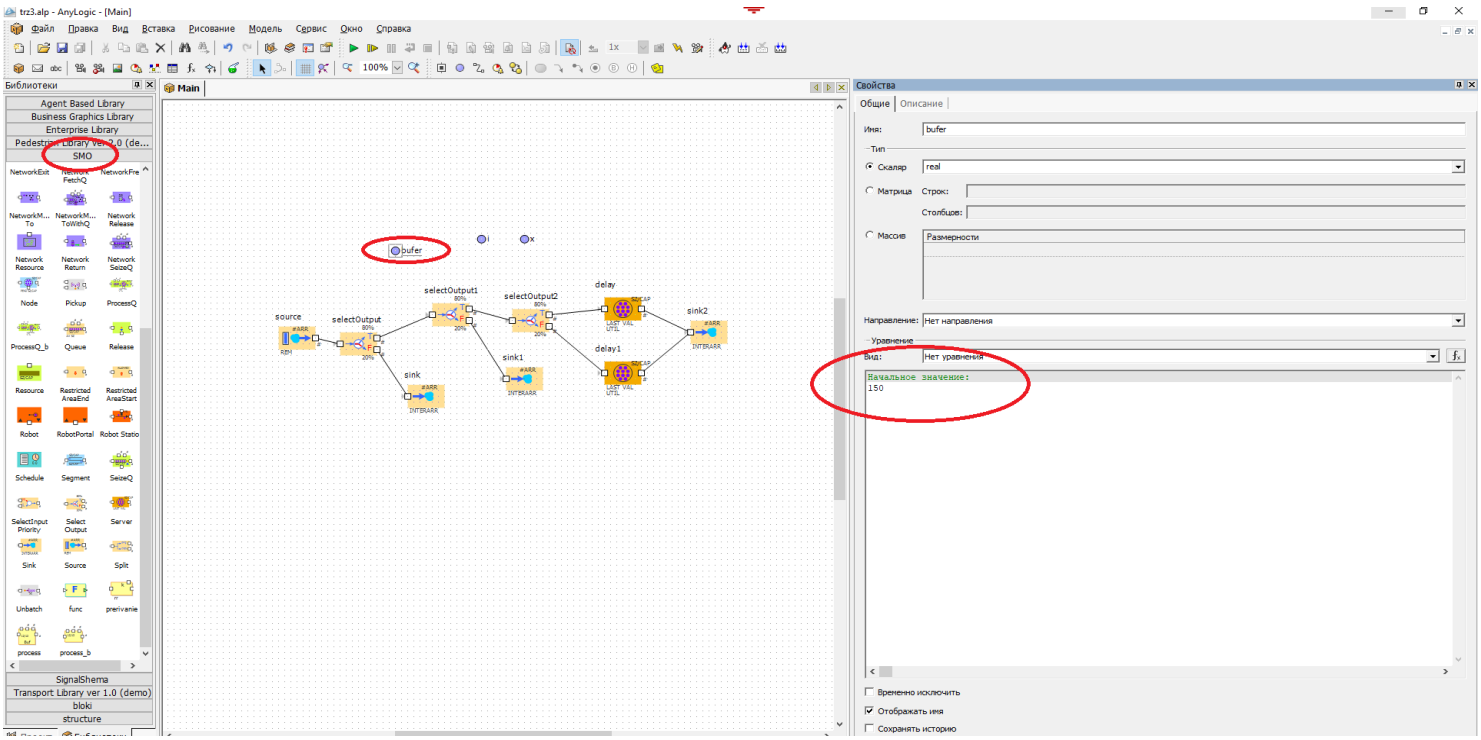
Тренировочная задача 3.

Заявки на обработку поступают равномерно со временем 60+-30 с и размещаются в буфере емкостью 150 ячеек. Каждая заявка требует для размещения следующее число ячеек: 30% - 20; 30% - 40; 20% - 60; 20% - 70. Если емкости для размещения заявки не хватает, она получает отказ первого рода и выводится из системы.

Заявки, находящиеся в буфере, обрабатываются на одном из двух приборов, время обработки 200+-30 сек. Выбор прибора по правилу: первый свободный с наименьшим номером. Если оба прибора заняты, заявка получает отказ 2 рода и удаляется из системы.

Постоим схему системы массового обслуживания, а уже затем начнём её настраивать. После генератора заявок source будет проверка на то, хватает ли места в буфере и если нет, то заявка удаляется из системы в блоке sink. Если места хватает, то заявка идёт на следующую проверку в блок selectOutput1, где проверяется есть ли вообще свободные приборы. Если оба прибора заняты, то заявка удаляется через блок sink1. Если есть хоть один свободный прибор, то заявка идёт на очередной блок проверки selectOutput, после которого отправляется на свободный прибор. После обработки на одном из приборов заявка удаляется из системы в блоке sink2.
Кроме того, нам понадобятся 2 переменных – bufer,i – для имитации буфера и процентных промежутков соответственно. Для переменной bufer устанавливаем начальное значение 150, так как именно столько ячеек изначально есть в буфере. i оставляем без изменений.



Настройки блоков source, delay, delay1 сделайте самостоятельно (или посмотрите в проекте, который я выложу).

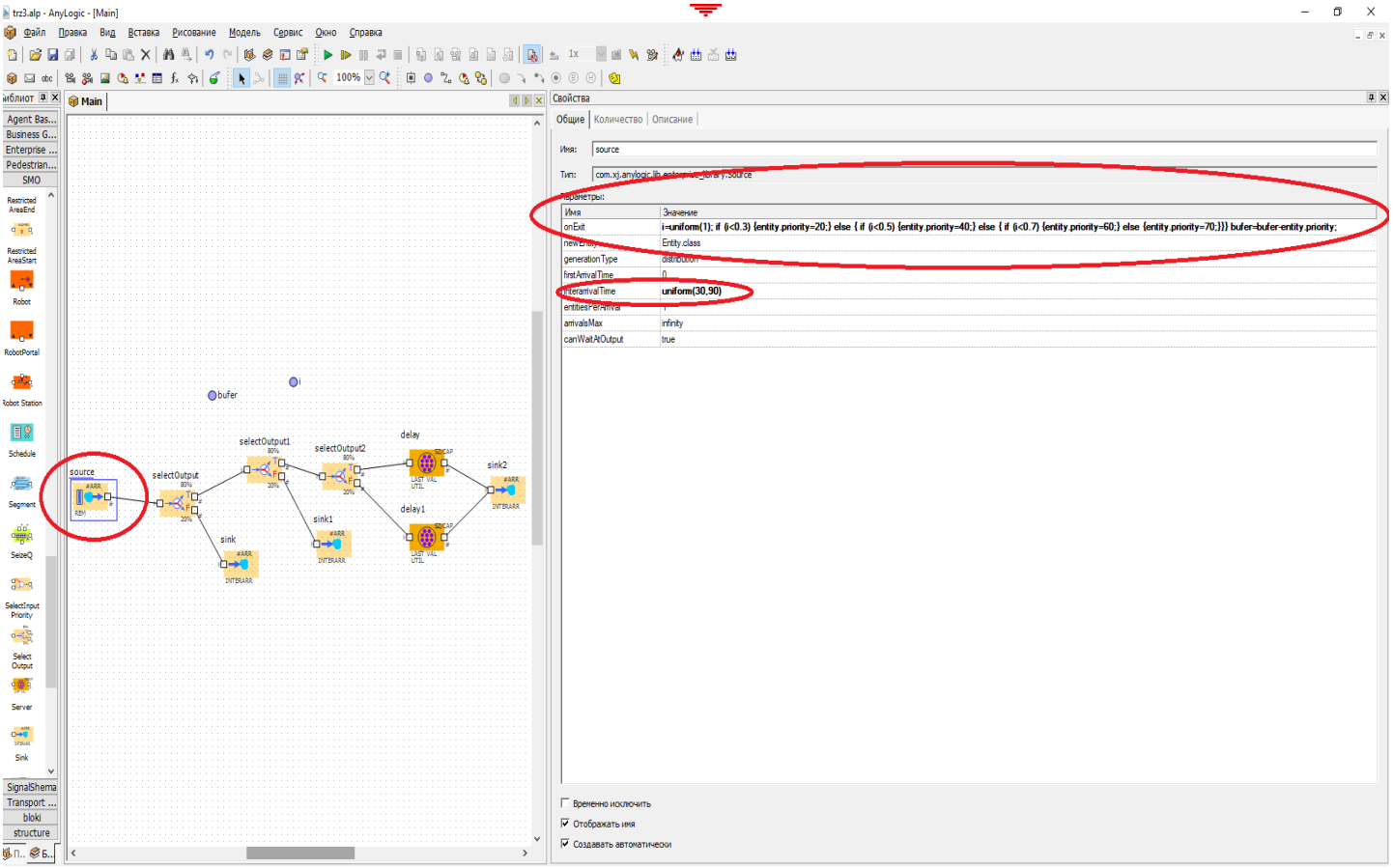
Ну а теперь самое главное – размещение заявок в буфере. У нас есть

несколько процентных промежутков. Поэтому переводим их в вероятности попадания заявки в один из этих процентных промежутков. Для этого в переменную i записываем результат работы функции uniform(1), которая случайно выбирает число от 0 до 1. Далее в зависимости от того в какой процентный промежуток попадёт наша заявка будем указывать количество ячеек, которое нужно для размещения этой заявки. В простую переменную записывать количество ячеек для каждой заявки нельзя, так как переменная глобальная, а заявок в системе может находится сразу несколько, поэтому нам нужна переменная или параметр, которые будут относится к каждой отдельной заявки. Для этого воспользуемся руководством Smo_min и описанием базового класса entity, которые реализует работу с заявками. На странице 8 мы можем увидеть все переменные и параметры этого класса. Логичней взять свободный параметр p_int, но в некоторых версиях anylogic может не сработать (нужно с правами администратора подключать дополнительные библиотеки), поэтому я возьму другой параметр (изначально по смыслу он нужен для указания приоритета), который по типу данных (integer, целое число) нам подходит – это параметр priority.
Получаем следующее выражение:

i=uniform(1); if (i<0.3) {entity.priority=20;} else { if (i<0.5) {entity.priority=40;} else { if (i<0.7) {entity.priority=60;} else {entity.priority=70;}}}

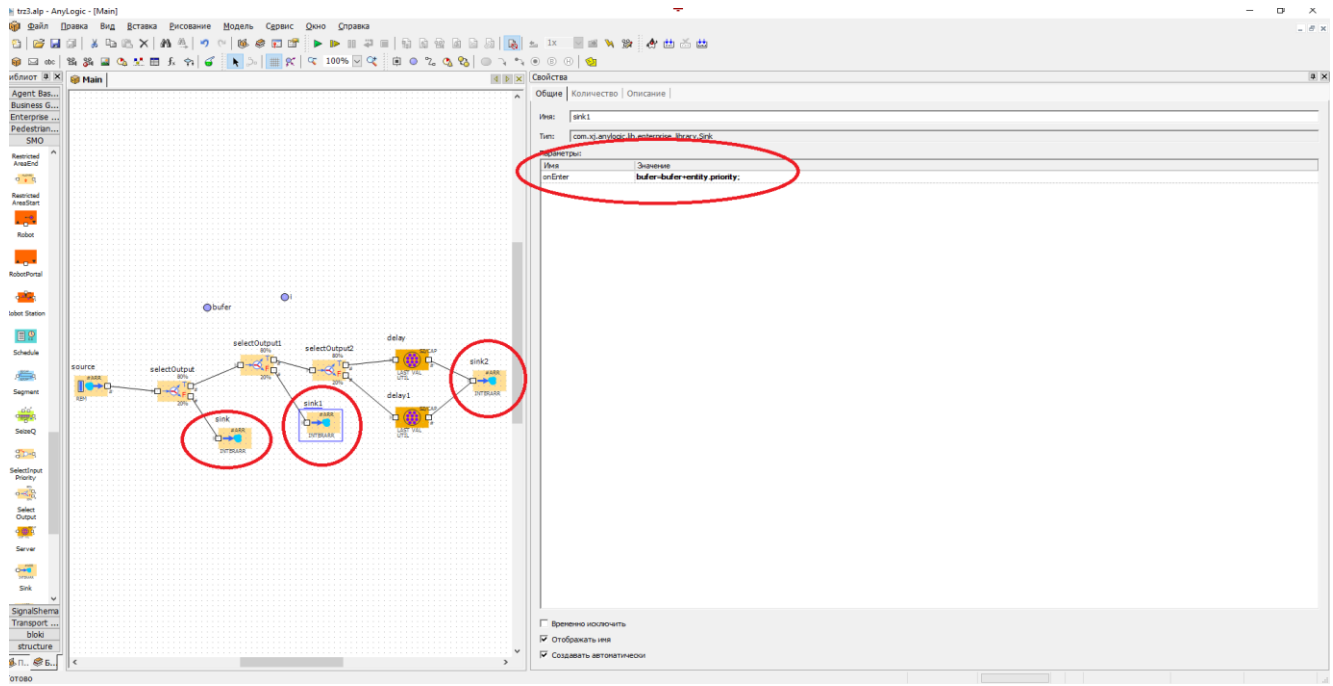
Т.е. если заявка попадает в первые 30% (***(i<0.3***), то для размещения заявки нужно 20 ячеек (***entity.priority=20;***) , если заявка попала в следующие 20% (***i<0.5, так как к имеющейся вероятности 0,2 добавляем вероятность 0,3***), то для размещения заявки нужно 40 ячеек (***entity.priority=40;***) и так далее.
После того как распределение мы запрограммировали нам нужно вычесть из буфера количество ячеек, которое необходимо для её размещения:

bufer=bufer-entity.priority;



Ну если мы вычли из буфера количество ячеек, необходимое для размещения заявки, при входе заявки в систему, то логично вернуть эти ячейки в буфер при удалении заявки из системы. Поэтому в блоках sink, sink1, sink2 нам нужно прописать следующее выражение:

bufer=bufer+entity.priority;



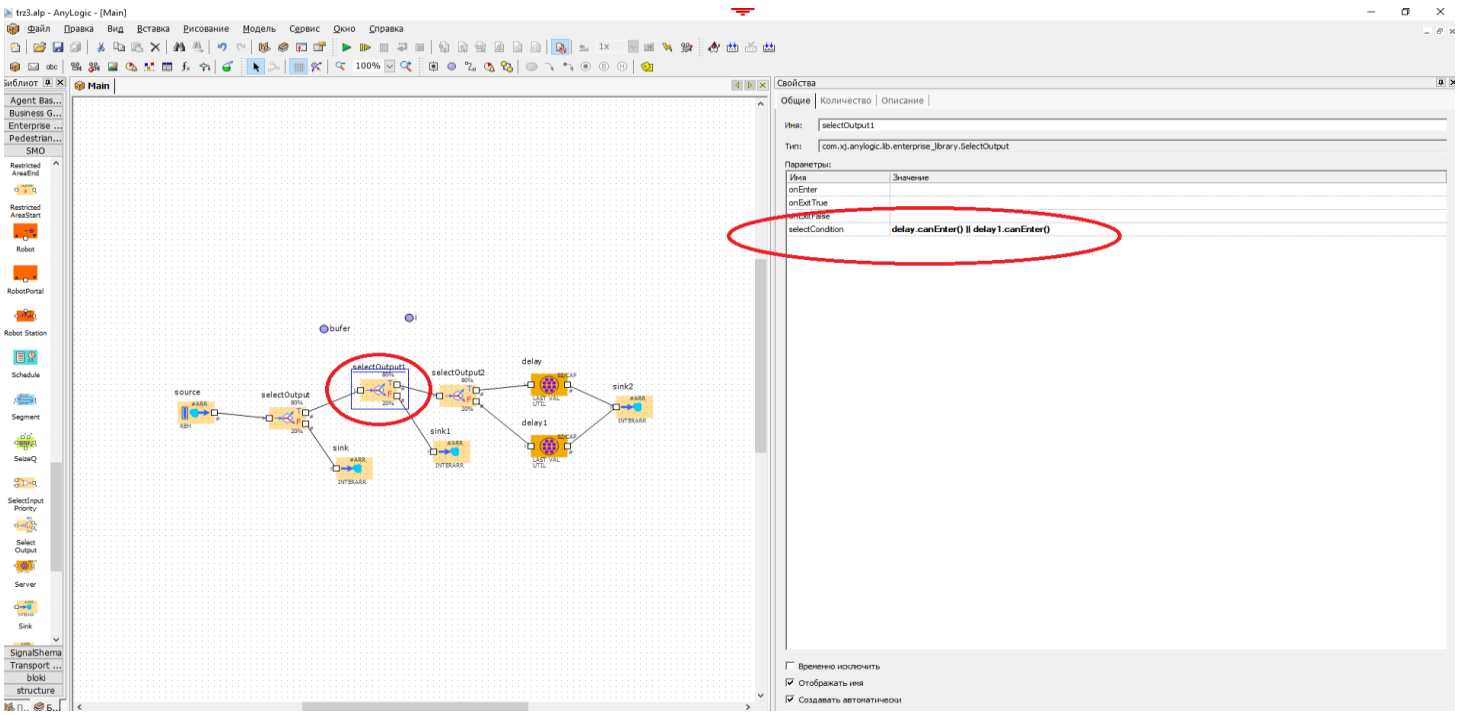
Теперь нужно настроить 3 блока проверки условий selectOutput. Для selectOutput, который идёт сразу после генератора заявок source критерий проверки будет:

bufer>=0

Он означает что если буфер отрицательный, то заявка только что сгенерированная «не влезает». Таким образом, критерий даёт значение false и заявка идёт через нижний выход в блок sink для удаления из системы. Если буфер положителен или равен 0, то заявка разместилась в буфере и дальше по верхнему выходу идёт на следующую проверку. На следующем блоке мы проверяем есть ли свободные приборы:

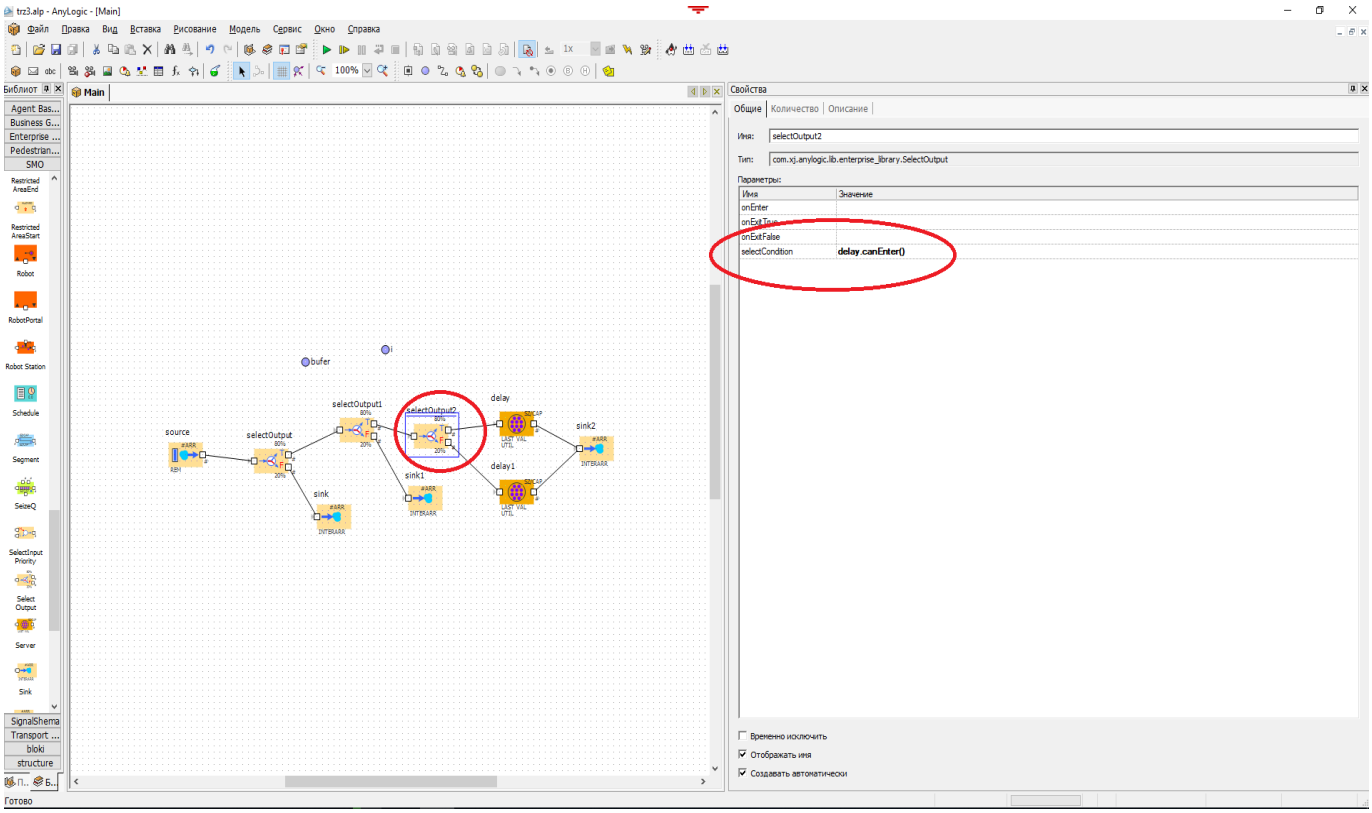
delay.canEnter() || delay1.canEnter()

Пользуемся уже известной нам функцией canEnter() для блока delay и оператором «логическое ИЛИ», потому что нам важно чтобы хотя бы один из приборов был свободен, а не сразу оба.



На следующем блоке проверке нам достаточно проверить свободен ли первый прибор - ***delay.canEnter()***

Если это выражение истинно, то заявка пойдёт на первый прибор, а если нет, то на второй. А по условию задачи нам это и нужно.


trz3.alp - AnyLogic - [Main]
Файл Правка Вид Вставка Рисование Модель Сервис Окно Справка
Библиот
Agent Bas...
Business G...
Enterprise ...
Pedestrian...
SMO
Restricted AreaEnd
Restricted AreaStart
Robot
RobotPortal
Robot Station
Schedule
Segment
SeizeQ
SelectInput Priority
Select Output
Server
Sink
SignalShema
Transport ...
bloki
structure
Main
bufer
source
selectOutput
selectOutput1
selectOutput2
sink
sink1
delay
delay1
sink2
Свойства
Общие
Количество
Описание
Имя:
selectOutput2
Тип:
com.xj.anylogic.lib.enterprise_library.SelectOutput
Параметры:
Имя
Значение
onEnter
onExitTrue
onExitFalse
selectCondition
delay.canEnter()
Временно исключить
Отображать имя
Создавать автоматически
Готово