Лабораторная работа 3

Разработка модели СМО с помощью библиотеки элементов

Цель работы: научиться описывать модели систем массового обслуживания с помощью инструментов AnyLogic.

Задание: Ознакомьтесь с теоретическим материалом и примером в этом файле, затем разберите примеры в файле **СМО_Задачи_тренировочные+комментарии** и попробуйте построить модели, используя комментарии. Если возникнут затруднения, обратитесь к файлам с пошаговым выполнением некоторых тренировочных задач. Справочное руководство по библиотеке SMO содержится в файле **SMO_min.**

После выполнения тренировочных задач необходимо построить модель объекта согласно своему варианту по описанию в файле **СМО_индивидуальные_задания.**

Теоретические сведения

Одним из очевидных путей упрощения построения моделей в некоторой специфической области применения является выделение и разработка библиотеки типовых блоков, часто встречающихся в моделях данного типа. Эти блоки затем могут быть использованы в конкретных моделях с необходимой настройкой параметров. В AnyLogic разработано несколько библиотек для таких областей, как моделирование производственных процессов *(Enterprise Library),* моделирование потоков различных материалов *(Material Flow Library),* моделирование в области системной динамики *(Dynamic Systems Library)* и др.

Для систем массового обслуживания, логистики, бизнес-процессов и широкого класса других событийно-дискретных систем библиотека *Enterprise Library* пакета AnyLogic предоставляет множество модулей, которые с помощью настройки параметров могут существенно ускорить разработку моделей в этих областях. Несколько расширенные и дополненные функции представляет библиотека *SMO*. Рассмотрим пример модели (рис.1).

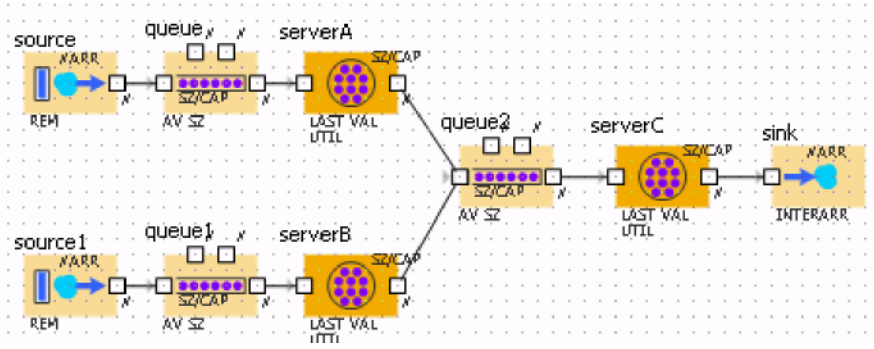


Рис.1. Архитектура модели СМО

Создание архитектуры модели

Откройте новый проект с именем *QueuingModelLibrary* и кликнув на

корневом объекте *Model,* откройте окно редактора структуры этого объекта. В левом окне классов откройте вкладку *Библиотеки* вместо открытой по умолчанию вкладки *Проект* и выберите библиотеку *SMO.* Нам нужны будут следующие четыре типа блоков из этой библиотеки: генератор (источник) - *Source,* очередь -*Queue,* сервер, который, фактически, будет выполнять только функции задержки -*Delay,* и сток - *Sink.* В окно редактора структуры объекта *Model* перетащите из библиотеки эти элементы и соедините их порты в соответствии со структурой рис.1. Три экземпляра класса *Delay* назовите *serverA, serverB* и *serverC.*

Архитектура модели готова. Запустите ее на выполнение, предварительно сняв ограничение на остановку модели по времени в окне свойств ***объекта Simulation.***

Настройка параметров

Каждый блок библиотеки имеет несколько предварительно определенных параметров, которые следует установить для конкретной модели. Для каждого типа блоков эти параметры свои. Например, для источников заявок важна интенсивность порождения заявок *interarrivalTime* (интервал времени между генерацией последовательных сущностей), для очереди должен быть указан ее максимальный объем *(capacity),* во многих блоках могут быть указаны операции, которые будут выполняться по приходе (в поле *onEnter)* и по выходе (в поле *onExit)* каждой заявки. Эти операции представляются кодом на языке Java.

Настройка источников

Пересылаемый по системе объект называется в этой библиотеке сущностью, или заявкой *(entity).* В источниках заявок *source* и *source1* для нас важна интенсивность порождения заявок. Источник заявок обычно стоит на входе в систему, поэтому этот параметр в источнике называется *interarrivalTime (ВремяМеждуПрибытиями).* Параметр этот - динамический, его можно задать любым распределением. В соответствии с заданием, установите в одном источнике *(source)* это время как распределенное по экспоненциальному закону со средним 16.5 (т.е. *exponential(1.0/16.5)*), в другом источнике *(source1)* - как *exponential(1.0/12.5).*

Настройка очередей

В соответствии с заданием, во всех трех очередях установите бесконечным объем очереди - для параметра *capacity* выберите значение *Infinity.*

Настройка серверов (блоков задержки)

Здесь нужно задать время задержки *(delayTime)* как *exponential с* нужными параметрами: у блока *server A* с параметром 1.0/15.0, у блока *serverB* с параметром 1.0/10, у блока *serverC* с параметром 1.0/6.5. Для параметра *capacity* выберите значение 1.

Сбор статистики

Некоторые характеристики параметров модели видны уже на простой анимированной блок-схеме, которая строится автоматически. Окно с этой блок-схемой можно открыть двойным щелчком мыши по корневому объекту *root* в окне классов.

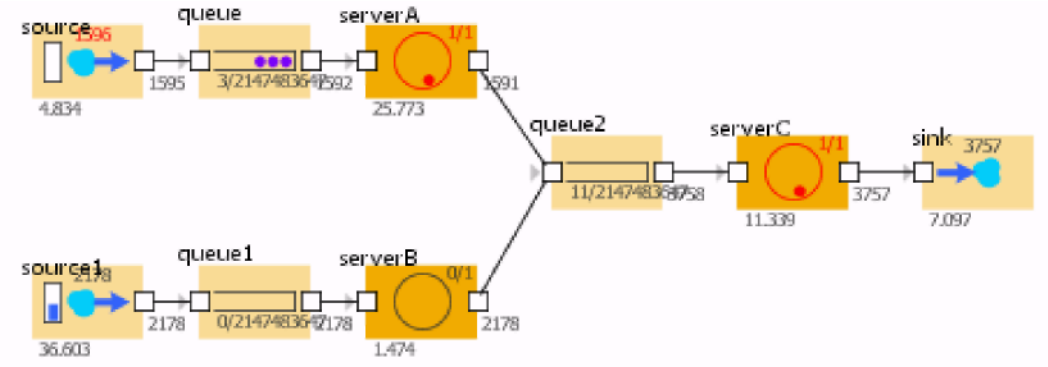


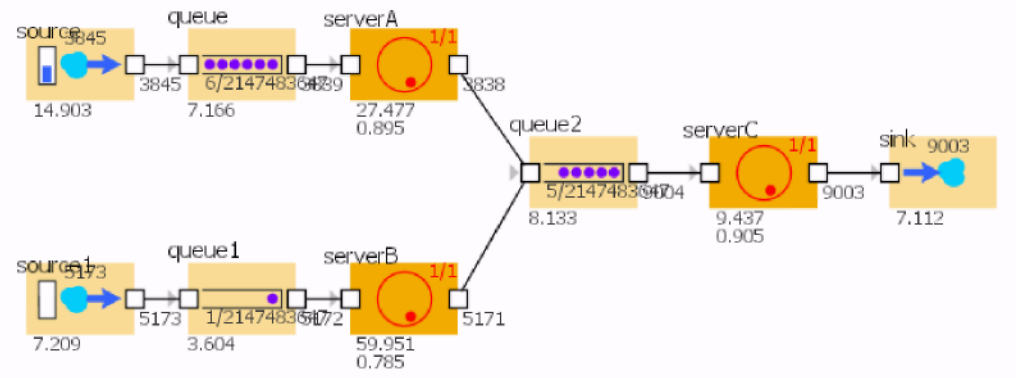
Рис.2. Анимированная структура модели СМО

На рис.2 видно, что к настоящему моменту источник *source* сгенерировал 1595 заявок, в текущий момент до генерации очередной заявки остается 4.834 сек. Далее, в очереди *queue* есть три заявки, в очереди *queuel* - нет заявок, а в *queue2* находится 11 заявок. Число 214748..., стоящее через слэш у очереди — это максимальная длина очереди (максимальное целое, представимое в вашей машине). Мы определили эту длину как Infinite. В сервере *serverA* обрабатывается одна заявка, последняя заявка обрабатывалась 25.773 единицы времени. В сервере *serverB* - нет ни одной, всего обработано 3757 заявок, средний интервал между выходящими из системы заявками 7.097.
Для сбора статистики нужно предпринять некоторые усилия, однако они, конечно, значительно меньшие, чем это нужно без использования библиотечных объектов.

Автоматически собираемая статистика

В блоках библиотеки предусмотрен сбор «базовой» статистики, т.е. определены наборы данных, отражающие наиболее часто используемые случайные параметры этих блоков. Так, например, для очереди таким параметром является длина очереди, для блока задержки - коэффициент его использования. Подсчет статистики во всех блоках библиотеки по умолчанию отключен для повышения производительности, при необходимости его можно включить.

В каждой из трех очередей и в каждом из трех серверов включите сбор статистики. Для этого в каждом из блоков выберите *true* из выпадающего меню в качестве значения параметра *statsEnabled*. После запуска модели на выполнение большая часть требуемой в задаче статистики уже будет видна на анимированной блок-схеме:



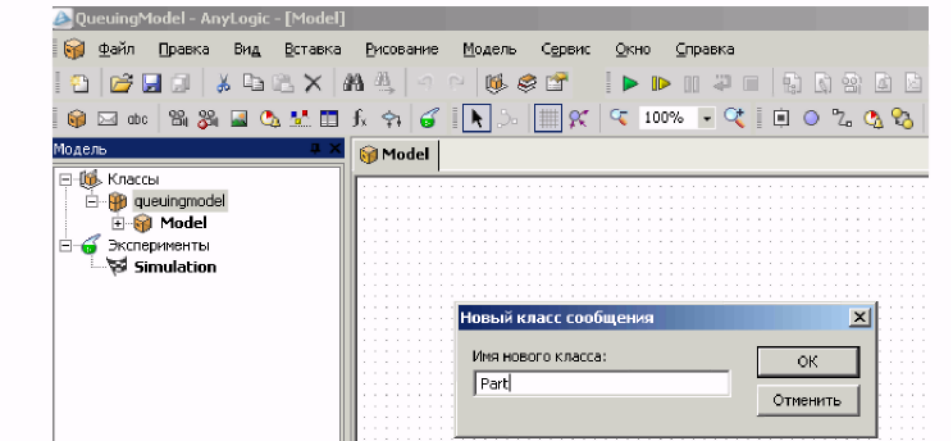
В частности, на этой блок-схеме видно, что после обработки 9 тысяч заявок средние длины очередей 7.166, 3.604 и 8.133 соответственно, коэффициенты использования серверов A, B и C 89.5%, 78.5% и 90.5%, средний интервал

выхода заявок после обработки в системе 7.112 (и, следовательно, производительность - среднее число обработанных заявок в единицу времени — 1/7.112).

Заметьте, что получение этих данных с использованием библиотеки *SMO* потребовало лишь нескольких минут.

Временная статистика прохождения заявок в системе

Определите новый класс сообщений, который назовите *Part* (деталь). Для этого в дереве классов проекта кликните правой кнопкой мыши по имени проекта и в появившемся контекстном меню выберите команду *Новый класс сообщения*. В появившемся диалоговом окне наберите имя *Part* в поле *Имя нового класса.*



Введите новый тип заявки — **Part**, который будет наследоваться от класса *Entity*. В классе **Part** (в окне свойств этого класса) введем пять полей — переменные *t0 — t4* с указанием в поле *Базовый класс* класса *Entity*, который можно выбрать из выпадающего меню.

Для включения этих полей в сообщение сначала измените описание класса сообщения **Part**. Откройте окно редактора и выделите класс **Part** в окне классов. В окне свойств этого объекта в поле *Поля* введите пять вещественных *(real)* переменных с именами t0, t1, t2, t3 и t4 с неопределенными (а значит, равными нулю) значениями этих переменных в поле *По умолчанию.*
Далее, в окне редактора структуры корневого объекта *Model* выделите элементы *source* и *source1*, и в каждом в поле *newEntity* выберите из выпадающего меню класс *Part.class*, указав тем самым класс порождаемого объекта. Нам следует в обоих этих блоках при выдаче заявки (сообщения) из блока установить время порождения заявки *getTime()*. Это можно сделать следующим образом. В поле *onExit* этих блоков выдаваемая заявка доступна под именем *entity*. Текущее время следует присвоить переменной *t0* текущей заявки, однако прежде этот объект должен быть приведен к типу (Part). Таким образом, в каждом из блоков *source* и *source 1* в поле *onExit* этих блоков необходимо вписать:

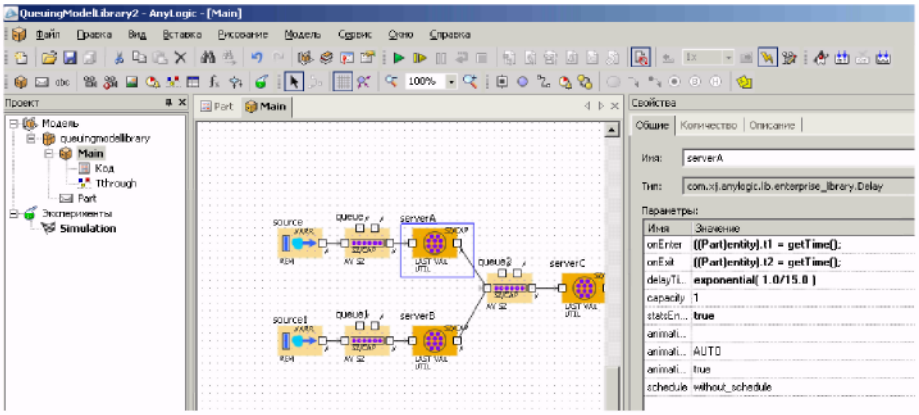
```
((Part)entity).t0 = getTime();
```

В серверах *server A* и *serverB* следует при входе заявки на обработку зафиксировать текущее время в поле *t1* (момент начала обработки в сервере), а при выходе заявки из блока отметить его в поле *t2* (момент окончания обработки и постановки в очередь II. Для этого в полях *onEnter* и *onExit* окна

свойств обоих этих блоков *serverA* и *serverB* следует вставить соответственно команды:

((Part)entity).t1 = getTime(); и

((Part)entity).t2 = getTime();



Аналогично, в сервере *serverC* при входе заявки на обработку следует зафиксировать текущее время в поле *t3* (момент начала обработки в сервере С), а при выходе заявки из этого блока зафиксировать текущее время в поле *t4* (момент окончания обработки в сервере С). Итак, в полях *onEnter* и *onExit* окна свойств блока *serverC* вставьте соответственно команды:

((Part)entity).t3 = getTime(); и

((Part)entity).t4 = getTime();

Подсчитать статистические характеристики случайной величины времени нахождения заявки в системе просто. Обозначим эту величину *Tthrough.*

Введем новый не временной набор данных *Tthrough* в модель (которая здесь по умолчанию получила имя *Main).* В блоке *sink* нужно при получении очередной заявки добавить к набору данных с именем *Tthrough* разность между моментом выхода заявки из системы и моментом ее входа в систему. Первая величина – *((Part)entity).t4.* Вторая величина – *((Part)entity).t0.*

Таким образом, добавив в поле *onEnter* блока *sink* модели оператор

Tthrough.add(((Part)entity).t4 - ((Part)entity).t0);

мы решим поставленную задачу.