозависимости динамики этих потоков была предложена Дж. Форрестером (Jay W. Forrester) почти 50 лет назад. Дж. Форрестер применил принципы исследования обратной информационной связи для демонстрации того, что динамика функционирования сложных систем, в первую очередь производственных и социальных, существенно зависит от структуры связей и временных

задержек в принятии решений и действиях, которые имеются в системе. Парадигма компьютерного моделирования, при которой для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие параметры во времени, а затем модель, созданная на основе этих диаграмм, имитируется на компьютере, получила название *системная динамика*.

Графическая нотация для моделирования всех компонентов системы и их взаимосвязей делают системную динамику очень удобным инструментом визуального представления всей системы, организации в целостном виде. Системная динамика представляет сегодня парадигму, метод и графический язык для представления моделей сложных систем, а также для их имитационного компьютерного выполнения. Сложные связи и взаимные влияния процессов часто встречаются в бизнесе, экологии, социальных системах, урбанистике и т.п. Системная динамика оказалась очень эффективным методом для представления и анализа проблем динамики организационных систем (таких как анализ рынка, управление проектами, управление цепочками поставок), она дает исследователю понимание эффекта, который производит на систему изменение тех или иных параметров, позволяет сравнить альтернативные решения по управлению системой с выбором наилучшего решения.

Рассмотрим пример разработки типичной системно-динамической модели, представляющей интерес в экономике. Это модель распространения среди населения инноваций и новых продуктов, разработанная Франком Бассом (Frank Bass) в 1969 г. Среди бизнес-

аналитиков она является одной из самых популярных моделей исследования рынка новых продуктов.

Модель представляет динамику процесса превращения потенциальных покупателей нового продукта во владельцев продукта. Разработка модели в соответствии с парадигмой системной динамики происходит так. На первом этапе эксперт выделяет интересующие исследователя количественные характеристики системы, выявляет все факторы, действующие на эти количества, и причинно-следственные соотношения между ними выражает в форме структурной диаграммы зависимостей переменных и параметров.

Системный анализ и системное мышление, инструментом которых является системная динамика, исходят из того, что структура системы определяет ее поведение. Многие важнейшие свойства динамики систем зависят только от взаимодействия компонентов, в частности, от обратных связей в системе, а не от сложности самих компонентов. В системной динамике различают два вида обратных связей: усиливающую (положительную) и уравновешивающую (отрицательную). *Усиливающая* (reinforcing) обратная связь возникает, когда воздействие в системе передается на вход системы (возможно, не непосредственно) и усиливается первоначальное изменение, приводя к еще большим изменениям в том же направлении. Такая обратная связь может привести к бесконтрольному экспоненциальному росту. *Уравновешивающая* (самокорректирующая, balancing) обратная связь, напротив, возникает, когда изменения в системе вызывают уменьшение первоначального воздействия и тем самым ослабляют эффект этого воздействия.

Уравновешивающая обратная связь поддерживает систему в устойчивом состоянии и вызывает сопротивление системы попыткам ее изменения.

Рассмотрим первый этап разработки на основе системного подхода модели распространения нового продукта, предложенной Франком Бассом.

Для построения модели Басс выделил две основные количественные характеристики: потенциальных покупателей (Potential_Adopters) и людей, уже купивших продукт, или владельцев продукта (Adopters). Они являются стоками модели. Далее он определил связь этих количественных характеристик введя понятие потока от потенциальных покупателей к владельцам продукта. Интенсивность этого потока естественно охарактеризовать как интенсивность приобретения ЛЮДЬМИ нового продукта (Adoption_Rate). Но интенсивность покупок не является константой. Люди покупают продукт либо под влиянием рекламы, либо узнав о нем от знакомых, по "сарафанному радио". Эффективность рекламы пропорциональна числу людей, на которых она действует, т. е. числу потенциальных покупателей. Аналогично, эффективность "сарафанного радио" зависит от числа людей, уже купивших продукт. Иными словами, в этой модели должна быть отражена структура взаимных зависимостей характеристик и параметров системы.

В своей модели Басс представил структуру зависимостей характеристик процессов распространения инновационных продуктов от различных факторов. Интенсивность покупок (Adoption_Rate) зависит от нескольких причин. Первая причина — реклама (параметр Adoption_from_Advertizing). Этот параметр зависит от двух факторов:

эффективности рекламы (Advertizing_Effectiveness) и количества потенциальных покупателей. Ясно, что чем больше значение параметра Adoption_from_Advertizing, тем больше покупателей купит продукт, число потенциальных покупателей уменьшится, потому уменьшится и эффект влияния рекламы. Следовательно, влияние этой петли обратной связи на параметр Adoption_from_Advertizing является уравновешивающим (Balancing).

Второй причиной, заставляющей покупателя приобрести продукт, является мнение окружающих об этом продукте. В области анализа рынка эта причина называется Word of Mouth — что можно перевести как "люди говорят" — это просто формализация "сарафанного радио". В модели предполагается, что мнение о продукте высказывают только те люди, которые уже приобрели этот продукт. Параметр Adoption_from_WOM зависит от многих факторов. Во-первых, это число потенциальных покупателей (potential_Adopters). Ясно, что чем больше параметр Adoption_from_WOM, тем больше людей покупают продукт, число потенциальных покупателей уменьшается, и параметр Adoption_from_WOM уменьшается. Поэтому мы имеем вторую петлю уравновешивающей обратной связи.

Другой фактор, который влияет на интенсивность покупок, это число людей, уже купивших продукт (Adopters). Чем больше значение параметра Adoption_from_WOM, тем больше купивших, и, следовательно, этот параметр увеличивается, поскольку больше людей будут распространять свое мнение о продукте. Потому в данной системе взаимозависимости параметров существует третий цикл обратной связи,

который является усиливающим (Reinforcing). Такие петли обратных связей аналитики рынка называют внутренним и внешним влиянием на покупателей.

Составление графического представления зависимостей параметров системы и анализ циклов обратных связей является полезным начальным шагом при анализе системы. Он дает общее представление о структуре зависимостей в системе и тенденциях изменений параметров, а именно того, как изменение (увеличение или уменьшение) одного параметра влияет на изменение другого. Количественное взаимное влияние параметров, или, что то же, конкретные функциональные зависимости параметров определяются с помощью экспертных оценок, сделанных предположений, собранной статистики или экспериментов с реальной системой. Поэтому следующий шаг построения модели после введения накопителей и потоков — это задание конкретных зависимостей одних характеристик системы от других. Именно вид функциональных зависимостей параметров делает модель более или менее адекватной для анализа проблемы, они определяют реальную динамику системы.

Анализ модели может состоять в том, что оценивается чувствительность модели к изменениям тех параметров, которыми может управлять компания, заинтересованная в продаже своего продукта. Например, аналитики компании могут сравнить влияние на динамику модели и интенсивность приобретения человеком нового продукта двух факторов: рекламы и "сарафанного радио". Первый фактор можно увеличить, расширяя круг потенциальных покупателей и повышая эффективность рекламы. Второй фактор напрямую зависит от параметра

Adoption_Fraction, который, конечно, является функцией качества товара. Так с помощью модели можно оценить, куда компании эффективнее вкладывать инвестиции: в рекламу товара, повышая Advertizing_Effectiveness, в расширение функциональности товара (повышая число Potentiai_Adopters) или в качество товара, повышая Adoption_Fraction.

Итак, в потоковых диаграммах системной динамики используется четыре базовых графических объекта: накопители (уровни, переменные состояния) потоки (связи между накопителями, вентили, регулирующие потоки, функциональные зависимости, определяющие взаимное влияние потоков). Накопители обозначаются прямоугольниками, потоки — направленными переходами, а вспомогательные переменные — кружками. Стрелки обозначают причинно-следственные зависимости в модели. *Накопитель* — это параметр, содержащий нечто. *Поток* — это непрерывное перемещение содержимого между накопителями. Накопители и потоки влияют друг на друга через связи, которые могут формировать цепи положительных и отрицательных обратных связей. Для задания влияний параметров используются *вспомогательные переменные*, обозначаемые кружками. Временная задержка моделируется блоками задержки. Параметры-константы в некоторых системах моделирования изображаются ромбиками.

Петли причинных связей отражают влияние некоторого процесса а на процесс в. Процесс в, в свою очередь, может также влиять на а, возможно, через длинную цепочку причинно-следственных связей. Изучение процессов а и в по отдельности невозможно. Только изучение

динамики всей системы со всеми ее связями и временными задержками может привести к корректному пониманию процессов развития системы. Это и составляет основную идею системного мышления. Многие авторы непосредственно связывают системную динамику с системным мышлением в экономике, социальной сфере и экологии.

В сложных моделях системной динамики ясно проявляется одна из главных причин моделирования: непосредственное влияние каждого конкретного процесса на другие процессы можно понять, проанализировать, описать и изобразить графически. Однако взаимное влияние многих процессов человек понять не может, только имитационное моделирование и компьютерный эксперимент дают возможность понять и оценить взаимное влияние каждого процесса на любой другой, даже если они не связаны непосредственно, а также влияние параметров процессов на важные системные характеристики.

Дж. Форрестер пишет, что воспитанная на наблюдении и использовании простых систем, наша интуиция не способна правильно оценить последствия тех или иных корректирующих действий в сложных системах. Простые системы обычно характеризуются только одной петлей обратной связи, имеют только одну основную переменную состояния. Интуитивный урок, который получают люди при использовании таких систем, тот, что причина и следствие тесно связаны и ясны. Более того, искусственные технические системы, используемые людьми, обычно создаются такими, чтобы управление ими отвечало тому же требованию. Например, хотя автомобиль отнюдь не является простой системой, специально разработанная система управления им обеспечивает простую

и ясную причинно- следственную связь управляющих воздействий и характеристик движения. Многие природные и искусственные системы, однако, не обладают такими свойствами.

В сложных системах существует несколько петель обратных связей параметров как положительных, так и отрицательных, и причины изменений состояний системы могут лежать далеко от следствий как во времени, так и в пространстве. Часто за причину некоторого эффекта в таких системах люди принимают то, что Дж. Форрестер называет "симптомом" — промежуточную характеристику системы, непосредственно связанную с этим эффектом в длинной цепочке причинно-следственных связей. Очевидно, что для коррекции эффекта воздействие на симптом неэффективно, оно обычно не может устранить причину возникновения эффекта. Задача моделирования — выявить реальные причинные зависимости в сложных системах и найти схемы управления, подавляющие нежелательное развитие событий.

3 Разработка моделей в AnyLogic

В сложных системах существует несколько петель обратных связей параметров как положительных, так и отрицательных, и причины изменений состояний системы могут лежать далеко от следствий как во времени, так и в пространстве. Часто за причину некоторого эффекта в таких системах люди принимают то, что Дж. Форрестер называет "симптомом" — промежуточную характеристику системы, непосредственно связанную с этим эффектом в длинной цепочке причинно-следственных связей. Очевидно, что для коррекции эффекта воздействие на симптом неэффективно, оно обычно не может устранить

причину возникновения эффекта. Задача моделирования — выявить реальные причинные зависимости в сложных системах и найти схемы управления, подавляющие нежелательное развитие событий.

3.1 Лабораторная работа 1. Моделирование динамики физического объекта

Создадим первую динамическую модель в AnyLogic, описывающую физическое перемещение пластмассового шарика вдоль вертикальной оси. В данной модели изучается поведение шара при столкновении с землей с потерей энергии.

Алгоритм выполнения работы следующий:

1. Создайте новый проект в AnyLogic с помощью команды **Файл → Создать → Модель**. Укажите имя модели (**DynamicBall**), задайте путь для размещения модели, а также установите переключатель на позицию «**Начать создание модели «с нуля» (будет создан чистый холст презентации)**», как показано на рисунке 3.1. Остальные параметры модели оставьте неизменными. Нажмите «**Готово**».

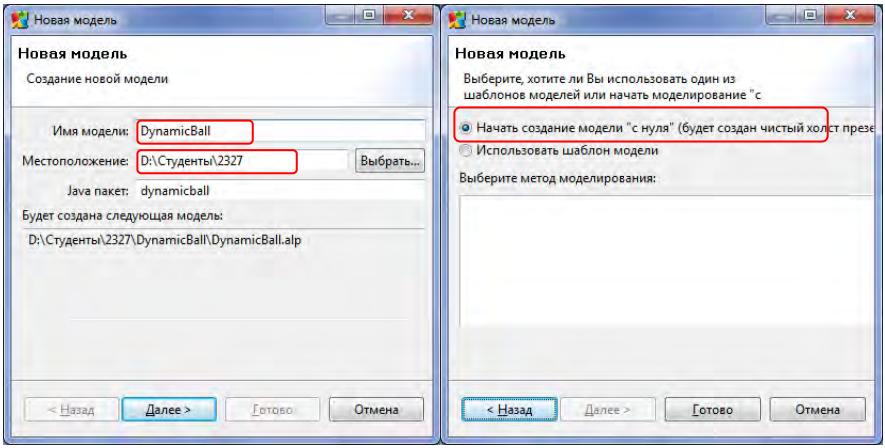


Рис. 3.1. Создание новой модели в AnyLogic

2. Переименуйте текущий класс **Main** в **Root**, используя окно свойств для данного объекта. Данный класс будет использоваться для хранения экземпляров класса Ball. Далее, создайте новый класс активного объекта (используя команду меню **Файл → Создать → Класс активного объекта**) и назовите его **Ball**. Данный класс используется для описания объекта Шар (**Ball**), а также для хранения его параметров и алгоритма действий. Перетащите объект **Ball** из окна проекта в окно структуры объекта **Root** (рисунок 3.2).

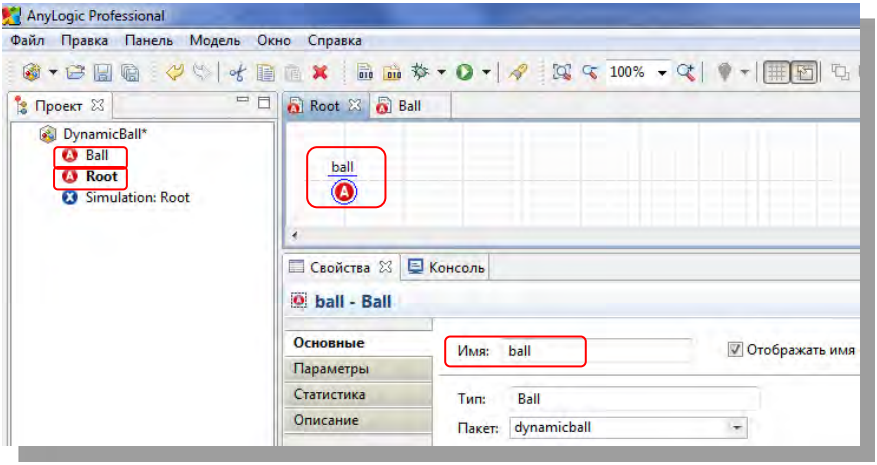


Рис. 3.2. Вставка объекта Ball в объект Root

3. Введем основные параметры модели. Для этого выделите класс **Ball** (путём выбора соответствующей вкладки) и в окно его структуры перетащите из вкладки **Основная** палитры объектов пять параметров, обозначенных значком . Введите следующие параметры для активного объекта **Ball**:

Таблица 1.1. – Параметры модели DynamicBall

Наименование параметра	Тип данных параметра	Значение по умолчанию	Описание
g	double	9.8	Ускорение свободного падения
r	double	20	Радиус мяча
k	double	0.02	Коэффициент уменьшения скорости мяча при каждом отскоке
x0	double	100	Начальные значения координат
y0	double	200	

4. Введем переменные модели. Для этого выделите класс **Ball** (путём выбора соответствующей вкладки) и в окно его структуры перетащите из вкладки **Основная** палитры объектов одну простую переменную, обозначенную значком , и из вкладки **Системная динамика** три накопителя, обозначенных значком . Введите следующие переменные для активного объекта **Ball**:

Таблица 1.2. – Переменные модели DynamicBall

Наименование переменной	Тип данных переменной	Тип переменной	Значение по умолчанию	Формула расчета d(X)/dt	Описание
x	double		x0	vx	Координаты мяча
y	double		y0	vy	
vx	double		0		Скорости мяча
vy	double		0	-g	

5. Введем в модель диаграмму состояний для описания поведения активного объекта **Ball**. Для этого используем объекты палитры **Диаграмма состояний**: начало диаграммы состояний , состояние и переход . Создайте диаграмму состояний в окне структуры объекта **Ball**, как показано на рис. 3.3. Назовите указанное по умолчанию состояние state

именем Movement в окне свойств.

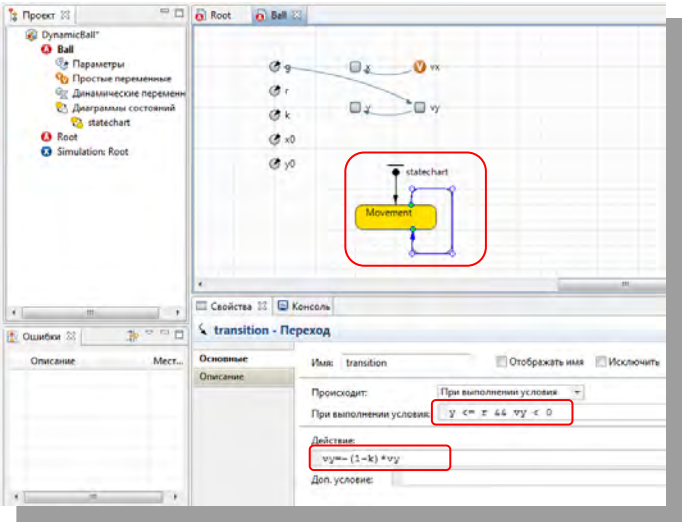


Рис. 3.3. Создание диаграммы событий

6. Таким образом, диаграмма действий модели прыгающего мяча состоит из одного состояния с именем **Movement** и одного перехода. Переход срабатывает при наступлении события касания поверхности земли при движении мяча вниз. Условие наступления данного события можно записать выражением:

```
y <= r && vy < 0
```

Это выражение означает, что вертикальная координата **y** центра мяча с радиусом **r** отстоит от поверхности на значение радиуса **r** и при этом скорость мяча **vy** направлена вниз, в сторону уменьшения координаты **y**. При наступлении данного события мяч отскакивает, т.е. его скорость меняет свой знак, уменьшаясь при этом на долю k, показывающую потерю энергии при отскоке. Таким образом, в поле **Действие** необходимо

записать следующее:

```
vy=-(1-k)*vy
```

7. Анимация в AnyLogic создается в виде динамических графических объектов, которые дают возможность наглядно представить динамику моделируемой системы, т. е. поведение ее во времени. Основная идея здесь состоит в том, что параметры элементов анимационной картинки (для круга это его координаты центра, радиус, цвет и т. п.) связываются с переменными и параметрами модели. Изменение переменных модели во времени заставляет изменяться во времени графический образ, что позволяет наглядно представить динамику моделируемой системы с помощью динамически меняющейся графики.

На свободном поле в окне структуры объекта **Root** нарисуйте круг инструментом **Овал** палитры **Презентация**. Цвет заливки укажите синим (**blue**). Перейдите на вкладку «Динамические» в окне свойств объекта **Овал**. В полях ввода **Радиус X** и **Радиус Y** укажите параметр объекта ball:

```
Радиус X: ball.r
Радиус Y: ball.r
```

В полях ввода **X** и **Y** укажите соответственно переменные **x** и **y** объекта **ball** следующим образом:

```
X: ball.x
Y: -ball.y
```

Таким образом, изменение данных переменных будет вызывать перемещение центра графического образа мяча на анимационной картинке при работе модели.

8. Нарисуйте рамку вокруг мяча, как показано на рис. 3.4., инструментом **Прямоугольник** палитры **Презентация**. Нижняя граница рамки должна совпадать с той линией, касаясь которой шар отпрыгивает обратно.

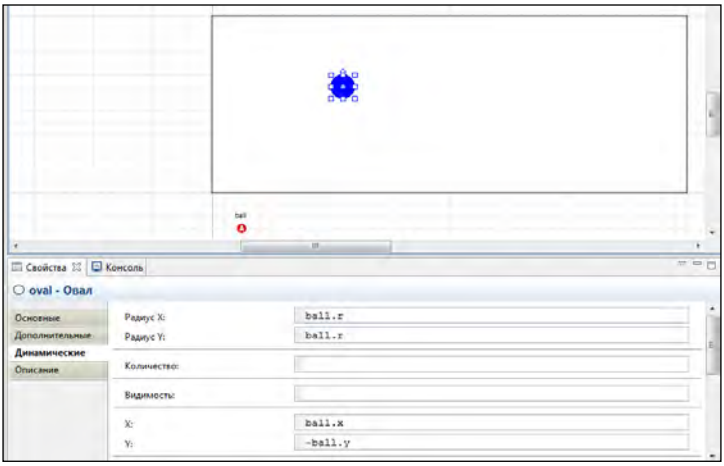


Рис. 3.4. Указание динамических свойств объекта

9. На свободном поле внутри рамки инструментом **Текст** палитры Презентация составьте краткое описание модели, как показано на рис. 3.5. вставьте бегунки (инструмент **Бегунок** палитры **Элементы управления**) для изменения ускорения свободного падения (параметр g), потери энергии (параметр k) и радиуса шара (параметр r). Для каждого из этих бегунков в палитре свойств установите флажок **Связать с** и укажите значения:

Для первого бегунка: *ball.g*

Для второго бегунка: *ball.k*

Для третьего бегунка: *ball.r*

Для каждого из бегунков установите соответствующие минимальное и максимальное значения: для первого бегунка: *-10* и *10*, для второго бегунка *0* и *5*, для третьего бегунка *10* и *100*. Подпишите каждый бегунок. Результат выполнения данного пункта показан на рис. 3.5.

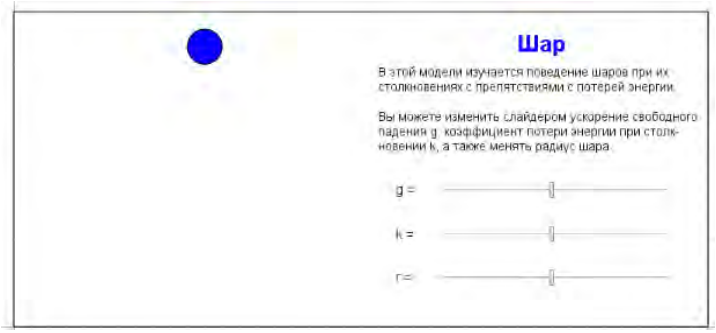


Рис. 3.5. Разработка анимации модели

10. Для запуска модели щелкните кнопку на панели инструментов. Этим действием запустится компилятор, который построит исполняемый код проекта на языке *Java*, оттранслирует его и затем запустит модель на исполнение. При этом первоначально откроется страница настроек эксперимента. **Нажмите на кнопку Запустить модель и открыть презентацию класса Main** для выполнения имитационной модели.

11. В открывшемся окне кроме движущегося изображения мяча можно видеть текстовый комментарий и бегунки – подвижные указатели для изменения параметров модели во время ее выполнения. Двигая слайдеры, можно менять в модели три параметра – ускорение свободного падения g, долю k потери скорости прыгающим мячом при каждом отскоке и радиус шара r. Изменение параметров позволяет исследовать

поведение модели в различных условиях – это и есть компьютерный эксперимент.

Для проведения компьютерных экспериментов необходимо использовать (кроме уже известной кнопки компиляции и запуска модели на выполнение) также следующие кнопки: паузы – , разрушения скомпилированной модели и возврата к странице настроек эксперимента – , переключение между виртуальным и реальным временем исполнения модели – , увеличение/уменьшение масштаба времени - x1 .

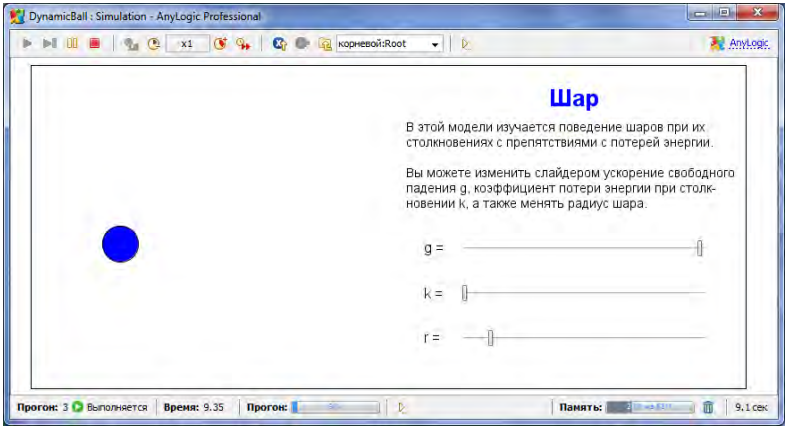


Рис. 3.6. Выполнение модели

Доработка модели

Выполним некоторые упражнения с моделью *DynamicBall*, которые дадут, некоторое представление о средствах разработки моделей в AnyLogic.

Для большей наглядности дополним анимационное представление поведения мяча так, чтобы при отскоке мяча его цвет на мгновение

изменялся на красный. Для этого нужно зафиксировать момент отскока (запомнить значение момента времени наступления этого события) и установить красным цвет шара в анимации на небольшой интервал времени, следующий за этим моментом.

Алгоритм выполнения работы следующий:

1. Введите переменную **tBounce**, которая будет фиксировать момент отскока. Для этого сделайте активным окно структуры активного объекта **Ball** и перенесите внутрь структурного окна объект **Накопитель** палитры **Системная динамика**. В поле **Имя** открывшегося окна свойств этого накопителя введите *tBounce*, а в поле **Начальное значение** введите *-1* (рис. 3.7).

Для того чтобы переменная **tBounce** фиксировала момент отскока, нужно значение текущего времени в модели при наступлении события "отскок" запомнить в этой переменной. За наступлением данного события следит диаграмма событий, поэтому выберите переход и в поле Действие перехода добавьте запись:

```
tBounce = time();
```

При каждом вызове функция `time` даёт текущее значение модельного времени.

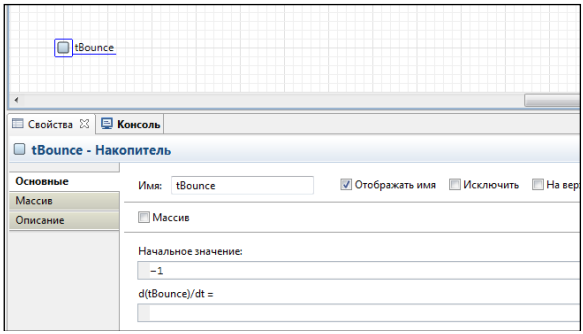


Рис. 3.7. Введение новой переменной tBounce

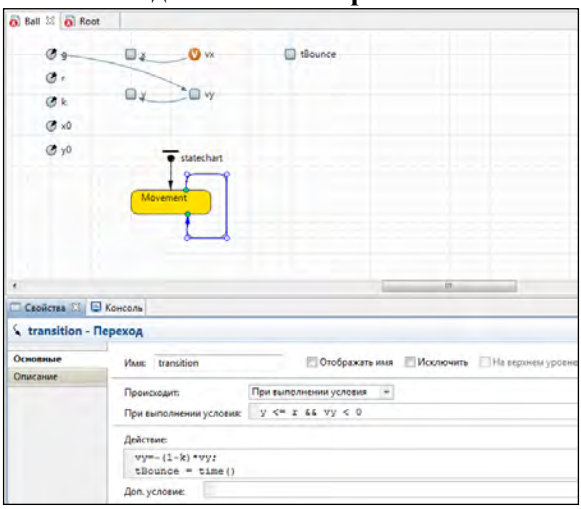


Рис. 3.8. Запоминание времени в момент отскока

Переменная **tBounce** имеет начальное значение `-1` и при работе модели хранит значение момента времени последнего отскока.

2. Для того, чтобы каждый раз при отскоке мяча его цвет изменялся на красный (скажем, в течение 0.1 с), нужно установить в поле **Цвет заливки** вкладки **Динамические** графического изображения в окне **Root** мяча в окне анимации динамическое значение цвета:

```
time() < ball.tBounce+0.1? Color.red: Color.blue
```

Это условное выражение устанавливает цвет заливки изображения мяча **ball** красным в течение 0.1 с после каждого отскока.

3. Запустите модель и проверьте результат.

4. Введём в модель второй мяч. Для этого сделайте активным окно структурной диаграммы **Root** и перенесите в него еще один экземпляр мяча, щелкнув левой кнопкой мыши сначала по имени **Ball** в окне проекта, а потом перетащим курсор в окно структурной диаграммы **Root**. Появившийся объект автоматически получит имя **ball1**. При этом снизу откроется окно свойств выделенного объекта (нового экземпляра мяча), в котором установлены те значения параметров мяча, которые были определены для активного объекта **Ball**.

5. Установите начальные значения **x0** и **y0** координат **x** и **y** нового мяча равными 200 и 300.

6. Чтобы движение нового мяча отобразить в анимации, в окне анимации продублируйте изображение мяча. Для чего нажмите левой кнопкой мыши на изображении мяча и при нажатой клавише **<Ctrl>** перенесите это изображение в другое место поля анимации. Для нового изображения круга в правой части окна редактора появится окно свойств, в котором динамические значения параметров (координат и цвета) связаны с характеристиками шара ball. Их нужно связать с новым объектом — шаром с именем **ball1**. Иными словами, вместо **ball.r**, **ball.x**, **ball.y** и **ball.tBounce** в соответствующих полях нужно поставить **ball1.r**, **ball1.x**, **ball1.y** и **ball1.tBounce**.

7. Запустите модель. В модели теперь будут имитироваться независимые движения двух шаров (рис. 3.9).

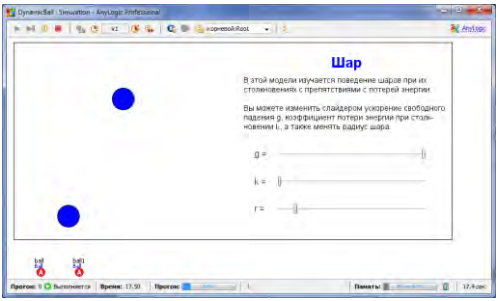


Рис. 3.9. Введение нового экземпляра мяча в модель

Произвольные перемещения мяча

В данной модели мячи движутся вертикально, отталкиваясь от поверхности с координатой 0. Это происходит потому, что начальная скорость мячей по координате **x** равна 0. Если мы изменим начальные скорости мячей по этой координате, сделав их, например, случайными, нам необходимо задать также, как мячи будут себя вести при встрече с потолком и вертикальными стенками.

Вернемся снова к экспериментам с одним мячом. Удалите из окна структурной диаграммы объекта **Root** объект **ball1**, а из окна анимации удалите шар с именем **oval1**, отображавший движение шара **ball1**.

Сделаем сначала случайными начальные значения скоростей **vx** и **vy** мяча. Активизируйте окно структурной диаграммы активного объекта **Ball**, выделите переменную **vx** и в поле Начальное значение этой переменной замените значение 0 на значение `uniform(-100, 100)`. Тем самым начальная скорость по координате **x** у различных экземпляров активного объекта Ball будет выбираться случайно из диапазона (-100, +100) метров в секунду как реализация случайной величины, равномерно

распределенной в этом диапазоне. То же самое сделайте для переменной **vy**.

Для учета отталкивания мяча от потолка нужно событие встречи препятствия мячом на переходе диаграммы состояний изменить. Размеры поля, в котором двигаются мячи, установлены 360x800(м). В поле **При выполнении условия** окна свойств перехода диаграммы состояния активного объекта **Ball** выражение:

```
y <= r && vy < 0
```

следует дополнить:

```
y <= r && vy < 0 || y >= 360-r && vy > 0
```

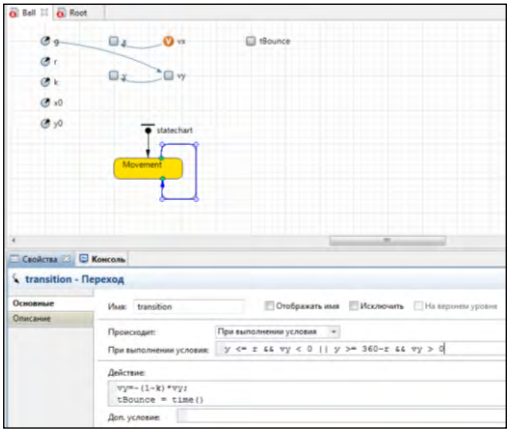


Рис. 3.10. Изменение поведения мяча

Действие, которое выполняется и в том, и в другом случае, будет тем же: изменение направления скорости **vy** с частичной ее потерей.

Для того чтобы учесть отталкивание мяча от вертикальных стен, нужно учесть это событие в диаграмме состояний введением дополнительного перехода. Сделайте окно диаграммы состояний

активным и добавьте к состоянию **Movement** дополнительный переход, используя объект **Переход** палитры **Диаграмма состояний** (рис. 3.11).

В появившемся справа окне свойств этого перехода в поле **Происходит** нужно выбрать вариант **При выполнении условия**, в появившемся поле **При выполнении условия** следует вставить условие наступления события касания мяча о вертикальную стенку:

```
x <= r && vx < 0 || x >= 800-r && vx > 0
```

а в поле **Действие** установить действия изменения направления составляющей **vx** скорости мяча и моментальное изменение цвета мяча при столкновении:

```
vx=-(1-k)*vx;

tBounce = time();
```

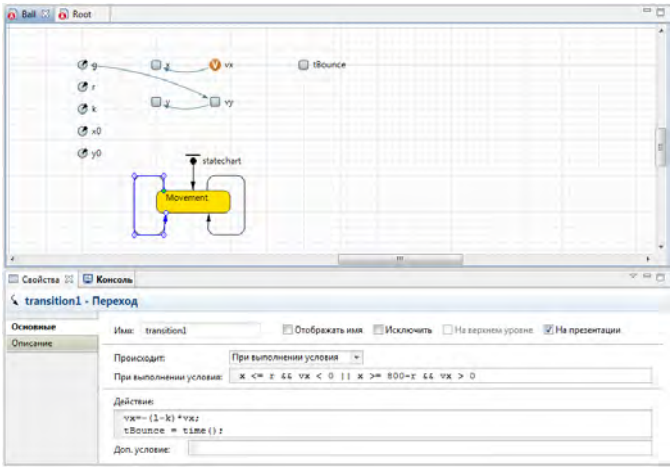


Рис. 3.11. Создание второго перехода

Запустите модель на исполнение. Используя слайдеры, можно изменять радиус мяча, коэффициент потери скорости при встрече с

препятствием, ускорение свободного падения, превращая мяч в воздушный шар при g<0 или в биллиардный шар при g=0.

3.2 Лабораторная работа 2. Моделирование производства

Создадим модель производства в AnyLogic, описывающую функционирование некоторого цеха автомобильного завода. Моделируемый цех выполняет сборку колёс из автомобилей из двух частей – покрышек и дисков – и отправляет готовую продукцию в сервисные центры для продажи.

Цех работает следующим образом:

- Покрышки и диски поступают в цех и транспортируются конвейерами к роботу сборки.
- Робот сборки собирает колесо с помощью шиномонтажных работ, используя покрышки и диски.
- Собранное автомобильное колесо транспортируется конвейером к зоне упаковки, где оно упаковывается рабочими в коробку.
- Каждые 50 готовых колёс составляют партию товара, которая забирается с завода на грузовике.

Для начала создадим простую модель, которая будет моделировать то, как автомобильные покрышки поступают в заводской цех и транспортируются по конвейеру к месту сборки.

Алгоритм выполнения работы следующий:

1. Создайте новый проект в AnyLogic с помощью команды **Файл →**

Создать → Модель. Укажите имя модели (**Supply Chain**), задайте путь для размещения модели (**D:\Студенты\2327**), а также установите переключатель на позицию «**Начать создание модели «с нуля» (будет создан чистый холст презентации)**». Остальные параметры модели оставьте неизменными. Нажмите «**Готово**».

2. Откройте палитру **Enterprise Library**. Перетащите объект Source (генератор заявок) из палитры на графическую диаграмму. Объект Source создает новые заявки и обычно используется в качестве начальной точки диаграммы процесса. После перетаскивания объекта в графический редактор его имя выделится во встроенном текстовом редакторе. Введите здесь новое имя объекта: **sourceTires** (рис. 3.12).

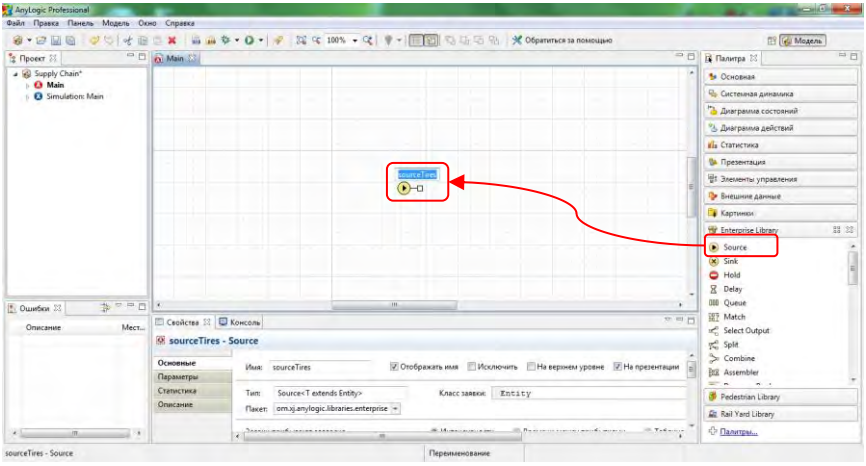


Рис. 3.12. Вставка объекта Source на диаграмму

3. Продолжим создание диаграммы процесса из объектов библиотеки **Enterprise Library**. Добавьте объект **Queue** (очередь заявок), назовите его **tires**. Очередь добавляется для того, чтобы хранить

поступившие покрышки до тех пор, пока они не будут помещены на конвейер. Добавьте объект **Conveyor** (конвейер), назовите его **conveyorTires**. В нашей модели он будет представлять конвейер, транспортирующий автомобильные покрышки. Добавьте объект **Sink** для уничтожения заявок. Объект **Sink** обычно используется в качестве конечной точки диаграммы процесса. Результат выполнения данного пункта показан на рис. 3.13.

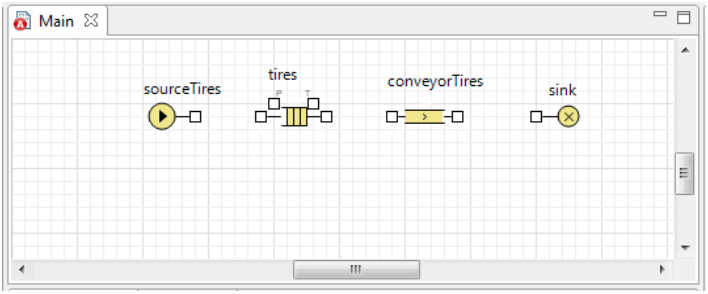


Рис. 3.13. Вставка остальных объектов на диаграмму

4. Соедините порт объекта **sourceTires** с левым портом объекта **Tires**. Если Вы правильно соедините порты, то после соединения конечные точки соединителя должны будут подсветиться зелеными точками. Соедините порты объектов диаграммы процесса, как показано на рис. 3.14. Соединяя порты объектов диаграммы процесса, Вы задаете путь следования заявок. При этом все объекты необходимо соединять слева направо – то есть правые порты объектов с левыми портами последующих объектов. Это связано с тем, что у объектов **Enterprise Library** есть входные и выходные порты, и входные порты можно соединять только с выходными.

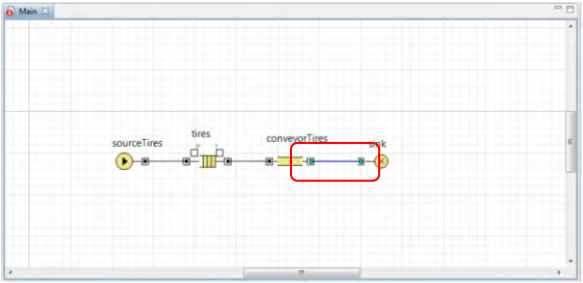


Рис. 3.14. Соединение объектов на диаграмме

5. Вы можете заметить звездочку рядом с элементом модели в дереве моделей в панели **Проекты** (рис. 3.15). Это значит, что в модели есть несохраненные изменения. Сохранить модель можно с помощью кнопки панели инструментов **Сохранить** .

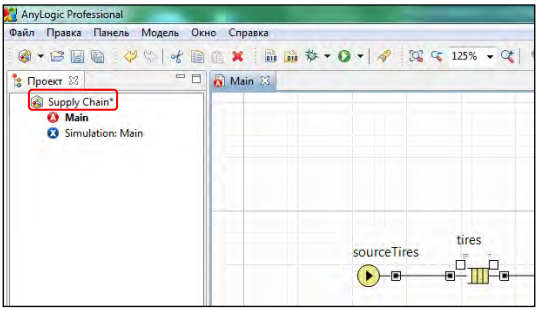


Рис. 3.15. Пример несохраненного проекта

6. Добавьте простую анимацию конвейера и зоны хранения автомобильных покрышек. Рисование анимации модели необходимо начать с прямоугольника – области, в которой будут храниться поступившие в цех автомобильные покрышки.

Откройте палитру **Презентация**. Перейдите в режим рисования,

сделав двойной щелчок по элементу **Прямоугольник** в **Палитре**. Нарисуйте прямоугольник движением мыши с нажатой левой кнопкой. Измените свойства только что нарисованного прямоугольника в панели **Свойства**: назовите прямоугольник **shapeTireStorage**. Сделайте прямоугольник прозрачным (установите в поле **Цвет заливки** значение **Нет заливки**), измените цвет контура прямоугольника на синий (рис. 3.16).

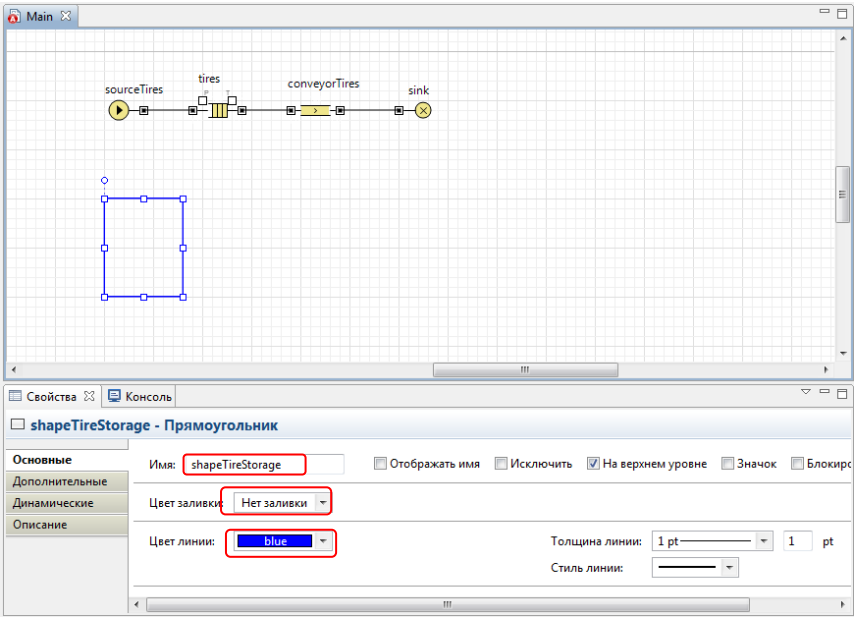


Рис. 3.16. Создание области хранения покрышек

7. Нарисуйте ломаную, которая будет обозначать конвейер на анимации модели: двойным щелчком мыши по элементу **Ломаная** в Палитре активируйте его режим рисования; нарисуйте ломаную так, как показано на рис. 3.17. Назовите ломаную **shapeConveyorTires** и измените

ее цвет.

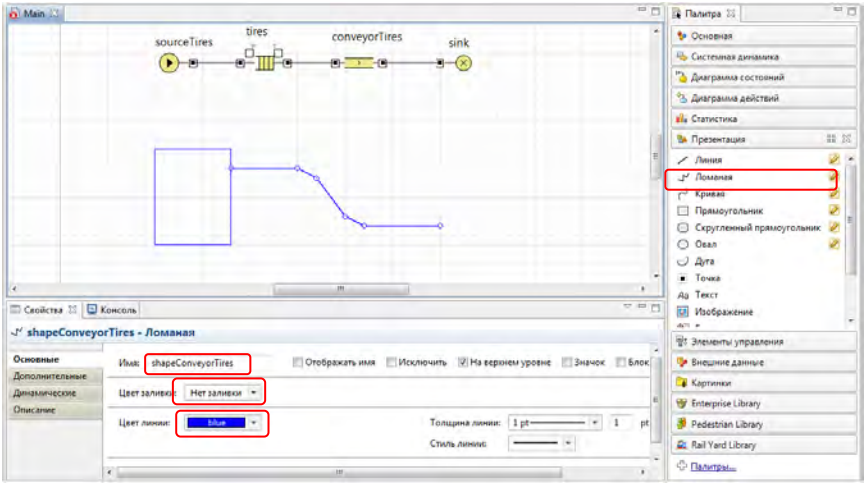


Рис. 3.17. Создание конвейера для транспортирования покрышек

8. Выделите объект **tires**. Расширьте область панели **Свойства**, перетащив ее границу вверх. Измените свойства объекта (рис. 3.18).

- установите флажок **Максимальная вместимость**, чтобы разрешить одновременное нахождение в очереди максимально возможного количества заявок.
- Задайте **shapeTireStorage** в качестве **Фигуры анимации**.
- Выберите мешок в качестве **Типа анимации**. При данном типе анимации заявки отображаются в случайных позициях внутри заданного прямоугольника.

9. Выделите объект **conveyorTires**. Задайте в качестве Фигуры анимации ломаную **shapeConveyorTires**.

10. Запустите модель. Ускорьте выполнение модели Выделите

объект **conveyorTires**. Задайте в качестве Фигуры анимации ломаную **shapeConveyorTires**.

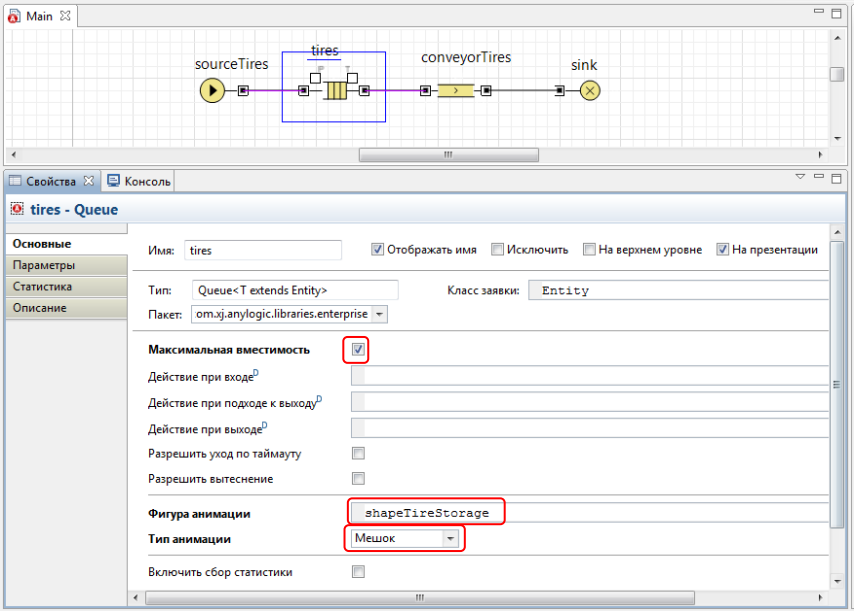


Рис. 3.18. Задание анимации для объекта tires

Продолжим создание модели. На этом этапе выполним следующие шаги:

- Добавим источник дисков и конвейер, ведущий от него к роботу сборки.
- Добавим и самого робота. Здесь будет завершаться процесс сборки путем соединения автомобильное покрышки с диском.
- Нарисуем картинки, обозначающие детали колеса, чтобы сделать анимацию более наглядной.

- Сконфигурируем объекты диаграммы процесса реальными значениями параметров: зададим длину конвейера, скорость, расстояние между движущимися по нему коробками и т.д.

Алгоритм выполнения работы следующий:

1. Нарисуйте кружком анимации ту зону робота сборки, в которую помещаются колеса. Назовите фигуру **shapeTireAtRobot** (рис. 3.19).

2. Нарисуйте прямоугольник, который будет представлять область хранения дисков, поступивших в заводской цех. Создайте копию ранее нарисованного прямоугольника **shapeTireStorage**, перетащив его в сторону с нажатой клавишей **Ctrl**. Назовите получившийся прямоугольник **shapeDiskDtorage** и измените его размер так, как показано на рис. 3.19.

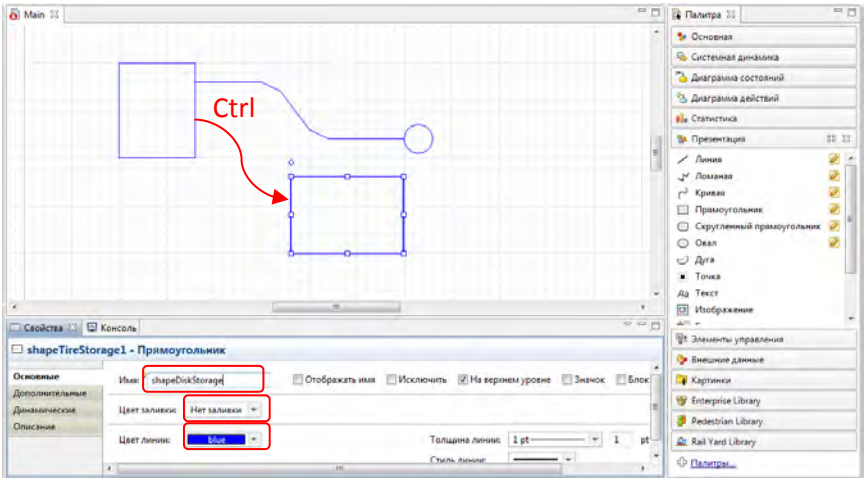


Рис. 3.19. Создание новых объектов анимации

3. Нарисуйте еще два кружка, которые будут обозначать на презентации те области сборочного робота, где располагаются диски и

готовые колёса в сборе (рис. 2.9). Эту операцию можно сделать, также перетащим имеющийся кружок при нажатой клавиши **Ctrl**. Назовите эти объекты соответственно **shapeDiskAtRobot** и **shapeAssembly**.

4. Создайте переменную из панели **Основная**, которая будет задавать соотношения между пикселями презентации и метрами моделируемого объекта. В свойствах переменной сбросьте флажок **На презентации**. Таким образом мы сообщаем AnyLogic, что этот элемент не должен показываться на презентации во время запуска модели. Задайте *10* в качестве **Начального значения** переменной, поскольку в нашей модели одному метру будет соответствовать 10 пикселей презентации (рис. 3.20).

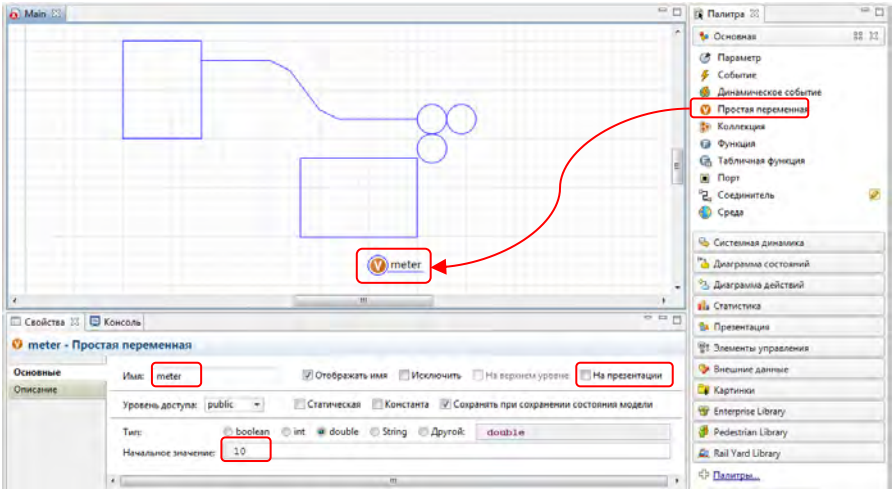


Рис. 3.20. Введение переменной в модель

5. Сейчас детали колеса отображаются на анимации в виде маленьких кружков. Нам же необходимо нарисовать для каждой детали

свою картинку, чтобы мы могли отличать их. Нарисуем автомобильную покрышку, диск и само колесо следующим образом:

- Подвиньте холст, «перетаскивая» его с нажатой правой кнопкой мыши. Вы увидите тонкую линию, обозначающую левую границу окна презентации. Фигурки деталей необходимо рисовать слева от этой линии, чтобы они не попали в видимую часть презентации во время выполнения модели.
- Нарисуйте два круга – один закрашенный в черный, другой в белый. Измените их размеры на странице свойств **Динамические**: поставьте оба радиуса белого круга равным 5, оба радиуса чёрного круга 8. Разместите их один над другим, выделите оба объекта и в вызванном контекстном меню выберите команду **Группировка → Создать группу**. Созданную группу назовите **shapeTire**.
- Нарисуйте один круг с радиусом в 5 точек. Нарисуйте две пересекающиеся прямые внутри круга. Выделите все три фигуры и и в вызванном контекстном меню выберите команду **Группировка → Создать группу.** Созданную группу назовите **shapeDisk.**
- Нарисуйте фигуру, которая включает в себя как внешний чёрный круг, так и внутренний круг с пересекающимися прямыми. Сгруппируйте фигуру и назовите ее **shapeWheel**. Такая фигура будет анимировать готовое колесо в сборке (рис. 3.21).

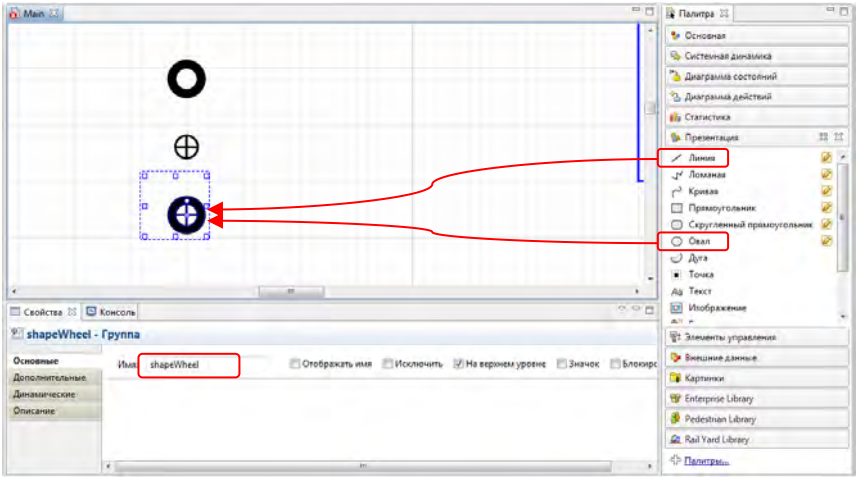


Рис. 3.21. Рисование фигурок заявок

6. Измените свойства объекта **sourceTires**. Выберите **shapeTire** в качестве фигуры анимации автомобильных покрышек (в поле **Фигура анимации заявки**). Установите флажок **Разрешить вращение**, чтобы анимации заявок могли поворачиваться согласно направлению их движения.

7. Измените свойства объекта **conveyorTires**. Пусть длина конвейера задаётся длиной его фигуры анимации (выберите в поле **Длина задается** значение **Согласно пути**). Задайте минимальное расстояние между двумя соседними заявками, движущимися по конвейеру. Введите *2*meter* в поле **Расстояние между заявками**, где **meter** – это введенная ранее переменная, задающая соотношение между пикселями презентации и метрами моделируемого пространства. Задайте скорость, с которой будет двигаться конвейер, в поле **Скорость**: *0.5*meter/second()*.

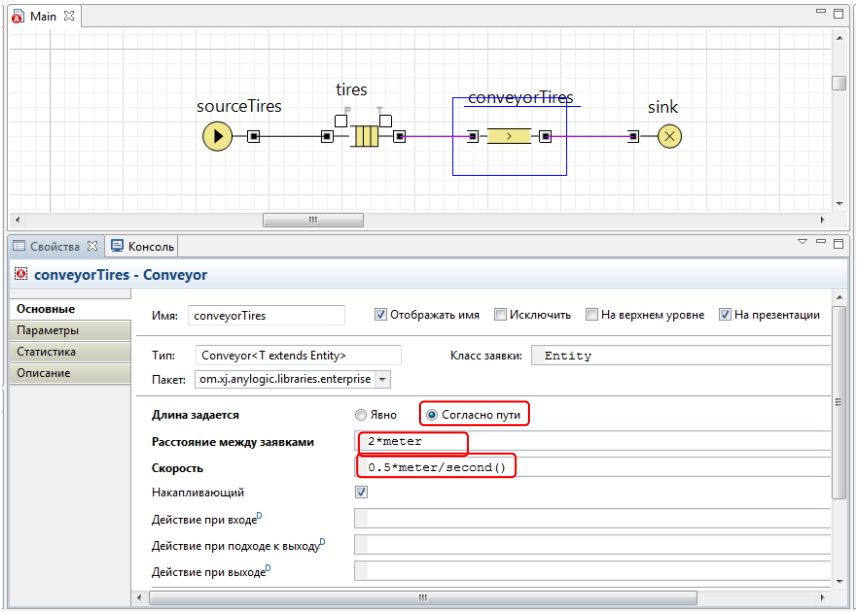


Рис. 3.22. Настройка объекта conveyorTires

8. Добавьте в диаграмму процесса еще два объекта:

- Объект **Source** для моделирования поступления дисков.
- Объект **Queue** для моделирования хранилища дисков.

Создайте эти объекты путём клонирования объектов **sourceTires** и **tires**.

Назовите только что созданный объект **Source** именем **sourceDisks**. Задайте **shapeDisk** в качестве **Фигуры анимации заявок**, создаваемых этим объектом (фигура в виде автомобильного диска).

Назовите созданный объект **Queue** именем **disks** и задайте **shapeDiskStorage** в качестве его **Фигуры анимации** с помощью мастера подстановки кода (комбинация клавиш **Ctrl+Пробел**).

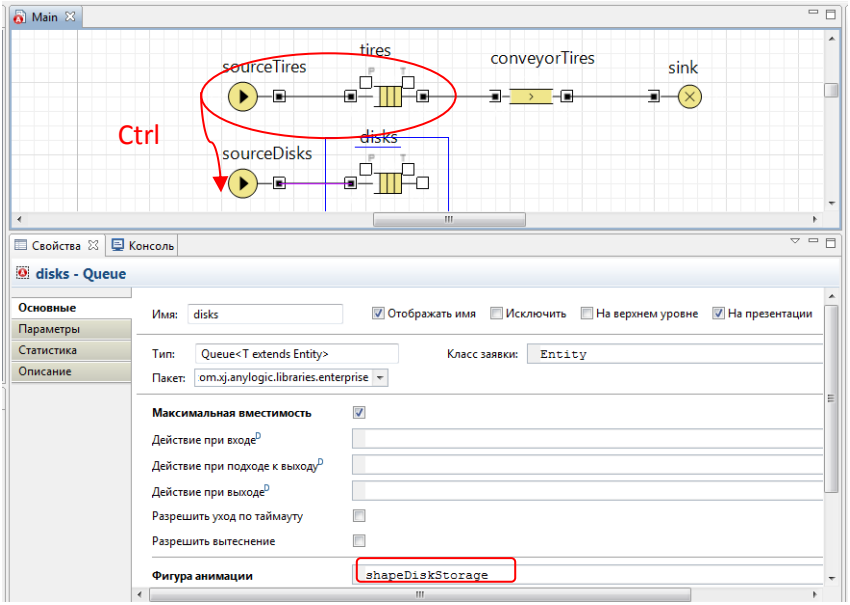


Рис. 3.23. Создание объектов для генерации дисков

9. Добавьте объект **Assembler** и соедините его с другими объектами, как показано на рис. 3.24. Объект **Assembler** собирает одну заявку из нескольких, поступающих во входные порты этого объекта. Используйте фигуру **shapeWheel** ( ) в качестве фигуры анимации заявок, собранных этим объектом. Укажите в поле **Количество ресурсов** значение *0*. Задайте фигуры анимации для очередей, ведущих к первым двум входным портам, и для операции сборки:

Время задержки: *minute()*

Фигура анимации (delay): *shapeAssembly*

Тип анимации (delay): *Одиночная*

Фигура анимации (queue 1): *shapeTireAtRobot*

Тип анимации (queue 1): *Одиночная*

Фигура анимации (queue 2): *shapeDiskAtRobot*

Тип анимации (queue 2): *Одиночная*

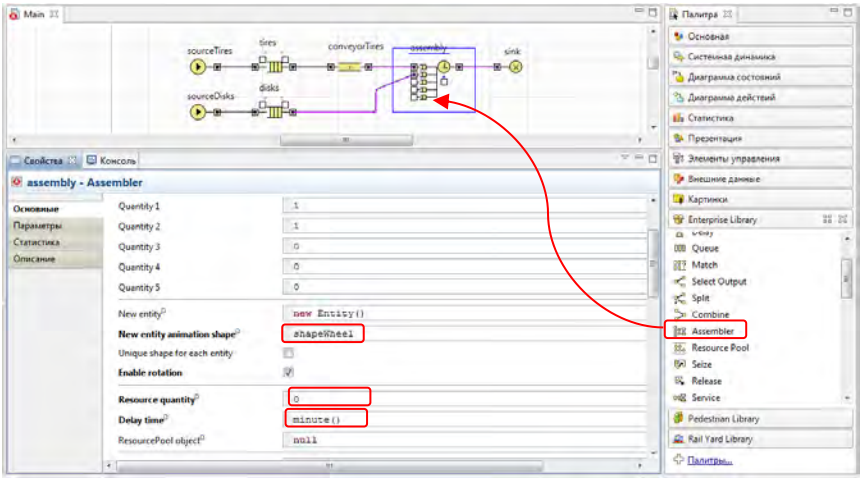


Рис. 3.24. Создание объекта Assembler

10. Измените настройки запуска модели. Для этого выделите элемент **Simulation** в окне Проект, перейдите на закладку **Модельное время** и в поле **Остановить** выберите значение **Нет**. Выбрав Нет из

списка Остановить, мы делаем так, что моделирование будет проводиться до тех пор, пока мы сами его не остановим.

11. Запустите модель на выполнение. Вы увидите анимацию процесса сборки.

3.3 Лабораторная работа 3. Моделирование производства (продолжение)

Промоделируем далее линию упаковки, на которой готовые товары будут запаковываться в коробки. Линия упаковки будет включать в себя собственно зону упаковки и ведущий к ней конвейер. Пусть при этом упакованные товары помещаются в зону погрузки. Каждые 50 единиц товара образуют новую партию и увозят с завода.

Алгоритм выполнения работы следующий:

1. Нарисуйте с помощью еще пяти фигур конвейер, ведущий к зоне упаковки, а также саму зону упаковки и зону погрузки. Ломаную **shapeMoveToPackaging** нарисуйте *слева направо*. Назовите фигуры так, как показано на рис. 3.25.

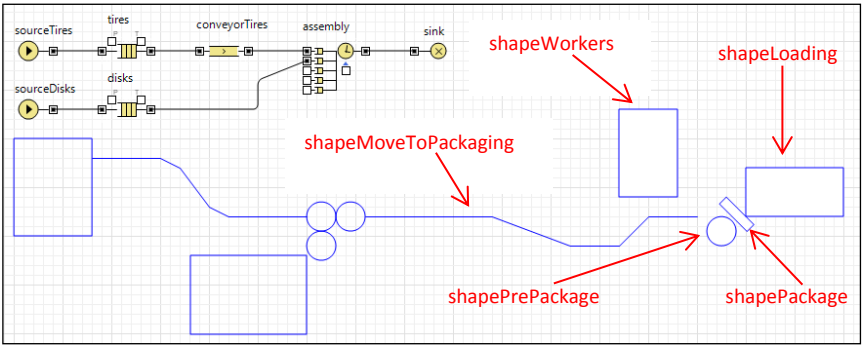


Рис. 3.25. Добавление новых элементов анимации

2. Добавьте картинку **Коробка** из палитры **Картинки**. Назовите эту картинку **pictureBox**. Для того, чтобы упростить рисование небольших фигур, увеличьте масштаб отображения диаграммы до 400%. Измените размер картинки так, чтобы она занимала приблизительно 3x3 ячейки (рис. 3.26).

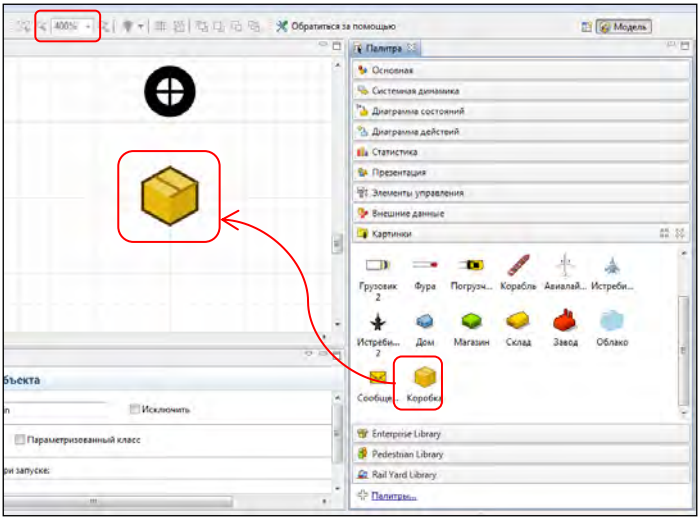


Рис. 3.26. Анимация упакованных колёс

3. Добавьте объект **Conveyor** путём клонирования объекта **conveyorBodies**, чтобы промоделировать конвейер, ведущий к зоне упаковки. Измените свойства объекта: задайте **Расстояние между заявками** *1.2*meter*; укажите имя фигуры, которая будет отображать конвейер на анимации (играть роль пути для анимаций движущихся по конвейеру заявок): **shapeMoveToPackaging**.

4. Добавьте объект **Service** на графическую диаграмму. Этот объект

Service будет моделировать упаковку колеса в коробку. На данном этапе эта операция будет выполняться без привлечения каких бы то ни было ресурсов – их мы добавим на следующей фазе. Задайте время, требуемое на упаковку (поле **Время задержки**): *triangular(40,50,120)*second()*. Напишите *entity.setShape(pictureBox);* в поле **Действие при выходе**. Здесь мы задаем картинку коробки в качестве фигуры анимации для заявок, покинувших этот объект **Service**. Задайте вместимость очереди равной 1. Мы полагаем, что в буферной зоне будет ожидать сборки еще одно колесо. Установите следующие значения полей для анимации зоны упаковки:

- **Фигура анимации (queue)**: *shapePrePackage*
- **Тип анимации (queue)**: *Одиночная*
- **Фигура анимации (delay)**: *shapePackage*
- **Тип анимации (delay)**: *Мешок*

5. Добавьте объект **Batch**. Этот объект будет моделировать погрузку партии товара на грузовик. Говоря в терминах объекта, он создает партию (грузовик) из набора исходных заявок (коробок). Сбросьте флажок **Постоянная партия**, поскольку мы хотим, чтобы у нас была возможность позднее разобрать эту партию на отдельные коробки (рис. 3.1527 Укажите **фигуру анимации** *shapeLoading* и **тип анимации**: *Мешок*.

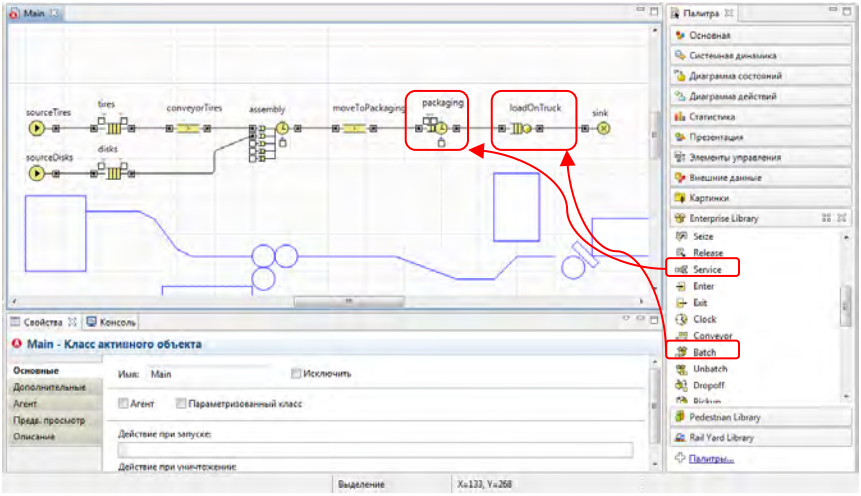


Рис. 3.27. Добавление новых объектов

6. Запустите модель. Вы увидите, как товары упаковываются в коробки, и партии этих коробок забираются с завода.

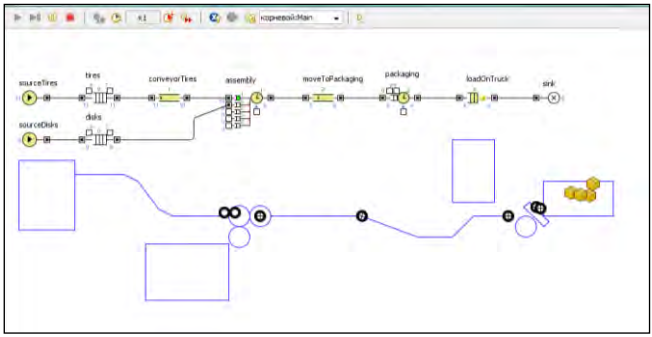


Рис. 3.28. Режим выполнения модели

На самом деле операции сборки и упаковки требуют участия ресурсов – сборочного робота и упаковщиков соответственно. Поэтому необходимо добавить в модель ресурсы: робот для выполнения сборки;

двое рабочих для упаковки готовых товаров. После этого произведем сбор статистики занятости рабочих и отобразим ее на столбиковой диаграмме. И, наконец, промоделируем поломки робота сборки.

Алгоритм выполнения работы следующий:

1. Нарисуйте две картинки, обозначающие свободного и занятого рабочих. Фигурки рабочих можно взять из набора стандартных картинок палитры **Картинки**.

2. Измените цвета картинок. Щелкните по первой картинке. Первый щелчок выделит всю группу фигур. Щелкните по конкретной фигуре, которую вы хотите изменить. Вы увидите начало координат группы и свойства выделенной фигуры. Измените цвет заливки этой фигуры. Вы увидите, что фигура изменит свой цвет. Аналогично измените цвет другой картинки и переименуйте эти фигуры (рис. 3.29).

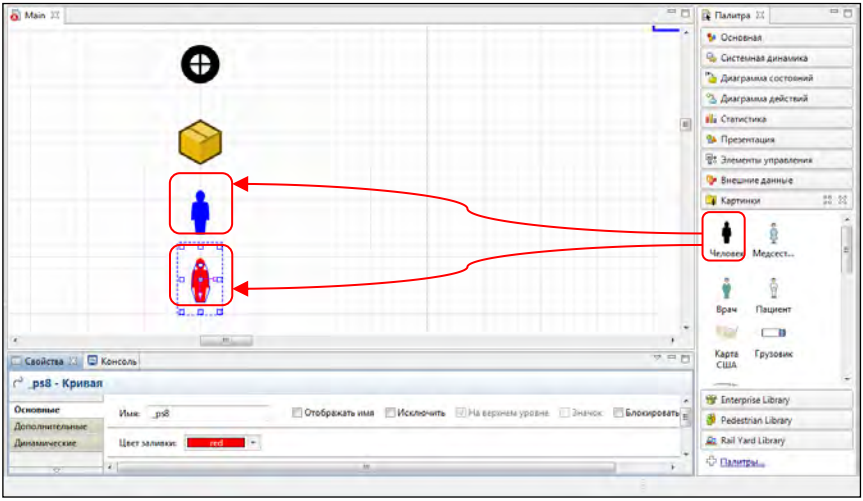


Рис. 3.29. Создание фигурок рабочих

3. Вставьте два объекта **ResoursePool** из палитры **Enterprise Library** в графическую диаграмму. Первый объект будет моделировать робота, производящего сборку колес. Назовите его robots и оставьте свойства без изменений. Второй объект **ResourcePool** будет моделировать упаковщиков. Назовите этот объект workers. Задайте количество рабочих в поле **Количество ресурсов** равным *2*. Выберите картинки **shapeWorkerIdle** и **shapeWorkerBusy** в качестве фигур, обозначающих свободного и занятого рабочего соответственно. Выберите прямоугольник **shapeWorkers** в качестве базового местоположения рабочих. Установите флажок **Включить сбор статистики**, чтобы разрешить объекту собирать статистику занятости задаваемых им ресурсов. Соедините объекты ResourcePool с теми объектами, которые будут работать с соответствующими ресурсами: **assembly** и **packaging** (рис. 3.30). Задайте количество ресурсов, необходимое объектам **assembly** и **packaging** для выполнения операций.

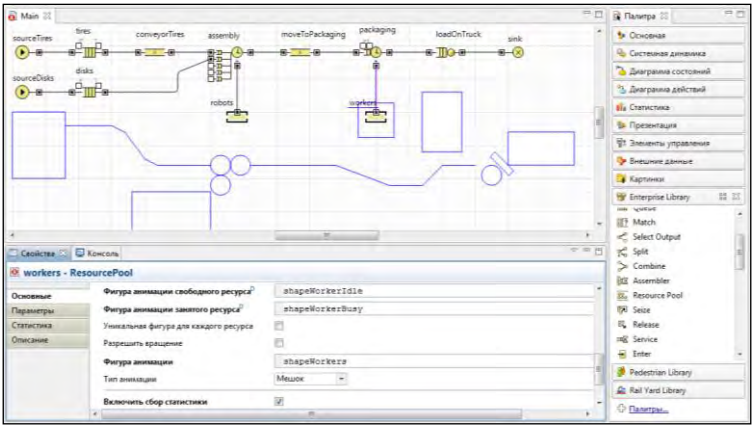


Рис. 3.30. Создание фигурок рабочих

4. Перетащите объект **Столбиковая диаграмма** из палитры **Статистика** на графическую диаграмму. Щелкните по кнопке **Добавить элемент данных** в панели **Свойства**. Задайте *Workers utilization* в качестве **Заголовка** элемента данных. Задайте Значение, которое будет отображаться этой столбиковой диаграммой: *workers.statsUtilization.mean()*. Здесь **workers** – имя нашего объекта **ResourcePool**, **statsUtilization** – функция, собирающая статистику занятости ресурсов, а **mean()** возвращает среднее значение собранной статистики. Перейдите на страницу свойств **Внешний вид** и измените **Направление роста** столбцов, как показано на рис. 3.31.

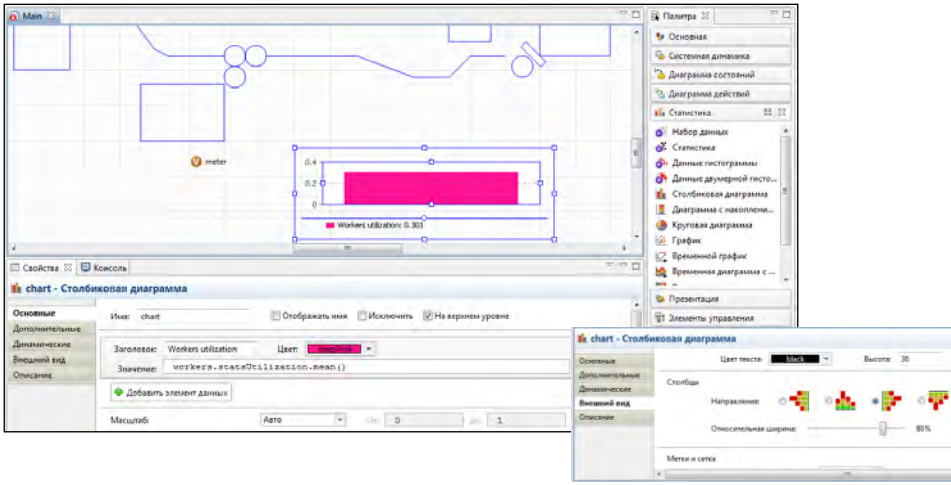


Рис. 3.31. Создание столбиковой диаграммы

5. Задайте два параметра: **MTTF** и **MTTR**. Параметр **MTTF** задает среднее время, после которого робот придет в неисправное

состояние. Мы полагаем это время равным 45 дням (с помощью функции AnyLogic **day()** мы поучаем значение, равное одному дню). Параметр **MTTR** задает среднее время, необходимое на то, чтобы отремонтировать робота и привести его в рабочее состояние. Мы задаем среднее время на восстановление равным одной неделе (*7*day()*).

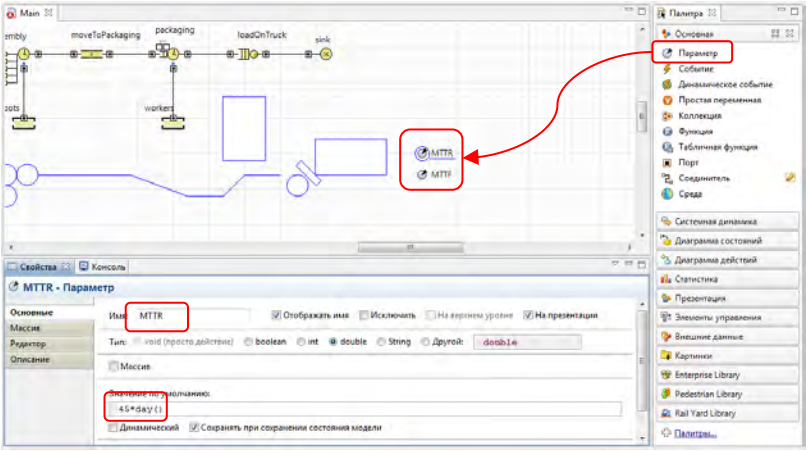


Рис. 3.32. Переменные для моделирования поломки робота

6. Задайте поведение робота с помощью диаграммы состояний (стейтчарта). Начните рисование диаграммы состояний с добавления двух состояний. Назовите состояния как показано на рис. 3.33: **Working** и **OutOfOrder**. Добавьте **Начало диаграммы состояний**, указывающее на верхнее состояние. Имя этого элемента будет играть роль и имени всей диаграммы состояний. Проверьте, соединена ли конечная точка этого элемента с состоянием (выделите его – в случае правильного соединения конечная точка должна будет подсветиться зеленым цветом, как показано на рисунке).

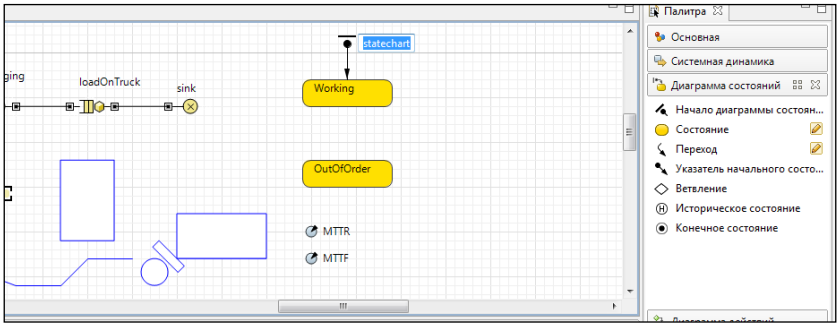


Рис. 3.33. Рисование стейтчарта

7. Нарисуйте переход, ведущий из состояния **Working** в состояние **OutOfOrder**. Чтобы нарисовать такой переход, нужно сделать двойной щелчок по элементу **Переход** в палитре **Диаграмма состояний**, затем щелкнуть по состоянию **Working** и наконец – по состоянию **OutOfOrder**. Этот переход будет моделировать поломку робота. Пусть он срабатывает по истечении экспоненциально распределенного таймаута со средним значением, равным MTTF (45 дней).

8. Нарисуйте переход, ведущий из состояния **OutOfOrder** в состояние **Working**. Этот переход будет моделировать окончание работы по восстановлению сломанного оборудования. Пусть этот переход срабатывает с интенсивностью 1/MTTR. Установите для каждого перехода флажок **Отображать имя** и измените положения меток в графическом редакторе.

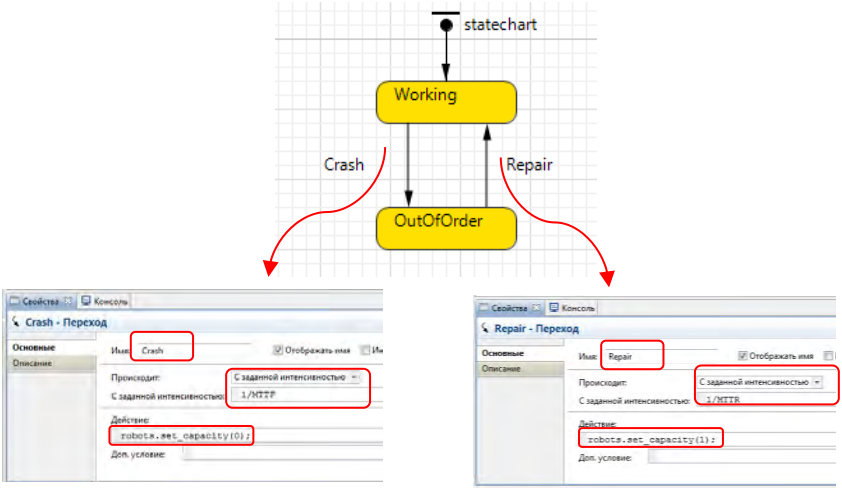


Рис. 3.34. Рисование переходов

9. Запустите модель. Вы можете проанализировать влияние поломок оборудования и длительности периода технического обслуживания на производительность завода.

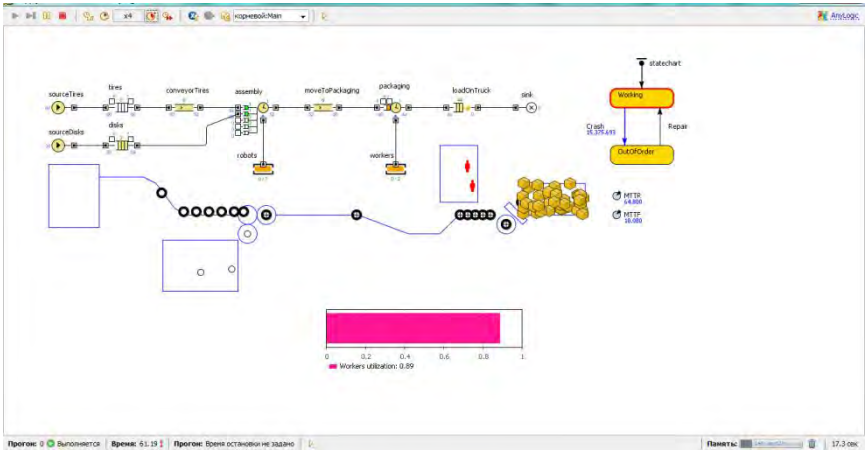


Рис. 3.35. Выполнение модели

3.4 Лабораторная работа 4. Модель ритейлера

Мы построим модель оптового магазина (дальнейшем будем называть его ритейлером). Для простоты предположим, что этот магазин торгует одним видом товара.

Ритейлер работает следующим образом:

Изначально ритейлер заказывает определенное количество товара. Эти товары производятся каким-то сторонним производителем и доставляются ритейлеру. По получении они помещаются на склад ритейлера.

Периодически товары продаются. Когда приходит очередной запрос на покупку, товар извлекается со склада и продается.

Когда уровень товарных запасов ритейлера достигает заданной нижней границы, ритейлер заказывает новую партию товара, так, чтобы по ее получении уровень товарных запасов пополнился до заданной для этого ритейлера верхней границы.

Начнем с простой постановки задачи.

Сделаем следующие предположения:

Товары поставляются ритейлеру со средней интенсивностью 1 единица в минуту. Доставленные товары помещаются в зону разгрузки.

После доставки каждый товар помещается на хранение в свободную ячейку склада ритейлера с помощью свободного погрузчика.

Товар хранится в ячейке от 20 до 45 минут, после чего он извлекается оттуда и доставляется с помощью погрузчика в зону выдачи, откуда и забирается потребителем.

Создайте новый класс активного объекта. Назовите его Retailer.

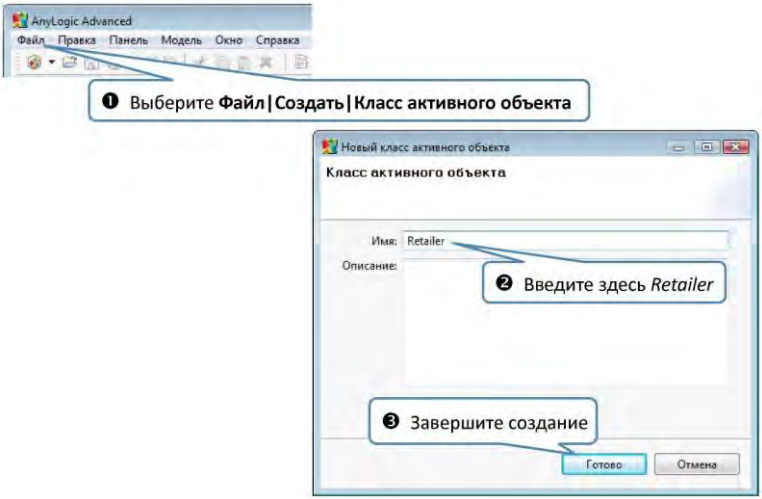


Рис. 3.36. Создание класса активного объекта

Активные объекты:

- Активные объекты являются основными строительными блоками модели AnyLogic. Активные объекты могут моделировать любые объекты реального мира: машины, людей, станки, цеха, города, компании, здания и т.д.

- Каждый активный объект обычно моделирует логически обособленную часть модели. Это позволяет проводить декомпозицию модели на необходимое количество уровней детальности.

До этого момента мы задавали логику всей модели только на диаграмме класса активного объекта Main. Теперь мы хотим создать другой компонент нашей глобальной модели цепочки поставок - модель ритейлера.

Поскольку этот компонент можно представить себе как логически обособленную часть, но в тоже время является частью той же модели,

которую мы начали разрабатывать ранее, мы создаем в рамках текущей модели еще один класс активного объекта, чтобы задать логику ритейлера на отдельной диаграмме, отдельно от логики работы завода.

Переименуйте активный объект Main в Factor:

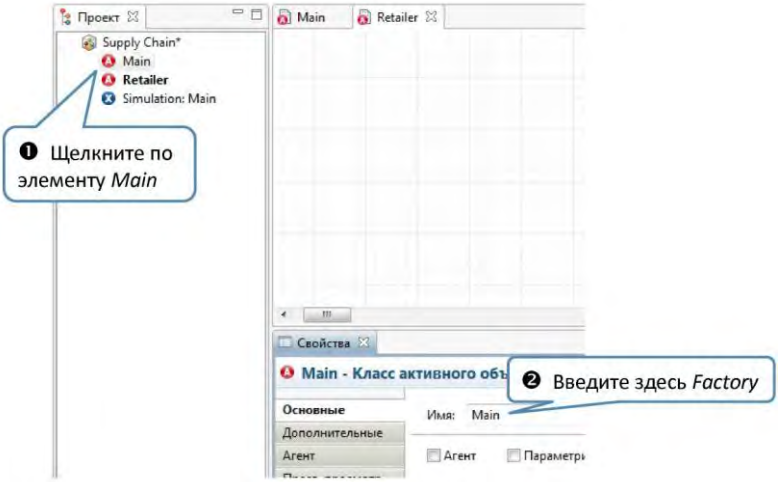


Рис. 3.37. Переименование класса активного объекта

Как узнать, какой активный объект редактируется?

• Теперь в Вашей модели уже два класса активных объектов. Начиная с этого момента, Вы будете время от времени открывать диаграмму того или другого класса, и в итоге у Вас может возникнуть естественный вопрос -диаграмму какого класса активного объекта я редактирую в данный момент?

• Чтобы ответить на этот вопрос, AnyLogic выделяет закладку открытой в данный момент в графическом редакторе диаграммы, а также выделяет редактируемый класс в дереве моделей, отображаемом в панели Проекты.

Добавьте план склада ритейлера. Добавление плана и рисование сети поверх этого плана является первым шагом создания сетевой модели в AnyLogic.

Щелкните по кнопке Добавить и выберите файл retailer_layout.png.

Установите флажок Блокировать, чтобы заблокировать фигуру.

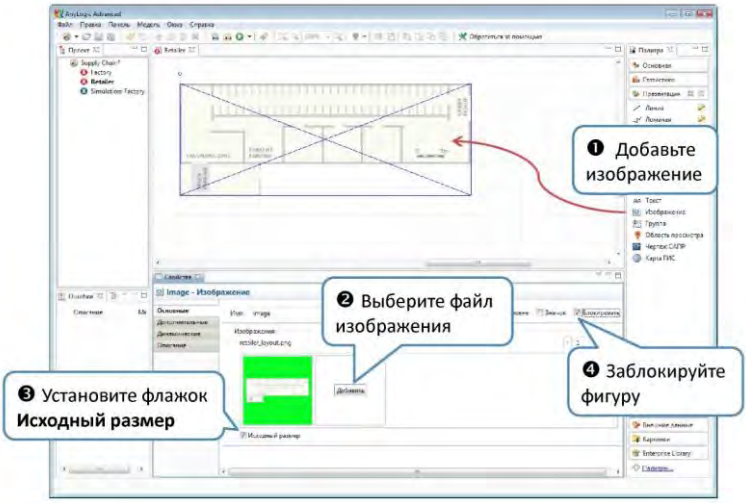


Рис. 3.38. Добавление плана склада

Блокировка фигур презентации:

• Вы можете заблокировать фигуру так, что она не будет выделяться на диаграмме по щелчку мыши на ней (пока Вы не снимете блокировку).

• Обычно это требуется, когда на презентации есть фоновый рисунок (например, план моделируемого Вами предприятия), используемый как подложка для анимации модели. В этом случае при редактировании какой-либо фигуры, лежащей поверх этой подложки, Вы можете случайно отредактировать (например, передвинуть) саму

подложку вследствие того, что иногда не получается точно попасть мышью на ту фигуру, которую Вы хотите выделить.

- Заблокировав Ваш фоновый рисунок, Вы значительно упростите редактирование анимации, поскольку запретите нежелательное выделение фонового рисунка неточными щелчками мыши.

Разметьте план. Задайте с помощью фигур AnyLogic все важные зоны и маршруты движения поверх плана. Логическая структура сети будет сгенерирована в соответствии с нарисованной Вами анимацией.

Нарисуйте зоны с помощью прямоугольников. Назовите их именно так, как показано на слайде. Нарисуйте также дополнительные узлы и соедините прямоугольники ломаными линиями. Эти ломаные линии будут играть роль путей движения.

Добавьте все эти фигуры в группу. Назовите ее *networkGroup*.

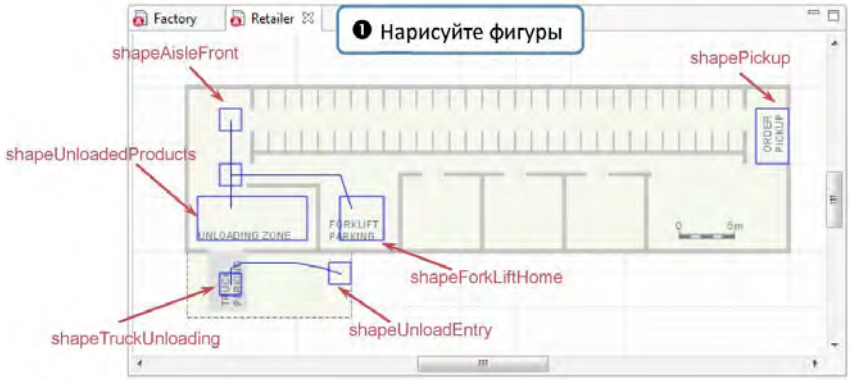


Рис. 3.39. Разметка плана

Нарисуйте прямоугольник, задающий фигуру прохода. Назовите его shapeAisle. Эту фигуру не нужно добавлять в группу, содержащую фигуры сети.

Добавьте картинку погрузчика из палитры Картинки.

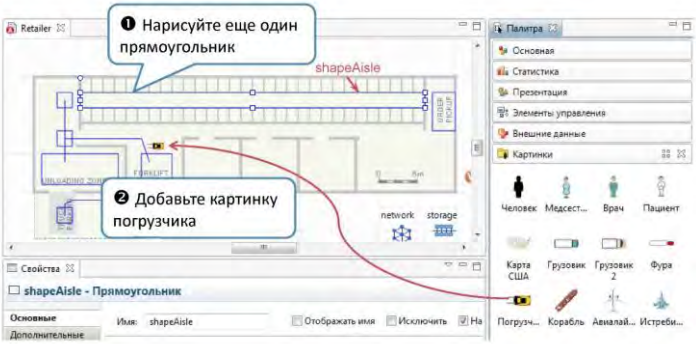


Рис. 3.40. Добавление погрузчика

Скопируйте некоторые элементы из класса *Factory* в класс *Retailer*. Мы делаем это для того, чтобы избавить себя от повторного задания уже имеющихся в другом классе элементов.

Раскрыть ветвь дерева можно щелкнув на значке  (или )слева от имени ветви.

Выделите несколько элементов, последовательно щелкая по ним с нажатой клавишей Ctrl.

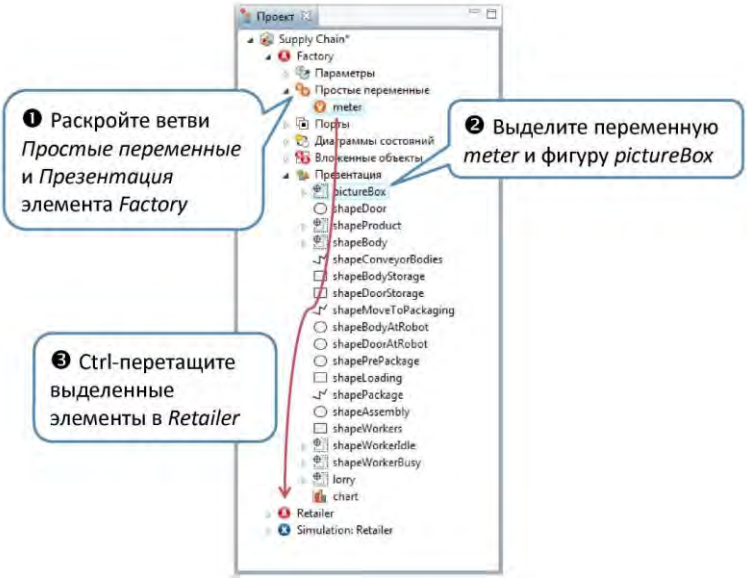


Рис. 3.41. Редактирование класса Retailer

Добавьте указанные объекты библиотеки Enterprise Library на диаграмму класса Retailer и измените их свойства:

Объект Network задает топологию и параметры сети, а также осуществляет управление сетевыми ресурсами.

- Укажите Группу фигур сети: networkGroup. Здесь networkGroup – имя нашей группы, содержащей прямоугольники и ломаные, задающие структуру сети.

Объект NetworkResourcePool задает набор сетевых ресурсов. С помощью этого объекта зададим набор движущихся ресурсов, моделирующих погрузчики.

- Назовите объект forkLiftTrucks;
- Задайте Количество ресурсов этого типа: 5;

• Задайте Скорость, с которой будут двигаться эти ресурсы: 1*meter/second();

• Укажите фигуру, которой будут отображаться эти ресурсы: fork;

• Установите флажок Разрешить вращение, чтобы фигуры погрузчиков могли поворачиваться согласно направлению их движения;

• Задайте базовое местоположение этих ресурсов в сети. В поле Базовый узел, введите имя прямоугольника, задающего соответствующий узел сети: shapeForkLiftHome;

Соедините объекты, чтобы указать ресурсам, что они будут работать в сети, заданной этим объектом Network.

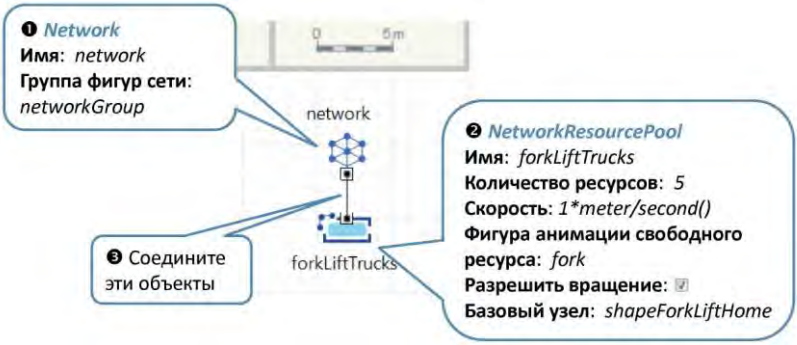


Рис. 3.42. Создание транспортной сети

Добавьте объект NetworkStorage, который будет моделировать область хранения склада. Измените свойства объекта:

• Пусть у нашего склада будет 30 ячеек в ряду.

• Чтобы задать склад, Вам нужно просто нарисовать три прямоугольника: Узел у начала прохода (shapeAisleFront), Фигура прохода (shapeAisle) и Узел у конца прохода (shapePickup). Оба узла у прохода

должны принадлежать сети (то есть быть добавленными в Группу фигур сети), в то время как фигура прохода, наоборот, не должна.

Для простоты мы будем моделировать только одну область хранения. Если вам нужно смоделировать несколько зон, воспользуйтесь объектом NetworkStorageZone.

Моделирование складов и зон хранения:

Для моделирования хранилищ с множеством ячеек, имеющих периодическую структуру (склады и т.д.), Enterprise Library предоставляет следующие объекты:

NetworkStorage моделирует два стоящих друг напротив друга стеллажа и проход между ними. В каждой ячейке может находиться только одна заявка.

NetworkStorageZone моделирует зону хранения, состоящую из набора стеллажей и проходов между ними (моделируемыми с помощью объектов NetworkStorage).

NetworkStoragePut помещает заявку в ячейку заданного стеллажа или зоны хранения. Заявка при этом перемещается из ее текущего местоположения в сети к ячейке (при необходимости - с помощью движущихся сетевых ресурсов).

NetworkStoragePick извлекает заявку из ячейки заданной зоны хранения и перемещает ее в заданное место сети (при необходимости - с помощью движущихся сетевых ресурсов).

Рис. 3.43. Создание хранилища

Создайте простую диаграмму процесса из объектов Enterprise Library.

Назовите эти объекты так, как показано на рисунке выше.

- Группа объектов Enterprise Library, имена которых начинаются с Network, используется при моделировании транспортных сетей, строящихся на базе имеющегося плана (помещения, местности и т.п.). Обычно они используются при моделировании процессов, проистекающих в каком-то определенном физическом пространстве и включающих в себя движение заявок и ресурсов.
- Вы можете легко отличить такие объекты по значкам синих оттенков.
- Такие «сетевые» объекты могут легко сочетаться в одной диаграмме процесса с «обычными» объектами Enterprise Library, такими как Delay, Queue, и т.д.

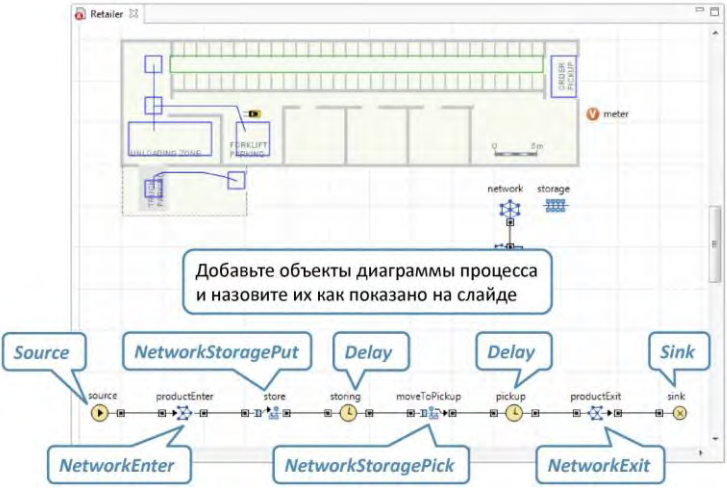


Рис. 3.44. Диаграмма процессов

Объект Source моделирует прибытие товаров.

• Укажите нашу картинку pictureBox в качестве фигуры анимации заявок,

создаваемых этим объектом Source.

Объект NetworkEnter добавляет входящую заявку в сеть и помещает ее в указанный узел сети.

• Параметр Сеть определяет сеть, в которую попадут заявки. Введите здесь имя нашего объекта Network: network.

• В поле Узел входа Вы задаете узел сети, в котором появится заявка. Введите здесь имя прямоугольника, задающего соответствующий узел: shapeUnloadedProducts.

• Задайте Скорость, с которой заявки будут перемещаться по сети. Вводя здесь forkLiftTrucks.speed мы задаем скорость равной скорости погрузчиков.

Объект NetworkStoragePut моделирует помещение поступающих товаров в ячейки склада.

• Задайте зону хранения, в которую должны быть помещены товары. Введите имя нашего объекта NetworkStorage в поле NetworkStorage или NetworkStorageZone.

• Установите флажок Перемещать с помощью ресурсов, поскольку мы хотим, чтобы товары доставлялись к ячейкам с помощью ресурсов - погрузчиков.

• Укажите, какие именно ресурсы понадобятся для перемещения. Для этого напишите список имен объектов NetworkResourcePool, задающих требуемые ресурсы в поле Список ресурсов {pool1, …}.

Обратите внимание, что список должен быть помещен в фигурные скобки.

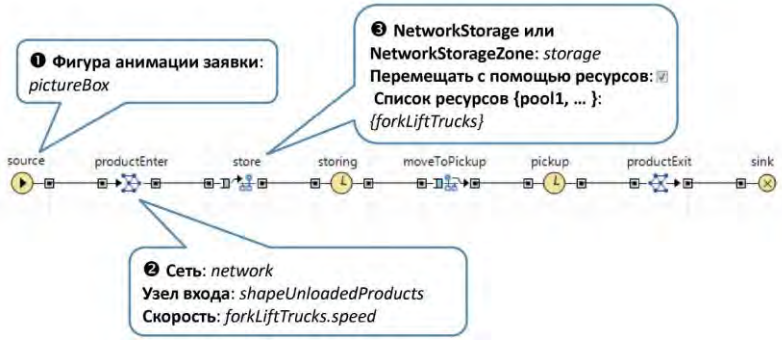


Рис. 3.45. Редактирование блоков

Объект Delay задерживает заявки на заданное время. Этим объектом мы хотим промоделировать хранение товаров на складе.

• Задайте время хранения в поле Время задержки: uniform(20, 45)* minute()

- Установите флажок Максимальная вместимость, чтобы разрешить объекту задерживать неограниченное количество заявок одновременно.

Объект NetworkStoragePick извлекает заявку из ячейки хранения и

перемещает ее в указанное место сети. Мы добавляем этот объект, чтобы промоделировать то, как погрузчик извлекает коробку с товаром из ячейки склада и перемещает ее к зоне выдачи товара.

- В поле NetworkStorage или NetworkStorageZone задайте имя объекта, задающего область хранения: storage.
- Задайте Узел назначения, в который будет перемещена заявка после ее извлечения из ячейки: shapePickup.
- Установите флажок Перемещать с помощью ресурсов, поскольку мы хотим, чтобы товары перемещались к зоне выдачи ритейлера с помощью погрузчиков.
- В поле Список ресурсов {pool1, …} напишите список имен объектов NetworkResourcePool, задающих ресурсы, которые должны использоваться для транспортировки товаров.

Этот объект Delay моделирует время, необходимое для того, чтобы забрать товар.

- Задайте Время задержки для этого объекта: minute().
- Установите флажок Максимальная вместимость.

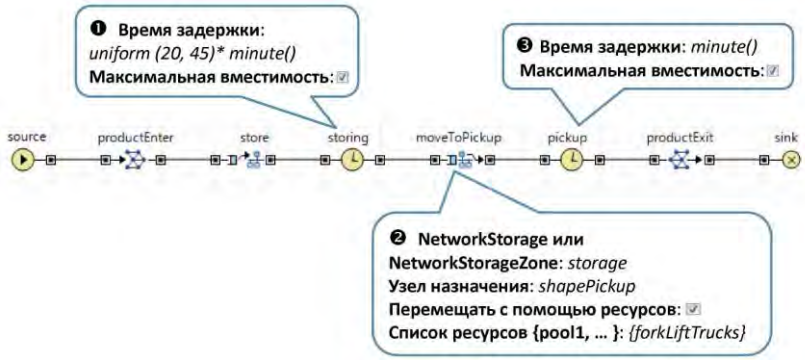


Рис. 3.46. Редактирование блоков

Измените главный (корневой) класс активного объекта эксперимента Simulation.

Главный (корневой) активный объект

• Модель AnyLogic обычно представляет собой дерево активных объектов, вложенных друг в друга. Объект, являющийся корнем этого дерева, называется корневым объектом модели. Корневой объект представляет самый верхний уровень абстракции модели.

• Выбирая корневой объект модели, Вы говорите AnyLogic, с какого объекта начать построение модели. Изменяя корневой объект эксперимента, Вы можете очень легко изменить структуру запускаемой модели.

Запустите модель.

Вы увидите как коробки с товаром появляются в зоне разгрузки, затем помещаются на хранение в ячейки склада и по прошествии определенного времени забираются со склада для продажи.

Усовершенствуем модель:

- Предположим, что наш товар доставляется на склад грузовиками, каждый из которых перевозит партию из 10 колес.
- По прибытии грузовик разгружается. Время разгрузки распределено по треугольному закону с параметрами 1, 2, 3 минуты.
- Полученные колеса так же, как это было и раньше, помещаются в свободные ячейки склада с помощью погрузчиков.

Добавьте в диаграмму процесса новые объекты:

Этот объект Batch будет моделировать создание партии товара, отправляемой на склад ритейлера.

- Сбросьте флажок Постоянная партия, чтобы разрешить последующую разборку партии на отдельные заявки (моделирующие коробки колесами внутри).
- Задайте нашу картинку lorry в качестве фигуры, которой будут отображаться заявки-партии (в нашем случае они будут моделировать грузовики).
- Разрешите вращение фигур анимации движущихся грузовиков согласно направлению их движения.

Этот объект NetworkEnter добавляет грузовики в нашу сеть.

- Укажите сеть, в которую будут добавлены грузовики.
- Укажите узел сети, в который попадут грузовики: shapeUnloadEntry.
- Задайте скорость движения грузовиков равной двум метрам в секунду: 2*meter/second().

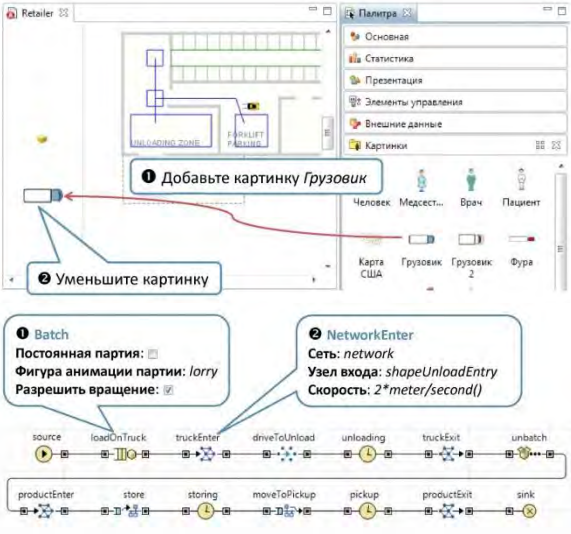


Рис. 3.47. Добавление грузовика

Объект NetworkMoveTo перемещает заявку в указанное место сети. С помощью этого объекта мы перемещаем грузовики от въезда на склад к зоне разгрузки.

- Укажите место назначения заявки в поле Узел: shapeTruckUnloading.
- Введите entity.setOffsets(0, 0, PI/2) в поле Действие при выходе. Таким способом мы поворачиваем фигурку грузовика, чтобы она не перемещалась «задом наперед».

Объект Delay моделирует задержку, требуемую на разгрузку коробок колесами.

- Задайте Время задержки: triangular(1, 2, 3)*minute()
- Сделайте вместимость объекта максимально возможной для того, чтобы разрешить одновременную разгрузку нескольких грузовиков.

Этот объект NetworkExit будет удалять из сети те грузовики, которые успешно доставят свой товар ритейлеру.

Объект Unbatch выполняет разборку заявки-партии на отдельные заявки, представляющие собой коробки с колесами.

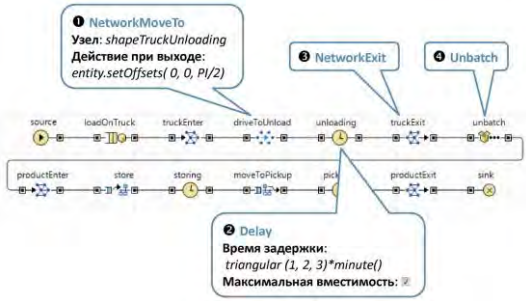


Рис. 3.48. Установка параметров блоков

Запустите модель.

Теперь мы реализуем политику пополнения товарных запасов ритейлера. Остановимся на широко применяемой политике (s, S), заключающейся в следующем:

- Заданы две границы уровня товарных запасов: нижняя (s) и верхняя (S).
- В том случае, если ритейлер испытывает дефицит товара, то есть количество товара, хранящегося на складе ритейлера, а также направляющегося к нему на грузовиках, падает до нижнего уровня s или ниже, отсылается заказ на столько единиц товара, сколько необходимо для пополнения товарных запасов до верхнего уровня S.

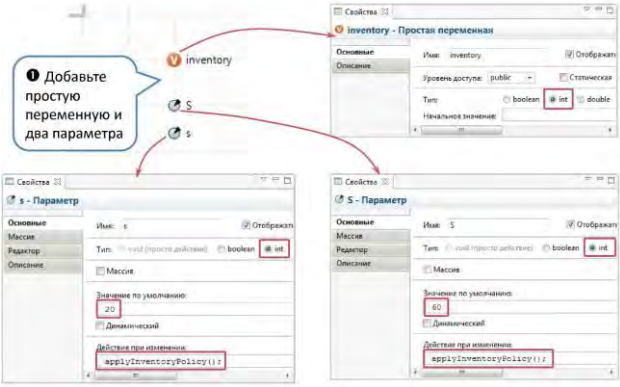


Рис. 3.49. Добавление параметров и переменных

Переменная inventory будет хранить текущее значение уровня товарных запасов ритейлера.

- Сделайте эту переменную целочисленной (типа int), поскольку она будет считать единицы товара. Параметр S будет задавать верхнюю границу уровня товарных запасов.

- Сделайте этот параметр целочисленным, задайте его Значение по умолчанию равным числу ячеек хранения на складе (60) и введите applyInventoryPolicy(); в его Действии при изменении. Этот код будет пересчитывать текущее значение уровня запасов ритейлера путем вызова функции, которую мы зададим позднее. Параметр s будет задавать нижнюю границу уровня товарных запасов.

- Создайте этот параметр путем клонирования ранее созданного параметра S. Переименуйте его в s и задайте его Значение по умолчанию равным 20.

Как Вы можете увидеть, здесь для задания на первый взгляд одинаковых сущностей используются различные элементы - переменная и параметр. В чем причина такого решения?

Параметры или переменные: что использовать?

• Параметры обычно используются для задания статических характеристик объекта Значение параметра обычно остается неизменным во время "прогона" модели и изменяется пользователем только в определенные моменты времени (обычно - между "прогонами" модели) при желании изменить характеристики модели.

• Переменные обычно используются для хранения результатов моделирования, а также для задания меняющихся по ходу моделирования данных и характеристик.

Задайте алгоритм, который будет проверять, является ли текущий уровень товарных запасов ритейлера достаточным, и заказывать необходимое количество единиц товара в случае такой необходимости. Давайте зададим этот алгоритм графически с помощью диаграммы действий.

Диаграммы действий

• Диаграмма действий представляет собой структурированную блок-схему, позволяющую графически задать алгоритм в стиле структурного программирования. Диаграмма действий собирается из блоков, расположенных на палитре Диаграмма действий.

• Диаграммы действий облегчают задание алгоритмов, делая необязательным знание синтаксиса Java операторов.

• С помощью диаграмм действий Вы можете визуализировать алгоритмы, делая их более понятными для других пользователей модели.

Начните рисование диаграммы действий с добавления элемента Диаграмма действий. Тем самым Вы создадите простейшую диаграмму действий, состоящую из начальной точки (задаваемой собственно блоком Диаграмма действий) и блока Вернуть значение. Теперь Вы можете добавлять в созданную структуру другие блоки диаграммы действий согласно логике задаваемого Вами алгоритма.

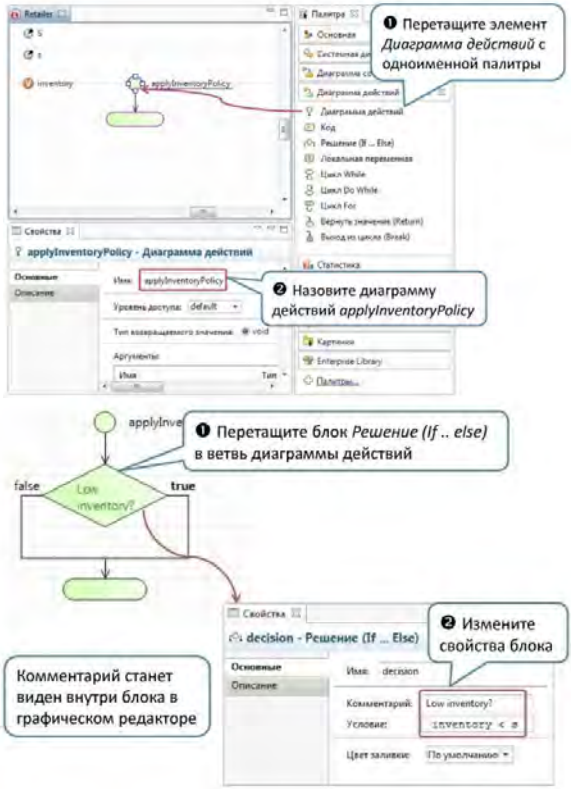


Рис. 3.50. Диаграмма действий

Для того, чтобы наш алгоритм проверял определенное условие и выполнял то или другое действие в зависимости от результата этой проверки, воспользуемся блоком Решение (If .. Else).

Задайте Условие блока: inventory < s. Здесь мы проверяем, не ниже ли текущий уровень товарных запасов минимально допустимого уровня s.

Блок Решение (If … Else):

• Блок Решение (If .. Else) позволяет осуществлять ветвление алгоритма. У блока есть две исходящие ветви - true и false. С помощью других блоков Вы можете задать последовательность действий для каждой из этих ветвей. Когда управление дойдет до данного блока, будет приниматься решение о том, по какой ветви управление пойдет дальше. Если заданное для блока условие будет выполнено, то будет выбрана ветвь true. В противном случае будет выбрана ветвь false.

Добавление блоков в диаграмму действий:

• Перетаскивая блок над областью графического редактора, Вы увидите, что определенные точки на ветвях диаграммы действий будут выделяться с помощью маленьких кружков. Чтобы вставить блок в одну из этих точек, отпустите кнопку мыши, когда курсор будет над нужной Вам точкой.

Поместите два блока в ветвь true блока Решение для того, чтобы реализовать заказ товаров в случае низкого уровня товарных запасов ритейлера.

Локальная переменная quantity будет подсчитывать количество единиц товара, которое должно быть заказано (путем вычисления разницы

между верхним уровнем товарных запасов и текущим уровнем: S - inventory).

Блок code будет выполнять следующий код: source.inject(quantity); inventory += quantity;

Первая строка создает в объекте source заданное число заявок (равное quantity). Вторая увеличивает текущее значение уровня запасов на это значение. При желании Вы можете добавить для блока комментарий. Он будет отображаться внутри блока и объяснять смысл блока другим пользователям..

Локальная переменная:

• Блок Локальная переменная используется для задания переменной внутри

диаграммы действий. Такая переменная видна только не во всей диаграмме

действий, а только в той ее части, которая следует за точкой объявления переменной.

Код:

• Блок Код позволяет добавлять в Вашу диаграмму действий фрагменты кода, выполняющие нужные Вам действия.

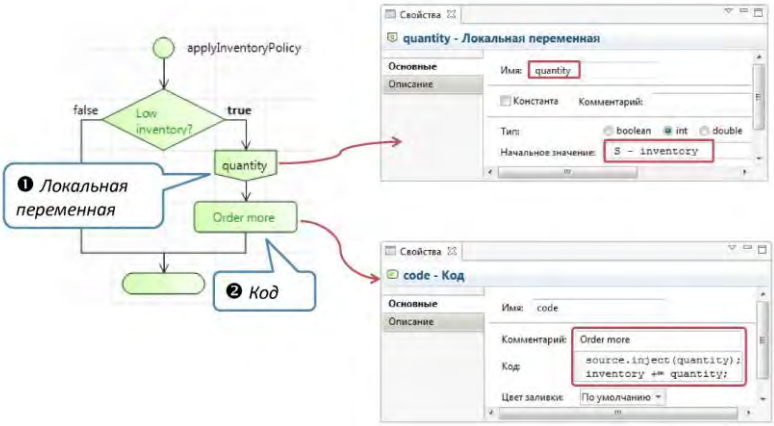


Рис. 3.51. Диаграмма действий

Нам необходимо проверять уровень товарных запасов при запуске модели и при продаже каждой единицы товара.

Первым делом реализуем проверку при запуске модели. Чтобы выполнить алгоритм при запуске, поместим вызов диаграммы действий в Действие при запуске класса Retailer.

Действие при запуске

• Действие при запуске выполняется на финальной стадии инициализации

модели, после того, как все объекты модели будут созданы, соединены и

проинициализированы, но до выполнения каких-либо активностей модели.

Здесь Вы можете, например, запустить какие-либо события этого объекта.

Как выполнить алгоритм, заданный диаграммой действий?

- Диаграммы действий выполняются так же, как и функции - Вы помещаете в код вызов диаграммы действий (имя диаграммы, за которыми следуют скобки):

applyInventoryPolicy();

- Если у диаграммы действий есть аргументы, то Вы должны указать значения мэтих параметров внутри скобок, разделив их запятыми, например:

moveTo(15, 20);

Измените режим генерации заявок объекта source. Пусть он создает заявки не согласно заданной интенсивности, а по вызовам метода inject() (как Вы помните, мы поместили вызов этого метода в блок кода code диаграммы действий applyInventoryPolicy).

Задайте действия, которые должны выполняться при продаже товара со склада:

inventory --; applyInventoryPolicy();

Первая строка кода уменьшает текущее значение уровня товарных запасов на единицу. Вторая выполняет диаграмму действий, которая проверяет новый уровень товарных запасов, и в случае его недостаточности заказывает производство и доставку новых товаров.

Добавьте диаграмму для отображения количества единиц товара, хранимого в текущий момент времени на складе, а также находящегося в пути к ритейлеру.

Добавьте Временную диаграмму с накоплением с палитры Статистика и измените ее размер, как показано на слайде выше.

Добавьте два элемента данных, один - отображающий Значение storage.size() с Заголовком On the stock, а другой - отображающий inventory-storage.size() с Заголовком Expecting.

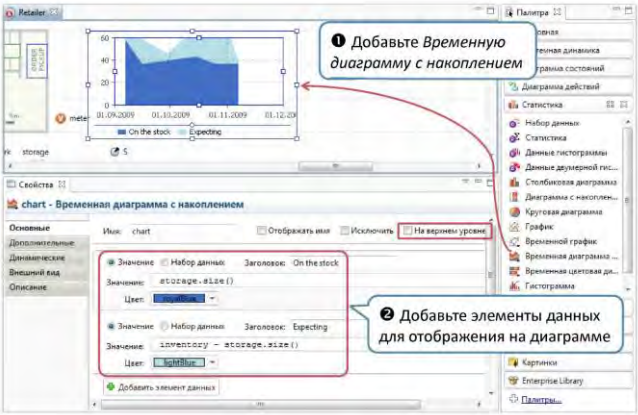


Рис. 3.52. Добавление диаграммы

Измените свойства диаграммы:

- Задайте временной диапазон диаграммы равным одной неделе.
- Задайте Фиксированную шкалу с максимальным значением для оси Y 60.
- Измените частоту обновления диаграммы.
- Пусть на диаграмме единовременно отображается до 200 значений.
- Задайте отображение модельного времени в метках временной оси диаграммы.

Отображение модельного времени в метках временных диаграмм:

Все временные диаграммы (временной график, временная диаграмма с накоплением и временная цветовая диаграмма) могут

отображать модельные даты в метках своей временной оси. Вы можете выбрать наиболее подходящий Вам формат в поле Формат временной оси на странице свойств Внешний вид.

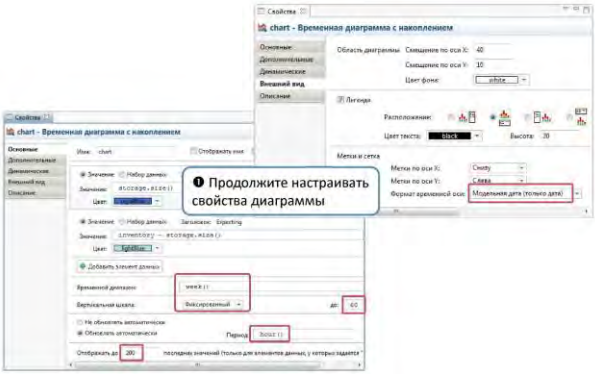


Рис. 3.53. Редактирование диаграммы

Добавьте два бегунка для того, чтобы интерактивно варьировать значения параметров s и S во время моделирования. Добавьте бегунок из палитры Элементы управления. Установите флажок Связать с и введите имя параметра, значение которого Вы хотите варьировать с помощью этого бегунка, в расположенном справа поле: S. Создайте копию этого бегунка. Пусть этот бегунок изменяет значение параметра s.

Увеличьте размер окна презентации, чтобы оно вместило в себя только что добавленные нами элементы. Разверните в дереве проектов элемент эксперимента Simulation:Main и затем разверните его подветвь Презентация. Щелкните на элементе frame, чтобы просмотреть его свойства.

Задайте ширину рамки равной 1000 и оставьте заданную по умолчанию высоту 600. Таким способом Вы задаете начальный размер окна презентации.

Запустите модель.

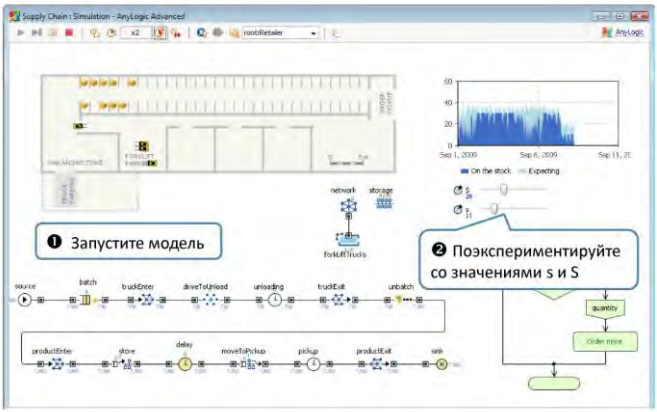


Рис. 3.54. Выполнение модели

4 Рекомендации по выполнению курсового проекта

4.1 Требования к оформлению курсового проекта

Содержание пояснительной записки (ПЗ) должно соответствовать выданному заданию на курсовую работу. Оформление пояснительной записки осуществляется в текстовом редакторе Microsoft Word. Шрифт Times New Roman № 14, интервал 1,5. Текст располагают на листе соблюдая следующие поля: верхнее – 1,0 см., нижнее -3 см., левое - 2,5 см., правое - 1,0 см. Отступ первой строки абзаца (красная строка) 1,25 см.

Заголовки структурных элементов ПЗ разделов основной части печатают прописными буквами, жирным шрифтом, не подчеркивая, с

высотой букв и цифр №16. Сокращения в заголовках не допускаются. Каждый структурный элемент (раздел) начинают с новой страницы.

Заголовки пунктов и подпунктов начинают с абзацного отступа и печатают с прописной буквы, не подчеркивая, без точки в конце. Заголовки пунктов и подпунктов выполняют жирным шрифтом №14. Страницы ПЗ нумеруют арабскими цифрами, соблюдая сквозную нумерацию по всему тексту ПЗ. Титульный лист включается в общую нумерацию страниц ПЗ. Номер страницы на нем не проставляется. Иллюстрации (чертежи, графики, схемы, диаграммы, фотоснимки) следует располагать в ПЗ непосредственно после текста, в котором они упоминаются впервые, или на следующей странице, отделяя от текста пустой строкой сверху и снизу. Иллюстрация обозначается словом «Рисунок 1.1. – Название рисунка», которое помещают после поясняющих данных и нумеруют арабскими цифрами с указанием порядкового номера раздела пояснительной записки и порядкового номера рисунка в пределах раздела. Ссылки на источники указывают порядковым номером по списку источников, выделенным квадратными скобками, например: [1]. Сведения об использованных источниках располагают в порядке появления ссылок на источники в тексте ПЗ и оформляют в соответствии с ГОСТ 7.1-2003.

4.2 Структура курсового проекта

Содержание работы должны включать следующие разделы и параграфы:

Введение

1. Теория системного моделирования

1.1. Системы и системный анализ

1.2. Моделирование процессов и систем

2. Имитационное моделирование

2.1. Применение имитационного моделирования

2.2. Виды имитационного моделирования

2.3. Области применения

3. Математическая модель решения задачи

3.1. Теория систем массового обслуживания

3.2. Решение задачи аналитическим методом

4. Решение задачи с помощью имитационного моделирования

4.1. Постановка задачи

4.2. Структура модели

4.3. Параметры блоков модели

4.4. Анимация заявок

4.5. Выполнение симуляционного эксперимента модели

4.6. Выполнение оптимизационного эксперимента модели

5. Интерпретация результатов

Заключение

Список использованных источников

В зависимости от специфики предметной области и задания на курсовое проектирование данная структура может быть изменена по согласованию с преподавателем.

4.3 Примерны темы курсовых проектов

Тема 1. Моделирование работы обрабатывающего цеха

Описание процесса

В обрабатывающий цех через $a \pm b$ минут поступают детали двух типов: с вероятностью p1 – первого типа, с вероятностью p2 – второго

типа.

Детали первого типа обрабатываются станком А (время обработки c ± d минуты, в каждый момент времени может обрабатываться одна деталь). С вероятностью p3 деталь не отвечает требованиям качества и возвращается на повторную обработку на станок А, в противном случае она поступает на станок С.

Детали второго типа обрабатываются станком В (время обработки e ± f минут, в каждый момент времени может обрабатываться только одна деталь). С вероятностью p3 деталь не отвечает требованиям качества и возвращается на повторную обработку на станок В, в противном случае она поступает на станок С. Станок С может обрабатывать до g деталей одновременно, время обслуживания одной детали составляет k ± m минут.

Параметр	Варианты		
	1	2	3
a ± b	5±1	6±2	7±2
p_1	0,4	0,5	0,7
p_2	0,6	0,5	0,3
c ± d	15±5	16±6	14±10
p_3	0,1	0,05	0,075
e ± f	8±4	12±6	16±8
g	5	4	3
k ± m	6±2	8±3	9±3
N	10	11	8

Задание

Промоделировать работу цеха на протяжении N часов.

Определить время нахождения детали на обработке в цехе для каждого из вариантов. Реализовать в виде эксперимента сравнения «прогонов» в AnyLogic.

Тема 2. Моделирование работы транспортного цеха

Описание процесса

Транспортный цех обслуживает три филиала А, В и С. Грузовики перевозят изделия из А в В и из В в С, возвращаясь потом в А без груза. Погрузка изделий в филиале А занимает 20 мин, переезд из А в В длится 30 мин, разгрузка и загрузка в филиале В – по 20 мин, переезд в С – 30

мин, разгрузка в С – 20 мин и переезд в А – 20 мин. Если на момент загрузки в филиалах А и В изделия отсутствуют, грузовики уходят дальше по маршруту пустыми. Изделия в А выпускаются партиями по 1000 шт. через 20 ± 3 мин, в В – такими же партиями через 20 ± 5 мин. На линии эксплуатируется восемь (N) грузовиков, каждый может перевозить по 1000 изделий. В начальный момент четыре грузовика находятся в А, четыре – в В.

Задание

Промоделировать работу транспортного цеха на протяжении 1000 ч. Определить частость пустых перегонов грузовиков между филиалами А и В, В и С. Реализовать оптимизационный эксперимент по определению такого оптимального количества N грузовиков, при котором частость пустых перегонов будет минимальной.

Тема 3. Моделирование производственного процесса

Описание процесса

Имеется некоторый производственный процесс, который реализуется линией с тремя последовательно установленными агрегатами: А, Б и В. Поток продукции, который поступает от агрегата А, является пуассоновским со средней нормой выработки 10 изделий за ч. Агрегат Б функционирует по равномерному закону, продолжительность обработки изделия составляет 4 ± 6 мин. Закон распределения времени обслуживания изделий роботами агрегата В приведен в табл.

Вероятность	0,1	0,2	0,4	0,2	0,1
Продолжительность обслуживания, мин	2	3	4	5	6

По умолчанию на агрегате В работают 2 робота.

При скоплении на входе агрегата В двух или более изделий в технологической линии возникает затор.

Задание

Промоделировать функционирование линии на протяжении 100 ч. Определить общее время затора на входе агрегата В. Реализовать оптимизационный эксперимент по определению такого оптимального количества N роботов агрегата В, при котором время затора на входе агрегата В будет минимальным, но при этом время простоя роботов агрегата В также будет минимальным.

Тема 4. Моделирование работы заправочной станции

Описание процесса

На заправке есть три вида топлива для автомобилей: низкооктановый, высокооктановый бензины и дизельное топливо. Для каждого вида топлива есть свои колонки (N колонок). Характеристики заправки приведены в табл.

Вид топлива	Количество колонок	Часть автомобилей, которые заправляются, %	Количество топлива, которым заправляют автомобиль, л	Скорость заправки, л/мин	Стоимость топлива за литр, Руб
Низко-октановый бензин	1	30	Равномерно распределено в интервале 5–60 л (через 5 л)	12	17,0
Высоко-октановый бензин	2	50	Равномерно распределено в интервале 5–40 л (через 5 л)	15	21,5
Дизельное топливо	1	20	Равномерно распределено в интервале 10–60 л (через 5 л)	18	15,0

Прибытие автомобилей на заправку распределено согласно закону Эрланга второго порядка со средним значением 2,2 мин.

В 10 % автомобилей после заправки доливают от 0,5 до 2 л масла. Доливание 0,5 л масла занимает 2 мин. Стоимость одного литра масла – 40 руб.

Перед заправкой предусмотрено 2*N мест для ожидания. Если автомобиль не попадает на эти места, он уезжает в поисках другой

заправки. При этом владелец заправки теряет в среднем 200 руб. от каждого уехавшего автовладельца.

Задание

Промоделировать функционирование заправки на протяжении десяти дней. Реализовать оптимизационный эксперимент по определению такого оптимального количества колонок, при котором прибыль заправки в течение этих 10 дней будет максимальной.

Тема 5. Моделирование работы станции технического обслуживания

Описание процесса

На станцию технического обслуживания (СТО) согласно закону Эрланга второго порядка со средним временем прибытия 14 мин прибывают автомобили для технического обслуживания (36 % автомобилей) и ремонта (64 % автомобилей). На СТО есть два бокса для технического обслуживания и три бокса для ремонта. Выполнение простого, средней сложности и сложного ремонтов – равновероятно.

Время и стоимость выполнения работ по техническому обслуживанию и ремонту зависит от категории выполняемых работ и представлены в табл.

Категория работ	Время ремонта, мин	Стоимость ремонта, руб
Техническое обслуживание	Равномерно распределено в интервале 10-55	Равномерно распределено в интервале 100–400
Простой ремонт	Равномерно распределено в интервале 12-45	Равномерно распределено в интервале 50–450
Ремонт средней сложности	Нормально распределено со средним 45 и среднеквадратичным отклонением 5	Равномерно распределено в интервале 100–1400
Сложный ремонт	Равномерно распределено в интервале 80–150	Равномерно распределено в интервале 350–2550

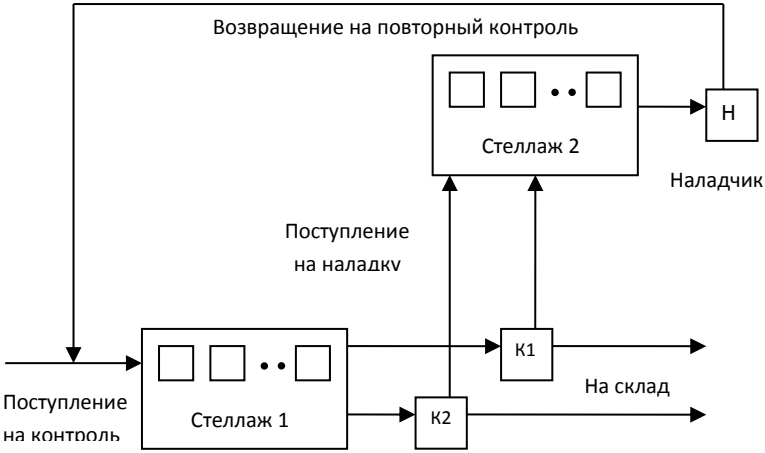
После технического обслуживания 12 % автомобилей поступают для выполнения ремонта средней сложности.

Задание

Промоделировать функционирование СТО на протяжении десяти дней. Реализовать оптимизационный эксперимент с двумя факторами по определению такого оптимального количества боксов для ТО и ремонта, при котором прибыль СТО в течение этих 10 дней будет максимальной.

Тема 6. Моделирование работы станции технического контроля изделий

Собранные коробки передач после сборки проходят испытания на станции технического контроля. Если в процессе контроля оказывается, что функционирование коробки передач ненормально, его переправляют на участок наладки, после которой он вновь возвращается на станцию контроля для повторной проверки. После одной или нескольких проверок коробка передач попадает на склад. На одном месте на любом из стеллажей может храниться одна коробка передач, ожидающая контроля или наладки, соответственно. Коробки передач попадают на станцию контроля по экспоненциальному закону распределения с параметром 2,25. На станции работают Kn контролеров, каждому из них на проверку коробки передач нужно 9±3 мин. Примерно 85% коробок передач проходят проверку успешно и попадают на склад, остальные 15% попадают на участок наладки, на котором работает один рабочий – наладчик. Наладка занимает 30±10 мин.



Каждый контролер получает заработную плату независимо от времени его полезной работы в течение рабочего дня. КПД контролера оценивается как отношение полезного времени его работы, связанного с контролем коробок передач, к продолжительности рабочего дня. Количество мест на стеллаже 1 ограничено десятью.

Задание

Определить, при каком количестве контролеров КПД максимально. Реализовать в виде оптимизационного эксперимента.

Тема 7. Моделирование работы шиномонтажного участка

Описание процесса

На участок могут приходить клиенты двух типов. Клиенты первого типа желают только отремонтировать колесо. Распределение интервалов их прихода 35±10 мин. Клиенты второго типа желают отремонтировать и сбалансировать колесо. Распределение интервалов их прихода 60±20 мин. Мастер обслуживает клиентов в порядке «первым пришел – первым обслужен». На ремонт уходит 15±6 мин., а на балансировку 7±2 мин.

На шиномонтажном участке оборудовано только одно место для обслуживания клиентов, менеджер рассматривает возможность установки второго дополнительного места.

Доходы от работы определяются количеством клиентов, обслуженных в течение рабочего дня (9 часов с часовым перерывом на обед), убытки определяются временем простоев (в отсутствие клиентов) и количеством не обслуженных клиентов в очереди.

Задание

Создать модель рынка клиентов шиномонтажного участка с использованием агентного моделирования.

Реализовать эксперимент на варьирование параметров для определения того, насколько целесообразна покупка оборудования второго места и прием на работу второго рабочего.

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

Тема 8. Моделирование работы мойки автомобилей

Описание процесса

В центре города автомойка обслуживает автомобили в различные времена года. Так, весной и осенью на автомойку прибывает около 5-8% от общего количества городского автопарка в день; для лета этот диапазон составляет 3-4%, зимой 1-2%. Время мойки автомобиля имеет нормальный

закон распределения с параметрами M=20 мин и σ=5,5. При этом автомобили ожидают своей очереди на мойку на специальной стоянке, число мест на которой ограничено. Если клиенты приезжают на мойку и не находят свободного места для ожидания, они уезжают и моют свой автомобиль в другом месте.

Прибыль от выполнения услуг по мойке и чистке салона варьируется от 100 до 400 руб. по равномерному закону. Однако, если автомобиль не находит свободного места на стоянке и уезжает, предприятие теряет в среднем около 200 руб.

Городской парк автомобилей, по статистике, насчитывает около 150000 автомобилей.

Задание

Создать модель рынка клиентов мойки с использованием агентного моделирования.

Определить, какое число мест на стоянке следует отвести для автомобилей, ожидающих мойки, с использованием оптимизационного эксперимента. Имитацию провести для пяти рабочих дней с учётом того, что рабочий день длится с 9:00 до 18:00.

Тема 9. Моделирование работы магазина автозапчастей

Описание процесса

Небольшой магазин автозапчастей состоит из трех отделов и одной кассы на выходе из магазина. По статистике, услугами магазина пользуются от 5-7% автовладельцев города. При этом общий парк автомобилей насчитывает около 240 тыс. единиц. Войдя в магазин, каждый из покупателей может обойти один или несколько прилавков. Вероятность обхода конкретного отдела приведена в таблице. Время, требуемое для обхода отделов, а также число покупок, выбранных у отдела, распределены равномерно. Подробная информация по каждому из отделов также приведена в таблице.

Прилавок	Вероятность покупок у отдела	Время, затраченное на покупки у отдела (сек)	Число покупок, сделанных у отдела (штук)
1	0.78	120±60	3±1
2	0.55	150±30	4±1
3	0.82	120±45	5±1

После того как товар отобран, покупатель становится в очередь к кассе. Уже стоя в очереди, покупатель может захотеть сделать еще 2±1 покупки. Время обслуживания покупателя в кассе пропорционально числу сделанных покупок, а для проверки одной покупки требуется 3 сек. После оплаты покупатель уходит.

Задание

Создать модель рынка клиентов магазина автозапчастей с использованием агентного моделирования.

Построить имитационную модель обслуживания покупателей в магазине, провести моделирование 8–часового рабочего дня, определить нагрузку кассира и максимальную длину очереди перед кассой.

Реализовать компьютерный эксперимент по определению оптимального числа корзин, одновременно находящихся у покупателей (вход в магазин покупателя возможен только с корзиной), при котором в очереди окажется не более 2 человек.

Тема 10. Модель использования оборудования на нескольких работах

Описание процесса

Выполнение кузовных работ определенного вида включает в себя длительный процесс восстановления кузова, заканчивающийся процессом окраски. Поскольку содержание окрасочной линии обходится довольно дорого, несколько бригад, каждая из которых занята работами по своему заказу, используют одну окрасочную линию, на которой одновременно можно окрашивать только один автомобиль. Бригада не может начать окраску другого автомобиля, пока не закончена окраска предыдущего.

Наименование операции	**Необходимое время выполнения (мин.)**
Ремонтные работы	200±30
Окраска	60±20

Таким образом, бригады трудятся в следующем режиме:

1) выполняют работы, предшествующие окраске;

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

3) используют окрасочную линию;

4) переходят к ремонту следующего автомобиля.

Задание

Требуется построить имитационную модель для определения такого количества бригад, при котором с одной стороны очередь минимальна, с другой – простои линии окраски минимальны.

Постройте имитационную модель и определите на ней число рабочих, использование которых приносит максимальную прибыль. Реализовать в виде оптимизационного эксперимента.

Статья затрат	**Размер затрат ($)**
Зарплата сборщика	3,75 $ в час
Содержание печи (за 8 – часовой рабочий день независимо от степени использования)	80 $
Стоимость материала, расходуемого на одно изделие	2 $
Стоимость готового изделия	7 $

Моделирование провести для 40 часов рабочего времени, предполагая, что в течение рабочего дня нет перерывов, а рабочие дни идут подряд без выходных.

За единицу модельного времени следует принять 1 мин.

Тема 11. Модель управления автотранспортным предприятием

Описание процесса

В автотранспортном предприятии 50 машин работают по 8 часов в день 5 дней в неделю. Любая из этих машин может в любой момент времени выйти из строя. В этом случае ее заменяют резервной машиной либо сразу (если есть резерв), либо по мере его появления. Тем временем сломанную машину отправляют в ремонтную мастерскую, где ее чинят и возвращают на предприятие, но уже в качестве резервной. В существующем замкнутом цикле движения машин можно выделить 4 фазы (см. рис.).

Предварительные прикидки по реорганизации производства показывают, что для организации надежной и выгодной работы следует использовать 59 машин: 50 из них используются непосредственно в производстве, 5 составляют резерв (так называемый «горячий» резерв), 2 могут одновременно ремонтироваться и 2 находиться в состоянии ожидания ремонта. Из таких предположений следует, что в ремонтном подразделении следует держать не менее двух рабочих.

Управляющий хочет знать, насколько оправданы такие прикидки, сколько рабочих следует нанять для работы в мастерской, сколько машин держать для использования в качестве резервных, чтобы ими можно было заменить собственные в случае отказа, какую платить за это арендную плату.

Опыт эксплуатации машин на аналогичных предприятиях показывает, что на ремонт сломанной машины уходит примерно 7 + 3 часа. Время безотказной работы машины (т. е. время от отказа до следующего отказа, так называемая «наработка на отказ») составляет примерно 157±25 часов и не зависит от того, собственные это машины или арендуемые.

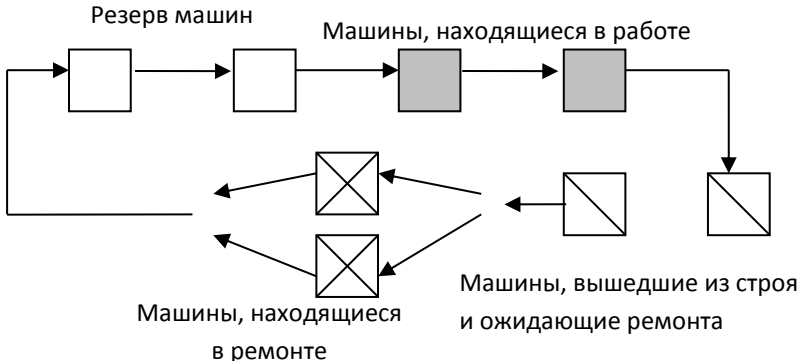


Рис. *Модель эксплуатации оборудования*

Плата за аренду машин не зависит от того, работают они или простаивают.

Почасовой убыток от снижения уровня производства при использовании менее 50 машин в производстве составляет примерно 20 $ на неработающую машину.

Оплата рабочих в мастерской – 3,75 $ в час.

За машины, находящиеся в резерве, надо платить по 30 $ в день.

Задание

Построить модель системы и исследуйте на ней организацию системы с целью определения минимальной стоимости эксплуатации. Реализовать в виде оптимизационного эксперимента.

Тема 12. Сравнение альтернативных вариантов систем обслуживания в автосервисе

Количество клиентов, обращающихся в автосервис, зависит от

погодных условий. Так, в ясную погоду в автосервис обращаются около 2-3% от общего количества автовладельцев; для должливой или снежной погоды этот диапазон составляет 6-8%. Общее количество автовладельцев в городе по статистике приблизительно равно 180 тыс. человек.

В течение рабочего дня в автосервисе работает 8 подъемников. К каждому подъемнику стоит очередь. В момент приезда каждый клиент встает в очередь, которая является самой короткой. После обслуживания клиент покидает автосервис.

При обслуживании на подъемнике выполняется одна из 5 различных видов работ. Время выполнения каждой из операций имеет нормальное распределение. Вероятности обслуживания клиента с выполнением определенной операции и соответствующее среднее время выполнения операций приведены в таблице.

Вид обслуживания	Вероятность обслуживания	Среднее время выполнения операции (мин)	Среднее квадратическое отклонение
1	0,10	45	15
2	0,19	75	20
3	0,32	100	35
4	0,24	150	45
5	0,15	300	75

Ни один из клиентов не требует выполнения более чем одного вида обслуживания за один визит в автосервис.

Руководство автосервиса хотело бы сократить время ожидания клиентов без приобретения дополнительных подъемников. Для этого предлагается организовать «быструю» очередь. При такой организации все приезжающие клиенты становятся в общую очередь, и, когда какой – либо подъемник освобождается, клиент, стоящий в очереди первым, идет к этому подъемнику.

Задание

Построить имитационную модель и сравнить на ней оба варианта организации обслуживания клиентов в автосервисе. Моделирование каждого варианта должно охватывать пять 6–часовых рабочих дней.

Тема 13. Реорганизация заправочной станции

Интервалы времени между	Суммарная относитель	Интервалы времени между	Суммарная относительна

прибытиями (сек)	ная частота	прибытиями (сек)	я частота
Меньше 0	0	400	0,81
100	0,25	500	0,9
200	0,48	600	1,0
300	0,69		

Интервалы времени между прибытиями автомашин на заправочную станцию характеризуются распределением, приведенным в таблице. Время обслуживания автомашин подчинено распределению, приведенному в таблице.

Время обслуживания (сек)	Суммарная относительная частота	Время обслуживания (сек)	Суммарная относительная частота
Меньше 100	0,0	500	0,77
200	0,06	600	0,83
300	0,21	700	1,0
400	0,48		

Заправочная станция оборудована пятью бензоколонками и открыта с 7 ч до 19 ч. В 19 ч. выключается свет. Машины, приехавшие после 19 ч., не обслуживаются, а машины, попавшие на заправку до 19 ч. и стоящие в очереди, должны быть обслужены.

Водитель машины останавливается на обслуживание, если он находит свободную колонку или колонку, освобождения которой ожидает не более одной машины. В противном случае он уезжает и тем самым уменьшает потенциально возможную прибыль заправочной станции.

Прибыль с одной обслуженной машины составляет в среднем $1. Служащий, работающий на заправочной станции, может обслуживать не только одну, но и несколько бензоколонок (в порядке очереди). Заработок служащего составляет $2,5 в час и выплачивается только за 12 – часовой рабочий день (обслуживание оставшихся машин после 19 ч не оплачивается). Постоянные расходы на бензозаправке составляют $75 в день. Владельцу колонки нужно определить, сколько служащих следует нанять для того, чтобы максимизировать дневную прибыль.

Задание

Построить имитационную модель системы и провести моделирование для каждого числа нанятых служащих в течение 5 рабочих дней. Определить оптимальное количество служащих на станции с

использованием оптимизационного эксперимента.

Тема 14. Обслуживание цистерн на нефтебазе

Описание процесса

Работа нефтебазы связана с заполнением цистерн горюче-смазочными материалами для их дальнейшей транспортировки. Имеется возможность заливать одновременно до трех цистерн. Цистерны, прибывающие на нефтебазу каждые 11 ±7 часов, могут быть одного из трех различных типов. Относительная частота прихода цистерн различных типов и требуемое время для их заливки приведены в таблице.

Тип цистерны	Относительная частота	Время заливки (ч)
1	0,25	18±2
2	0,55	24±3
3	0,20	36±4

Прибывшей цистерне любого типа для подхода к стоянке и отхода от нее требуются услуги тягача. На нефтебазе имеется один тягач, который транспортирует цистерну в один конец примерно за 1 час.

В этом регионе часты снежные бури, а в период бури для цистерны невозможен ни подход к стоянке, ни отход от нее. Продолжительность бури 4±2 часа, время между окончанием бури и началом следующей подчиняется экспоненциальному распределению со средним значением в 48 часов.

Грузоотправитель намеревается заключить контракт на перевозку ГСМ с этой нефтебазы. Он считает, что 5 цистерн второго типа могли бы полностью удовлетворить условия контракта. После заливки и отъезда от стоянки они должны ехать в пункт назначения, выгружать ГСМ, возвращаться на нефтебазу для новой заправки и т. д. График движения цистерн грузоотправителя предусматривает, что время их пребывания в пути (от нефтебазы до места назначения и обратно), включая выгрузку ГСМ, должно составлять 240 + 24 часа.

Задание

Построить модель, имитирующую работу нефтебазы, и определить на ней, как повлияет грузооборот на нефтебазе на график движения этой группы цистерн. Моделирование провести для 1 года перевозок и собрать на модели статистику времени отставания от графика.

Тема 15. Задача о запасных деталях

Описание процесса

В процессе эксплуатации технологического оборудования в любое время может произойти его поломка, связанная с выходом из строя какой-либо детали. Как только происходит поломка станка, его выключают, отказавшую деталь заменяют другой (если она есть), и станок вновь включают. Неисправные детали могут быть отремонтированы.

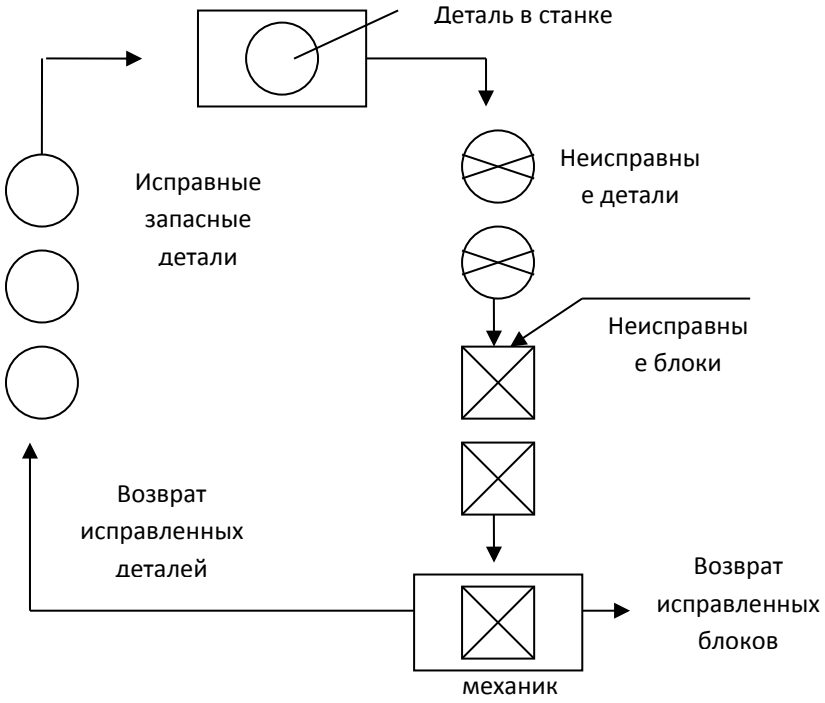


Рис. *Модель схемы организации замены деталей*

Наладчик станка отвечает за съем и установку деталей. Ремонтом занимается механик, в обязанности которого также входит ремонт других блоков. Эти блоки поступают в ремонтное подразделение в среднем с интенсивностью 1 блок за 9 ч (простейший поток). Время, требуемое на ремонт блока, 8 ±4 ч. Блоки имеют более высокий приоритет при ремонте, чем рассматриваемые детали.

Организация описываемой системы представлена следующим рисунком (см. рис.).

Время работы детали распределено нормально со средним 350 ч и стандартным отклонением 70 ч. Съем детали со станка занимает 4 ч, время установки новой детали – 6ч. Время ремонта детали распределено нормально со средним 8 ч и стандартным отклонением 0,5 ч.

Задание

Построить модель системы и использовать ее для определения полезного времени работы станка как функции от числа запасных деталей, имеющихся в системе.

Исследовать систему для случаев, когда запасных деталей нет, а также, когда имеются 1, 2 и 3 запасные детали. Для каждого случая выполнить прогон модели в течение 5 лет при условии 40 – часовой рабочей недели. Реализовать в виде эксперимента на варьирование параметрами.

Тема 16. Задача о складе запчастей

Описание процесса

На складе без открытого доступа любой водитель может получить запасную деталь. Он должен представить кладовщику лист запроса. После этого кладовщик ищет в хранилище деталь и возвращается с ней к столу выдачи. Затем происходит процедура выдачи, после чего водитель уходит. Если обслуживания ожидают несколько человек, то кладовщик может сэкономить время, забирая листки запроса сразу у нескольких водителей.

Задание

Построить имитационную модель выдачи деталей на складе, реализующую следующие условия:

Поток водителей к столу выдачи простейший с интенсивностью 5 человек в час.

Каждый водитель хочет получить ровно одну деталь.

Обслуживание посетителей идет в порядке очереди.

Число кладовщиков, работающих у стола выдачи, должно быть переменным.

Кладовщик одновременно берет листки запроса у нескольких посетителей, стоящих в очереди, но не более чем у четырех.

Если к моменту прихода посетителя свободны 2 или более кладовщиков, то его обслуживает тот из них, кто был свободен дольше других.

Временные характеристики модели:

• время, затрачиваемое на прохождение в один конец от стола выдачи до хранилища 1±0,5 мин.;

• время поиска в хранилище одной, двух, трех и четырех деталей распределено по нормальному закону со средним соответственно 3, 6, 9 и 12 мин. и стандартным отклонением, равным 20% от среднего;

• время оформления выдачи после возвращения кладовщика из хранилища 2±1 мин на человека. Определить на модели:

1) распределение времени, затраченного водителем на ожидание выдачи детали;

2) распределение числа листков запроса, забираемых кладовщиком перед уходом в хранилище.

Провести моделирование для случаев, когда у стола выдачи работают 3, 4 и 5 кладовщиков. Для каждого случая продолжать моделирование до тех пор, пока не будут полностью обслужены 100 посетителей.

Тема 17. Модель организации перевозок при нестационарных пассажиропотоках

Описание процесса

Поток пассажиров, приходящих на автобусную остановку, характеризуется переменной интенсивностью, заданной в таблице. В каждом из приведенных в таблице интервалов времени поток имеет характер простейшего.

Вместимость автобусов, циркулирующих на маршруте, – 30 человек. Если на маршруте циркулирует один автобус, то интервалы времени между приходами его на остановку определяются величиной 40±10 мин. Увеличение числа автобусов на маршруте в п раз соответственно уменьшает интервалы между приходами автобусов на остановку в п раз. Загруженность приходящего автобуса определяется в процентах от его вместимости эмпирической зависимостью $(10*\lambda_i/n)\%$. Например, для i=3 (см. таблицу) автобус, приходящий на остановку, будет загружен на 100%, если на линии работает 1 автобус, на 50%, если на линии 2 автобуса, и т. д.

Интервал времени (i)	Время суток (ч)	Интенсивность пассажиропотока (человек/мин.) (λ_i)
1	0 – 4	0,5
2	4 – 8	1

3	8 – 12	10
4	12 – 16	5
5	16 – 20	10
6	20 – 24	3

Перевозка одного пассажира приносит доход в размере $0,5. Перевозка «пустых мест» (отсутствие пассажиров на остановке) приносит убытки в таком же размере на одно пустое место. Невозможность перевозки пассажира по причине переполненности автобуса связана с упущенной выгодой, так же оцениваемой величиной в $0,5 на пассажира, который не смог сесть в автобус.

Суточное движение автобусов организовано в три смены по 8 часов.

Задание

Построить имитационную модель и на ее основе определить расписание смен и количество автобусов, работающих в сменах, при котором перевозка пассажиров будет наиболее рентабельной. Прогон каждого варианта организации перевозок проводить для 10 суток. Реализовать в виде оптимизационного эксперимента.

Тема 18. Составление оптимального расписания работ

Описание процесса

Поток покупателей, приходящих в автомагазин, имеет характер простейшего, но характеризуется интенсивностью, которая изменяется в зависимости от дней недели (см. таблицу).

Время обслуживания покупателя продавцом – 4±2 мин. Покупатель, пришедший в магазин и обнаруживший, что все продавцы заняты обслуживанием, ожидает освобождения продавца не более 2±1 мин. Если за это время ни один из продавцов не освободится, покупатель уходит из магазина без покупки. Это событие связано с упущенной выгодой для владельца магазина, поэтому он заинтересован в определении такого количества продавцов, при котором такие уходы были бы сведены к минимуму. Величина упущенной выгоды на одного ушедшего покупателя составляет в среднем $2. Простой продавца в течение 1 часа из–за отсутствия покупателей связан с убытками в размере $3.

День недели	Интенсивность потока (покупатель/мин.)	Число продавцов, обслуживающих покупателей
Понедельник	0,2	?
Вторник	0,2	?

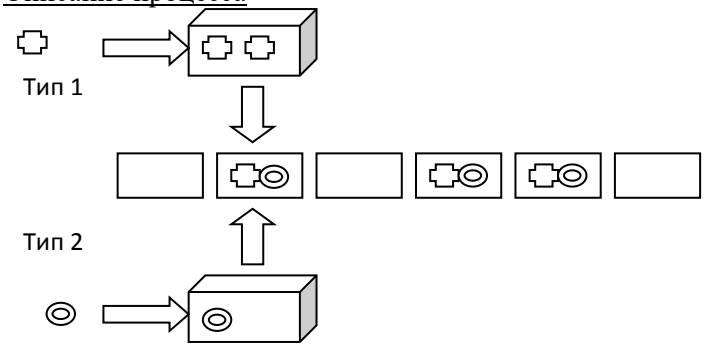
Среда	0,25	?
Четверг	0,5	?
Пятница	0,75	?
Суббота	1	?
Воскресенье	0,5	?

Задание

Построить имитационную модель обслуживания клиентов в магазине и с ее помощью определить для каждого дня недели количество продавцов, при котором размеры дневной упущенной выгоды и убытки от простоя сводятся к минимуму. В результате решения этой (первой) задачи вы должны заполнить третий столбец приведенной таблицы.

Тема 19. Моделирование работы комплектовочного конвейера

Описание процесса



На комплектовочный конвейер сборочного цеха каждые T1 минут поступают N1 деталей первого типа и каждые T2 минут – N2 деталей второго типа. Изделие комплектуется из N3 деталей каждого типа. Комплектация начинается только при наличии деталей обоих типов в необходимом количестве (см. рис.).

Конвейер движется ритмично с шагом T3 минут. При отсутствии необходимого количества деталей секция конвейера перемещается пустой («холостой ход»).

Определить целесообразность перехода на другие режимы работы конвейера, варьируя такими параметрами:

- размерами секции – количеством деталей каждого типа, из которых комплектуется изделие (возможны дополнительные варианты – по N4 и N5 изделий);
- шагом конвейера (возможны дополнительные варианты – T4 и

T5 минут).

Задание

Оценить вероятность «холостого хода», средние и максимальные длины очередей каждого типа изделий, по каждому из вариантов, заданий, приведенных в таблице. Реализовать с использованием эксперимента на варьирование параметров.

Вариант	Параметры									
	T_1	N_1	T_2	N_2	N_3	T_3	N_4	N_5	T_4	T_5
1	5±1	5	20±5	20	10	10	20	5	20	5
2	10±3	8	40±10	32	16	20	32	8	36	9
3	15±5	12	60±12	36	24	30	36	12	45	15
4	12±4	6	48±10	24	24	24	24	6	45	15

Тема 20. Моделирование работы цехового и центрального складов

Описание процесса

Детали, необходимые для работы цеха, находятся на цеховом и центральном складах. На цеховом складе может храниться до n комплектов деталей, потребность в которых возникает через A ± B минут и составляет один комплект. В случае понижения уровня запасов до k комплектов на протяжении C минут формируется запрос на пополнение запасов цехового склада до полного объёма в n комплектов. Запрос посылают на центральный склад, где на протяжении D ± E минут комплектуются детали и через F ± G минут доставляются в цех. Следующий запрос на пополнение запасов может подаваться только после выполнения предыдущего.

Хранение одного комплекта на цеховом складе требует S1 единиц стоимости за единицу времени. Штраф за задержку поставки комплекта составляет S2 единиц стоимости за единицу времени.

Задание

Определить, при каких значениях n и k достигается максимальная экономическая эффективность работы склада. Реализовать с использованием оптимизационного эксперимента.

Выполнить анализ аналитических методов решения задачи управления запасами.

Исходные данные приведены в таблице.

Параметры

n	A	B	k	C	D	E	F	G	S_1	S_2
20	60	10	3	60	80	20	70	10	10	150

Тема 21. Моделирование работы сервисного центра

Описание процесса

На автосервисное предприятие (АСП) согласно закону Эрланга второго порядка со средним временем прибытия 13 мин прибывают автомобили для технического обслуживания (35 % автомобилей) и ремонта (65 % автомобилей). На АСП есть два бокса для технического обслуживания и три бокса для ремонта. Выполнение простого, средней сложности и сложного ремонтов – равновероятно.

Время и стоимость выполнения работ по техническому обслуживанию и ремонту зависит от категории выполняемых работ и представлены в табл.

Категория работ	Время ремонта, мин	Стоимость ремонта, руб
Техническое обслуживание	Равномерно распределено в интервале 10-55	Равномерно распределено в интервале 100–400
Простой ремонт	Равномерно распределено в интервале 12-45	Равномерно распределено в интервале 50–450
Ремонт средней сложности	Нормально распределено со средним 45 и среднеквадратичным отклонением 5	Равномерно распределено в интервале 100–1400
Сложный ремонт	Равномерно распределено в интервале 80–150	Равномерно распределено в интервале 350–2550

После технического обслуживания 12 % автомобилей поступают для выполнения ремонта средней сложности.

Задание

Промоделировать функционирование АСП на протяжении двух недель. Реализовать оптимизационный эксперимент с двумя факторами по определению такого оптимального количества боксов для ТО и ремонта, при котором прибыль АСП в течение этих двух недель будет максимальной.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ivanov, D. Operations and supply chain simulation with AnyLogic 7.2 [Текст]: – Berlin School of Economics and Law, 2017. – 97 с.

2. Боев, В.Д. Компьютерное моделирование: Пособие для курсового и дипломного проектирования [Текст] / В.Д. Боев, Д.И. Кирик, Р.П. Сыпченко — СПб.: ВАС, 2011. — 348 с.

3. Боев, В.Д. Компьютерное моделирование: Пособие для практических занятий, курсового и дипломного проектирования в AnyLogic7:. — СПб.: ВАС, 2014. — 432 с.

4. Григорьев, И. AnyLogic за три дня: Практическое пособие по имитационному моделированию [Электронный ресурс] / The AnyLogic Company. – 2017. – https://www.anylogic.ru/upload/al-in-3-days/anylogic_in_three_days(rus).pdf.

5. Карпова, И.П. Имитационное моделирование управления в групповой робототехнике на основе многоагентного подхода: Метод. указания к практическим занятиям [Текст]. – М., 2014. – 32 с.

6. Каталевский, Д.Ю. Основы имитационного моделирования и системного анализа в управлении: учебное пособие; 2-е изд., перераб. и доп. [Текст] / Д.Ю. Каталевский. — М.: Издательский дом «Дело» РАНХиГС, 2015. — 496 с., ил.

7. Маликов, Р. Ф. Практикум по имитационному моделированию сложных систем в среде AnyLogic 6 [Текст]: учеб. пособие / Р. Ф. Маликов. – Уфа: Изд-во БГПУ, 2013. – 296с.

Подписано в печать 10.05.2018 г.
Бумага офсетная. Печать ризографическая.
Формат 60x84 1/16. Гарнитура «Times New Roman». Усл. печ. л. 8,19
Уч.-изд. л. 5,46. Тираж 50 экз. Заказ 991

Отпечатано в типографии
Издательско-полиграфического центра
Набережночелнинского института
Казанского (Приволжского) федерального университета

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