А.Л. Королев, Н.Б. Паршукова

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ В СРЕДЕ ПРОГРАММНОГО КОМПЛЕКСА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ANYLOGIC

Учебное пособие

Челябинск

2024

Министерство просвещения РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования «Южно-Уральский

государственный гуманитарно-педагогический университет»

А.Л. Королев, Н.Б. Паршукова

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ В СРЕДЕ ПРОГРАММНОГО КОМПЛЕКСА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ANYLOGIC

Учебное пособие

Челябинск

2024

УДК 681.4(021)
ББК 32.973:2-018я73
К 68

Королев, А.Л. Компьютерное моделирование процессов и систем в среде программного комплекса имитационного моделирования AnyLogic: учебное пособие / А.Л. Королев, Н.Б. Паршукова; Министерство просвещения РФ; Южно-Уральский государственный гуманитарно-педагогический университет – Челябинск: Изд-во Южно-Урал. гос. гум. пед. ун-та, 2024. – 196 с. – ISBN 978-5-907869-04-2. – Текст: непосредственный.

Пособие посвящено одной из важных информационных технологий – имитационному компьютерному моделированию процессов, объектов и систем. Предназначено для студентов бакалавриата, обучающихся по следующим специальностям:

1. Направление: 44.03.05 «Педагогическое образование» по профилям: «Информатика–Английский язык», «Информатика–Математика» при изучении курсов «Компьютерное моделирование», «Виртуальные лаборатории в школьном курсе информатики».

2. Направление: 09.03.02 «Информационные системы и технологии», профиль: «Информационные технологии в образовании» при изучении курсов «Моделирование систем», «Автоматизированные лабораторные комплексы».

Цель пособия – сформировать у студентов способности разрабатывать компьютерные модели процессов и систем на основе современной технологии моделирования с использованием основных естественно-научных законов, а также проводить модельные исследования и компьютерные эксперименты в области профессиональной деятельности. Пособие содержит теоретические положения моделирования, материалы для лабораторных работ по построению различных типов имитационных моделей. Практическая часть курса построена на доступном для образовательных целей программном обеспечении.

ISBN 978-5-907869-04-2

Рецензенты:
Т.В. Карпета, канд. физ.-мат. наук, доцент
О.А. Дмитриева, канд. пед. наук, доцент

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4

ИНСТРУМЕНТАЛЬНАЯ СИСТЕМА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ANYLOGIC 7

ГЛАВА 1. МОДЕЛИ ДИСКРЕТНЫХ СОБЫТИЙ

1.1. AnyLogic-модель сервера 19

1.2. AnyLogic-модель отделения банка 48

1.3. AnyLogic-модель билетной кассы 61

1.4. AnyLogic-модель сборки изделия 80

1.5. AnyLogic-модель работы поликлиники 98

ГЛАВА 2. ANYLOGIC – ПЕШЕХОДНЫЕ МОДЕЛИ

2.1. AnyLogic-модель работы турникетов в школе 111

2.2. AnyLogic-модель движения пешеходов в замкнутом пространстве 122

ГЛАВА 3. МОДЕЛИ СИСТЕМНОЙ ДИНАМИКИ

AnyLogic-модель распространения продукта 132

ГЛАВА 4. АГЕНТНЫЕ МОДЕЛИ

AnyLogic-модель внутризаводской логистики 162

ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ 189

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 195

ВВЕДЕНИЕ

Известно, что моделирование является общенаучным методом изучения законов окружающего мира, свойств объектов и систем самой различной природы или новых объектов и систем. Моделирование как метод – мощный инструмент науки и техники. Как показывает опыт, активное участие в моделировании и проведении модельных компьютерных экспериментах вырабатывает более глубокое понимание сути протекающих процессов и наблюдаемых явлений.

Развитие компьютерных технологий предоставляет в профессиональной деятельности новые возможности с максимальной степенью наглядности и оперативности получить и представить информацию о свойствах объектов и характере протекающих в них процессов. Применение компьютерных методов моделирования, проведение компьютерных экспериментов способствуют углублению и расширению знаний в конкретной предметной области о процессах, протекающих в конкретных системах, существующих или проектируемых.

Долгое время достаточно сильным препятствием в этом направлении была необходимость создания моделей средствами какой-либо системы программирования. В этом случае собственно моделирование отодвигалось на второй план, так как разработка модели путем непосредственного программирования с получением в итоге программного комплекса требует значительных затрат времени и средств.

В этом смысле компьютерное моделирование на основе специализированных инструментальных программных комплексов предоставляет возможность построить процесс моделирования, который будет принципиально отличаться тем, что модель создается средствами быстрой разработки, визуальными методами, с автоматическим выбором численных методов и генерацией программы. Это позволяет использовать технологию компьютерного моделирования для решения задач по ряду дисциплин профессионального и естественно-научного цик-

лов. Таким образом, инструментальные программные комплексы моделирования, дающие возможность конструирования моделей с наглядным представлением результатов при минимальной потребности в программировании, имеют особую ценность.

Визуализация – уникальная возможность компьютерной технологии моделирования, так как показать невидимое способны только компьютерные модели. Моделирование составляет неотъемлемую часть не только современной науки, но и образования, причем по важности оно приобретает первостепенное значение. На современном этапе термин «моделирование» в большинстве случаев означает «компьютерное моделирование», так как применение компьютеров существенно расширило возможности и породило новые технологии моделирования.

Цель настоящего пособия – дать представление о современных методах построения, реализации и исследования моделей объектов, процессов и систем разнообразной природы. Расширить представления студентов о моделировании как о методе научного познания, познакомить с методологией моделирования, научить применять компьютер как средство познания и научных исследований в различных областях практической деятельности. Научить студентов применять методы моделирования для решения конкретных задач, сформировать навыки в области моделирования процессов и систем различной природы. Таким образом, пособие позволяет решить следующие задачи подготовки специалистов:

- знакомство студентов с современными методами и технологиями построения моделей и проведения модельных экспериментов средствами специализированных программных комплексов;
- обучение эффективному применению моделирования и модельного эксперимента;
- развитие творческого потенциала будущего специалиста, необходимого ему для дальнейшего самообучения в условиях непрерывного развития и совершенствования информационных технологий.

В результате изучения представленного в пособии материала студенты должны быть способны использовать основные законы естественно-научных дисциплин в профессиональной деятельности, применять методы математического анализа и моделирования, теоретического и экспериментального исследования,

проводить моделирование процессов и систем. Результатом формирования компетенции в области моделирования будет являться готовность к участию в постановке и проведению экспериментальных исследований; способность обосновывать правильность выбранной модели, сопоставляя результаты экспериментальных данных и полученных решений; способность использовать математические методы обработки, анализа и синтеза результатов профессиональных исследований; способность применять естественно-научные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности, а так же применение математических моделей, методов и средств проектирования при моделировании систем.

Пособие будет полезно студентам направлений 44.03.01 Педагогическое образование, профиль «Информатика», 44.03.05 Педагогическое образование, профиль «Информатика (с дополнительной специальностью)», 09.03.02 Информационные системы и технологии.

Настоящее пособие является дополнением и продолжением учебных пособий:

Королев, А.Л. Компьютерное моделирование объектов, процессов и систем: учебно-практическое пособие / А.Л. Королев, Н.Б. Паршукова. – Челябинск: Изд-во Южно-Урал. гос. гуманитар.-пед. ун-та, 2022. – 308 с. – ISBN 978-5-907611-11-5 и Королев, А.Л. Компьютерное моделирование объектов, процессов и систем: учебное пособие / А.Л. Королев, Н.Б Паршукова. – Челябинск: Изд-во Южно-Урал. гос. гуманитар.-пед. ун-та, 2020. – 329 с. – ISBN 978-5-907409-15. При создании данного пособия были использованы материалы, представленные в работах [1–8].

За время, прошедшее с момента написания данных пособий, изменились требования по составу компетенций и их содержание в курсах «Компьютерное моделирование» и «Моделирование систем» и появились новые программные комплексы и технологии. Таким образом, переработка и обновление учебно-методического пособия становятся актуальной задачей.

ИНСТРУМЕНТАЛЬНАЯ СИСТЕМА МОДЕЛИРОВАНИЯ ANYLOGIC

Система AnyLogic, разработанная компанией XJ Technologies (Россия 1999 г., сейчас это фирма AnyLogic), это среда компьютерного имитационного моделирования общего назначения. Это комплексный инструмент, охватывающий основные в настоящее время направления моделирования: дискретно-событийное моделирование, моделирование системной динамики и агентное моделирование. Использование AnyLogic дает возможность оценить эффект принимаемых решений в сложных системах реального мира.

В AnyLogic разработчик может гибко использовать различные уровни абстрагирования и различные стили и концепции и смешивать их при создании одной и той же модели (рис. 1).

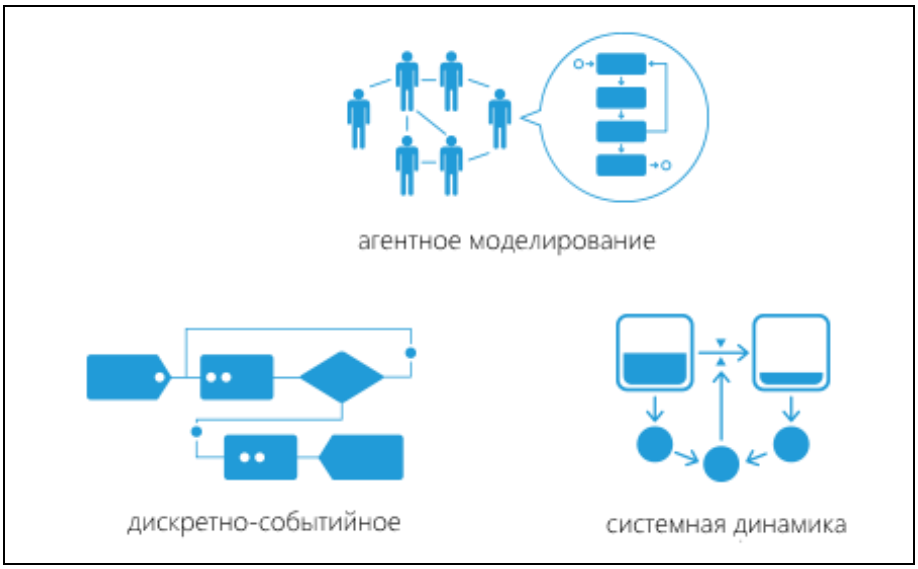


Рис. 1. Парадигмы моделирования в AnyLogic

AnyLogic, так же как и AnyDynamics Rand Model Designer, использует объектно-ориентированный подход к представлению сложных систем. Этот подход позволяет простым и естественным образом организовать и представить структуру сложной системы с помощью иерархии абстракций.

Например, на некотором уровне абстракции автомобиль можно считать неким единым объектом. Но более детально его можно представить как совокупность взаимодействующих подсистем: двигателя, рулевого управления, тормозной системы и т. п. Каждая из этих подсистем может быть представлена, если это необходимо, своей структурой взаимодействующих подсистем.

При построении модели в среде AnyLogic не предусмотрена возможность использования никаких других средств, кроме средств визуальной разработки (введения состояний и переходов стейтчарта, введения пиктограмм переменных и т.п.), задания численных значений параметров, аналитических записей соотношений переменных и аналитических записей условий наступления событий. Основной парадигмой, принятой в AnyLogic при разработке моделей, является визуальное проектирование – построение с помощью графических объектов и пиктограмм иерархий структуры и поведения активных объектов.

Решения, которые позволяет принять AnyLogic-моделирование:

1. Управление сложными современными производственными процессами и финансовыми операциями требует оперативности, планирования и оценки результата:

- оценить последствия принятия нескольких альтернативных решений и выбрать лучшее;
- использовать прогностическую модель для оперативного, среднесрочного или стратегического планирования;
- оценить работу процесса «в виртуальном мире», проверить изменения до их реализации.

2. По построенной программной модели, имитирующей работу реальной системы, можно исследовать производительность системы, ее «узкие места», оценить влияние параметров (например, оценить, куда направить ограниченный объем капиталовложений).

3. Проводить эксперименты «что, если...», оценивать альтернативные варианты и выбирать технологию, дающую наилучшие результаты.

4. Проводить моделирование и компьютерную имитацию функционирования сложных систем совместно с их визуализацией, что становятся повседневным современным средством поддержки и обоснования принятия решений.

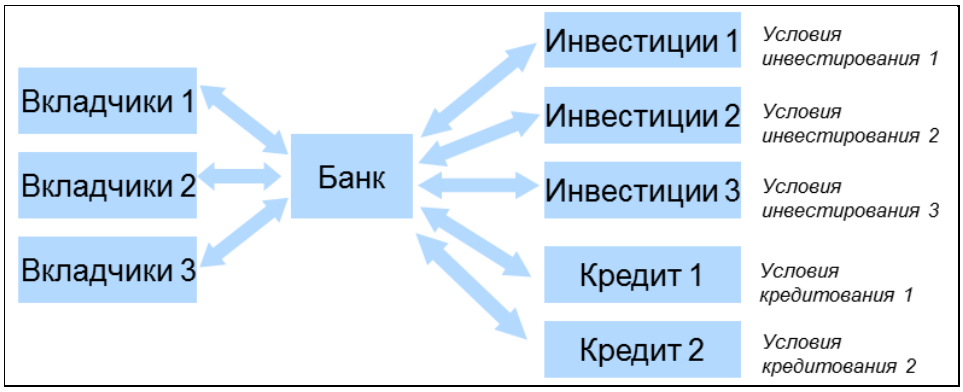


Рис. 2. Модель финансовых потоков банка

Например, представленная на рис. 2 схема, по которой будет построена модель банка, позволяет:

- оперативно оценить положение банка в будущем, учитывая сроки погашение кредитов и внесение вкладов, произвести оценку разрывов ликвидности;
- определить наилучшую схему развития банка, учитывая прибыльность и устойчивость к рискам;
- оценить последствия рисков непогашения кредитов или кризиса доверия к депозитам;
- моделировать финансовые потоки, что позволяет улучшить управляемость, получить точные и своевременные оценки, выявить перспективные направления и оперативно оценить разрывы ликвидности;
- моделировать фондовые рынки и поведение инвесторов на фондовых рынках по известным моделям составить инвестиционную стратегию, повысить отдачу от использования капитала и оценить общую ситуацию на рынке;
- оперативно оценить положение банка в будущем, учитывая сроки погашения кредитов и внесение вкладов, произвести оценку разрывов ликвидности;
- определить наилучшую схему развития банка, учитывая прибыльность и устойчивость к рискам;
- оценить последствия рисков непогашения кредитов или шока (кризиса доверия) депозитов.

Для моделирования систем с дискретными событиями существует множество пакетов, облегчающих разработку дискретно-событийных моделей и проведение экспериментов с моделями в этой традиционной области моделирования.

В первую очередь это GPSS (General Purpose Simulation System), который стал революцией более 50 лет назад, задав парадигму моделирования в этой области в виде блоков и транзактов. Транзакты отображают динамические объекты моделирования (заявки), а блоки — объекты, обрабатывающие эти заявки. Большинство других инструментов моделирования (Arena, Extend, ProModel, SimProcess и др.) также используют эту парадигму.

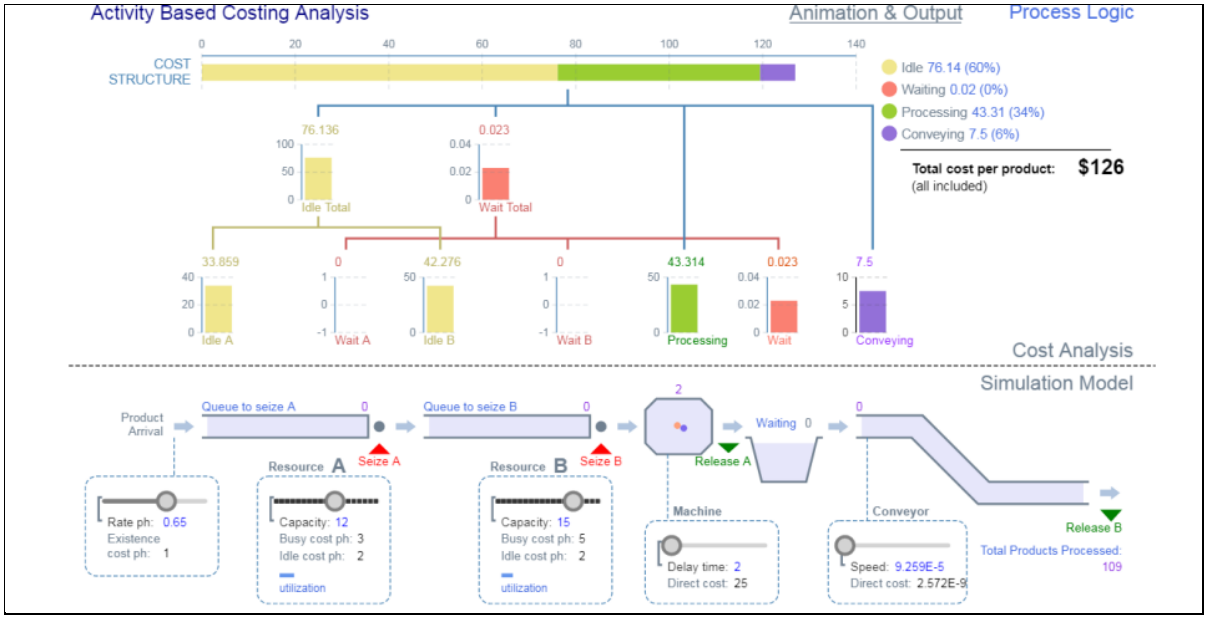


Рис. 3. AnyLogic-модель дискретных событий (производство)

Моделирование системной динамики — это методология изучения и моделирования систем, характеризующихся циклами обратных связей в сложных взаимных причинных зависимостях их параметров. Математически эти системы описываются системами дифференциальных уравнений, приведенных к форме Коши. Эти модели применяются для корпоративного планирования и анализа политики управления корпорацией, политик управления социальными и экономическими системами, в экологии и т. п. Джей Форрестер американский инженер и системолог, разработчик теории системной динамики, в 1970-х годах он разработал модели «Мир-1» и «Мир-2», нацеленные на выработку сценариев развития всего человечества в его взаимоотношении с биосферой. Так родилось первое поколение компьютерных моделей, предназначенных для исследования долгосрочных тенденций мирового развития. Он опубликовал книгу «Мировая дина-

мика», в которой обобщил свой вклад в создание первых компьютерных моделей, анализирующих глобальную систему. Он заложил методологические принципы системной динамики с использованием графического представления причинных зависимостей переменных, в виде так называемых «stock and flow diagrams», которые и сейчас являются основой инструментов моделирования, конкурирующих на рынке: VisSim, PowerSim, Stella, ModelMaker и др.

Моделирование динамических систем – это область моделирования систем управления, физических, механических систем, систем обработки сигналов и т.п. Широкое применение в этой области имеет пакет моделирования Simulink, являющийся составной частью пакета Matlab. Пакет имеет библиотеку предопределенных блоков, из которых можно методом «drug-and-drop» строить блочную структуру, аналогичную блочной структуре моделей, когда-то давно, лет 60 назад набираемых для их решения на аналоговых вычислительных машинах из интеграторов, усилителей, сумматоров, источников сигналов и т.п.

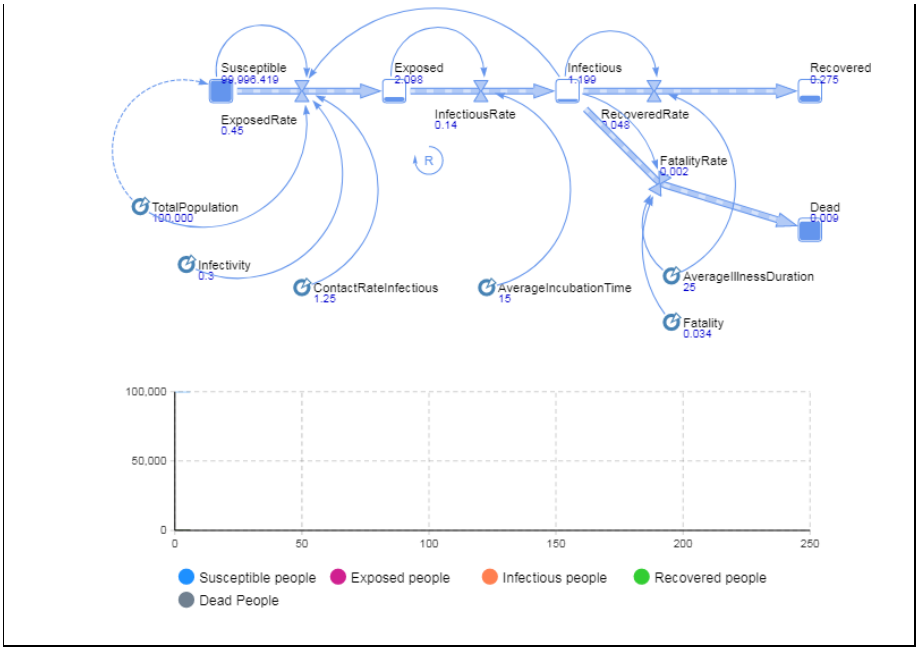


Рис. 4. AnyLogic-модель системной динамики

Агентное моделирование. Под агентом понимается активный объект, обладающий поведением и имеющий возможность взаимодействия с другими аген-

тами и со средой. Многоагентное моделирование позволяет вывести характеристики целого (множества агентов) из совокупности локальных поведений и характеристик отдельных активных элементов целого, распределенных в среде.

Моделирование многоагентных систем используется в анализе социальных процессов, процессов урбанизации и даже при исследовании рынка в анализе предпочтений различных социальных групп или корпораций, выступающих как агенты со своим поведением.

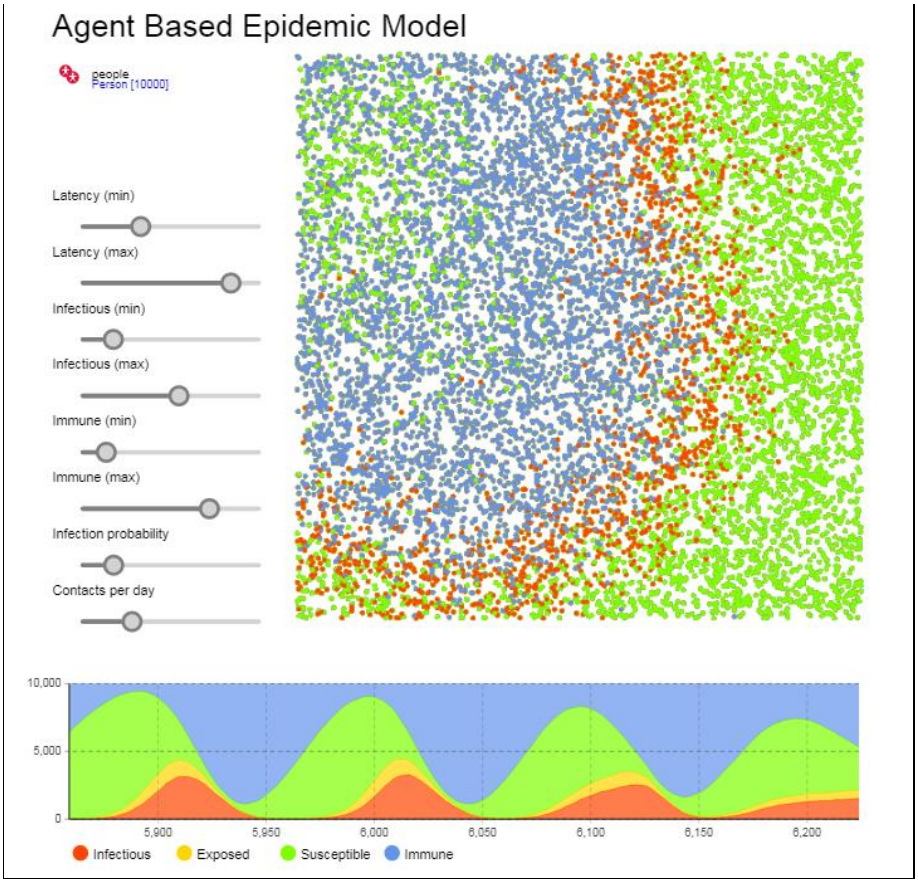


Рис. 5. AnyLogic – агентная модель развития эпидемии

- Если модель предполагает много индивидуальных объектов – используйте **агентное моделирование**.
- Если вы владеете информацией только о глобальных зависимостях – используйте **системную динамику**.
- Если система может быть описана как процесс – используйте **дискретно-событийное** моделирование.

• Если системе присущи все эти особенности – комбинируйте различные методы.

Рис. 6. Применение различных методов моделирования

На рис. 6 показано, как производство, распределение и рынок могут быть совмещены в одной модели с применением трех подходов одновременно. Дискретно-событийная модель описывает процессы на каждом складе. На уровне распределительной сети склады действуют как независимые агенты. И, наконец, управляющий системой рынок моделируется с помощью системной динамики. Модель детально отражает все элементы и процессы этой системы. С AnyLogic разработчик не ограничен одним методом моделирования. Используя подходящий метод или их комбинацию, он может создать оптимальную для решения конкретной проблемы имитационную модель.

Таким образом, идеи и методы, направленные на управление сложностью, выработанные в последние десятилетия в области создания программных систем, позволяют разработчикам моделей в среде AnyLogic организовать мышление, структурировать разработку и, в конечном счете, упростить и ускорить создание моделей.

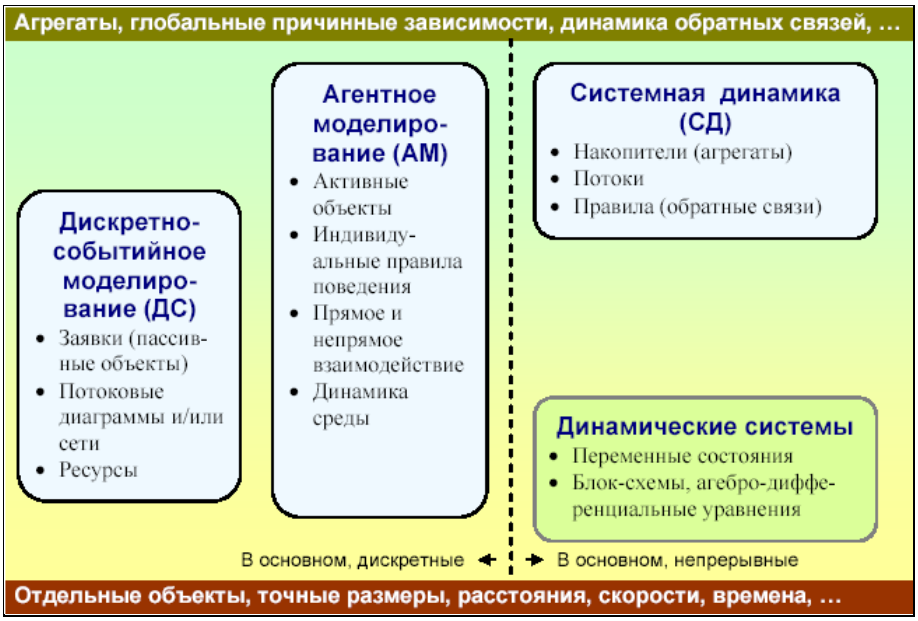


Рис. 7. Методы моделирования, реализуемые в AnyLogic

Другой базовой концепцией AnyLogic является представление модели как набора взаимодействующих параллельно функционирующих активностей. Такой подход к моделированию интуитивно очень понятен и естественен во многих приложениях, поскольку системы реальной жизни состоят из совокупности активностей, взаимодействующих с другими объектами. Активный объект AnyLogic — это объект со своим собственным функционированием, взаимодействующий с окружением. Он может включать в себя любое количество экземпляров других активных объектов. Активные объекты могут динамически порождаться и исчезать в соответствии с законами функционирования системы. Так могут моделироваться социальные группы, холдинги компаний, транспортные системы и т. п.

Графическая среда моделирования AnyLogic поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов с моделью, включая различные виды анализа – от анализа чувствительности до оптимизации параметров модели относительно некоторого критерия. В результате AnyLogic не ограничивает пользователя одной-единственной парадигмой моделирования, что является характерным фактически для всех инструментов моделирования, существующих сегодня. В AnyLogic разработчик может гибко использовать различные уровни абстрагирования, различные стили и концепции, строить модели в рамках той или иной парадигмы и смешивать их при создании одной и той же модели, использовать ранее разработанные модули, собранные

в библиотеки, дополнять и строить свои собственные библиотеки модулей. При разработке модели на AnyLogic можно использовать концепции и средства из нескольких «классических» областей моделирования, например, в агентной модели использовать методы системной динамики для представления изменений состояния среды или в непрерывной модели динамической системы учесть дискретные события.

Например, анализ IT-инфраструктуры компании (анализ производительности серверов, узких мест локальной сети и т. п.). Он легко производится с помощью методов дискретного событийного моделирования, имеет немного пользы, если в модели не отражено влияние возможных изменений параметров этой инфраструктуры на бизнес-процессы и, в конечном счете, на прибыль компании, а такая связь в модели не может быть реализована только средствами дискретно-событийного моделирования. В AnyLogic легко строятся подобные модели с требуемым уровнем адекватности, позволяющие ответить на многие вопросы, интересующие исследователя. Богатые возможности анимации и визуального представления результатов в процессе работы модели позволяют понять суть процессов, происходящих в моделируемой системе, упростить отладку модели. Удобный интерфейс и многочисленные средства поддержки разработки моделей в AnyLogic делают не только использование, но и создание компьютерных имитационных моделей в этой среде моделирования доступным даже для начинающих.

В AnyLogic существуют две фазы имитационного моделирования – разработка модели и ее анализ. Разработка модели выполняется в среде редактора AnyLogic, анализ модели происходит в среде исполнения. В каждой фазе существуют свои средства управления. Переход из одной фазы в другую производится очень легко. Можно многократно использовать переход между фазами редактирования и исполнения модели при разработке модели.

В AnyLogic основным структурным блоком при создании моделей являются классы активных объектов. Использование активных объектов является естественным средством структуризации модели сложных систем: мир состоит из множества параллельно функционирующих и взаимодействующих между собой сущностей. Различные типы этих сущностей и представляют разные активные объекты.

Чтобы создать модель AnyLogic, нужно создать классы активных объектов (или использовать объекты библиотек AnyLogic). Определение активного объекта задает шаблон, и отдельные объекты, построенные в соответствии с этим шаблоном (экземпляры активного объекта), могут использоваться затем как элементы других активных объектов. В любой класс могут быть включены несколько экземпляров других классов, которые могут различаться своими параметрами. Всегда один класс в модели является корневым.

Каждый активный объект имеет структуру (совокупность включенных в него активных объектов и их связи), а также поведение, определяемое совокупностью переменных, параметров, стейтчартов и т. п. Каждый экземпляр активного объекта в работающей модели имеет свое собственное поведение. Он может иметь свои значения параметров, функционирует независимо от других объектов, взаимодействуя с ними и с внешней средой.

При построении модели в среде AnyLogic не предусмотрена возможность использования никаких других средств, кроме средств визуальной разработки (введения состояний и переходов стейтчарта, введения пиктограмм переменных и т. п.), задания численных значений параметров, аналитических записей соотношений переменных и аналитических записей условий наступления событий. Основной парадигмой, принятой в AnyLogic при разработке моделей, является визуальное проектирование – построение с помощью графических объектов и пиктограмм иерархий структуры и поведения активных объектов.

Основным средством спецификации поведения объектов в AnyLogic являются переменные, таймеры и стейтчарты. Переменные отражают изменяющиеся характеристики объекта. Таймеры можно взводить на определенный интервал времени и по окончании этого интервала выполнять заданное действие. Стейтчарты позволяют визуально представить поведение объекта во времени под воздействием событий или условий, они состоят из графического изображения состояний и переходов между ними. Любая сложная логика поведения объектов модели в AnyLogic может быть выражена с помощью комбинации стейтчартов, дифференциальных и алгебраических уравнений, переменных, таймеров и программного кода на Java. Алгебраические и дифференциальные уравнения, как и логические выражения, записываются в AnyLogic аналитически.

Интерпретация любого числа параллельно протекающих процессов в модели AnyLogic скрыта от пользователя. Никаких календарей событий разработчик модели на AnyLogic не ведет, отслеживание событий во всех процессах, определенных в модели, выполняется системой автоматически.

Понятие модельного времени является базовым в системах имитационного моделирования. **Модельное время** – это условное логическое время, в единицах которого определено поведение всех объектов модели.

В моделях AnyLogic модельное время может изменяться либо непрерывно, если поведение объектов описывается дифференциальными уравнениями, либо дискретно, переключаясь от момента наступления одного события к моменту наступления следующего события, если в модели присутствуют только дискретные события. Моменты наступления всех планируемых событий в дискретной модели исполнительная система хранит в так называемом календаре событий, выбирая оттуда наиболее раннее событие для выполнения связанных с ним действий. Значение текущего времени в моделях AnyLogic может быть получено обращением к функции **getTime ()**.

Единицу модельного времени разработчик модели может интерпретировать как любой отрезок времени: секунду, минуту, час или год. Важно только, чтобы все процессы, зависящие от времени, были выражены в одних и тех же единицах. При моделировании физических процессов все параметры и уравнения должны быть выражены в одной и той же системе физических величин. Например, все физические величины выражаются в системе СИ, в которой единица времени – *секунда*, единица длины – *метр*, единица массы – *килограмм*.

Интерпретация модели выполняется на компьютере. **Физическое время***,* затрачиваемое процессором на имитацию действий, которые должны выполняться в модели в течение одной единицы модельного времени, зависит от многих факторов. Поэтому, конечно, единица физического и единица модельного времени не совпадают.

В AnyLogic приняты два режима выполнения моделей: режим виртуального времени и режим реального времени. В режиме виртуального времени процессор работает с максимальной скоростью без привязки к физическому времени. Этот режим используется для факторного анализа модели, набора статистики, оптимизации параметров модели и т. п.

В режиме реального времени пользователь задает связь модельного времени с физическим временем, то есть устанавливается ограничение на скорость работы процессора при выполнении модели. В этом режиме задается количество единиц модельного времени, которые должны выполняться процессором в одну секунду. Обычно данный режим включается для того, чтобы визуально представить функционирование системы в реальном темпе наступления событий, проникнуть в суть процессов, происходящих в модели.

Удобные средства разработки анимационного представления модели в AnyLogic позволяют представить функционирование моделируемой системы в живой форме динамической анимации, что позволяет «увидеть» поведение сложной системы.

Средства анимации позволяют пользователю легко создать виртуальный мир (совокупность графических образов, ожившую мнемосхему и т.п.), управляемый динамическими параметрами модели по законам, определенным пользователем с помощью уравнений и логики моделируемых объектов. Визуальное представление поведения системы помогает пользователю проникнуть в суть процессов, происходящих в системе.

ГЛАВА 1. МОДЕЛИ ДИСКРЕТНЫХ СОБЫТИЙ

1.1. AnyLogic-модель сервера

Рассмотрим задачу построения модели сервера, описанную в работе В.Д. Боева [2]. При клиент-серверном взаимодействии между браузерами и удаленным сервером происходит обработка множества запросов со средним значением 1 минута (используется экспоненциальный закон распределения). Пусть сервер обрабатывает один запрос со средним значением 10 секунд с таким же распределением. Максимальное количество запросов, которые сервер может хранить в своем буфере, установим в 50 запросов. Необходимо построить имитационную модель работы сервера для подсчета количества обработанных запросов и количества отказов, а также вероятности обработки запроса. Модель будет представлять собой систему массового обслуживания (СМО). Для построения таких имитационных моделей в AnyLogic используется палитра компонентов.

Модель будет состоять из следующих блоков: **Source, Queue, Delay, Sink** (рис. 1).

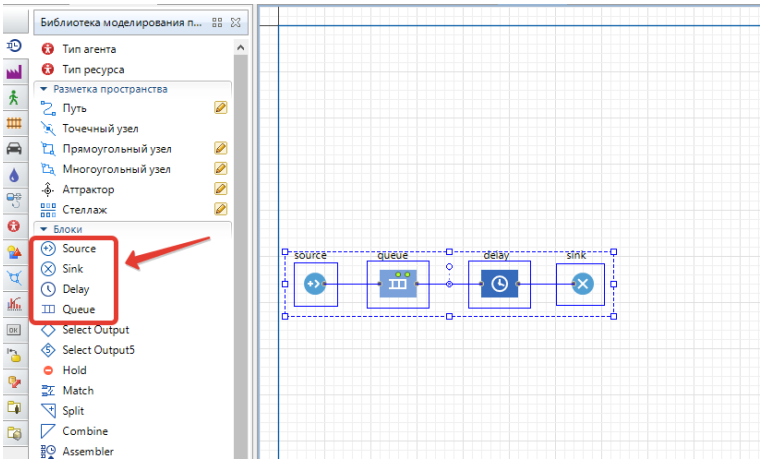


Рис. 1. Установка объектов для модели сервера

Объект **Source** генерирует заявки (**entities**) определенного типа через заданный временной интервал. Заявки представляют собой объекты, которые обрабатываются, обслуживаются или еще каким-нибудь образом подвергаются действию моделируемого процесса: это могут быть клиенты в системе обслуживания, детали в модели производства, документы в модели документооборота и т. д. В нашем примере заявками будут запросы к серверу от рабочих станций, а объект **Source** будет моделировать их поступление серверу.

Чтобы узнать детальное описание объектов библиотеки моделирования процессов, достаточно навести на них мышью в палитре и немного «зависнуть» на нем. Появится всплывающее окно, в котором представлено подробное описание объекта.

Объект **Queue** моделирует очередь из запросов, ожидающих обработки сервером.

Объект **Delay** моделирует задержку, которая вызвана обслуживанием. В нашем примере это будет время обработки сервером одного запроса.

Объект **Sink** обозначает конец блок-схемы и выполняет удаление обработанных запросов из системы.

1. Создание модели

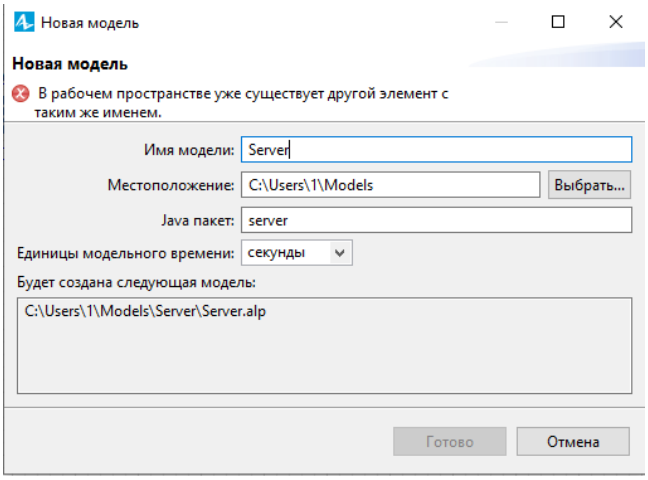


Рис. 2. Создание проекта **Server**

1. Для создания нового проекта в AnyLogic: выполним **Файл – Создать – Модель**. Появится диалоговое окно **Новая модель** (рис. 2). Задайте имя новой модели. В поле **Имя модели** введите **Server**.

2. Выберите каталог, в котором будут сохранены файлы модели. Если хотите сменить предложенный по умолчанию каталог на какой-то другой, то можете ввести путь к нему в поле **Местоположение** или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося нажатием кнопки **Выбрать**. Щёлкните кнопку **Готово**.

3. Далее откроется пользовательский интерфейс (рис. 3). Рассмотрим элементы окна проекта.

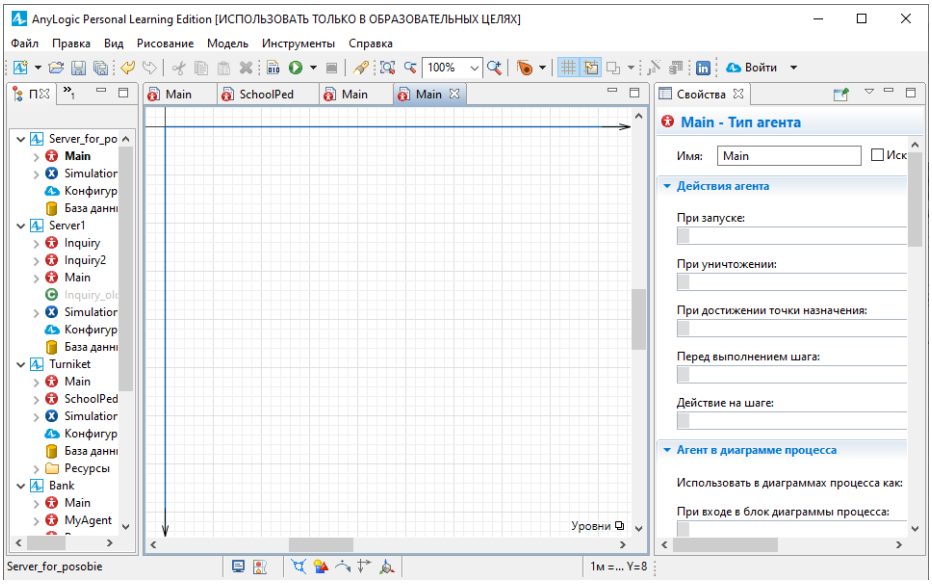


Рис. 3. Элементы окна проекта

В левой части рабочей области находятся панели **Проект** и **Палитра**. Панель **Проект** обеспечивает навигацию по элементам моделей, открытых в текущий момент времени. Модель организована иерархически. Она отображается в виде дерева. Сама модель образует верхний уровень дерева. Эксперименты, классы активных объектов и Java-классы образуют следующий уровень. Элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т. д.

Панель **Палитра** содержит разделённые по категориям элементы, которые могут быть добавлены на графическую диаграмму типа агентов или эксперимента (на рис. 1 раскрыта панель **Палитра**). Для того чтобы открыть нужную палитру, следует подвести курсор к иконке и щелкнуть мышью. Иконка будет иметь рамку выделения. В правой части отображается панель **Свойства**. Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент

элемента (или элементов) модели. В центре рабочей области AnyLogic размещён графический редактор диаграммы агента **Main**.

4. Создайте диаграмму процесса. Для этого в **Палитре** выделите **Библиотеку моделирования процессов** (рис. 1). Из неё перетащите объекты **Source, Queue, Delay, Sink** на диаграмму и соедините, как на рис. 1.

Для добавления объекта на диаграмму, надо щёлкнуть его мышью и, не отпуская её, перетащить в графический редактор.

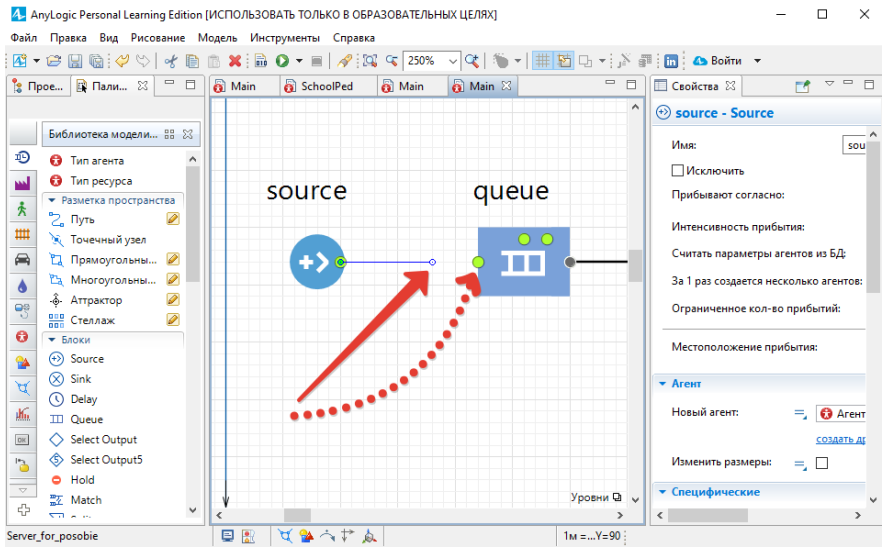


Рис. 4. Соединение портов двух объектов

1. При перетаскивании объектов вы можете видеть, как появляются соединительные линии между объектами. Эти соединительные линии должны соединять только **порты**, находящиеся слева и справа от объекта. Если у вас не получится автоматическое соединение нужных объектов, то можно дважды щёлкнуть на начальный порт. Он станет зелёным. Затем нужно протащить курсор к конечному порту. Он также станет зелёным. Для завершения процесса соединения дважды щёлкните конечный порт (рис. 4).

2. Введем параметр **time_mean** – параметр со значением 10 (среднее время обработки запроса сервером). Для этого на **Палитре** перейдите на вкладку **Агент** и перетащите объект **Параметр** в графический редактор (рис. 5). В свойствах параметра **time_mean** установите тип – **int**, значение поля **Значение по умолчанию** равным 10.

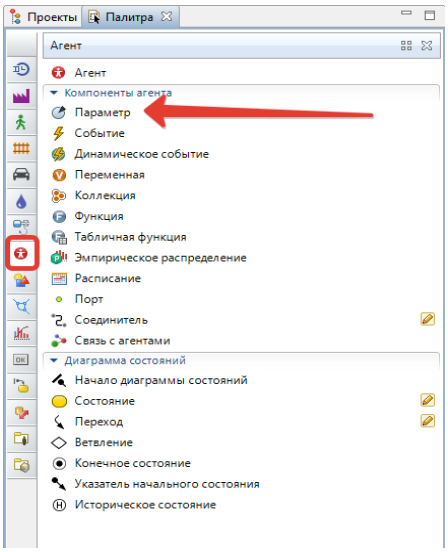


Рис. 5. Создание параметра **time_mean**

3. Аналогично предыдущему пункту создайте параметр **emkBuf** (емкость буфера сервера) и задайте ему тип – **int**, **Значение по умолчанию** равно 50.

4. Установите для объектов следующие свойства (выделяя нужный объект, справа меняется окно **Свойства**):

- **Source** – время между прибытиями заявок на обслуживание, задается функцией **exponential(time_mean)**. Заявки прибывают согласно **Времени между прибытиями.**
- **Queue** – вместимость очереди, в поле **Вместимость** внести **emkBuf**;
- **Delay** – время обработки сервером сигнала (поле **Время задержки**), значение вычисляется функцией **exponential (time_mean);**
- **Sink** – уничтожение заявки (запрос считается обработанным), нет параметров.

Запуск модели

1. Уже с данными параметрами модель можно запустить и изучить ее поведение. Вы можете сконфигурировать выполнение модели в соответствии с вашими требованиями. Модель выполняется с тем набором установок, которые задаются специальным элементом модели – «**Эксперимент»**. Можно создать несколько экспериментов с различными параметрами и изменять конфигурацию модели, просто запуская тот или иной эксперимент модели.

2. В панели **Проект Simulation: Main** эксперименты отображаются в дереве модели. Один эксперимент, названный **Simulation**, создается по умолчанию (см.

справа). Это простой эксперимент, позволяющий запускать модель с заданными значениями параметров, поддерживающий режимы виртуального и реального времени, анимацию и отладку модели. Если нужно наблюдать поведение модели в течение длительного периода (до того момента, пока вы сами не остановите выполнение модели), то по умолчанию времени остановки нет. Обработку запросов сервером планируется исследовать в течение одного часа, т. е. 3 600 с.

3. В панели «**Проект»** выделите эксперимент **Simulation:Main**.
4. Щелчком раскройте вкладку **Модельное время**.
5. Установите **Режим выполнения** равным **Виртуальное время (максимальная скорость)** (рис. 6).

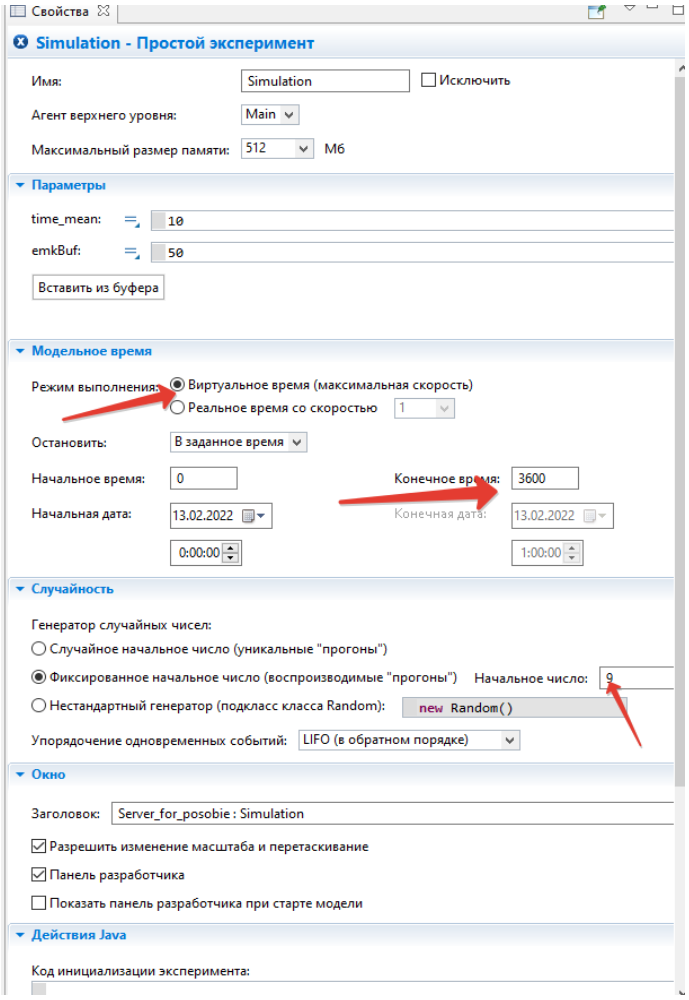


Рис. 6. Настройка параметров симуляции модели

6. В поле **Остановить** выберите из списка **В заданное время**.
7. В поле **Конечное время** установите 3600.

8. Раскройте вкладку **Случайность**. Выберите опцию **Фиксированное начальное число (воспроизводимые прогоны)**. В поле **Начальное число** установите **9**.

9. В панели **Проект** выделите **Server**. Из выпадающего списка **Единицы модельного времени** выберите **секунды** (рис. 7).

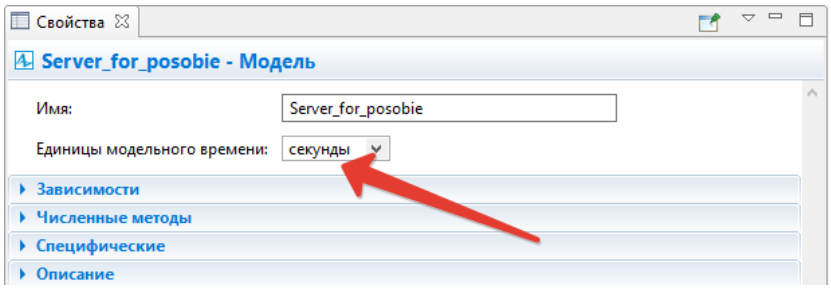


Рис. 7. Настройка единиц модельного времени

10. Запуск модели выполняется нажатием кнопки  на панели инструментов. После запуска появится окно симуляции модели и можно посмотреть, как она работает (рис. 8).

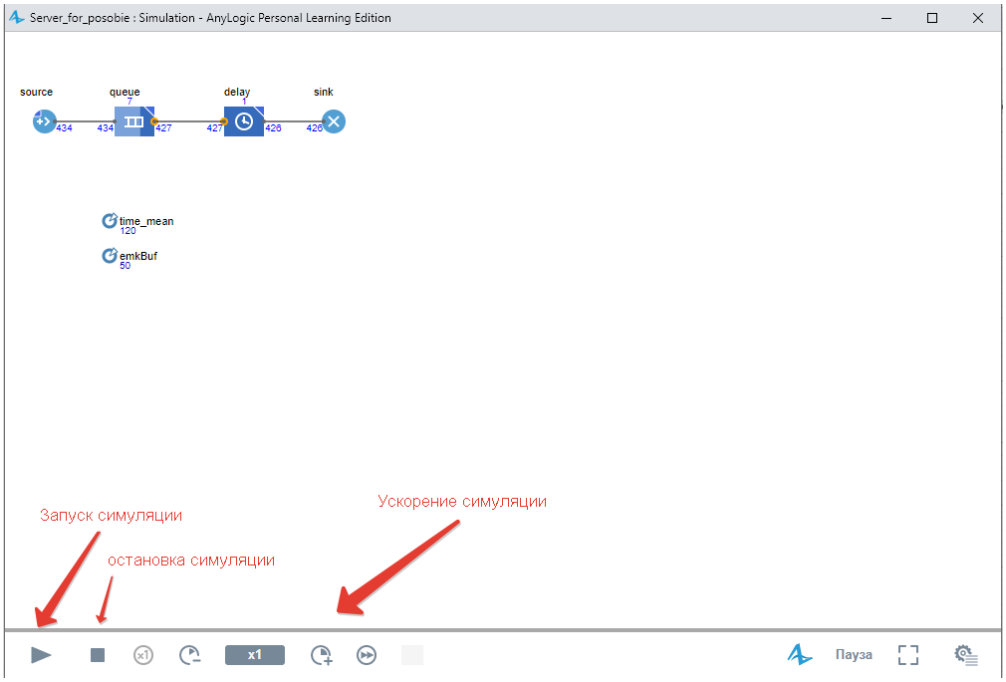


Рис. 8. Запуск модели на симуляцию

11. В дальнейшем нажатием кнопки **Запустить** будет запускаться тот эксперимент, который запускался в последний раз. Чтобы выбрать другой эксперимент, вам будет нужно щелкнуть мышью по стрелке, находящейся в правой части кнопки **Запустить**, и выбрать нужный вам эксперимент из открывшегося списка

(или щелкнуть правой кнопкой мыши по этому эксперименту в панели **Проект** и выбрать **Запустить** из контекстного меню).

12. После запуска модели вы увидите окно презентации этой модели. В нем будет отображена презентация запущенного эксперимента. AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию, нарисованную вами для главного класса активного объекта этого эксперимента (**Main**). Нажмите на кнопку запуска модели (рис. 8).

Рис. 8. Панель управления запуском модели

13. После запуска модели вы увидите презентацию корневого класса активного объекта запущенного эксперимента. Для каждой модели, созданной в **Библиотеке моделирования процессов**, автоматически создается блок-схема с наглядной визуализацией процесса, с помощью которой можно изучать текущее состояние модели, например, длину очереди, количество обработанных запросов и так далее (рис. 8). На рисунке показано, сколько на каждом узле входящих заявок (запросов к серверу), сколько обработано, сколько стоит в очереди, сколько запросов было полностью отработано.

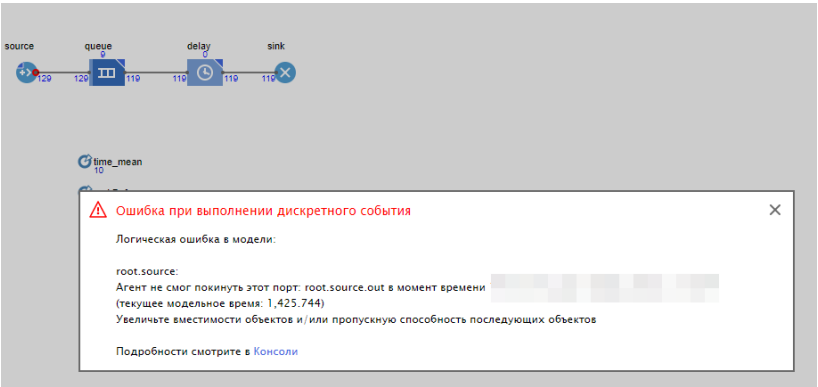


Рис. 9. Ошибка при выполнении дискретного события

14. Если емкость буфера будет недостаточной (для данных параметров установить значение **emkBuf** равным 10), то может возникнуть ошибка, показанная на рис. 9.

Для устранения ошибки измените свойства параметра **emkBuf**, т. е. увеличьте максимальную вместимость очереди. Можно задать значение 50–60 и т. д. Ошибка, возможно, снова появится, так как зависит от длительности времени моделирования. Но это не критично.

Снова запустите модель.

15. Вы можете следить за состоянием любого объекта диаграммы процесса во время выполнения модели с помощью окна проверки этого объекта. Чтобы открыть окно проверки, щёлкните мышью по значку нужного блока. Окно проверки, подведя курсор, можно перемещать в нужное вам место. Также, подведя курсор к правому нижнему углу окна проверки, можно при необходимости изменять его размеры. В окне проверки будет отображена базовая информация по выделенному объекту: например, для объекта **queue** будут отображены вместимость очереди, количество заявок, прошедшее через каждый порт объекта и т. д. Такая же информация содержится в окне проверки и для объекта **delay** (рис. 10).

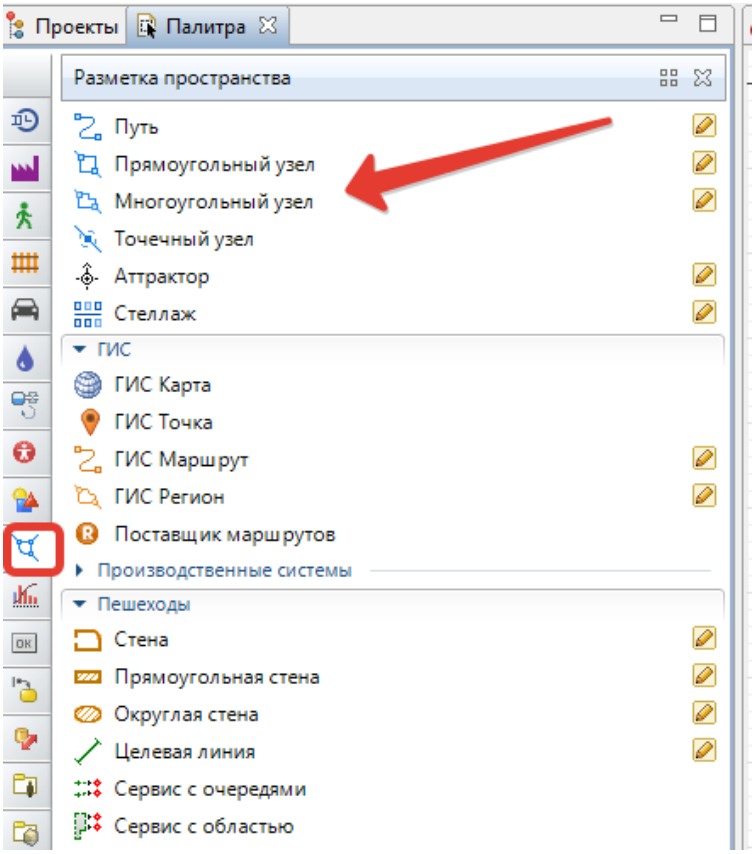


Рис. 11. Палитра **Разметка пространства**

16. Если нужно остановить выполнение модели, то можно щёлкнуть мышью кнопку **Прекратить выполнение эксперимента**  панели управления окна презентации.

17. Для предотвращения остановок модели по ранее указанной ошибке — недостаточной ёмкости объекта **Queue** — мы увеличили ёмкость объекта **Queue**. Однако можно было бы изменять среднее время имитации поступления запросов объектом S**ource** и среднее время обработки запросов сервером, т.е. среднее время задержки объекта **Delay**, оставляя неизменной длину очереди и добиваясь безошибочной работы модели. Конечно, при изменении свойств объектов модели нужно обязательно исходить из целей её построения.

Анимация результатов моделирования

Можно ограничиться анализом и интерпретацией уже запущенной модели, но в ряде случаев этого недостаточно. AnyLogic позволяет построить более наглядную визуализацию с помощью анимации. Для данной задачи визуализируем процесс поступления запросов на сервер и обработку запросов сервером.

В данном случае нас не интересует конкретное расположение объектов в пространстве, поэтому мы можем просто добавить схематическую анимацию интересующих нас объектов — **сервер** и **очередь запросов к нему**. Анимация модели рисуется в той же диаграмме (в графическом редакторе), в которой задается и диаграмма моделируемого процесса. Для анимации будем использовать объекты палитры **Презентация**.

1. Нарисуйте прямоугольный узел, который будет обозначать на анимации **сервер**. Для этого откройте палитру **Разметка пространства** (рис. 11).

2. Палитра **Презентация** содержит в качестве элементов различные фигуры, используемые для рисования презентаций моделей. Это **Путь, Прямоугольный узел, Многоугольный узел, Точечный узел, Аттрактор** и др.

3. Выделите элемент **Прямоугольный узел** и перетащите его на диаграмму класса активного объекта. Поместите элемент **Прямоугольный узел** так, как показано на рис. 12.

4. Настроим работу **Прямоугольного узла** так, чтобы его цвет менялся в зависимости от того, обрабатывает ли сервер в данный момент времени запрос или нет.

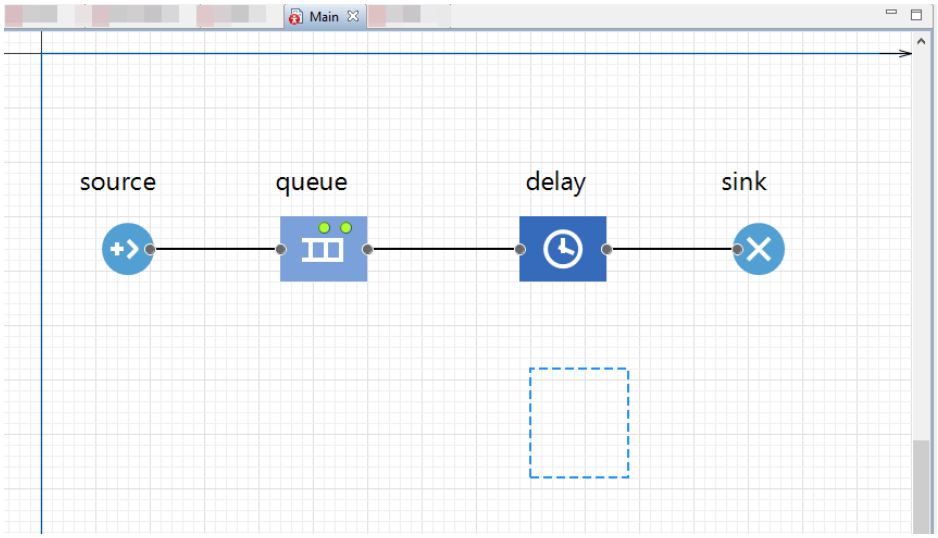


Рис. 12. Элемент **«Прямоугольный узел»** на диаграмме

Для этого выделите нарисованный **Прямоугольный узел** на диаграмме. Перейдите на страницу **Внешний вид** панели свойств (рис. 13).

Если нужно, чтобы по ходу моделирования то или иное свойство фигуры меняло своё значение в зависимости от каких-то условий, то можно ввести в поле соответствующего динамического свойства выражение, которое будет постоянно вычисляться заново при выполнении модели. Возвращаемый результат вычисления будет присваиваться текущему значению этого свойства.

Пусть во время моделирования цвет нашей фигуры будет меняться, поэтому щёлкните в поле **Цвет заливки**: по стрелке, выберите **Динамическое значение** и введите там следующую строку:

delay.size()>0?red:green

Здесь **Delay** – это имя нашего объекта **Delay**. Функция **size()** возвращает число запросов, обслуживаемых в данный момент времени. Если сервер занят, то цвет кружка будет **красным**, в противном случае — **зелёным**.

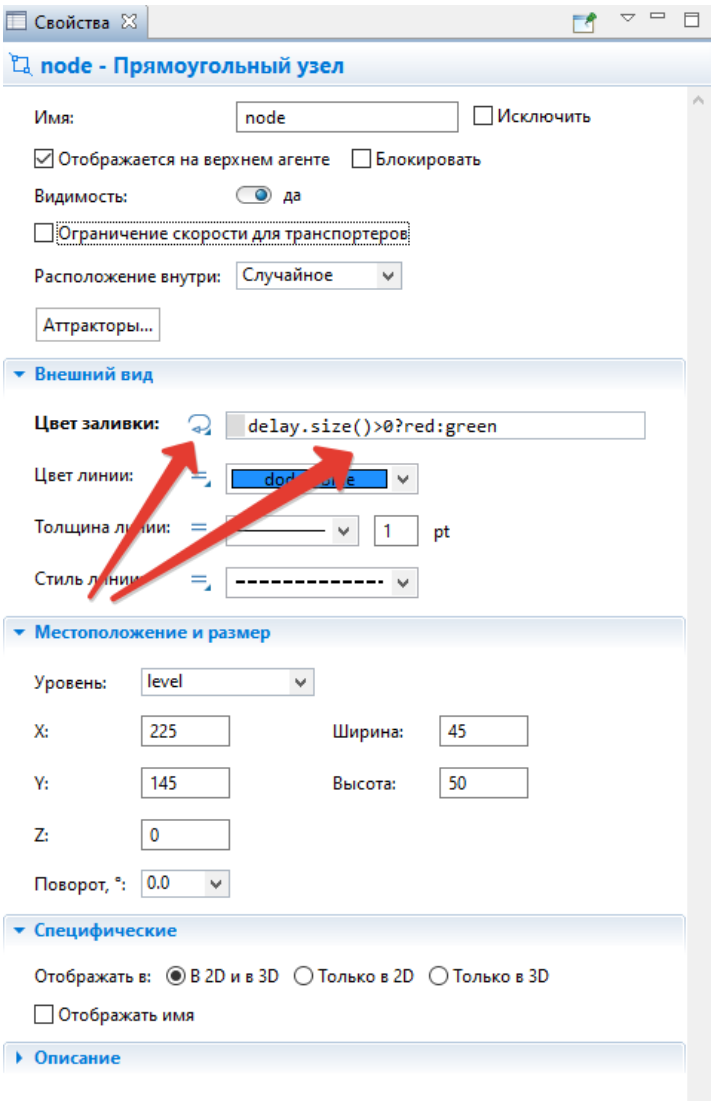


Рис. 13. Динамическое свойство для установки цвета заливки

5. Нарисуйте путь, который будет обозначать на анимации очередь к серверу (рис. 14). Чтобы нарисовать путь, сделайте двойной щелчок мышью по элементу **Путь** палитры **Разметка пространства**, чтобы перейти в режим рисования. Теперь вы можете рисовать путь точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда вы хотите поместить вершины пути. Чтобы завершить рисование, добавьте последнюю точку пути двойным щелчком мыши.

Очень важно, какую точку пути вы создаете первой. Заявки будут располагаться вдоль нарисованного вами пути в направлении от конечной точки к начальной точке. Поэтому обязательно начните рисование пути слева и поместите рядом с сервером конечную точку пути, которая будет соответствовать в этом случае началу очереди.

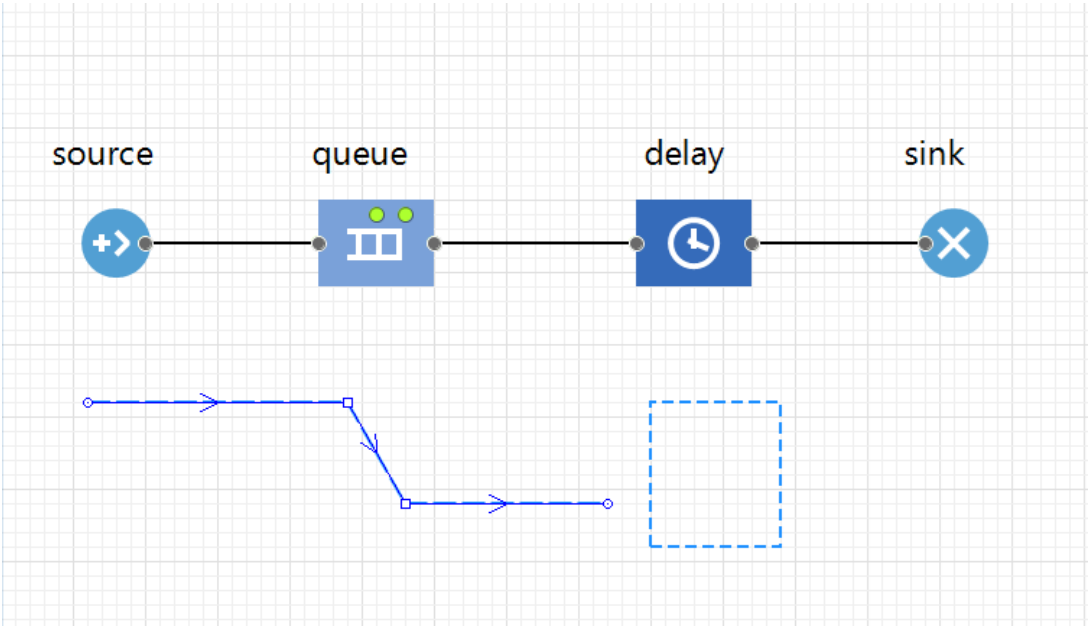


Рис. 14. Путь на диаграмме процесса

6. Зададим созданные анимационные объекты в качестве анимационных фигур объектов диаграммы процесса.

Задайте путь в качестве фигуры анимации очереди. Выделите объект **queue**. На странице свойств объекта **queue** в поле **Место агентов** выберите из выпадающего списка **path** (рис. 15).

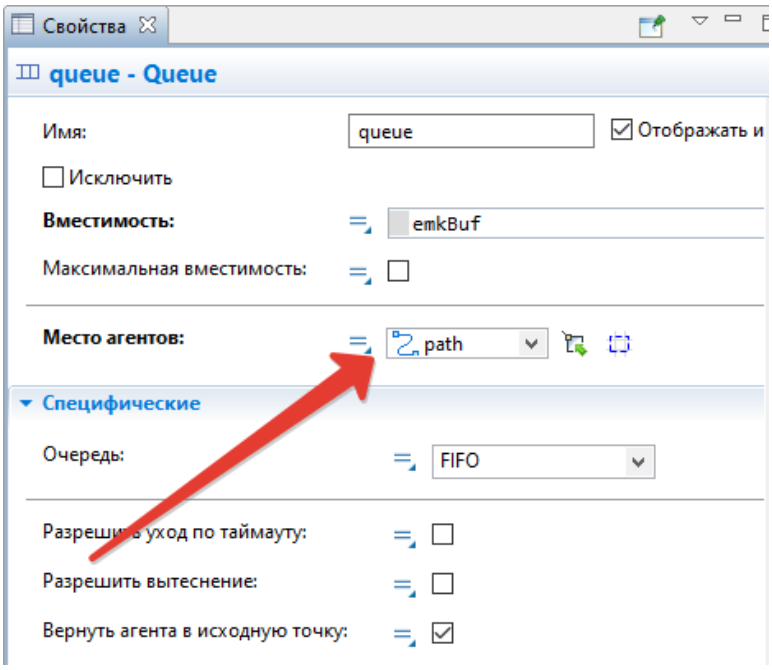


Рис. 15. Задание пути в качестве фигуры анимации очереди

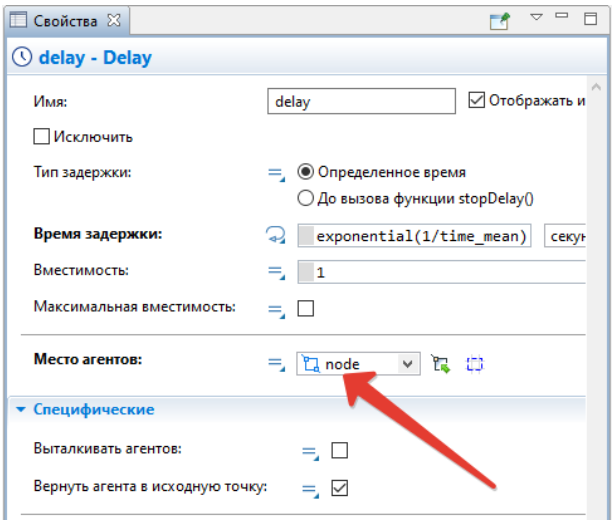


Рис. 16. Задание прямоугольного узла в качестве фигуры анимации сервера

Задайте прямоугольный узел в качестве фигуры анимации сервера. Выделите объект **Delay**. Введите в поле **Место агентов** из выпадающего списка имя прямоугольного узла **node** (рис. 16).

7. Запустим модель. Можно увидеть, что у модели теперь есть простейшая анимация – сервер и очередь запросов к нему (рис. 17). Цвет фигуры сервера будет меняться в зависимости от того, обрабатывается ли запрос в данный момент времени или нет.

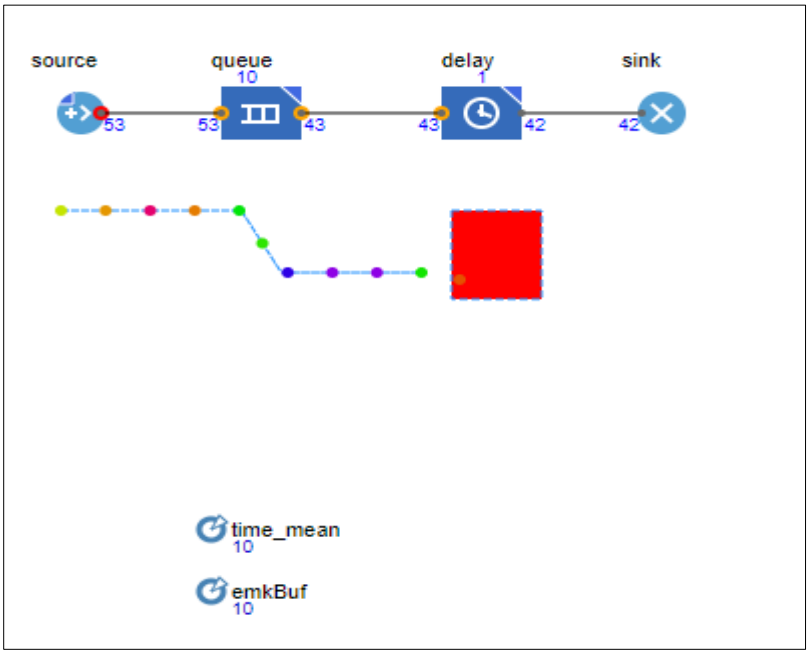


Рис. 17. Анимация модели

Если в процессе запуска модели у вас все же появляется логическая ошибка, то можно изменить настройки симуляции. Выберите в дереве проекта узел **Simulation: Main**. В окне **Свойств** на вкладке **Модельное время** установите **Режим выполнения** в значение **Реальное время со скоростью** 10. Запустите модель.

Сбор статистики использования ресурсов

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Объекты **Enterprise Library** самостоятельно производят сбор основной статистики. Все, что нужно сделать — это включить сбор статистики для объекта. Мы уже выставили данный параметр для объектов **Delay** и **Queue**, поэтому теперь можно, например, просмотреть интересующую нас статистику (статистику занятости сервера и длины очереди) с помощью диаграмм.

Добавьте диаграмму для отображения среднего коэффициента использования сервера:

1. Откройте палитру **Статистика**. Эта палитра содержит элементы сбора данных и статистики, а также диаграммы для визуализации данных и результатов моделирования.

2. Перетащите элемент **Столбиковая диаграмма** из палитры **Статистика** на диаграмму класса и измените ее размер (рис. 18).

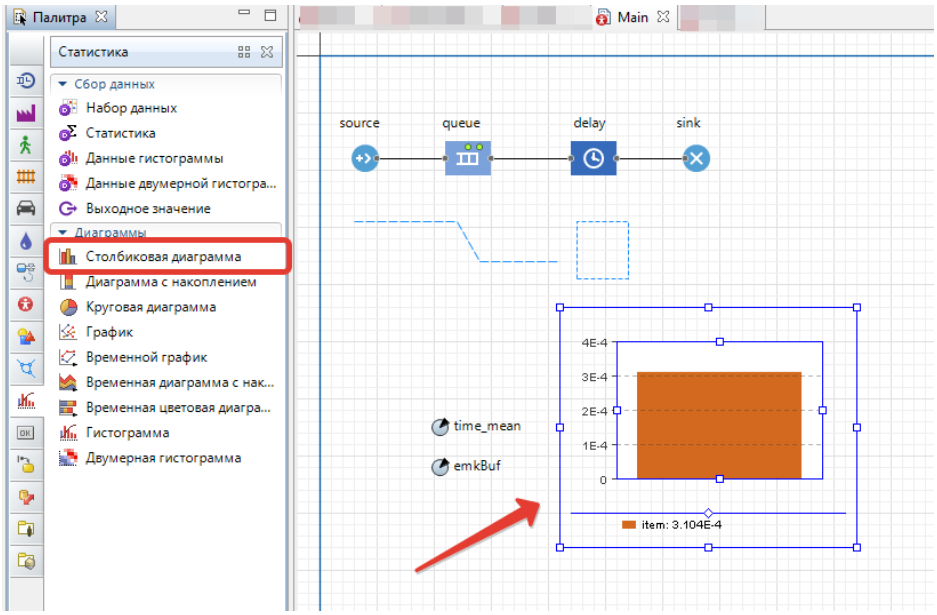


Рис. 18. Элемент **Столбиковая диаграмма** на диаграмме класса

3. Перейдите на панель **Свойства**. Щёлкните кнопку **Добавить элемент данных**. После щелчка появится секция свойств того элемента данных (**chart** – **Столбиковая диаграмма**), который будет отображаться на этой диаграмме (рис. 19).

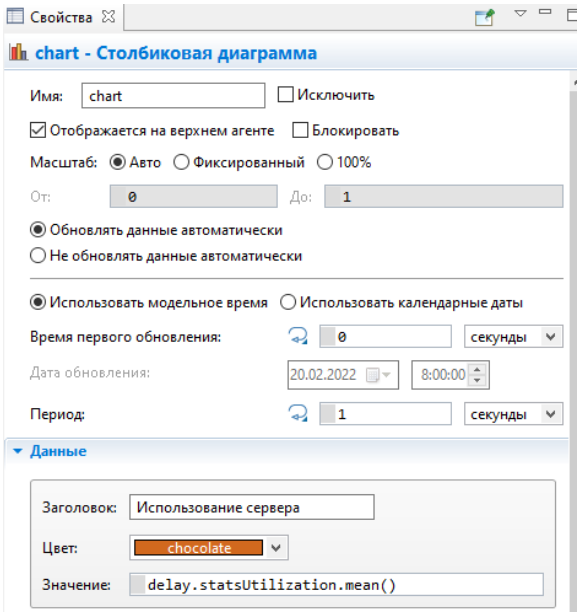


Рис. 19. Страница **Свойства**

4. Измените Заголовок на **Использование сервера**.

5. Введите **delay.statsUtilization.mean()** в поле **Значение**. Здесь **delay** — это имя нашего объекта **delay**. У каждого объекта **delay** есть встроенный набор данных **statsUtilization**, занимающийся сбором статистики использования этого объекта. Функция **mean()** возвращает среднее из всех измеренных этим набором данных значений. Вы можете использовать и другие методы сбора статистики, такие как **min()** или **max()**.

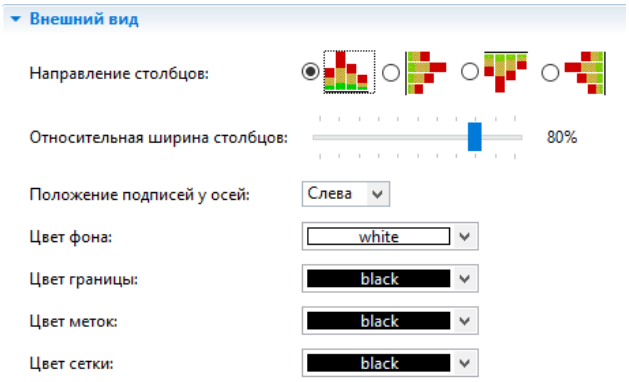


Рис. 20. Вкладка **Внешний вид**

6. Щёлкните **Внешний вид** (рис. 20). Установите свойства: **направление столбцов, цвета фона, границ, меток, сетки, положение подписей у столбцов**.

7. Раскройте щелчками страницы (вкладки) **Местоположение и размер, Легенда, Область диаграммы** (рис. 21). Установите свойства, чтобы изменить расположение легенды относительно диаграммы (она будет внизу), размер диаграммы, высоту, ширину, координаты размещения на диаграмме, цвета текста, границы.

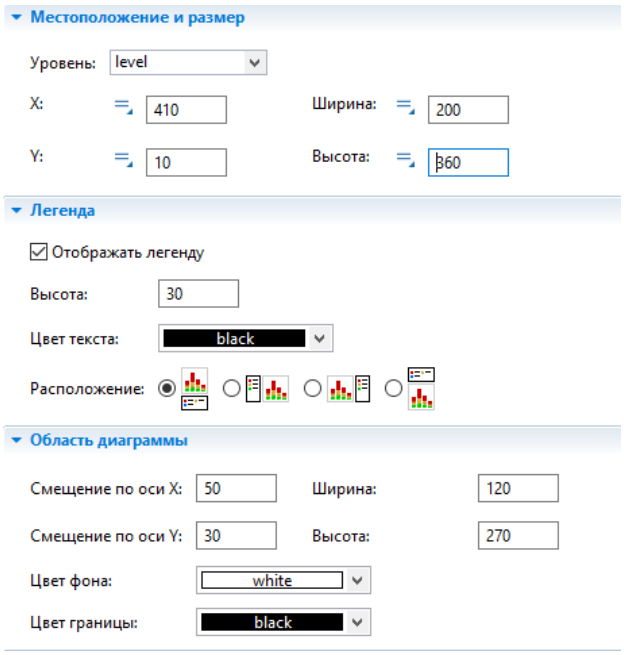


Рис. 21. Вкладки **Местоположение и размер, Легенда, Область диаграммы**

8. Аналогичным образом добавьте еще одну столбиковую диаграмму для отображения средней длины очереди. На панели **Свойства** щёлкните **Добавить элемент данных**. После щелчка появится страница **Данные свойств элемента данных** (**chart1** – **Столбиковая диаграмма**), который также будет отображаться на этой диаграмме.

Заголовок сделайте **Длина очереди**, значение **queue.statsSize.mean()**.

На страницах **Внешний вид**, **Местоположение и размер**, **Легенда**, **Область диаграммы** установите свойства самостоятельно. Столбцы диаграммы должны размещаться горизонтально (рис. 22).

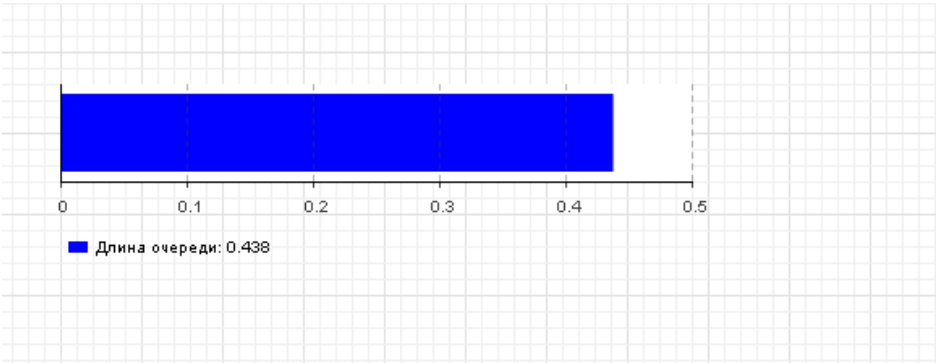


Рис. 22. Столбиковая диаграмма для отображения длины очереди

9. Запустите модель с двумя столбиковыми диаграммами, установив модельное время 3600 единиц, и понаблюдайте за её работой (рис. 23).

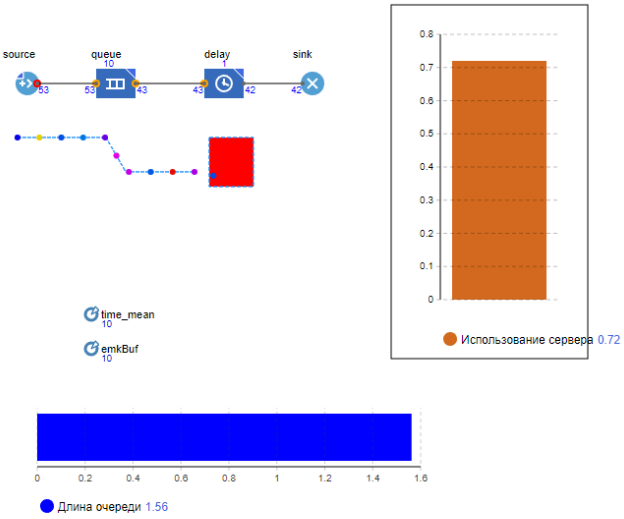


Рис. 23. Наблюдение за моделью с двумя столбиковыми диаграммами

Уточнение модели согласно ёмкости входного буфера

Объект **Queue** моделирует очередь заявок, ожидающих приёма объектами, следующими за ним в потоковой диаграмме, или же моделирует хранилище заявок общего назначения. При необходимости можно задать максимальное время ожидания заявки в очереди. Можно с помощью написанной вами программы извлекать заявки из любых позиций в очереди. Заявка может покинуть объект **Queue** различными способами: обычным способом через порт **out**, когда объект, следующий в блок-схеме за этим объектом, готов принять заявку; через порт

outTimeout, если заявка проведет в очереди заданное количество времени (если включен режим **таймаута**); через порт **outPreempted**, будучи вытесненной другой поступившей заявкой при заполненной очереди (если включен режим вытеснения); «вручную», путем вызова функции **remove()** или **removeFirst()**. В первом случае объект **Queue** покидает заявка, находящаяся в самом начале очереди (в нулевой позиции).

Если заявка направлена в порт **outTimeout** или **outPreempted**, то она должна покинуть объект мгновенно. Если включена опция вытеснения, то объект **Queue** всегда готов принять новую заявку, в противном случае при заполненной очереди заявка принята не будет. Поступающие заявки помещаются в очередь в определенном порядке: либо согласно правилу FIFO (в порядке поступления в очередь), либо согласно приоритетам заявок. Приоритет может либо явно храниться в заявке, либо вычисляться согласно свойствам заявки и каким-то внешним условиям. Очередь с приоритетами всегда примет новую входящую заявку, вычислит её приоритет и поместит в очередь в позицию, соответствующую её приоритету. Если очередь будет заполнена, то приход новой заявки вынудит последнюю хранящуюся в очереди заявку покинуть объект через порт **outPreempted**. Но если приоритет новой заявки не будет превышать приоритет последней заявки, то тогда вместо неё будет вытеснена именно эта новая заявка. Для выполнения условия постановки задачи воспользуемся последним способом вытеснения. Все запросы, вырабатываемые объектом **«Source»**, имеют один и тот же приоритет. Поэтому при полном заполнении накопителя (5 запросов) теряться будет последний запрос.

Изменим параметры очереди:

1. Выделите объект **emkBuf**. Измените значение с 10 на 5 запросов.
2. У объекта **«Queue»** установите **Разрешить вытеснение**.
3. Для уничтожения потерянных запросов вследствие полного заполнения накопителя нужно добавить второй объект **Sink**. Откройте в Палитре **Библиотеку моделирования процессов** и перетащите блок **Sink** на диаграмму (рис. 24).

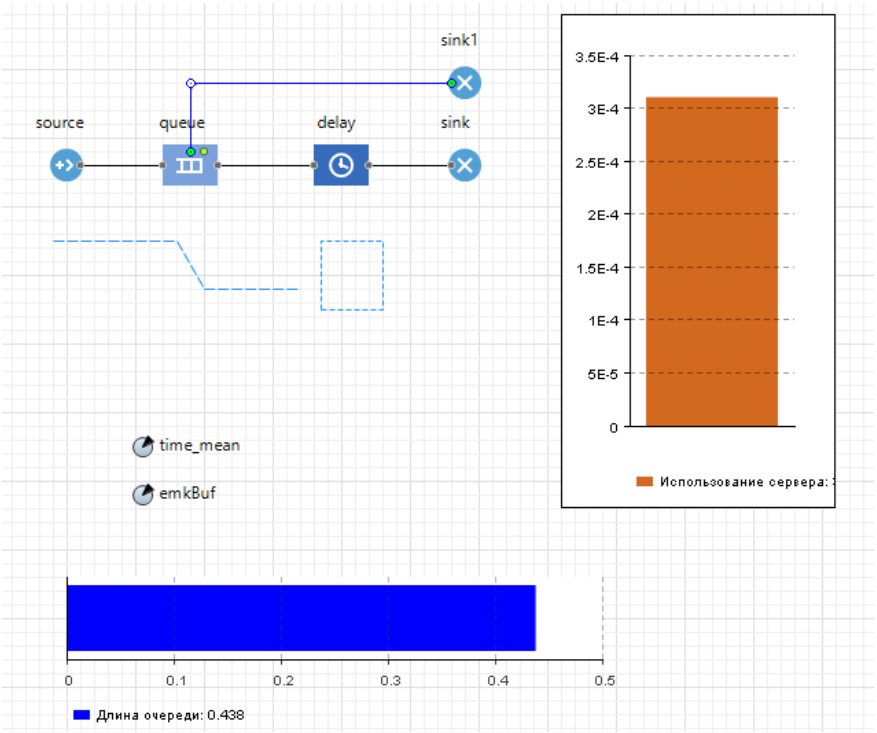


Рис. 24. Измененная модель

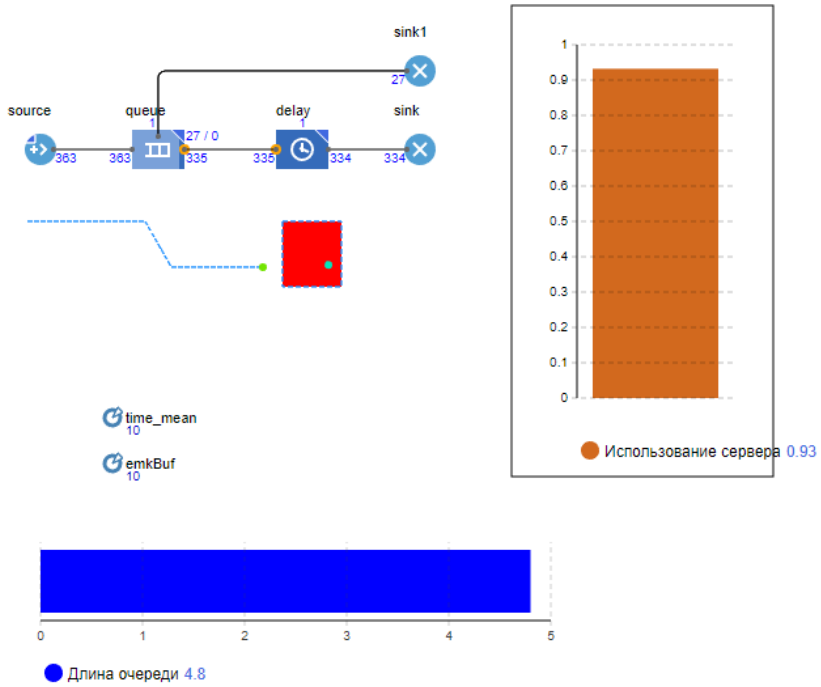


Рис. 25. Измененная модель в запущенном состоянии

4. Запустите уточненную модель и понаблюдайте за ее работой. Сравните рис. 25 с рис. 23. На рис. 25 видно, что запросы при длине очереди в 10 запросов теряются, и ошибки при этом не возникает. Модель по ограничению ёмкости

входного буфера и значениям других параметров соответствует постановке задачи. Однако согласно постановке задачи требуется определить математическое ожидание времени обработки одного запроса и математическое ожидание вероятности обработки запросов.

Сбор статистики по показателям обработки запросов

Entity (заявка) является базовым классом для всех заявок, которые создаются и работают с ресурсами в процессе, описанном вами с помощью диаграммы из объектов **Библиотеки моделирования процессов**. **Entity** по существу является обычным **Java-классом** с теми функциональными возможностями, которые необходимы и достаточны для обработки и отображения анимации заявки объектами **Библиотеки моделирования процессов**. Эти функциональные возможности можно расширить добавлением дополнительных полей и методов и работой с ними из объектов диаграммы, описывающей моделируемый процесс. Согласно постановке задачи нужно определять математическое ожидание времени и вероятности обработки запросов сервером.

Математическое ожидание или среднее время обработки одного запроса определяется как отношение суммарного времени обработки *n* запросов к их количеству, т. е. к *n*. Для определения суммарного времени нужно знать время обработки *i*-го запроса. Для этого введем дополнительные поля:

time_vhod — время входа запроса в буфер сервера;

time_vihod — время выхода запроса с сервера (входа в блок **sink**).

Тогда **time_obrabotki=time_vihod-time_vhod.** Вероятность обработки запросов сервером определяется как отношение количества обработанных запросов к количеству всех поступивших запросов. Значит, нужно вести счет запросов на выходе источника запросов и на выходе с сервера (входе в блок **sink**). Для этого также введем дополнительные поля:

col_vhod — количество поступивших всего запросов;

col_vihod — количество обработанных сервером запросов.

Тогда **ver_obrabotki=col_vihod/col_vhod.**

Замечание. Ничего необычного во введённых дополнительных полях нет. Это параметры реальных элементов потоков, в данном случае запросов. AnyLogic предоставляет возможность создавать запросы с теми параметрами, которые необходимы в модели.

Для включения в запросы дополнительных полей необходимо создать нестандартный тип заявки.

1. Откройте палитру **Библиотека моделирования процессов**. Перетащите элемент **Тип агента** в графический редактор. Появится тип заявки **Entity** (рис. 26).

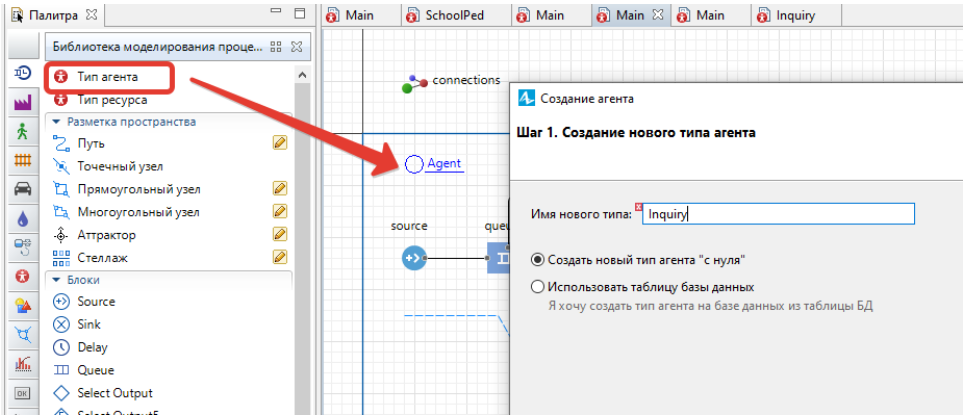


Рис. 26. Создание нового агента

Появится диалоговое окно **Создание агента** (рис. 26).

Шаг 1. Создание нового типа агента. В поле **Имя нового типа** введите **Inquiry**.

Шаг 2. Выберите анимацию агента: установите 2D и выберите из выпадающего списка, например, **Сообщение**. Щёлкните **Далее** (рис. 27).

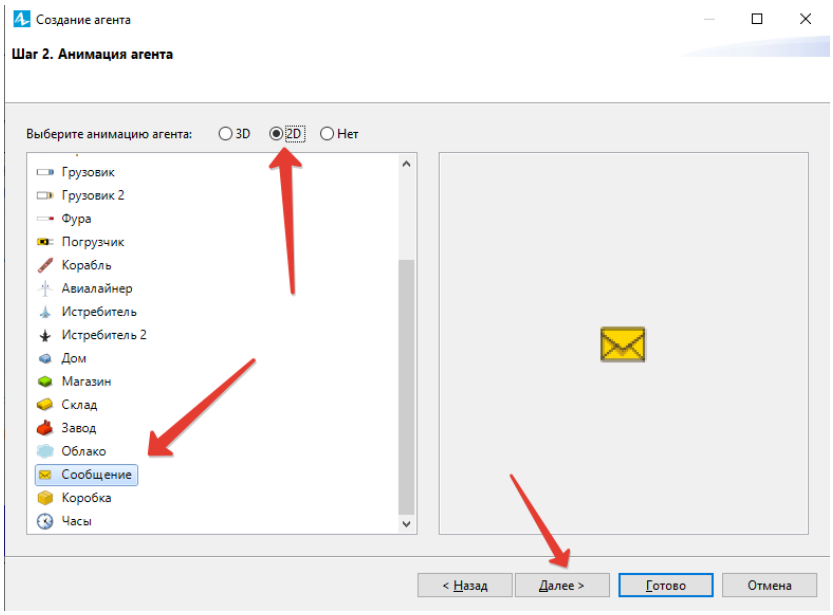


Рис. 27. Настройка анимации нового агента

Шаг 3. Параметры агента. Щёлкните **<добавить… >** (рис. 28). В поле **Параметр** введите

- **time_vhod**, из выпадающего списка **Тип** выберите **double;**
- **time_vihod**, из выпадающего списка **Тип** выберите **double;**
- **col_vhod**, из выпадающего списка **Тип** выберите **double**;

- **col_vihod**, из выпадающего списка **Тип** выберите **double**;

Так как в поле **Значение по умолчанию** мы не устанавливали никаких значений, то всем параметрам будет установлен 0.

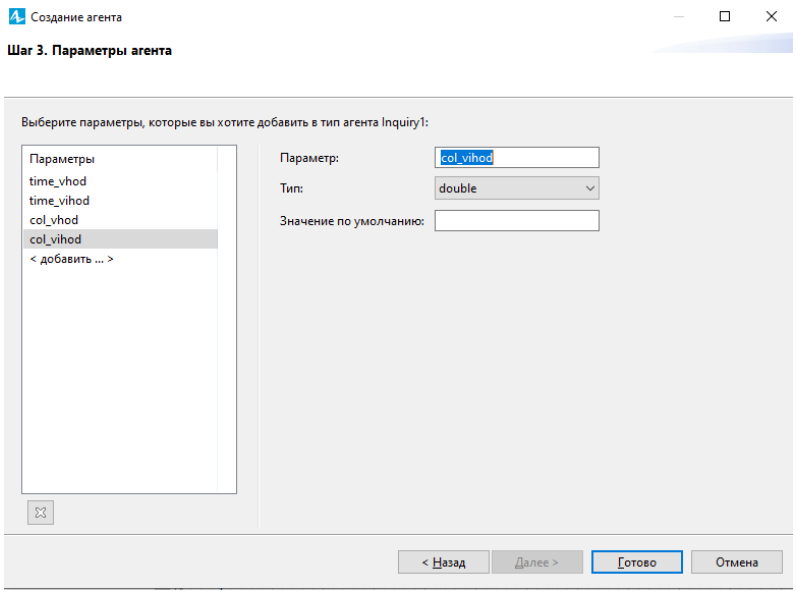


Рис. 28. Добавление параметров агента

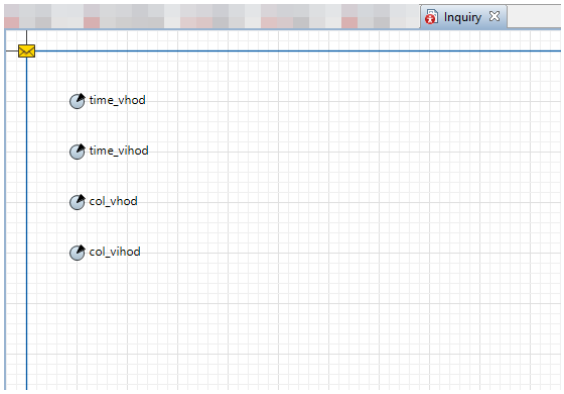


Рис. 29. Окно с параметрами нестандартного типа заявки **Inquiry**

2. Щёлкните кнопку **Готово**. Вы увидите окно, в котором будут показаны автоматически созданные параметры нестандартного типа заявок **Inquiry** (рис. 29). Закройте окно, щелкнув крестик в закладке рядом с его названием.

3. Для сбора статистических данных о времени обработки запросов сервером необходимо добавить элемент статистики. Этот элемент будет запоминать значения времен для каждого запроса. На основе этого он предоставит пользователю стандартную статистическую информацию (среднее, минимальное, максимальное из измеренных значений, среднеквадратичное отклонение и т. д.).

Чтобы добавить элемент сбора данных гистограммы на диаграмму, перетащите элемент **Данные гистограммы** с палитры **Статистика** на **диаграмму активного класса**.

Задайте свойства элемента (рис. 30): измените **Имя** на **time_obrabotki**; сделайте **Кол-во интервалов** равным 50; задайте **Нач. размер интервала** 0,01.

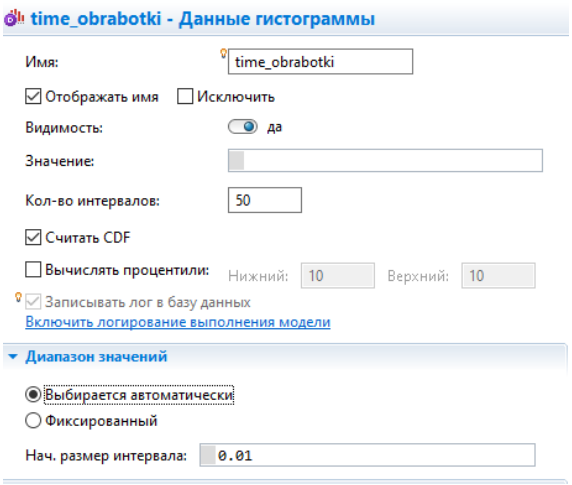


Рис. 30. Элемент сбора статистики о времени обработки запросов

4. Добавьте еще элемент сбора статистики для определения вероятности обработки запросов.

Перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса.

Задайте свойства элемента (рис. 31): измените **Имя** на **ver_obrabotki**; сделайте Кол-во интервалов, равным 50; задайте **Нач. размер интервала** 0,01. Диаграмма после добавления элементов сбора статистики представлена на рис. 31.

5. Чтобы создавать заявки нестандартного типа, как в нашем случае **Inquiry**, нужно поместить вызов конструктора этого типа в поле **Новая заявка** объекта **Source**. Но, несмотря на то, что заявки в потоке теперь и будут типа **Inquiry**, остальные объекты диаграммы будут продолжать их считать заявками типа **Агент**. Поэтому они не позволят явно обращаться к дополнительным полям класса **Inquiry**. Чтобы разрешить доступ к полям вашего нестандартного типа заявки в коде динамических параметров объектов потоковой диаграммы, нужно указать имя нестандартного типа заявки в качестве Типа агента этого объекта. В нашей потоковой диаграмме с учётом блока **Source** всего пять объектов.

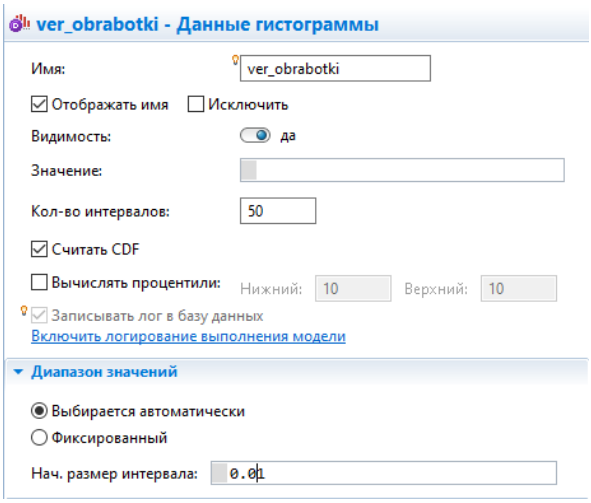


Рис. 31. Элемент сбора статистики о вероятности обработки запросов

Для объектов **source**, **queue**, **delay**, **sink**, **sink1** установите в качестве поля **Новый агент** значение **Inquiry** (рис. 32).

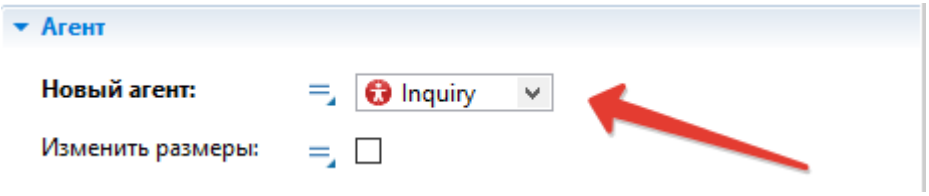


Рис. 32. Изменение поля **Новый агент**

6. Для объекта «**source**» в окне **Свойства** задайте в поле **Действие при выходе** равным **agent.time_vhod=time()**;

Код будет сохранять время создания заявки-запроса в параметре **time_vhod** нашего типа заявки **Inquiry**. Функция **time()** возвращает текущее значение модельного времени.

7. Для объекта **Sink** на вкладке **Действие** в поле **При входе** введите **time_obrabotki.add(time()-agent.time_vhod)**;

Этот код добавляет время обработки одного запроса в объект сбора данных гистограммы **time_obrabotki**. Данное время определяется как разность между текущим модельным временем **time()** и временем входа запроса в модель. **add** — встроенная функция добавления элемента в массив:

agent.col_vihod=sink.count();

agent.col_vhod=source.count().

Эти коды заносят количество запросов, вошедших в блок **Sink** и вышедших из блока **«Source»** соответственно. **count()** — встроенная функция этих блоков возвращает количество вошедших в блок **Sink** и количество вышедших из блока **Source** заявок.

ver_obrabotki.add(entity.col_vihod/entity.col_vhod).

Этот код добавляет относительную долю обработанных запросов в объект сбора данных гистограммы **ver_obrabotki** при поступлении каждого обработанного запроса в блок **Sink**. На основе множества таких относительных долей определяется математическое ожидание вероятности обработки запросов сервером.

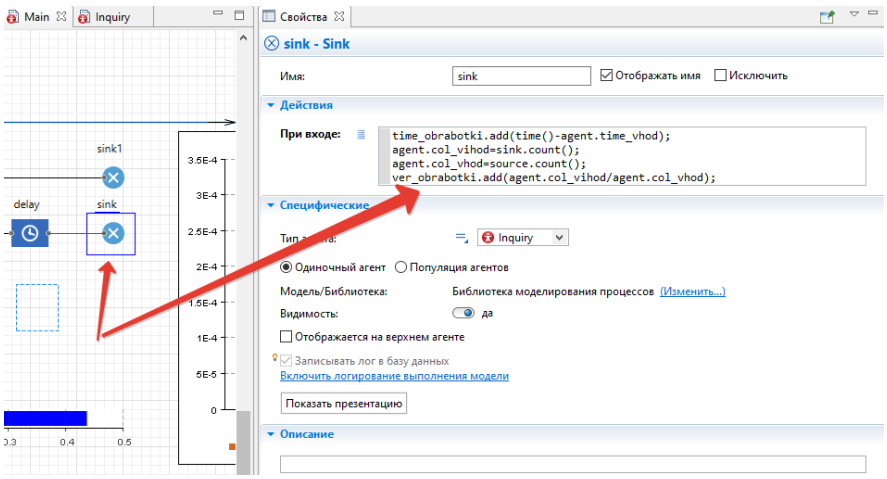


Рис. 33. Обработка действия **При входе** объекта **sink**

8. Запустите модель (рис. 34).

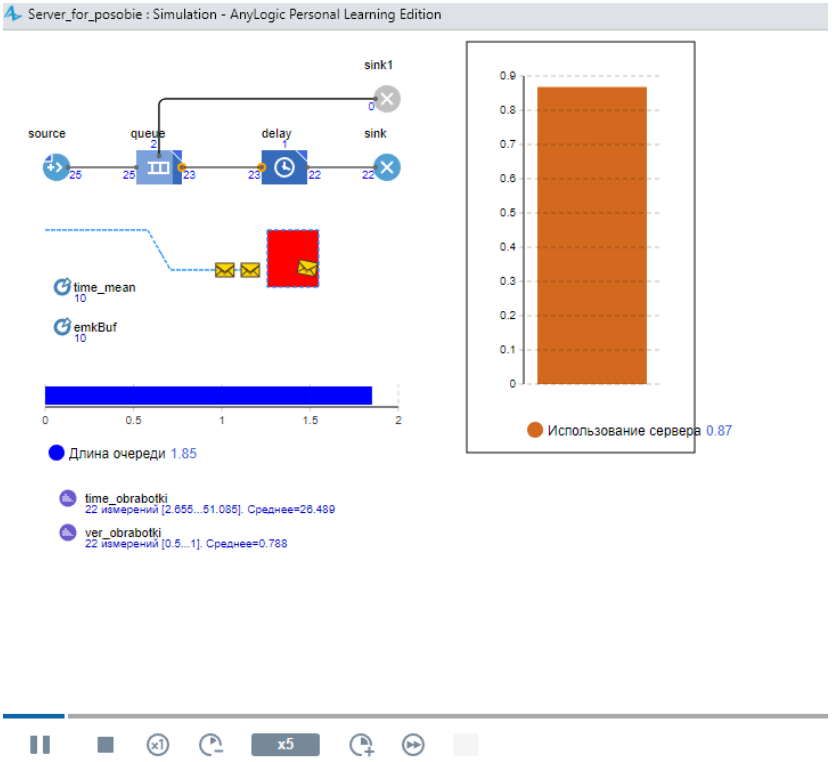


Рис. 33. Модель сервера в режиме запуска

Добавление параметров и элементов управления

Пусть вы хотите изменять среднее время обработки запросов **time_mean** в ходе моделирования. Используйте для этого элемент управления – бегунок.

1. Откройте палитру **Элементы управления** и перетащите элемент **Бегунок** из палитры на диаграмму класса **Main** (рис. 34).

2. Поместите бегунок под параметром **time_mean**, чтобы было понятно, что с помощью этого бегунка будет меняться среднее время обработки запросов объектом **Delay**.

3. Пусть вы хотите варьировать среднее время от 1 до 300. Поэтому введите 1 в поле **Минимальное значение**, а 300 – в поле **Максимальное значение** (рис. 34).

4. Установите флажок **Связать с** и в активизированное поле введите **time_mean**.

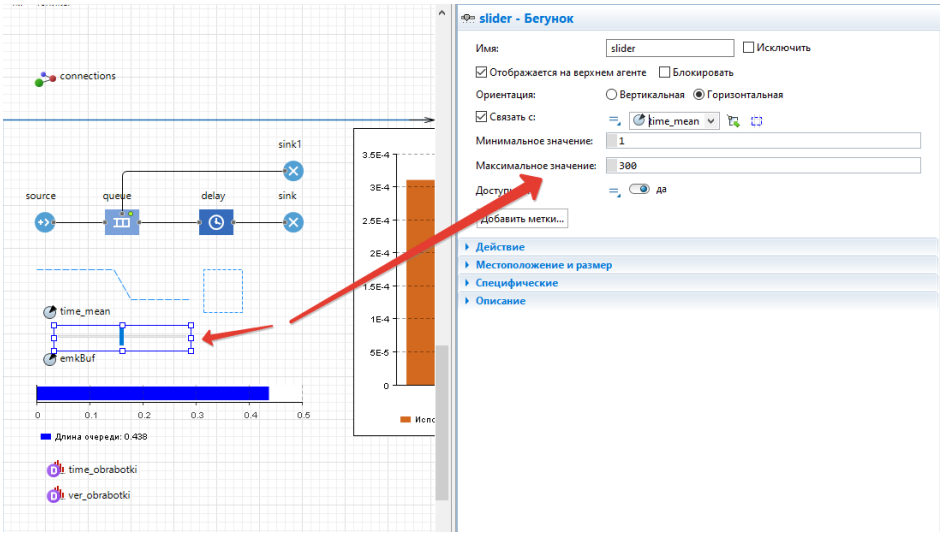


Рис. 34. Установка бегунка для параметра **time_mean**

5. Пусть теперь вы хотите также изменять ёмкость буфера в ходе моделирования. Используйте для этого также бегунок. Откройте палитру **Элементы управления** и перетащите элемент Бегунок из палитры на диаграмму класса **Main** (рис. 35).

6. Запустите модель. Теперь вы можете изменять в процессе моделирования ёмкость входного буфера и среднее время обработки запросов с помощью бегунков. Можете также командой **Приостановить** работу модели, изменить значения параметров, а затем продолжить моделирование.

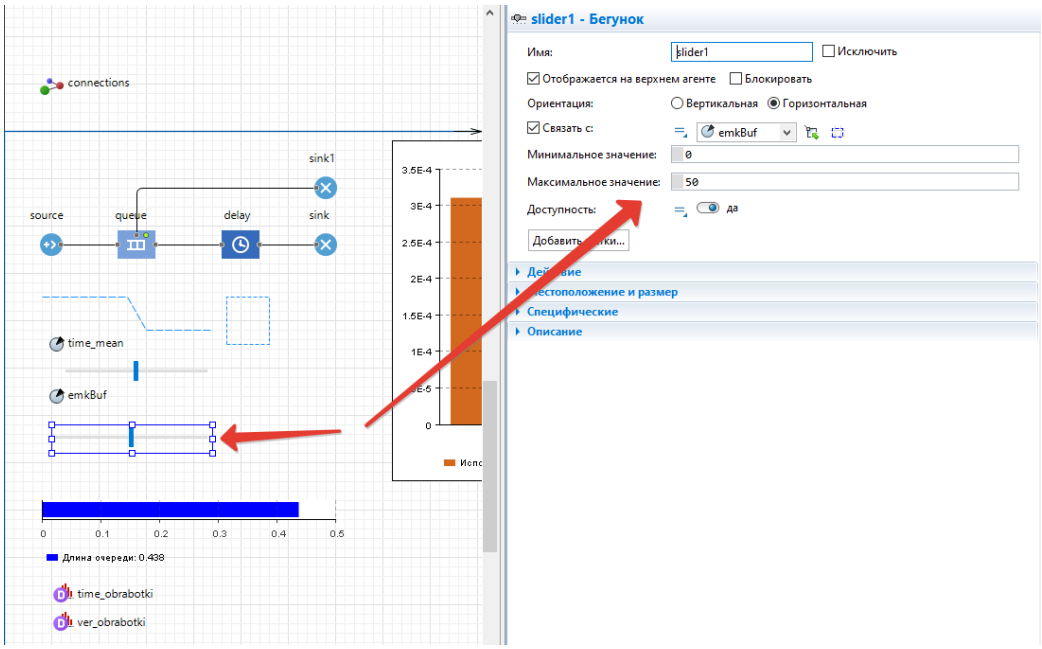


Рис. 35. Установка бегунка для параметра **emkBuf**

Добавление гистограмм

1. Теперь добавим на диаграмму нашего потока гистограмму, которая будет отображать собранную временную статистику.

Перетащите элемент **Гистограмма** из палитры **Статистика** в то место графического редактора, куда хотите ее поместить.

Укажите, какой элемент сбора данных хранит данные, которые вы хотите отображать на гистограмме: щёлкните кнопку **Добавить данные** и введите в поле **Данные** имя соответствующего элемента: **time_obrabotki**.

Установите галочку в поле **Отображать среднее.** В поле **Заголовок**: введите **Histogram Time obrabotki** (рис. 36).

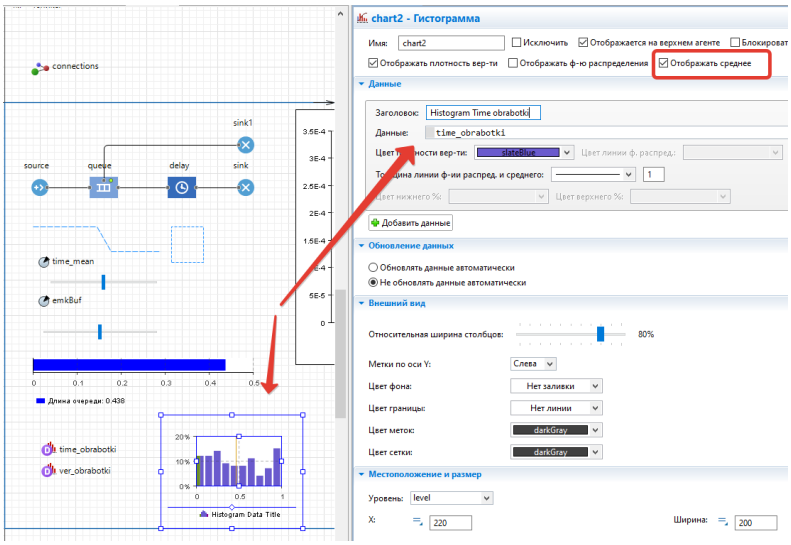


Рис. 36. Установка гистограммы для параметра time_**obrabotki**

2. Добавим на диаграмму нашего потока гистограмму, которая будет отображать собранную вероятностную статистику. Перетащите элемент **Гистограмма** из палитры **Статистика**.

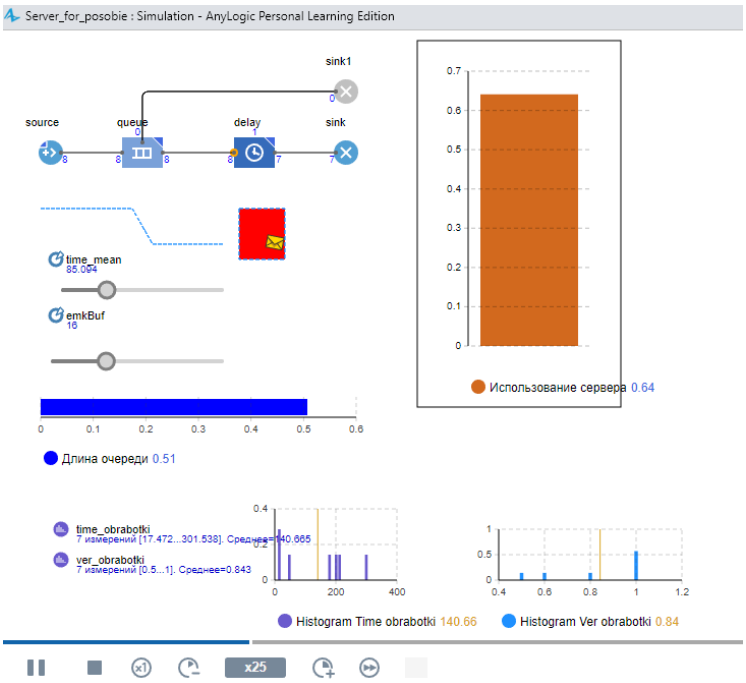


Рис. 37. Фрагмент работы модели с элементом управления и гистограммами

Щёлкните кнопку **Добавить данные** и введите в поле **Данные имя элемента** значение **ver_obrabotki**. Установите **Отображать среднее**.

В поле **Заголовок** введите **Histogram Ver obrabotki**.

3. Запустите модель. Фрагмент работы показан на рис. 37.

Исследовательские задачи

1. При каких сочетаниях параметров модели получим значение вероятности обработки заявки (на уровне 0.90–0.95)?

2. При каких параметрах высоконагруженная система будет иметь отказы в обслуживании (емкость буфера, интенсивность поступления заявок)?

3. Как повлияет на среднюю длину очереди увеличение производительности сервера?

4. Как изменятся характеристики системы, если увеличить производительность сервера?

5. Если увеличить емкость входного буфера, то как изменится среднее время обработки запроса и средняя длина очереди?

6. Если увеличить интенсивность поступления запросов, то как это повлияет на количество обработанных запросов?

7. Как изменятся характеристики системы, если увеличить производительность сервера?

1. 2. AnyLogic-модель банковского отделения

Цели лабораторной работы:

- Познакомиться с моделью системы массового обслуживания на примере работы банковского отделения.
- Провести исследование значения параметров длины очереди и времени на обслуживание заявки в зависимости от количества точек обслуживания.

AnyLogic поддерживает дискретно-событийный, или, если быть более точным, «процессный» подход в моделировании. С помощью блоков **Библиотеки моделирования процессов** вы можете моделировать системы реального мира, динамика которых представляется как последовательность операций (**прибытие, задержка, захват ресурса, разделение,** …) над агентами, представляющими клиентов, документы, звонки, пакеты данных, транспортные средства и т.п. Эти агенты сами не контролируют свою динамику, но могут обладать определёнными атрибутами, влияющими на процесс их обработки (например, тип звонка, сложность работы) или накапливающими статистику (общее время ожидания, стоимость).

Диаграммы процессов AnyLogic иерархичны, масштабируемы, расширяемы и объектно-ориентированы, что позволяет пользователю моделировать сложные системы любого уровня детальности. Другой важной особенностью **Библиотеки моделирования процессов** является возможность создания достаточно сложных анимаций процессных моделей.

В этом разделе мы создадим модель простой системы обслуживания, а именно модель банковского отделения. В банковском отделении находятся банкомат и стойки банковских кассиров, что позволяет быстро и эффективно обслуживать посетителей банка. Операции с наличностью клиенты банка производят с помощью банкомата, а более сложные операции, такие как оплата счетов — с помощью кассиров.

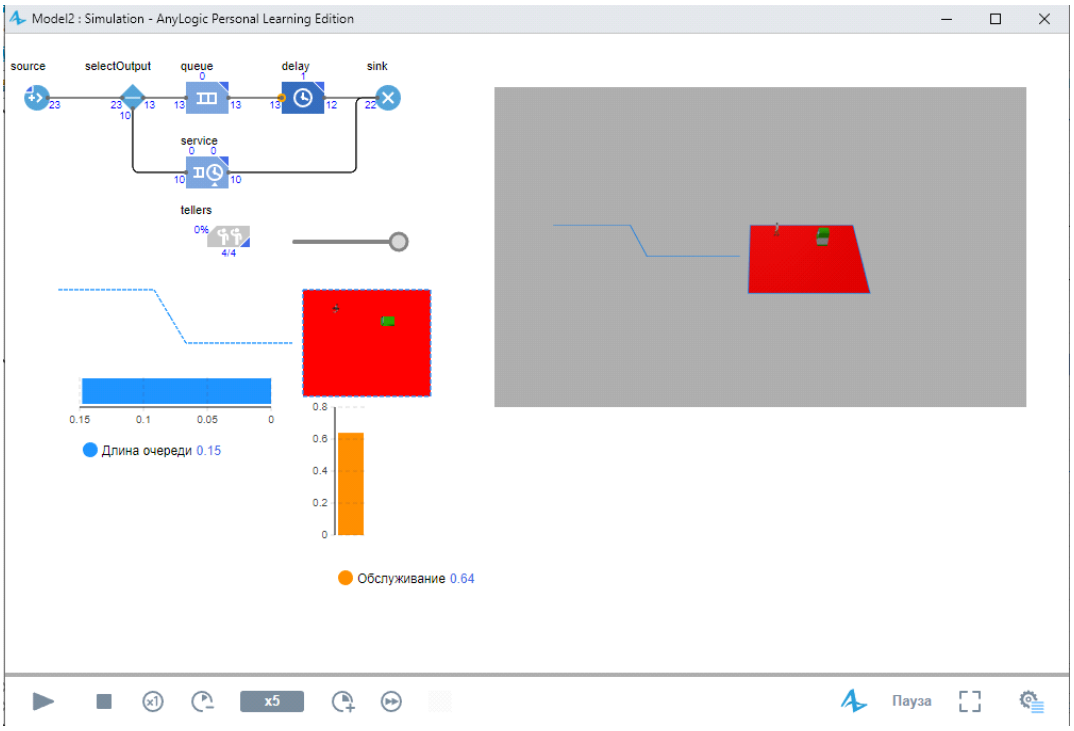


Рис. 1. Конечный результат создания модели отделения банка

Создадим простейшую модель, в которой будем рассматривать обслуживание людей банкоматом.

Создайте новую модель. Для этого:

- Щелкните мышью по кнопке панели инструментов **Создать** . Появится диалоговое окно **Новая модель**.
- Задайте имя новой модели. В поле **Имя модели** введите **Bank.**

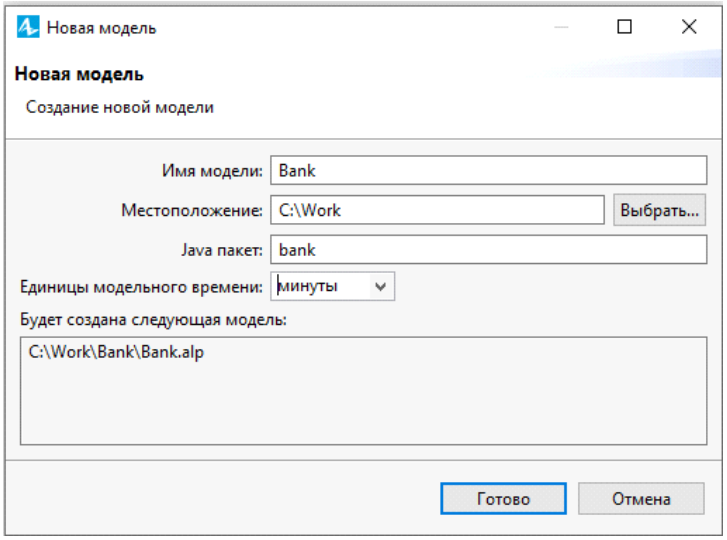


Рис. 2. Создание проекта

• Выберите каталог, в котором будут сохранены файлы модели. Если вы хотите сменить предложенный по умолчанию каталог на какой-то другой, вы можете ввести путь к нему в поле **Местоположение** или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося по нажатию на кнопку **Выбрать**.

• Выберите **минуты** в качестве **Единиц модельного времени**.

• Щелкните мышью по кнопке **Готово**, чтобы завершить процесс.

Вы создали новую модель. В ней уже имеется один тип агента **Main** и эксперимент **Simulation**. Агенты – это главные строительные блоки модели AnyLogic. В нашем случае агент **Main** послужит местом, где мы зададим всю логику модели: здесь мы расположим чертеж банковского отделения и зададим диаграмму процесса потока клиентов.

В центре рабочей области находится графический редактор диаграммы типа агента **Main**.

В левой части рабочей области находятся панель **Проекты** и панель **Палитра**. Панель **Проекты** обеспечивает легкую навигацию по элементам моделей, открытых в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева. Панель **Палитра** содержит разделенные по палитрам элементы, которые могут быть добавлены на диаграмму типа агента или эксперимента.

В правой рабочей области будет отображаться панель **Свойства**. Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели. Когда вы выделяете какой-либо элемент, например, в панели **Проекты** или графическом редакторе, панель **Свойства** показывает свойства выбранного элемента.

Теперь мы можем настроить нашу модель, созданную с помощью **Мастера создания модели**.

Создание диаграммы процесса

Теперь мы зададим динамику процесса, создав диаграмму из блоков **Библиотеки моделирования процессов**.

Каждый блок задает определенную операцию, которая будет производиться над проходящими по диаграмме процесса агентами.

Диаграмма процесса в AnyLogic создается путем добавления блоков библиотеки из палитры на диаграмму агента, соединения их портов и изменения значений свойств блоков в соответствии с требованиями вашей модели.

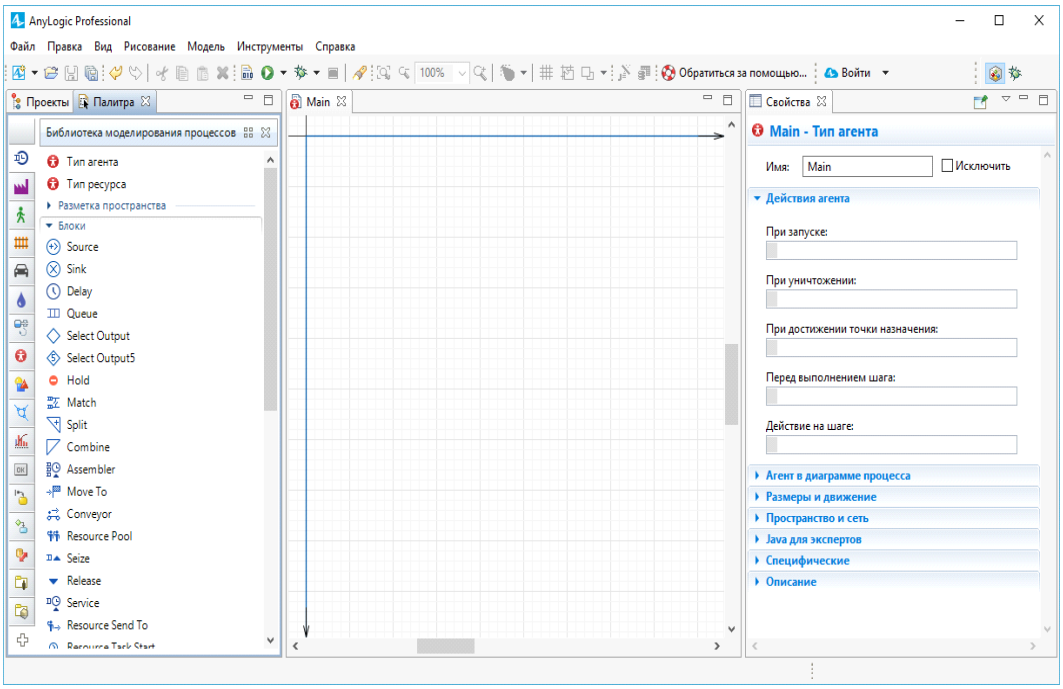


Рис. 3. Рабочая область проекта

Создайте диаграмму процесса

- По умолчанию при создании новой модели в панели **Палитра** открывается **Библиотека моделирования процессов**. Вы можете открывать палитры щелчком по соответствующей иконке на вертикальной панели слева от палитры:

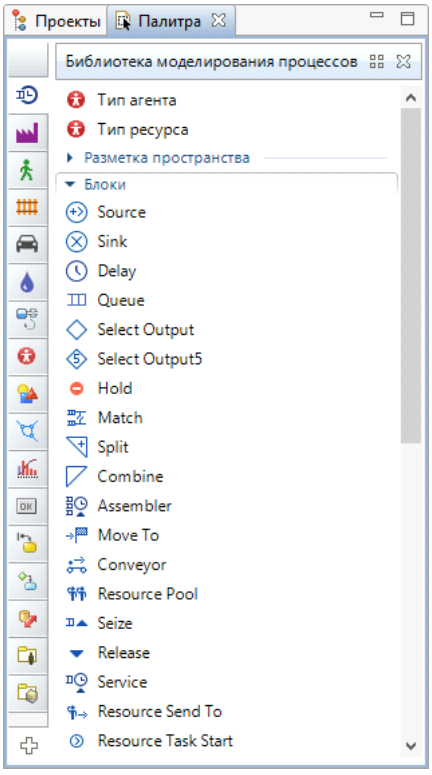


Рис. 4. Палитра **«Библиотеки моделирования процессов»**

• Добавьте блоки **Библиотеки моделирования процессов** на диаграмму и соедините их, как показано на рисунке. Чтобы добавить блок на диаграмму, перетащите требуемый элемент из палитры в графический редактор.

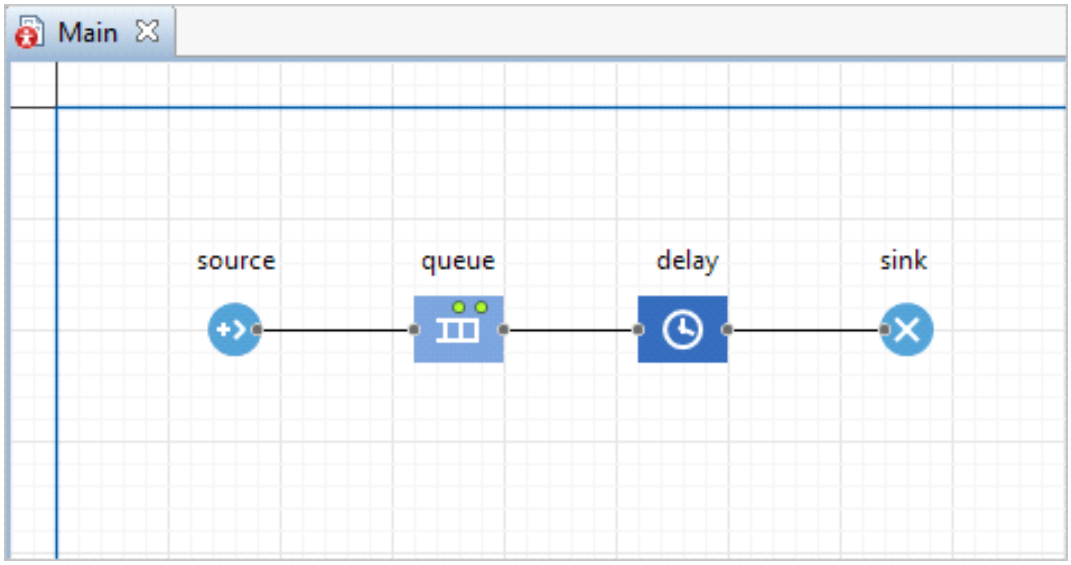


Рис. 5. Диаграмма процесса

• Когда вы перетаскиваете блоки и располагаете их рядом друг с другом, то можете видеть, как появляются соединительные линии между блоками. Будьте внимательны, эти линии должны соединять только порты, находящиеся с правой или левой стороны иконок.

Данная схема моделирует простейшую систему очереди, состоящую из источника агентов, задержки (и очереди перед задержкой) и финального уничтожения агентов.

Блок **Source** генерирует агентов определенного типа. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток агентов. В нашем примере агентами будут посетители банка, а блок **Source** будет моделировать их приход в банковское отделение.

Блок **Queue** моделирует очередь агентов, ожидающих приема блоками, следующими за данным в **диаграмме процесса**. В нашем случае он будет моделировать очередь клиентов, ждущих освобождения банкомата.

Блок **Delay** задерживает агентов на заданный период времени, представляя в нашей модели банкомат, у которого посетитель банковского отделения тратит свое время на проведение необходимой ему операции.

Блок **Sink** уничтожает поступивших агентов. Обычно он используется в качестве конечной точки потока агентов (и диаграммы процесса соответственно).

За детальным описанием блоков **Библиотеки моделирования процессов** обращайтесь к Справочному руководству по библиотеке моделирования процессов.

Настройте блоки диаграммы

• Чтобы изменить свойства элемента, выделите элемент в графическом редакторе или в панели **Проекты**, щелкнув по нему мышью. Свойства элемента откроются в панели **Свойства**.

• Выделите блок **Source**. В панели **Свойства** укажите, как часто должны прибывать клиенты. Введите 0.3 и выберите **в минуту** в поле **Интенсивность прибытия**.

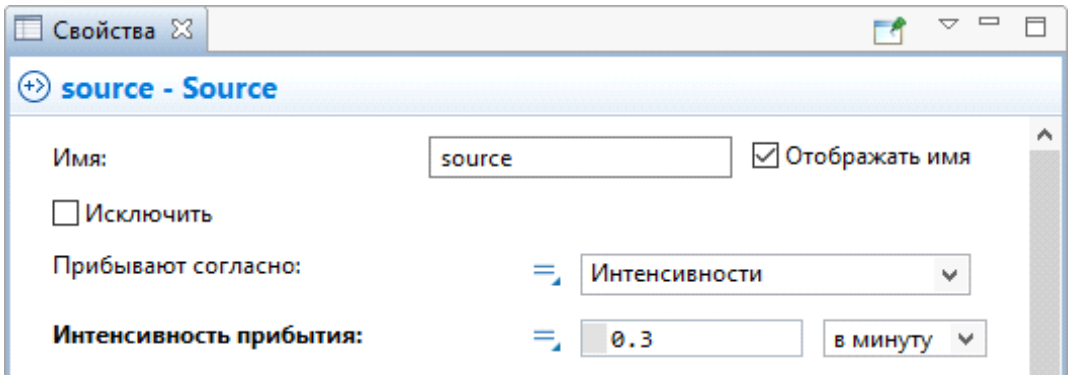


Рис. 6. Свойства блока **Source**

• Измените свойства блока **Queue**. Введите в поле **Вместимость** 15. В очереди будут находиться не более 15 человек.

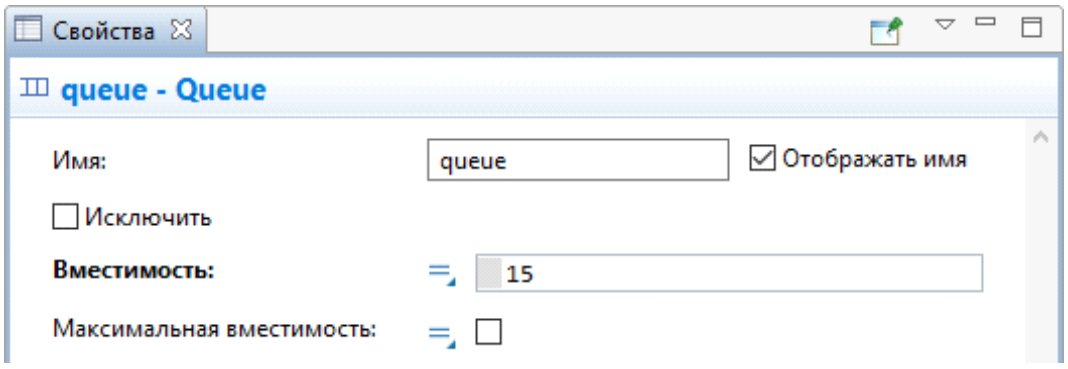


Рис. 7. Свойства блока **Queue**

• Подпишите над блоком **delay** его название **– ATM** (Automated Teller Machine) — аппарат для выдачи и приёма денег, оплаты услуг и погашения кредитов без участия сотрудника банка, с использованием банковских карт.

Automated Teller Machine — аппарат для выдачи и приёма денег, оплаты услуг и погашения кредитов без участия сотрудника банка, с использованием банковских карт). Подписать блок можно с помощью инструмента **Текст** палитры **Презентация**.

Задайте время обслуживания в поле **Время задержки**, распределенное по треугольному закону со средним значением, равным 1,5, минимальным, равным 0,8 и максимальным – 3,5 минутам.

Функция **triangular()** является стандартной функцией генератора случайных чисел AnyLogic. AnyLogic предоставляет функции и других случайных распределений, таких как нормальное, равномерное и т. д.

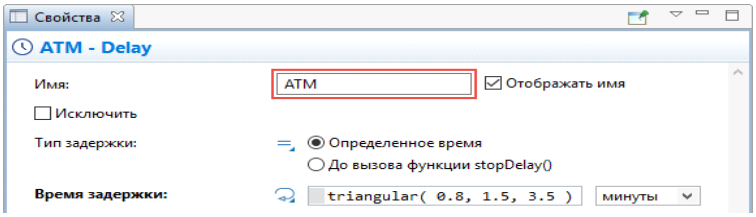


Рис. 8. Свойства блока **Delay**

Запуск модели

Мы закончили моделирование простейшей системы очереди и готовы запустить созданную модель. Сначала постройте вашу модель с помощью кнопки панели инструментов **Построить модель** (при этом в рабочей области AnyLogic должен быть выбран какой-то элемент именно этой модели). Если в модели есть какие-нибудь ошибки, то построение не будет завершено, и в панель **Ошибки** будет выведена информация об ошибках, обнаруженных в модели. Двойным щелчком мыши по ошибке в этом списке вы можете перейти к месту ошибки, чтобы исправить ее.

После того, как вы исправите все ошибки и успешно построите вашу модель, вы можете ее запустить. Запуская модель, вы автоматически обновляете ее.

Запустите модель

• Щелкните мышью по кнопке панели инструментов **Запустить** и выберите из открывшегося списка эксперимент, который вы хотите запустить. Эксперимент этой модели будет называться **Bank/Simulation**.

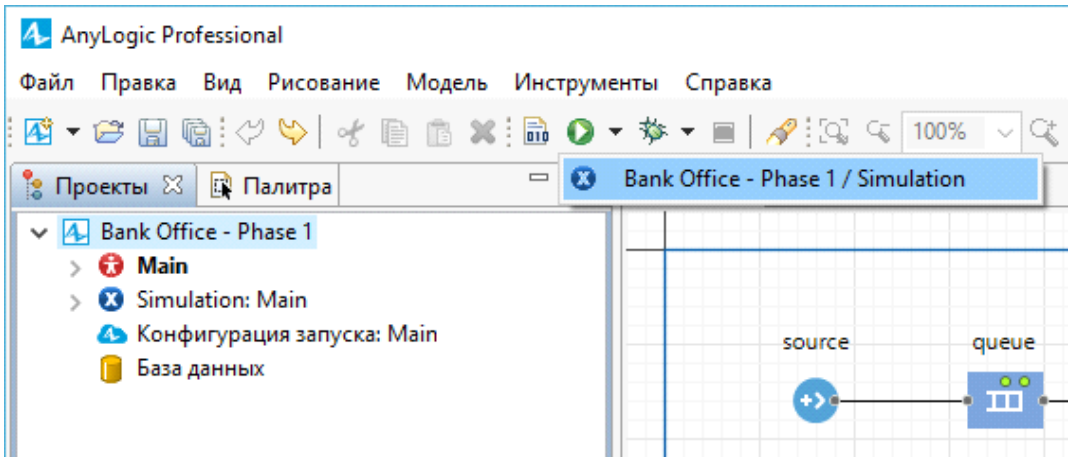


Рис. 9. Запуск эксперимента

На момент запуска этого конкретного эксперимента наша модель – единственная открытая модель в рабочем пространстве. В дальнейшем будет запускаться тот эксперимент, который запускался вами в последний раз. Чтобы выбрать какой-то другой эксперимент, вам будет нужно щелкнуть правой кнопкой мыши по этому эксперименту в панели **Проекты** и выбрать пункт **Запустить** из контекстного меню.

Запустив модель, вы увидите окно модели. В нем будет отображена презентация агента верхнего уровня модели. По умолчанию это тип агента **Main**.

Для каждой модели, созданной с помощью блоков **Библиотеки моделирования процессов**, автоматически создается блок-схема с наглядной визуализацией процесса, с помощью которой вы можете изучить текущее состояние модели, например, длину очереди, количество обслуженных человек и так далее.

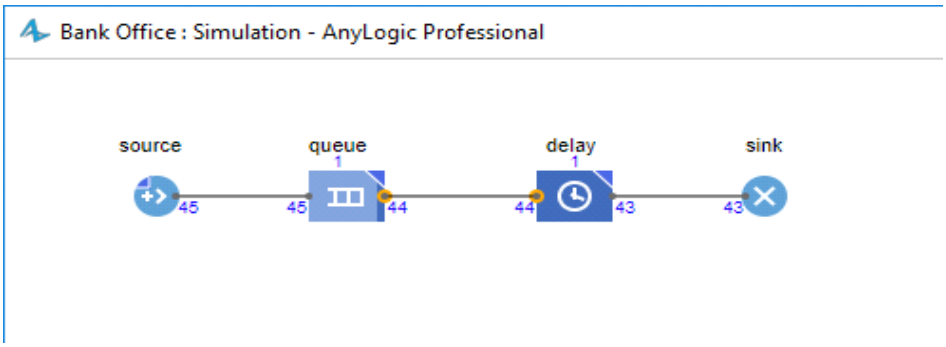


Рис. 10. Схема визуализацией процесса

Вы можете изменить скорость выполнения модели с помощью кнопок панели инструментов **Замедлить** и **Ускорить**.

Дополнительное задание

Представленную модель Банка можно дополнить визуальными компонентами и организовать сбор статистических данных на диаграмме. Наглядная анимация позволяет лучше понять происходящие в модели процессы и представить модель не только как данные, но и как поведение агентов.

Банковское отделение – это, прежде всего, некое помещение, пространство, в котором работают операторы и установлены банкоматы. Обработка поступающих заявок сотрудниками банка и банкоматами может создавать очереди из клиентов банка, которую можно представить в виде агентов (точек, пиктограмм).

Добавим к нашей модели анимацию банкомата.

- На панели **Палитра** в разделе **Разметка** пространства перетащите точечный узел, обозначающий банкомат  .

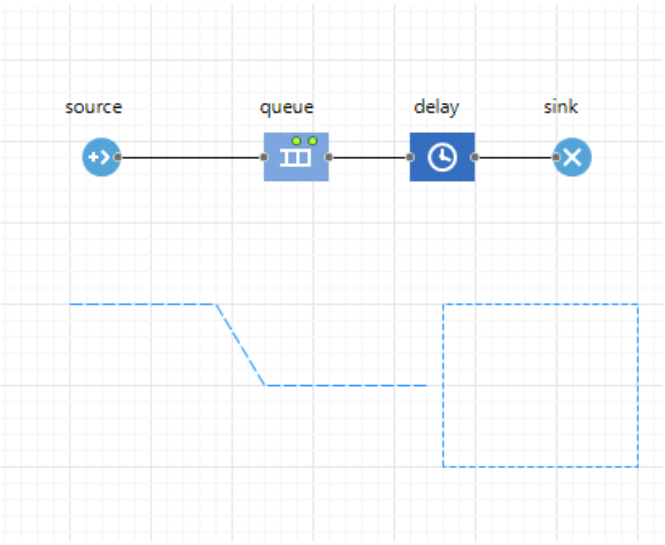


Рис. 11. Создание объектов для анимации банкомата и очереди

Создадим объекты **Банкомат** и **Очередь**. Нарисуйте очередь с помощью инструмента **Путь** (рис. 12).

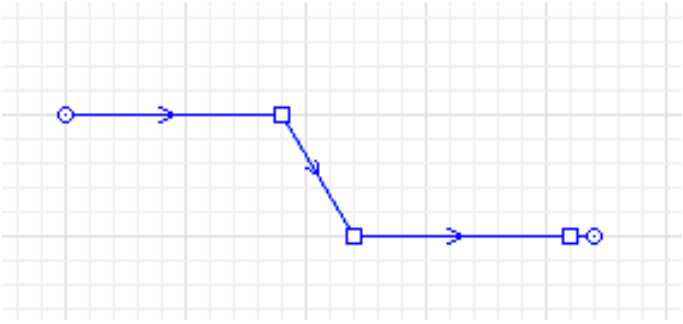
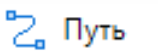


Рис. 12. Путь движения очереди

Для этого перейдите на вкладку **Палитра** и выберите раздел **Библиотека моделирования процессов**. Щелкните двойным щелчком мыши по кнопке **Путь** 

и поместите первую точку щелчком мыши по диаграмме. Щелкните в других местах диаграммы, чтобы добавить промежуточные точки. Последнюю точку добавьте двойным щелчком. Важно помнить, что рисовать нужно слева направо, при этом стрелками будет показано направление движения посетителей. После того, как нарисовали линию, задайте ей имя в окне свойств **GoToService**.

Нарисуйте прямоугольник с помощью инструмента Прямоугольный узел (**Палитра | Библиотека моделирования процессов**) (рис. 11, 13):

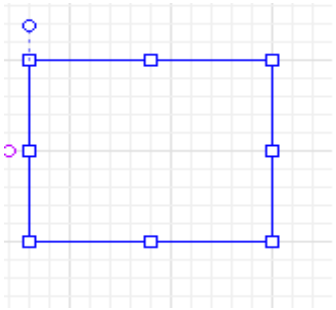


Рис. 13. Зона обслуживания

Назовите прямоугольник **servicePoint (зона обслуживания)**, задайте цвет заливки. Цвет будет меняться динамически, поэтому в поле **Цвет** нужно с помощью кнопки  выбрать **Динамическое значение**. В поле для изменения цвета нужно ввести строчку кода:

delay.size()>0 ? Color.red : Color.green

Задайте следующие свойства (рис. 14):

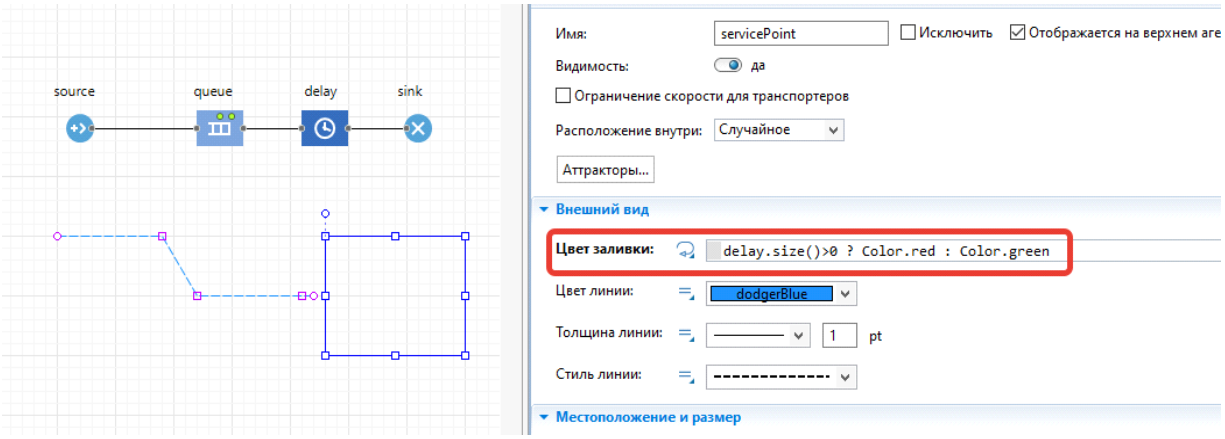


Рис. 14. Свойства для прямоугольника точки обслуживания

Здесь функция **size()** объекта **Delay** возвращает число человек, обслуживаемых в данный момент времени. Если банкомат занят, то цвет прямоугольника будет красным, в противном случае – зеленым. **Color** – это класс **Java**, позволяющий использовать стандартные цвета (черный, синий, красный, голубой, желтый и т. д.), и создавать любые другие.

• Теперь мы должны задать созданные анимационные объекты в качестве анимационных фигур объектов блок-схемы нашей системы.

Снова выделите объект **ATM (Delay)** и откройте его окно свойств. В поле **Место агентов** укажите из списка **servicePoint**. Проверьте (или укажите, если это не так), что в разделе **Специфические** стоит флажок у параметра **Одиночный агент**.

Выделите объект **Queue** на диаграмме процесса. В окне свойств укажите в поле **Место агентов** значение **GoToService**.

• Установим некоторые параметры эксперимента для данной модели. В окне Проекты выделите кликом объект **Simulation**, в окне свойств разделе **Модельное время** установите Реальное время со скоростью 5. Тем самым мы ускорим модель в 5 раз.

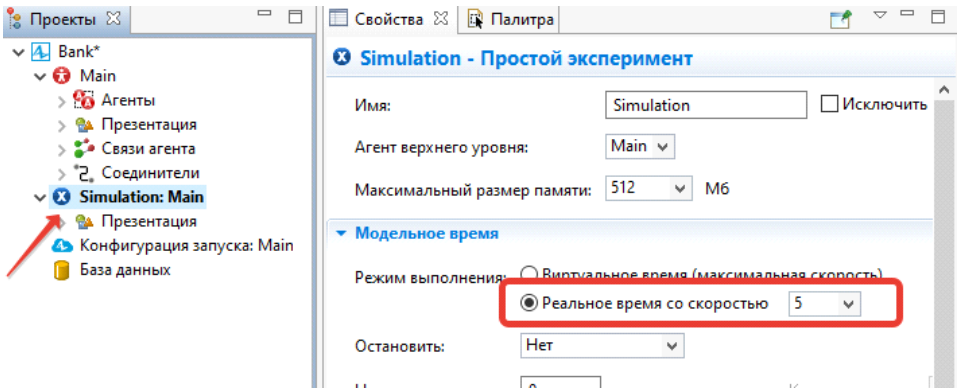


Рис. 15. Настройка симуляции модели

Запустите модель (рис. 16).

Щелкните по кнопке **Запустить** . Чтобы остановить выполнение, щелкните по кнопке Остановить . Модель начнет выполнение, и появится окно анимации. Запустив модель, вы увидите окно анимации. Цвет в прямоугольнике будет меняться в зависимости от того, обслуживается ли клиент в данный момент времени.

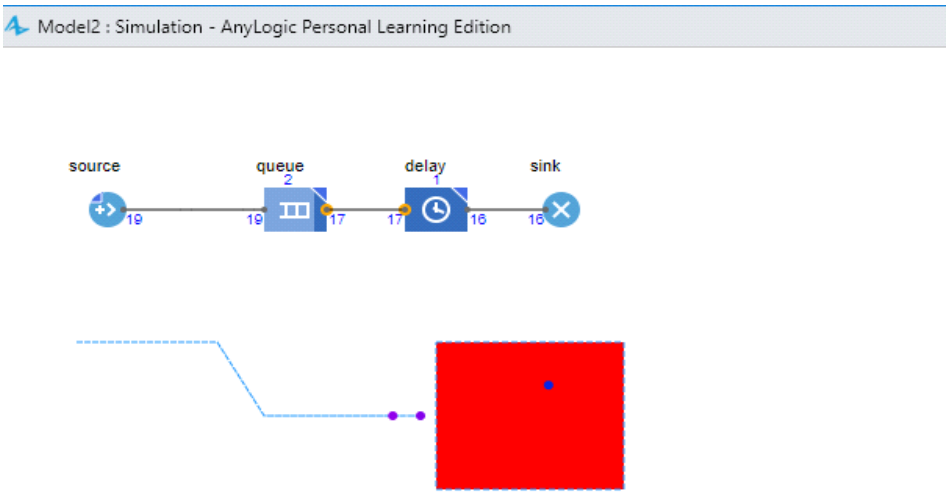


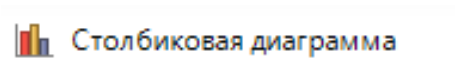
Рис. 16. Модель в запущенном состоянии

Сбор статистики

AnyLogic позволяет производить сбор сложной статистики. Для этого нужно лишь включить у объекта режим сбора статистики, поскольку по умолчанию он отключен для повышения скорости выполнения модели.

- Проверьте (или включите, если это не так) сбор статистики: для элементов **Queue** и **Delay** в окне свойств должен стоять флажок **Включить сбор статистики**.

Вы можете просмотреть собранную статистику с помощью диаграмм и графиков, или выводя числовые значения на анимацию. Мы же покажем статистику занятости банкомата с помощью индикатора.

- Добавьте индикатор на анимацию. Для этого перейдите на вкладку Палитра в раздел **Статистика**. Найдите элемент Столбиковая диаграмма 

и перенесите этот объект на свободное поле в окне структуры проекта. С помощью окна свойств установите **Направление столбцов** горизонтальное (рис. 17).

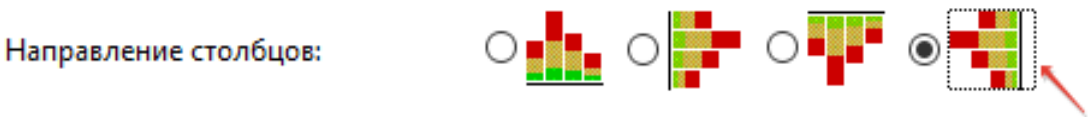


Рис. 17. Установка направления столбцов диаграммы

Установите **Масштаб** в значение **Фиксированный**.

На вкладке **Данные** оставьте только один элемент с данными. В поле **Заголовок** установите значение **Длина Очереди**, а в поле **Значение** напишите **queue.statsSize.mean() (**рис. 18**).**

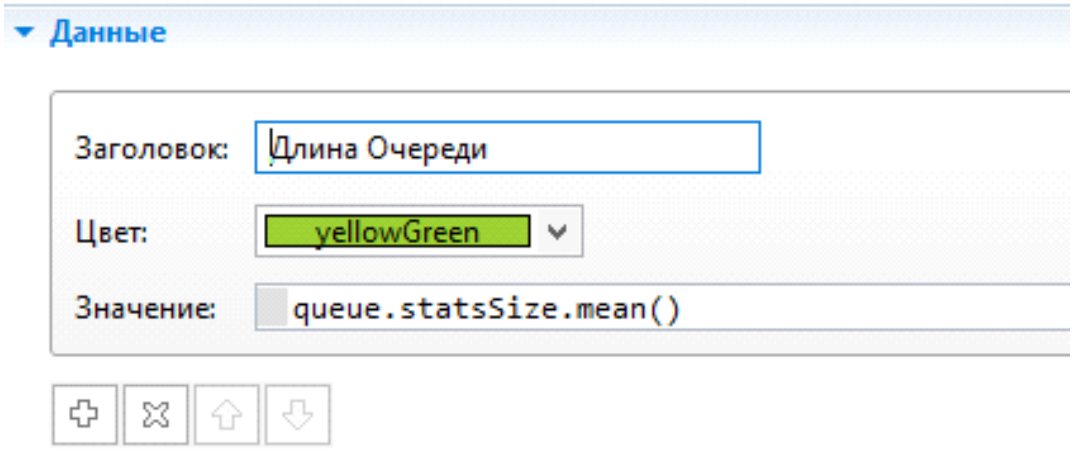


Рис. 18. Установка значений для диаграммы **Chart**

Здесь метод **mean()** – возвращает среднюю длину очереди.

- Вторую диаграмму расположите рядом с изображением банкомата. Назовите диаграмму «**Обслуживание**», а в качестве значения задайте выражение:

delay.statsUtilization.mean(),

задающее среднее время обслуживания заявки в процессоре. Направление столбцов **вертикальное**.

Установите **Масштаб** в значение **Фиксированный**.

- Запустите модель на выполнение.

Вид получившейся модели показан на рисунке 19.

Для ускорения работы модели переключитесь в режим виртуального времени . В режиме виртуального времени модель будет выполняться с максимально возможной скоростью.

Вы научились основам создания дискретно-событийных моделей с применением библиотеки моделирования процессов.

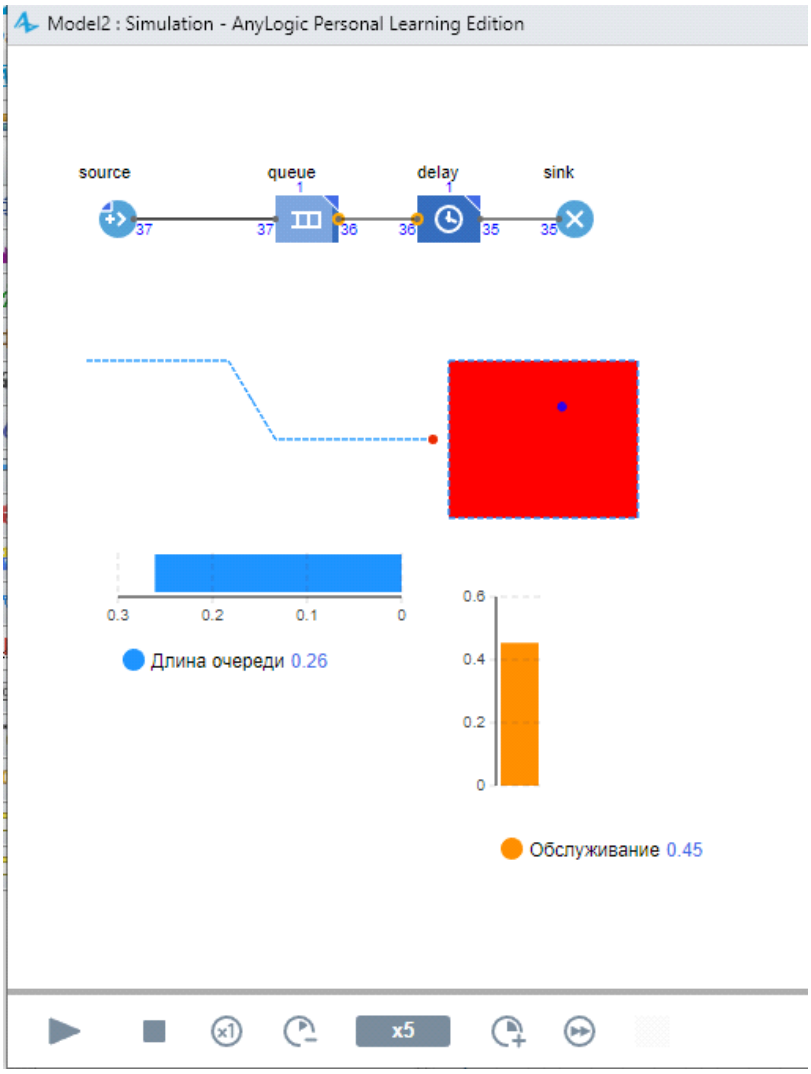


Рис. 19. Модель банка в режиме запуска с диаграммами

1.3. AnyLogic-модель билетной кассы

Смоделировать работу билетных касс. В кассы есть единая очередь, которую обслуживают две основные кассы. Если основные кассыне справляются с потоком покупателей, то открывается третья касса. Поток покупателей меняется в зависимости от времени суток и становится больше в выходные дни. Расписание потока покупателей приведено ниже.

Рабочие дни:

8:00–13:00 — десять человек в час; 13:00–16:00 — пятнадцать человек в час;16:00–22:00 — двадцать человек в час.

Выходные дни:

9:00–12:00 — 20 человек в час;12:00–21:00 — 40 человек в час.

Покупатели, время ожидания покупки у которых превысило час, уходят из касс, не купив билета. Время обслуживания одного покупателя в кассах меняется случайным образом от 2 до 15 минут и в среднем составляет 5 минут. Предусмотреть в модели учет, купивших и не купивших билеты.

Задание логики работы модели

Создание новой модели

Создайте новую модель (рис. 1) в среде AnyLogic, нажав на кнопку **Создать** и выбрав **Модель** из выпадающего меню.

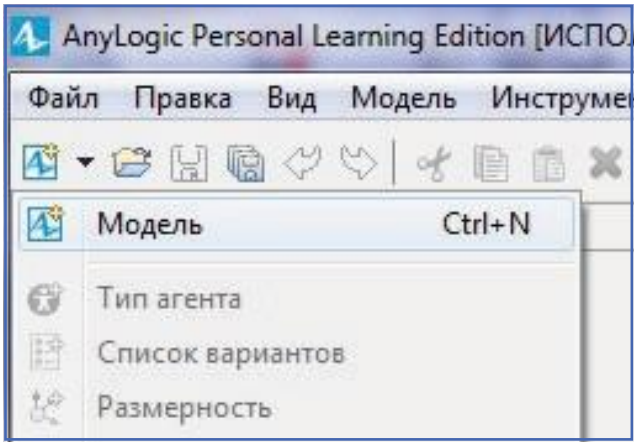


Рис. 1. Создание новой модели

Введите имя модели **ticket** в открывшемся диалоговом окне и задайте единицы модельного времени — минуты (рис. 2). Нажмите кнопку **Готово**.

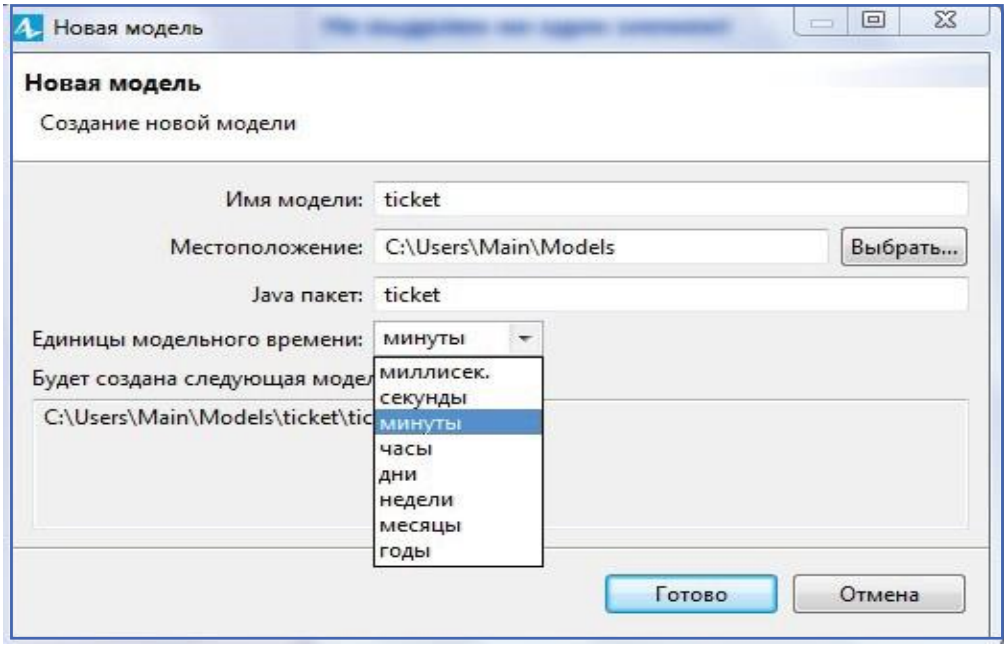


Рис. 2. Задание параметров новой модели

Моделирование прихода покупателей

В открывшемся окне модели перейдите на вкладку **Палитра** и откройте первую библиотеку из списка — **Библиотеку моделирования процессов**. Из нее перетащите на рабочее поле блок **Source** (рис. 3).Как известно, именно этот блок моделирует появление заявок в модели. В нашем случае покупатели — это заявки.

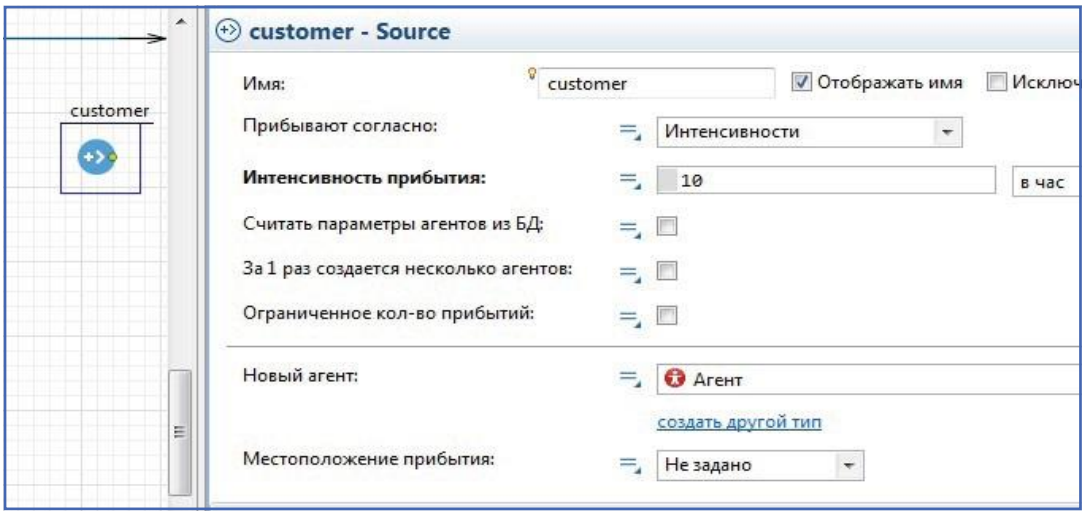


Рис. 3. Моделирование прихода покупателей

В начале моделирования мы не будем учитывать изменение интенсивности потока покупателей в течение дня и недели. Просто зададимнаименьшую границу потока. Для этого выберем в пункте **Прибывают согласно режиму Интенсивности** и зададим значение интенсивности 10. Это значит, что в среднем 10 раз в час будут приходить покупатели, то есть примерно каждые 6 минут.

Моделирование очереди покупателей

Перетащите из библиотеки моделирования процессов блок **Queue**.Будьте внимательны и постарайтесь сделать так, чтобы блок **Queue** связался с блоком **Source**. Это позволит избежать лишних хлопот по дорисовыванию связей между блоками. Если не удастся связать блоки, то нужно дважды щелкнуть на выходной порт блока **Source** и потянуть связь до блока **Queue**. Заявка проходит в модели только по связям.

Если блоки не связаны, то заявка не сможет перейти в следующий блок. Блок **Queue** имитирует накопление заявок в модели, фактически моделирует очередь. Оставим все ее параметры по умолчанию. Это значит,что ее логика будет соответствовать логике очереди «первый пришел,первый ушел» и не будет задано никакой привязки на местности. Отметить пункт Максимальная вместимость (рис. 4), поскольку в задании ничего не было сказано об ограничениях длины очереди.

Моделирование процесса покупки билетов

Для процесса покупки билетов требуется ресурс — кассиры. Для моделирования ресурсов в модели используется блок **ResourcePool**. Перетащите его на рабочее поле модели. Этот блок, не связывается ни с каким блоком в модели, так как через него не должны проходить заявки.

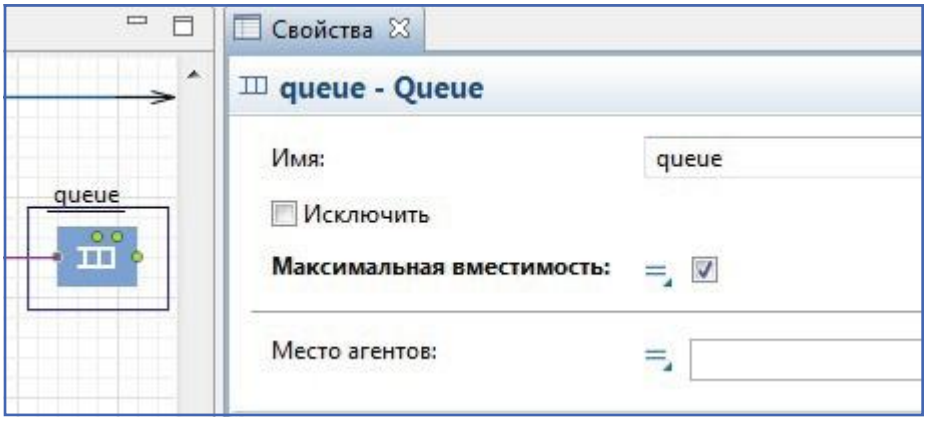


Рис. 4. Моделирование очереди покупателей

В свойствах блока задайте его имя **booking_clerk**. Поскольку кассирымогут передвигаться от кассы к кассе, то задайте тип ресурса как Движущийся и укажите их скорость 2 км/ч. Пока не будем задавать расписание работы кассиров и в пункте Количество задано оставим режим Напрямую и укажем количество кассиров, например, 2 (рис. 5).

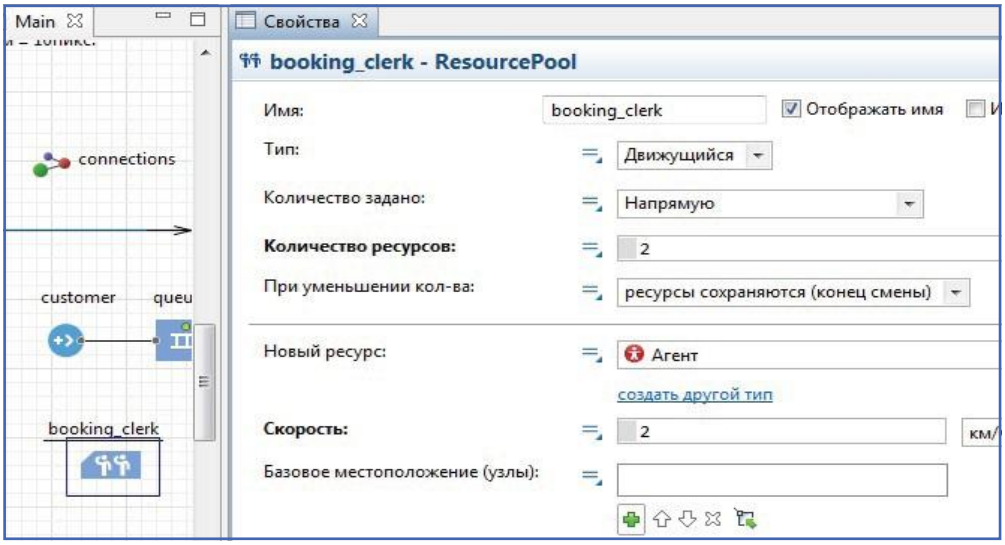


Рис. 5. Моделирование ресурсов для продажи билетов

Перед тем как моделировать сам процесс покупки билетов используем блок выбора движения заявок, поскольку покупатели из очереди могут идти как в первую, так и во вторую кассу. Перетащите блок **selectOutput** на рабочее поле и соедините его с блоком **Queue**. Условие выбора касс введем позже.

Далее нужно смоделировать сам процесс покупки билетов. Процесс покупки билетов – это процесс обслуживания заявки. Для этого используется блок **Service**, который включает в себя логику работы трех блоков **Sieze** (захват ресурсов), **Delay** (удержание заявки и ресурсов), **Release** (освобождение ресурсов). Также блок **Service** имеет собственную очередь, то есть в него еще встроен блок **Queue**. Поэтому если бы к каждой кассе шла своя очередь, то блок **Queue** отдельно можно было бы не использовать.

Перетащите два блока **Service** на рабочее поле. В свойствах блоковзадайте их имена **booking1 и booking2**. Задайте вместимость собственных очередей 2 и время задержки (рис. 6). Время задержки задается функцией треугольного распределения со средним значением 5 минут, минимальным значением 2 минуты и максимальным значением 15 минут. В пункте Набор(ы) ресурсов выберите из

списка **ресурсов**только что созданный ресурс **booking_clerk**. Теперь в первой и во второй кассах будут работать указанные в ресурсах 2 кассира.

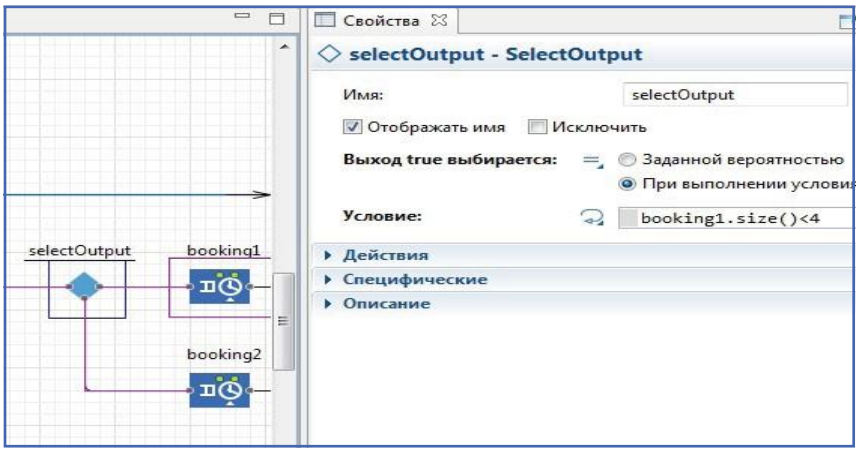


Рис. 6. Моделирование процесса продажи билетов

Теперь, когда обе кассы промоделированы, вернемся к условию выбора кассы покупателем. Пусть покупатель идет во вторую кассу, увидев, что в первой кассе собралось более трех человек. Такое поведениезаявкам можно задать, если использовать в условии выбора функцию объекта **service –size ()**, которая возвращает количество клиентов, обслуживаемых в данный момент объектом. Если величина, возвращенная функцией **size ()** объекта **booking1**, будет больше *3* (рис. 7), то значит, что в эту кассу покупатель не пойдет.

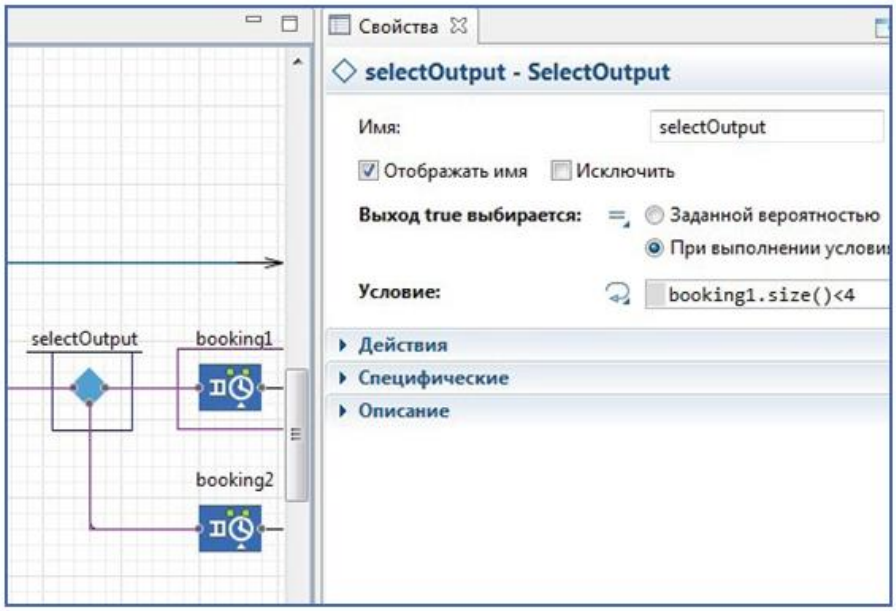


Рис. 7. Условие выбора касс

Моделирование ухода покупателей из касс

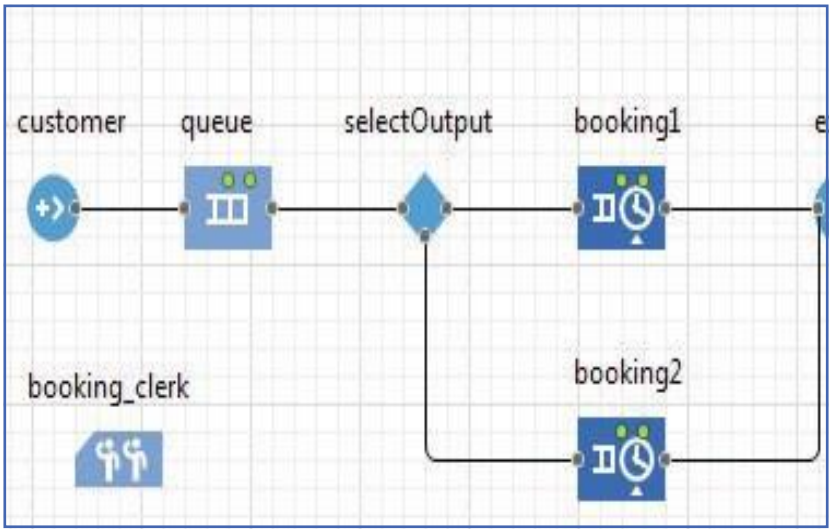


Рис. 8. Моделирование ухода покупателей из касс

Проверка работоспособности модели

Набросок модели закончен. Он должен выглядеть как на рис. 9.

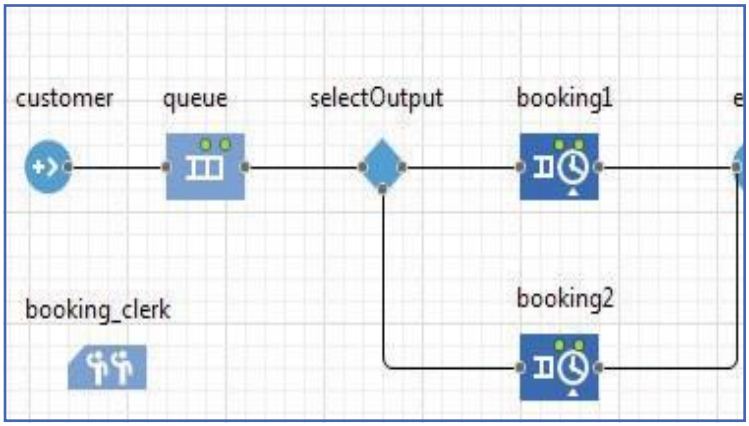


Рис. 9. Черновик модели

Запустите модель. Результат работы модели спустя некоторое модельное время должен выглядеть как на рис. 10.

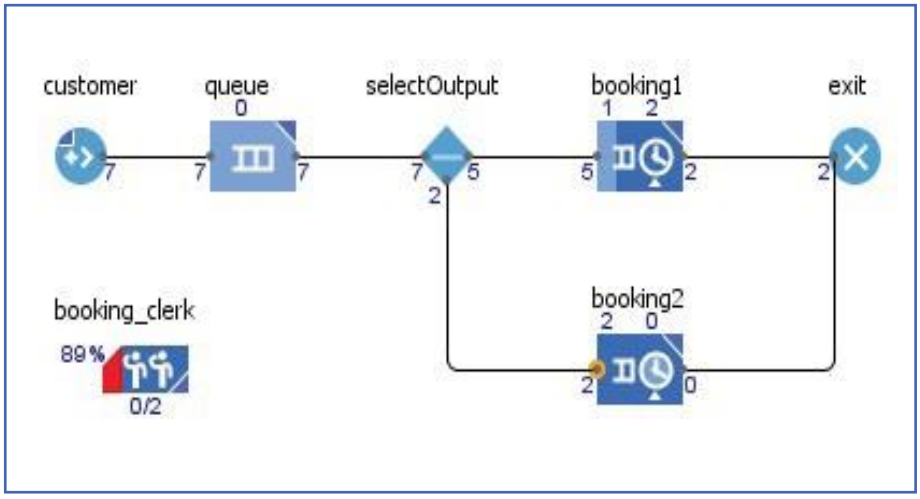


Рис. 10. Работа модели

Задание расписания прихода покупателей

В задании было сказано, что интенсивность прихода покупателей меняется в течение дня и становится больше в выходные. Любые периодические изменения в модели удобно задавать объектом Расписание. Перетащите объект Расписание из библиотеки на рабочее поле модели. Этот объект не нужно объединять ни с каким объектом.

В свойствах объекта задайте его имя **schedule_customer**. Поскольку будет задаваться расписанием Изменение интенсивности прибытия покупателей, то выбираем тип расписания — **Расписание интенсивности**. Тип расписания задается в разделе Данные. Значение по умолчанию оставляем 0, это будет означать, что в часы, не указанные в расписании, покупатели не приходят. Расписание будет задавать интервалы, в которых интенсивность остается постоянной. Задаем расписание на неделю, поэтому длительность расписания выбираем Неделя. Само расписание задается в таблице в пункте **Повторять расписание еженедельно**. Справа от таблицы есть кнопки для добавления (+), удаления (x) и прокрутки расписания. Нажмите на кнопку «+» для добавления новой строки в таблице. Каждая строка в таблице расписания задает один временной интервал, **начало** и **конец** которого задаются в соответствующих столбцах таблицы. В столбце Значение задайте значение интенсивности прибытия покупателей. В результате должно получиться расписание как на рис. 11.

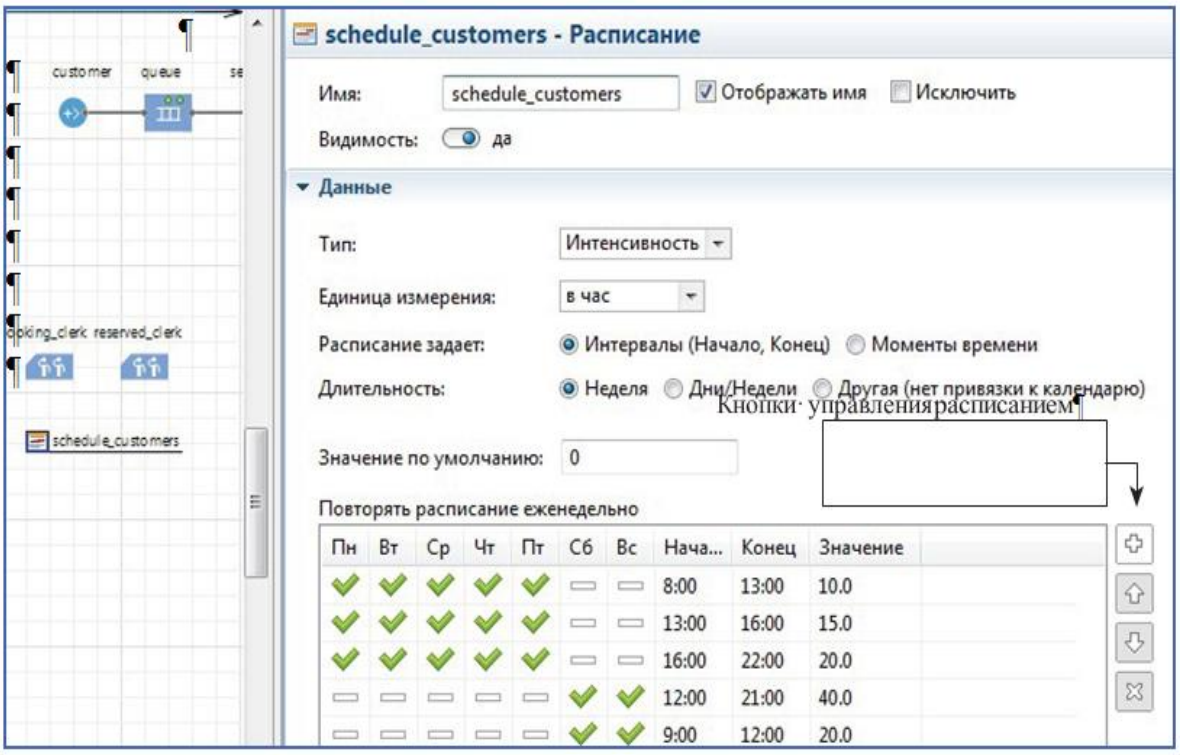


Рис. 11. Расписание интенсивности прибытия покупателей

Задание расписания работы кассиров

Расписание работы кассиров будет задавать время работы кассиров и количество работающих кассиров (рис. 12). Поскольку число кассиров — это целое число, то в пункте Тип выбираем целое. Значение по умолчанию задаем 0, значение в течение рабочих часов — 3.

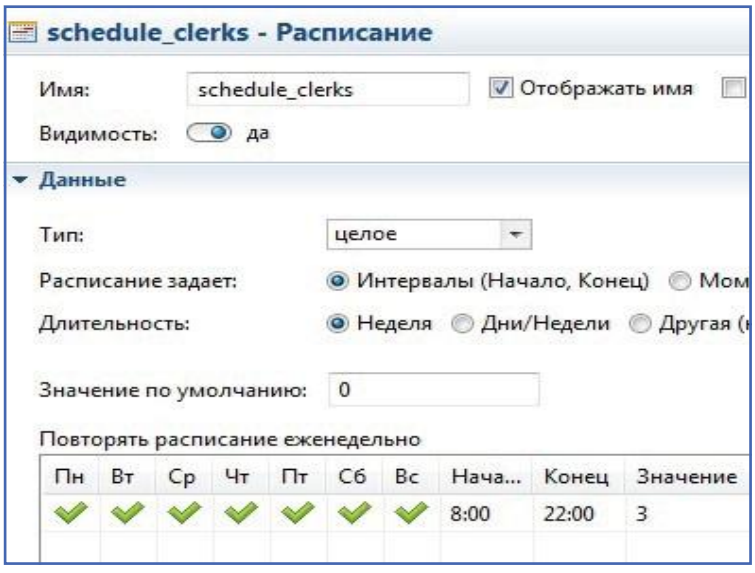


Рис. 12. Расписание работы кассиров

Задание расписания перерывов в работе кассиров

В работе кассиров предполагаются 15-минутные перерывы, в ходе которых кассиры как ресурсы не доступны. Следовательно, для задания расписания перерывов нужно создать расписание доступности, для которого выбираем Тип: да/нет (рис. 13). По умолчанию делаем ресурс доступным, то есть значение **да**. В столбце указываем значение **нет**. Также в свойствах расписания задаем его имя **schedule_timeout**.

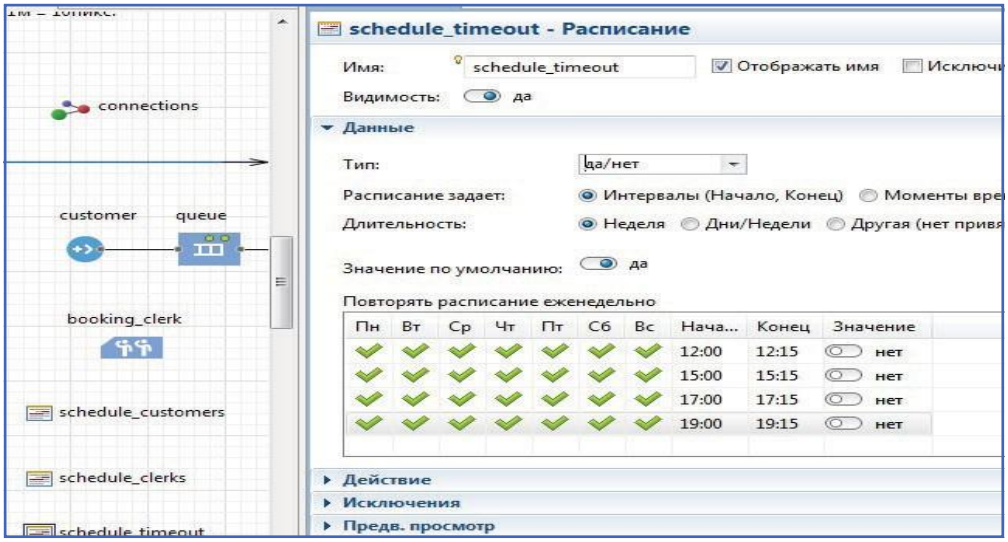


Рис. 13. Расписание перерывов работы кассиров

Привязка расписания прихода покупателей к объекту source

Вначале мы создавали объект **Source** и указывали, что заявки прибывают согласно интенсивности, а затем задавали интенсивность прибытия.

Расписанию интенсивностей (рис. 14).

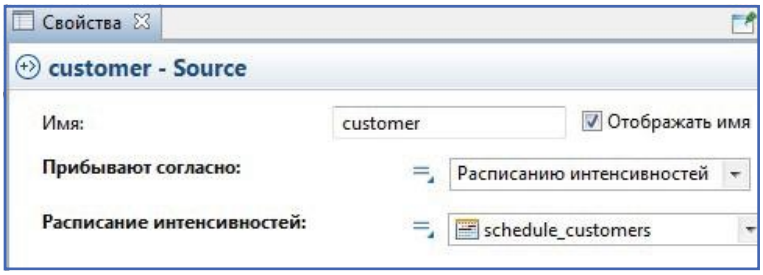


Рис. 14. Привязка расписания прибытия покупателей

Теперь пришла пора исправить ситуацию. В свойствах объекта **source** в пункте **Прибывают согласно** выберите вариант.

Появится новый пункт в свойствах **Расписание интенсивностей**. Выберите в этом пункте созданное ранее расписание прибытия покупателей.

Теперь покупатели будут приходить с разной интенсивностью в течение дня. Привязка расписания работы кассиров к объекту **ResourcePool.**

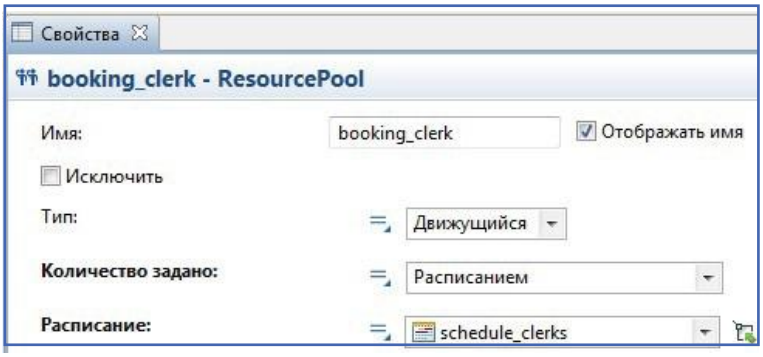


Рис. 15. Привязка расписания работы кассиров

Ранее нами количество ресурсов в объекте **booking_clerk** было задано напрямую. Теперь зададим расписание их работы (рис. 15).

В пункте **Количество задано** выберите вариант **Расписанием** и в появившемся пункте **Расписание** выберите ранее созданное расписание работы кассиров.

Привязка расписания перерывов в работе кассиров

Откройте раздел **Смены, перерывы, аварии, обслуживание**… в свойствах объекта **booking_clerk** (рис. 16). Поставьте галочку в пункте **Перерывы**. Появится окно, в котором нужно задать расписание перерывов. Выберите ранее созданное расписание перерывов работы кассировв пункте Расписание перерывов.

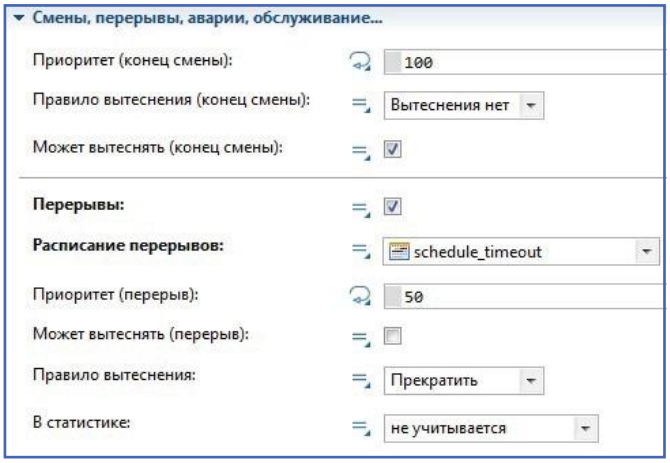


Рис. 16. Привязка расписания перерывов в работе кассиров

Моделирование резервной кассы

Для моделирования резервной кассы используется блок **service** (рис. 17) и все свойства задаются такие же, как и у первых двух касс. Только время обработки задается чуть меньше, чем в первых двух кассах.

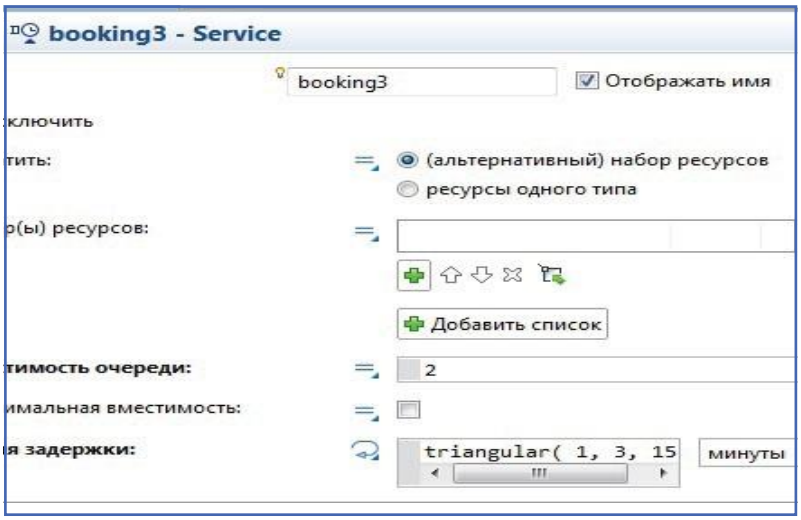


Рис. 17. Моделирование резервной кассы

Задание ресурсов для резервной кассы

В резервной кассе также будут работать кассиры, но из другой бригады, поэтому нужно создать новый ресурс. Назовем его **reserved_clerk** (рис. 18). Расписание работы кассиров резерва такое же, как и у основной бригады.

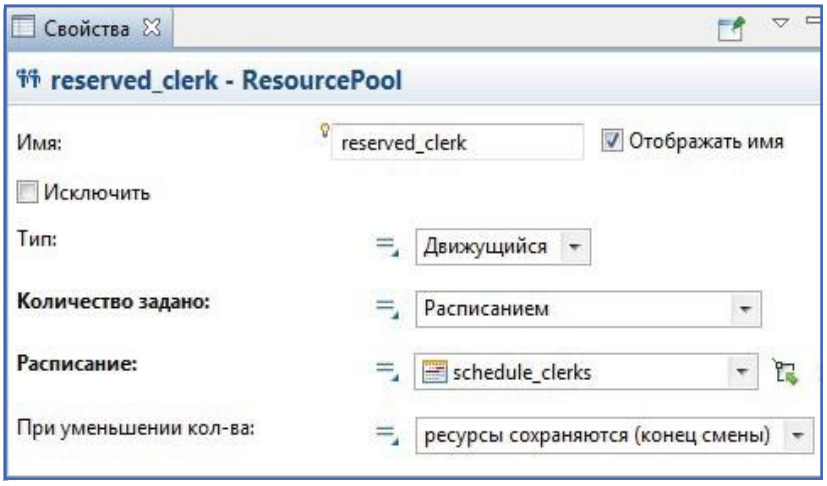


Рис. 18. Моделирование резервной бригады кассиров

Привязка ресурсов к резервной кассе

Для того чтобы привязать резервную бригаду к резервной кассе, укажите в пункте **Набор**(ы) ресурсов только что созданный ресурс **reserved_ clerk** (рис. 19). Выход кассы соедините с объектом **exit**.

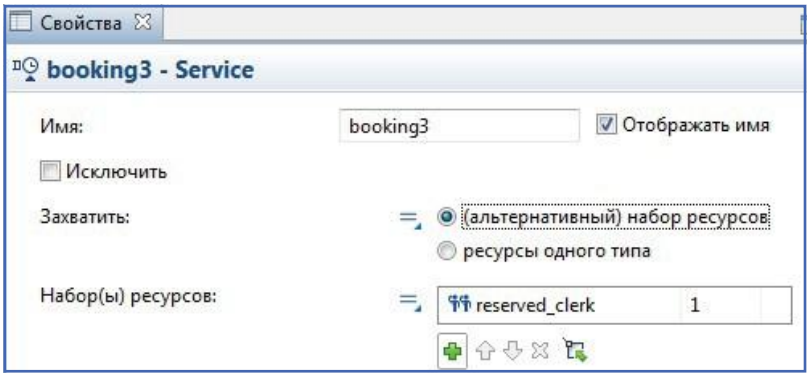


Рис. 19. Привязка ресурсов к резервной кассе

Организация ограничения времени ожидания в кассах

В условии задачи было сказано, что клиенты, чье ожидание покупки билета превысило час, уходят из касс, не купив билеты. Для реализации этого нужно задать тайм-аут, после которого заявка уходит из объекта **service** через выход **OutTimeout**. Время тайм-аута задается в разделе **Специфические** (рис. 20). Задайте тайм-ауты в первой и второй кассе, равные 30 минутам, в третьей кассе — 20 минут.

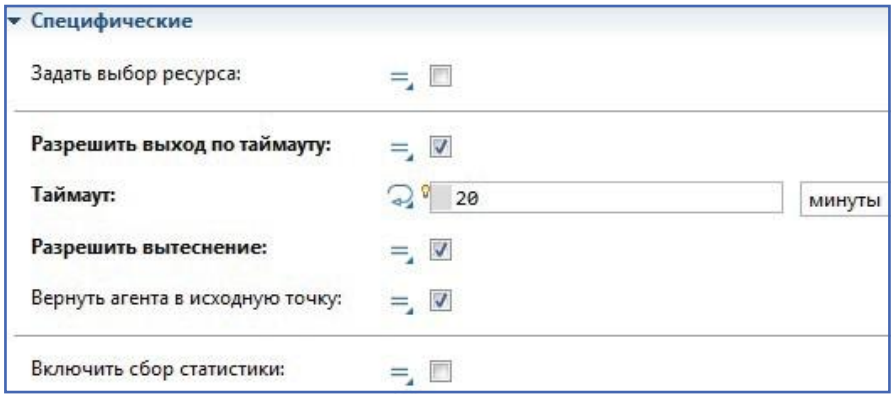


Рис. 20. Задание **тайм-аута**

Организация выхода заявок из первой и второй кассы, чье время ожидание превысило 30 минут

Соедините выходы **OutTimeout** объектов **booking1** и **booking2** со входом в объект **booking3**. Таким образом, покупатели, чье время ожидания в первой и второй кассах превысило 30 минут, идут в резервную кассу.

В итоге модель должна получиться, как показано на рис. 21.

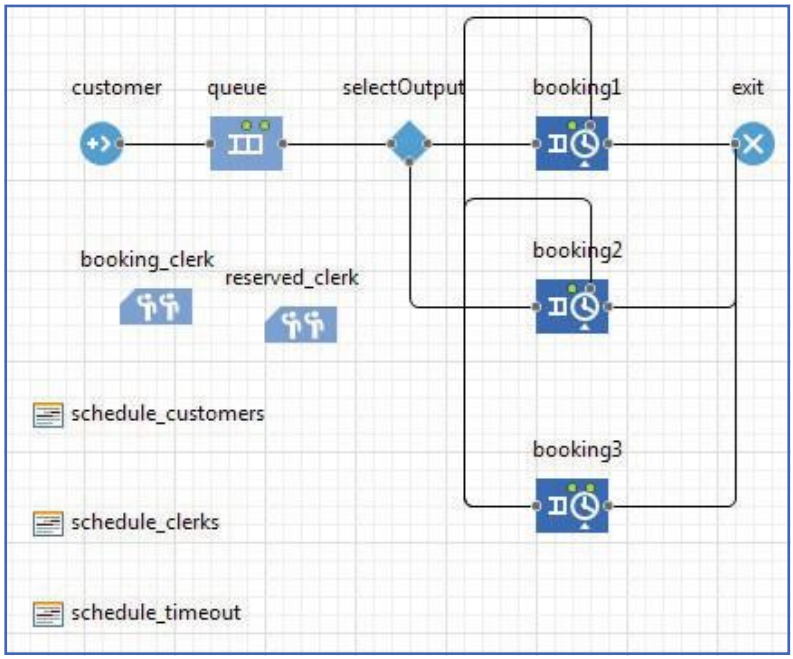


Рис. 21. Модель с учетом резервной кассы

Проверка работоспособности модели

Запустите модель. Спустя некоторое модельное время должно получиться, как показано на рис. 22.

Добавление объекта Часы

Для удобства просмотра работы модели нужно поместить на рабочее поле объект Часы, которые будут показывать текущее модельное время (рис. 23). Этот объект находится на вкладке Картинки в палитре инструментов. Перетащите его на рабочее поле.

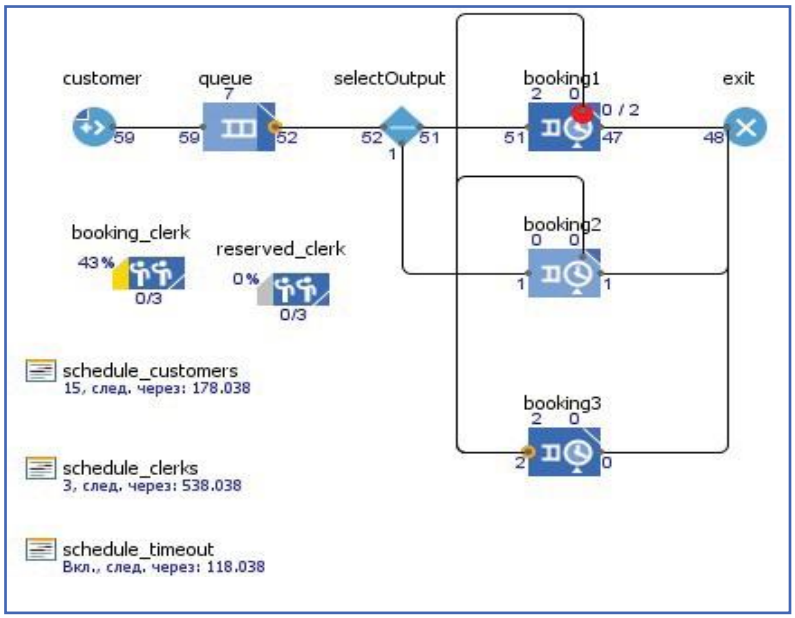


Рис. 22. Работа модели с учетом резервной кассы

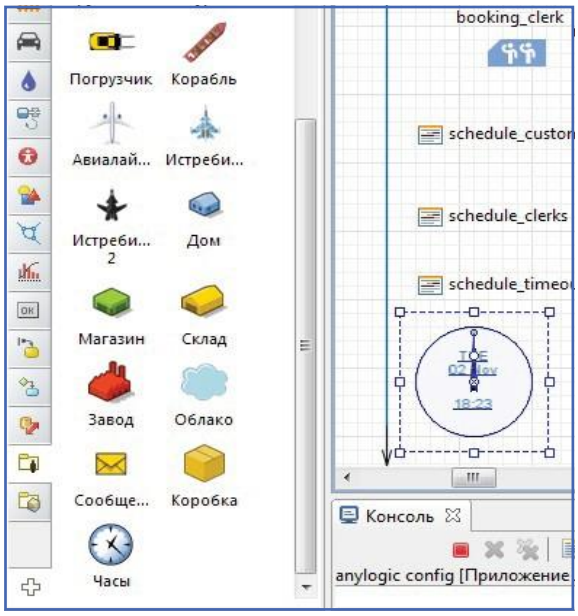


Рис. 23. Объект Часы

Организация ухода покупателей, не купивших билет

Уход покупателей, не купивших билет, моделируется так же, как и уход покупателей, купивших билет (рис. 24).

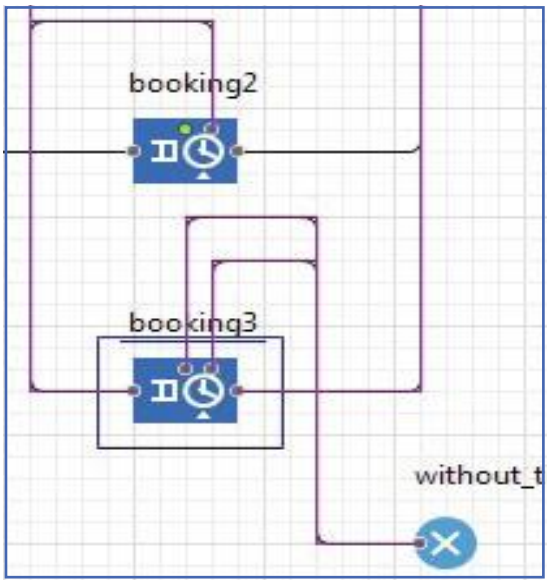


Рис. 24. Уход покупателей, не купивших билет

Проверка работоспособности модели

Запустите модель. Должно получиться как на рис. 25.

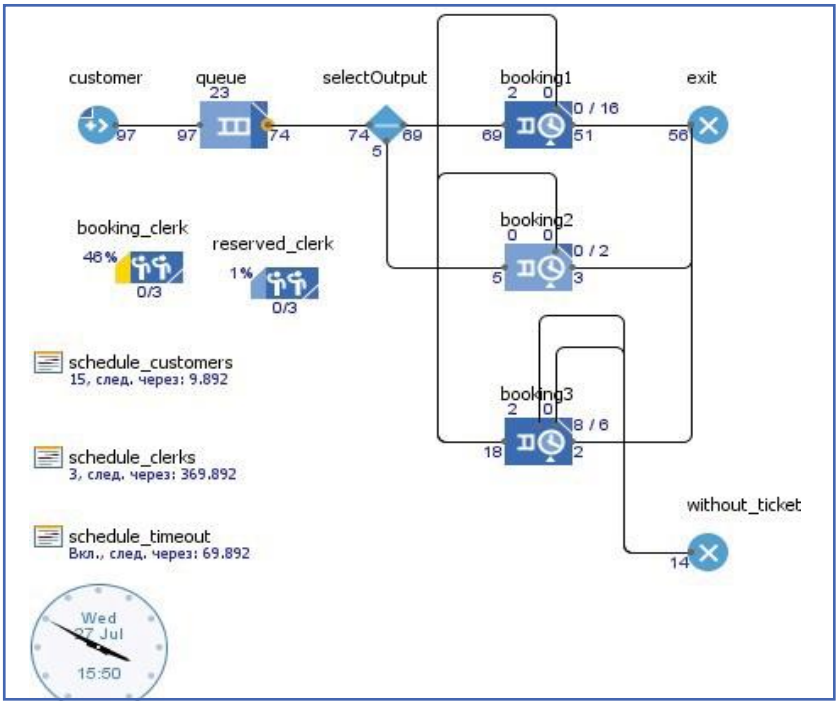


Рис. 25. Работа модели с учетом ухода покупателей

Добавление динамических графиков в модель

Создание набора данных по проданным билетам

Будем полагать, что каждый покупатель покупал 1 билет. Конечно, это немного упрощенный подход. Для сбора данных по количеству проданных билетов во время работы модели используется объект Набор данных (рис. 26). Этот объект представляет собой двумерный массив, хранящий значения X и Y. Этот объект находится в библиотеке **Статистика** на палитре инструментов. Перетащите этот объект на рабочее поле. Перейдите в его свойства. Задайте имя набора – **sold_tickets .**

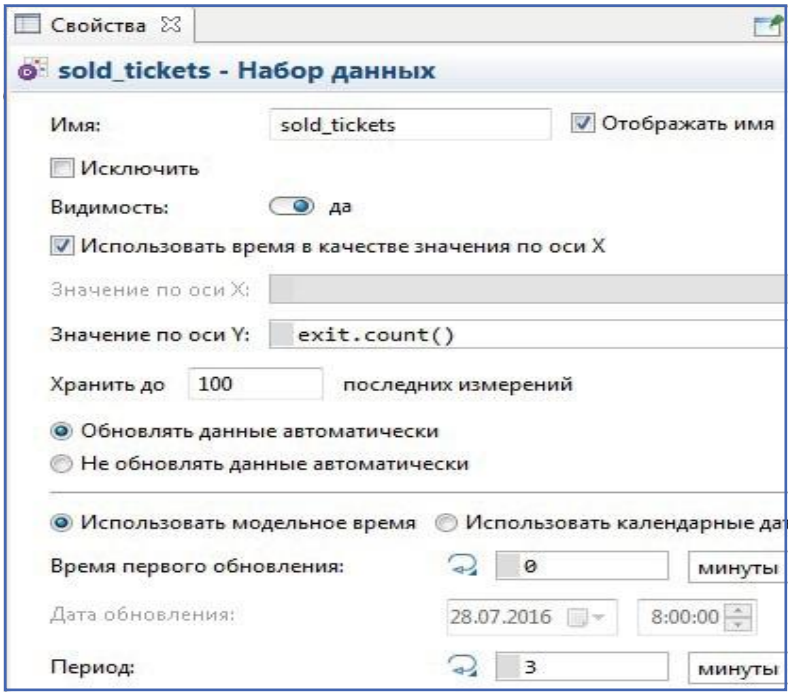


Рис. 26. Набор данных покупателей, купивших билет

Поскольку нас интересует динамика продаж билетов, то по оси X зададим модельное время. По оси Y будем сохранять количество покупателей, прошедших через выход **exit**.

Эту величину возвращает функция **size ()** объекта **Sink**. Данные в наборе будут обновляться каждые три минуты.

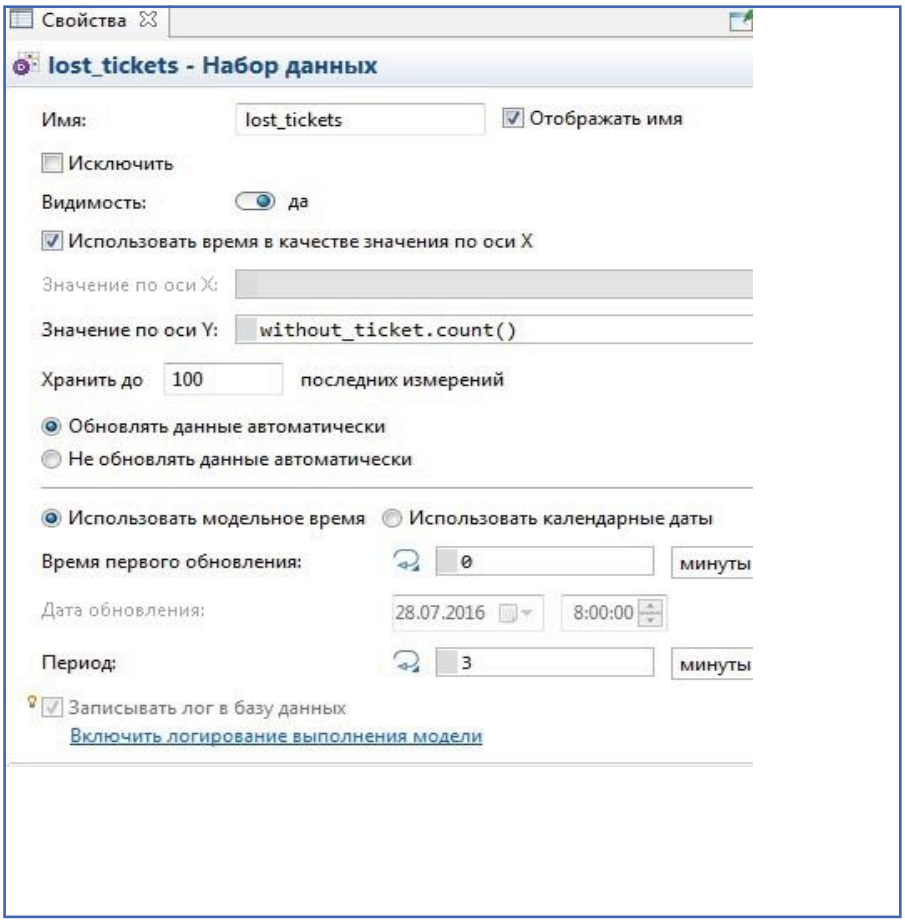


Рис. 27. Набор данных покупателей, не купивших билет

Создание круговой диаграммы по покупателям

Для того чтобы отслеживать в режиме работы модели соотношение покупателей, купивших и не купивших билет, используем инструмент **Круговая диаграмма**. Этот инструмент также находится в библиотеке **Статистика**. Перетащите его на рабочее поле. Перейдите в его свойства и в разделе Данные нажмите на «+», задайте название для данных, а в поле Значение вызовите метод **count()** объекта **exit**. Аналогично задайте еще один набор данных в диаграмме.

Создание временного графика, отражающего количество покупателей, купивших и не купивших билет.

Для отображения содержимого наборов данных, созданных в шаге 1 и шаге 2, используем инструмент Временной график (рис. 28). Этот инструмент позволяет отслеживать во время работы модели, динамически меняющиеся наборы данных или значения. Инструмент также находится на панели **Статистика**. Перетащите его на рабочее поле и задайте в его свойствах созданные ранее наборы данных.

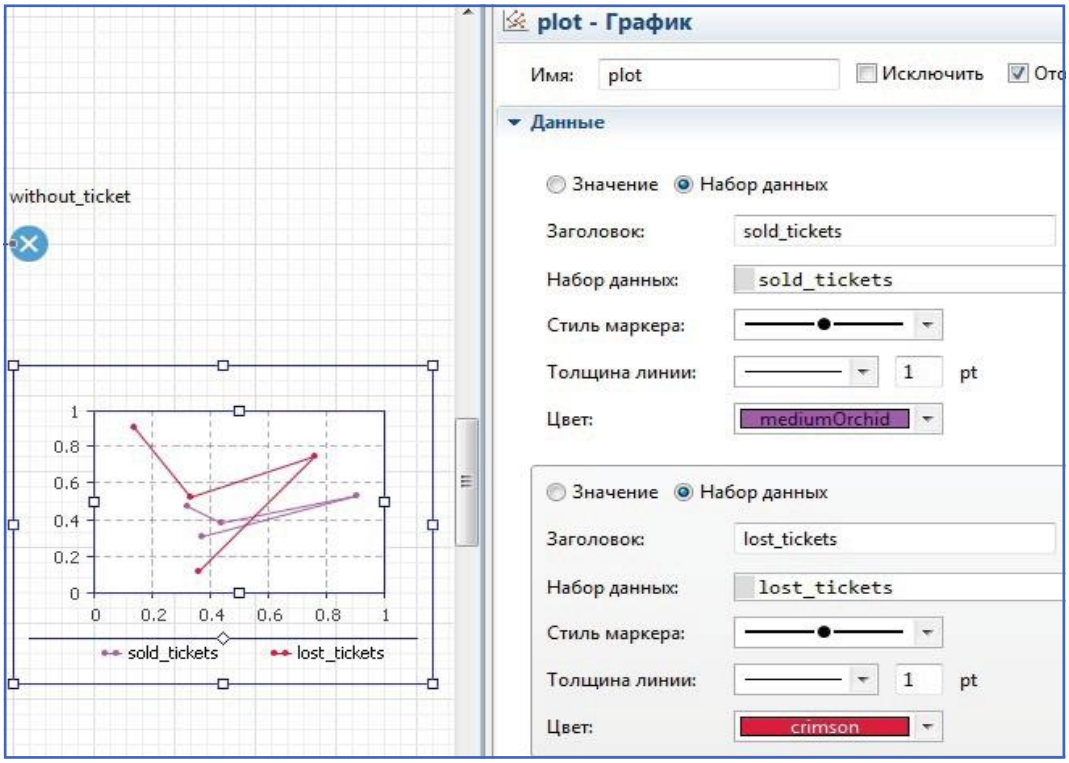


Рис. 28. Временной график

Проверка работоспособности модели

Запустите модель. Спустя некоторое время работы модели должно получиться как на рис. 29.

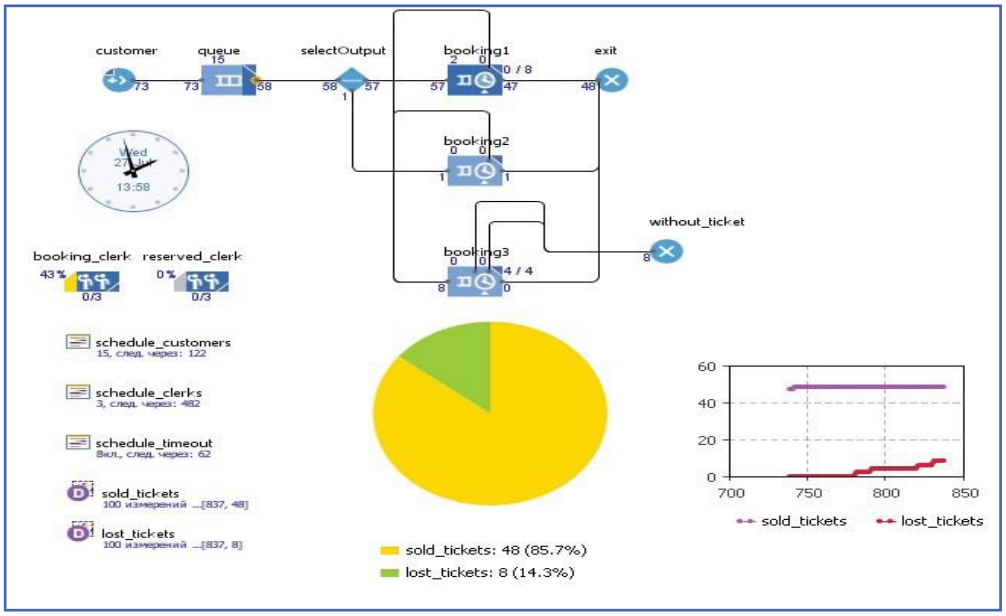


Рис. 29. Временной график

1.4. AnyLogic-модель сборки изделия

Построить модель работы технологической цепочки по сборке изделия, состоящего из двух деталей.

Первая деталь изделия подвергается двум технологическим операциям до сборки, вторая деталь изделия подвергается одной технологической операции до сборки. Первая технологическая операция над первой деталью длится **от 3 до 5 минут** и выполняется **1 роботом**.

Вторая технологическая операция с первой деталью длится **от 4 до 8 минут** и выполняется 1 рабочим, который работает согласно расписанию **(с 8 до 17 по рабочим дням с перерывом на обед с 12 до 13)**.

Технологическая операция по обработке **второй детали длится от 6 до 10 минут и выполняется рабочим.**

Сборка изделия **выполняется роботом и длится от 6 до 12 минут.** Изделие после сборки упаковывается по 5 штук. **Упаковка изделий осуществляется рабочим и длится от 10 до 16 минут.** Первая деталь для сборки поставляется **со склада1 в количестве 1 штуки в час**. Вторая деталь для сборки поставляется **со склада2 в количестве 2 штуки в час.**

Для решения поставленной задачи будет использоваться **дискретно-событийный подход**. В этом подходе рассматривается заявка-агент, которую обслуживают на различных операциях. В качестве заявки агента в данной задаче рассматриваются детали и само изделие. Технологические операции, выполняемые над деталями, рассматриваются как обслуживание заявки-агента различными сервисами.

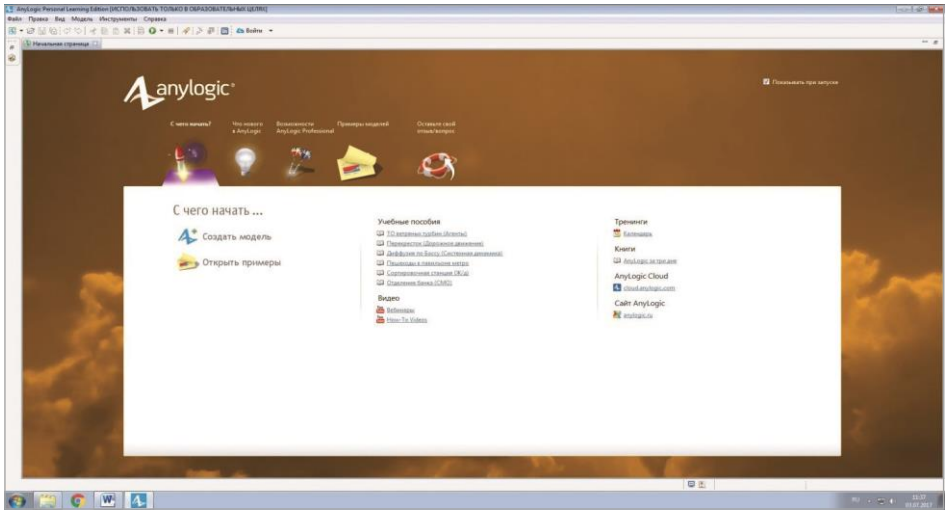


Рис. 1. Начальная страница AnyLogic

Это окно в дальнейшей работе не нужно, поэтому его можно закрыть. В рабочем окне программы выберите из строки меню **Файл** → **Создать** → **Модель** (рис. 2).

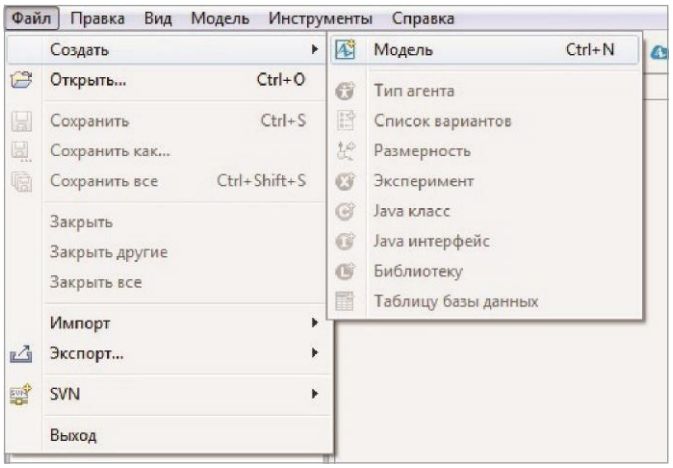


Рис. 2. Создание модели

В открывшемся окне мастера создания модели задайте имя модели и единицы модельного времени — **минуты** (рис. 3).

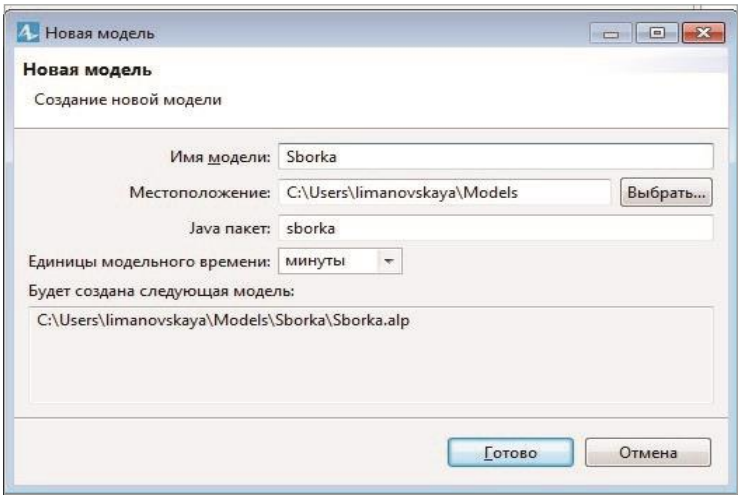


Рис. 3. Задание параметров модели

Откроется рабочее окно модели (рис. 4), которое разделено на три части. В левой части находятся закладки **Проекты** и **Палитра**.

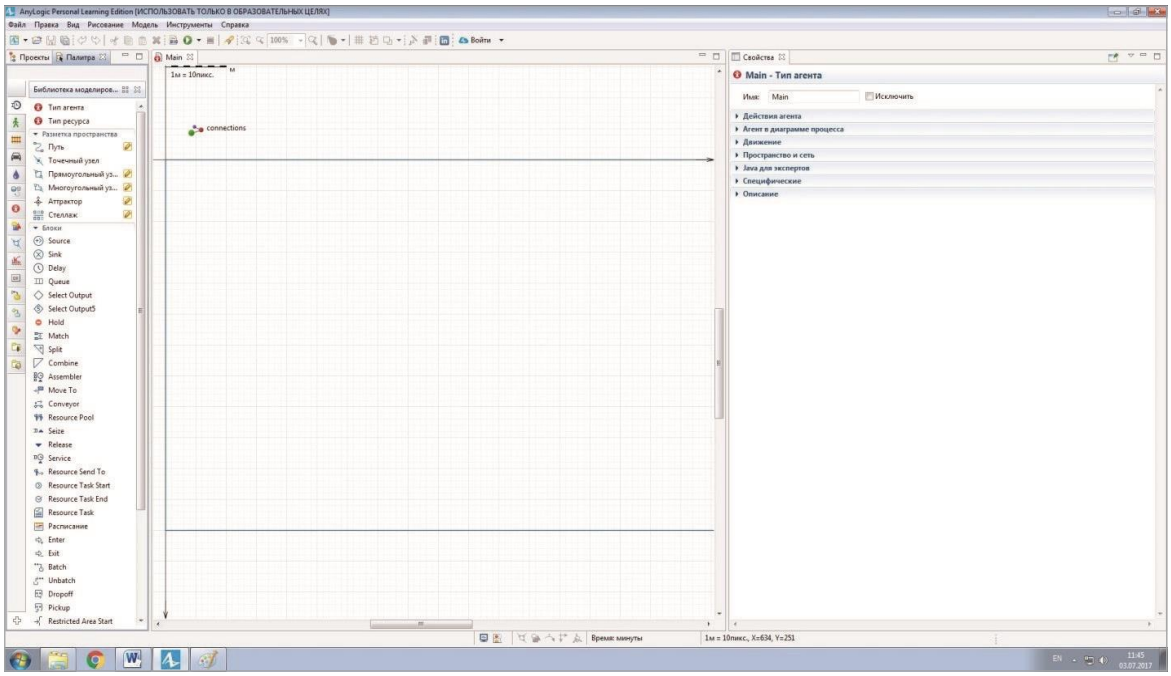


Рис. 4. Рабочее окно модели

В закладке **Проекты** представлены все открытые проекты и их содержимое. В закладке **Палитра** находятся все инструменты моделирования, которые разделены на разные библиотеки. Все необходимые инструменты для дискретно-событийного моделирования находятся в **Библиотеке моделирования процессов**, которая будет автоматически открываться. Средняя зона представляет собой рабочее поле, в котором будет собираться модель. В правой части отображаются свойства **выделенного в данный момент** элемента модели.

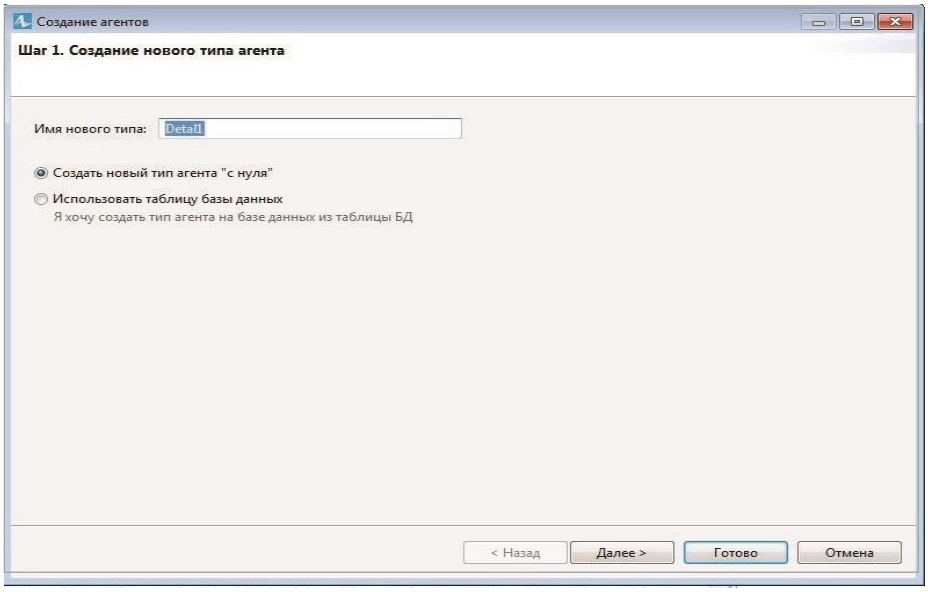


Рис. 5. Окно мастера создания агента

Моделирование агентов – деталей и изделия

Детали, из которых будет производиться сборка и само изделие, будут представлены в модели как **Тип агентов**. Перетащите ярлык **Тип агента** из библиотеки на рабочее поле. Откроется окно мастера создания агента (рис. 5).

Задайте в нем имя агента и нажмите **Готово**. После этого автоматически откроется окно агента. Закройте его. Повторите операцию для каждой детали и изделия. Имена агентов – для детали1 – **Detal1**, для детали2 – **Detal2**, для изделия – **Isdelie**. После всех операций на вкладке **Проекты** должен быть список из 4 агентов: **Detal1**, **Detal2**, **Isdelie**, **Main** (рис. 6).

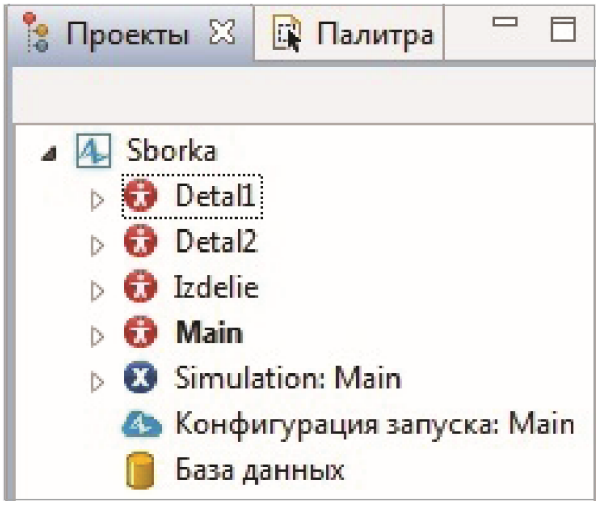


Рис. 6. Список агентов модели

Моделирование поставок деталей

Для того чтобы моделировать появление заявок-агентов в модели, необходимо использовать блок **Source** (рис. 7).

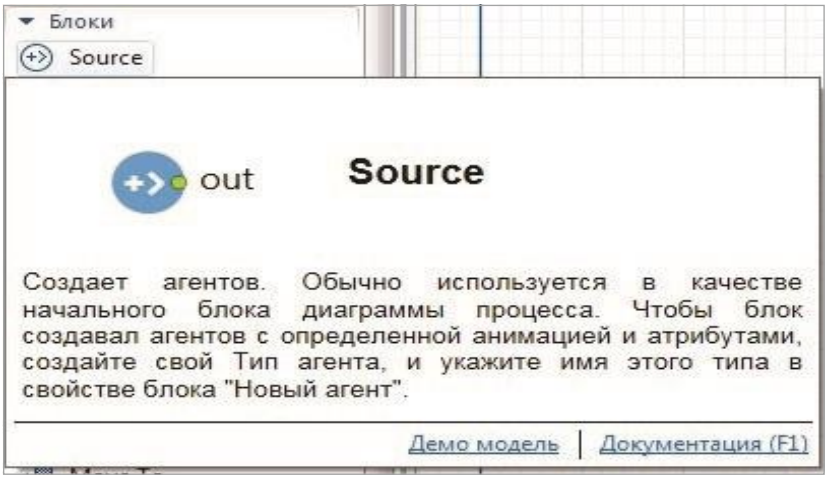


Рис. 7. Блок **Source**

Для моделирования поставок **деталей1** перетащите блок **Source** на рабочее поле модели. Задайте в свойствах блока **имя склада**, интенсивность поставки 1 в час и в разделе **Новый агент** — тип агента **Detal1** (рис. 8).

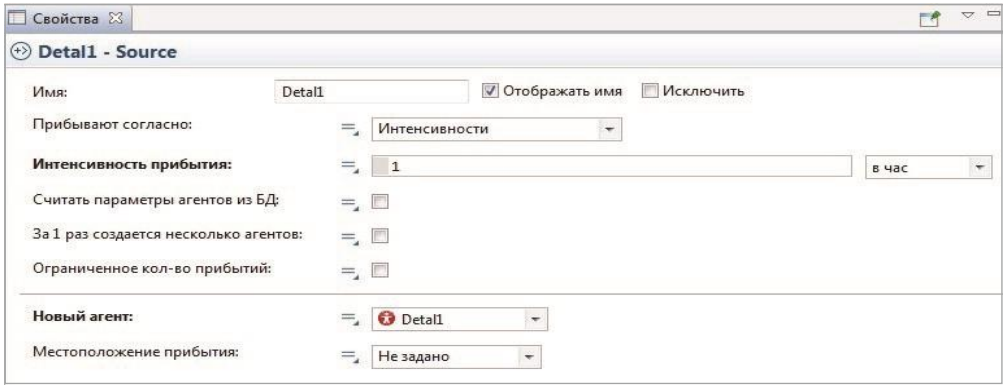


Рис. 8. Задание свойств склада **деталей1**

Повторите те же операции для **деталей2**, задав интенсивность прибытия 2 в час и выбрав в разделе **Новый агент** тип агента **Detal2** (рис. 9).

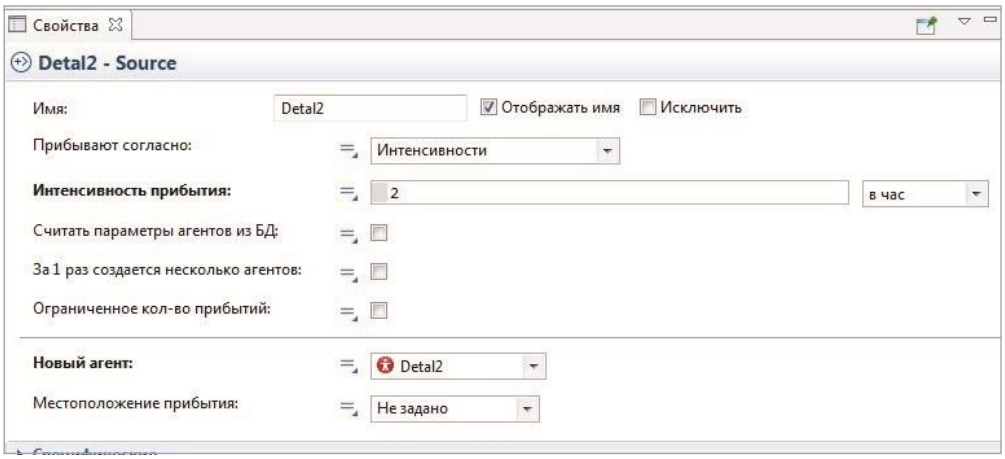


Рис. 9. Задание свойств склада **деталей2**

Моделирование обработки деталей

Первая деталь подвергается двум технологическим операциям. Каждая операция рассматривается как обслуживание агента **Detal1**. Для обслуживания агента используется блок **Service** (рис. 10).

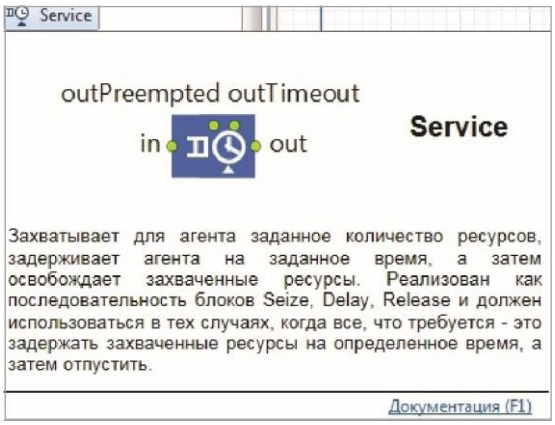


Рис. 10. Блок **Service**

Данный блок состоит из очереди на обслуживание и блока, имитирующего задержку детали на время обслуживания. Блок имеет один вход и три выхода: **out**, **outTimeout**, **outPreempted**. Из выхода **out** выходят те детали, чье обслуживание было завершено. Из выхода **outTimeout** выходят те детали, чье время ожидания обслуживания в очереди истекло. Из выхода **outPreempted** выходят детали, вытесненные деталями с более высоким приоритетом на обработку.

В данной задаче все детали должны пройти обслуживание, поэтому будет использоваться только выход **out**.

Задание первой технологической операции

Перетащите блок **Service** на рабочее поле модели так, чтобы его вход соединился с выходом блока **Detal1** (рис. 11).

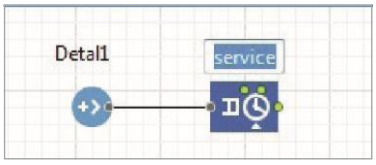


Рис. 11. Перенос блока **Service** на рабочее поле модели

В свойствах блока задайте его имя (**Operaciya1**), вместимость очереди – **Максимальная вместимость**, длительность операции (рис. 12). Длительность операции задается треугольным распределением, в котором первый параметр означает минимальное время операции (3 минуты), второй параметр – среднюю продолжительность операции (4 минуты), третий — максимальную продолжительность операции (5 минут).

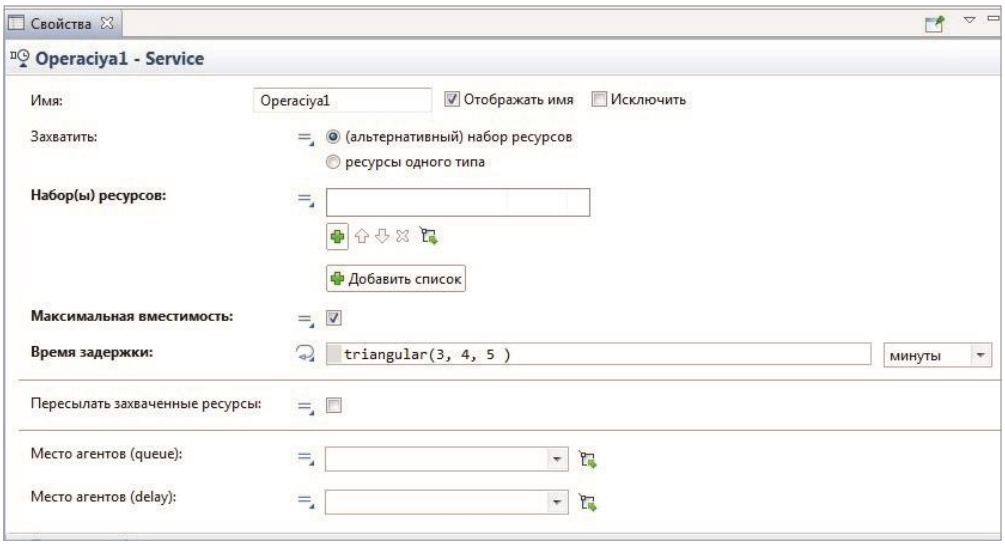


Рис. 12. Задание свойств **операции1**

Задание ресурсов для выполнения первой операции

Как было сказано в задании, первую технологическую операцию с **деталью1** выполняет робот. Для моделирования задания операции для робота нужно в блоке **Service** указать **Набор ресурсов**, который выполняет операцию. Чтобы указать **Набор ресурсов**, нужно заранее создать ресурсы в модели. Ресурсы в модели задаются блоком **Resource Pool** (рис. 13).

Перетащите блок **Resource Pool** на рабочее поле модели. В его свойствах (рис. 14) задайте имя ресурса **Robot1**, количество и тип — **Переносной**. Вообще, ресурсы в модели могут быть трех типов: движущийся, переносной и статический. Если выбрать движущийся тип, то робот может обслуживать несколько операций, передвигаясь от одной к другой.

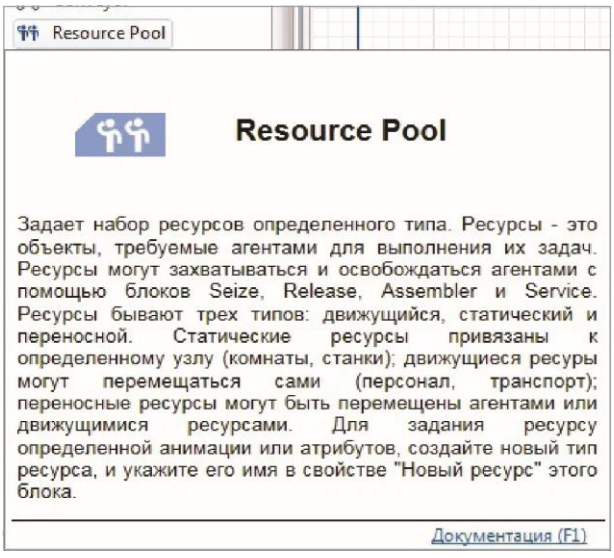


Рис. 13. Блок **Resource Pool**

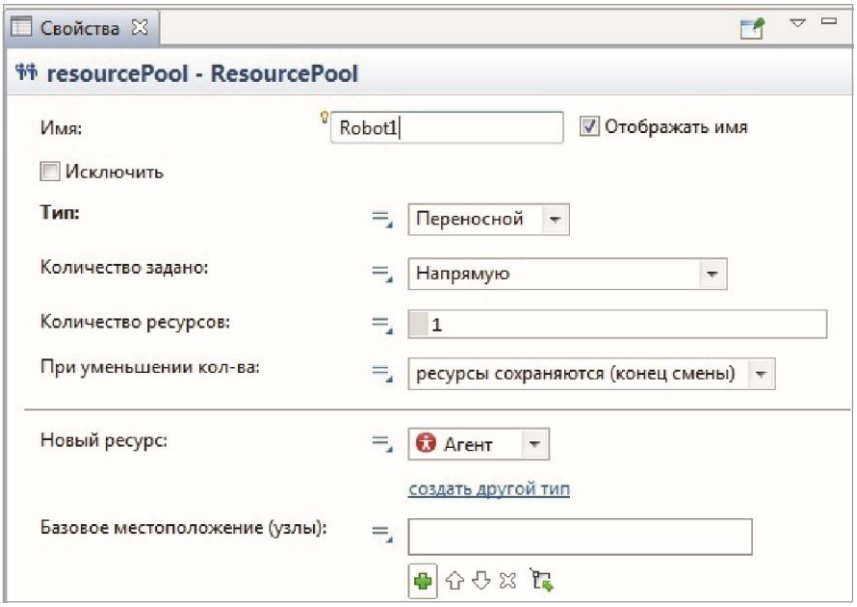


Рис. 14. Свойства ресурса **робот1**

Теперь можно добавить ресурс **Robot1** к первой технологической операции. Для этого перейдите в свойства блока **Operaciya1** и в разделе **Набор ресурсов** нажмите на **+** и добавьте ресурс **Robot1** (рис. 15).

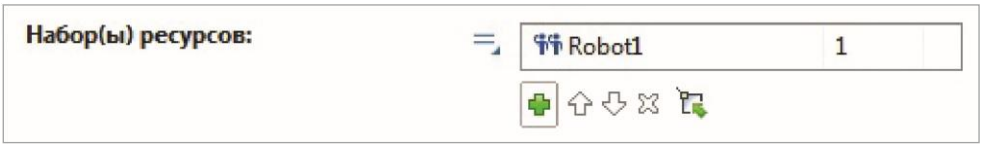


Рис. 15. Добавление ресурса **Robot1**

В результате в свойствах блока **Operaciya1** получится (рис. 16).

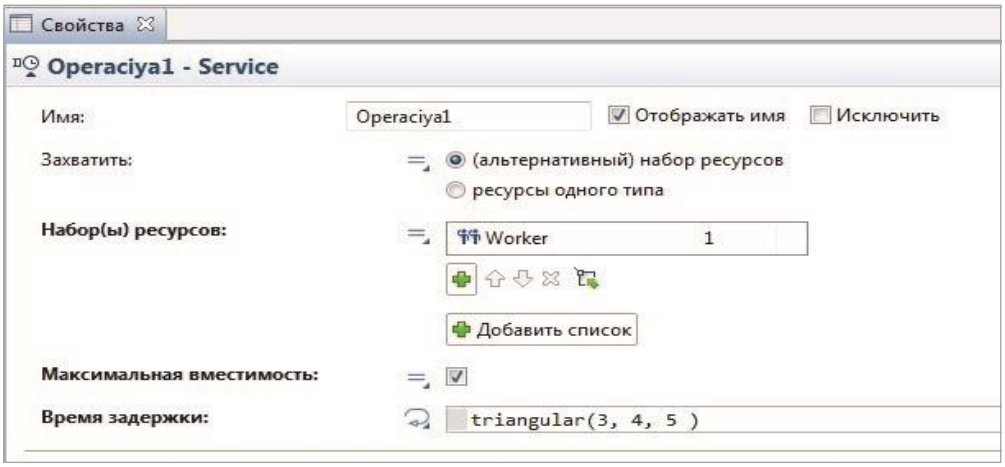


Рис. 16. Свойства **Operaciya1**

Задание второй технологической операции

Перетащите блок **Service** так, чтобы выход **out** блока **Operaciya1** соединился со входом нового блока (рис. 17).

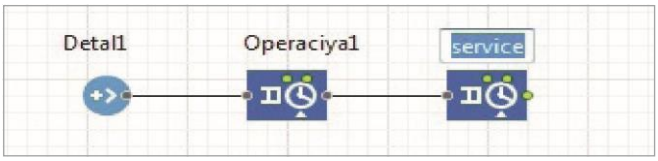


Рис. 17. Задание второй операции

В свойствах блока задайте его имя **Operaciya2**, вместимость очереди 10 и время выполнения — 4, 6 и 8 минут (рис. 18).

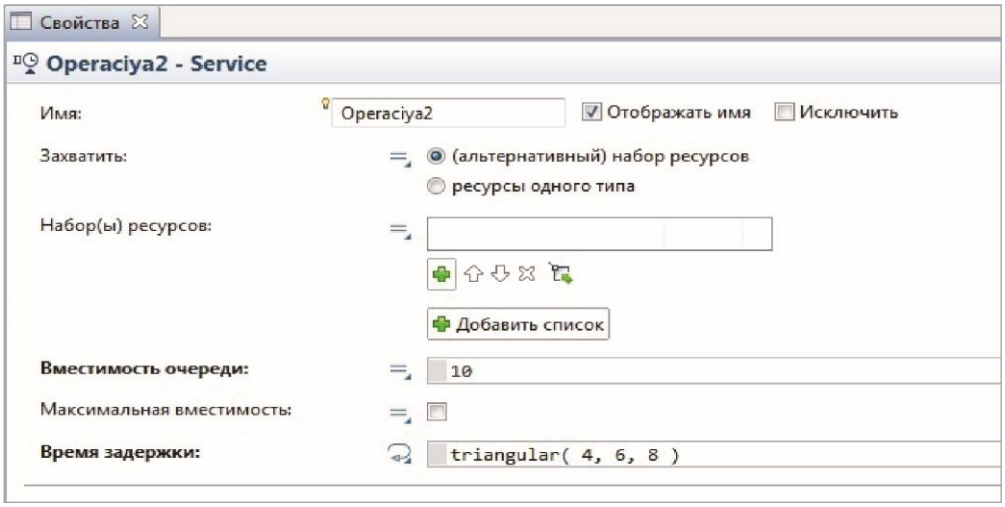


Рис. 18. Задание свойств второй операции

Задание ресурсов для выполнения второй технологической операции

Вторую технологическую операцию выполняет рабочий, который работает по заданному расписанию. Для задания расписания работы рабочего используется блок **Расписание** (рис. 19).

Рис. 19. Блок **Расписание**

Перетащите блок **Расписание** на рабочее поле модели. В свойствах блока **Расписание** задайте два расписания — с 8 до 12 и с 13 до 17. Добавить расписание необходимо с помощью кнопки **+** справа от таблицы **Расписание**. Тип расписания задайте **да/нет** (показывает информацию о том, занят или свободен ресурс, работающий по данному расписанию, рис. 20).

Далее задайте ресурс с помощью блока **Resource Pool**. Поскольку работа рабочего задается расписанием, то в свойствах блока **Resource Pool** в пункте **Количество задано** выбираем пункт **Расписание доступности**. Если выбрать просто пункт **Расписание**, то нужно будет в самом блоке **Расписание** задавать количество рабочих, занятых в указанные часы. Поскольку в блоке **Расписание** выбран тип расписания **да/нет**, т.е. доступен или нет ресурс, то этот тип расписания называется **Расписанием доступности**. В появившемся пункте **Расписание доступности** выберите только что созданное расписание. В итоге свойства блока **ResourcePool** должны выглядеть как на рис. 21.

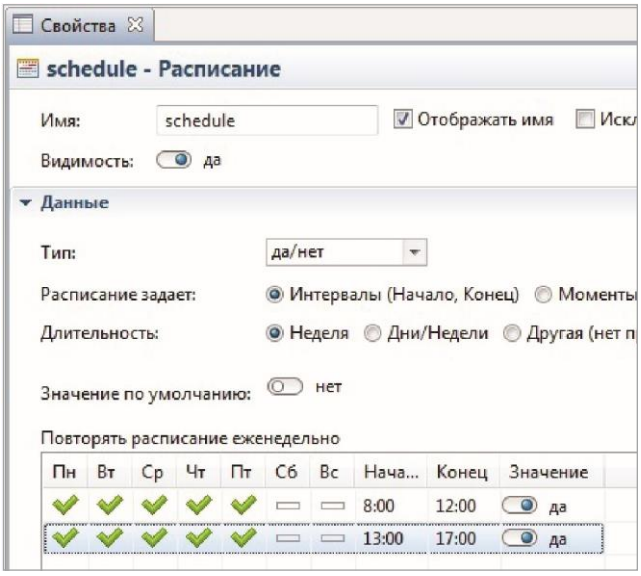


Рис. 20. Задание расписания работы рабочего

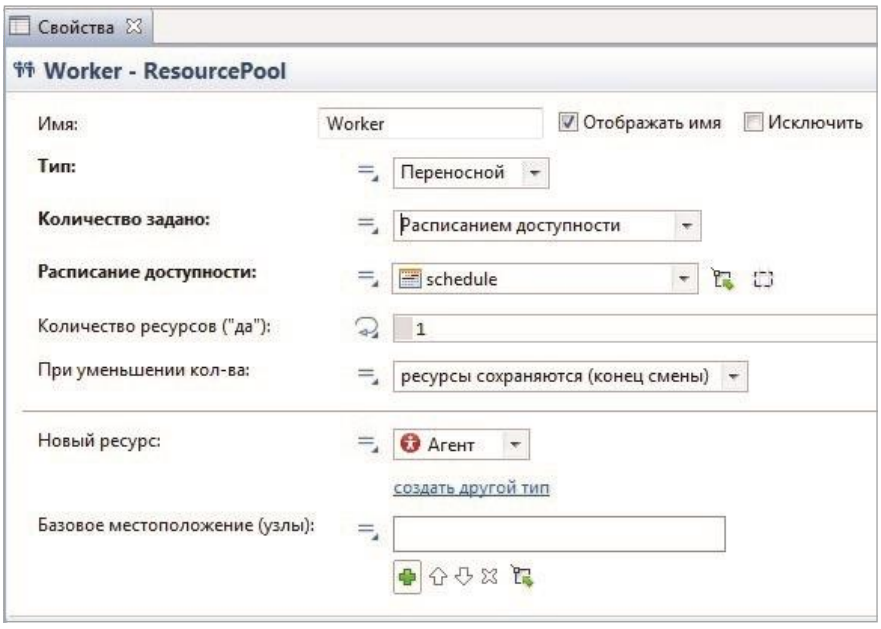


Рис. 21. Свойства ресурса **Рабочий**

Добавьте ресурс ко второй операции (рис. 22).

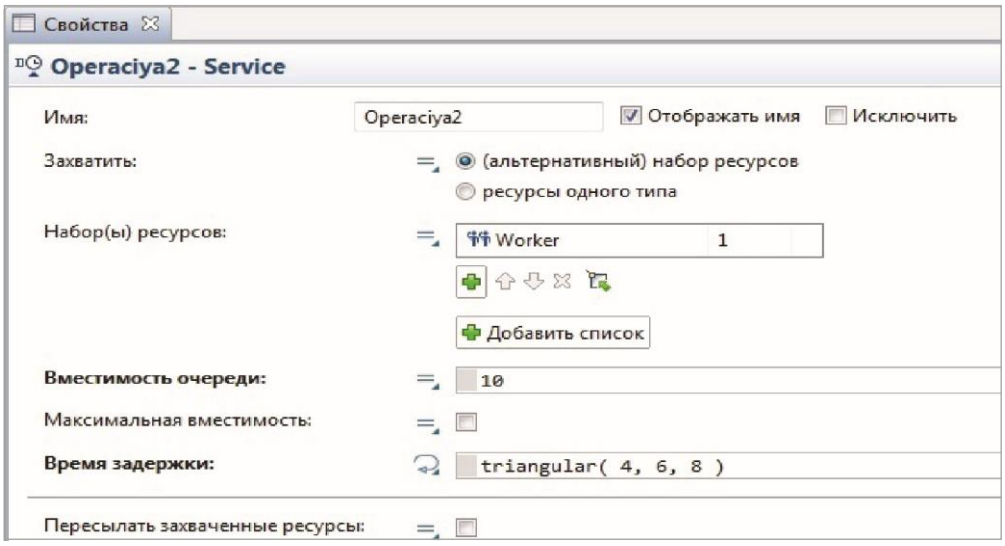


Рис. 22. Свойства второй операции с добавленным ресурсом

Задание операции обработки второй детали

Промоделируем выполнение операции с помощью блока **Service**. Перетащите блок **Service** на рабочее поле модели так, чтобы его вход соединился с выходом блока **Detal2** (рис. 23).

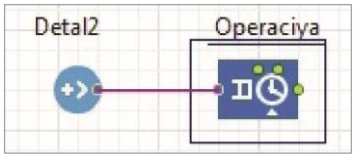


Рис. 23. Присоединение блока **Operaciya**

В свойствах блока задайте его имя (**Operaciya**), вместимость очереди (**Максимальная**) и время выполнения операции (рис. 24).

Технологическая операция выполняется рабочим. Причем это другой рабочий, не тот, который обрабатывает первую деталь. Поэтому нужно создать новый ресурс. А вот работать этот рабочий будет по тому же расписанию, что и рабочий, обрабатывающий первую деталь, поэтому создавать новое расписание не нужно. Итак, создайте новый ресурс, работающий по имеющемуся расписанию. Для этого перетащите блок **Resource Pool** на рабочее поле модели и задайте его свойства (рис. 25).

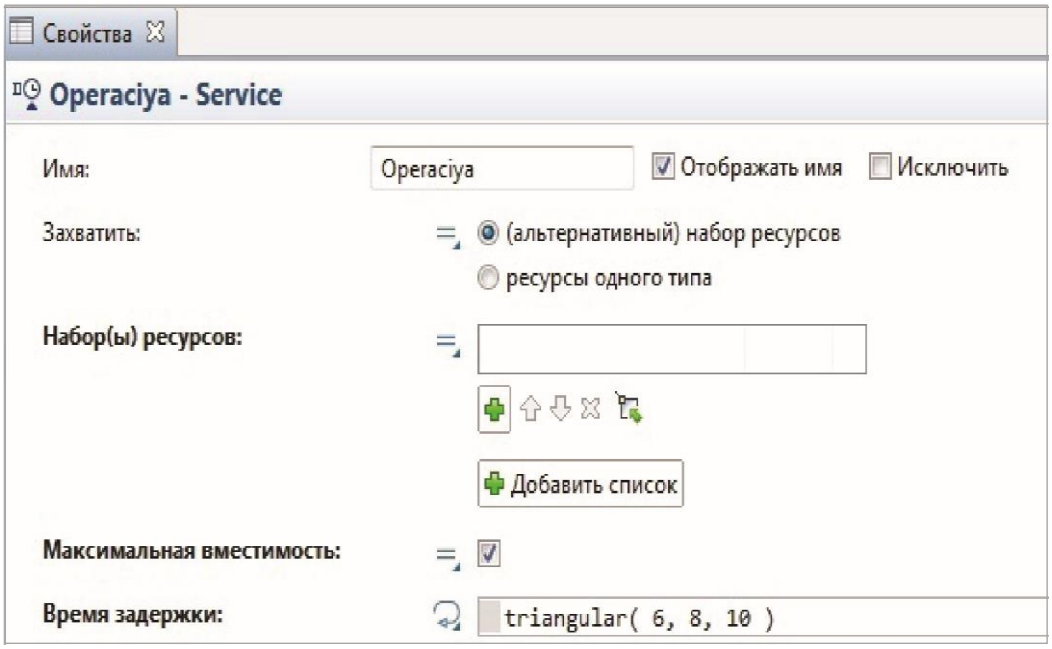


Рис. 24. Свойства блока **Operaciya**

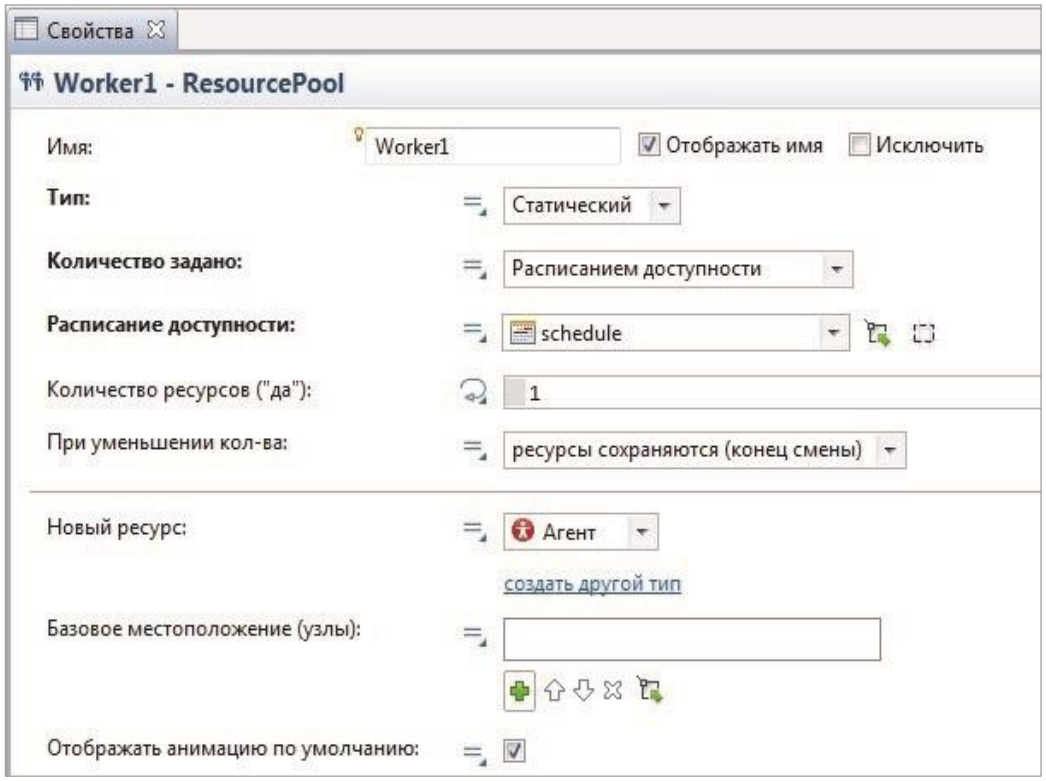


Рис. 25. Свойства блока **Worker1**

Теперь добавьте ресурсы к операции обработки детали, зайдя в свойства блока **Opraciya** и добавив **Набор ресурсов** (рис. 26).

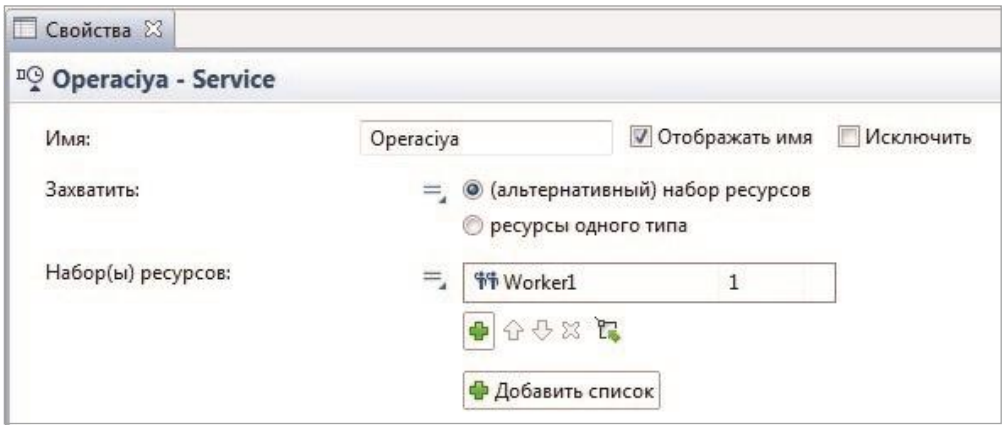


Рис. 26. Свойства блока **Opraciya** с добавленными ресурсами

Задание операции сборки

Для моделирования процесса сборки в среде **AnyLogic** используется блок **Assembler**, который имеет 5 входов и один выход (рис. 27). На вход в него подаются детали, из которых собирается изделие, на выходе из блока получается новый агент-заявка, а именно собранное изделие.

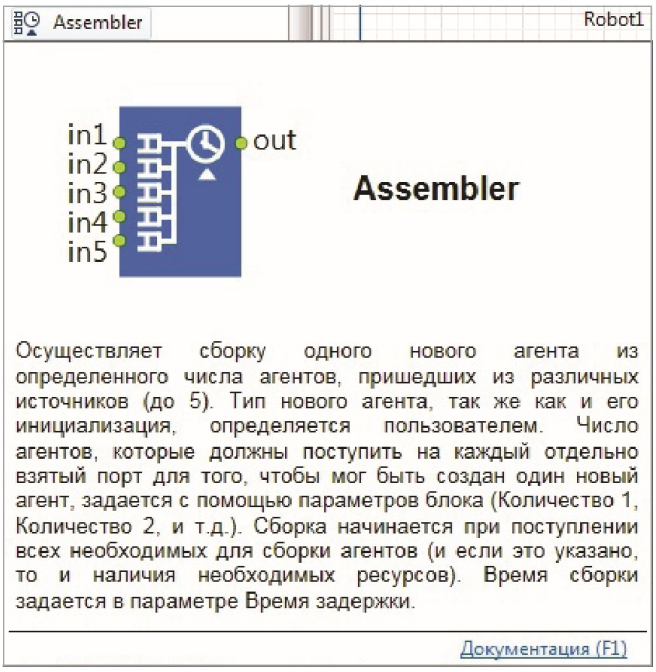


Рис. 27. Блок **Assembler**

Перетащите блок **Assembler** на рабочее поле модели и соедините его входы с выходами операций обработки деталей.

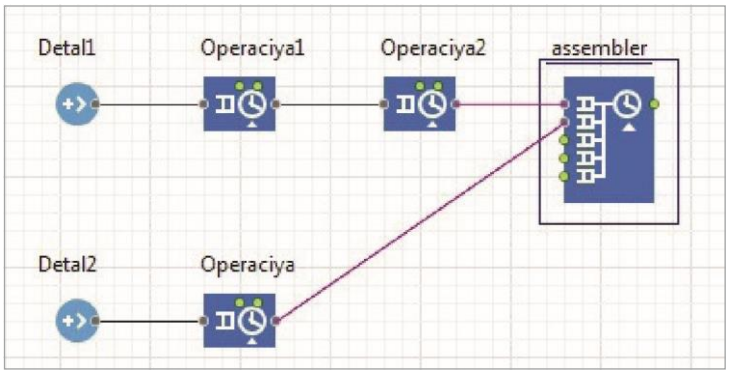


Рис. 28. Присоединение операции сборки

В свойствах блока **Assembler** задайте время выполнения сборки и тип агента на выходе в разделе **Новый агент** (рис. 29).

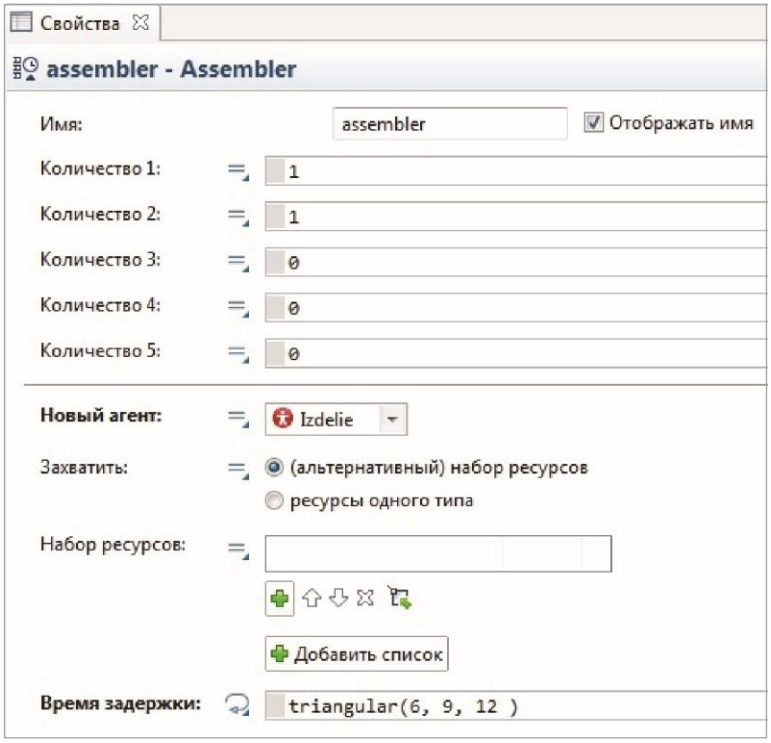


Рис. 29. Свойства блока **Assembler**

Сборку осуществляет робот, предназначенный именно для этой операции. Поэтому нужно с помощью блока **Resource Pool** создать еще один тип ресурсов **RobotAssembler**. Для этого перетащите блок **Resource Pool** на рабочее поле модели и задайте его свойства (рис. 30).

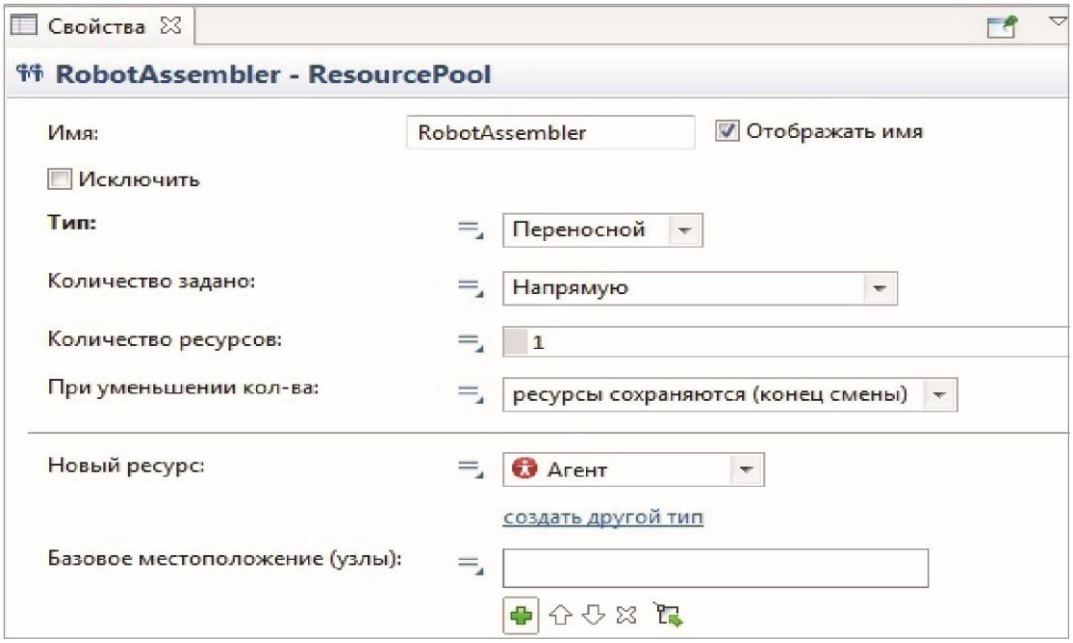


Рис. 30. Свойства блока **RobotAssembler**

Присоедините ресурс робот к процессу сборки (рис. 31).

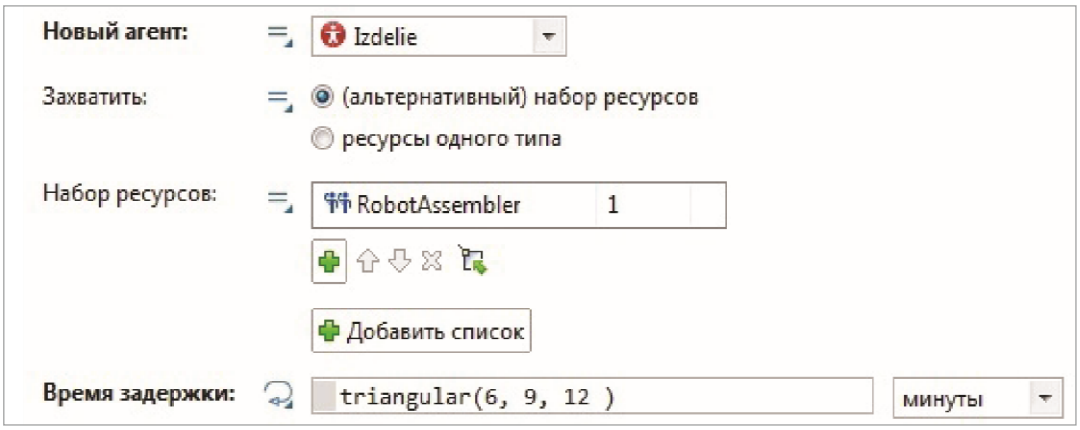


Рис. 31. Назначение ресурса для процесса сборки

Задание операции упаковки

Операция сборки также задается блоком **Service**, а рабочий, выполняющий ее, задается блоком **Resource Pool**. Поскольку этот рабочий работает по тому же расписанию, что и все остальные рабочие, то используется ранее созданное расписание доступности.

Перетащите блок **Resource Pool** на рабочее поле модели и задайте его свойства (рис. 32).

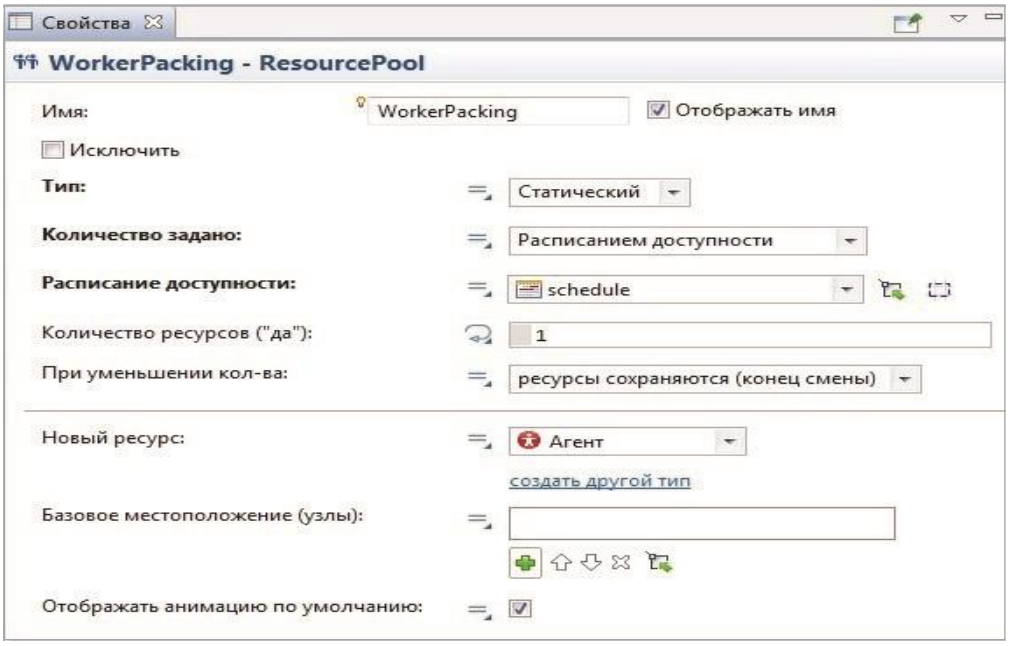


Рис. 32. Свойства блока **WorkerPacking**

Перетащите блок **Service** на рабочее поле модели так, чтобы его вход соединился с выходом блока **Assembler**, и задайте его свойства (рис. 33).

Из операции упаковки выходят коробки, содержащие по 5 изделий. Для моделирования коробок введем нового агента в модель — **Box**.

Перетащите на рабочее поле модели блок **Тип агента** и в открывшемся мастере создания агента введите его имя **Box**. Нажмите кнопку **Готово**. Закройте автоматически открывшееся окно агента **Box**.

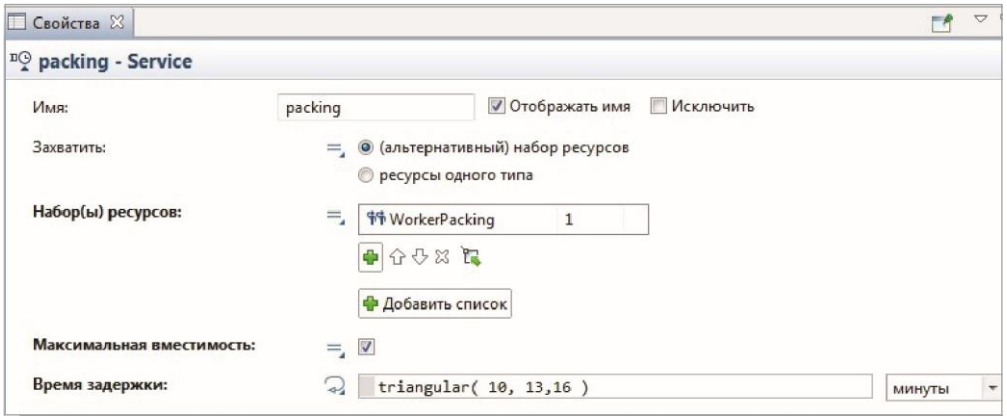


Рис. 33. Свойства блока **packing**

Упаковку изделий промоделируем с помощью блока **Batch** (рис. 34). Этот блок принимает на вход заданное количество изделий и выпускает их партию.

Если задать постоянную партию, то ее уже нельзя будет разобрать на отдельные изделия.

Рис. 34. Блок **Batch**

Перетащите блок **Batch** на рабочее поле модели так, чтобы его вход соединился с выходом блока **packing** (рис. 35). Задайте свойства блока (рис. 36).

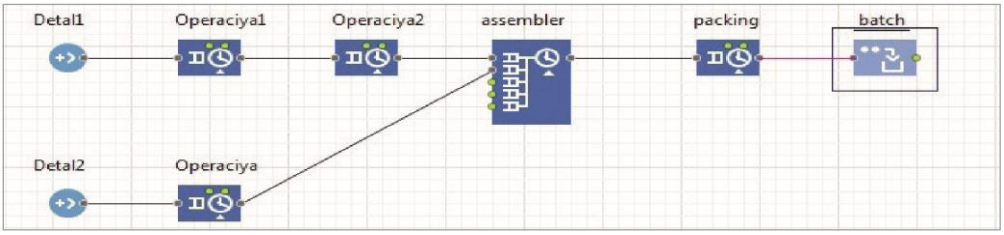


Рис. 35. Присоединение блока **Batch**

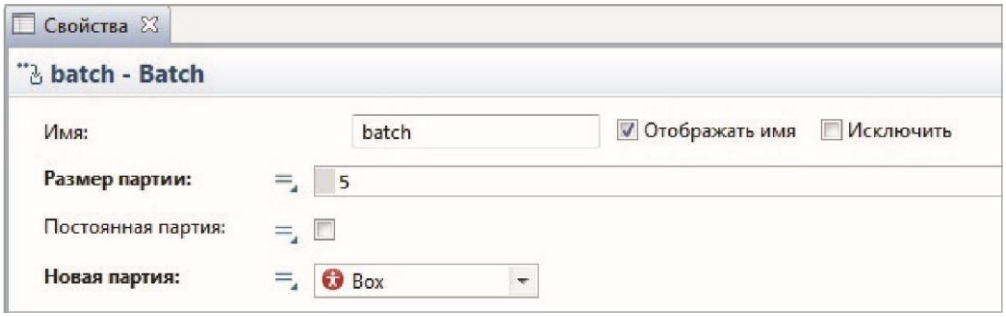


Рис. 36. Свойства блока **Batch**

Собранные коробки с изделиями увозятся из цеха. Для моделирования ухода коробок из модели используется блок **Sink** (рис. 37). Он имеет один вход и просто уничтожает входящие в него заявки.

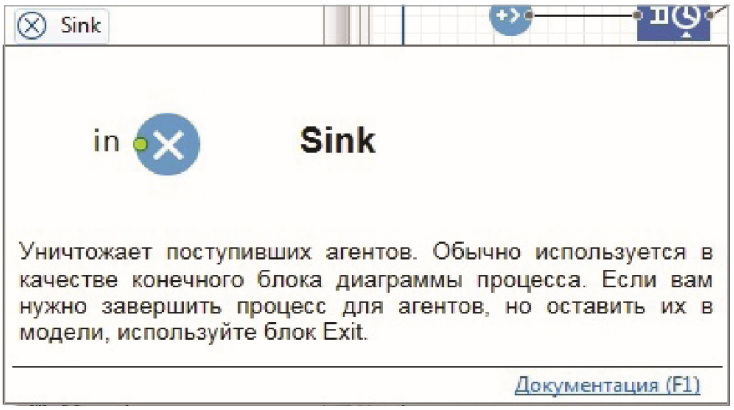


Рис. 37. Блок **Sink**

Перетащите блок **Sink** на рабочее поле модели так, чтобы его вход соединился с выходом блока **batch** (рис. 38).

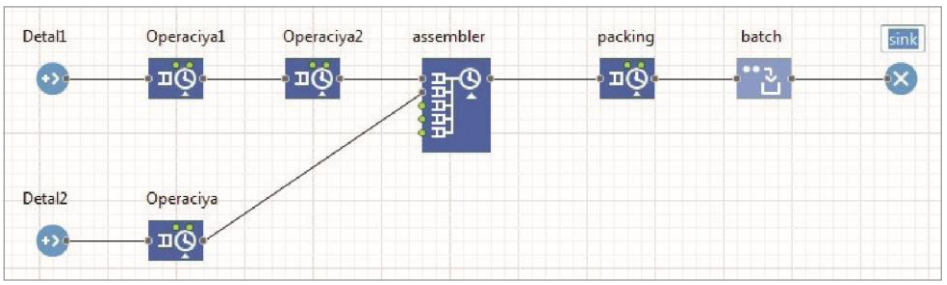


Рис. 38. Модель производства

Запустите полученную модель, нажав на кнопку  на панели инструментов в главном меню. Откроется окно запуска модели (рис. 39).

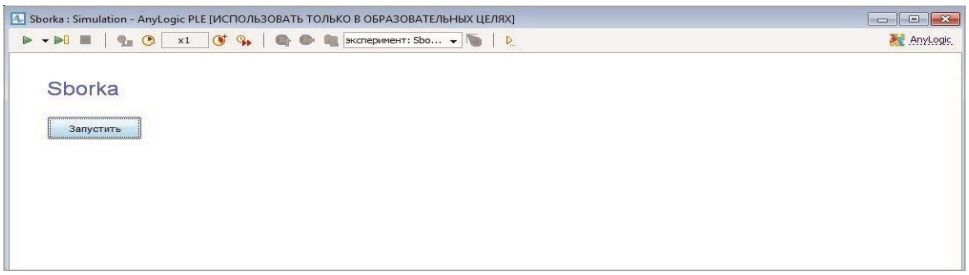


Рис. 39. Окно запуска модели

Нажмите на кнопку **Запустить** — откроется окно работающей модели (рис. 40).

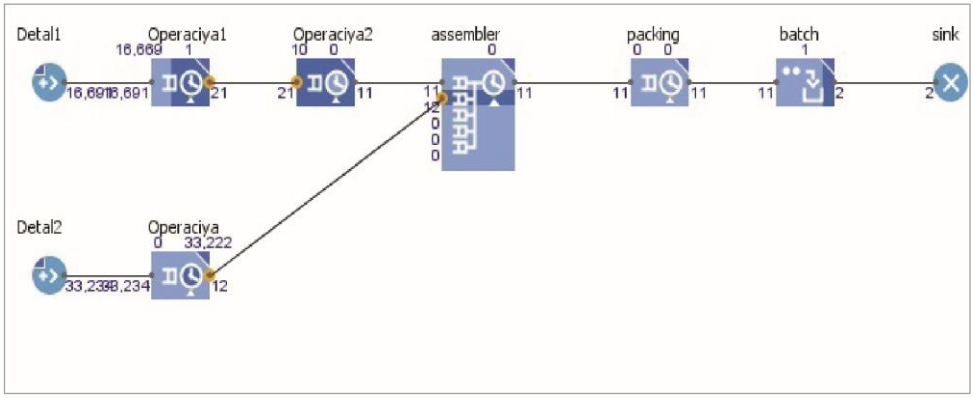


Рис. 40. Работа модели

Проблемные места в модели выделяются красным. Как видно из рис. 40, проблемы возникают на выходе из первой технологической операции и на входе во вторую технологическую операцию. Также есть проблемы на выходе из операции обработки второй детали. Поварьируйте настройки модели, включая время выполнения сборки, чтобы получить наибольшую занятость робота-сборщика и ликвидировать затор из вторых деталей на входе в операцию сборки.

1.5. AnyLogic-модель поликлиники

Каждый из нас хотя бы однажды проходил медицинский осмотр в поликлинике. Обычно посетитель медицинского осмотра приходит в поликлинику и, первым делом, встает в очередь в регистратуру. Дойдя, наконец, до регистратора, посетитель получает на руки медицинскую карту и список направлений на осмотр. Задача пациента: пройти всех специалистов по очереди.

Желание посетителя – выполнить эту задачу как можно скорее. Естественно, посетитель столкнется с очередями к каждому специалисту и, скорее всего, займет очередь к тому специалисту, очередь к которому минимальна. Каждый специалист принимает пациента определенное время. На медицинский осмотр запланировано не более 6 часов.

Требуется установить, сколько посетителей успеет пройти осмотр за заданное время. Где будут копиться очереди? Какое максимальное количество посетителей сможет пройти медицинский осмотр.

В модели, с помощью которой мы можем оценить эти параметры, нам не важна траектория движения людей, их взаимодействие между собой и т. д. Нам важно только время на каждом этапе обслуживания. При этом обслуживание для нас дискретно: нельзя же одновременно осматривать полтора человека.

Для моделирования будем использовать дискретно-событийную модель, в которой основной единицей является заявка на обслуживание, обрабатываемая некоторое время на каждом этапе.

• Создайте новую модель в AnyLogic: **Файл – Создать – Модель**. Название модели – **Policlinic**. Модель должна быть сохранена в папке **C:\Work** (рис. 1).

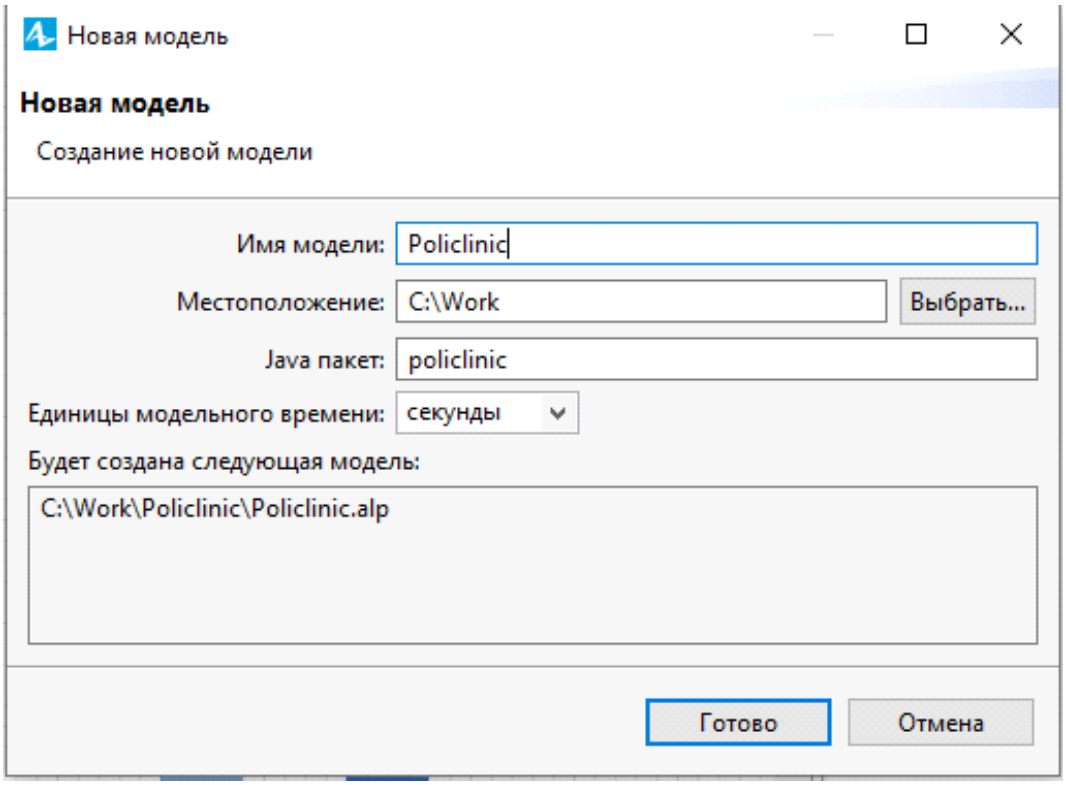


Рис. 1. Создание модели **Policlinic**

• Воспользуемся палитрой **Библиотека моделирования процессов**. Перейдите на вкладку **Палитра**. Разместите с палитры Библиотека моделирования процессов на диаграмме процессов **Main** следующие объекты:

Source – источник заявок агентов;

Queue – очередь агентов;

Delay – задержка агентов;

Exit – удаление объектов из диаграммы процесса.

Созданная диаграмма показана на рис. 2

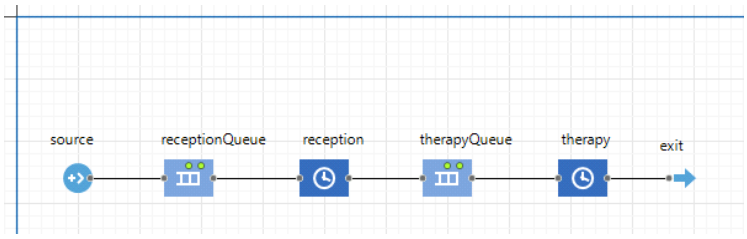


Рис. 2. Создание диаграммы процессов в поликлинике

Переименуйте объекты **Queue** в свойствах в **receptionQueue** и **therapyQueue**. Это очереди в регистратуру и к терапевту соответственно. Блоки **Delay** переименуйте в **reception** и **therapy** – это будет временная задержка в обслуживании у соответствующих специалистов.

- После того, как посетители пройдут терапевта, они смогут пройти оставшихся специалистов по маршрутному листу (направлению). Посещение стоматолога, окулиста и сдача анализа крови может осуществляться в любом порядке. Будем считать, что посетители будут выбирать того специалиста, очередь к которому минимальна. После обхода всех специалистов посетители будут покидать медицинское учреждение.

На диаграмме процессов это будет выглядеть в виде следующих блоков – рис. 3. Блоки можно копировать. Как видно, каждая очередь состоит из объектов **Enter**, **Queue**, **Delay** и **Exit**. В конце для уничтожения заявок поставьте блоки **Enter** и **Sink**.

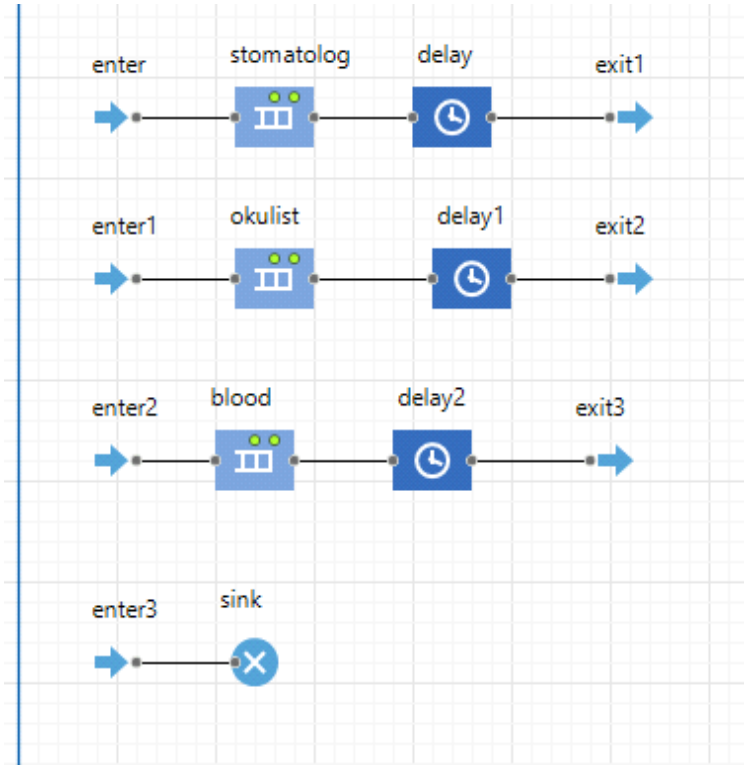


Рис. 3. Создание диаграммы процесса обхода оставшихся специалистов

• Для модели нужно сделать объект для сбора статистики и гистограмму для отображения данных. Статистика будет собираться по параметру **Время осмотра**, т. е. сколько в среднем посетитель обслуживался. Для вычисления среднего времени осмотра необходимо знать количество посетителей, обслуженных за определенное время. Это можно узнать из чисел на объектах.

Обратим внимание еще вот на какой момент: очереди к специалистам (стоматолог, окулист и забор крови) не связаны с остальной очередью (рис. 2 и рис. 3). К специалистам заявки будет отправлять специальная функция, которую мы опишем чуть позже.

Учитывая эти моменты, создадим новый класс заявок, который назовем **Patient**. Для этого в главном меню выберите **Файл – Создать – Тип агента**. Пункт **Создать новый тип агента с «нуля»** должен быть выбран.

Задайте название новому классу – **Patient**.

В дереве проекта у вас появится новый узел **Patient** (рис. 4).

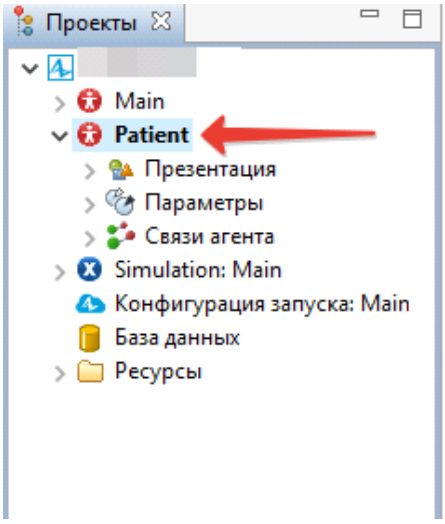


Рис. 4. Создание объекта **Patient**

Кликните по нему двойным щелчком мыши и в новом окне этого объекта создайте 4 параметра (палитра **Агент**, компонент **Параметр**):

• **ArrivalTime**() – тип **double**, значение по умолчанию **time()**; **blood** – тип **boolean**, значение по умолчанию **false**;

• **okulist** – тип **boolean**, значение по умолчанию **false**;

• **stomatolog** – тип **boolean**, значение по умолчанию **false**.

Примерное расположение параметров показано на рис. 5.

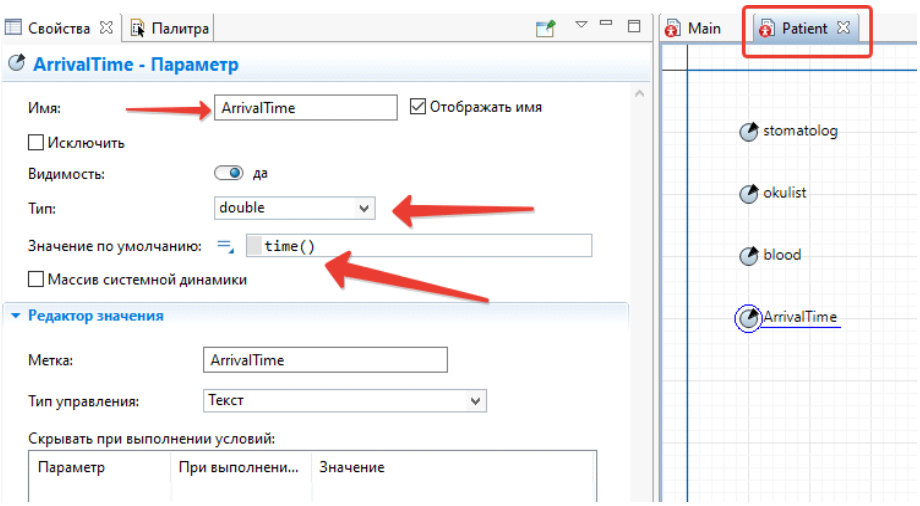


Рис. 5. Создание параметров для агента **Patient**

Параметры **blood**, **okulist**, **stomatolog** хранят информацию о том, посетил ли соответствующего специалиста пациент **(true)** или нет **(false)**.

Параметр **ArrivalTime** запоминает время прибытия посетителя в поликлинику. Его значение по умолчанию time() – это функция, возвращающая текущее модельное время, прошедшее с начала запуска модели.

- Класс **Patient** установите в качестве агента для каждого блока вашей диаграммы процесса. Для этого перейдите на диаграмму процесса **Main**. Выберите первый объект **source,** а в свойствах объекта в списке **Новый агент** выберите **Patient** (рис. 6). Аналогичную операцию сделайте для каждого блока всей диаграммы процесса.

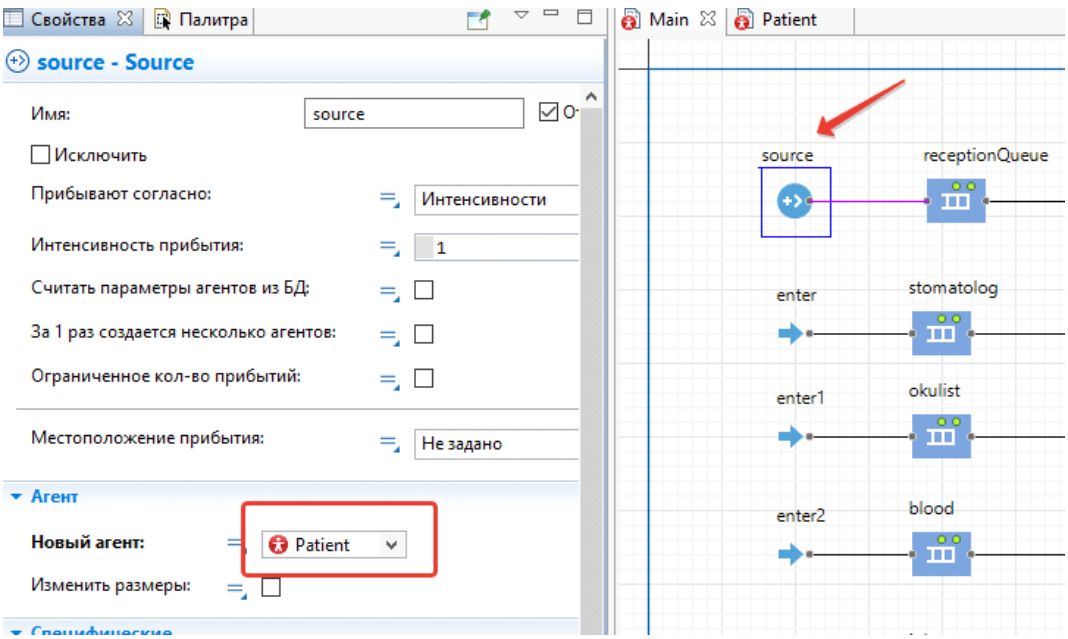


Рис. 6. Установка нового типа агента для блока

• Создадим специальную функцию, которая будет выполнять выбор, в какую очередь поставить заявку.

Перетащите из палитры **Библиотека моделирования процессов** объект **Функция** (рис. 7).

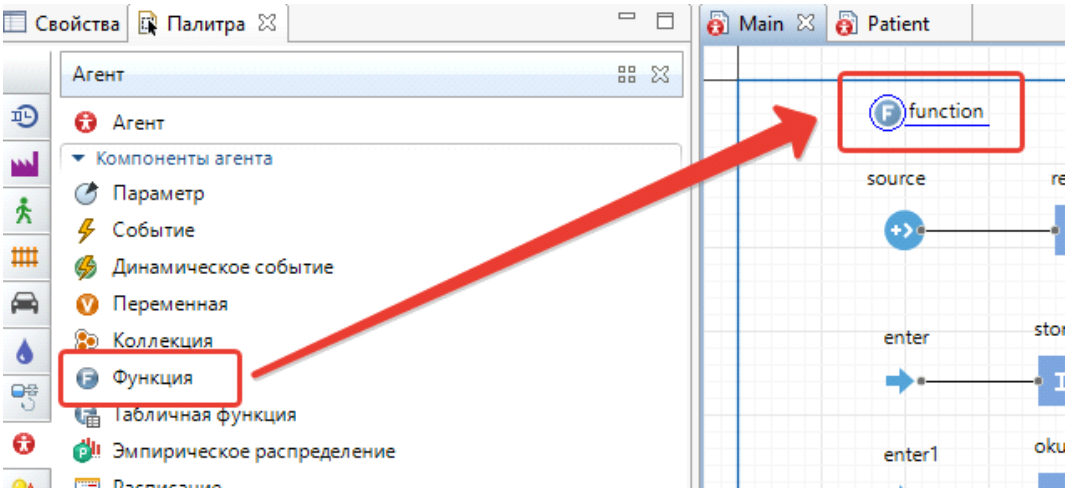


Рис. 7. Создание функции

В свойствах функции установите имя **choiceDoctor** (выбор доктора). В таблице **Аргументы функции** добавьте пустую заявку типа **Patient** в качестве аргумента функции (рис. 8).

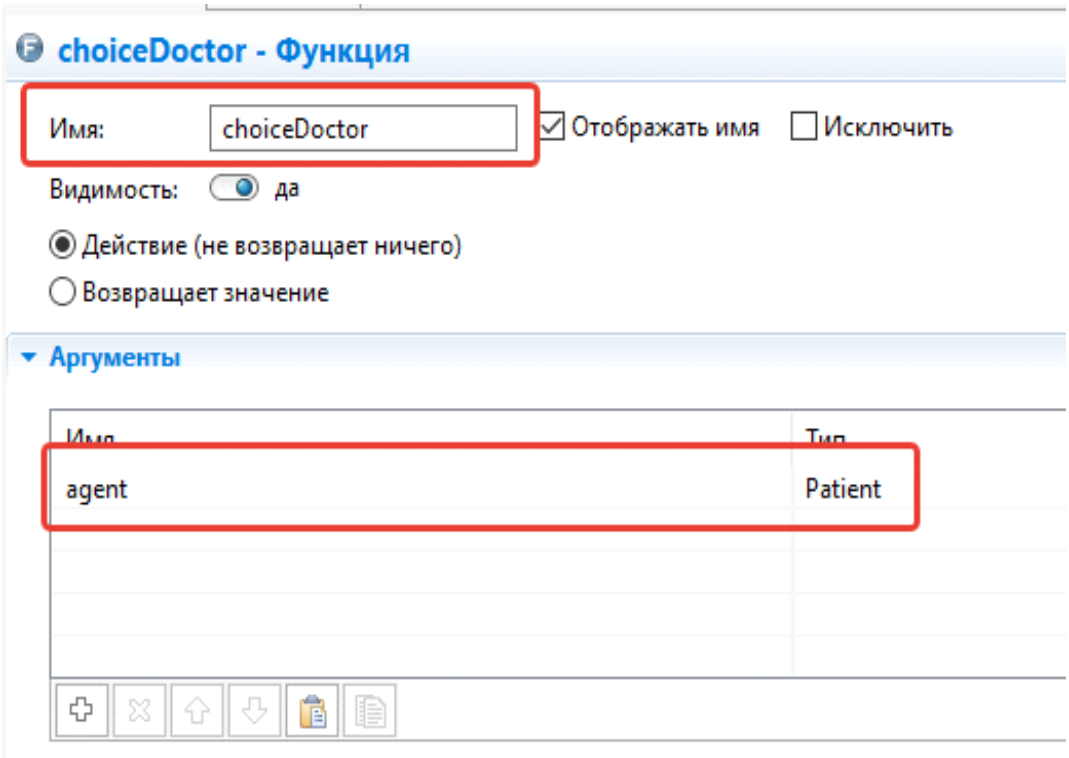


Рис. 8. Настройка функции

- В свойствах функции в разделе **Тело функции** внесите следующий код:

```
if ((agent.blood)&&(agent.stomatolog)&&((agent.okulist)))
  enter3.take(agent);
else {
  int bloodSize = ((!agent.blood)?( blood.size()):(255));
  int okulistSize = ((!agent.okulist)?( okulist.size()):(255));
  int stomatologSize = ((!agent.stomatolog)?( stomatolog.size()):(255));
  if (( bloodSize <= okulistSize)&&(bloodSize <= stomatologSize))
  {
        agent.blood = true;
        enter2.take(agent);
  } else if ( okulistSize <= stomatologSize )
  {
        agent.okulist = true;
        enter1.take(agent);
  } else
  {
        agent.stomatolog = true;
        enter.take(agent);
  }
}
```

Поясним алгоритм работы данной функции.

- Если человек прошел всех специалистов – он свободен (заявка поступает на **enter3** и затем уничтожается).
- Если нет, то мы выясняем размер очереди для каждого из них. Для тех специалистов, которые уже обработали заявку-обращение, делаем этот размер заведомо неприемлемым (размерность **255**).
- Выбираем самую короткую очередь, ставим туда заявку и делаем отметку, что человек этого специалиста прошел – все равно, пока заявка не выйдет из очереди, эти данные не используются.

Этой функцией мы будем пользоваться всякий раз, когда нам нужно будет выяснить, куда теперь направить заявку. Для этого во всех объектах **exit** (**exit**,

exit1, **exit2**, **exit3**) укажем на странице свойств, что при выходе нужно вызвать действие: **choiceDoctor(agent)** (рис. 9).

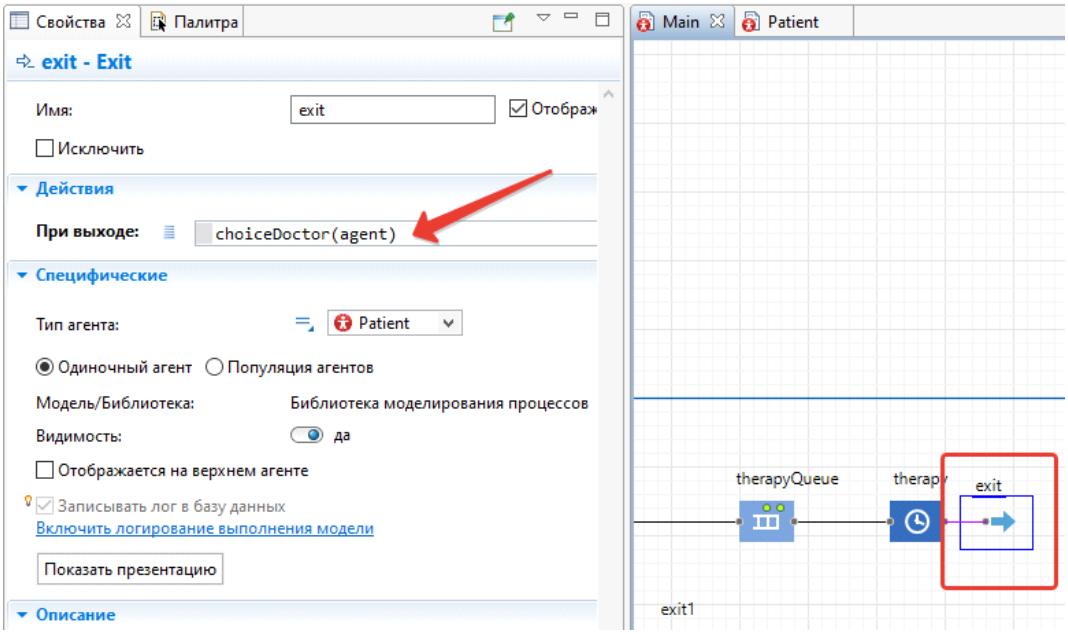


Рис. 9. Назначение функции на действие при выходе

• Основная часть модели готова, но нужно задать еще несколько важных параметров: **время задержки**.

Очевидно, что время задержки на разных этапах будет различным для заявок всех видов. Это может зависеть от процедур, проводимых в кабинетах разных специалистов, но сами причины нам не важны. Важно, как распределится время обработки каждой заявки на своем этапе. У нас нет полных статистических сведений, но есть средние оценки. Мы будем считать время обработки заявки случайной величиной. Форма ее распределения нам неизвестна, воспользуемся простейшим подходом – предположим, что это «треугольник» с вершиной в области среднего значения. В реальности время обслуживания посетителей измеряется в **минутах**, однако для модели удобнее выбрать **секунды** в качестве единицы времени, так она быстрее отработает, и мы получим результаты. По окончании моделирования результаты, полученные в секундах, отметим как минуты.

Предположим, что:

• Время поиска и выдачи документов в регистратуре – от 3 до 10 секунд, в среднем – 5. Зададим время задержки для блока **reception** так: **triangular (3, 5, 10)**. Для начала у нас только одно окно выдачи (рис. 10).

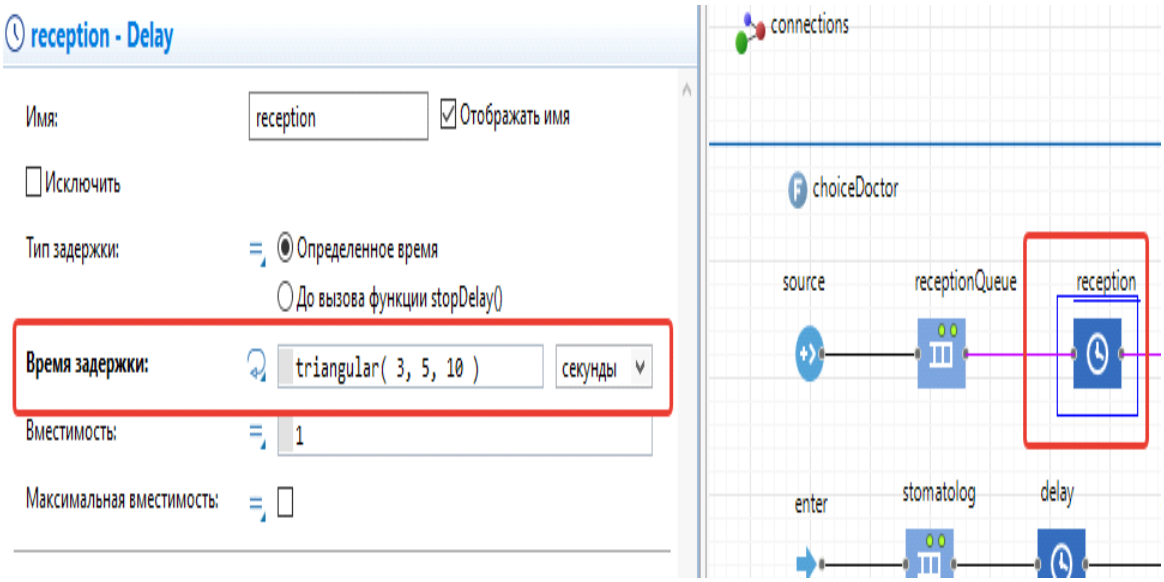


Рис. 10. Настройка времени задержки для регистратуры

- Терапевт работает быстрее, время осмотра одного человека – от 2 до 5 минут, в среднем – 3. Для блока **therapy** значение времени задержки установите **triangular(2,3,5) секунд**.
- Стоматолог (блок **delay**): от 5 до 15, в среднем – 10 секунд.
- Окулист (блок **delay1**) – от 5 до 10, в среднем – 7.
- Забор крови (блок **delay2**) от 1 до 5 минут, в среднем – 3.

Время на переходы мы игнорируем – предположим, что все происходит на одном этаже и человек видит все очереди.

Нетрудно посчитать, что формально максимум времени на полный осмотр – **45 секунд**.

- Теперь посмотрим, как это будет происходить в более точной модели. Будем считать, что у нас прибывает один пациент в 2 секунды, то есть в параметрах источника (блок **source**) укажем **Интенсивность прибытия** 0,5 единиц в секунду. Там же проверим, что не забыли указать класс заявки **Patient** при создании (рис. 11).

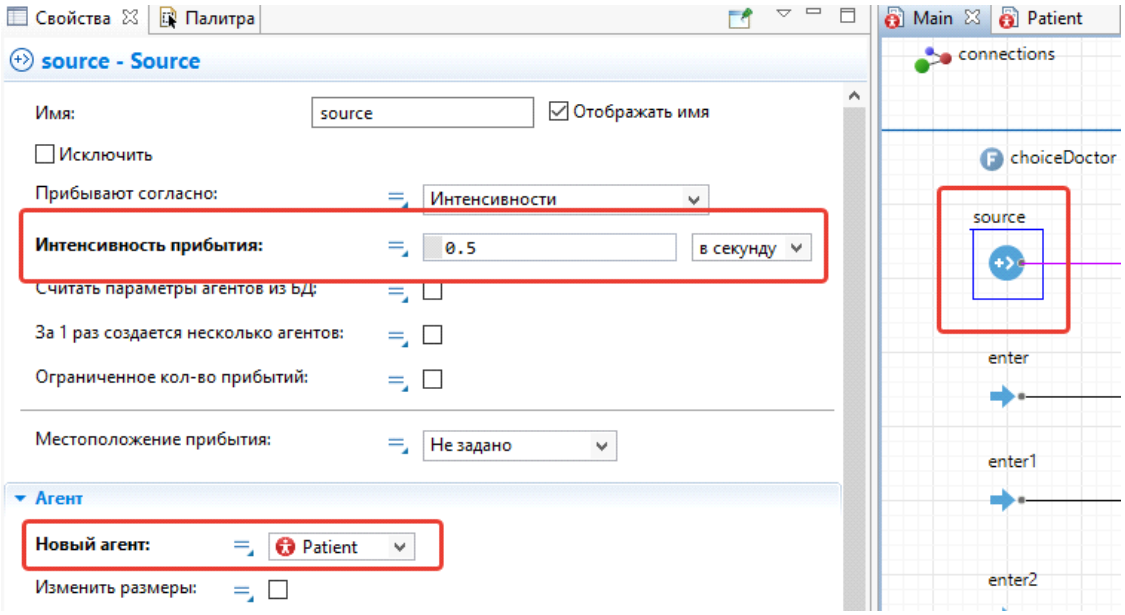


Рис. 11. Настройка свойств для источника заявок

- Ограничим время работы модели: выберем ветку **Simulation**, и в ее свойствах на закладке **Модельное время** укажем конечное время работы – 480 (рис. 12), что примерно соответствует восьми часам работы поликлиники.

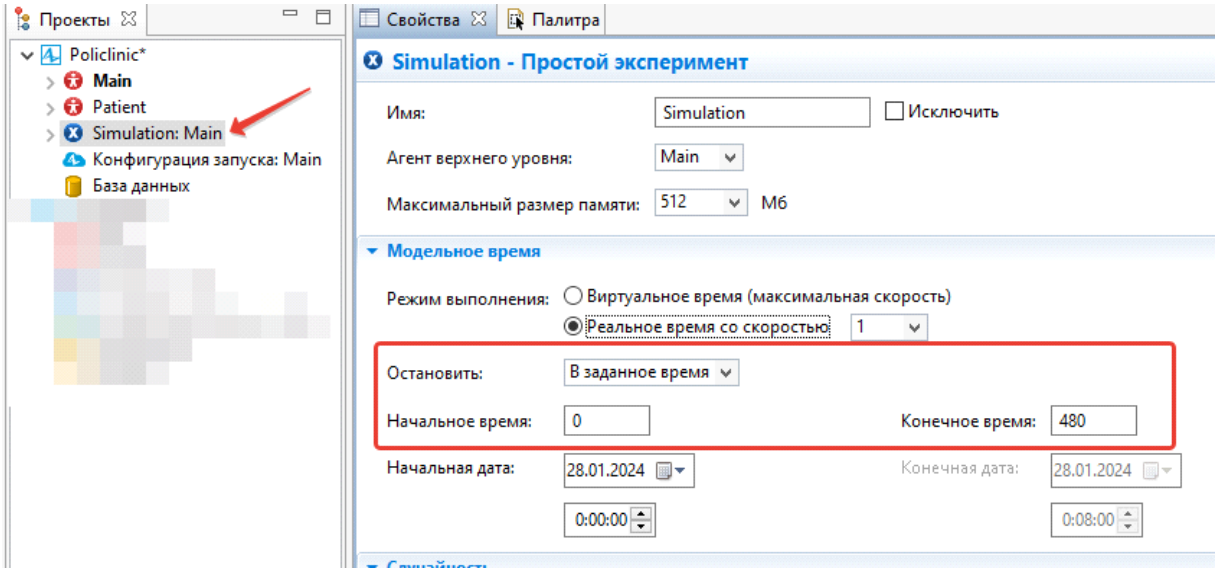


Рис. 12. Настройка свойств симуляции

- Разместим на диаграмме объекты **Гистограмма** и **Данные гистограммы** с палитры **Статистика** (рис. 13).

Рис. 13. Настройка гистограммы

Для объекта **Данные гистограммы** задайте имя **moveTime** (время прохождения медосмотра).

Для объекта **Гистограмма** в разделе **Данные** установите заголовок «Среднее время прохождения осмотра», в поле данные **moveTime** (рис. 14).

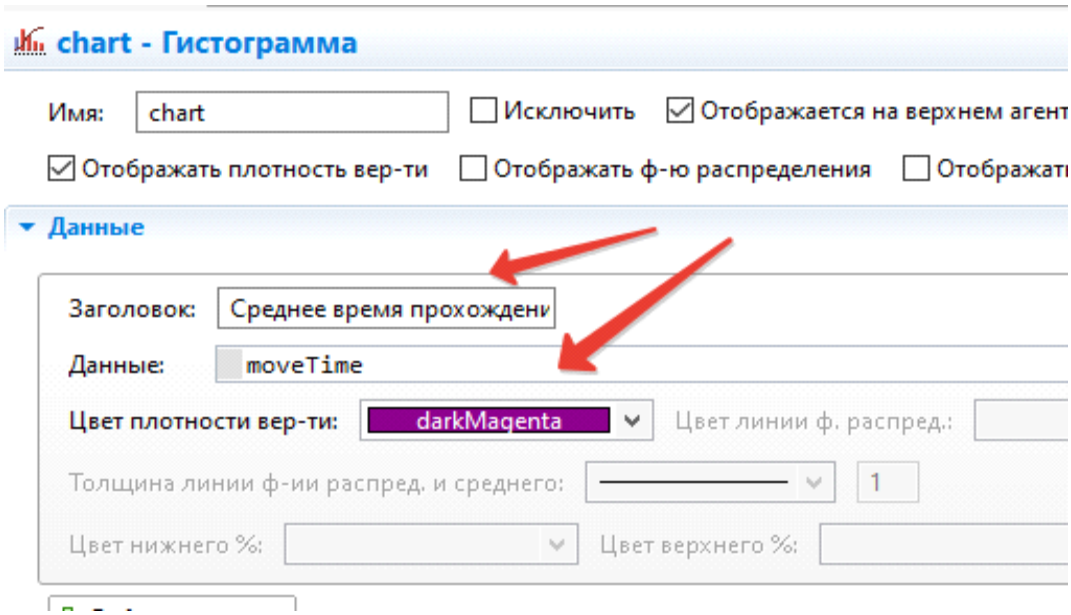


Рис. 14. Настройка гистограммы среднего времени

Фиксировать время будем после обхода всех специалистов. Для нашей модели это будет объект **enter3**. У него будет вычисляться параметр **moveTime** по следующему алгоритму (рис. 15):

moveTime.add(time()-agent.ArrivalTime);

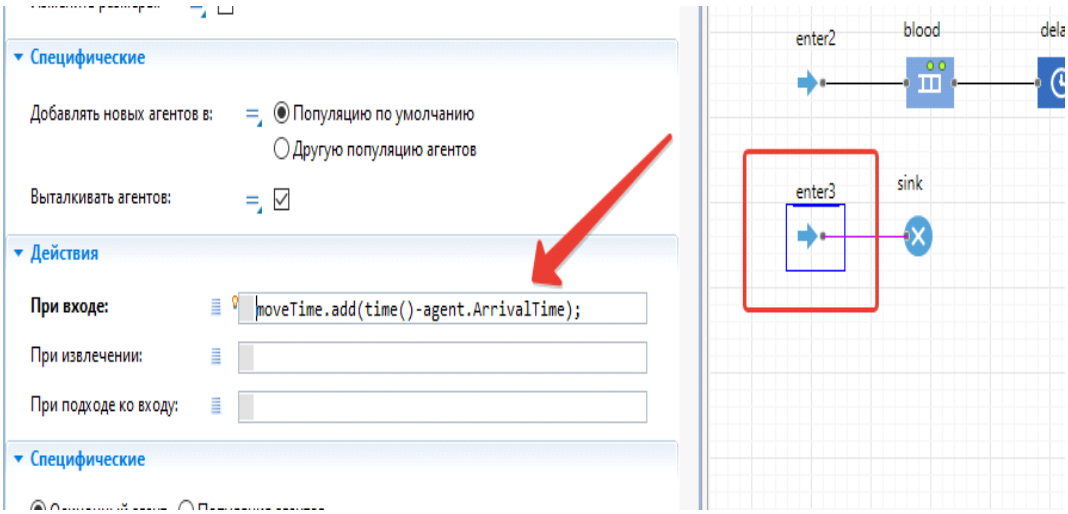


Рис. 15. Настройка алгоритма вычисления времени прохождения осмотра

Это означает, что к диаграмме будет добавлено очередное значение, являющееся разницей между текущим модельным временем и временем появления заявки в модели. Именно эту величину будет отображать нам гистограмма.

• Запустите модель на выполнение (клавиша F5). Мы можем ее ускорить или вовсе включить режим виртуального времени и выполнить ее мгновенно. Все необходимые данные будут на модели (рис. 16).

Сразу можно отметить несколько особенностей:

• У нас появляется очередь в регистратуру, причем она все время растет.

• К окулисту и стоматологу всегда стоит очередь, а на анализ крови ее нет.

Результаты исполнения таковы: примерно через 5,5 минут (в реальности часа) очередь в регистратуру переполняется. За все это время осмотр смогли пройти только 29–30 человек, причем большая часть из них потратила на это более 3 часов.

Вывод. Очевидно, что без снижения времени на осмотр и улучшения работы регистратуры такая система не сможет фактически выполнить свои задачи.

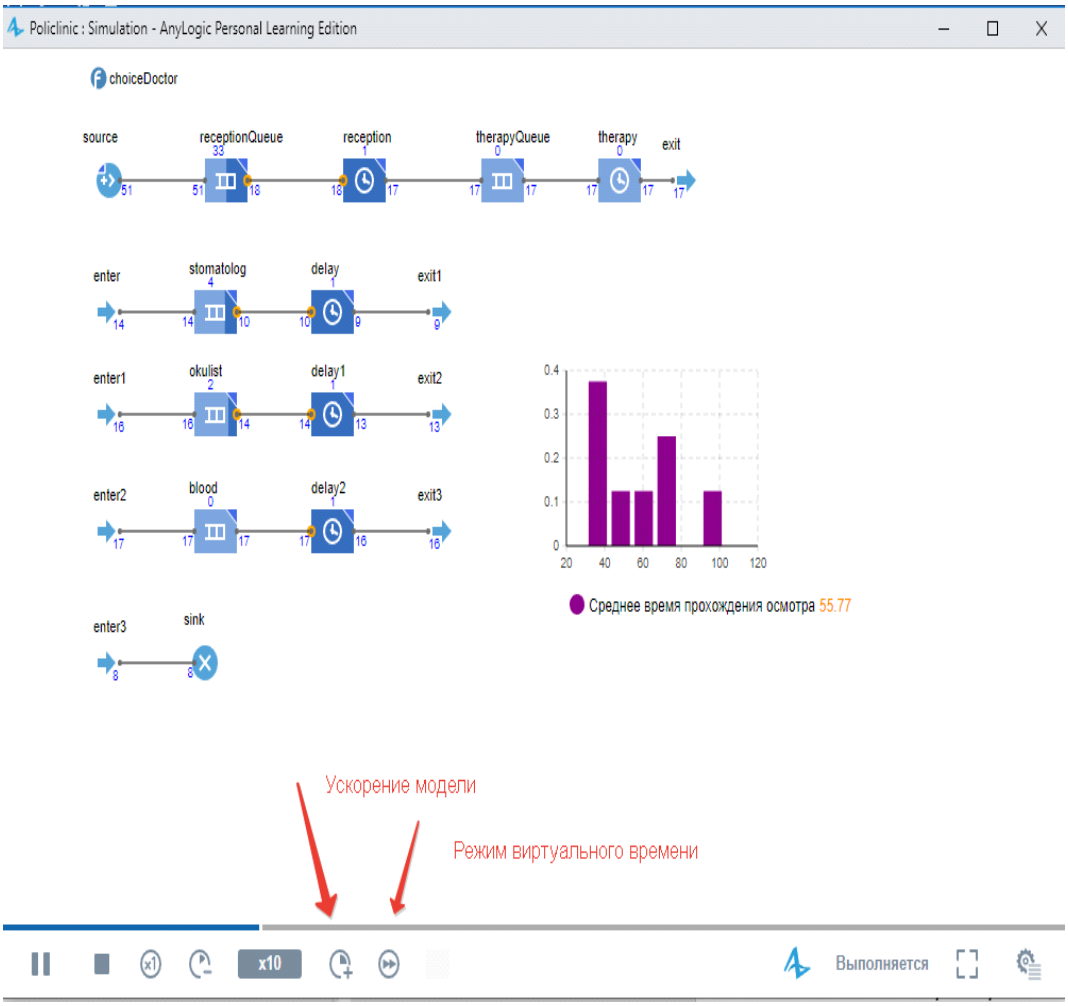


Рис. 16. Модель в исполняемом виде

После построения модели ответьте на исследовательские вопросы:

- Как отразить в параметрах модели тот факт, что коридор, где ждут своей очереди на осмотр – не безразмерный?
- Почему на анализ крови практически не было очереди?
- Что произойдет, если в очереди будут появляться заявки с длительным временем обслуживания – например, сложные случаи в диагностике? Какие параметры нужно будет изменить, чтобы это выяснить?
- Что произойдет в системе, если за счет внедрения электронной карточки врачи смогут уменьшить время на заполнение документов на 2 минуты каждый?
- Как повлияет на такую систему появление второго сотрудника в регистратуре? Вы можете реализовать это в модели с помощью объектов **Service** и **ResourcePool** (они заменят очередь и обслуживание в регистратуре) основной библиотеки.

ГЛАВА 2. ANYLOGIC-ПЕШЕХОДНЫЕ МОДЕЛИ

2.1. AnyLogic-модель работы турникетов в школе

Агентное моделирование позволяет решать задачи, которые невозможно описать математическими методами. Они состоят из набора взаимодействующих отдельных объектов (агентов), каждый из которых имеет какие-то заданные правила поведения и на основе поступающих сведений принимает решение об их применении. Например, такой агент может описывать поведение особи в стаде или человека в толпе. Поведение всей системы будет складываться как результат взаимодействия агентов, но спрогнозировать его математическим соотношением трудно. Например, поведение толпы на выходе из здания.

В AnyLogic содержится специальная пешеходная библиотека, с помощью которой можно создавать модели, имитирующие движение людских потоков. Это может быть актуально при изучении узких мест при проходах в здания, загруженность транспортных потоков, изучение планов эвакуации и другое.

Постановка задачи

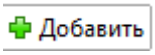
В некоторой школе планируется повысить уровень обеспечения безопасности. В первую очередь, специалисты рекомендовали ограничить проход в школу посторонних, для чего на входе установят два турникета и начнут проверять всех входящих с помощью специальной карты. По мнению директора школы, реальное применение такого подхода приведет к существенным затруднениям перед началом занятий. Как подтвердить или проверить эти предположения?

Ход создания модели

Создадим модель, показывающую первый этаж школы с турникетами и агентами-пешеходами [5], единственной задачей которых будет – пройти через турникет. При этом мы учтем, что проход через турникет (а точнее – проверка) занимает некоторое время, и время от времени будет попадаться человек без документов (например, без карточки), который будет возвращаться назад. Время прохода может меняться (человек проверяет документы медленнее, чем автомат

карточки), вероятность появления «неправильного» документа тоже. Чтобы модель выглядела реалистично, найдем и загрузим типовой план первого этажа школы. Авторы использовал план из типового проекта №45-621/1. Найти его можно в интернете и сохранить на своем компьютере.

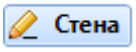
Запустим AnyLogic и создадим новый проект, как мы это делали на предыдущих лабораторных работах. На втором шаге создания проекта укажем, что модель создаем «с нуля», не используя заготовки.

Возьмем за основу типовой проект школы 65-426/1. Найдем в сети план первого этажа, очистим его от лишних артефактов и разместим на рабочем листе. Для этого перейдите на **Палитру**, раздел **Презентация**, перенесите на рабочее поле объект **Изображение**. После переноса откроется диалоговое окно, в котором нужно выбрать файл с планом этажа. Если это диалоговое окно не открылось, то можно в свойствах объекта **Изображение** нажать на кнопку 

.

Заблокируйте изображение, установив флажок **Блокировать** в свойствах изображения. Вы не сможете выбрать заблокированную фигуру в графическом редакторе до тех пор, пока вы не снимете с нее блокировку. Мы делаем так потому, что мы будем рисовать другие фигуры поверх этого изображения, и поэтому мы хотим исключить возможность случайного редактирования изображения при рисовании этих фигур.

Примерное расположение показано на рис. 1.

Теперь прямо поверх плана мы отметим стены (поскольку на картинке, автоматически, система стены выделить не может). Для этого перейдите на **Палитре** в раздел **Пешеходная библиотека** и выберите объект **Стена**. Двойным кликом перейдите в режим редактирования стен 

. Последовательно щелкайте мышью в тех точках диаграммы, куда вы хотите поместить углы стены. Каждый щелчок добавляет часть линии той стены, которую вы рисуете.

Чтобы добавить кривую линию, щелкните левой кнопкой мыши в точку конца кривой линии и двигайте указатель мыши, удерживая кнопку. Пока вы ведете указатель мыши, то заметите, как изменяется радиус кривизны. Чтобы нарисовать окружность, двигайте указателем ровно вдоль сетки координат. Отпустите левую кнопку мыши, когда рисунок готов.

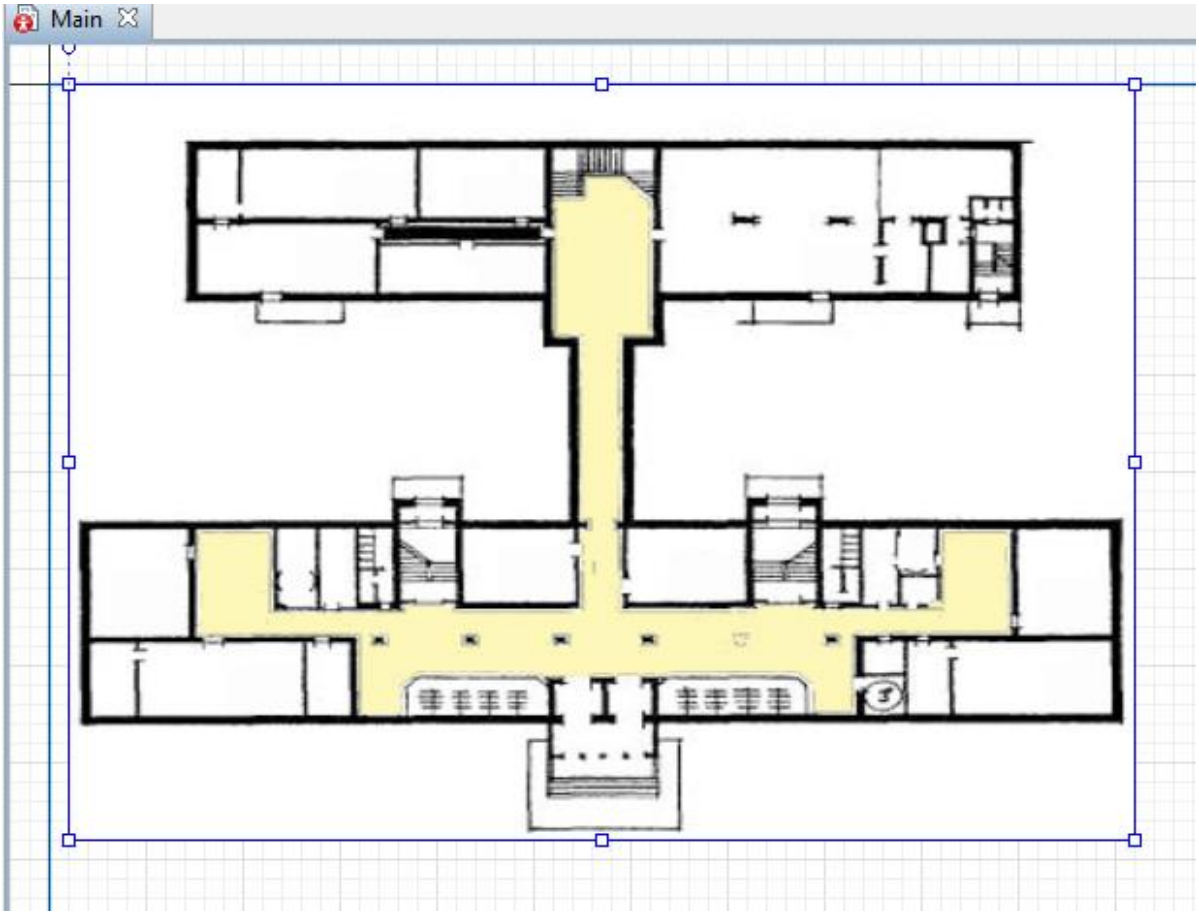


Рис. 1. План школы

Чтобы завершить рисование, добавьте последнюю точку стены двойным щелчком мыши. Нарисуйте стены, как показано на рис. 2.

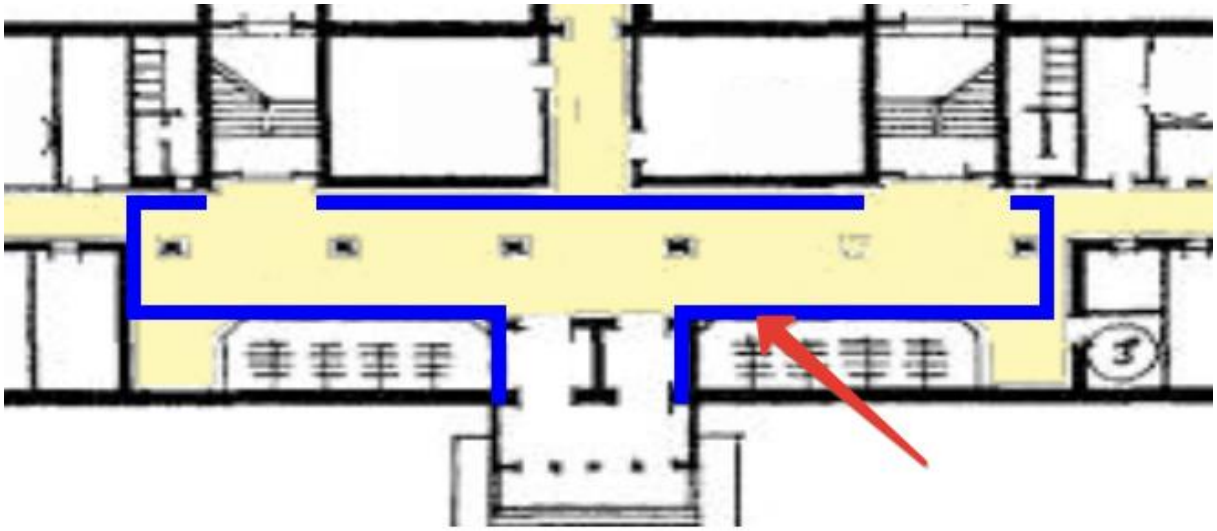
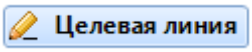


Рис. 2. Стены на плане школы

Там, где в школе предполагаются выходы на другие этажи, сделайте разрывы стен. В этом случае двойным кликом нужно завершить редактирование предыдущей линии и затем нарисовать новую. Выделите все стены с помощью зажатой клавиши **Shift**, кликая на каждую стену. В свойствах линий в разделе **Внешний вид** укажите цвет линий **синий,** толщиной **4 pt.**

Ученики будут приходить в школу со стороны крыльца. За это будет отвечать специальный объект **Целевая линия**. Найдите ее на **Палитре** в разделе **Пешеходная библиотека**. Двойным кликом по объекту **Целевая линия** переведите его в режим редактирования 

. Поместите перед входом, как показано на рисунке 3, и в свойствах задайте имя **enter.**

С помощью объекта **Целевая линия** разместите две линии в районе переходов на верхние этажи. Эти линии назовем: левый выход – **exit1**, правый – **exit2** (рис. 3).

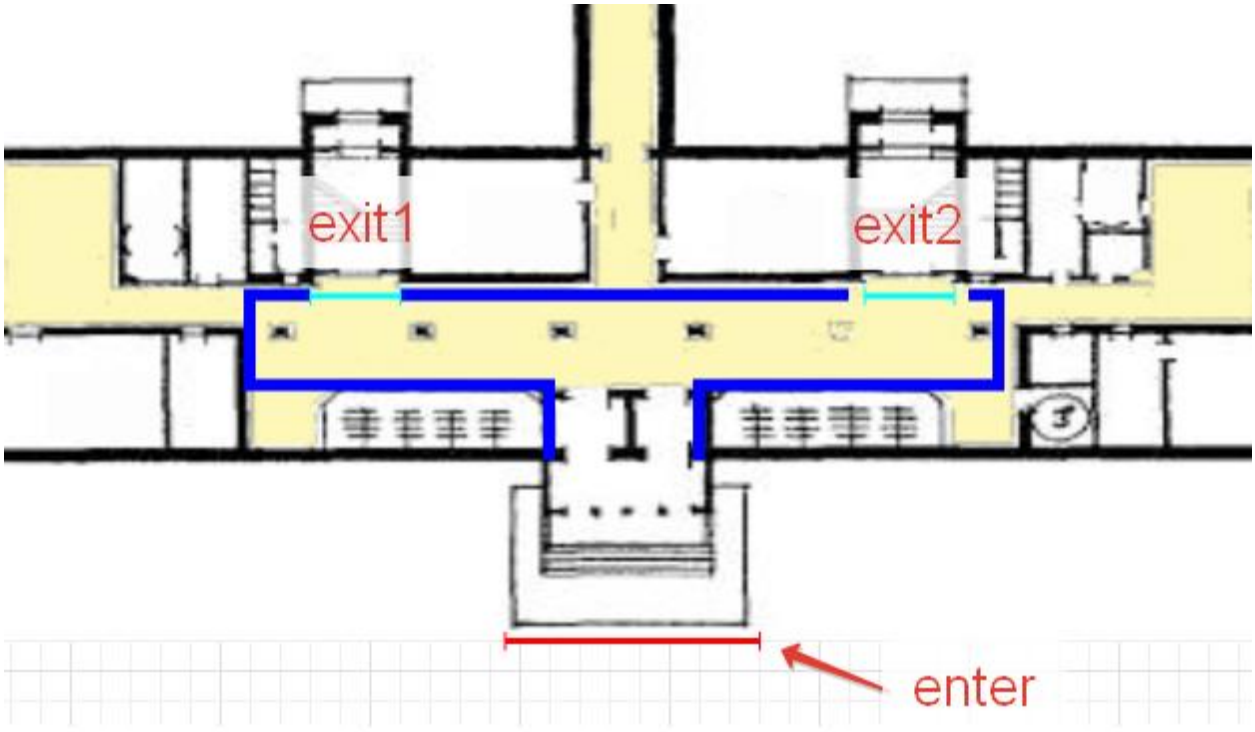


Рис. 3. Создание входа и выходов на плане школы

Построим логику движения пешеходов.

Перенесите в редактор следующие блоки: **pedSource, pedSelectOutput, pedGoTo, pedSink** из **Палитры**, раздела **Пешеходная библиотека** на форму и с помощью портов соедините, как показано на рисунке 4. Если автоматически соединительные линии не получилось создать или линия соединилась не с тем портом,

то ее можно выделить кликом мыши и, нажав на кнопку **Delete**, удалить. Новый порт рисуется от точки (двойной клик по точке) до другой точки (нужно попасть прямо в зеленую точку порта). В свойствах данных объектов укажите те названия, как предложено на рисунке, чтобы в дальнейшем не запутаться.

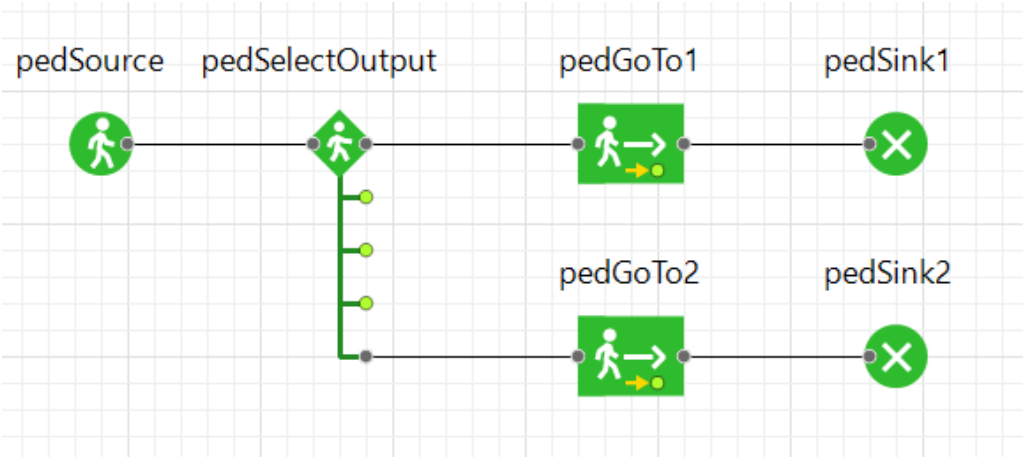


Рис. 4. Логическая схема движения пешеходов

Рассмотрим указанные блоки подробнее.

- Объект **pedSource** создает пешеходов. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток пешеходов. В нашем примере он моделирует приход пассажиров в павильон.
- Объект **pedSelectOutput** направляет входящих в объект пешеходов на один из пяти выходных портов. Выходной порт будет выбираться: либо в соответствии с заданными коэффициентами предпочтения, либо в зависимости от того, для какого из этих портов будет выполнено заданное условие.
- Объект **pedGoTo** моделирует перемещение пешеходов из текущего местоположения в другое (заданное параметром этого объекта). С помощью этого объекта мы будем моделировать то, как пассажиры перемещаются от входа в павильон к поездам метро.
- Объект **pedSink** удаляет поступивших в объект пешеходов из моделируемой среды. Обычно объект используется в качестве конечной точки диаграммы процесса.

Изменим свойства блоков данной диаграммы. Для объекта **pedSource** в свойствах укажите:

Место появления – **линия:**

Целевая линия – выберите из списка **enter**

Прибывают согласно интенсивности

Интенсивность – **1200** в час

Количество прибытий ограничено (галочка)

Максимальное количество прибытий **600**

pedSelectOutput

Использовать вероятности

Коэффициент предпочтения у соединенных портов 0,5, у свободных портов поставьте 0.

pedGoto1

Режим: **достичь цели**

Цель: **линия**

Целевая линия: выбрать из списка **exit1**

pedGoto2

Режим: **достичь цели**

Цель: линия

Целевая линия: выбрать из списка **exit2**

Свойств для **pedSink1** и **pedSink2** менять не будем.

Установите на вкладке **Проекты** для объекта **Simulation** в окне свойств параметр **Реальное время со скоростью** 10 (рис. 5).

Запустите проект и посмотрите, что получилось.

Результат будет вполне предсказуемым – никаких препятствий нет, 600 человек попадают внутрь примерно за 30 минут (то есть, по дороге нет затруднений – время зависит только от частоты появления и скорости движения).

Дополним нашу модель. Теперь мы поставим два турникета, двери перед каждым из них и две линии как цели для движения в случае отказа (например, забытой или отказавшей карты).

Создайте объекты **Прямоугольная область** из **Палитры** (раздел **Пешеходная библиотека**), разместите в районе входа в школу и назовите соответственно левый – **turniket1**, правый – **turniket2** (рис. 6). Для наглядности можно границу этих прямоугольных областей сделать каким-нибудь цветом.

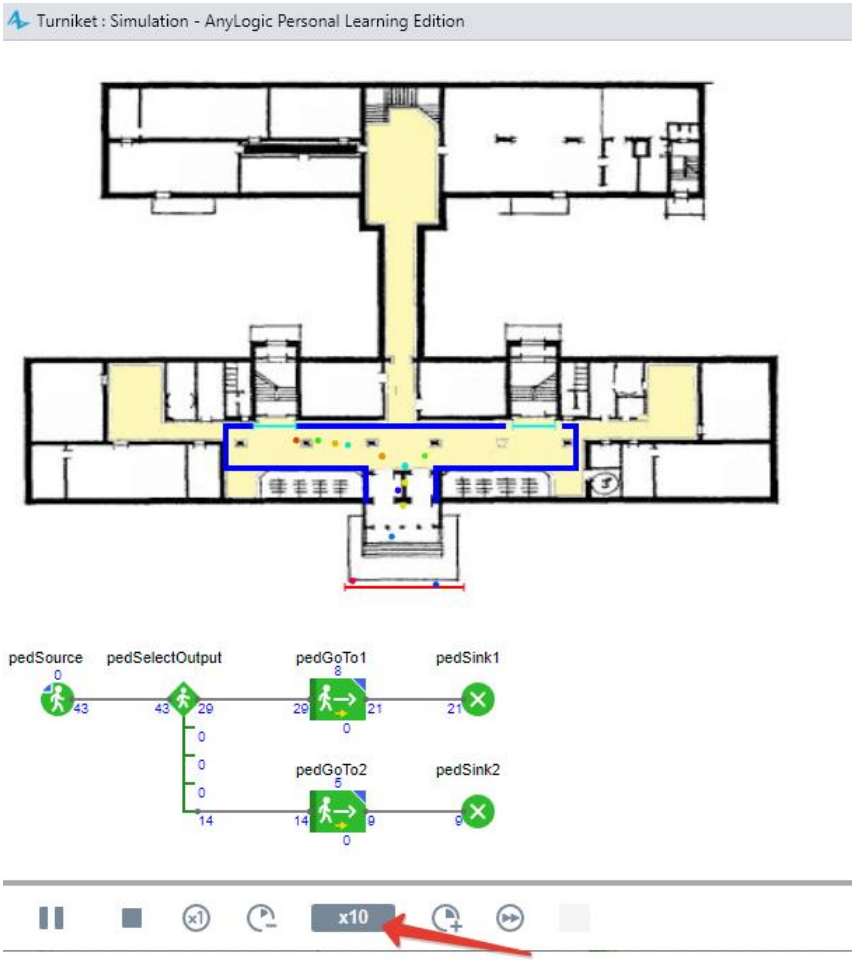


Рис. 5. Проект с турникетами в режиме запуска

Создайте два объекта **Целевая линия** из **Палитры** (раздел **Пешеходная библиотека**) и разместите перед турникетами. Назовите соответственно левый – **gate1**, правый – **gate2** (рис. 6).

Создайте два объекта **Целевая линия** из **Палитры** (раздел **Пешеходная библиотека**) и разместите по разные стороны от школы, это будут линии ухода из школы для тех, кто забыл карточку. Назовите соответственно левый – **home1**, правый – **home2** (рис. 6).

Диаграмму процессов нам тоже нужно изменить. В схеме мы разобьем путь на три этапа: до турникета, ожидание пропуска и проход либо на этаж, либо на выход. Все объекты нужно связать, заполнив параметры. В новых ветвлениях мы уже укажем неравное разделение: будем считать, что **95 %** посетителей имеют действительные карты (первый исход имеет коэффициент выбора **0,95**), а **5 % (0,05)** – нет.

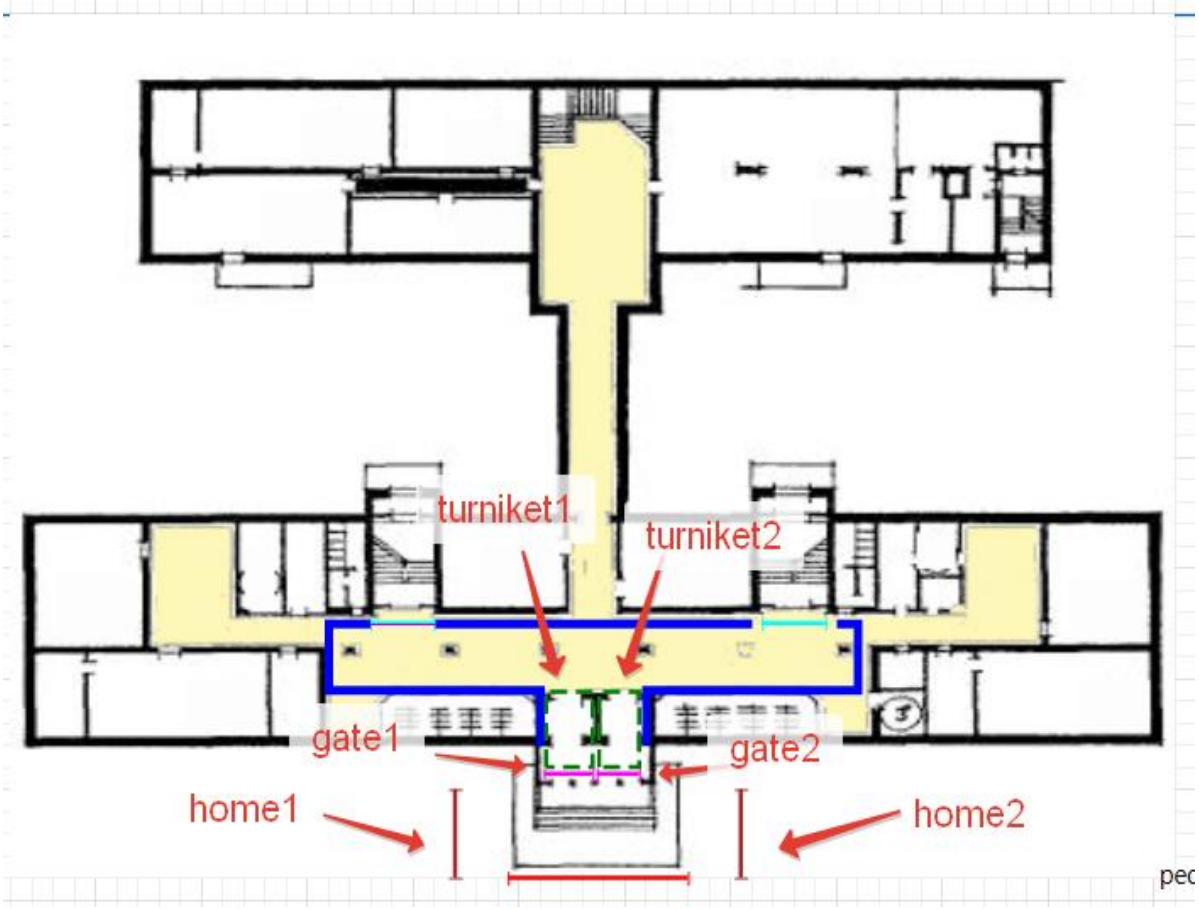


Рис. 6. Создание турникетов, дверей и линий ухода из школы

Создайте диаграмму процесса, как показано на рис. 7. Все объекты, представленные на ней, можно найти на **Палитре** в разделе **Пешеходная библиотека**.

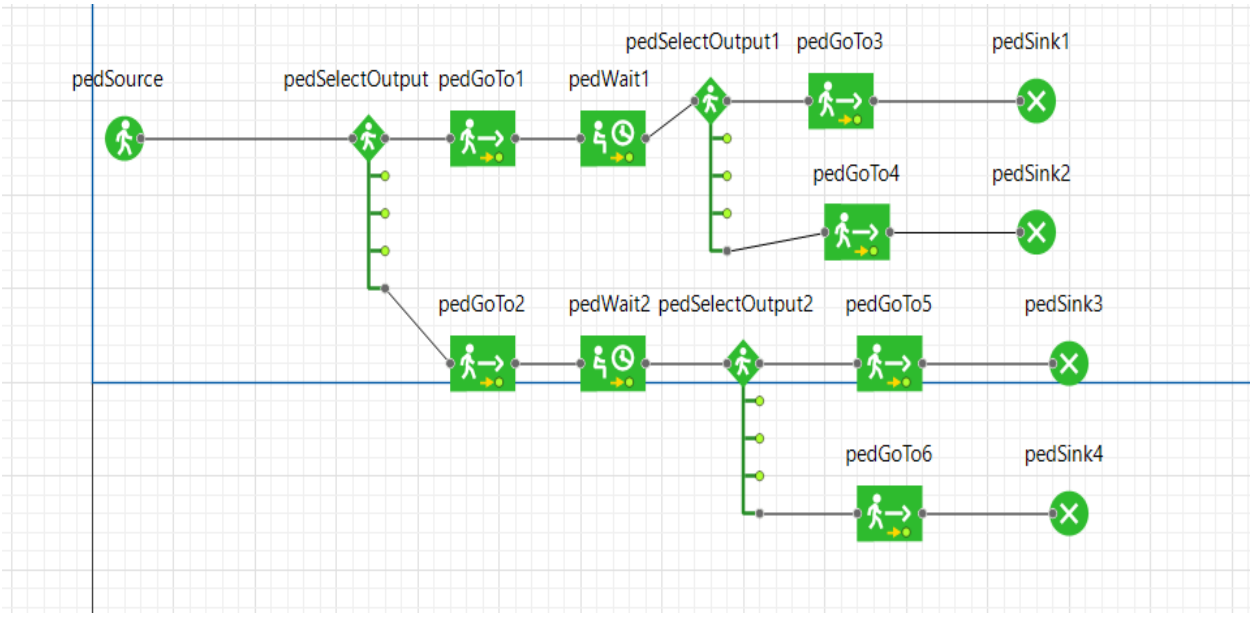


Рис. 7. Диаграмма процесса движения школьников

Здесь добавляется новый объект **pedWait**, который заставляет перейти пешеходов в заданное место и ждать там в течение определенного периода времени. В нашем случае, пройдя двери, школьники будут искать карточку для прохода через турникеты. В 5 % случаев карточки найти не смогут и пойдут за ней домой.

Свяжем все блоки на диаграмме.

Первый блок **pedSelectOutput**: в свойствах объекта поставьте коэффициенты предпочтений **0,5** у крайних портов, остальные заполните нулями.

Для объектов **pedGoto1** и **pedGoto2** укажите в качестве целевых линий соответственно **gate1** и **gate2** (их можно выбрать из списка поля **Целевая линия**).

Для объектов **pedWait1** и **pedWait2** укажите в свойствах параметр **Место ожидания** значение **область**. В параметре **Область** выберите из списка соответственно **turniket1** и **turniket2**. Время задержки для объектов установите равным **uniform(0,5, 1,0)**, что означает – на проход через турникет у школьников будет уходить от 0,5 до 1 секунды.

Для объектов **pedSelectOutput1** и **pedSelectOutput1** установите коэффициенты предпочтения **1** равным **0,95**, коэффициенты предпочтения **5** равным **0.05**, в остальные введите 0. Это означает, что большинство учеников сможет пройти в школу, а **5 %** пойдут домой за карточкой.

Для объектов **pedGoTo3** и **pedGoTo4** установите в свойствах Режим значение **Достичь цели**, параметр **Цель** значение **Линия**, параметр **Целевая линия** выберите из списка соответственно **exit1** и **home1**. Большинство школьников будет проходить в школу и направляться к левому переходу на второй этаж (exit1), а те, у кого нет карточки – направляются за ней домой (home1). Аналогичным образом настройте свойства для объектов **pedGoTo5** и **pedGoTo6** соответственно на правые выходы.

Запустите модель и проанализируйте, что получилось.

Вы увидите, что в определенный момент школьники будут толпиться в дверях, и будет происходить затруднение прохода в школу. Но точное среднее время прохода в школу отдельного ученика мы увидеть не сможем. Попробуем усовершенствовать модель, чтобы происходил подсчет времени прохода в школу для агентов.

Мы видели, что во время работы модели некоторые пешеходы могут «заблудиться» и застрять на этаже. На реальный результат они повлиять не могут, поэтому мы можем их отбросить. Параметр, который нас на самом деле интере-

сует – это время, которое потребуется пешеходу, чтобы пройти через этаж. Изначально время входа и выхода у пешехода не фиксируется, поэтому создадим нового пешехода, отмечающего время.

Создайте нового агента: на вкладке **Проекты** кликните правой кнопкой мыши по любому объекту и выберите **Создать | Тип агента**. В появившемся окне задайте Имя нового типа **SchoolPed**, флажок **Создать новый тип агента с «нуля»**. Нажмите на кнопку далее (рис. 8).

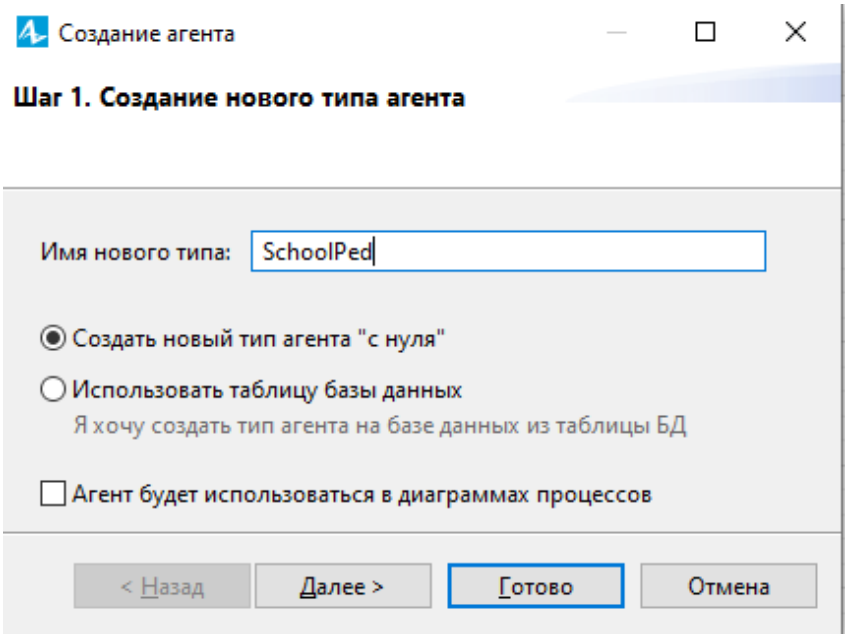


Рис. 8. Создание нового типа агента

На втором шаге укажите анимацию **агента 2D** (рис. 9).

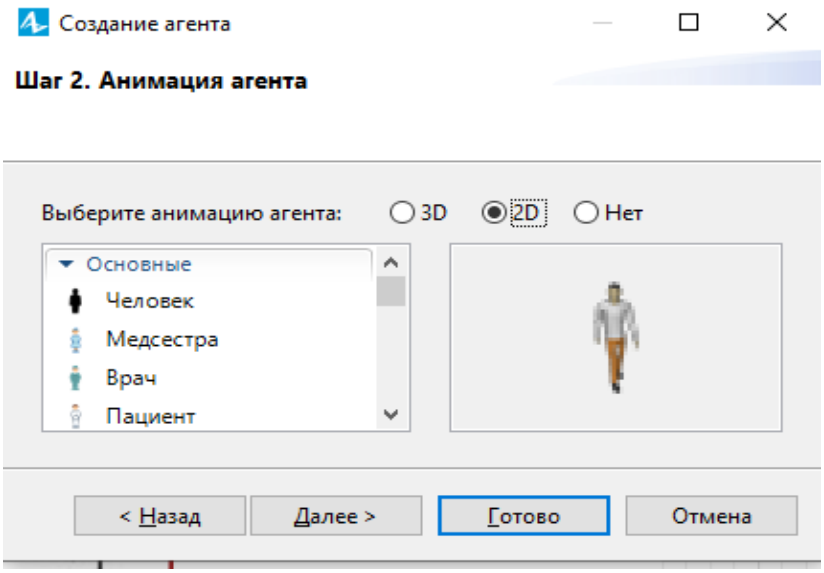


Рис. 9. Шаг второй создания нового агента

На третьем шаге введите параметр **startMove** (тип **double**), который будет хранить время появления агента в диаграмме (рис. 10).

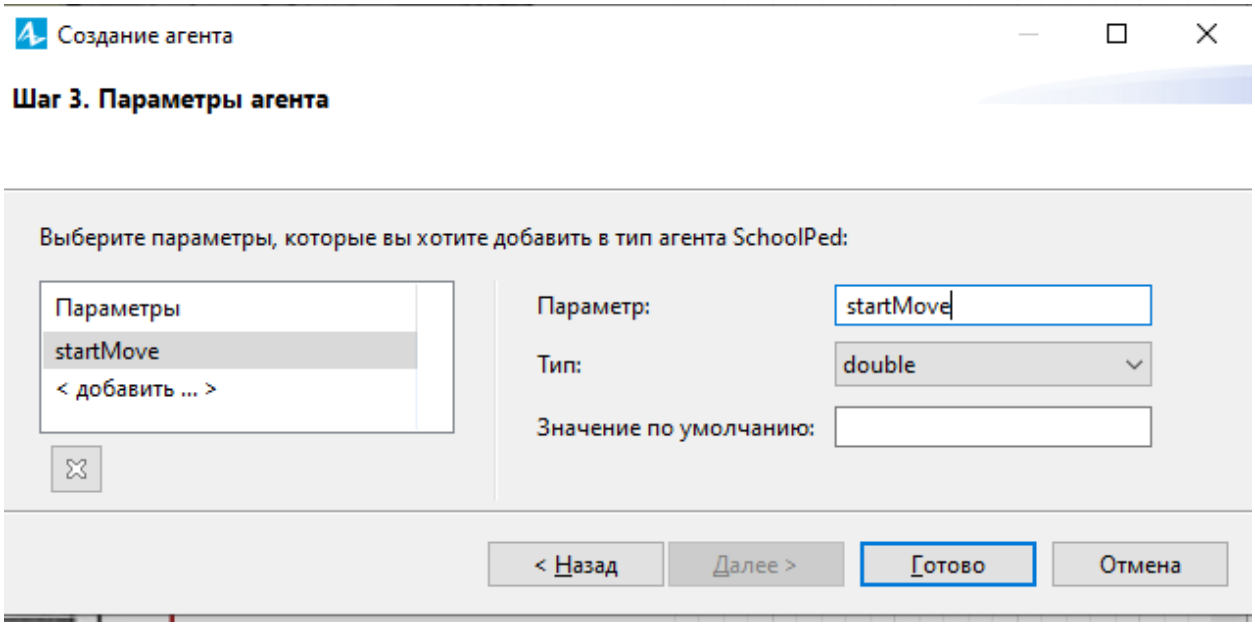
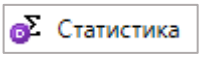


Рис. 10. Шаг третий создания нового агента

На диаграмме процесса для объекта **pedSource** в окне свойств установите параметр **Новый пешеход** значение **SchoolPed** (можно выбрать из списка). И ниже для параметра **Тип пешехода** также выберите **SchoolPed**. Найдите поле **Действия** в окне свойств и задайте действие **при выходе**:

((SchoolPed)ped).startMove = time();

Добавим объект сбора статистики: из палитры **Статистика** перетащим объект **Статистика** 

и назовем его **moveTime**. На всех объектах-выходах (**pedSink** от 1 до 4) укажем действие при входе, добавляющее данные к статистике:

moveTime.add(time()-((SchoolPed)ped).startMove);

Как видно, мы добавляем к статистике время – вычислив его как разницу между текущим временем и временем входа.

Теперь мы можем при выполнении прогонов не наблюдать за счетчиком, а просто отслеживать накопленные данные: минимальное, максимальное и среднее время нахождения прохода школьников. Увидеть результаты можно, просто щелкнув мышью на статистике (рис. 11).

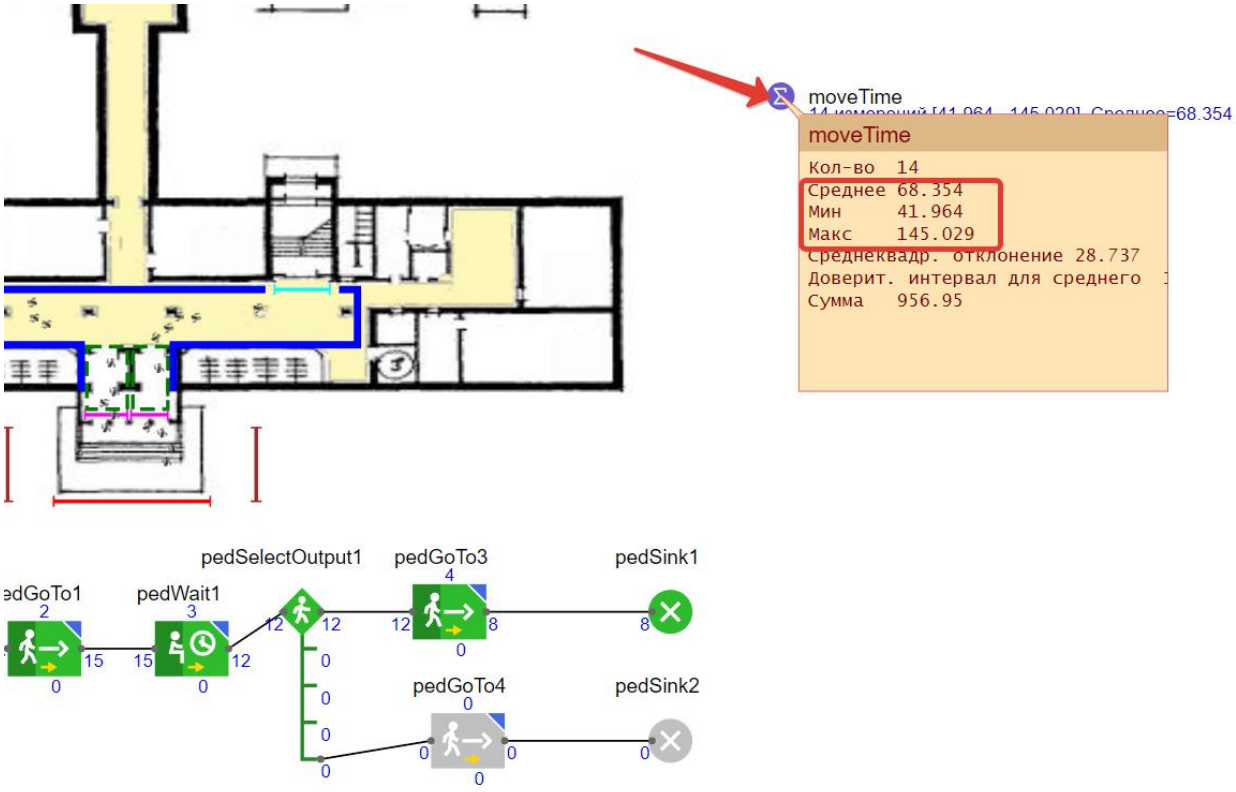


Рис. 11. Просмотр статистики по времени движения школьников

Задания для самостоятельной работы

1. Как изменится время, если карта будет обрабатываться от 1 до 2 секунд?
2. Сколько нужно будет времени на проход, если детям придется еще и переодеваться (задержка от 2 до 5 минут в раздевалке)?
3. Модель предполагает, что люди приходят в течение всех 30 минут равномерно. Тем не менее, очевидно, что на последних 10 минутах плотность будет значительно выше. Исследуйте поведение модели с повышенной частотой появления людей.

2.2. AnyLogic-модель движения пешеходов в замкнутом пространстве

В вузе происходит приемная кампания. Абитуриенты входят через центральный вход университета и концентрируются на первом этаже, встречая информационные стенды. С одной стороны, такие стенды снижают нагрузку со специалистов приемной кампании, однако расположенные в неправильном месте при высокой

пешеходной нагрузке будут приводить к коллапсам, что может негативно сказаться и на приемной кампании вуза. Построим модель движения абитуриентов и исследуем местоположение стенда.

Конечный вид модели показан на рис. 1.

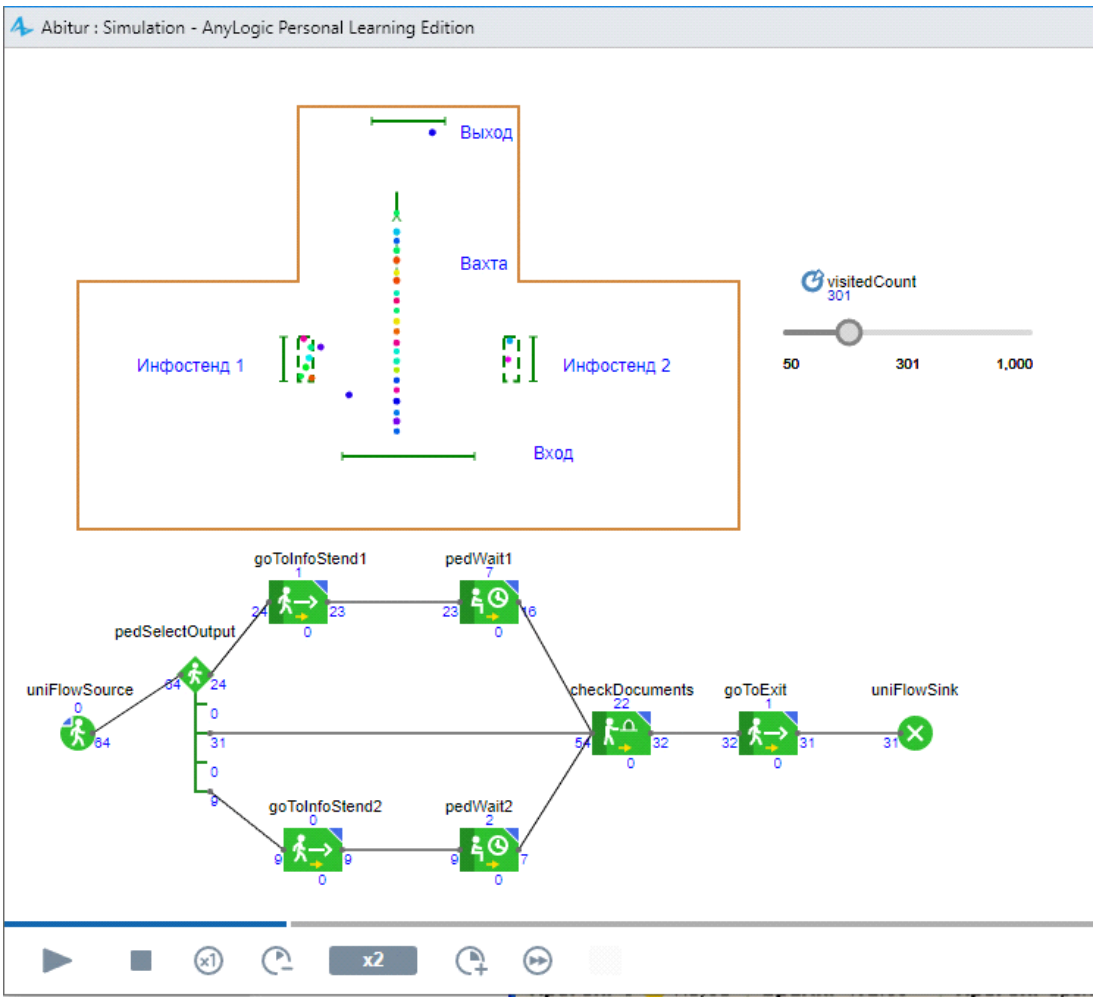


Рис. 1. Конечный вид модели с информационными стендами

Ход создания модели

Создадим в AnyLogic новую модель, назовем ее **Stand**. Модель сохраните в папке C:\Work (рис. 2).

- На диаграмме **Main** постройте замкнутую ломаную линию, с помощью которой будут задаваться границы движения пешеходов. В нашем случае эта зона будет означать фойе университета.
- В палитре **Презентация** двойным щелчком мыши выберите **Ломаная** и создайте примерно такую фигуру, как показано на рис. 3. В свойствах объекта задайте имя **wall** (стена). Также для удобства можно сменить цвет ломаной.

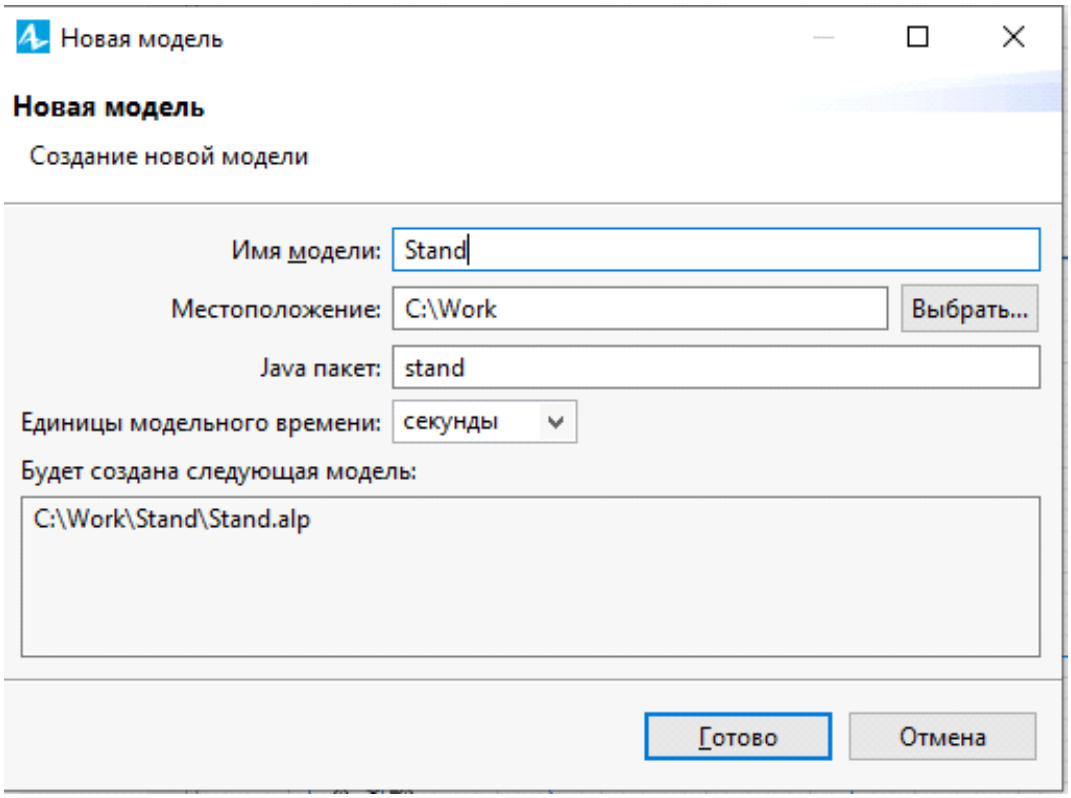


Рис. 2. Создание модели **Stand**

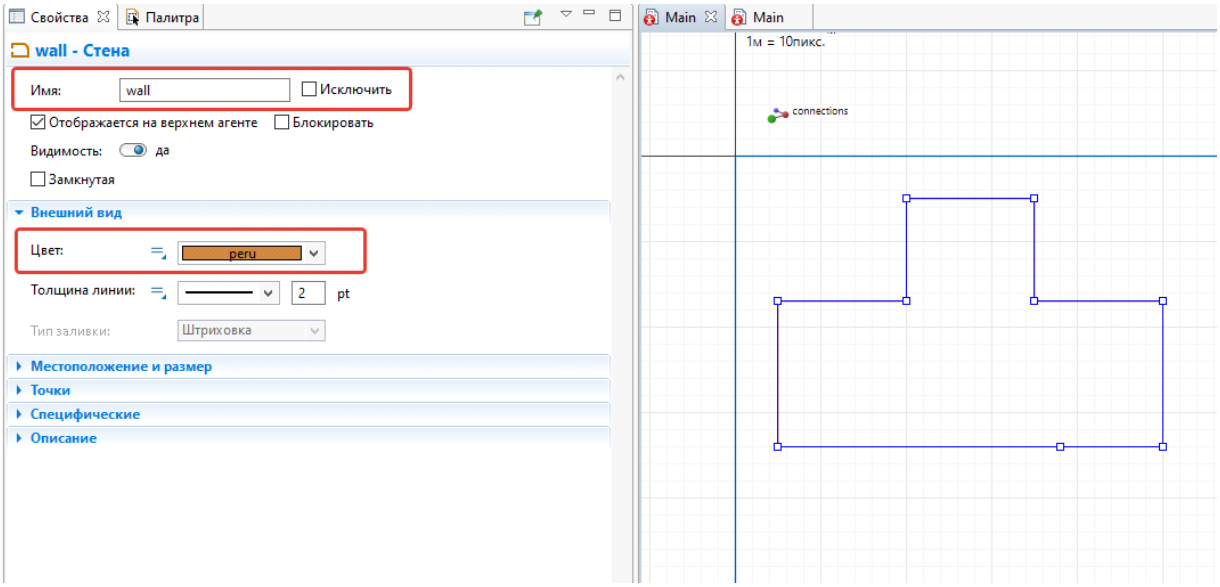


Рис. 2. Создание фигуры для определения зоны фойе

- Отметим внутри нашего фойе следующие объекты (таблица 1). Для подписи объектов можно использовать объект палитры **Презентация Текст**.

Таблица 1

Перечень объектов диаграммы **«main»**

№ п/п	**Тип объекта**	**Палитра**	**Подпись объекта**	**Имя объекта в свойствах**
•	Линия	Презентация	Вход	enter
•	Линия	Презентация	Выход	exit
•	Линия	Презентация	Инфостенд1	infoStand1
•	Прямоугольная область	Презентация		areaInfoStand1
•	Линия	Презентация	Инфостенд2	infoStand2
•	Прямоугольная область	Презентация		areaInfoStand2
•	Сервис с очередями	Пешеходная библиотека	Вахта	post

Вид после размещения объектов показан на рис. 3.

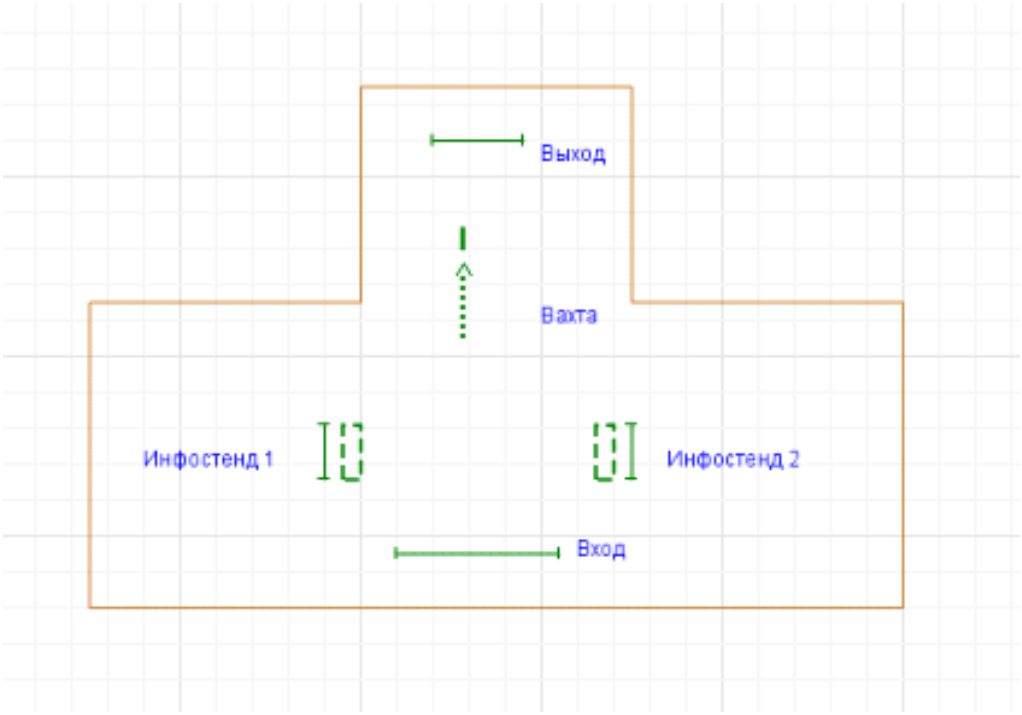


Рис. 3. Фойе с заданными входом, выходом, стендами

• Для связи объектов на диаграмме нужно разместить управляющие элементы пешеходной библиотеки.

Перед размещением объектов поясним идею модели. Абитуриенты попадают в фойе через вход. Далее у них есть 3 возможных пути: к левому стенду, к правому стенду и вперед к вахте. Выбор направления движения будет зависеть от коэффициентов предпочтений, которые мы зададим для управляющего объекта.

Абитуриенты, подошедшие к стендам, находятся возле них какое-то время. Время просмотра стенда будет у всех разным, но в некоторых допустимых границах. Находясь возле стендов, абитуриенты концентрируются в некоторой области (**areaInfoStand1** и **areaInfoStand2**). Это соответствует реальному поведению людей, читающих информацию на стенде, т.е. слишком близко они не подходят, но и не уходят слишком далеко.

Создайте на диаграмме **Main** управляющую схему из объектов, представленных в таблице 2. Общий вид диаграммы показан на рис. 4.

Таблица 2

Описание управляющих элементов диаграммы

№ п/п	Тип объекта	Имя объекта на вкладке Свойства	Значение управляющего элемента
1	2	3	4
•	**Ped Source**	uniFlowSource	Создает посетителей университета. Обычно используется в качестве начальной точки блок-схемы, формализующей поток пешеходов. Создает пешеходов через случайные промежутки времени
•	**Ped Select Output**	pedSelectOutput	Направляет входящих в фойе абитуриентов на один из пяти выходных портов. Выходной порт будет выбираться в соответствии с заданными коэффициентами предпочтения
•	**Ped Go To**	goToInfoStand1	Заставляет пешеходов перейти в заданное место моделируемого пространства, которое может быть задано линией, точкой или областью. Переход будет считаться выполненным, когда пешеход пересечет заданную линию, либо достигнет заданной точки или области. Пешеходы будут искать путь к заданной цели в пределах текущего уровня
•	**Ped Go To**	goToInfoStand2	

Окончание таблицы 2

1	2	3	4
•	**Ped Wait**	pedWait1	Заставляет пешеходов перейти в заданное место и ожидать там в течение определенного периода времени. Место ожидания может быть линией, точкой или областью
•	**Ped Wait**	pedWait1	
•	**Ped Service**	checkDocuments	Сервис задает группу одинаковых физических объектов обслуживания (например, несколько турникетов или автоматов по продаже билетов). Объект позволяет задавать очереди и сервисы в любой комбинации и задавать правила выбора сервисов — какую очередь выбрать, какой сервис выбрать, к какой очереди должен обращаться сервис, может ли сервис обслуживать несколько очередей и т. д.)
•	**Ped Go To**	goToExit	Заставляет пешеходов перейти в заданное место моделируемого пространства, которое может быть задано линией, точкой или областью. Переход будет считаться выполненным, когда пешеход пересечет заданную линию, либо достигнет заданной точки или области. Пешеходы будут искать путь к заданной цели в пределах текущего уровня
•	**Ped Sink**	uniFlowSink	Заставляет пешеходов перейти в заданное место моделируемого пространства, которое может быть задано линией, точкой или областью. Переход будет считаться выполненным, когда пешеход пересечет заданную линию, либо достигнет заданной точки или области. Пешеходы будут искать путь к заданной цели в пределах текущего уровня

Установите необходимые связи между управляющими элементами, как показано на рис. 4. Напомним, чтобы установить связь между элементами, нужно дважды щелкнуть левой кнопкой мыши по порту (маленькой зеленой точке) и провести линию до входящего порта в следующий элемент. По окончании соединения нажать на левую кнопку мыши.

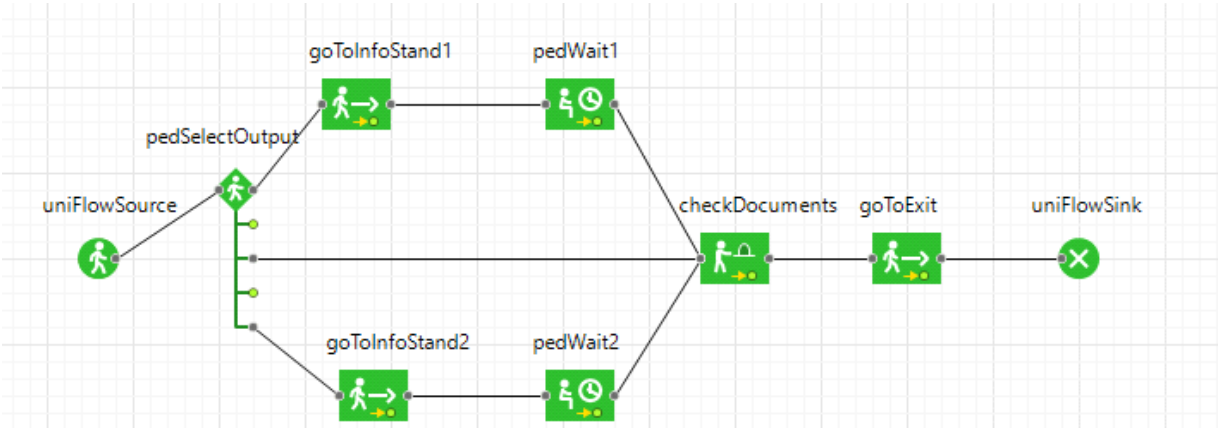


Рис. 4. Общий вид управляющей диаграммы

• Настроим свойства управляющих элементов. Ниже представлены управляющие элементы и значения их свойств для вкладки.

Свойства:

UniFlowSource

- Место появления: **линия**
- Целевая линия: **enter**
- Прибывают согласно: **интенсивности**

PedSelectOutput

- Использовать: **вероятности**
- Коэффициент предпочтения 1: **0,4**
- Коэффициент предпочтения 1: **0**
- Коэффициент предпочтения 1: **0,5**
- Коэффициент предпочтения 1: **0**
- Коэффициент предпочтения 1: **0.,1**

goToInfoStand1

- Режим: **достичь цели**
- Цель: **область**
- Область (выбрать из списка): **areaInfoStand1**
- Точность достижения цели: **0,25 м**

goToInfoStand2

- Режим: **достичь цели**

- Цель: **область**
- Область (выбрать из списка): **areaInfoStand2**

- Точность достижения цели: **0,25 м**

pedWait1

- Место ожидания: **область**
- Область (выбрать из списка): **areaInfoStand1**
- Время задержки: **triangular(30,140,240)**

pedWait2

- Место ожидания: **область**
- Область (выбрать из списка): **areaInfoStand2**
- Время задержки: **triangular(30,140,240)**

CheckDocuments

- Выбирается очередь (выбрать из списка): **самая короткая очередь**
- Время задержки: **uniform(5,0, 20,0)**

goToExit

- Режим: **достичь цели**
- Цель: **линия**
- Целевая линия: **exit**

uniFlowSink

Оставить все свойства по умолчанию.

• Настроим интенсивность прибытия будущих студентов в университет. В обычной жизни в разные периоды времени интенсивность прибытия разная, поэтому и мы настроим динамическое изменение интенсивности.

Перейдите на **палитру Системная динамика** и перенесите **Параметр** в поле своей диаграммы, как показано на рис. 5.

В свойствах параметра установите:

- Имя: **visitedCount**
- Тип (выбрать из списка): **int**
- Значение по умолчанию: **200**

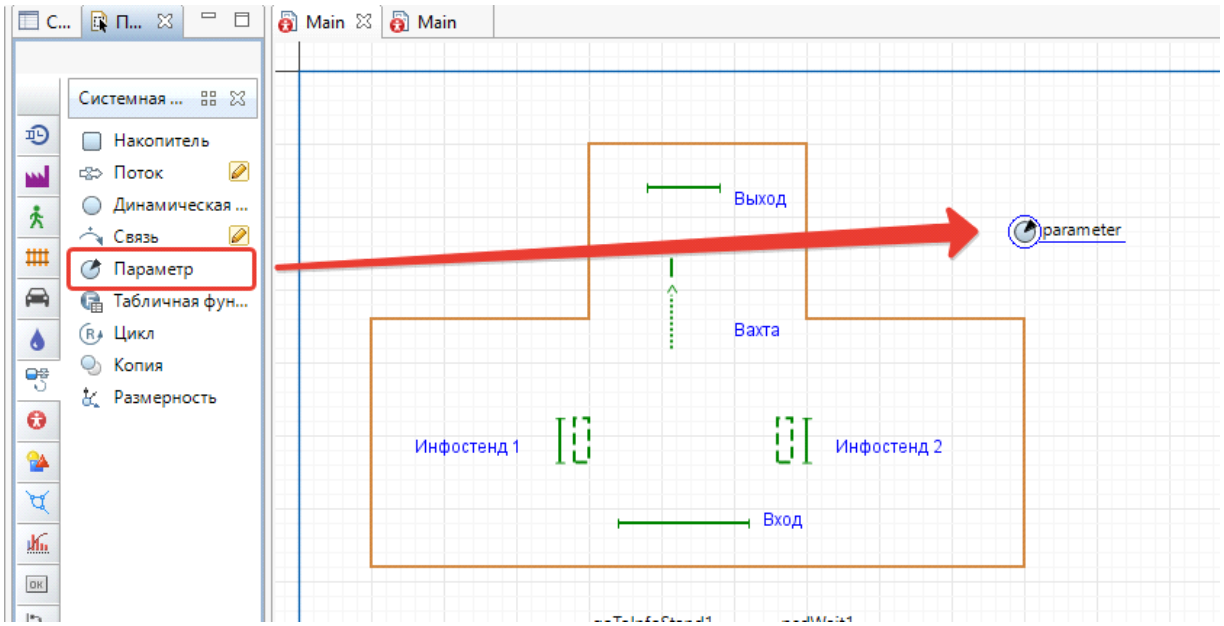


Рис. 5. Создание параметра интенсивности

Добавьте рядом с параметром **visitedCount** объект **Бегунок** с палитры **Элементы управления** (рис. 6).

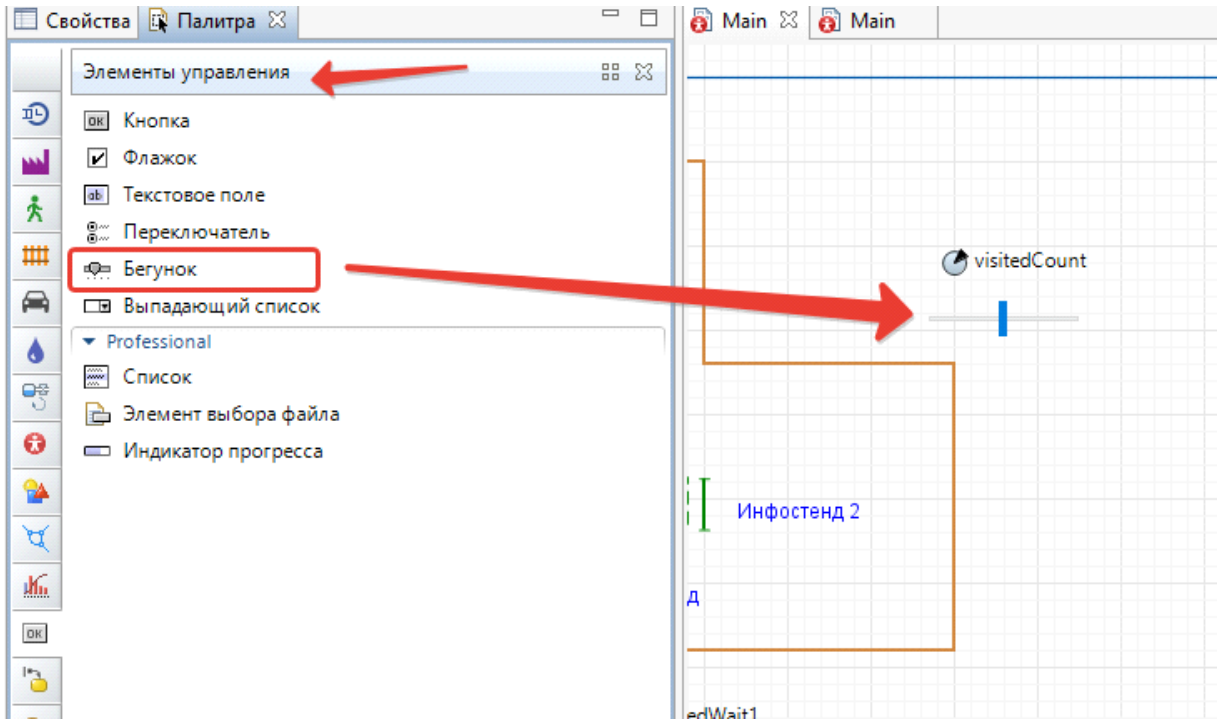


Рис. 6. Создание бегунка управления параметром интенсивности

Для бегунка установите свойства

- Связать с: **visitedCount**
- Минимальное значение: **50**
- Максимальное значение: **1000**

• Запустите модель на выполнение. Изменяя положение бегунка, меняйте интенсивность прибытия абитуриентов. Обратите внимание, когда на вахте образуются очереди?

После построения модели ответьте на вопросы

• Как меняется движение посетителей университета, если интенсивность увеличилась до 300 человек в час? Приведет ли это к возникновению очередей?

• Влияет ли расположение информационных стендов на характер движения и на возникновение очереди при проверке и приеме документов? Где лучше располагать информационные стенды?

• Как повлияет на работу вахты, если приход в университет абитуриентов может совпасть с приходом на занятия студентов?

• Какое оптимальное количество охранников для проверки документов следует привлечь?

ГЛАВА 3. МОДЕЛИ СИСТЕМНОЙ ДИНАМИКИ

AnyLogic-модель распространения продукта

«Системная динамика — это подход имитационного моделирования, своими методами и инструментами позволяющий понять структуру и динамику сложных систем. Также системная динамика — это метод моделирования, использующийся для создания точных компьютерных моделей сложных систем для дальнейшего использования с целью проектирования более эффективной организации и политики взаимоотношений с данной системой. Вместе эти инструменты позволяют нам создавать микромиры-симуляторы, где пространство и время могут быть сжаты и замедлены так, чтобы мы могли изучить последствия наших решений, быстро освоить методы и понять структуру сложных систем, спроектировать тактики и стратегии для большего успеха. (Джон Штерман. Бизнес-процессы: Системное мышление и моделирование сложного мира).

AnyLogic поддерживает различные подходы в моделировании. В этом документе описывается системно-динамический подход моделирования, успешно применяемый во многих сферах, в том числе для описания социальных, урбанистических, экологических, бизнес-систем. AnyLogic позволяет создавать комплексные динамические модели, используя стандартную графическую нотацию системной динамики.

Эта работа кратко ознакомит вас с процессом создания имитационной модели в AnyLogic, описывающим системно-динамический подход. Мы создадим простой и наглядный пример – модель распространения нового продукта. По аналогичным законам распространяются эпидемии, слухи и новые технологии. Мы создадим классическую модель распространения инноваций. Модель описывает процесс распространения нового продукта.

Изначально продукт никому не известен, и для того чтобы люди начали его приобретать, он рекламируется. В итоге определенная доля людей приобретает

продукт под воздействием рекламы. Также люди приобретают продукт в результате общения с теми, кто этот продукт уже приобрел (сарафанное радио). Процесс приобретения нового продукта под влиянием убеждения его владельцев чем-то похож на распространение эпидемии и слухов.

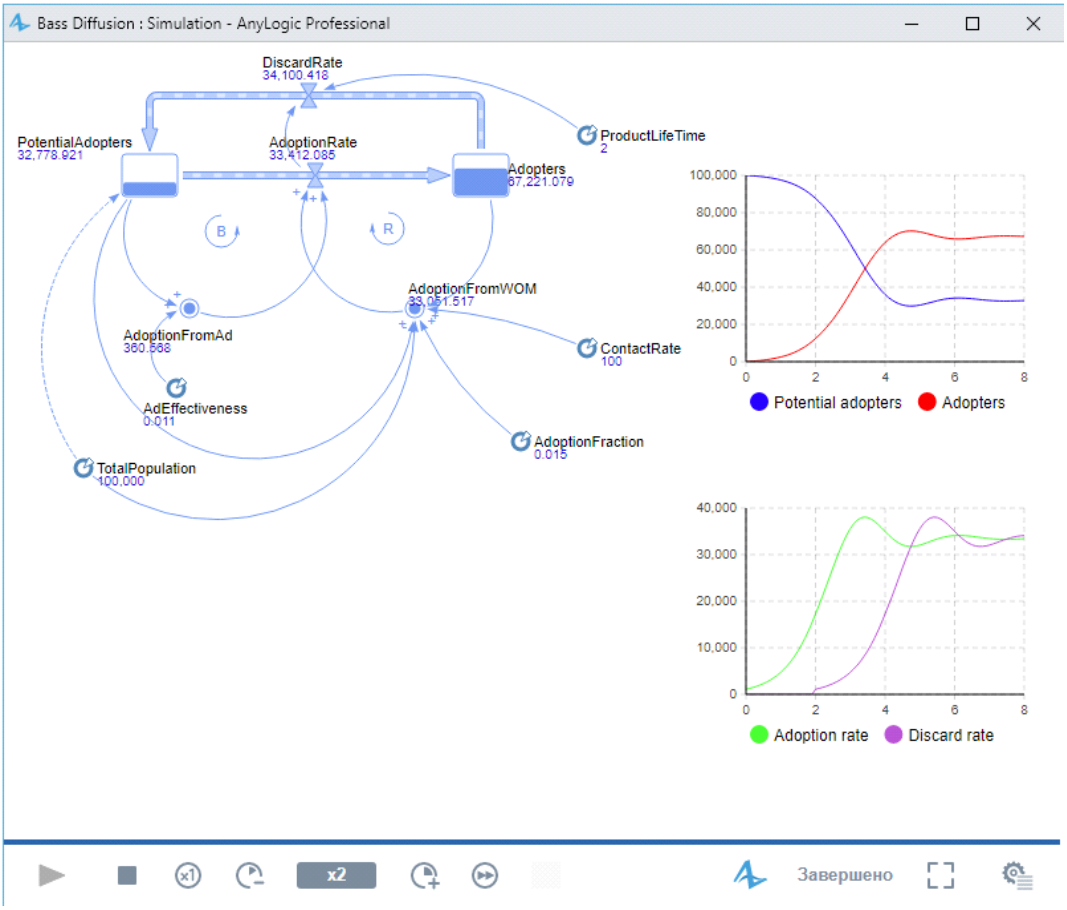


Рис. 1. Модель распространения инноваций

Мы должны проанализировать нашу модель, чтобы решить, как ее можно описать в терминах системной динамики. Мы должны определить ключевые переменные модели и то, как они влияют друг на друга, а затем создать потоковую диаграмму модели. При создании потоковой диаграммы мы должны учесть, какие переменные должны быть представлены накопителями, какие потоками, а какие – динамическими переменными.

Накопители (также называемые уровнями или фондами) представляют собой такие объекты реального мира, в которых сосредотачиваются некоторые ресурсы; их значения изменяются непрерывно. **Потоки** – это активные компоненты системы, они изменяют значения накопителей. В свою очередь, накопители си-

стемы определяют значения потоков. **Динамические переменные** помогают преобразовывать одни числовые значения в другие; они могут произвольно изменять свои значения или быть константами.

При создании потоковой диаграммы выявите переменные, которые накапливают значения с течением времени. В нашей модели численности потребителей и потенциальных потребителей продукта являются накопителями, а процесс приобретения продукта – потоком.

Системно-динамическое представление нашей модели показано на рисунке ниже. Накопители обозначаются прямоугольниками, поток – вентилем, а динамические переменные – кружками. Стрелки обозначают причинно-следственные зависимости в модели.

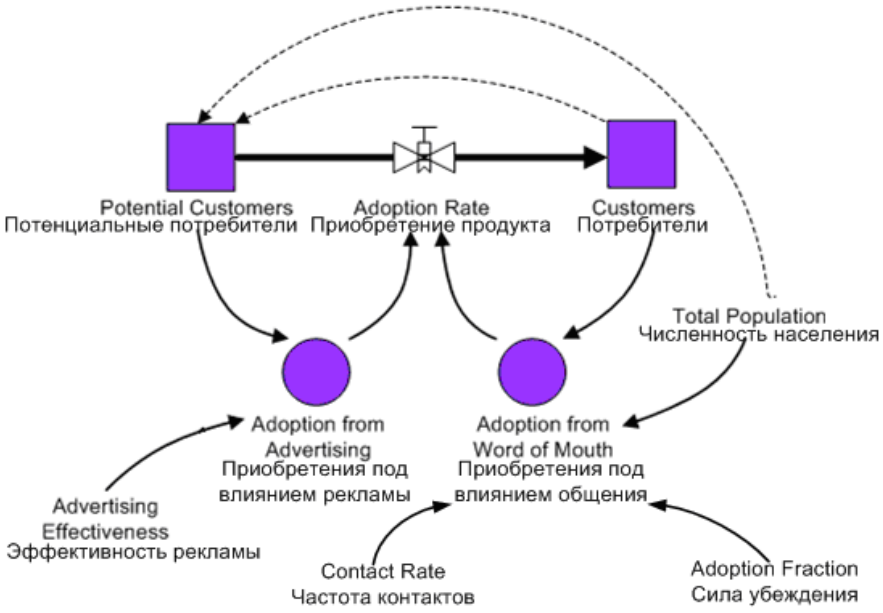


Рис. 2. Схема процесса распространения инновации

Создайте новую модель:

• Щелкните мышью по кнопке панели инструментов **Создать** . Появится диалоговое окно **Новая модель**.

• Задайте имя новой модели. В поле **Имя модели** введите **Bass Diffusion**.

• Выберите каталог, в котором будут сохранены файлы модели. Если вы хотите сменить предложенный по умолчанию каталог на какой-то другой, вы можете ввести путь к нему в поле **Местоположение** или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося по нажатию на кнопку **Выбрать**.

• Выберите **годы** в качестве **Единиц модельного времени**.

- Щелкните мышью по кнопке **Готово** (рис. 3).

Вы создали заготовку новой модели.

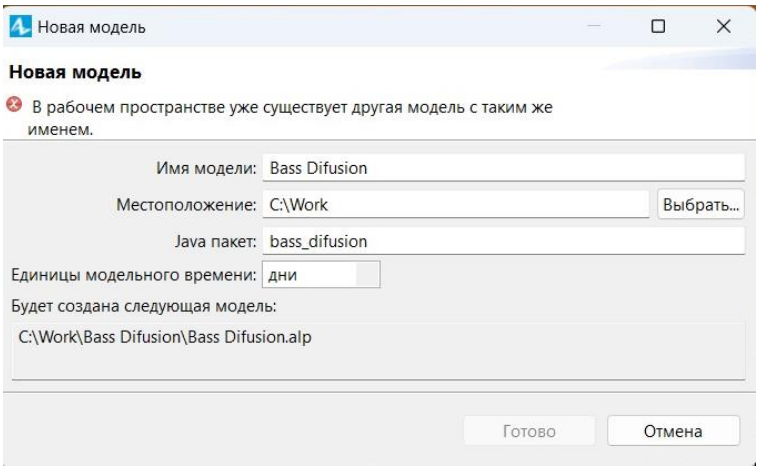


Рис. 3. Новая модель

Пользовательский интерфейс AnyLogic

В левой части рабочей области находятся панель **Проекты** и панель **Палитра**. Панель **Проекты** обеспечивает легкую навигацию по элементам моделей, открытым в текущий момент времени. Модель организована иерархически. Она отображается в виде дерева: сама модель образует верхний уровень дерева. Эксперименты, типы агентов и **Java** классы образуют следующий уровень; элементы, входящие в состав агентов, вложены в соответствующую подветвь типа агента и т. д.

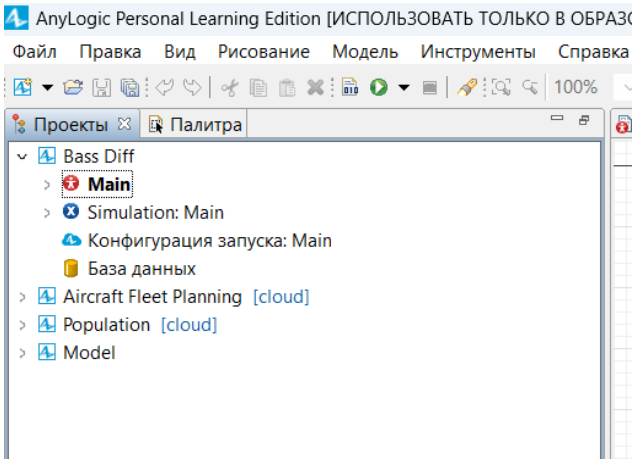


Рис. 4. Вкладка панели **Проекты**

Панель **Палитра** содержит разделенные по категориям элементы, которые могут быть добавлены на графическую диаграмму типа агентов или эксперимента.

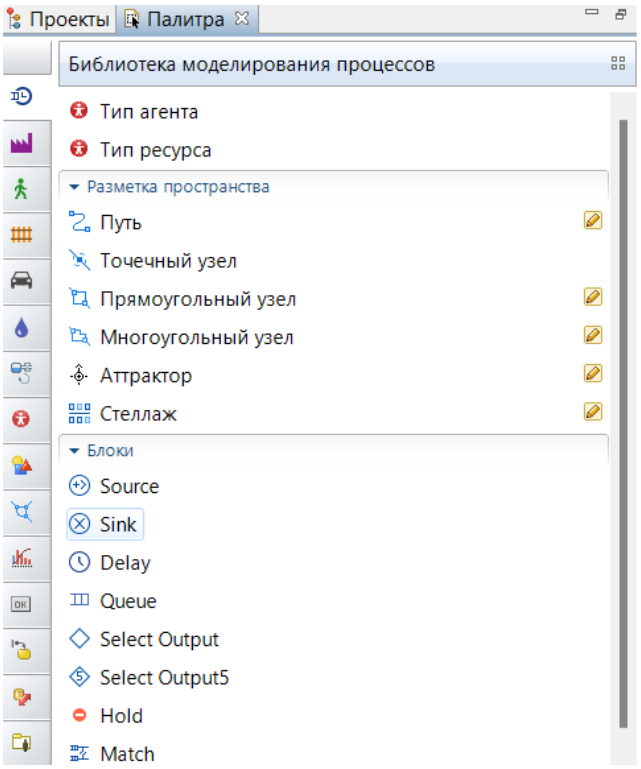


Рис. 5. Вкладка панели **Палитра**

В правой части рабочей области отображается панель **Свойства**. Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели.

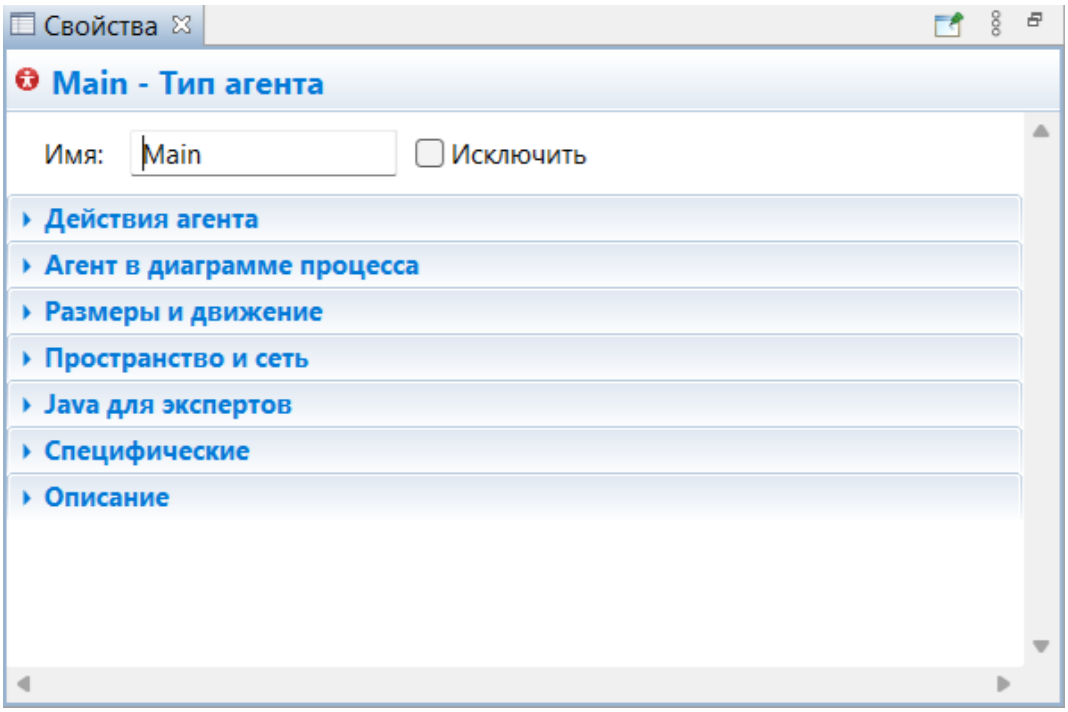


Рис. 6. Вкладка панели **Свойства**

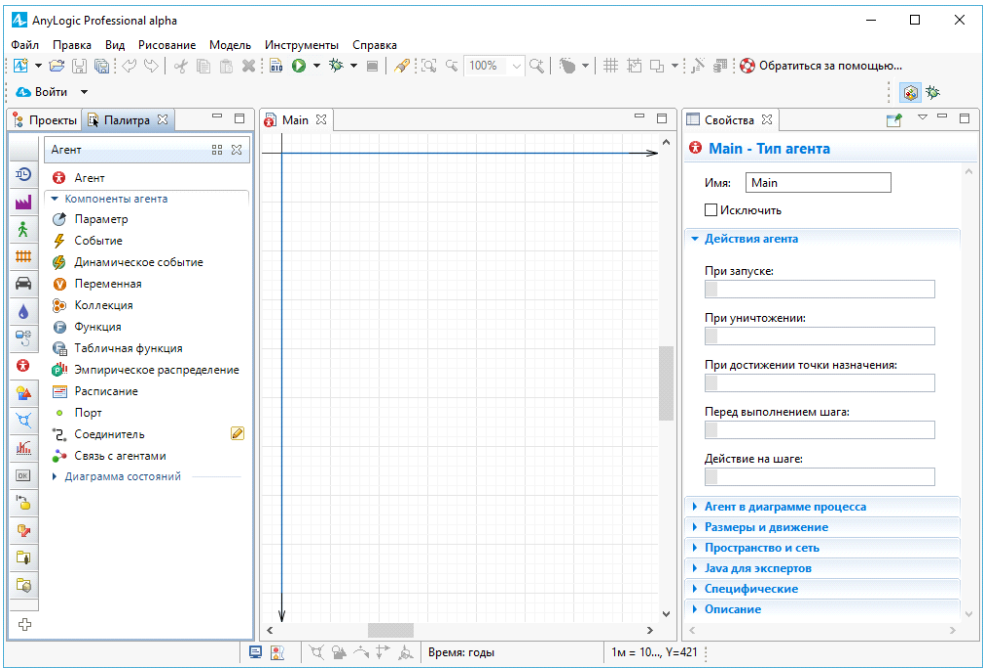


Рис. 7. Вкладка панели **Main**

В центре рабочей области AnyLogic вы увидите **графический редактор** диаграммы агента **Main**.

Создайте накопитель для моделирования численности потенциальных потребителей:

- Вначале откройте палитру **Системная динамика** в панели **Палитра**. Чтобы открыть какую-либо палитру, нужно щелкнуть мышью по соответствующей иконке на вертикальной панели слева от палитры. Пока вы привыкаете к иконкам палитры, можете навести указатель мыши на панель и подождать, пока появится всплывающее окно, в котором вы увидите названия палитр.

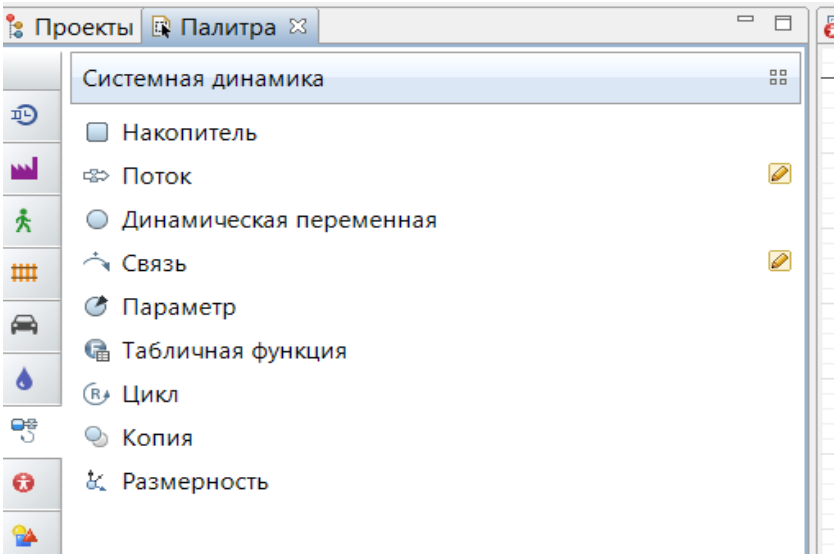


Рис. 8. Вкладка **Системная динамика**

• Перетащите элемент **Накопитель** из палитры **Системная динамика** на диаграмму типа агентов.

• На диаграмме появится маленький голубой прямоугольник, обозначающий переменную-накопитель (что соответствует классической нотации системной динамики).

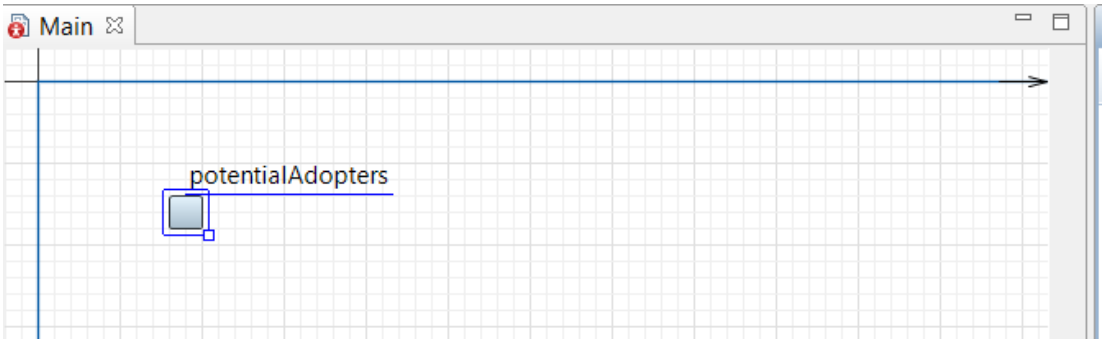


Рис. 9. Первоначальное состояние модели

• Когда вы добавите элемент на диаграмму, он будет выделен, и его свойства будут отображены на панели **Свойства**. Здесь вы можете изменить свойства элемента в соответствии с вашими требованиями. Обратите внимание, что панель **Свойства** является контекстно-зависимой – она отображает свойства выделенного в текущий момент элемента. Поэтому позднее для изменения свойств элемента нужно будет предварительно щелчком мыши выделить его в графическом редакторе.

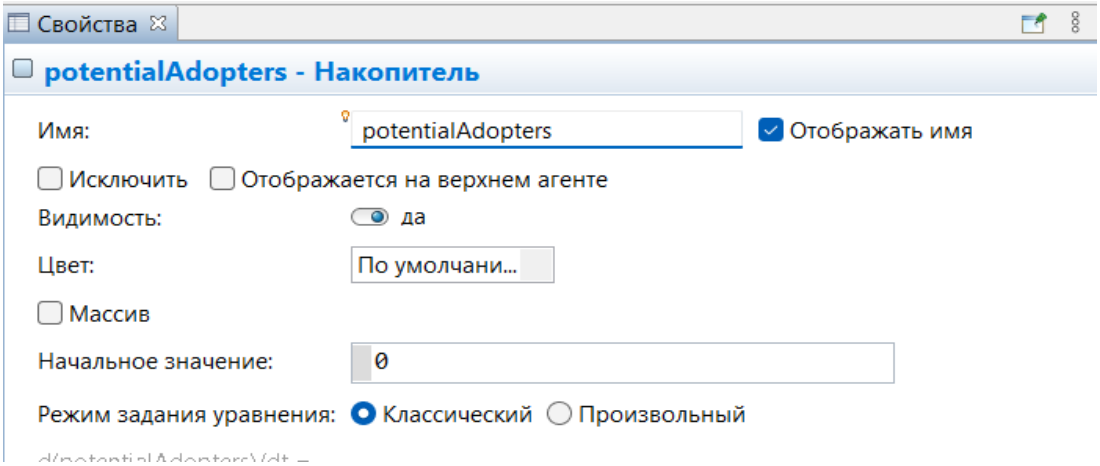


Рис. 10. Свойства **Накопителя**

В панели **Свойства** обращайте внимание на первую строку, показываемую в панели **Свойства** – в ней отображается имя выбранного в текущий момент времени элемента и его тип.

На данном рисунке в панели отображаются свойства выделенного накопителя.

- В поле **Имя** введите имя накопителя: **PotentialAdopters**

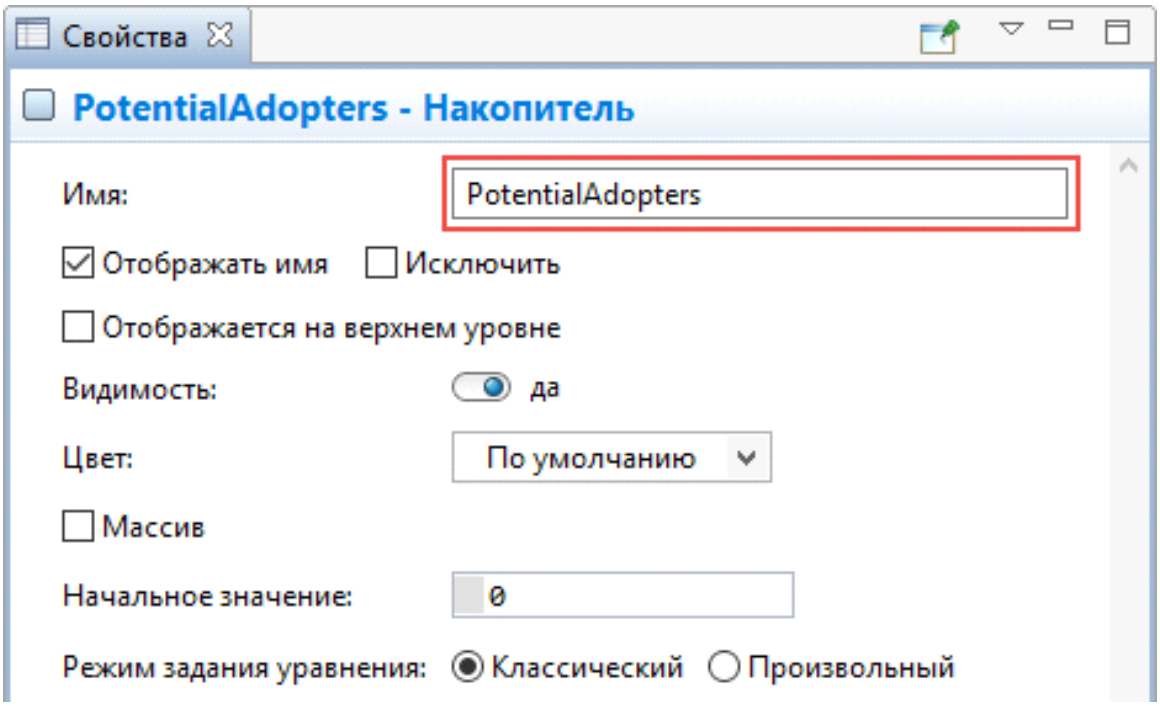


Рис. 11. Свойства накопителя **PotentialAdopters**

- Увеличьте размер значка накопителя. Для этого выделите его щелчком мыши в графическом редакторе и перетащите в сторону появившийся в нижнем правом углу значка маркер.

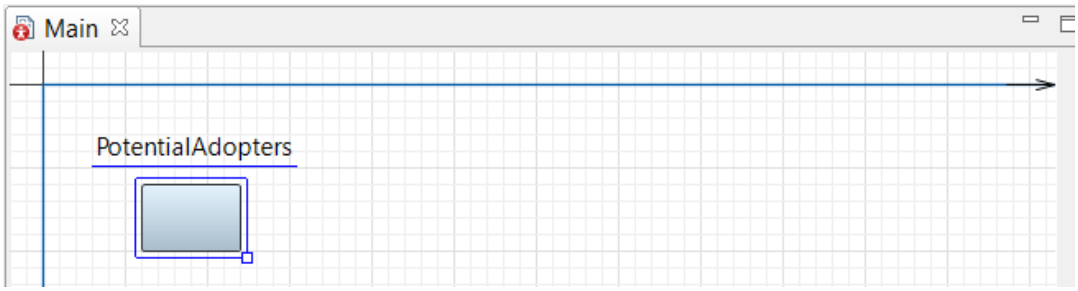


Рис. 12. Накопитель **PotentialAdopters**

Создайте накопитель для моделирования численности потребителей.

- Аналогично создайте еще один накопитель.

Чтобы создать накопитель такого же размера, проще всего клонировать существующий накопитель, перетащив его мышью с нажатой клавишей **Ctrl** (при этом свойства нового элемента будут теми же, что и у клонированного, но в данном случае это не важно, поскольку мы изменили только одно свойство накопителя – его имя).

- Поместите новый накопитель справа от накопителя **PotentialAdopters**, как показано на приведенном ниже рисунке.
- Назовите его **Adopters**.

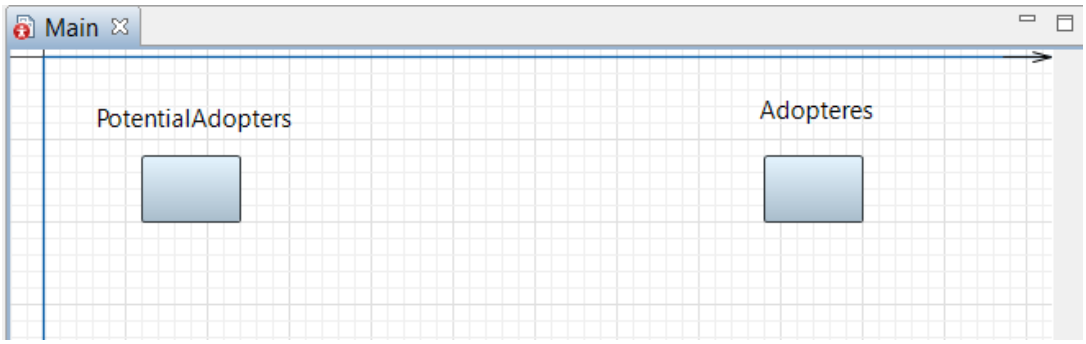


Рис. 13. Накопители **PotentialAdopters» и «Adopters**

На данный момент задание накопителей еще не закончено. Позднее мы зададим начальные значения накопителей, а также формулы, по которым будут вычисляться их значения. Но вначале нам нужно создать поток приобретения продукта.

Добавьте поток, ведущий из накопителя **PotentialAdopters** в накопитель **Adopters.**

- Сделайте двойной щелчок мышью по накопителю, из которого поток вытекает (**PotentialAdopters**), а затем щелкните по тому накопителю, в который он втекает (**Adopters**).
- AnyLogic создаст новый поток и сделает его исходящим потоком для накопителя **PotentialAdopters** и входящим — для **Adopters**. Поток отображается в виде стрелки со значком вентиля посередине. Стрелка показывает направление потока – в данном случае поток будет уменьшать значение накопителя **PotentialAdopters** и увеличивать значение **Adopters**.

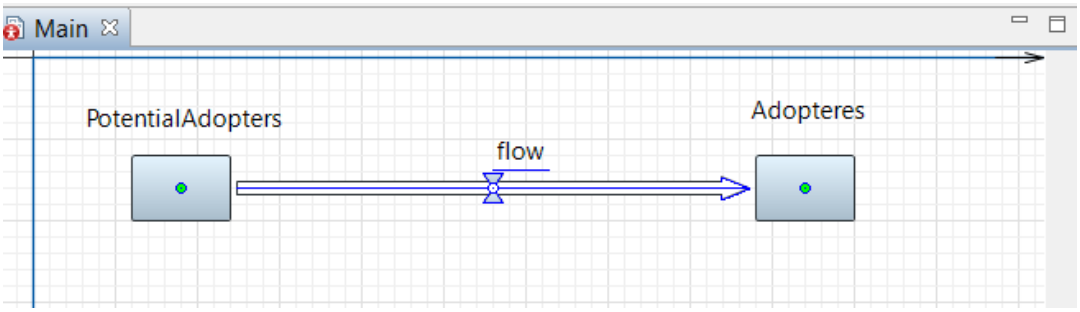


Рис. 14. Поток между накопителями **PotentialAdopters» и «Adopters**

• Выделите стрелку созданного потока в графическом редакторе и измените имя потока на **AdoptionRate**.

В результате диаграмма потоков и накопителей должна будет выглядеть следующим образом:

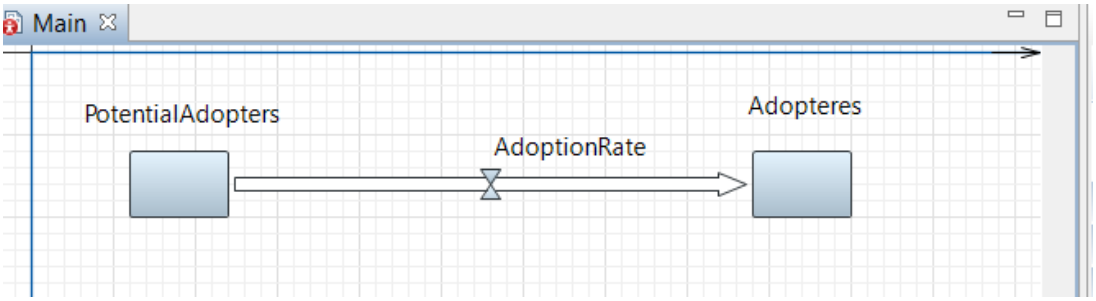


Рис. 15. Поток **AdoptionRate** между накопителями

Можете теперь взглянуть на свойства накопителей. Формулы накопителей должны выглядеть следующим образом:

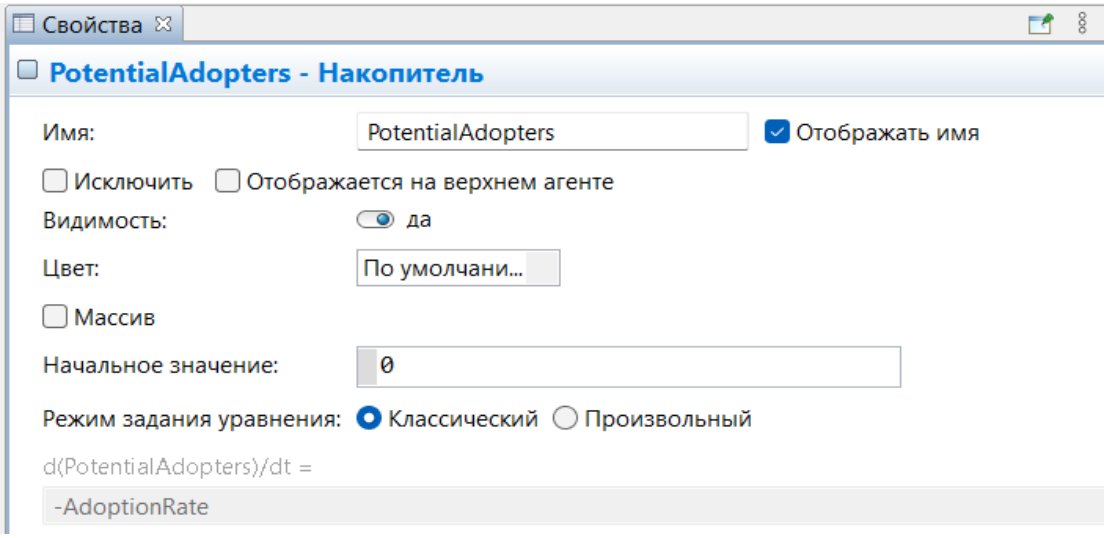


Рис. 16. Свойства накопителя **PotentialAdopters**

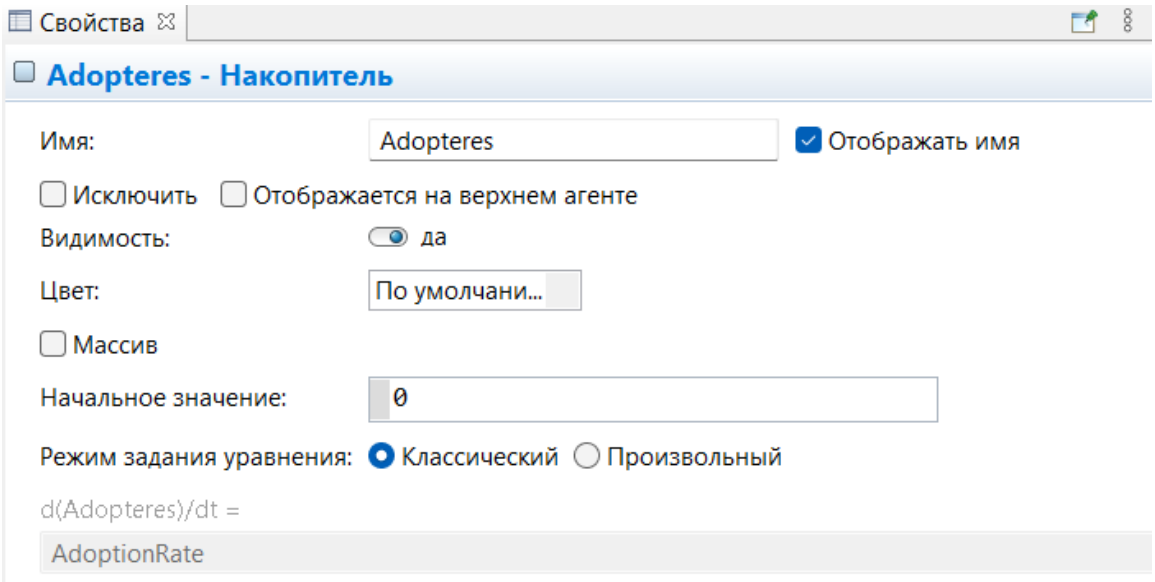


Рис. 17. Свойства накопителя **Adopters**

Эти формулы были автоматически заданы при добавлении потока. Значения входящих потоков, то есть потоков, которые увеличивают значение накопителя, прибавляются, а значения исходящих потоков, уменьшающих значение накопителя, вычитаются из текущего значения накопителя. Выражения, согласно которым будут вычисляться значения потоков, зададим позднее.

Создайте константу, задающую общую численность населения.

- Перетащите элемент **Параметр** из палитры **Системная динамика** на диаграмму типа агентов.
- В панели **Свойства** задайте свойства этого параметра.
- Измените имя параметра. Введите **TotalPopulation** в поле **Имя**.
- В поле **Значение по умолчанию** введите 100000. Пусть общая численность населения в нашей модели будет именно такой.

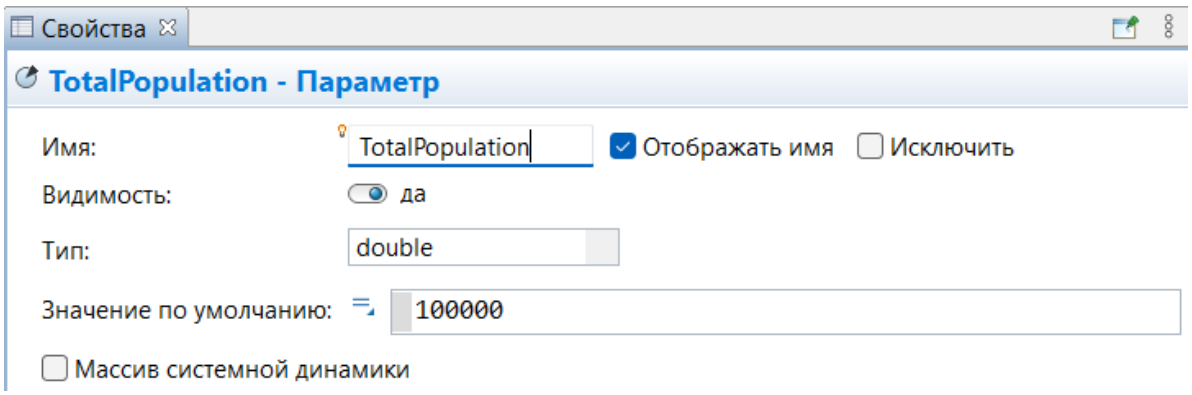


Рис. 18. Свойства параметра **TotalPopulation**

Вы можете задать краткое описание параметра в секции свойств **Описание**. Введите текст, который поможет объяснить смысл константы тем, кто в дальнейшем будет работать с этой моделью. Частота, с которой потенциальные потребители общаются с потребителями, в нашей модели будет постоянной величиной. Поэтому мы зададим частоту контактов константой.

Создайте константу с именем **ContactRate.**

- Предположим, что каждый человек в среднем встречается со 100 людьми в год. Введите 100 в поле **Значение по умолчанию**.

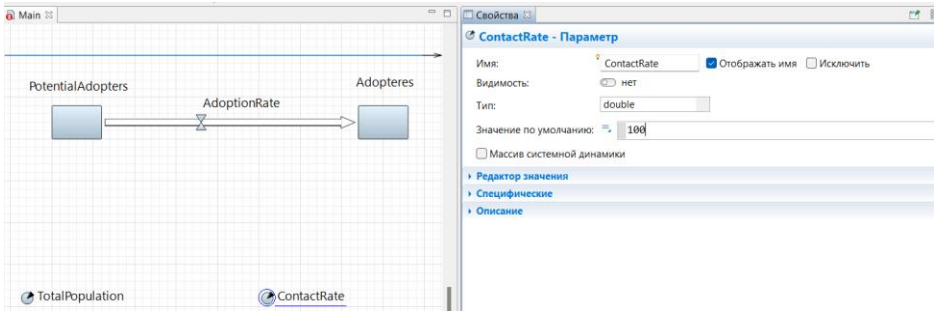


Рис. 19. Свойства параметра **ContactRate**

В этой модели интенсивность рекламы и вероятность того, что продукт будет приобретен под ее влиянием, полагаются постоянными. Поэтому мы зададим эффективность рекламы константой. Эффективность рекламы определяет, какая доля людей купит продукт под ее влиянием.

Создайте константу **AdEffectiveness**, задающую эффективность рекламы.

- Создайте еще одну константу и назовите ее **AdEffectiveness**.
- Задайте значение по умолчанию 0,01.

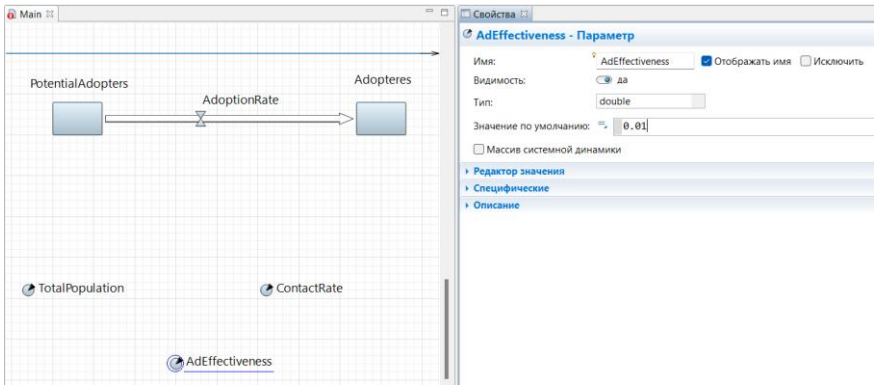


Рис. 20. Свойства параметра **AdEffectiveness**

Задайте константой силу убеждения владельцев продукта, определяющую ту долю контактов, которая приводит к продажам продукта.

• Создайте еще одну константу и назовите ее **AdoptionFraction** со значением 0,015.

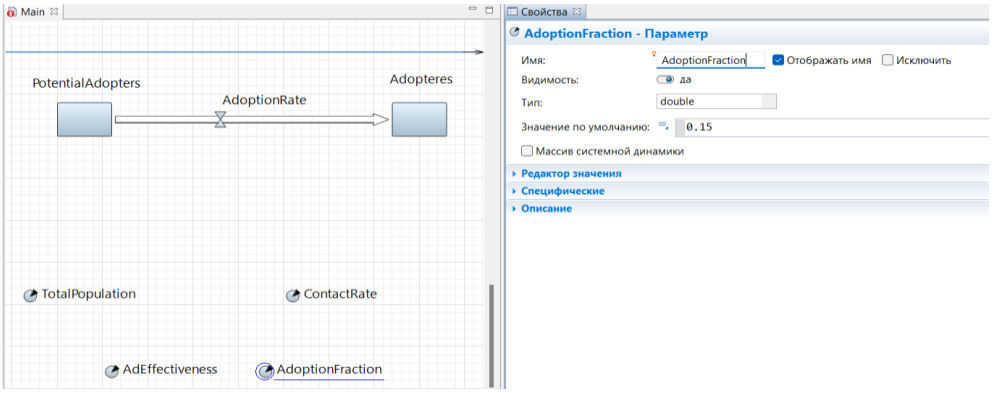


Рис. 21. Свойства параметра **AdoptionFraction**

Теперь мы можем задать начальные значения накопителей.

Мы хотим задать общую численность людей в нашей модели (заданную параметром **TotalPopulation**) в качестве начального значения накопителя **PotentialAdopters**.

Когда вы указываете какой-либо элемент в выражении начального значения накопителя, то должны вначале соединить этот элемент с накопителем с помощью связи. **Связь** позволяет явно задавать существующие зависимости между элементами диаграммы потоков и накопителей.

Чтобы задать связь сделайте двойной щелчок по элементу **Связь** палитры **Системная динамика**. Значок элемента при этом должен измениться на следующий: .

• Сразу после этого щелкните в графическом редакторе по элементу, который упоминается в выражении начального значения **(TotalPopulation)**.

• Затем щелкните по накопителю **PotentialAdopters**, к которому должна следовать создаваемая связь зависимости.

Задайте начальное количество потенциальных потребителей продукта.

• Выделите накопитель **PotentialAdopters** щелчком мыши. В панели **Свойства** введите **TotalPopulation** в поле **Начальное значение**. Чтобы не печатать полностью имена функций и переменных в формулах, можете воспользоваться **Мастером** подстановки кода. Чтобы открыть **Мастер**, щелкните мышью в том месте

поля (в нашем случае – поля **Начальное значение**, куда вы хотите поместить **имя**), а затем нажмите **Ctrl + пробел**.

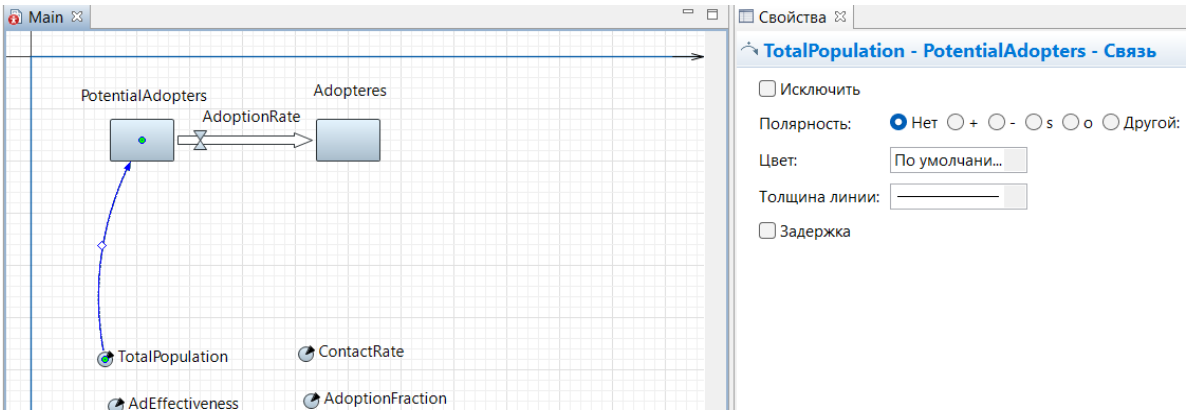


Рис. 22. Создание связи

- Появится окно **Мастера подстановки кода**, перечисляющего переменные модели и функции, доступные в текущем контексте. Прокрутите список к имени, которое вы хотите вставить, или введите первые буквы имени, пока оно не будет выделено в списке. Двойным щелчком мыши по имени добавьте его в поле формулы.

В результате в поле **Начальное значение** должно быть добавлено имя параметра **TotalPopulation**, значение которого и будет определять начальное значение этого накопителя (начальную численность потенциальных покупателей продукта).

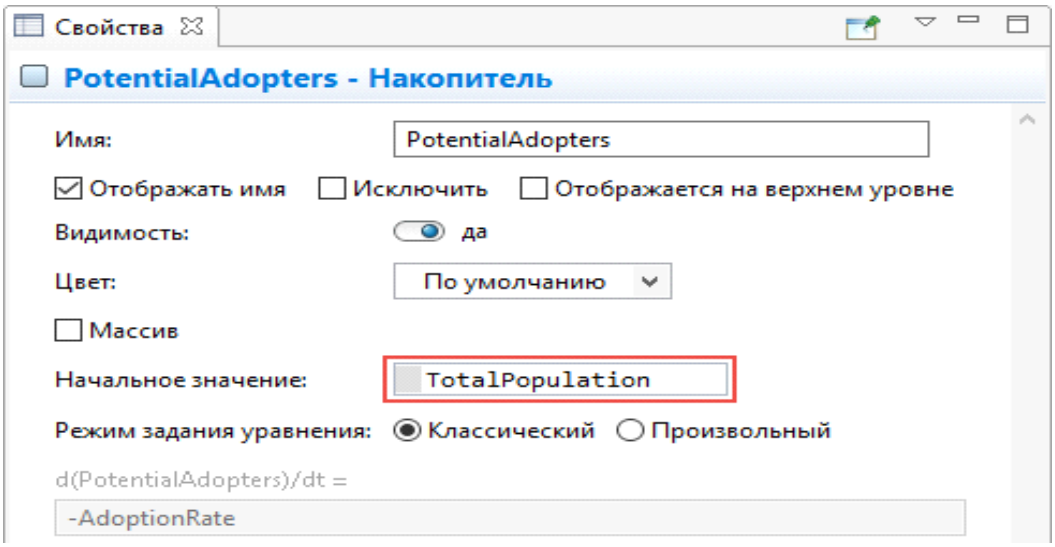


Рис. 23. Накопитель **PotentialAdopters**

Начальное значение накопителя **Adopters**, моделирующего потребителей продукта, задавать не нужно, поскольку изначально число потребителей равно нулю, а накопитель по умолчанию и так инициализируется нулем.

Теперь мы закончили задание накопителей. Нам осталось добавить на диаграмму потоков и накопителей динамические переменные – и модель будет готова.

Создайте динамическую переменную **AdoptionFromAd.** Перетащите элемент **Динамическая переменная** из палитры **Системная динамика** на диаграмму типа агентов.

- В панели **Свойства** введите новое **Имя** переменной: **AdoptionFromAd**.

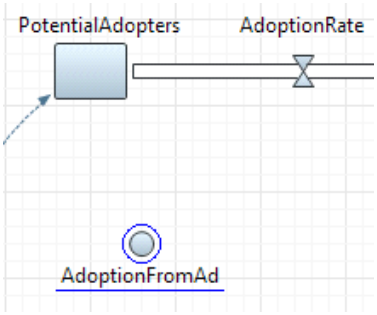


Рис. 24. Переменная **AdoptionFromAd**

Теперь мы хотим задать формулу для этой динамической переменной. Влияние рекламы моделируется следующим образом: некий постоянный процент потенциальных клиентов **AdEffectiveness** всё время становятся клиентами. Их доля в **AdoptionRate** равна, соответственно, **PotentialAdopters** * **AdEffectiveness**.

Опять, как и в случае составления выражения начального значения накопителя, когда какая-либо переменная задействована в формуле динамической переменной или потока, между этими переменными должна существовать связь. Добавьте связи от двух переменных к зависимой от них **AdoptionFromAd.**

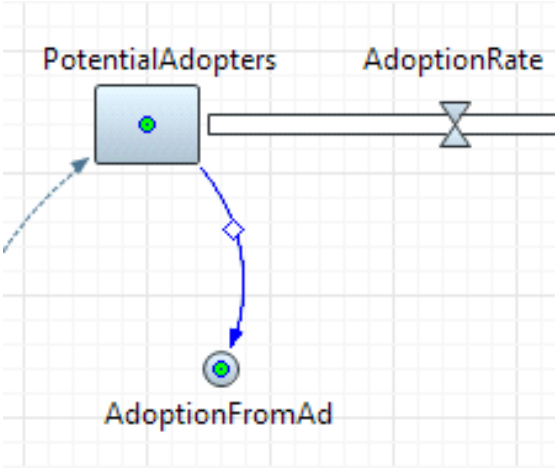


Рис. 25. Связь между **PotentialAdopters** и **AdoptionFromAd**

• Добавьте связи, ведущие от **AdEffectiveness** и **PotentialAdopters** к **AdoptionFromAd**.

Пусть связь выглядит как на приведенном ниже рисунке:

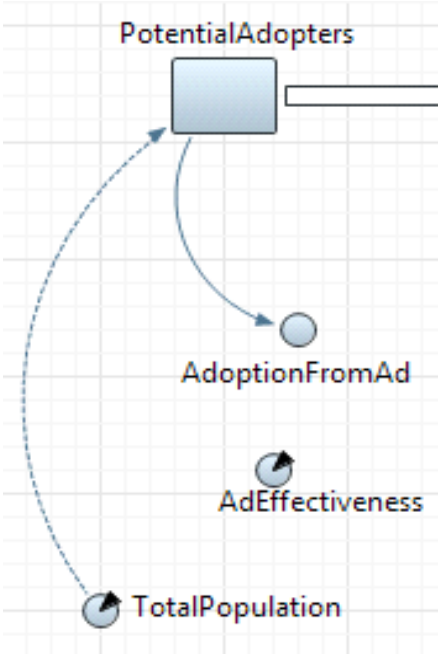


Рис. 26. Вид связей

Вы могли заметить, что эта связь выглядит немного иначе, чем та, что ведет от **TotalPopulation** к **PotentialAdopters**. Связи с переменными, упоминающиеся в начальных значениях накопителей, рисуются пунктирными линиями, в то время как все остальные рисуются сплошными линиями.

• Добавьте еще одну связь, ведущую от **AdEffectiveness** к **AdoptionFromAd**.

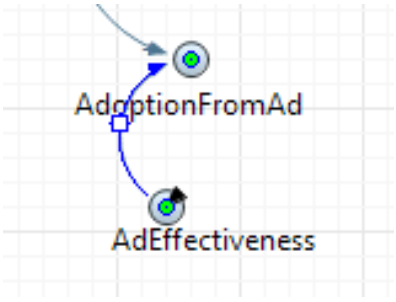


Рис. 27. Связь между **AdEffectiveness к AdoptionFromAd**

• Задайте формулу, согласно которой будет вычисляться значение переменной. В свойствах переменной **AdoptionFromAd** в поле **AdoptionFromAd =** введите: **AdEffectiveness * PotentialAdopters** (вы можете воспользоваться **Мастером** подстановки кода или копировать/вставить).

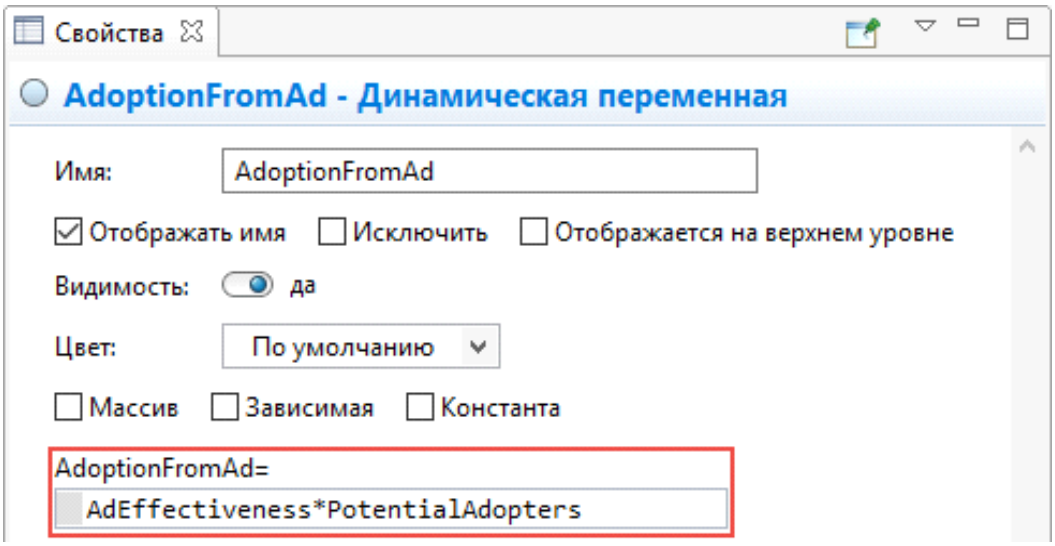


Рис. 28. Свойства переменной **AdoptionFromAd**

Для тех, кто не знаком с классической моделью «Диффузии по Бассу», давайте попробуем самостоятельно составить формулу интенсивности продаж продукта под влиянием устного общения потребителей продукта с теми, кто данный продукт еще не приобрел.

Мы делаем предположение, что в нашей модели человек может общаться с любым другим человеком. Количество контактов человека в единицу времени (а под единицей времени в нашей модели подразумевается год) задается параметром **ContactRate**. Запишем **ContactRate** в качестве первого сомножителя нашей формулы.

Количество людей, которые владеют продуктом и могут убеждать остальных приобрести его, в нашей модели в каждый момент времени будет определяться значением накопителя **Adopters**, и поскольку каждый потребитель будет общаться в единицу времени с **ContactRate** людей, то количество контактов в единицу времени у всех потребителей продукта будет равно **Adopters * ContactRate**.

Теперь нужно учесть тот факт, что в результате общения не все те, кто еще не купил этот продукт, сразу побегут его покупать – если кого-то доводы своего знакомого, успешно пользующегося изучаемым нами продуктом, могут убедить, то кто-то может остаться к ним равнодушным, и своего решения не покупать продукт не изменить. Поэтому мы добавим в нашу формулу еще один сомножитель **AdoptionFraction**, задающий силу убеждения владельцев продукта, определяющую ту долю контактов, которая приводит к продажам продукта. Таким образом, наша формула приобретает вид **Adopters * ContactRate * AdoptionFraction**.

И наконец, нам нужно учесть, что на данный момент наша формула не учитывает того, что владельцы продукта будут общаться как с потенциальными потребителями, так и с теми, кто уже владеет продуктом. И общение с последними ни к каким новым продажам продукта не приведет. Поэтому нам нужно учесть в нашей формуле и вероятность того, что тот, с кем общался потребитель, ещё не владеет интересующим нас продуктом. Эта вероятность задается так: **PotentialAdopters / TotalPopulation**.

В итоге наша формула будет выглядеть следующим образом: **Adopters * ContactRate * AdoptionFraction * PotentialAdopters / TotalPopulation.**

Именно столько потенциальных потребителей будут приобретать продукт в единицу модельного времени под воздействием общения с владельцами этого продукта.

Создайте динамическую переменную. Назовите ее **AdoptionFromWOM.** Задайте для этой переменной следующую формулу:

ContactRate * AdoptionFraction * PotentialAdopters * Adopters / TotalPopulation

Вам может показаться утомительным рисовать связи для задействованных в формулах переменных и параметров. Чтобы облегчить этот процесс, AnyLogic предлагает пользователям механизм быстрого исправления ошибок, связанных с отсутствующими или избыточными связями. Когда вы зададите указанную формулу, то щелкнув в этом поле, вы увидите индикатор ошибки:

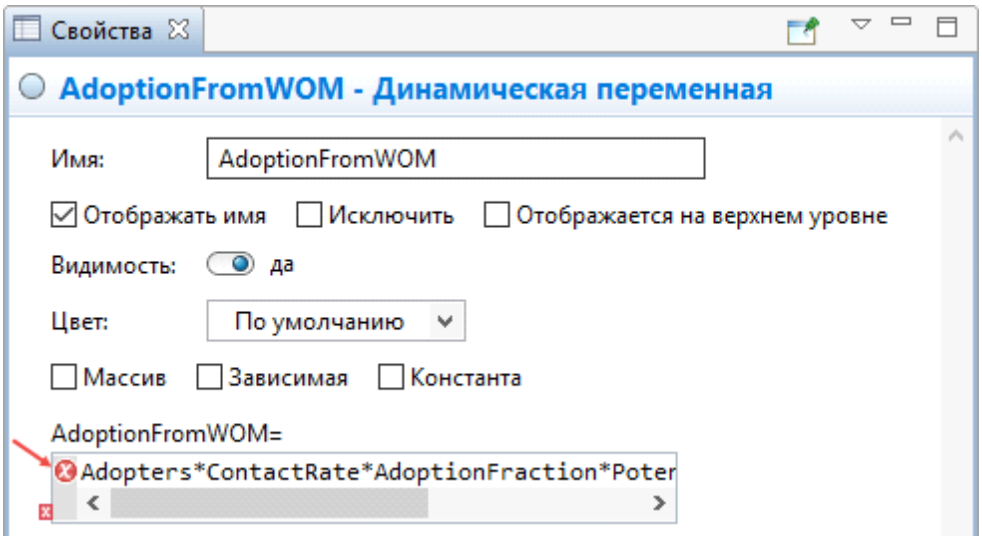


Рис. 29. Свойства переменной **AdoptionFromAd**

Щелкните мышью по индикатору. Вы увидите контекстное меню, состоящее из пунктов **Добавить отсутствующую связь для: ...** Поочередно щелкнув по всем этим пунктам, вы добавите на диаграмму все недостающие связи.

В итоге у вас должна будет получиться диаграмма следующего вида:

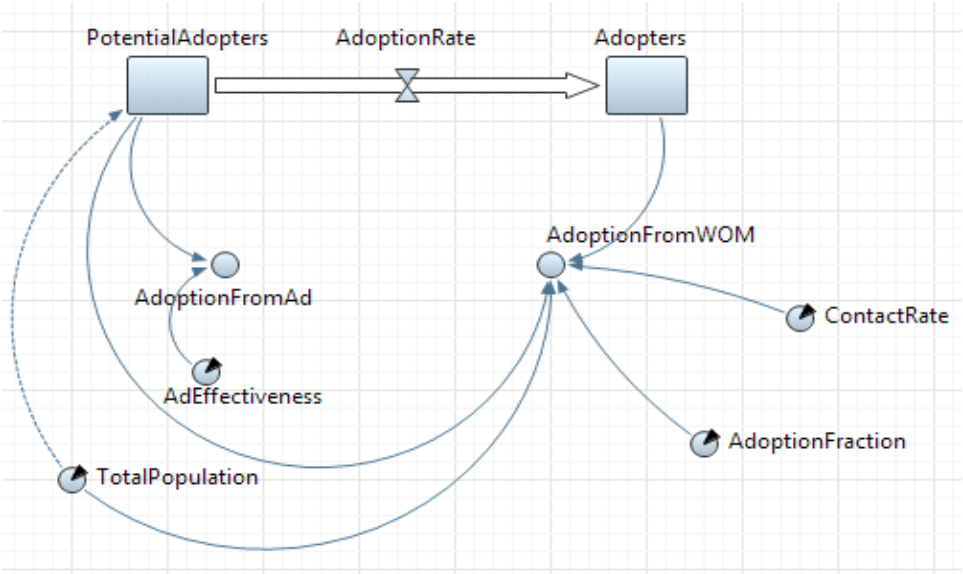


Рис. 30. Диаграмма

Теперь мы можем задать формулу для потока приобретения продукта. Значение потока определяется суммой двух его независимых составляющих – продаж в результате рекламного влияния и продаж под влиянием общения с потребителями продукта.

Задайте формулу потока. Выделите поток **AdoptionRate** щелчком мыши. Перейдите в панель **Свойства**.

Введите правую часть формулы, по которой будет вычисляться значение потока, в поле **AdoptionRate=: AdoptionFromAd** + **AdoptionFromWOM.**

Добавьте соответствующие связи от этих переменных к потоку **AdoptionRate**.

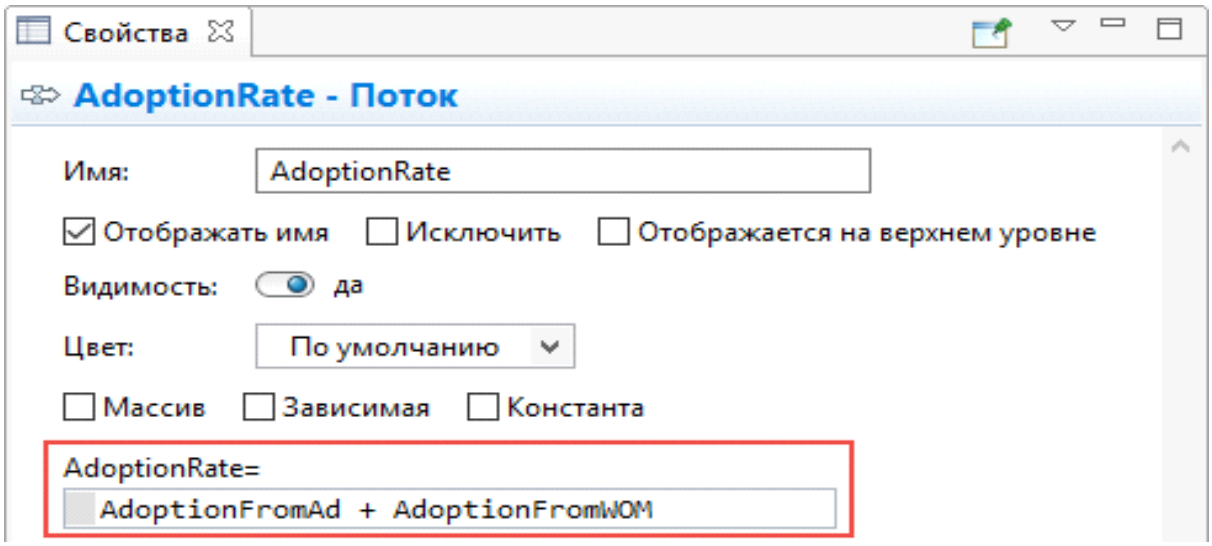


Рис. 31. Свойства потока **AdoptionRate**

Теперь мы закончили создание нашей модели. Диаграмма накопителей и потоков должна выглядеть как на приведенном ниже рисунке:

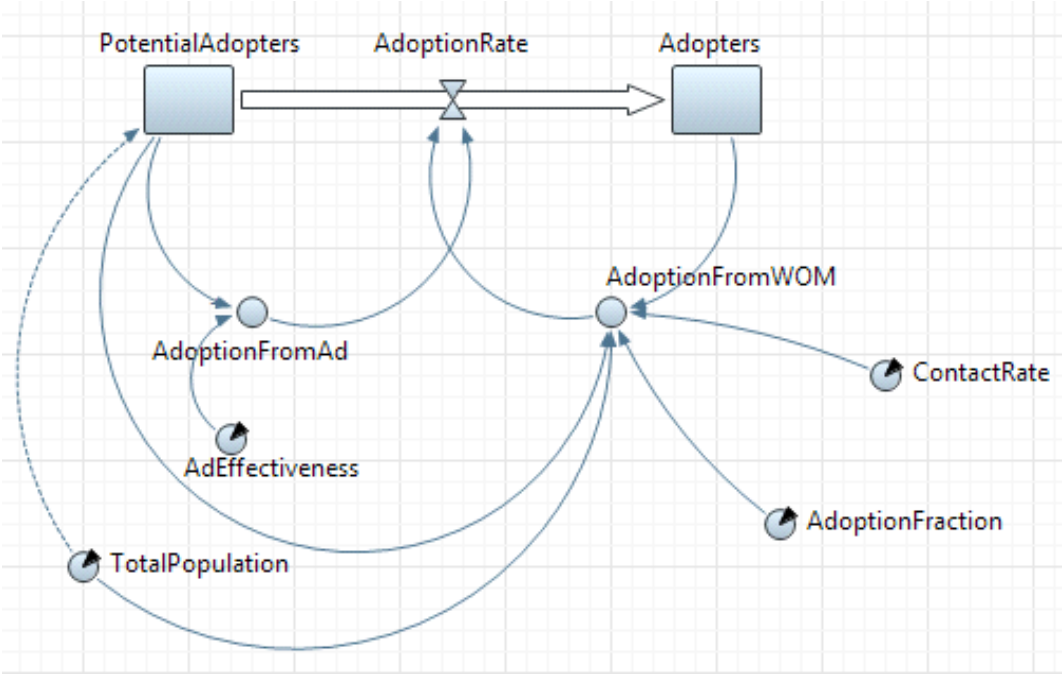


Рис. 32. Диаграмма

Связи имеют полярность, положительную или отрицательную:

• Положительная связь означает, что два элемента системной динамики изменяют свои значения в одном направлении, т. е., если значение элемента, из которого направлена связь, уменьшается, значение другого элемента уменьшается тоже. Аналогично, если увеличивается значение одного элемента, то и значение зависимого от него элемента увеличивается тоже.

• Отрицательная связь означает, что два элемента системной динамики изменяют свои значения в противоположных направлениях, т. е. если значение элемента, из которого направлена связь, уменьшается, то значение другого элемента увеличивается, и наоборот.

Вы можете добавить рядом со связями метки, которые будут обозначать полярность этих связей. Обычно полярность обозначается с помощью символов /- рядом со стрелкой связи. Таким образом, вы можете показать, как зависимая переменная изменяет свое значение при изменении значения независимой переменной.

Проставьте полярности у связей.

В нашей модели все связи, за исключением той, что ведет от **TotalPopulation** к **AdoptionFromWOM**, имеют положительную полярность.

- Чтобы отобразить у связи значок полярности, выделите связь и выберите нужный символ (+ или -) из группы кнопок **Полярность** в панели свойств связи.
- Здесь же при желании можно изменить и цвет линии связи, а также ее толщину.

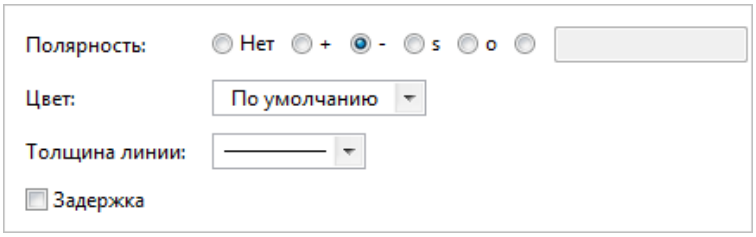


Рис. 33. Установка полярности

Можно увидеть, что наша модель содержит два цикла с обратной связью: один компенсирующий и один усиливающий.

- Компенсирующий цикл с обратной связью воздействует на поток приобретения продукта, вызванный рекламой. Поток приобретения продукта сокращает число потенциальных потребителей, что в свою очередь приводит к снижению интенсивности приобретения продукта.

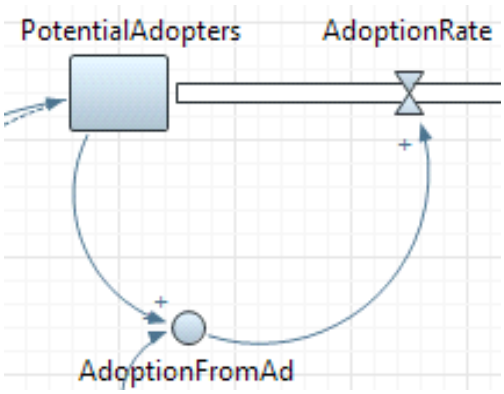


Рис. 34. Компенсирующий цикл

- Усиливающий цикл с обратной связью воздействует на поток приобретения продукта, вызванный общением с потребителями продукта. Поток приобретения продукта увеличивает численность потребителей продукта, что приводит к росту интенсивности приобретения продукта под влиянием общения с потребителями продукта, и, следовательно, к росту интенсивности приобретения продукта.

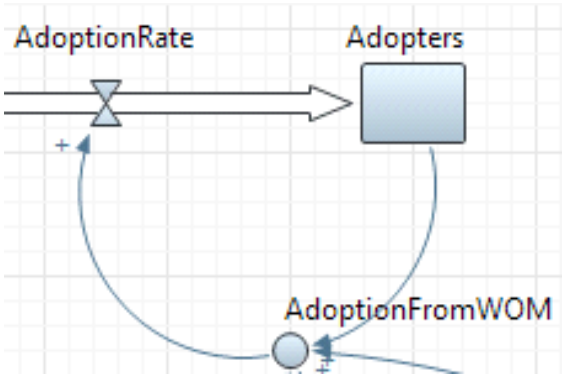


Рис. 35. Усиливающий цикл

Добавьте идентификатор цикла, вызывающего насыщение рынка:

- Перетащите элемент **Цикл** ® из палитры **Системная динамика** на графическую диаграмму, как показано на рисунке ниже:

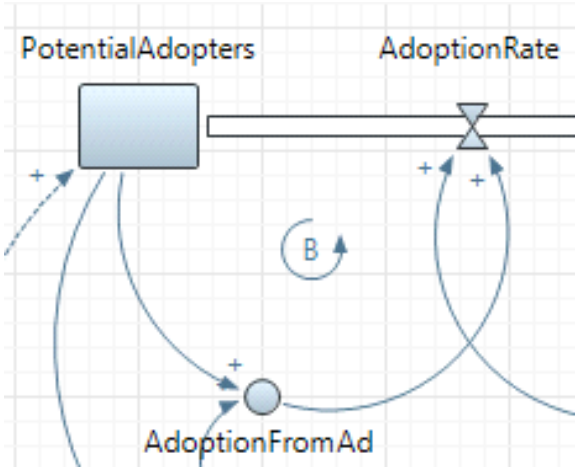


Рис. 36. Компенсирующий цикл

- Перейдите в панель **Свойства**, чтобы изменить свойства цикла.
- Задайте **Направление** цикла – этот цикл направлен **Против часовой стрелки**.
- В поле **Текст** введите краткое описание этого цикла, объясняющее его смысл: **Market Saturation** (или Насыщение рынка). Этот текст будет показан на презентации.
- Из группы кнопок **Тип** выберите символ, который будет отображаться для данного цикла. Выберите символ ***B*** (обозначающий **Balancing**, то есть **компенсирующий цикл**).

Чтобы определить, является ли цикл усиливающим или уравновешивающим, вы можете начать с предположения, что, например, значение переменной ***A*** увеличивается, и проследить за изменением значений входящих в цикл переменных.

Цикл является усиливающим, если после прохождения по циклу вы видите тот же результат, что был допущен при начальном предположении (в нашем случае – увеличение значения).

- Цикл является уравновешивающим или компенсирующим, если результат противоречит начальному предположению.

Добавьте идентификатор для цикла, задающего общение людей друг с другом.

- Добавьте еще один идентификатор цикла, как показано на рисунке ниже:

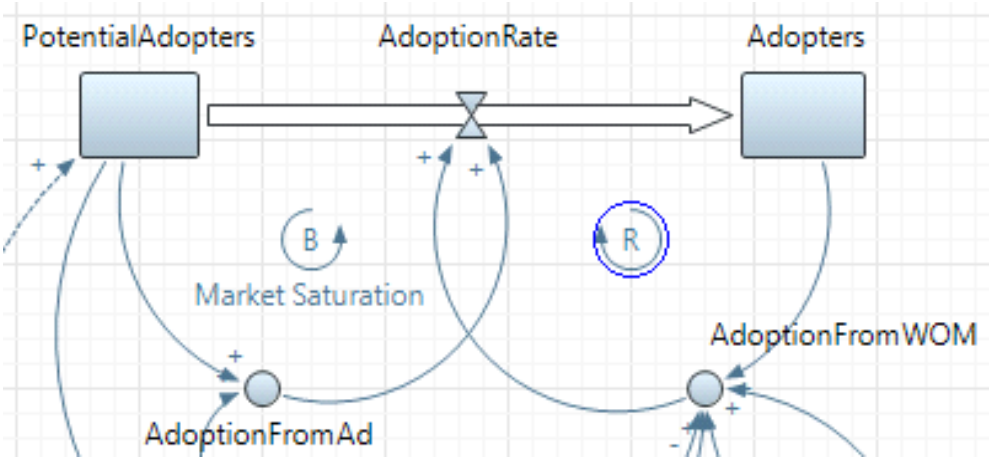


Рис. 37. Усиливающий цикл

- Этот цикл соответствует общению людей друг с другом. Он является усиливающим, поэтому выберите для него символ **R** (обозначающий **усиливающий**, **Reinforcing** цикл).
- Задайте **Word of Mouth** (или **Устное общение**) в качестве текста.
- Задайте **Направление** цикла – этот цикл направлен **По часовой стрелке**.

В итоге ваша диаграмма должна выглядеть следующим образом:

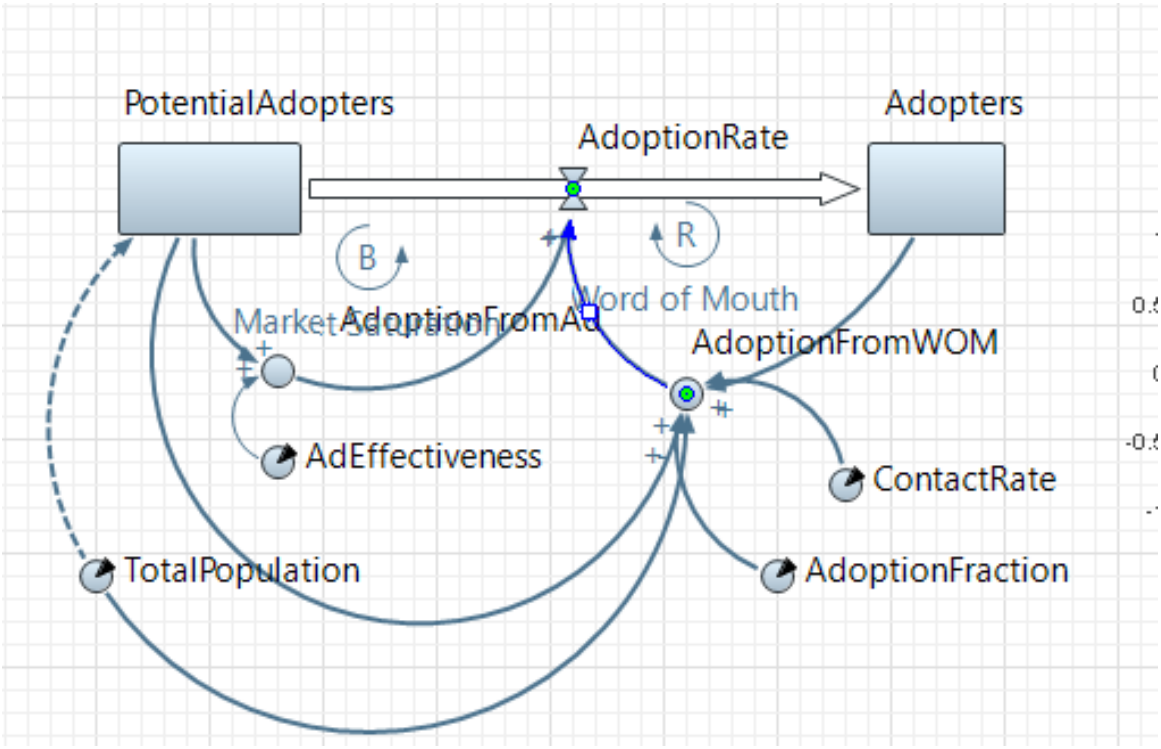


Рис. 38. Итоговый вид диаграммы

Вы можете сконфигурировать выполнение модели в соответствии с вашими требованиями. Модель выполняется в соответствии с набором установок, задаваемым специальным элементом модели — **экспериментом**. Вы можете создать несколько экспериментов с различными установками и изменять рабочую конфигурацию модели, просто запуская тот или иной эксперимент модели.

В панели **Проекты** эксперименты отображаются в нижней части дерева модели. Один эксперимент, названный **Simulation**, создается по умолчанию.

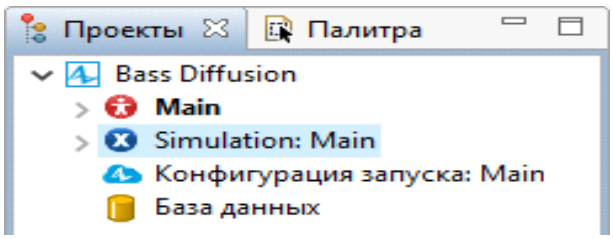


Рис. 39. Выбор типа эксперимента

Это **простой эксперимент**, позволяющий запускать модель с заданными значениями параметров, поддерживающий режимы виртуального и реального времени, анимацию и отладку модели.

Существуют также и другие типы экспериментов, которые используются в тех случаях, когда параметры модели играют существенную роль, и требуется проанализировать, как они влияют на поведение модели, или когда нужно найти оптимальные значения параметров модели.

Если мы сейчас запустим модель, то она будет моделироваться бесконечно, пока мы сами не остановим ее выполнение. Поскольку мы хотим наблюдать поведение модели только тогда, когда происходит процесс распространения продукта, нам нужно остановить модель, когда система придет в точку равновесия. Процесс распространения продукта в этой модели длится примерно 8 лет.

Задайте остановку модели по прошествии 8 единиц модельного времени:

- В панели **Проекты** выделите эксперимент **Simulation: Main** щелчком мыши.

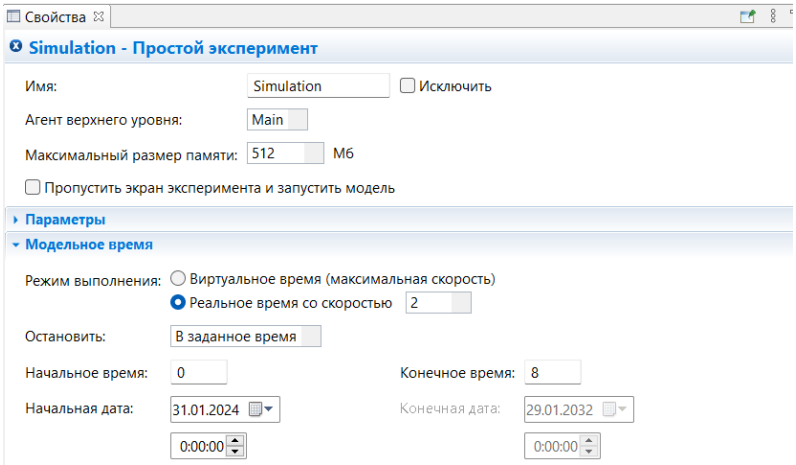


Рис. 40. Установка свойств простого эксперимента

В секции **Модельное время** панели **Свойства** выберите **В заданное время** из выпадающего списка **Остановить**. В расположенном ниже поле введите 8. Модель остановится после того, как истекут 8 единиц модельного времени. Перед тем, как запустить модель, выберем режим ее выполнения. Модель AnyLogic может выполняться либо в режиме виртуального, либо в режиме реального времени. В **режиме виртуального времени** модель выполняется без привязки к физическому времени — она просто выполняется настолько быстро, насколько это возможно. Этот режим **лучше всего подходит в том случае, когда требуется моделировать работу системы в течение достаточно длительного периода времени.** В **режиме реального времени** задается связь модельного времени с физическим, то есть задается количество единиц модельного времени, выполняемых

в одну секунду. Это часто требуется, когда вы хотите, чтобы анимация модели отображалась с той же скоростью, что и в реальной жизни.

Задайте выполнение модели в режиме реального времени:

• В панели **Проекты** выделите эксперимент **Simulation: Main** щелчком мыши. Перейдите в секцию **Модельное время** и выберите опцию **Реальное время со скоростью**.

• Задайте скорость выполнения модели, т.е., сколько единиц модельного времени будет соответствовать одной секунде реального времени. Задайте число 2 (рис. 40).

Теперь изменим свойство, не эксперимента, а модели.

В дальнейшем по нажатию на кнопку **Запустить** (или по нажатию F5) будет запускаться тот эксперимент, который запускался вами в последний раз. Чтобы выбрать какой-то другой эксперимент, вам будет нужно щелкнуть мышью по стрелке, находящейся в правой части кнопки **Запустить**, и выбрать нужный вам эксперимент из открывшегося списка (или щелкнуть правой кнопкой мыши по этому эксперименту в панели **Проекты** и выбрать **Запустить** из контекстного меню).

Запустив модель, вы увидите ее окно. Щелкните по кнопке **Запустить** . Вы увидите диаграмму потоков и накопителей. Рядом с каждым элементом будет отображаться его текущее значение.

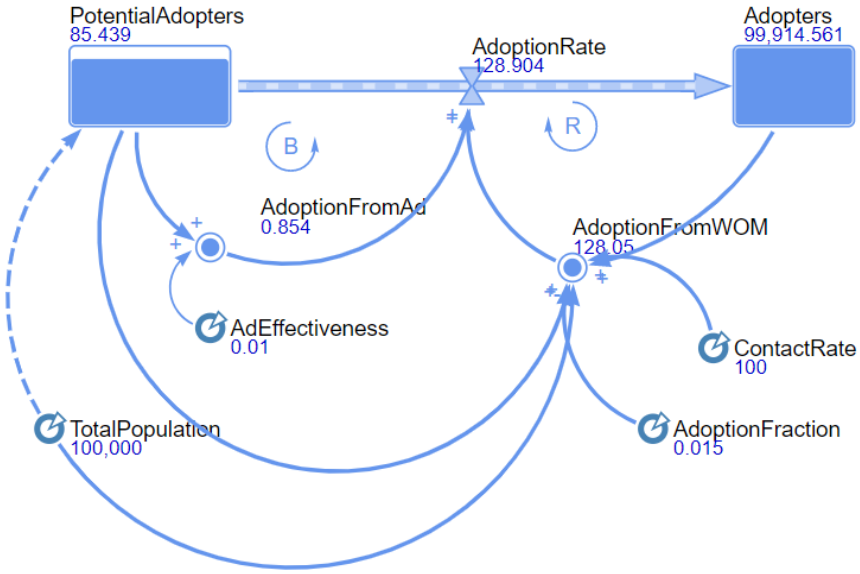


Рис. 41. Результат работы модели

Вы можете изменить скорость выполнения модели с помощью кнопок управления окна модели **Замедлить** или **Ускорить**

AnyLogic поддерживает различные инструменты для сбора, отображения и анализа данных во время выполнения модели.

Исследуйте динамику обеих составляющих потока продаж.

Для визуализации и анализа данных используются диаграммы и объекты сбора данных, расположенные на палитре **Статистика.**

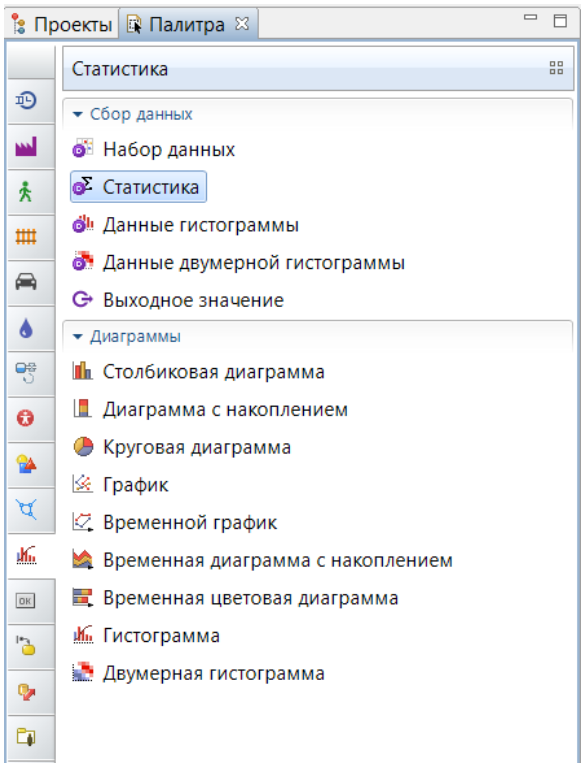


Рис. 42. Палитра **Статистика**

С помощью этих элементов вы можете добавлять на презентацию любые графики, диаграммы и гистограммы и вести статистический анализ собранных данных.

Добавим диаграммы, с помощью которых мы будем изучать, как изменяются со временем численности потребителей и потенциальных потребителей продукта, а также как изменяется интенсивность продаж продукта.

Добавьте график, отображающий динамику изменения численностей потребителей и потенциальных потребителей продукта:

- Перетащите элемент **Временной график** из палитры **Статистика** на диаграмму агента **Main**.
- Перейдите в панель **Свойства**. Добавьте элементы данных, историю изменения значений которых вы хотите отображать на этом временном графике, в секции **Данные**.
- Чтобы добавить новую секцию свойств, в которой задаются свойства отображаемого на графике элемента данных, щелкните мышью по кнопке **Добавить** .
- Задайте выражение, результат вычисления которого будет отображаться на графике. Мы хотим отображать численность потенциальных потребителей, поэтому введите в поле **Значение** имя соответствующего накопителя: **PotentialAdopters**.

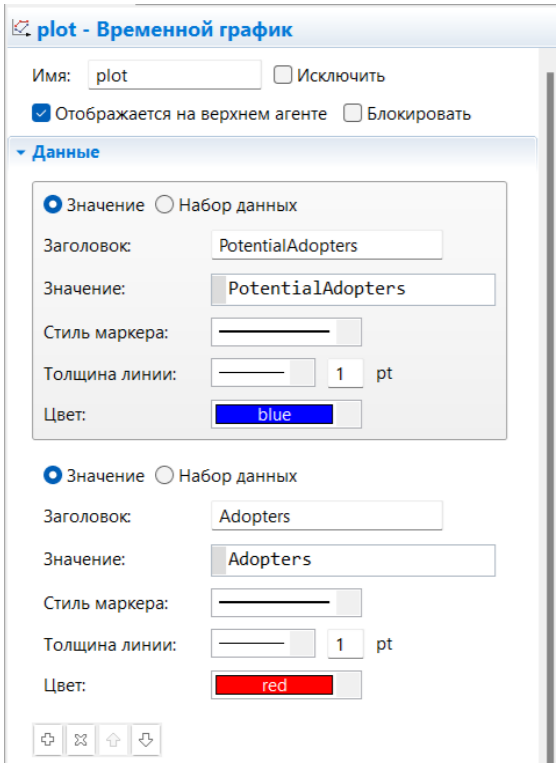


Рис. 43. Свойства **Временного графика**

• В поле **Заголовок** введите **Potential adopters**. Эта строка будет отображаться в легенде диаграммы для этого элемента данных.

Аналогично добавьте на график еще один элемент данных, который будет отображать значение накопителя.

• В поле **Временной диапазон** секции свойств **Масштаб** задайте диапазон временной оси диаграммы (количество единиц модельного времени **(N)**, для которого будут отображаться значения переменной): 8, в качестве единиц времени выберите **годы**. Диаграмма будет отображать график только для заданного временного интервала (равного в нашем случае длительности периода моделирования).

• Измените частоту обновления графика новыми данными в секции свойств **Обновление данных**. Введите 0,1 в поле **Период** для опции, **Обновлять автоматически**, в качестве единиц времени выберите **годы**.

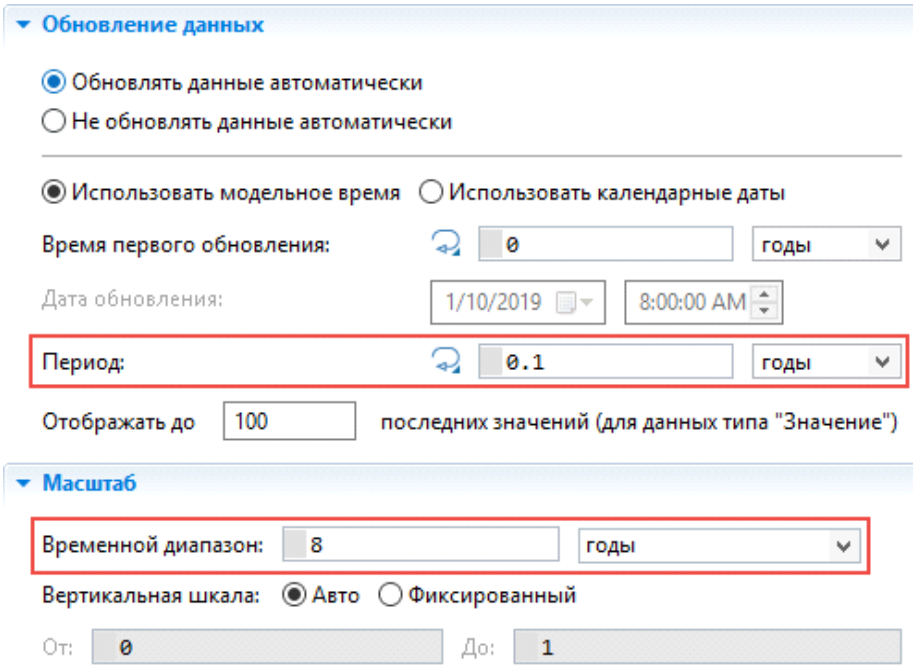


Рис. 44. Обновление данных

Добавьте график, отображающий изменение интенсивности продаж.

• Добавьте на диаграмму еще один временной график. Поместите его под добавленным ранее графиком.

• Добавьте на график новый элемент данных (в качестве **Значения** в этом случае должно быть задано имя потока **AdoptionRate**).

• Измените другие свойства графика.

Теперь вы можете запустить модель и изучить динамику изменения численностей потребителей и потенциальных потребителей продукта. Вы увидите классические для рассматриваемого примера системной динамики кривые S-формы.

С помощью нижнего графика вы можете проследить, как с течением времени будет изменяться интенсивность продаж продукта. Если модель была создана правильно, то вы увидите колоколообразную кривую.

Мы закончили создание простейшей модели системной динамики. Эта модель часто используется в классических учебниках по системной динамике. На этом примере мы хотели продемонстрировать, как создаются типовые модели системной динамики в AnyLogic.

ГЛАВА 4. АГЕНТНЫЕ МОДЕЛИ

ANYLOGIC-МОДЕЛЬ ВНУТРИЗАВОДСКОЙ ЛОГИСТИКИ

Задача смоделировать внутризаводскую логистику между складами заготовок, цехом сборки и складом готовой продукции.

У цеха есть свой парк грузовиков, которые доставляют детали для сборки или продукцию на склады. Детали доставляются в цех сборки, если их запас в цехе стал менее 10 штук. Продукция вывозится из цеха раз в час.

Создание агентов модели

Создайте новую модель и назовите ее **Logistic** (рис. 1). Задайте единицы модельного времени – минуты.

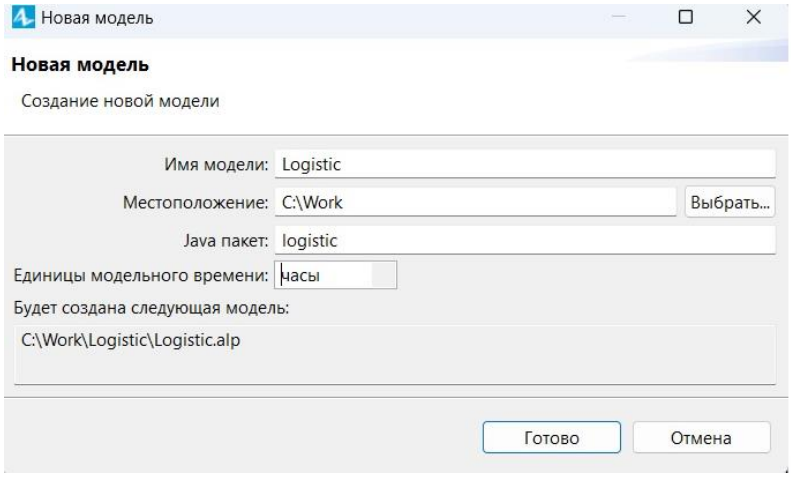


Рис. 1. Создание новой модели

В этой работе будет использован агентный подход. Все инструменты, необходимые для агентного подхода, находятся в библиотеке **Агент** на вкладке **Палитра** (рис. 2).

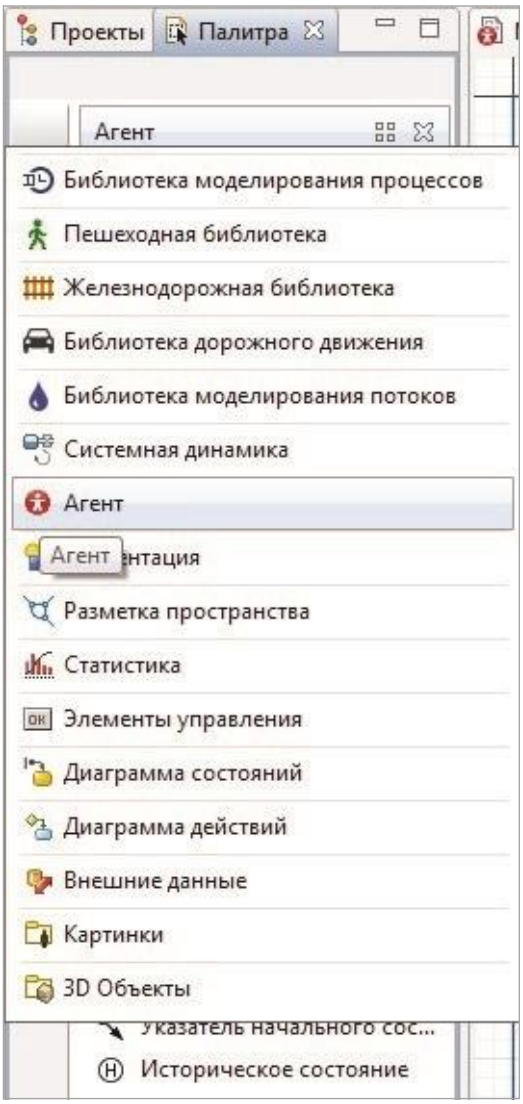


Рис. 2. Библиотека **Агент**

Задание агента Storage

Перейдите в библиотеку и перетащите компонент **Агент** на рабочее поле модели (рис. 3).

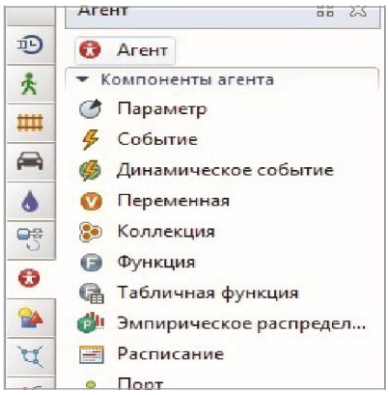


Рис. 3. Компонент **Агент**

Откроется мастер создания агентов. Поскольку все склады и цех будут в единственном экземпляре, то мы будем создавать **Единственного агента**. Выберите **Я создаю единственного агента** на первом шаге **Мастера** (рис. 4).

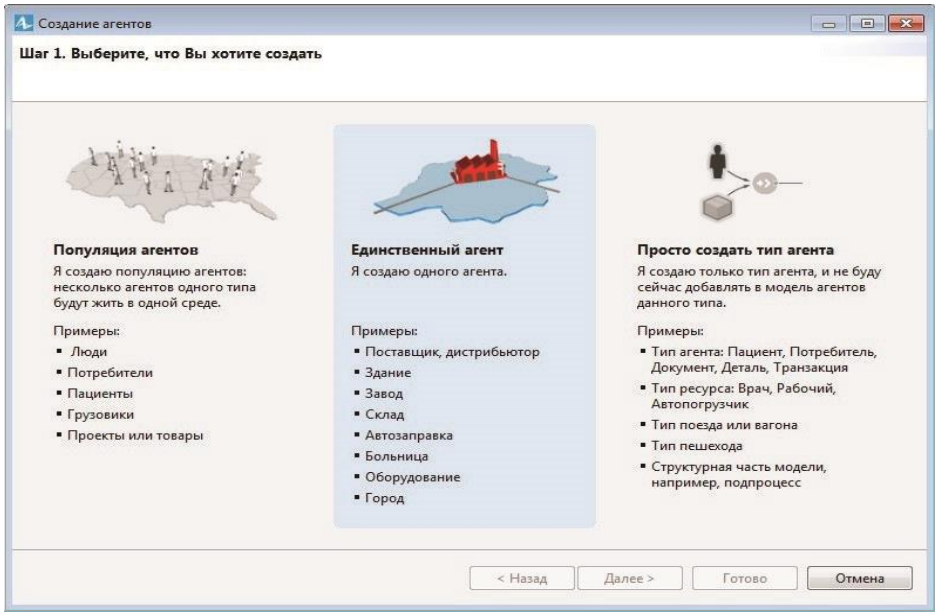


Рис. 4. Первый шаг мастера создания агента **Storage**

На втором шаге мастера задайте имя создаваемого агента **Storage** (рис. 5).

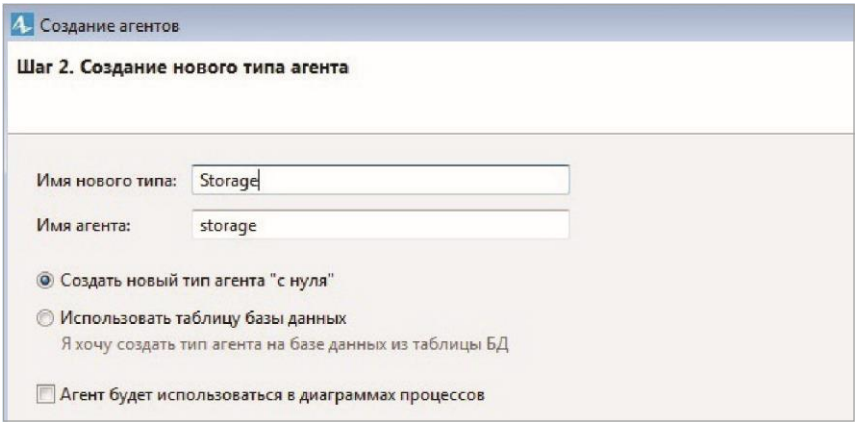


Рис. 5. Второй шаг мастера создания агента **Storage**

На третьем шаге мастера задайте анимацию агента (рис. 6).

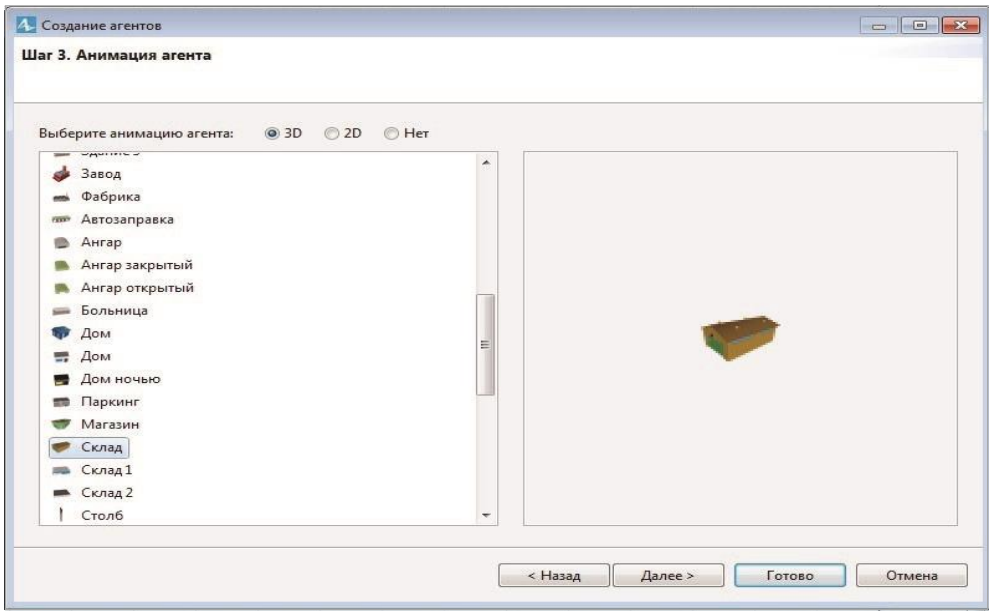


Рис. 6. Третий шаг мастера создания агента **Storage**

На следующем шаге мастера (рис. 7) задайте параметры агента, а именно переменную для хранения количества деталей на складе — **number_of_detal**. Поскольку детали измеряются целыми числами, то тип этой переменной **int**.

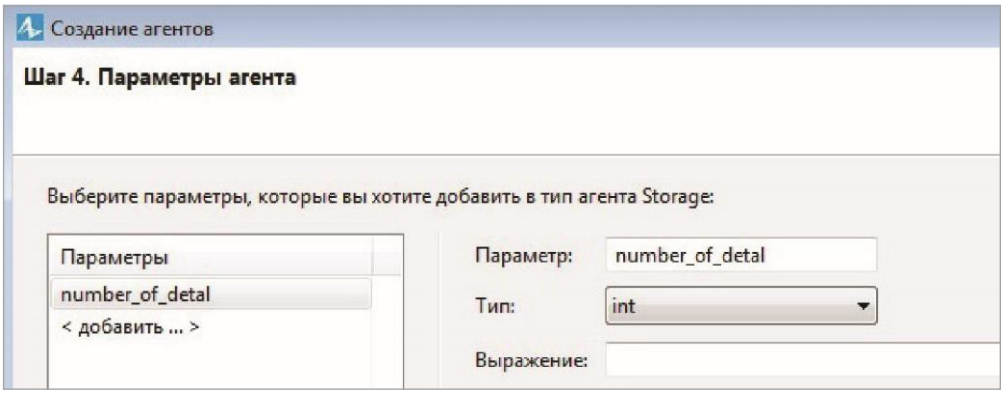


Рис. 7. Четвертый шаг мастера создания агента **Storage**

Нажмите кнопку **Готово** — на рабочем поле появится агент **Storage** (рис. 8).

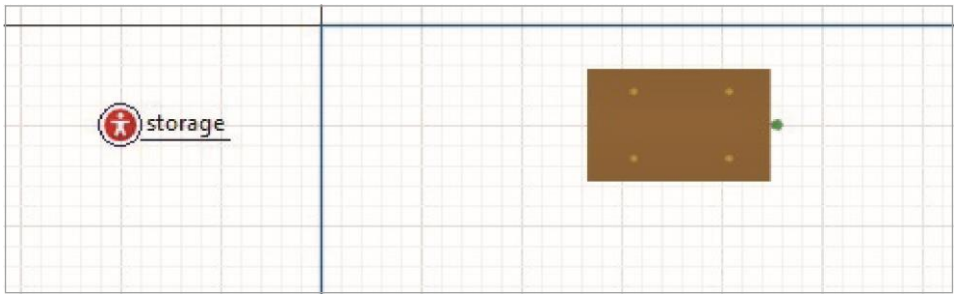


Рис. 8. Агент **Storage** на рабочем поле модели

Задание агента Storage_Production

Теперь создайте агента **Storage_Production**. Для этого перетащите элемент **Агент** на рабочее поле модели и пройдите шаги мастера с 1 по 4, задав имя агента (**Storage_Production**) и его анимацию (**Склад 1**). На втором шаге мастера будет предложен выбор между созданием нового типа агента и использованием существующего. Выберите новый тип агента. В результате на рабочем поле будут два агента **Storage** и **Storage_ Production** (рис. 9).

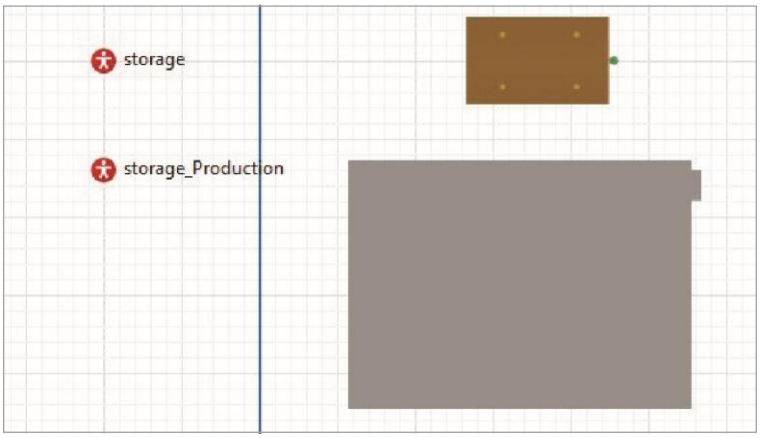


Рис. 9. Агенты **Storage** и **Storage_Production** на рабочем поле модели

Задание популяции агентов Truck

Теперь нужно создать множество агентов для моделирования грузовиков. Перетащите элемент **Агент** и на первом шаге мастера создания агента выберите **Популяция агентов** (рис. 10).

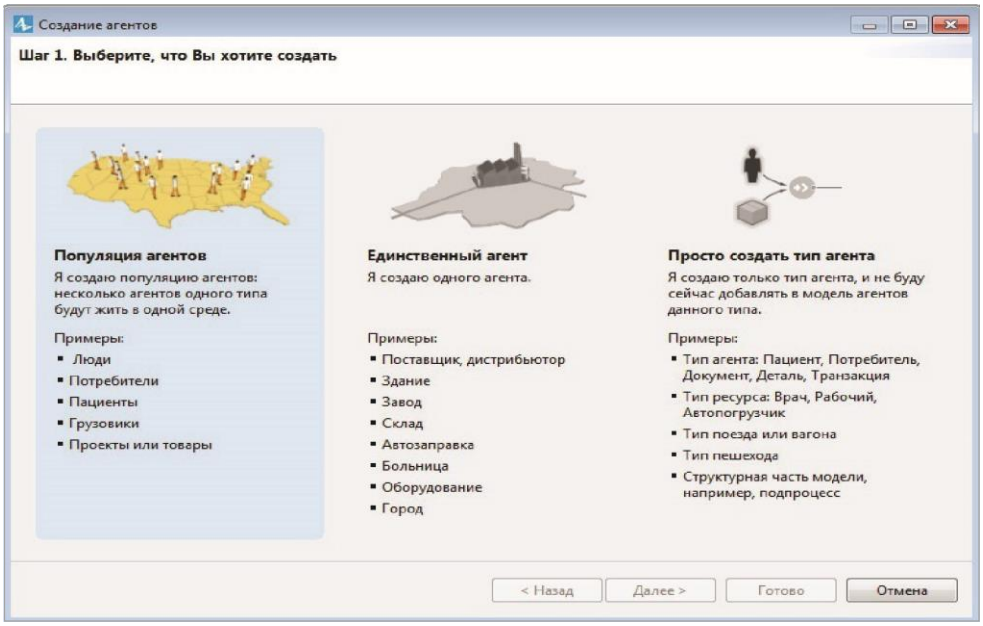


Рис. 10. Первый шаг мастера создания агентов **Truck**

На втором шаге мастера выберите новый тип агента. На третьем шаге мастера задайте имя агента **Truck**. При этом одновременно будет создана популяция с именем **trucks**. На четвертом шаге выберите анимацию агента (рис. 11).

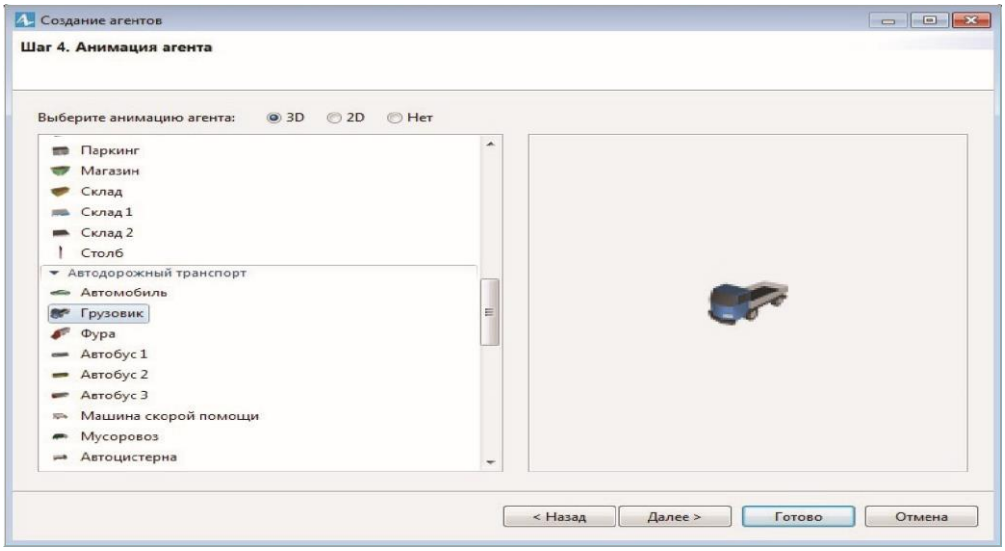


Рис. 11. Задание анимации агента **Truck**

На пятом шаге ничего не нужно задавать. На шестом шаге задайте объем популяции — 5 грузовиков. Нажмите кнопку **Готово**. Должно получиться **три агента** на рабочем поле (рис. 12).

Создание агента Plant

Создайте агент **Plant**. Для этого перетащите элемент **Агент** на рабочее поле модели и на первом шаге **Мастера** выберите **Создать единственного агента**. На втором шаге необходимо создать новый тип агента. На третьем шаге задайте имя агента **Plant**. На четвертом выберите анимацию (рис. 13).

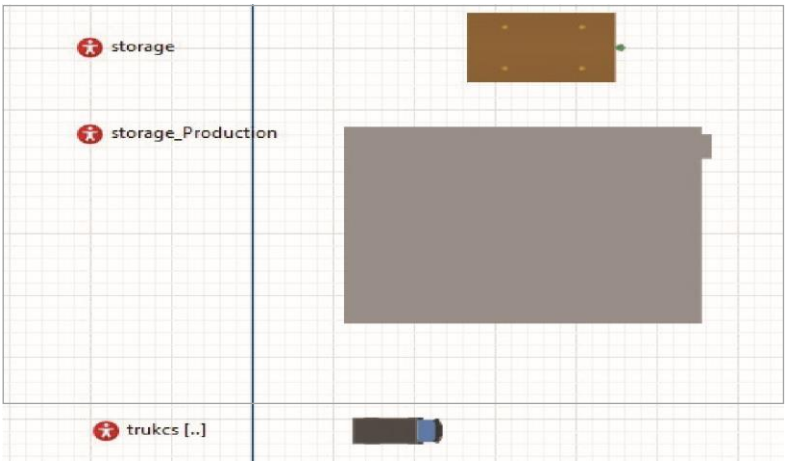


Рис. 12. Агенты на рабочем поле

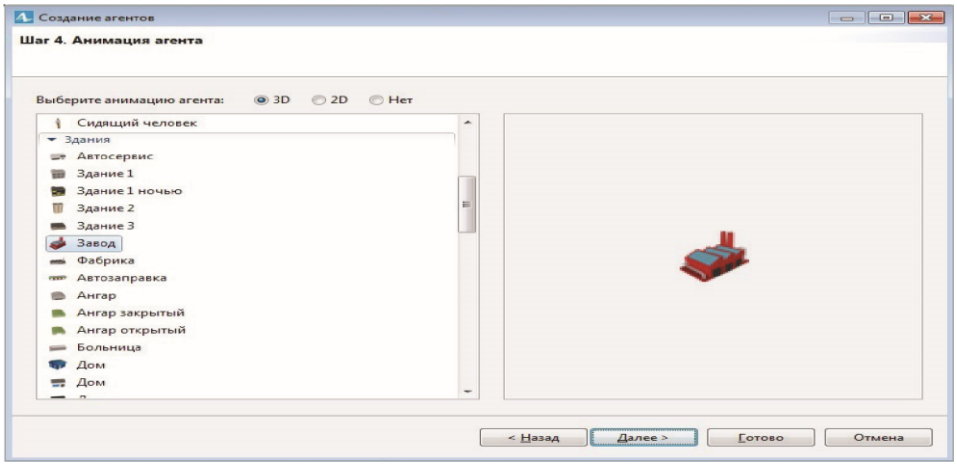


Рис. 13. Задание анимации агента **Plant**

На пятом шаге мастера создайте переменную, в которой будет храниться количество деталей в цехе — **number_of_detal_in_plant**, типа **int**.

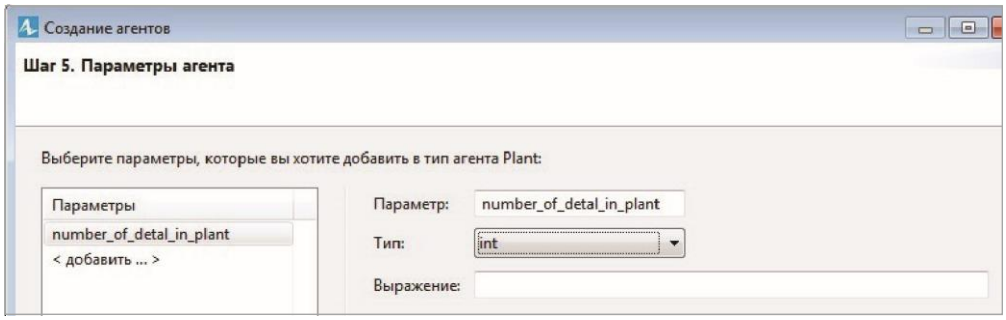


Рис. 14. Задание параметров агента **Plant**

Нажмите кнопку **Готово**. Должно быть так же, как на рис. 15.

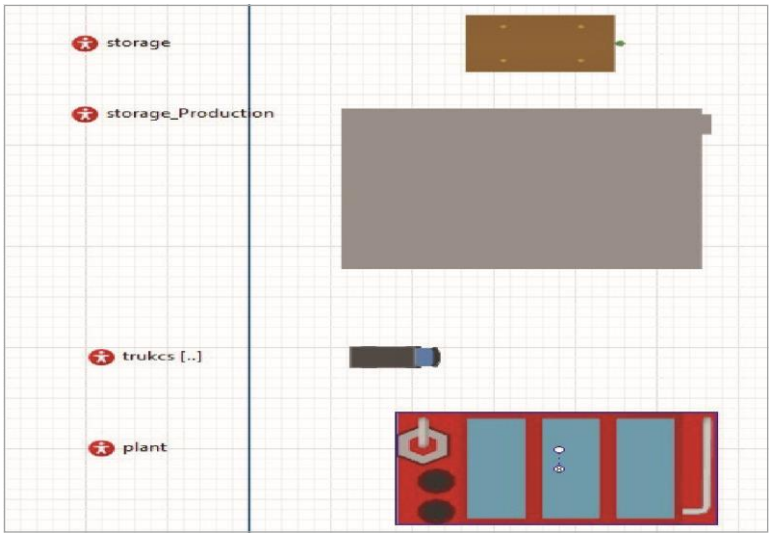


Рис. 15. Результаты этапа 1

Размещение агентов в пространстве модели

Агенты после их создания находятся в пространстве модели там, куда их поместили. Разместим их по координатам модельного пространства. Склад с деталями находится в точке с координатами X = 100, Y = 200. Склад готовой продукции — X = 1000, Y = 400. Сам цех — X = 300, Y = 200.

Для расстановки всех агентов по их координатам напишем функцию. Для создания функции используется элемент **Функция** (рис. 16).

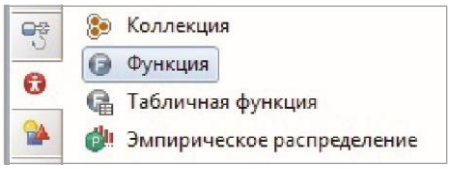


Рис. 16. Элемент **Функция**

Перетащите его на рабочее поле модели и перейдите в его свойства. В первом разделе свойств задайте имя функции — **Set**. Далее, поскольку эта функция не должна ничего считать, а должна просто расставить по координатам наших агентов, отметьте пункт **Действие** (рис. 17).

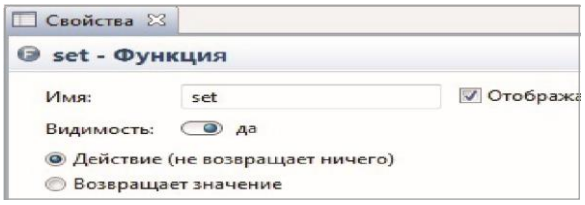


Рис. 17. Задание функции **Set**

Далее нужно написать тело функции, т.е. ту программу, которую она будет выполнять. Для этого перейдите в раздел свойств **Тело функции** и наберите представленный на рис. 18 код.

В этом коде идет обращение к функции **SetXY (X, Y)** агентов **storage**, **storage_Production**, **plant** и каждого агента из популяции **trukcs**. Функция **SetXY ()** ставит агента в указанные в аргументах функции координаты.

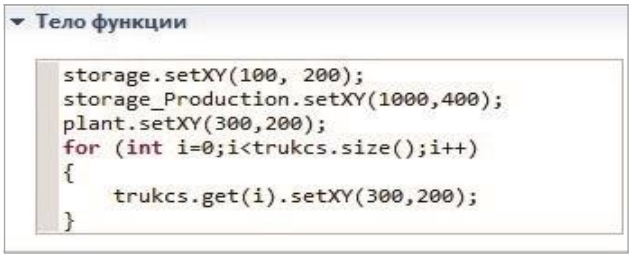


Рис. 18. Тело функции **Set**

Для того чтобы функция запустилась на выполнение, ее нужно вызвать. Организуем ее вызов при запуске главного агента **main**. Для этого откройте агент **main** (рис. 19) и щелкните на свободном в модели месте так, чтобы ни один из элементов агента **main** не был выделен.

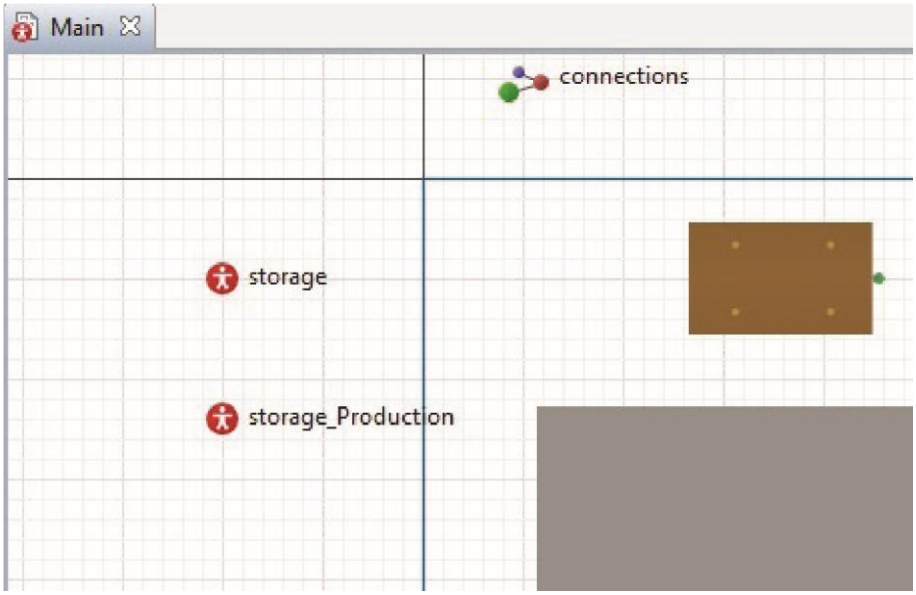


Рис. 19. Агент **main**

Перейдите в свойства агента **main** и в разделе **Действия агента** в пункте **При запуске** вызовите функцию **set ()** (рис. 20). Обратите внимание, что после **set ()** нужно поставить знак **;**.

Запустите модель на выполнение (рис. 21).

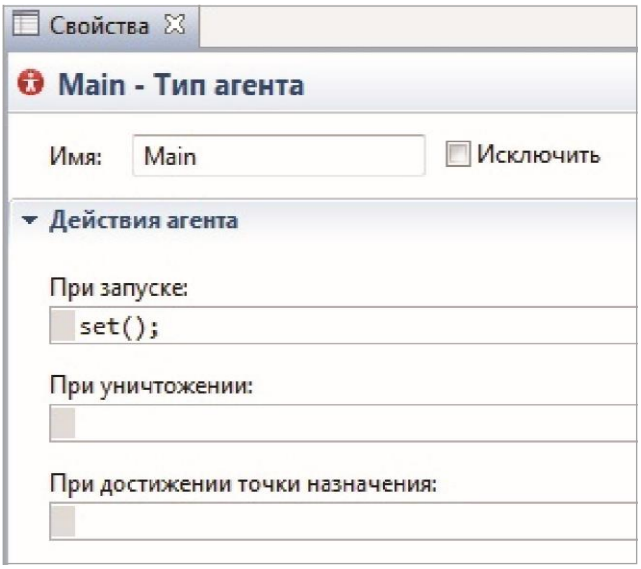


Рис. 20. Вызов функции **set ()**

Рис. 21. Расположение агентов в модели

Моделирование производства деталей на складе

На складе производятся детали с интенсивностью 5 штук в час. Для моделирования их производства в упрощенном виде, т.е. будем просто моделировать увеличение количества деталей на 5 штук в час, перейдите в агент **storage**. В этом агенте уже сеть параметр **number_of_detal**, который содержит количество деталей на складе (рис. 22).

Увеличение количества деталей на складе зададим с помощью элемента **Событие**. Этот элемент запускает на выполнение либо заданную функцию, либо заданный в нем код.

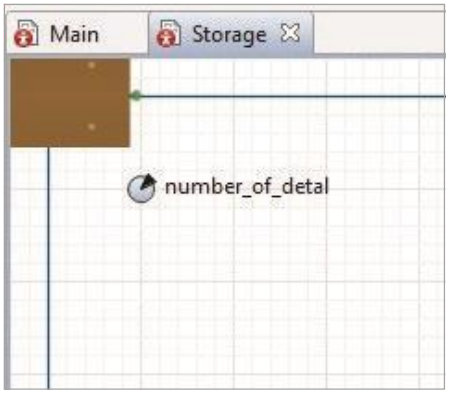


Рис. 22. Окно агента **storage**

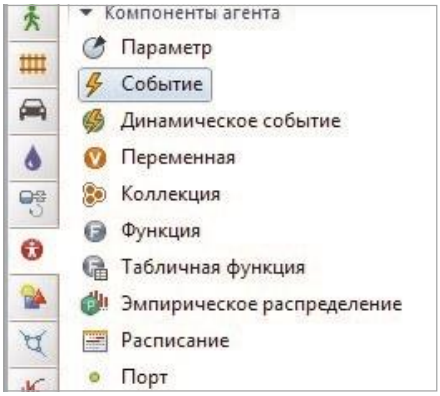


Рис. 23. Элемент **Событие**

Перетащите элемент **Событие** на рабочее поле агента **storage** и зайдите в свойства элемента **Событие** (рис. 24).

Задайте имя события — **detal_production**, тип события — **С заданной интенсивностью** и **Интенсивность 1** раз в час. Это значит, что событие будет выполняться раз в час. В разделе **Действие** задайте увеличение параметра **number_of_detal** на 5 штук. Значит, раз в час параметр **number_of_detal** будет увеличиваться на 5.

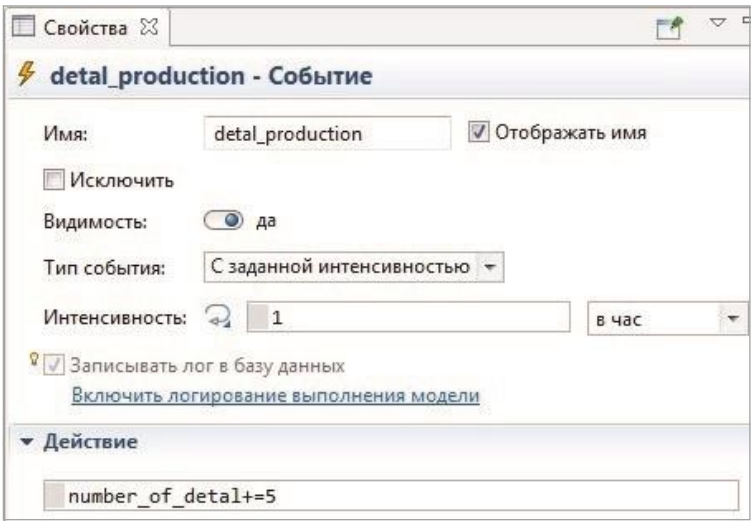


Рис. 24. Свойства элемента **detal_production**

Запустите модель на выполнение и во время работы модели зайдите в агент **storage** (рис. 25).

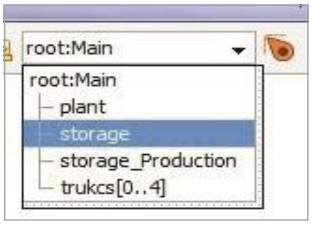


Рис. 25. Переход в агент **storage** во время выполнения моделирования

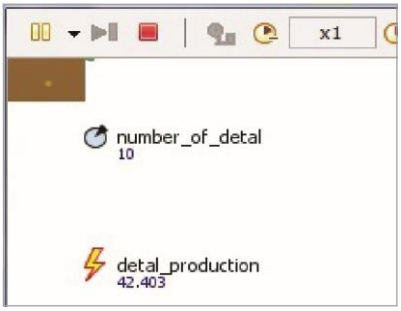


Рис. 26. Агент **storage** во время выполнения моделирования

Как видно из рис. 26, событие запускается, и параметр **number_of_ detal** растет.

Моделирование расхода деталей в цехе

Расход деталей в цехе промоделируем также с помощью элемента **Событие**. Перейдите в агент **plant**. В нем уже есть параметр **number_of_detal_ in_plant**. Добавим событие, которое будет уменьшать этот параметр на 4 штуки в час. Для этого перетащите элемент **Событие** на рабочее поле агента **plant** и задайте его свойства (рис. 27).

Запустите модель и зайдите в агент **plant**. Значение параметра **number_of_detal_in_plant** должно уменьшаться.

Рис. 27. Свойства события **detal_consumption**

Моделирование производства изделий в цехе

Для отображения количества произведенных в цехе изделий создадим параметр **number_of_izdeliya** в агенте **plant**.

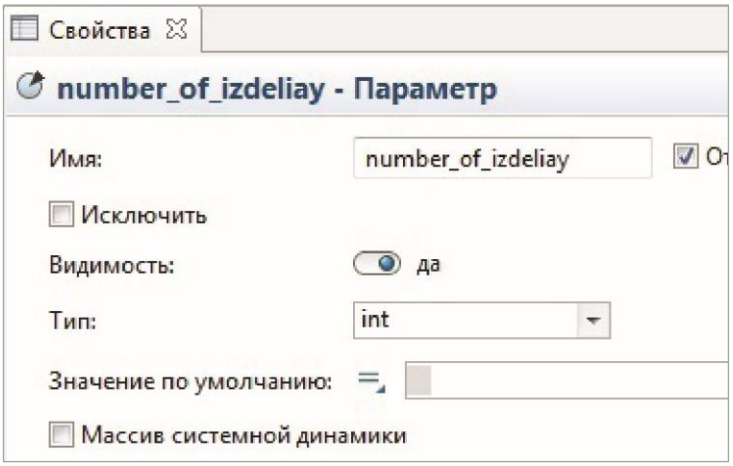


Рис. 28. Задание параметра **number_of_izdeliya**

Для этого перетащите элемент **Параметр** на рабочее поле агента **plant** и задайте его свойства (имя параметра и его тип) так же, как на рис. 28.

Производство изделий смоделируем путем увеличения параметра **number_of_izdeliya** на 5 штук в час с помощью элемента **Событие**. Для этого перетащите элемент **Событие** на рабочее поле агента **plant** и задайте его свойства так же, как показано на рис. 29.

Запустите модель и перейдите в агент **plant**. Значение параметра **number_of_izdeliay** должно расти.

Рис. 29. Свойства события **izdeliay_production**

Моделирование движения грузовика с деталями от цеха к складу

Перейдите в агент **Truck**. Добавьте в него параметр **order** типа **int**. Для этого перетащите элемент **Параметр** на рабочее поле агента **Truck** и задайте его параметры так же, как на рис. 30.

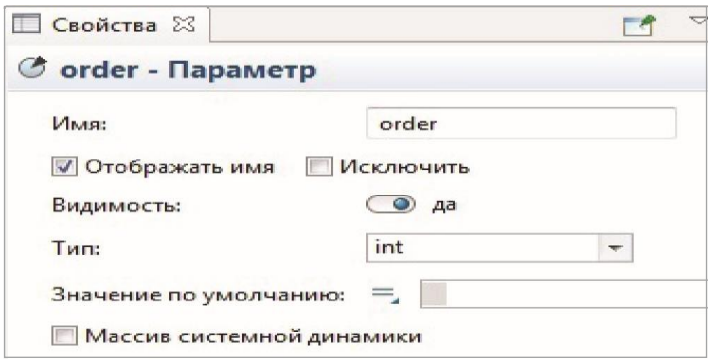


Рис. 30. Свойства параметра **order**

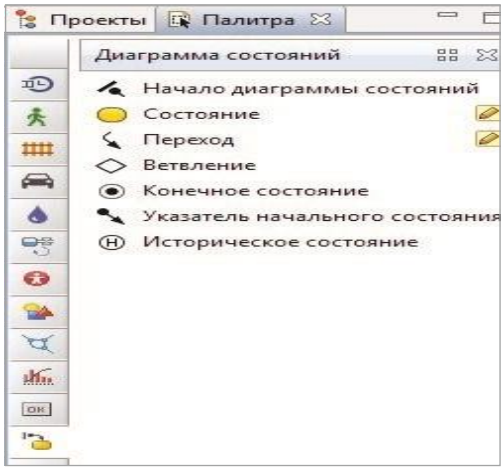


Рис. 31. Библиотека **Диаграмма состояний**

Грузовик в модели может быть в трех состояниях: находиться в цехе, находиться на складе, ехать на склад. Для моделирования состояний грузовика используется диаграмма состояний. Все инструменты для ее построения находятся в библиотеке **Диаграмма состояний** вкладки **Палитра** (рис. 31).

Перетащите на рабочее поле агента **Truck** элемент **Начало диаграммы состояний** и 3 элемента **Состояние**, дайте им имена и соедините их элементами **Переход** так же, как показано на рис. 32.

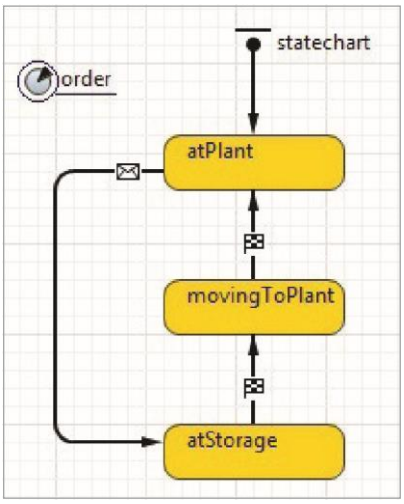


Рис. 32. Диаграмма состояний агента **Truck**

Зайдите в свойства перехода из состояния **atPlant** в состояние **atStorage** и задайте их, как показано на рис. 33.

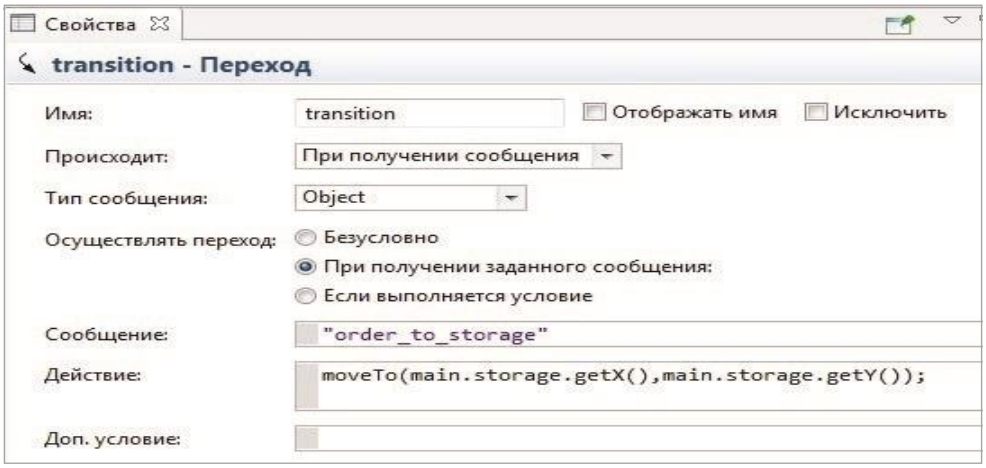


Рис. 33. Свойства перехода из состояния **atPlant** в состояние **atStorage**

Этот переход будет срабатывать, когда агент **Truck** получит сообщение «**order_to_storage»**. При срабатывании перехода агент **Truck** перейдет в состояние **atStorage** и выполнит действие, прописанное в разделе **Действие**, а именно

поедет к месту склада. Задать сообщение, при котором будет срабатывать переход, можно любое.

Задайте свойства перехода из состояния **atStorage** в состояние **movingToPlant** так, как показано на рис. 34.

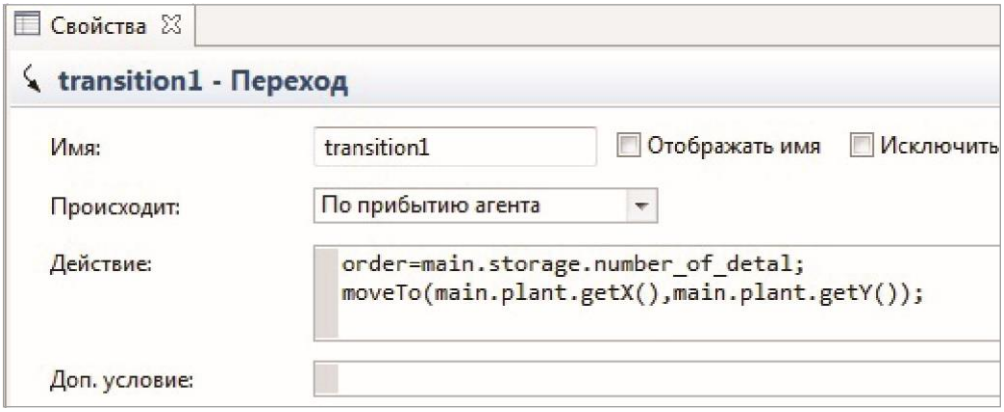


Рис. 34. Свойства перехода из состояния **atStorage** в состояние **movingToPlant**

Этот переход срабатывает сразу, как только агент прибывает на склад. При этом агент **Truck** переходит в состояние **movingToPlant** и выполняются действия, прописанные в разделе **Действие**, а именно параметр **order** становится равным количеству деталей на складе и грузовик уезжает в цех.

Задайте свойства перехода из состояния **movingToPlant** в состояние **atPlant** так, как показано на рис. 35.

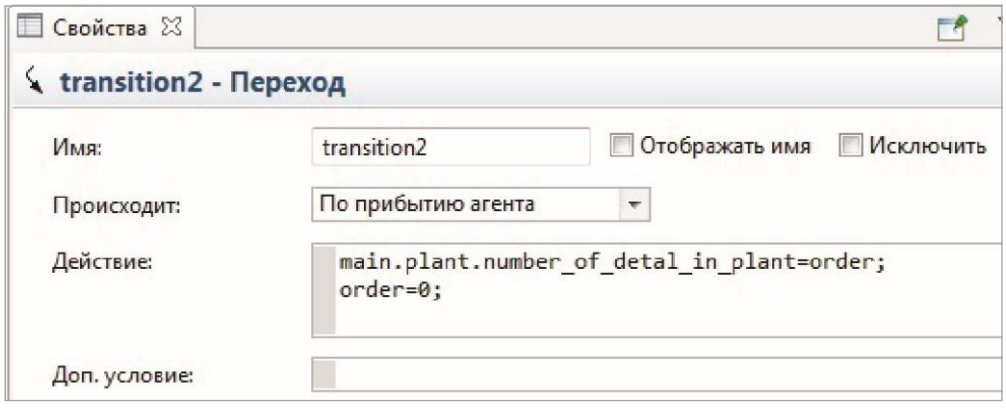


Рис. 35. Свойства перехода из состояния **movingToPlant** в состояние **atPlant**

Этот переход срабатывает сразу, как только агент начинает движение в цех. При этом агент **Truck** переходит в состояние **atPlant,** и выполняются действия, про-

писанные в разделе **Действие**, а именно все содержимое параметра **order** переписывается в параметр **number_of_ detal_in_plant** агента **Plant**, а сам параметр **order** обнуляется. Таким образом, моделируется доставка деталей со склада в цех.

Задайте скорость движения грузовика 5 км в час, зайдя в свойства агента **Truck** (рис. 36).

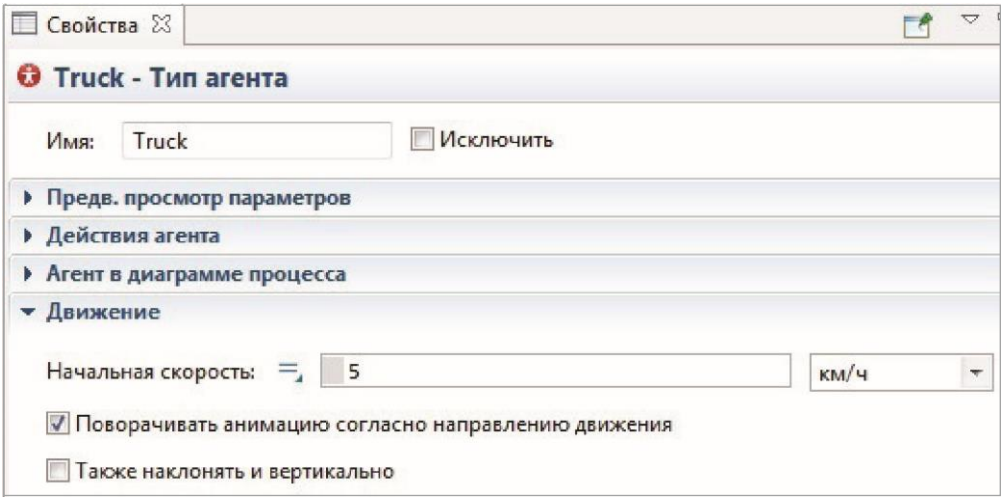


Рис. 36. Задание скорости движения грузовиков

Моделирование поиска свободного грузовика в цехе

Нам нужно найти в цехе свободный грузовик. Для этого перейдите в агент **Plant**. Грузовик является свободным, если находится в цехе. В модели это состояние грузовика обозначено как **atPlant**. Для того чтобы найти свободный грузовик, нужно перебрать все грузовики коллекции **trucks** агента **Main** и найти такие, которые находятся в состоянии **atPlant**. Первый найденный и будет нужным нам грузовиком.

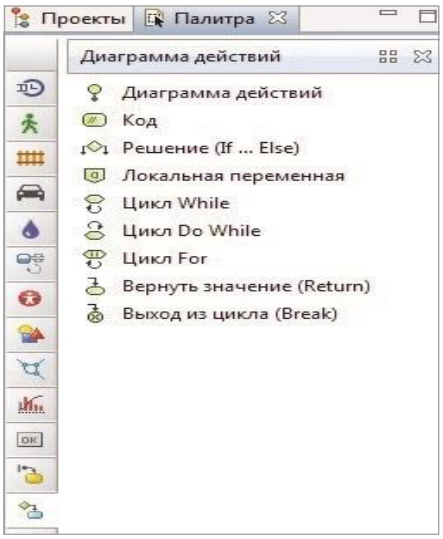


Рис. 37. Библиотека **Диаграмма действий**

Ради разнообразия для написания функции поиска свободного грузовика воспользуемся инструментом **Диаграмма действий**. Такой инструмент находится в библиотеке **Диаграмма действий** панели **Палитра** (рис. 37).

Перетащите элемент **Диаграмма действий** на рабочее поле агента **Plant** (рис. 38).

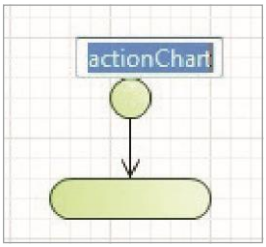


Рис. 38. Начало диаграммы действий

Перейдите в свойства диаграммы действий и задайте их, как показано на рис. 39. В свойствах задается имя функции **Find_truck** и тип возвращаемого значения – **Truck**. Это значит, что функция должна найти свободный грузовик, который в модели представлен агентом **Truck**, т.е. типом данных **Truck**. Если функция не найдет грузовик, то она вернет **Null**.

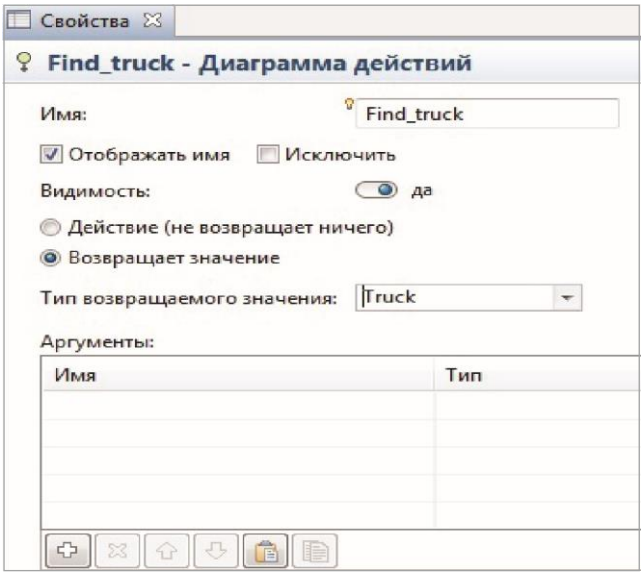


Рис. 39. Свойства функции **Find_truck**

Теперь перетащите элемент **Локальная переменная** так, чтобы появилась зеленая точка между началом и концом диаграммы (рис. 40). Должно получиться так, как показано на рис. 41.

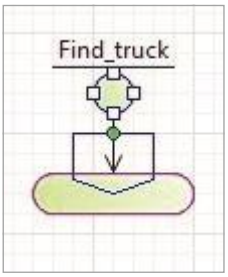


Рис. 40. Вставка элемента **Локальная переменная**

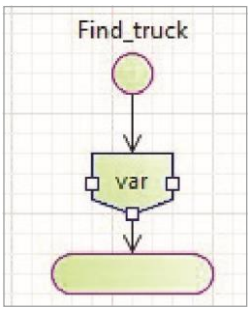


Рис. 41. Результат вставки элемента **Локальная переменная** в диаграмму действий

Задайте свойства элемента **Локальная переменная** так, как показано на рис. 42.

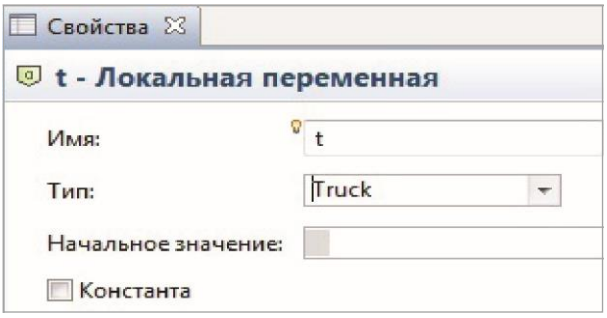


Рис. 42. Свойства элемента **Локальная переменная**

Таким образом, была создана локальная переменная для сохранения найденного грузовика. Поскольку надо организовать поиск по всей популяции грузовиков, то нужен цикл, проверяющий каждый грузовик. Перетащите элемент **Цикл For** и вставьте его между локальной переменной и концом диаграммы (рис. 43).

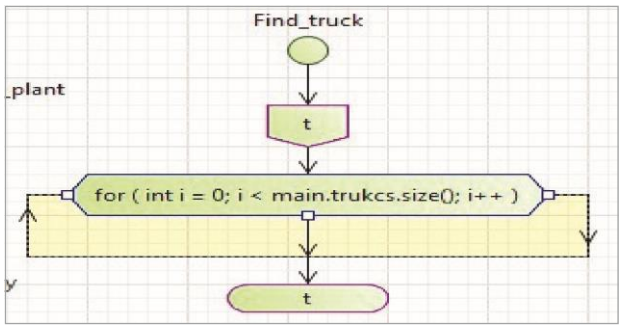


Рис. 43. Вставка цикла по коллекции

Задайте свойства цикла так, как показано на рис. 44.

Здесь задается начало и конец цикла. Цикл начинается с первого элемента коллекции и идет до последнего элемента.

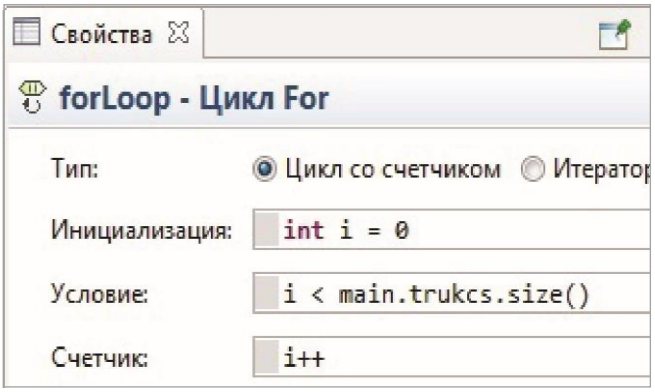


Рис. 44. Свойства цикла

Перетащите элемент **Код** и вставьте его между циклом и концом диаграммы так, как показано на рис. 45.

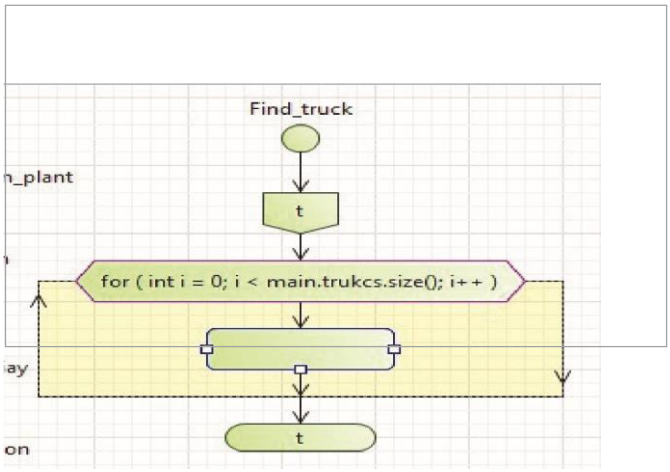


Рис. 45. Результат вставки элемента **Код**

Задайте свойства элемента **Код** так, как показано на рис. 46.

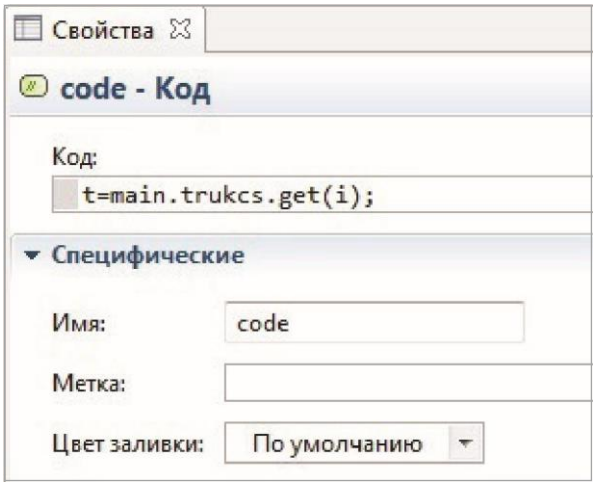


Рис. 46. Свойства элемента **Код**

В свойствах в переменную **t** записывается очередной грузовик из коллекции. Перетащите элемент **Решение** и вставьте его между элементами **Код** и концом диаграммы (рис. 47). Задайте свойства элемента **Решение** так, как показано на рис. 48.

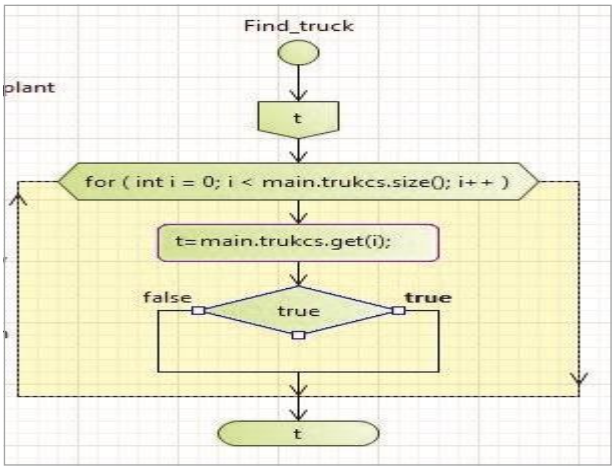


Рис. 47. Вставка элемента **Решение**

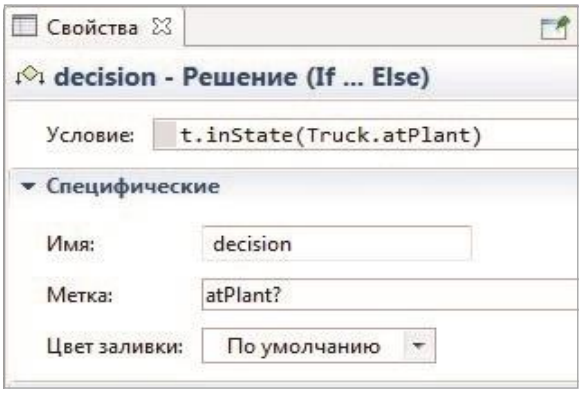


Рис. 48. Свойства элемента **Решение**

В свойствах задано условие проверки, а именно состояние элемента **t** (т. е. грузовика). Для красоты вводится метка, на которой написано само условие **atPlant**.

Перетащите элемент **Вернуть значение** в ветку **false** и введите в его свойствах **null**. В элемент **Вернуть значение** по ветке **true** введите **t** (рис. 49). Это означает, что если грузовик находится в состоянии **atPlant**, то функция вернет его значение; а если грузовик находится в любом другом состоянии, то функция вернет значение **null**.

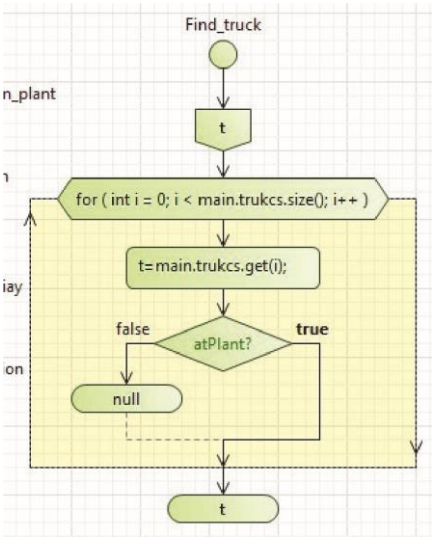


Рис. 49. Окончательный вид функции **Find_truck**

Моделирование отправки грузовика из цеха за деталями на склад

Теперь нужно найденный свободный грузовик отправить на склад за деталями. Для этого используем функцию. В функции будет вызываться функция **Find_truck**. И, если грузовик будет найден и число деталей будет меньше 5, грузовику будет посылаться сообщение, по которому он стартует на склад. Перетащите элемент **Функция** на рабочее поле агента **Plant** и задайте его свойства так, как показано на рис. 50.

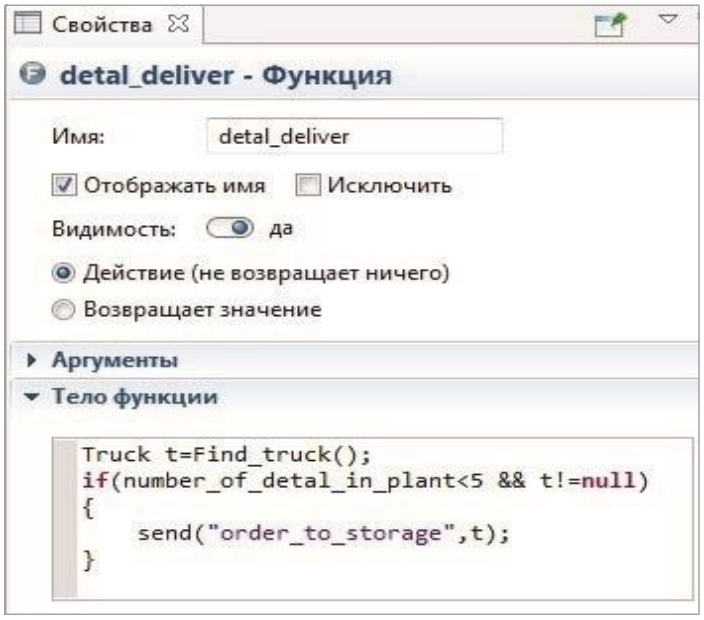


Рис. 50. Свойства функции **detal_deliver**

Для вызова функции используется событие, запускаемое каждую минуту. Перетащите элемент **Событие** и задайте его свойства так, как показано на рис. 51.

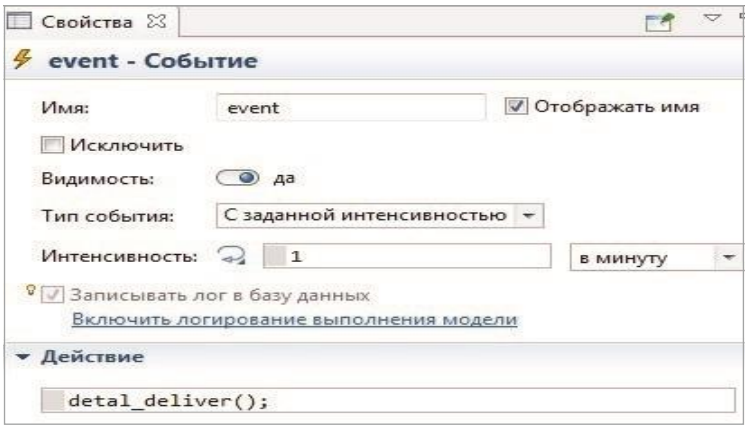


Рис. 51. Свойства события

Запустите модель. Грузовики должны ездить от склада к цеху и обратно. Количество деталей в цехе должно пополняться (рис. 52).

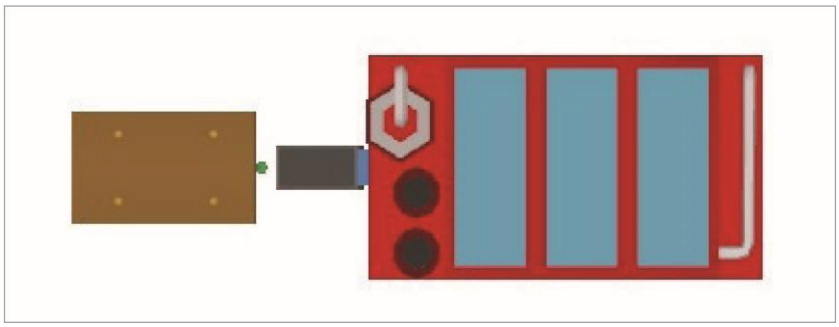


Рис. 52. Работа модели

Моделирование доставки готовой продукции из цеха на склад

Для доставки готовых изделий на склад готовой продукции используется грузовик большего объема, чем грузовики для доставки на склад деталей. Для него создадим отдельный агент **Lorry**.

Для этого перейдите в агент **main** и перетащите элемент **Агент** на рабочее поле модели. На первом шаге мастера выберите пункт **Создать единственного агента**. На втором шаге — создать новый тип агента. На третьем шаге задайте имя агента **Lorry**. На четвертом — выберите анимацию (фура). На пятом шаге создайте параметр агента **order** типа **int**. Перейдите в агента **Lorry**.

Грузовик в модели может быть в трех состояниях: находиться в цехе, находиться на складе готовой продукции и ехать на склад. Промоделируем все эти состояния с помощью диаграммы состояний. Постройте диаграмму состояний из трех состояний, просто перетаскивая элементы из библиотеки **Диаграмма состояний**. Соедините состояния переходами так, как показано на рис. 53.

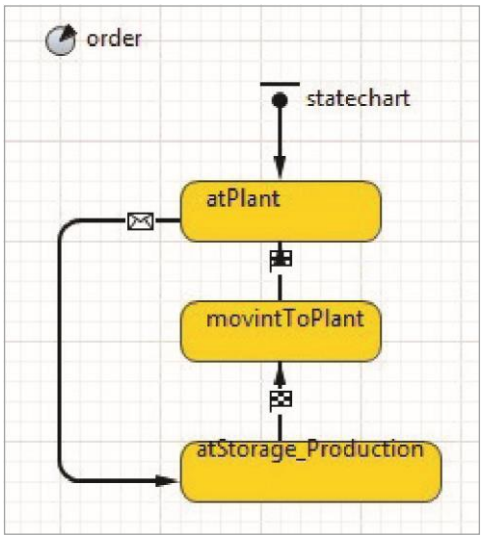


Рис. 53. Диаграмма состояний агента **Lorry**

Зайдите в свойства перехода из состояния **atPlant** в состояние **atStorage_Production** и задайте их так, как на рис. 54.

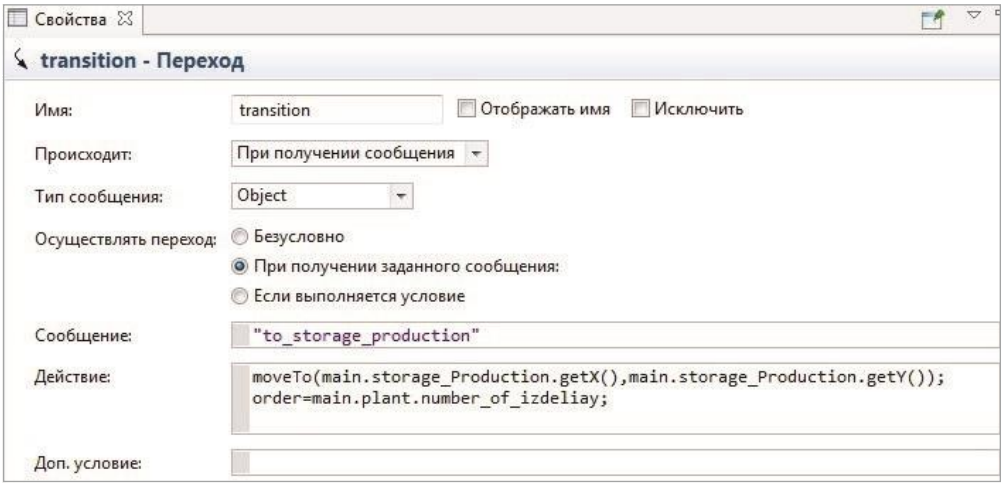


Рис. 54. Свойства перехода из состояния **atPlant** в состояние **atStorage_Production**

Такой переход будет происходить при получении сообщения **“to_storage_production”** агентом **Lorry**. При этом агент начнет движение к агенту **storage_Production** и в переменную **order** агента **Lorry** запишется количество произведенной в цехе продукции.

Выделите переход из состояния **atStorage_Production** в состояние **movingToPlant**. Перейдите в его свойства и задайте их так, как показано на рис. 55.

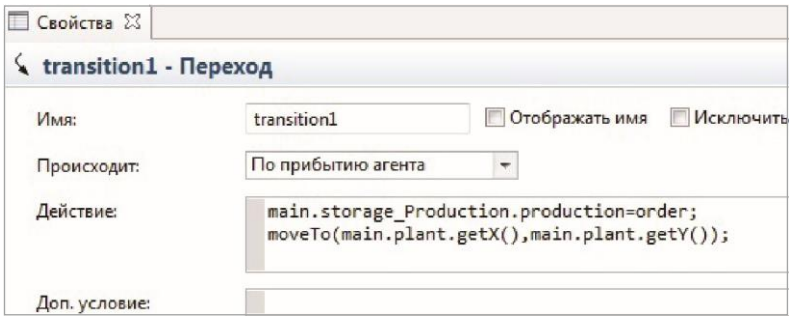


Рис. 55. Свойства перехода из состояния **atStorage_Production** в состояние **movingToPlant**

Такой переход будет происходить сразу после прибытия агента **Lorry** на склад готовой продукции, причем содержимое переменной **order** агента **Lorry** будет записываться в параметр **production** агента **storage_ Production**. После этого грузовик отправляется к цеху. Выделите переход из состояния **movingToPlant** в состояние **atPlant** и перейдите в его свойства. Задайте свойства перехода, как показано на рис. 56.

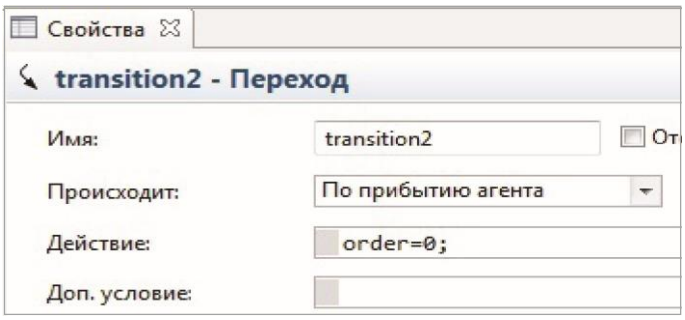


Рис. 56. Свойства перехода из состояния **movingToPlant** в состояние **atPlant**

Такой переход сработает, когда агент начнет движение в цех. При этом содержимое его параметра **order** обнулится.

Поскольку грузовик не всегда находится в цехе, в агенте **Plant** нужно организовать функцию для определения того, свободен ли грузовик в агенте **Plant**. Назовем ее **isFree** и построим с помощью **Диаграммы действий**. Перейдите в агент **Plant**. С помощью элементов библиотеки **Диаграмма действий** постройте функцию **isFree** так, как показано на рис. 57.

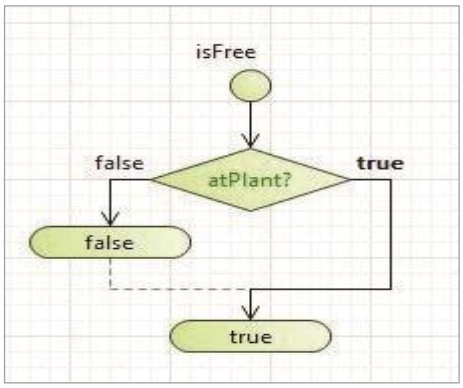


Рис. 57. Функция **isFree**

Выделите начальный элемент диаграммы и задайте свойства функции так, как показано на рис. 58.

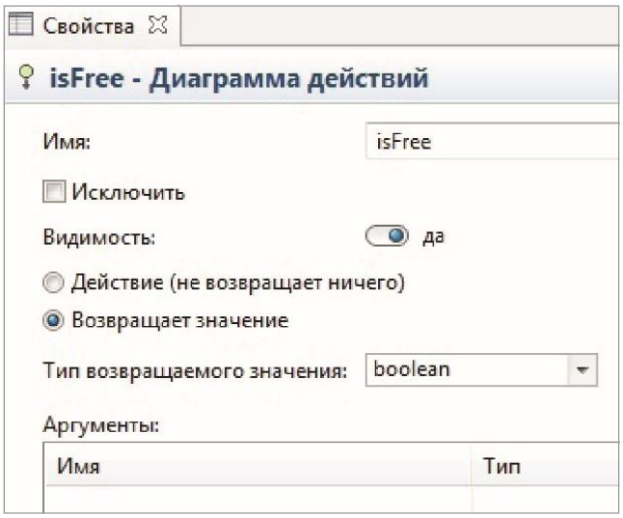


Рис. 58. Свойства функции **isFree**

Данная функция будет возвращать истину, если грузовик свободен, и ложь, если он занят. Выделите элемент **Решение** и задайте его свойства так, как показано на рис. 59.

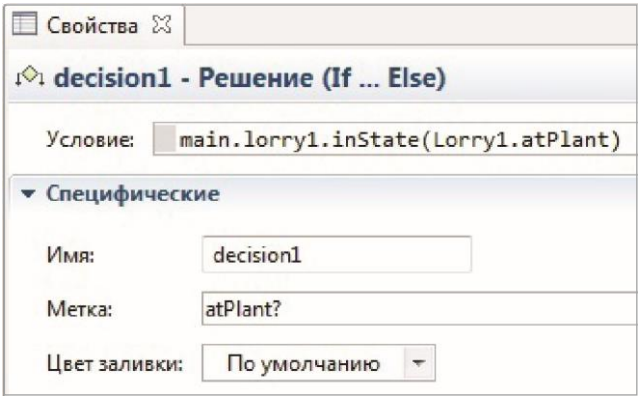


Рис. 59. Свойства элемента **Решение**

В элементе **Решение** проверяется информация о нахождении грузовика (агент **Lorry**) в состоянии **atPlant**, т.е. он находится в цехе. Если грузовик в цехе, то он свободен.

Отправкой грузовика на склад готовой продукции будет заниматься функция **izdeliya** агента **Plant**. Перетащите элемент **Функция** из агентной библиотеки и задайте его свойства так, как показано на рис. 60.

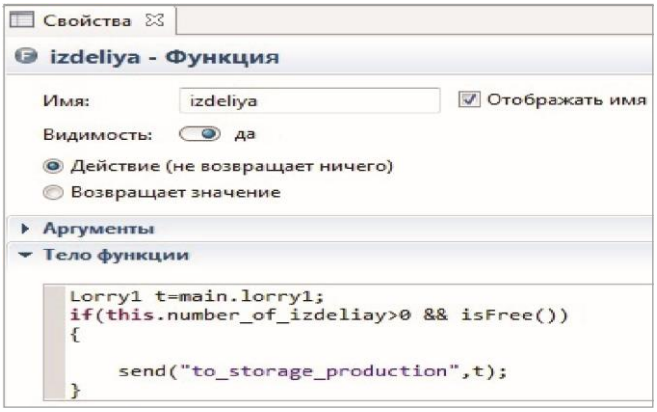


Рис. 60. Свойства функции **izdeliya**

В этой функции грузовик записывается в виде переменной **t**. Далее, если в цехе есть готовая продукция (параметр **number_of_izdeliya>0**) и есть свободный грузовик (функция **isFree ()** возвращает истину), то грузовику посылается сообщение «**to_storage_production»** и он отправляется на склад готовой продукции. Для активации этой функции используется событие **izdeliya_deliver**. Перетащите элемент **Событие** из агентной библиотеки на рабочее поле агента **Plant** и задайте его свойства так, как показано на рис.61. Такое событие будет происходить 10 раз за час, и каждый раз будет вызываться функция **izdeliya**.

Рис. 61. Свойства события **izdeliya_deliver**

Запустите модель. От цеха должны двигаться грузовики на склад деталей и на склад готовой продукции (рис. 62).

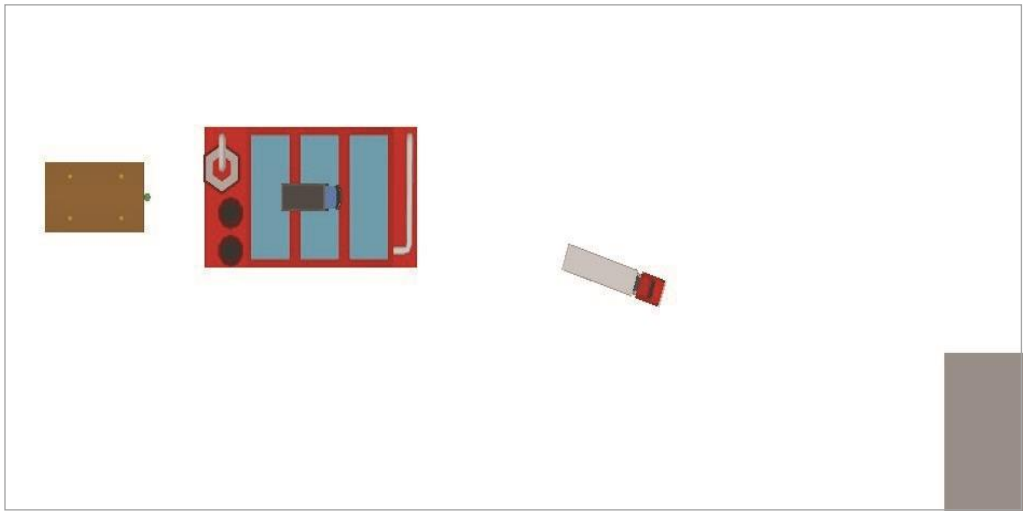


Рис. 62. Работа конечной модели

ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

В среде AnyLogic построить компьютерные модели

Задание 1. На сборочный участок цеха предприятия через интервалы времени, распределенные экспоненциально со средним значением 10 мин, поступают партии, каждая из которых состоит из трех деталей. Половина всех поступающих деталей перед сборкой должна пройти предварительную обработку в течение 7 мин. На сборку подаются обработанная и необработанная детали. Процесс сборки занимает всего 6 мин. Затем изделие поступает на регулировку, продолжающуюся в среднем 8 мин (время выполнения ее распределено экспоненциально). В результате сборки возможно появление 4 % бракованных изделий, которые не поступают на регулировку, а направляются снова на предварительную обработку.

Смоделировать работу участка.

Задание 2. На обрабатывающий участок цеха поступают детали в среднем через 50 мин. Первичная обработка деталей производится на одном из двух станков. Первый станок обрабатывает деталь в среднем 40 мин и имеет до 4 % брака, второй соответственно 60 мин и 8 % брака. Все бракованные детали возвращаются на повторную обработку на второй станок. Детали, попавшие в разряд бракованных дважды, считаются отходами. Вторичную обработку проводят также два станка в среднем 100 мин каждый. Причем первый станок обрабатывает имеющиеся в накопителе после первичной обработки детали, а второй станок подключается при образовании в накопителе задела больше трех деталей. Все интервалы времени распределены по экспоненциальному закону. Смоделировать обработку на участке.

Задание 3. На регулировочный участок цеха через случайные интервалы времени поступают по два агрегата в среднем через каждые 30 мин. Первичная регулировка осуществляется для двух из **166** агрегатов одновременно и занимает около 30 мин. Если в момент прихода агрегатов предыдущая партия не была обработана, поступившие агрегаты на регулировку не принимаются. Агрегаты после

первичной регулировки, получившие отказ, поступают в промежуточный накопитель. Из накопителя агрегаты, прошедшие первичную регулировку, поступают попарно на вторичную регулировку, которая выполняется в среднем за 30 мин, а не прошедшие первичную регулировку поступают на полную, которая занимает 100 мин для одного агрегата. Все величины, заданные средними значениями, распределены экспоненциально. Смоделировать работу участка.

Задание 4. Система передачи данных обеспечивает передачу пакетов данных из пункта *A* в пункт С через транзитный пункт *B.* В пункт *A* пакеты поступают через 10 ± 5 мс. Здесь они буферируются в накопителе емкостью 20 пакетов и передаются по любой из двух линий *AB1* — за время 20 мс или *AB2* — за время 20 ± 5 мс. В пункте *B* они снова буферируются в накопителе емкостью 25 пакетов и далее передаются по линиям *BC1* (за 25 ± 3 мс) и *BC2* (за 25 мс). Причем пакеты из *AB1* поступают в *BC1,* а.из *AB2* — в *BC2.* Чтобы не было переполнения накопителя, в пункте *B* вводится пороговое значение его емкости — 20 пакетов. При достижении очередью порогового значения происходит подключение резервной аппаратуры и время передачи снижается для линий ***BC1*** и ***BC2*** до 15 мс. Смоделировать прохождение через систему передачи.

Задание 5. Система обработки информации содержит мультиплексный канал и три мини-ЭВМ. Сигналы от датчиков поступают на вход канала через интервалы времени 10 ± 5 мкс. В канале они буферируются и предварительно обрабатываются в течение 10 ± 3 мкс. Затем они поступают на обработку в ту мини-ЭВМ, где имеется наименьшая по длине входная очередь. Емкости входных накопителей во всех мини-ЭВМ рассчитаны на хранение величин 10 сигналов. Время обработки сигнала в любой мини-ЭВМ равно 33 мкс. Смоделировать процесс обработки сигналов, поступающих с датчиков.

Задание 6. На участке термической обработки выполняются цементация и закаливание шестерен, поступающих через 10 ± 5 мин. Цементация занимает 10 ± 7 мин, а закаливание — 10 ± 6 мин. Качество определяется суммарным временем обработки. Шестерни с временем обработки больше 25 мин покидают участок, с временем обработки от 20 до 25 мин передаются на повторную закалку и при времени обработки меньше 20 мин должны пройти повторную полную обработку. Детали с суммарным временем обработки меньше 20 мин считаются вторым сортом. Смоделировать процесс обработки на участке.

Задание 7. Магистраль передачи данных состоит из двух каналов (основного и резервного) и общего накопителя. При нормальной работе сообщения передаются по основному каналу за 7 ± 3 с. В основном канале происходят сбои через интервалы времени 200 ± 35 с. Если сбой происходит во время передачи, то за 2 с запускается запасной канал, который передает прерванное сообщение с самого начала. Восстановление основного канала занимает 23 ± 7 с. После восстановления резервный канал выключается и основной канал продолжает работу с очередного сообщения. Сообщения поступают через 9±4 сек. и остаются в накопителе до окончания передачи. В случае сбоя передаваемое сообщение передается повторно по запасному каналу. Смоделировать работу магистрали передачи данных.

Задание 8. На комплектовочный конвейер сборочного цеха каждые 5 ± 1 мин поступают 5 изделий первого типа и каждые 20 ± 7 мин поступают 20 изделий второго типа. Конвейер состоит из секций, вмещающих по 10 изделий каждого типа. Комплектация начинается только при наличии деталей обоих типов в требуемом количестве и длится 10 мин. При нехватке деталей секция конвейера остается пустой. Смоделировать работу конвейера сборочного цеха.

Задание 9. В системе передачи данных осуществляется обмен пакетами данных между пунктами *A и B по* дуплексному каналу связи. Пакеты поступают в пункты системы от абонентов с интервалами времени между ними 10 ± 3 мс. Передача пакета занимает 10 мс. В пунктах имеются буферные регистры, которые могут хранить два пакета (включая передаваемый). В случае прихода пакета в момент занятости регистров пунктам системы предоставляется выход на спутниковую полудуплексную линию связи, которая осуществляет передачу пакетов данных за 10 ± 5 мс. При занятости спутниковой линии пакет получает отказ. Смоделировать обмен информацией в системе передачи.

Задание 10. Транспортный цех объединения обслуживает три филиала *A, B и C.* Грузовики перевозят изделия из *A* в *B* и из *B в C,* возвращаясь затем в *A* без груза. Погрузка в *A* занимает 20 мин, переезд из *A* в *B* длится 30 мин, разгрузка и погрузка в *B* – 40 мин, переезд в *C* – 30 мин, разгрузка в С – 20 мин и переезд в *A* – 20 мин. Если к моменту погрузки *в A и B* отсутствуют изделия, грузовики уходят дальше по маршруту. Изделия в *A* выпускаются партиями по 1000 шт. через 20 ± 3 мин, в *B* – такими же партиями через 20 ± 5 мин. На линии работает 8 грузовиков, каждый

перевозит 1000 изделий. В начальный момент все грузовики находятся в *A.* Смоделировать работу транспортного цеха объединения.

Задание 11. Специализированная вычислительная система состоит из трех процессоров и общей оперативной памяти. Задания, поступающие на обработку через интервалы времени 5 ± 2 мин, занимают объем оперативной памяти размером в страницу. После трансляции первым процессором в течение 5 ± 1 мин их объем увеличивается до двух страниц и они поступают в оперативную память. Затем после редактирования во втором процессоре, которое занимает 2,5 ± 0,5 мин на страницу, объем возрастает до трех страниц. Отредактированные задания через оперативную память поступают в третий процессор на решение, требующее 1,5 ± 0,4 мин на страницу, и покидают систему, минуя оперативную память. Смоделировать работу вычислительной системы.

Задание 12. На вычислительном центре в обработку принимаются три класса заданий *A, B и C.* Исходя из наличия оперативной памяти ЭВМ задания классов *A* и *B* могут решаться одновременно, а задания класса С монополизируют ЭВМ. Задания класса *A* поступают через 20 ± 5 мин, класса *B* – через 20 ± 10 мин и класса С – через 30 ± 10 мин и требуют для выполнения: класс *A* – 20 ± 5 мин, класс *B* – 21 ± 3 мин и класс С – 28 ± 5 мин. Задачи класса *C* загружаются в ЭВМ, если она полностью свободна. Задачи классов *A* и *B* могут дозагружаться к решающейся задаче. Смоделировать работу.

Задание 13. В студенческом машинном зале расположены две мини-ЭВМ и одно устройство подготовки данных (УПД). Студенты приходят с интервалом в 8 ± 2 мин, и треть из них хочет использовать УПД и ЭВМ, а остальные только ЭВМ. Допустимая очередь в машинном зале составляет четыре человека, включая работающего на УПД. Работа на УПД занимает 8 ± 1 мин, а на ЭВМ –17 мин. Кроме того, 20 % работавших на ЭВМ возвращается для повторного использования УПД и ЭВМ. Смоделировать работу машинного зала.

Задание 14. К мини-ЭВМ подключено четыре терминала, с которых осуществляется решение задач. По команде с терминала выполняют операции редактирования, трансляции, планирования и решения. Причем, если хоть один терминал выполняет планирование, остальные вынуждены простаивать из-за нехватки оперативной памяти. Если два терминала выдают требование на решение,

то оставшиеся два простаивают, и если работают три терминала, выдающих задания на трансляцию, то оставшийся терминал блокируется. Интенсивности поступления задач различных типов равны. Задачи одного типа от одного терминала поступают через экспоненциально распределенные интервалы 1, времени со средним значением 160 с. Выполнение любой операции * длится 10 с. Смоделировать работу мини-ЭВМ.

Задание 15. В системе передачи цифровой информации передается речь в цифровом виде. Речевые пакеты передаются через два транзитных канала, буферуясь в накопителях перед **170** каждым каналом. Время передачи пакета по каналу составляет 5 мс. Пакеты поступают через 6 ± 3 мс. Пакеты, передававшиеся более 10 мс, на выходе системы уничтожаются, так как их появление в декодере значительно снизит качество передаваемой речи. Уничтожение более 30 % пакетов недопустимо. При достижении такого уровня система за счет ресурсов ускоряет передачу до 4 мс на канал. При снижении уровня до приемлемого происходит отключение ресурсов. Смоделировать работу системы.

Задание 16. ЭВМ обслуживает три терминала по круговому циклическому алгоритму, предоставляя каждому терминалу 30 с. Если в течение этого времени задание обрабатывается, то обслуживание завершается; если нет, то остаток задачи становится в специальную очередь, которая использует свободные циклы терминалов, т. е. задача обслуживается, если на каком-либо терминале нет заявок. Заявки на терминалы поступают через 30 ± 5 с и имеют длину 300 ± 50 знаков. Скорость обработки заданий ЭВМ равна 10 знаков/с. Смоделировать работу ЭВМ.

Задание 17. В узел коммутации сообщений, состоящий из входного буфера, процессора, двух исходящих буферов и двух выходных линий, поступают сообщения с двух направлений. Сообщения с одного направления поступают во входной буфер, обрабатываются в процессоре, буферируются в выходном буфере первой линии и передаются по выходной линии. Сообщения со второго направления обрабатываются аналогично, но передаются по второй выходной линии. Применяемый метод контроля потоков требует одновременного присутствия в системе не более трех сообщений на каждом направлении. Сообщения поступают через интервалы 15 ± 7 мс. Время обработки в процессоре равно 7 мс на сообщение, время передачи по выходной линии равно 15 ± 5 мс. Если сообщение поступает при наличии трех сообщений в направлении, то оно получает отказ. Смоделировать работу узла коммутации.

Задание 18. Распределенный банк данных системы сбора информации организован на базе ЭВМ, соединенных дуплексным каналом связи. Поступающий запрос обрабатывается на первой. ЭВМ и с вероятностью 50 % необходимая информация обнаруживается на месте. В противном случае необходима посылка запроса во вторую ЭВМ. Запросы поступают через 10 ± 3 с, первичная обработка запроса занимает 2 с, выдача ответа требует 18 ± 2 с, передача по каналу связи занимает 3 с. Временные характеристики второй ЭВМ аналогичны первой. Смоделировать прохождение запросов.

Задание 19. Система автоматизации проектирования состоит из ЭВМ и трех терминалов. Каждый проектировщик формирует задание на расчет в интерактивном режиме. Набор строки задания занимает 10 ± 5 с. Получение ответа на строку требует 3 с работы ЭВМ и 5 с работы терминала. После набора десяти строк задание считается сформированным и поступает на решение, при этом в течение 10 ± 3 с ЭВМ прекращает выработку ответов на вводимые строки. Вывод результата требует 8 с работы терминала. Анализ результата занимает у проектировщика 30 с, после чего цикл повторяется. Смоделировать работу системы.

Задание 20. Из литейного цеха на участок обработки и сборки поступают заготовки через 20 ± 5 мин. Треть из них обрабатывается в течение 60 мин и поступает на комплектацию. Две трети заготовок обрабатывается за 30 мин перед комплектацией, которая требует наличия одной детали первого типа и двух деталей второго. После этого все три детали подаются на сборку, которая занимает 60 ± 2 мин для первой детали и 60 ± 8 мин для двух других, причем они участвуют в сборке одновременно. При наличии на выходе одновременно всех трех деталей изделие покидает участок. Смоделировать работу участка.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Бенькович, Е.С. Практическое моделирование динамических систем / Е.С. Бенькович, Ю.Б. Колесов, Ю.Б. Сенюченков. – Санкт Петербург: БХВ-Петербург, 2002. – 464 с. – ISBN 5-94157-099-6.

2. Боев, В.Д. Моделирование в среде AnyLogic / В.Д. Боев. – Москва: ЮРАЙТ, 2017. – 299 с. – ISBN 9-785534-048-056.

3. Григорьев, И.И. AnyLogic за три дня. Практическое пособие по имитационному моделированию / И.И. Григорьев // Интернет-портал фирмы AnyLogic: [сайт]. – URL: https://www.anylogic.ru/resources/books (дата обращения: 12.12.2023). — 273 с.

4. Интернет-портал фирмы AnyLogic: Справочные руководства по AnyLogic. – URL: https://anylogic.help/ru/tutorials/ (дата обращения: 12.12.2023).

5. Калинин, И.А. Информатика. Углубленный уровень: задачник-практикум для 10–11 классов / И.А. Калинин, Н.Н. Самылкина, П.В. Бочаров. – Москва: БИНОМ. Лаборатория знаний, 2015. – 248 с. – ISBN 978-5-9963-1722-6.

6. Карпов, Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic / Ю.Г. Карпов. – Санкт-Петербург: БХВ-Петербург, 2005. – 400 с. – ISBN 5-94157-148-8.

7. Лимановская, О.В. Моделирование производственных процессов в AnyLogic 8.1. Лабораторный практикум / О.В. Лимановская, Т.И. Алферьева. – Екатеринбург: Изд-во Урал. ун-та, 2019. – 136 с. – ISBN 978-5-7996-2680-8.

8. Советов, Б.Я. Моделирование систем / Б.Я. Советов, С.А. Яковлев. – Москва: Высшая школа, 2001. – 343 с. – ISBN 5-06-003860-2.

9. Советов, Б.Я. Моделирование систем. Лабораторный практикум / Б.Я. Советов, С.А. Яковлев. – Москва: ЮРАЙТ, 2022. – 295с. – ISBN 978-5-9916-2858-7.

Учебное издание

Королев Александр Леонидович

Паршукова Наталья Борисовна

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ В СРЕДЕ ПРОГРАММНОГО КОМПЛЕКСА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ANYLOGIC

Учебное пособие

ISBN 978-5-907869-04-2

Рукопись рекомендована РИС ЮУрГГПУ

Протокол № 30 от 2024 г.

Редактор Е.М. Сапегина

Технический редактор Н.А. Усова

Издательство ЮУрГГПУ

454080, г. Челябинск, пр. Ленина, 69

Подписано в печать 25.04.2024 г.

Объем 5,5 уч.-изд. л. (22,8 усл. п. л.)

Формат 60*84/8 Тираж 100 экз.

Заказ №

Отпечатано с готового оригинал-макета в типографии ЮУрГГПУ

454080, г. Челябинск, пр. Ленина, 69