ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ AnyLogic
Часть 1

ХАНТЫ-МАНСИЙСКИЙ АВТОНОМНЫЙ ОКРУГ – ЮГРА

БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем обработки информации и управления

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ AnyLogic Часть 1

Учебно-методическое пособие

Сургут
Издательский центр СурГУ
2023

УДК 004.94(095.8)
ББК 32.973.26я73
И524

Издается по решению
Редакционно-издательского совета СурГУ

Рецензенты
канд. тех. наук, доцент кафедры автоматики и компьютерных систем СурГУ
П. В. Гришмановский;
канд. тех. наук, доцент кафедры информационно-вычислительной техники СурГУ
Д. А. Федоров

И524 **Имитационное моделирование в среде AnyLogic**: учеб.-метод. пособие: в 2 ч. Ч. 1 / сост. К. И. Бушмелева, А. В. Никифоров ; Сургут. гос. ун-т. – Сургут : ИЦ СурГУ, 2023. – 22 с.

Учебно-методическое пособие содержит учебный материал о методах и средствах имитационного моделирования сложных систем и объектов в инструментальной среде AnyLogic; предназначено для студентов технических специальностей всех форм обучения по.

УДК 004.94(095.8)
ББК 32.973.26я73

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Моделирование является универсальным методом получения и использования знаний об окружающем мире, всегда используется человеком в целенаправленной деятельности, особенно в исследовательской.

Моделирование – метод решения задач, при использовании которого исследуемая система заменяется более простым объектом, описывающим реальную систему и называемым моделью.

Моделирование применяется в случаях, когда проведение экспериментов над реальной системой невозможно или нецелесообразно, например, из-за высокой стоимости или длительности проведения эксперимента в реальном времени.

Различают физическое и математическое моделирование. Примером физической модели является уменьшенная копия самолета, продуваемая в потоке воздуха. При использовании математического моделирования поведение системы описывается с помощью формул. Особым видом математических моделей являются имитационные модели.

МОДЕЛИРОВАНИЕ. ОСНОВНЫЕ ПОНЯТИЯ

Моделирование является общепризнанным средством познания действительности. Этот процесс состоит из двух основных этапов: разработки модели и ее последующего анализа. Моделирование позволяет исследовать суть сложных процессов и явлений с помощью экспериментов не с реальной системой, а с ее моделью.

В области создания новых систем моделирование является средством исследования наиболее значимых характеристик будущей системы на самых ранних стадиях ее разработки. С помощью моделирования можно исследовать «узкие» места будущей системы, оценить производительность, стоимость, пропускную способность, то есть все главные ее характеристики, еще до того, как система будет создана.

Моделирование – это процесс изучения реального объекта путем построения и исследования его модели, которая отображает только те элементы реального мира, которые исследуются. При этом то, что необходимо включать в модель, определяется целями моделирования.

Ключевой момент в моделировании – использование в исследовании модели объекта, а не самого объекта. При построении модели главное помнить о цели моделирования и, уже основываясь на этом, учитывать только те свойства реального объекта (системы), которые важны с позиции целей моделирования. Если результаты экспериментов над моделью подтверждаются результатами экспериментов над реальным объектом, то такая модель является адекватной. Адекватность модели определяется целями моделирования и принятыми критериями.

Реальные объекты представляют собой, как правило, сложные системы, поскольку процессы в них зависят от многих параметров и часто не могут быть представлены аналитически. Кроме того, в реальном мире сложные системы связаны в сеть и представляют собой большие системы. Их особенность заключается в непостоянстве процессов в них из-за нестабильного поведения объектов системы. Как правило, это системы с участием людей, ведь именно человеческий фактор вносит эффект нестабильности.

Тогда встает вопрос: а как же создавать модель таких систем? Очень просто – по принципу «разделяй и властвуй»! Систему разбивают на подсистемы. Если подсистемы оказываются слишком сложными, то разбивают и их. Предел разбиения, то есть элементарная единица системы, зависит от целей моделирования. После разбиения системы на подсистемы устанавливаются связи между ними.

Несмотря на огромное разнообразие моделей, их довольно легко сгруппировать по типу вычислительного процесса и типу данных. Вычислительный процесс может быть непрерывным и дискретным. Непрерывный вычислительный процесс рассматривает постепенное развитие модели во времени. Дискретный процесс рассматривает развитие системы как набор переходов из одного состояния в другое, причем нет промежуточных состояний. Типы данных могут быть детерминированными и вероятностными. Детерминированные данные изменяются по определенному аналитическому закону. Вероятностные данные меняются случайным образом. В результате можно отметить непрерывно-детерменированную модель (далее – D-модель) и непрерывно-вероятностную модель (далее – Q-модель), а также дискретно-детерменированную модель (далее – F-модель) и дискретно-вероятностную модель (далее – P-модель). Кроме того, есть комбинированные модели, которые называют A-моделями, а также сети Петри (сетевые модели), которые выделяют в отдельные N-модели.

С помощью моделей разрабатываются оптимальные операционные планы и расписания функционирования существующих сложных систем. В организационных системах моделирование становится основным инструментом сравнения различных вариантов решений и поиска наиболее эффективного из них на различных уровнях.

Модели сложных систем строятся в виде программ, реализуемых на компьютере. Компьютерное моделирование (далее – КМ) существует уже многие делятки лет, с момента появления первых компьютеров. С тех пор появились два пересекающихся направления КМ – математическое и имитационное.

Математическое моделирование связано в основном с разработкой математических моделей физических явлений, с созданием и обоснованием численных методов.

Имитационное моделирование (далее – ИМ) связано с разработкой и выполнением на компьютере модели в виде программы, отражающей поведение и структуру моделируемого объекта. Эксперимент с моделью состоит в выполнениии на компьютере данной программы с разными значениями параметров и анализе результатов этих действий.

Имитационное моделирование решает проблемы реального мира безопасно и разумно. Это удобный инструмент для анализа: он нагляден, прост для понимания и проверки. В разных областях бизнеса и науки ИМ помогает найти оптимальные решения и дает четкое представление о сложных системах.

Имитационное моделирование – эксперимент над достоверным цифровым представлением любой системы. В отличие от физического моделирования, такого как создание макета здания, ИМ основано на компьютерных технологиях, использующих алгоритмы и уравнения. Имитационную модель можно анализировать в динамике, а также просматривать анимацию в 2D или 3D; она отображает гораздо больше деталей, чем аналитическая, что делает ее точнее, а прогнозы на ее основе – более определенными. Например, горнодобывающие компании смогут значительно сократить расходы, если оптимизируют использование ресурсов и сделают прогноз потребности в оборудовании. Возможность представления имитационных моделей в 2D и 3D сделает любые идеи и концепции более наглядными, их проще проверять и обсуждать с коллегами. Аналитики и инженеры, увидев модель в действии, смогут сделать правильные выводы сами и четко донести их до руководства.

Имитационное моделирование используется в бизнесе, когда проведение экспериментов на реальной системе невозможно или непрактично, чаще всего из-за их стоимости или длительности. Виртуальные эксперименты с имитационными моделями обойдутся гораздо дешевле и займут меньше времени, чем эксперименты с реальными системами. Например, можно тестировать эффективность маркетинговых кампаний, не привлекая внимания конкурентов и избегая лишних затрат. Неопределенность во времени и результатах операций легко отражается с помощью имитационной модели, что позволяет оценить степень риска и найти наиболее надежные решения. Например, можно получить реалистичную картину работы логистической сети, если включить в модель случайные величины, например, срок доставки.

Возможность анализировать модель в действии отличает ИМ от других методов, например, от использования Excel или линейного программирования. Пользователь изучает процессы и вносит изменения в имитационную модель в ходе работы, что позволяет лучше проанализировать работу системы и быстро решить поставленную задачу. В отличие от аналитики на основе таблиц или методом линейной оптимизации, моделирование дает возможность наблюдать поведение реальной системы во времени с необходимым уровнем детальности. Например, можно проверить уровень загрузки складского помещения на заданную дату.

Моделирование предполагает наличие следующих этапов:

1. Изучение предметной области моделирования и определение целей моделирования.
2. Выделение критериев отбора характеристик процесса (явления, объекта), важных для целей моделирования.
3. Разработка модели.
4. Адаптация модели и экспериментальных данных, а также проверка ее адекватности.
5. Проведение экспериментов над моделью [1, 2].

Разработка модели идет, как правило, в три этапа:

1. Этап моделирования «черный ящик». Система представляется в виде черного ящика с набором входов и выходов. На этом этапе устанавливаются границы модели, отделяющие ее от внешней среды. Также на этом этапе устанавливаются входные данные системы. Решается вопрос об их представлении, периодичности, меняются ли они с определенной вероятностью или по аналитическому закону. На этом этапе устанавливаются выходные данные, их вид, периодичность получения, пользователи этих данных.

Кроме того, на данном этапе при проектировании динамической модели учитывается тип динамики системы: функционирование или развитие. Функционирование – это процессы, происходящие в системе стабильно и реализующие фиксированную цель, например процессы, протекающие в технологическом конвейере. Развитие – процессы, протекающие в системе при изменении ее целей. Например, решено развивать производство: в технологическую цепочку, выпускающую один набор изделий, добавляется новое изделие, изготовление которого предполагает некоторые изменения.

2. Этап «Модель состава системы» предполагает, что система разбивается на элементы по уже описанному выше сценарию.

Элемент системы – объект, обладающий рядом свойств, обеспечивающих выполнение некоторых функций, внутреннее строение которых для целей моделирования не интересно. Части системы, состоящие из более чем одного элемента и имеющие собственную подцель, называются подсистемами.

Но далеко не все модели достигают этого этапа. Так, при исследовании черных дыр доступен только первый этап, то есть можно исследовать только взаимосвязь определенных входных и выходных данных.

3. Этап «Модели структуры системы» предполагает установление связей между элементами системы.

Связью между элементами называется процесс их взаимодействия, существенный для целей моделирования. Совокупность необходимых и достаточных для достижения целей моделирования связей между элементами называется структурой системы.

Структура системы, как правило, представляется в виде графа, в котором его вершины – элементы системы, ребра – связи. В некоторых моделях выделяют направления связей, тогда создается ориентированный граф, например при создании модели транспортной сети. Ребрам графа можно приписать веса, что позволяет учесть загрузки определенных направлений. В этом случае создается взвешенный граф. Для лучшей иллюстрации динамики процесса используют раскраску, то есть обозначения вершин или ребер. Некоторые типы структур являются очень распространенными и важными для практических целей и поэтому получили специальные названия.

Так, для описания и моделирования систем управления используют организационные системы, которые представляются тремя структурами, – линейной, древовидной и матричной.

При организации процесса моделирования можно выделить два подхода: классический и системный.

Классический подход в моделировании основан на принципе «от частного к общему» и синтезирует систему слиянием ее элементов, разработанных ранее независимо друг от друга. Моделируемый объект разбивается на отдельные подсистемы, выбираются исходные данные, ставятся цели моделирования для каждой части системы. Система собирается из отдельных частей – компонентов. Каждый из компонентов решает свою задачу и не зависит от других, взаимосвязь компонентов не учитывается.

При таком подходе невозможно смоделировать сложную систему, в которой элементы взаимосвязаны и влияют друг на друга. Поэтому классический подход применим к простым системам. Часто его используют вынужденно, когда часть элементов системы уже смоделирована «предшественниками» и автору приходится включать их в создаваемую систему.

Системный подход использует принцип «от общего к частному», то есть система рассматривается как интегрированное целое, состоящее из подсистем, связанных между собой. На основе исходных данных и цели моделирования формируются исходные требования к модели системы. На базе этих требований формируются подсистемы. Но далеко не всегда можно и нужно включать все проектируемые подсистемы в конечную систему, поскольку часто они удовлетворяют противоречащим требованиям. Поэтому производится выбор из набора сформированных подсистем на основе определенных критериев выбора и формируются элементы системы, которые формируют модель. Очевидно, что такой подход применим к сложным системам и учитывает взаимосвязь компонентов системы [1, 2].

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ. ОСНОВНЫЕ ПОНЯТИЯ

Итак, если мы имеем дело с системой, для исследования свойств которой мы способны составить математическое описание всех ее процессов, то мы получаем аналитическую модель.

Если время в аналитической модели квантовано, то получаем F- или P-модели, которые описываются конечными автоматами.

Если время в аналитической модели непрерывное, то в зависимости от того, детерминированы ли входные данные модели, получаем D-модели для детерминированных данных и Q-модели для вероятностных данных. Именно аналитические модели с вероятностными данными и (или) процессами, то есть Q-модели, и являются по сути своей имитационными моделями, но с некоторыми оговорками. Имитационная модель совсем не обязана иметь полное математическое описание всех процессов. Таким образом, имитационные модели включают в себя множество Q-моделей, но охватывают гораздо более широкий спектр моделирования.

Имитационная модель строится на основе перехода системы из одного состояния в другое. Предполагается, что известен набор правил, по которым определяется следующее состояние системы на основе текущего. В каждом состоянии системы получается набор выходных параметров.

Имитационная модель позволяет анимировать динамику системы, что гораздо нагляднее любых таблиц и графиков. Кроме того, в ходе ИМ можно получить значение любой величины в системе в любой момент модельного времени.

Имитационное моделирование имеет очень широкий круг применения: от моделей с высоким уровнем абстракции, в которых детали не важны и целью моделирования является стратегическое развитие системы (например, моделирование социальных систем, экосистем и т. д.), до моделей с детальной проработкой отдельных объектов (например, движение пешеходов, военные игры, склады и т. д.).

В ИМ используются три основных подхода – дискретно-событийное моделирование, агентное моделирование, системная динамика. Дискретно-событийное применяется для моделирования систем с низким уровнем абстракции, в которых для целей моделирования важны свойства отдельных элементов системы. Системная динамика применяется для моделирования систем, для которых важны стратегические цели, а не свойства элементов. Агентное моделирование является универсальным инструментом и может быть применено практически для моделей любого уровня абстракции [1, 2].

Остановимся более подробно на каждом из перечисленных подходов.

Дискретно-событийное моделирование рассматривает систему как последовательность мгновенных событий. Событие – это важный с точки зрения целей моделирования момент в системе. Например, нужно промоделировать работу банка со счетами. Важным моментом с точки зрения цели моделирования будет открытие счета, что и станет событием. События в системе происходят мгновенно и могут вызывать другие события. Например, событие «открытие счета» вызовет событие «обращение к базе данных банка» для идентификации клиента. Время меняется дискретно от события к событию, то есть в дискретно-событийном моделировании длительные процессы не предусмотрены. Но тогда как же моделировать реальные процессы, которые, как правило, не мгновенны? Для этого в дискретно-событийном моделировании имитируется задержка времени на заданный промежуток, который равен длительности процесса. Другой вопрос: а что будет с событиями, если они должны произойти в один момент времени? Ответ прост – такие события сериализуются (запускаются одно за другим). И поскольку события происходят мгновенно, то для системы это будет один момент времени.

При реализации дискретно-событийного моделирования в различных средах моделирования используется некий пассивный элемент с набором параметров, которые могут меняться. Этот пассивный элемент чаще всего называется заявкой. Заявки в модели могут создаваться, обслуживаться, накапливаться, менять маршрут, порождать новые заявки и удаляться. Под заявкой может пониматься как клиент, который пришел на обслуживание,

так и деталь, которую нужно обработать. Например, клиент пришел в парикмахерскую сделать стрижку. Тогда заявкой является клиент, а стрижка будет процессом обслуживания заявки. Для того чтобы обслуживание заявки состоялось, нужны свободное кресло и свободный мастер. Кресло и парикмахер являются ресурсами процесса, т. е. стрижки. Если ресурс занят, то процесс не может быть запущен, и тогда заявка ожидает своей очереди на обслуживание. После завершения всех работ над заявкой она удаляется из системы, что имитирует уход клиента из парикмахерской.

В агентном моделировании процесс функционирования модели представляется как результат взаимодействия агентов. Агент – это класс с заданным поведением. Поведение класса задается конечным автоматом, где состояния агента являются состояниями автомата, а переход между состояниями задается по какому-либо условию. Также поведение агента может быть задано обычными для объектно-ориентированного подхода средствами: событиями, функциями и пр.

Агенты помещаются в среду, в которой устанавливаются связи между ними, что позволяет им пересылать сообщения между собой. При получении сообщения агент может переходить в новое состояние, выполняться какое-либо событие агента и т. д. Среда жизни агента может быть дискретной и непрерывной. Дискретная среда подобна шахматной доске – агент может находиться строго в заданных точках среды. Непрерывная среда аналогична реальному миру, и агент может быть в любой ее точке. Есть некоторые стандарты расположения агентов в среде.

При случайном расположении агентов координаты их начального положения создаются случайным образом, при упорядоченном – начальное положение агентов в среде задается в узлах сетки пространства, при кольцевом – по кругу.

В дискретной среде начальное расположение агентов может быть случайным или упорядоченным.

Сети, в которые связаны агенты, также имеют стандартные конфигурации: случайное, решеточно-упорядочное кольцо, малый мир и согласно расстоянию.

В сети случайной конфигурации агенты связаны между собой случайным образом. В сети «решеточно-упорядочное кольцо» агенты, расположенные по кругу, связаны между собой через одного, в сети «малый мир» – через одного и в малые группы (агенты расположены по кругу). В сети, согласно расстоянию, агенты связаны с ближайшими соседями.

Все начальные расположения агентов и стандартные сети могут быть изменены в ходе моделирования.

Рассмотрим пример применения агентного подхода для моделирования работы парикмахерской. Агентами в этой модели будут клиент, мастер, кресло, уборщица. Для первого диаграмма будет содержать два состояния – ожидание обслуживания и собственно обслуживание. Для агентов-ресурсов (мастер, кресло, уборщица) диаграмма также будет состоять из двух состояний – занят и свободен.

При входе в парикмахерскую агент «клиент» переходит в состояние «Ожидание обслуживания» до тех пор, пока агенты «мастер» и «кресло» не перейдут в состояние «Свободен». Когда агенты «мастер» и «кресло» будут в состоянии «Свободен», агент «клиент» захватывает ресурсы и переходит в состояние «Обслуживание». В это же время агенты «мастер» и «кресло» переходят в состояние «Занят». Спустя заданное на обслуживание время агент «клиент» покидает парикмахерскую, а агенты «мастер» и «кресло» переходят в состояние «Свободен». По окончании стрижки агент «мастер» посылает сообщение «Вызов» агенту «уборщица», и этот агент переходит в состояние «Занят», из которого агент «уборщица» выходит спустя заданное время.

В основе системной динамики лежат два основных понятия: уровень и поток (оценка уровня). Уровень является очень широким понятием, но основное его значение – это возможность изменения (накопления) значения. Под уровнем можно понимать как количество товаров на складе, так и уровень стресса сотрудников или уровень удовлетворения потребностей клиентов и т. д. Поток – это скорость изменения уровня. Величина потока опреде-

ляется функциями решения (вентилями), линиями задержки и вспомогательными переменными. Функции решения (вентили) определяют зависимость величины потока от уровня. Например, уровень удовлетворенности клиентов банка определяет то, как они отзовутся о банке и, следовательно, сколько клиентов придет в банк. Значит, в формулу расчета величины потока клиентов нужно включить величину уровня удовлетворенности клиентов банка. Линии задержки показывают, как быстро начнет работать поток, то есть отражают время на раскрутку проекта или процесса.

Примером системы, удобной для моделирования методом системной динамики, может служить экспертная модель продаж, где оценивается эффективность рекламы по разным каналам – устная реклама, реклама по телевидению, контекстная реклама. Одним накопителем-уровнем задается количество потенциальных покупателей, другим – количество купивших клиентов. Потоком будет служить скорость перехода из потенциальных покупателей в клиенты, то есть продажи. Продажи идут за счет рекламы и «сарафанного радио» (устной рекламы), то есть благодаря тем людям, которые уже купили товар и рекомендуют его своим знакомым. Тогда объем продаж будет зависеть от оплаченной рекламы и «сарафанного радио», которые являются причинными зависимостями. Добавив возможность менять количество и рекламных роликов на телевидении, и контекстной рекламы в интернете, можно оценить изменение скорости продаж и понять, какой из каналов рекламы более эффективен [1, 2].

СИСТЕМА МОДЕЛИРОВАНИЯ ANYLOGIC

Общие сведения

Современные системы моделирования поддерживают весь арсенал новейших информационных технологий, включая развитые графические оболочки для конструирования моделей и интерпретации выходных результатов моделирования, мультимедийные средства, анимацию в реальном масштабе времени, объектно-ориентированное программирование, Internet-решения и др.

В данном пособии описываются методы и приемы построения моделей с помощью инструментальной системы AnyLogic.

Система AnyLogic, разработанная компанией «Экс Джей Текнолоджис» (XJ Technologies, г. Санкт-Петербург), – это среда компьютерного моделирования общего назначения. Это отечественный профессиональный инструмент нового поколения, поддерживающий на единой платформе основные направления ИМ: дискретно-событийные, непрерывные, системной динамики, агентные.

AnyLogic был разработан на основе новых идей в области информационных технологий, теории параллельных взаимодействующих процессов и теории гибридных систем. Благодаря этим идеям чрезвычайно упрощается построение сложных имитационных моделей, имеется возможность использования одного инструмента при изучении различных стилей моделирования.

Построенная на их основе инструментальная система AnyLogic не ограничивает пользователя одной единственной парадигмой моделирования, что является характерным для существующих на рынке инструментов моделирования.

Программный инструмент AnyLogic основан на объектно-ориентированной концепции. Другой базовой концепцией является представление модели как набора взаимодействующих, параллельно функционирующих активностей. Активный объект в AnyLogic – это объект со своим собственным функционированием, взаимодействующий с окружением. Он может включать в себя любое количество экземпляров других активных объектов.

Среда имитационного моделирования AnyLogic написана на языке Java, что делает ее кроссплатформенной. Особенностью среды AnyLogic является открытый код, который дает широкие возможности пользователю среды. После создания модели процесса в AnyLogic инструменты среды создают симуляцию работы модели в виде анимации. Такой подход делает

результаты работы более наглядными и понятными для пользователей. Например, после создания модели работы кафе можно запустить симуляцию ее работы, где все элементы (официанты кафе, посетители) будут двигаться, садиться за столики, уходить и пр.

Симуляцию (эксперимент) можно выгрузить из модели в виде Java-аплета и запускать его в любом из браузеров.

Среда AnyLogic по своей сути является полностью агентно-ориентированной. Точкой старта является агент Main, создание которого происходит по умолчанию. Именно на этом агенте размещаются все основные элементы модели. Пользователь среды может создавать новых агентов и давать им любые имена, но список всех агентов модели всегда будет на агенте Main.

На сайте компании AnyLogic доступна для скачивания студенческая версия среды, функционал которой, к сожалению, органичен: в ней нельзя выгрузить симуляцию модели в виде Java-аплета, так как код этой версии является закрытым, что сильно «бьет по рукам» студентов и, пожалуй, самое неприятное – в ней нельзя создавать более 10 агентов.

Графическая среда моделирования поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов, оптимизацию параметров относительно некоторого критерия.

При разработке модели можно использовать элементы визуальной графики: диаграммы состояний (стейтчарты), сигналы, события (таймеры), порты и т. д.; синхронное и асинхронное планирование событий, библиотеки активных объектов.

Удобный интерфейс и многочисленные средства поддержки разработки моделей в AnyLogic делают не только использование, но и создание компьютерных имитационных моделей в этой среде моделирования доступными даже для начинающих.

При разработке модели на AnyLogic можно использовать концепции и средства из нескольких классических областей имитационного моделирования: динамических систем, дискретно-событийного моделирования, системной динамики, агентного моделирования. Кроме того, AnyLogic позволяет интегрировать различные подходы с целью получить более полную картину взаимодействия сложных процессов различной природы.

Применение AnyLogic дает возможность оценить эффект конструкторских решений в сложных системах реального мира. Кроме того, разработчик может гибко использовать различные уровни абстрагирования и различные стили и концепции и смешивать их при создании одной и той же модели.

AnyLogic имеет исключительно развитый базовый язык дискретного и смешанного дискретно-непрерывного моделирования, на основе которого и построена библиотека Enterprise Library, предоставляющая высокоуровневый интерфейс для быстрого создания дискретно-событийных моделей с помощью блок-схем.

Графическое представление систем с помощью блок-схем широко используется во многих важных сферах деятельности: производстве, логистике, системах обслуживания, бизнес-процессах, моделировании компьютерных и телекоммуникационных сетей и т. д. AnyLogic позволяет моделировать при помощи визуальных, гибких, расширяемых, повторно-используемых объектов, как стандартных, так и разработанных. Библиотека Enterprise Library содержит традиционные объекты: очереди, задержки, конвейеры, ресурсы и т. п., так что модель быстро строится в стиле «перетащить и оставить» (drag-and-drop) и гибко параметризуется.

Реализация стандартных объектов Enterprise Library открыта для пользователя, их функциональность может быть как угодно расширена, вплоть до создания собственных библиотек [3, 4].

Этапы компьютерного моделирования

Имитационное моделирование состоит из двух этапов: создания модели и анализа полученных с помощью результатов с целью принятия решения (табл. 1).

При детальном рассмотрении построение действительно полезной имитационной модели требует большой работы. Сначала разработчик определяет, какие задачи будут решаться с ее помощью, т. е. формулирует цель моделирования, от которой зависит то, какие

процессы в реальной системе следует выделить и отразить, а от каких процессов абстрагироваться, какие характеристики этих процессов учитывать, а какие – нет, какие соотношения между переменными и параметрами модели должны быть отражены в ней. Данный этап можно охарактеризовать как создание концептуальной (содержательной) модели. На нем происходит структуризация, т. е. выделение подсистем, определение элементарных компонентов модели и их связей на каждом уровне иерархии.

В ИМ структура модели отражает структуру реального объекта на некотором уровне абстракции, а связи между компонентами модели являются отражением реальных связей.

Элементы системы, их связи, параметры и переменные, а также их соотношения и законы их изменения должны быть выражены средствами среды моделирования, т. е. в ней должны быть определены переменные параметры модели, построены процедуры переменных и характеристик модели во времени.

Для понимания процессов, протекающих в модели, необходимо разработать анимационное представление этих процессов. Затем построенная модель должна быть проверена с точки зрения корректности ее реализации.

Следующий этап – это калибровка или идентификация модели, т. е. сбор данных и проведение измерений тех характеристик в реальной системе, которые должны быть введены в модель в виде значений параметров и распределений случайных величин.

Далее, необходимо выполнить проверку правильности модели (ее валидацию), заключающуюся в тестировании в нескольких режимах, в которых характеристики поведения реальной системы известны либо очевидны. Последним этапом работы с моделью является компьютерный эксперимент, т. е. собственно то, ради чего и создавалась модель.

Выполнение модели при различных значениях существующих параметров (факторов) и наблюдение поведения с регистрацией характеристик использования называется прогнозом, или экспериментом типа «что будет, если…» [1, 2].

Таблица 1

Этапы компьютерного моделирования

№ п/п	Название этапа	Результат
1	Анализ системы	Понимание того, что происходит в системе, подлежащей анализу, какова ее структура, какие процессы в ней протекают
2	Формулировка моделирования системы	Список задач, которые предполагается решить с помощью будущей модели. Список входных параметров модели, исходных данных, а также критерии завершенности будущего исследования
3	Разработка концептуальной структуры модели	Структура модели, состав существенных процессов, подлежащих отображению, зафиксированный уровень абстракции для каждой подсистемы (список допущений), описание управляющей логики для подсистем
4	Реализация модели в среде моделирования	Реализованные подсистемы, их параметры и переменные, их поведение, реализованная логика и связи подсистем
5	Реализация анимационного представления модели	Анимационное представление модели, интерфейс пользователя
6	Проверка корректности реализации модели	Уверенность в том, что модель корректно отражает те процессы реальной системы, которые требуется анализировать
7	Калибровка модели	Фиксация значений параметров, коэффициентов уравнений и распределений случайных величин, отражающих те ситуации, для анализа которых модель будет использоваться
8	Планирование и проведение компьютерного эксперимента	Результаты моделирования (графики, таблицы и т. п., дающие ответы на поставленные вопросы)

Компьютерное моделирование позволяет не только получить прогноз, но и определить, какие управляющие воздействия на систему приведут к благоприятному развитию событий. Более сложные эксперименты позволяют выполнить анализ чувствительности

модели, оценку рисков различных вариантов управляющих решений, а также оптимизацию для определения параметров и условий рационального функционирования модели.

Для анализа результатов моделирования в инструментальной среде, представления данных в структурированном или графическом виде, а также для интеграции с внешними базами данных и т. п. могут быть использованы специальные средства для обработки статистической информации.

Часто имитационная модель используется в качестве модуля большей системы принятия решения, получающей в режиме реального времени данные мониторинга состояния управляемой системы, оценивающей к каким последствиям может привести текущая ситуация, и предлагающей оптимальное (или просто рациональное) управляющее решение для минимизации отрицательных последствий развития системы в будущем. Для этого обычно требуется интеграция с другими информационными системами и разработка специального интерфейса пользователя. В AnyLogic две фазы ИМ – разработка модели и ее анализ – явно разделены. Разработка модели выполняется в среде редактора, анализ модели – 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 [1, 2].

Основные концепции моделирования

Разработка модели выполняется в графическом редакторе AnyLogic с использованием многочисленных средств поддержки, упрощающих работу, затем компилируется (встроенным компилятором) и запускается на выполнение. В процессе выполнения модели пользователь может наблюдать ее поведение, изменять параметры, выводить результаты моделирования в различных формах и экспериментировать [3–5].

Для реализации специальных вычислений и описания логики поведения объектов AnyLogic позволяет использовать мощный современный язык Java.

Основными строительными блоками модели AnyLogic являются *активные объекты*, которые позволяют моделировать любые объекты реального мира.

Класс в программировании является мощным средством, позволяющим структурировать сложную систему, определяет шаблон, в соответствии с которым строятся отдельные экземпляры класса (они могут быть определены как объекты других активных объектов).

Активный объект является экземпляром класса активного объекта. Чтобы построить модель AnyLogic, необходимо создать классы активных объектов (или использовать объекты библиотек AnyLogic) и задать их взаимосвязи. AnyLogic интерпретирует создаваемые графически классы активных объектов в классы Java, поэтому можно пользоваться всеми преимуществами объектно-ориентированного моделирования.

Активные объекты могут содержать вложенные объекты, причем уровень вложенности не ограничен. Это позволяет производить декомпозицию модели на любое количество уровней детализации.

Активные объекты имеют четко определенные интерфейсы взаимодействия: они взаимодействуют со своим окружением только посредством своих интерфейсных элементов. Это облегчает создание систем со сложной структурой, а также делает активные объекты повторно используемыми. Создав класс активного объекта, вы можете создать любое количество объектов – экземпляров данного класса.

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 [5].

При построении модели используются средства *визуальной разработки* (введения состояний и переходов стейтчарта, пиктограмм переменных и т. п.), задания численных

значений параметров, аналитических записей соотношений переменных и условий наступления событий.

Основной технологией программирования в AnyLogic является визуальное программирование – построение с помощью графических объектов и пиктограмм иерархий структуры и поведения активных объектов.

AnyLogic является надстройкой над языком Java – одним из самых мощных и в то же время самых простых современных объектно-ориентированных языков.

Все объекты, определенные пользователем при разработке модели с помощью графического редактора, компилируются в конструкции языка Java, а затем происходит компиляция всей собранной программы на Java, задающей модель, в исполняемый код. Хотя программирование сведено к минимуму, разработчику модели необходимо иметь некоторое представление об этом языке (например, знать синтаксически правильные конструкции).

Таким образом, хотя при построении модели на AnyLogic разработчик использует конструкции языка Java в большей или меньшей степени, в действительности он никогда не разрабатывает полные программы, а лишь вставляет фрагменты кода в специально предусмотренные для этого поля окна «Код» и окон свойств объектов модели. Эти фрагменты выражают логику конкретных шагов или действий [4, 5].

Основными средствами описания поведения объектов являются *переменные, события и диаграммы состояний*. Переменные отражают изменяющиеся характеристики объекта. События (или таймеры) могут наступать с заданным интервалом времени и выполнять заданное действие. Диаграммы состояний (или стейтчарты) позволяют визуально представить поведение объекта во времени под воздействием событий или условий, они состоят из графического изображения состояний и переходов между ними (т. е. по сути это конечный автомат).

Любая сложная логика поведения объектов модели 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 аналитически.

Интерпретация любого числа параллельно протекающих процессов в модели AnyLogic скрыта от пользователя. Никаких усилий разработчика модели для организации квазипараллелизма интерпретации не требуется; отслеживание всех событий 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 это 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 с помощью функции time.

Единицу модельного времени разработчик 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 одних и тех же единицах измерения физических величин [4, 5].

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 единица физического времени 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 макси-

мальной скоростью без привязки к физическому времени. Данный режим используется для факторного анализа модели, набора статистики, оптимизации параметров и т. д. Поскольку анимация и другие окна наблюдения за поведением обычно существенно замедляют скорость интерпретации модели на компьютере, для повышения скорости выполнения эти окна нужно закрыть.

В режиме реального времени пользователь задает связь модельного времени с физическим временем, т. е. устанавливает ограничение на скорость процессора при интерпретации модели. В этом режиме задается количество единиц модельного времени, которые должны интерпретироваться 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.

Соотношение физического и модельного времени можно понять на таком примере: при коэффициенте ускорения 4, если процессор успевает выполнить менее чем за 1 с все операции, которые определены в течение четырех единиц модельного времени, то он будет ждать до конца секунды; если же процессор не успевает выполнить все операции, то у него не будет интервала ожидания, и коэффициент ускорения будет меньше того, который установлен пользователем.

AnyLogic имеет удобные средства представления функционирования моделируемой системы в живой форме *динамической анимации*, что позволяет «увидеть» поведение сложной системы.

Визуализация процесса функционирования моделируемой системы позволяет проверить адекватность модели, выявить ошибки при задании логики.

Средства анимации позволяют пользователю легко создавать виртуальный мир (совокупность графических образов, ожившую мнемосхему), управляемый динамическими параметрами модели по законам, определенным пользователем с помощью уравнений и логики моделируемых объектов. Графические элементы, добавленные на анимацию, называются динамическими, поскольку все их параметры (видимость, цвет и т. п.) можно сделать зависимыми от переменных и параметров модели, которые меняются со временем при выполнении модели.

С помощью совершенной технологии визуализации работающих моделей AnyLogic можно создавать интерактивные анимации произвольной сложности, связывая графические объекты, в том числе импортированные чертежи, во встроенном редакторе с объектами модели. Как и модель, анимация имеет иерархическую структуру, которая может динамически изменяться. Возможно создание нескольких точек зрения или нескольких уровней детальности в пределах одной анимации. Элементы управления и развитая бизнес-графика превращают анимацию модели в настоящую панель управления для оценки эффективности решений. В AnyLogic поддерживается как двумерная, так и трехмерная анимация.

Многие системы моделирования позволяют менять параметры модели только до запуска модели на выполнение. AnyLogic позволяет пользователю вмешиваться в работу модели, изменяя ее параметры в процессе функционирования. Примером таких средств являются слайдеры, которые могут быть введены в окно анимации [3–5].

Система моделирования AnyLogic обеспечивает поддержку всех этапов ИМ для различных типов динамических моделей – дискретных, непрерывных и гибридных, детерминированных и стохастических в любых их комбинациях в рамках одного инструмента. Создание модели, ее выполнение, оптимизация параметров, анализ полученных результатов, верификация модели – все эти этапы удобно выполнять в среде AnyLogic.

Данный инструмент обладает большим спектром разнообразных возможностей проведения как отдельных прямых экспериментов типа «если, то», так и серий подобных экспериментов для решения всевозможных обратных задач, направленных на поиск параметров модели, оптимизирующих ее функционирование. Удобный интерфейс и различные средства поддержки разработки в AnyLogic делают не только использование, но и создание компьютерных имитационных моделей в этой среде моделирования доступными даже для тех, кто в области вычислительной техники и программирования не является профессионалом.

Пользовательский интерфейс

После первого запуска AnyLogic откроется окно справочной страницы среды, на этой странице представлены ссылки на учебные пособия и примеры моделей. Особое внимание хочется уделить сайту www.RunTheModel.com, куда можно выгружать модели AnyLogic, что дает возможность просматривать работу без использования самой среды. И это является существенной помощью студентам, работающим с версией среды, которая лишена возможности выгрузки эксперимента модели. Кроме того, сайт может быть полезен при обучении работе в среде, поскольку там есть множество работающих моделей, которые можно скачать. После закрытия начальной страницы откроется интерфейс среды [3].

В дальнейшем, после запуска AnyLogic, открывается рабочее окно, где для продолжения работы надо создать новый проект или открыть уже существующий.

Начиная с версии 6.4, AnyLogic предоставляет пользователям возможность использовать шаблоны при создании новых моделей. Чтобы начать новый проект, нужно нажать соответствующую кнопку на панели инструментов или выбрать пункт меню «Файл» | «Создать проект», затем из ниспадающего меню – «Модель». Откроется диалоговое окно «Новая модель», где задается имя и местоположение нового проекта. Далее необходимо следовать указаниям «Мастера создания модели». Можно создавать новую модель «с нуля» или использовать шаблон.

При открытии проекта (нового или существующего) AnyLogic всегда открывает среду разработки проекта – графический редактор модели (рис. 1).

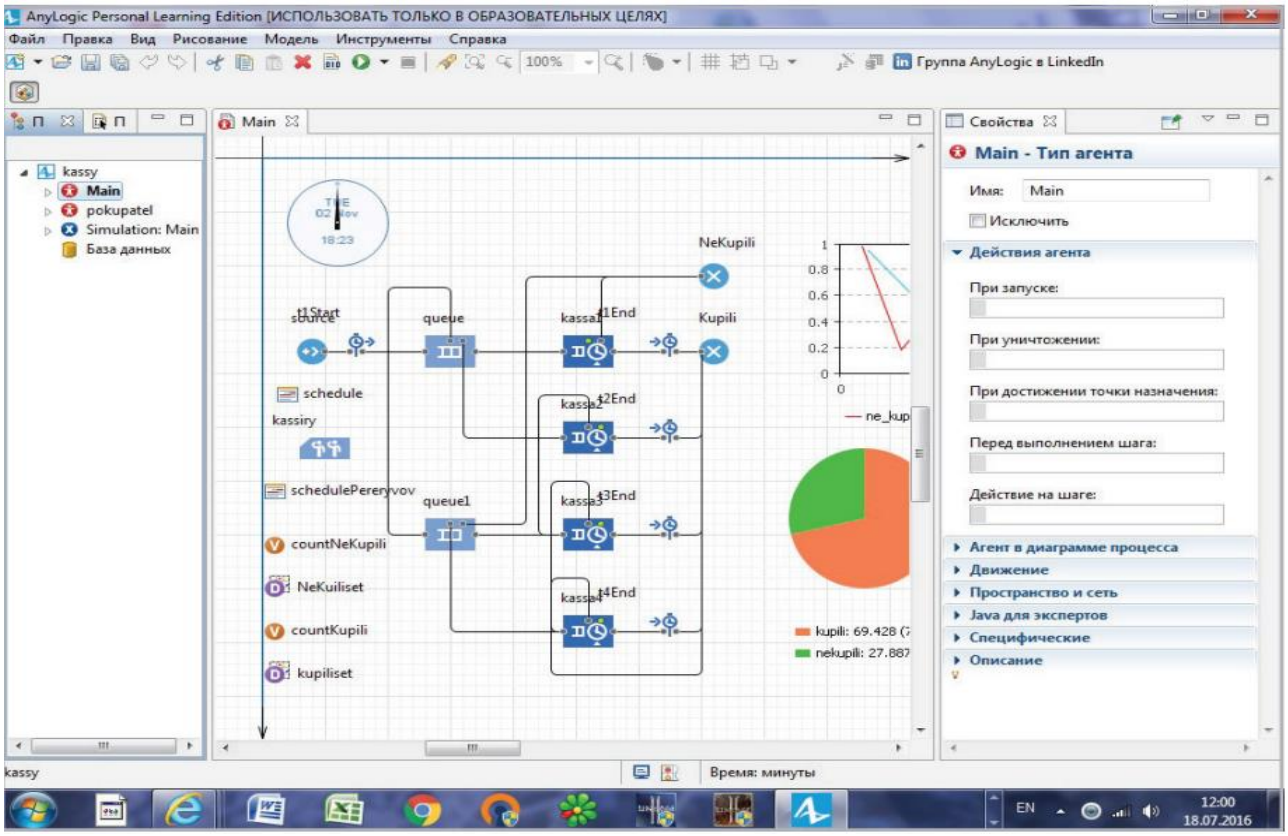

Рис. 1. Пользовательский интерфейс AnyLogic

Рассмотрим основные составляющие этого редактора.

Окно проекта обеспечивает легкую навигацию по элементам, таким как пакеты, классы и т. д. Поскольку проект организован иерархически, то он отображается в виде дерева: сам проект образует верхний уровень дерева рабочего проекта, пакеты – следующий уровень, классы активных объектов и сообщений – следующий и т. д. Можно копировать, перемещать и удалять любые элементы дерева объектов, легко управляя рабочим проектом.

Одна из ветвей в дереве проекта имеет название Simulation:Main (эксперимент). Эксперимент хранит набор настроек, с помощью которых конфигурируют работу модели. Один эксперимент – простой – создается автоматически при создании проекта. Этот простой эксперимент с именем Simulation, позволяющий визуализировать модель с помощью анимации и поддерживающий инструменты для отладки модели, используется в большинстве случаев. Поддерживается еще несколько типов экспериментов для различных задач моделирования [5].

Структурная диаграмма. При построении модели нужно задать ее структуру (т. е. описать, из каких частей состоит модель системы) и поведение отдельных объектов. В AnyLogic структурными элементами модели являются так называемые активные объекты. Активный объект имеет структуру и поведение. Элементы структуры – это другие активные объекты, включенные как составные элементы данного активного объекта, и связи, которые существуют между включенными активными объектами. Активные объекты могут содержать: события, стейтчарты, переменные, функции, уравнения, параметры. Их структура задается графически на структурной диаграмме, поведение задается с помощью стейтчарта и определяет реакции активного объекта на внешние события – логику его действий во времени.

Диаграмма состояний (или стейтчарт – statechart) – это модифицированные графы переходов конечного автомата. Стейтчарт позволяет графически задать пространство состояний алгоритма поведения объекта, а также события, которые являются причинами срабатывания переходов из одних состояний в другие, и действия, происходящие при смене состояний. Стейтчарты соответствуют стандарту UML. Они сохраняют графический вид, атрибуты и семантику выполнения, определенную в UML (Unified Modeling Language). Стейтчарты в AnyLogic поддерживают следующие типы событий: сигнал – объект может послать сигнал другому объекту, чтобы уведомить его о чем-то; тайм-аут – в течение заданного промежутка времени в стейтчарте ничего не происходит; событие – событие, при котором значение булева выражения становится «истина».

Кроме того, в окне редактора для модели можно построить двумерное или трехмерное анимационное представление, которое помогает понять, что происходит с моделью во времени. Элементы анимационной картинки имеют свои параметры, они могут быть связаны с переменными и параметрами модели. Изменение переменных модели во времени ведет к изменению графического образа, что позволяет пользователю наглядно представить динамику моделируемой системы с помощью динамически меняющейся графики.

Окно свойств. В редакторе AnyLogic для каждого выделенного элемента модели существует свое окно свойств, где указываются свойства (параметры) этого элемента. Окно свойств содержит несколько вкладок, каждая из них включает в себя элементы управления, такие как поля ввода, флажки, переключатели, кнопки и т. д., с их помощью которых можно просматривать и изменять свойства элементов модели. Число вкладок и их внешний вид зависит от типа выбранного элемента.

Окно палитры содержит элементы (графические объекты), которые могут быть добавлены на структурную диаграмму. Элементы разбиты по группам, отображаемым на разных вкладках. Чтобы добавить объект палитры на диаграмму, нужно «щелкнуть» сначала по элементу в палитре, а затем по диаграмме.

Параметры. Активный объект может иметь параметры. Обычно они используются для задания характеристик объекта. Можно задать различные значения параметров для разных объектов одного и того же класса, что требуется в тех случаях, когда объекты имеют одинаковое поведение, но их характеристики разные. Возможно описание параметров любых Java-классов.

Чтобы создать параметр класса активного объекта, в окне «Проект» нужно нажать на иконку класса активного объекта; в окне «Свойства» – на кнопку «Новый параметр», после чего задать свойства параметра в открывшемся диалоговом окне «Параметр».

Переменные. Активный объект может содержать переменные, которые могут быть либо внутренними, либо интерфейсными. Активный объект может иметь переменные, моделирующие, меняющиеся во времени величины.

Переменные могут быть вынесены в интерфейс активного объекта и связаны с переменными других активных объектов. Тогда при изменении значения одной переменной будет немедленно меняться и значение связанной с ней зависимой переменной другого объекта. Этот механизм обеспечивает непрерывное и/или дискретное взаимодействие объектов.

Передача сообщений. AnyLogic позволяет передавать информацию от одного объекта другому с помощью специальных пакетов данных – сообщений, благодаря чему можно реализовать механизм оповещения – сообщения будут представлять команды или сигналы, посылаемые системой управления. Можно также смоделировать поток заявок, в этом случае сообщения будут представлять собой заявки – объекты, которые производятся, обрабатываются, обслуживаются или еще каким-нибудь образом подвергаются воздействию моделируемого процесса (документы – в модели бизнес-процесса, клиенты – в модели системы массового обслуживания, детали – в производственных моделях).

Сообщения принимаются и посылаются через специальные элементы активных объектов – порты. Обмен сообщениями возможен только между портами, соединителями – элементами, играющими роль путей движения сообщений.

Чтобы соединить порты вложенных объектов, необходимо нажать на кнопку «Соединитель» в панели инструментов, а затем – поочередно по обоим портам. Чтобы добавить точку изгиба, нужно нажать на кнопку «Редактировать точки» в панели инструментов.

Запуск и просмотр модели. Запускать и отлаживать модель можно с помощью меню «Модель» и панели инструментов.

При исполнении модели запустится компилятор, который построит исполняемый код модели в языке Java, скомпилирует его и затем запустит модель на исполнение.

Для запуска модели необходимо нажать на кнопку «Выполнить», затем выбрать эксперимент из выпадающего списка. После этого откроется окно, отображающее созданную презентацию для запущенного эксперимента. Необходимо щелкнуть по кнопке, чтобы запустить модель и перейти на презентацию.

В окне презентации можно увидеть: анимированную диаграмму модели, окна инспекта элементов модели, ожившую анимацию, диаграммы состояний, графики статистики [5].

Проведение экспериментов

С помощью экспериментов задаются конфигурационные настройки модели. AnyLogic поддерживает несколько типов экспериментов: простой эксперимент, эксперимент для варьирования параметров, оптимизационный и др. На рис. 2 показано окно выбора эксперимента.

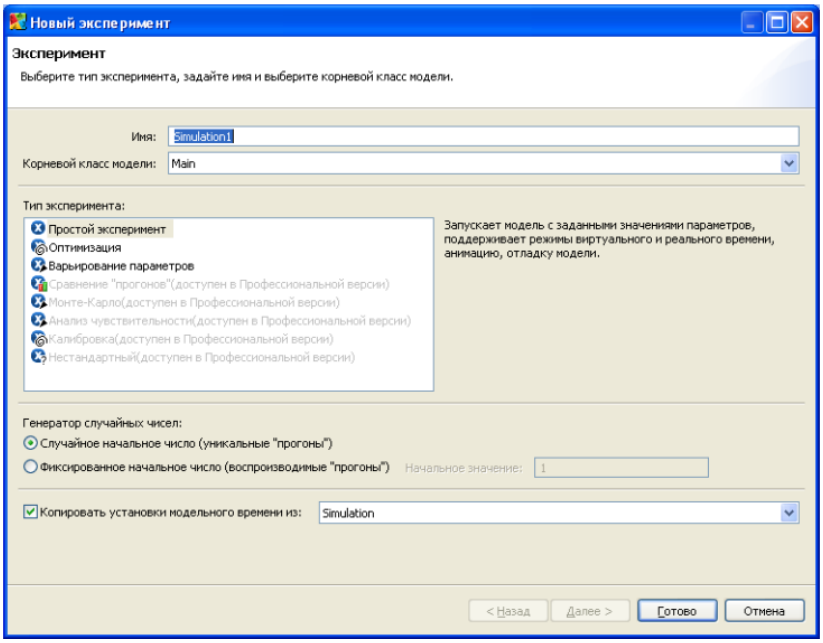

Рис. 2. Окно «Эксперимент»

Простой эксперимент. Задачи вида «что, если» (так называемая прямая задача ИМ) в AnyLogic решаются с помощью «простого эксперимента» (с именем Simulation), который создается автоматически при открытии проекта. Он позволяет визуализировать модель с помощью анимации, графиков (диаграмм) и т. п. Широкие возможности для отображения данных предоставляет библиотека бизнес-графики (Business Graphics Library).

Для построения, например, графика зависимости переменных от времени в поле анимации сначала нужно построить прямоугольник, в переделах которого будет размещаться график, после чего в любое место поля редактора перенести экземпляр объекта ChartTime из Business Graphics Library. Затем в окне свойств данного объекта следует настроить параметры, определяющие цвет и толщину линий, имена отображаемых переменных, названия переменных, которые будут отображаться, цвет текста и т. д.

Простой эксперимент используется в большинстве случаев при разработке и анализе моделей, созданных в AnyLogic. В частности, он поддерживает средства для отладки. Можно организовать несколько простых экспериментов с различными значениями исходных факторов и, сделав один из этих экспериментов текущим, запустить модель на выполнение [4].

Эксперимент для варьирования параметров. Анализ чувствительности модели – процедура оценки влияния исходных гипотез и значений ключевых факторов на выходные показатели модели. Обычно эксперимент с варьированием параметров и анализом реакции модели помогает оценить, насколько чувствительным является выдаваемый моделью прогноз к изменению гипотез, лежащих в основе модели. При анализе чувствительности обычно рекомендуется выполнять изменение значений факторов по отдельности, что позволяет ранжировать их влияние на результирующие показатели.

В AnyLogic доступен механизм автоматического запуска модели заданное количество раз с изменением значений выбранных параметров – это эксперимент для варьирования параметров, при запуске которого пользователь может изучить и сравнить поведение модели при разных значениях параметров с помощью графиков.

Чтобы запустить такой эксперимент, нужно выполнить следующие действия:

- создать эксперимент для варьирования параметров;

- сконфигурировать эксперимент, выбрав параметры, которые необходимо изменить, и, задав значения, которые эти параметры должны будут принять за определенное вами количество прогонов модели, в окне свойств данного эксперимента, запустить модель, выбрав данный эксперимент в качестве текущего.

Такой вид эксперимента не поддерживает визуализацию с помощью анимации.

Оптимизационный эксперимент используется для решения задач количественного анализа (расчет показателей эффективности системы). Поиск тех значений факторов, которые определяют наиболее предпочтительный вариант решения, называется обратной задачей ИМ. Такие задачи отвечают на вопрос о том, какое решение из области допустимых решений обращает в максимум показатель эффективности системы. Для решения обратной задачи многократно решается прямая задача. В случае, когда число возможных вариантов решения невелико, решение обратной задачи сводится к простому перебору всех возможных решений: сравнивая их между собой, можно найти оптимальное.

Если перебрать все варианты решений невозможно, то используются методы направленного перебора с применением эвристик. При этом оптимальное или близкое к оптимальному решение находится после многократного выполнения последовательных шагов (решений прямой задачи и нахождения для каждого набора входных параметров модели вектора результирующих показателей). Правильно подобранная эвристика приближает эксперимент к оптимальному решению на каждом шаге.

В качестве блока регистрации значений выходных показателей и выбора очередного приближения при оптимизации пользователь может использовать любой внешний оптимизатор или же оптимизатор OptQuest (встроен в AnyLogic), разработанный на основе метаэвристик рассеянного поиска (scatter search) и поиска «табу» (tabu search). OptQuest является лучшим для решения сложных проблем оптимизации, запускается прямо из среды разработ-

ки модели. Чтобы настроить оптимизацию в AnyLogic необходимо выполнить следующие действия:

- создать в разработанной модели оптимизационный эксперимент;
- задать оптимизационные параметры и области их изменения;
- задать условие остановки модели после каждого прогона. Это может быть либо остановка по времени выполнения прогона, либо остановка по условиям, накладываемым на переменные модели;
- задать целевую функцию, т. е. исследуемую реакцию системы;
- задать ограничения, которые в конце каждого прогона определяют, допустимо ли значение вектора исходных входных факторов. Ограничения можно не задавать (т. е. это опционально);
- задать условия прекращения эксперимента.

После запуска модели оптимизационный эксперимент найдет наилучшие значения входных параметров, при которых заданная целевая функция обратится в минимум или максимум [4].

СПИСОК ИСТОЧНИКОВ

1. Асадуллаев Р. Г. Моделирование систем : учеб. пособие. Белгород, 2016. 236 с.
2. Бобков С. П., Бытев Д. О. Моделирование систем : учеб. пособие. Иваново, 2008. 156 с.
3. Карпов Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. СПб. : БХВ-Петербург, 2005. 400 с.
4. Киселева М. В. Имитационное моделирование систем в среде AnyLogic : учеб.-метод. пособие. Екатеринбург : УГТУ; УПИ, 2009. 88 с.
5. Лимановская О. В. Имитационное моделирование в AnyLogic 7 : учеб. пособие. В 2 ч. Екатеринбург : Изд-во Урал. ун-та, 2017. Ч. 1. 152 с.

Учебное издание

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ AnyLogic.
Часть 1

Учебно-методическое пособие

Составители
Бушмелева Кия Иннокентьевна
Никифоров Антон Владимирович

Редактор Ю. Р. Бобрус
Верстка Е. А. Мельниковой

Дата опубликования 30.01.2023. Формат 60 × 84/8
Уч.-изд. л. 2,2. Заказ № 442/22

Оригинал-макет подготовлен
в Издательском центре СурГУ
Тел. (3462) 76-31-79

БУ ВО «Сургутский государственный университет»
628400, Россия, Ханты-Мансийский автономный округ,
г. Сургут, пр. Ленина, 1
Тел. (3462) 76-31-00