Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчёт

по лабораторной работе №5

дисциплины «Методы исследования и моделирования информационных процессов и технологий»

на тему «Среда имитационного моделирования Anylogic. Модель функционирования системы связи»

Выполнил:

Ст. гр. ИС/м-21о Дядюшенко С.Е.

Проверил:

Проф. Доронина Ю.В.

Севастополь

2017

1. **Цель работы**

Закрепления навыков работы со средой имитационного моделирования Anylogic.

1. **Постановка задачи**

На дежурстве находятся n1 средств связи (СС) n2 типов (n21 + n22 + ... + n2n2 = n2) в течение n3 часов. Каждое СС может в любой момент времени выйти из строя. Интервалы времени T21, T22, ..., T2n2 между отказами СС, находящимися на дежурстве, случайные. В случае выхода из строя СС заменяют резервным, причем либо сразу, либо по мере появления исправного СС. Тем временем, вышедшее из строя СС ремонтируют, после чего содержат в качестве резервного или направляют его на дежурство. Всего количество резервных СС — n4. Ремонт неисправных СС производят n5 мастеров. Время T1, T2, ..., Tn2 ремонта случайное и зависит от типа СС, но не зависит от того, какой мастер это СС ремонтирует. Прибыль от СС, находящихся на дежурстве, составляет S1 денежных единиц в час. Почасовой убыток при отсутствии на дежурстве одного СС — S2, ..., S2n2 денежных единиц в час. Оплата мастера за ремонт неисправного СС — S31, S32, ..., S3n2 денежных единиц в час соответственно. Затраты на содержание одного резервного СС составляют S4 денежных единиц в час.

Разработать имитационную модель бизнес-процесса предоставления услуг по средствам связи в течение 1000 часов. Исследовать влияние на ожидаемую прибыль различного количества резервных СС и мастеров. Сделать выводы об использовании СС, мастеров и необходимых мерах по совершенствованию системы предоставления услуг связи.

1. **Разработка модели**

В ходе выполнения лабораторной работы была разработана модель AnyLogic. Графическое изображение модели представлено на рисунке 1.

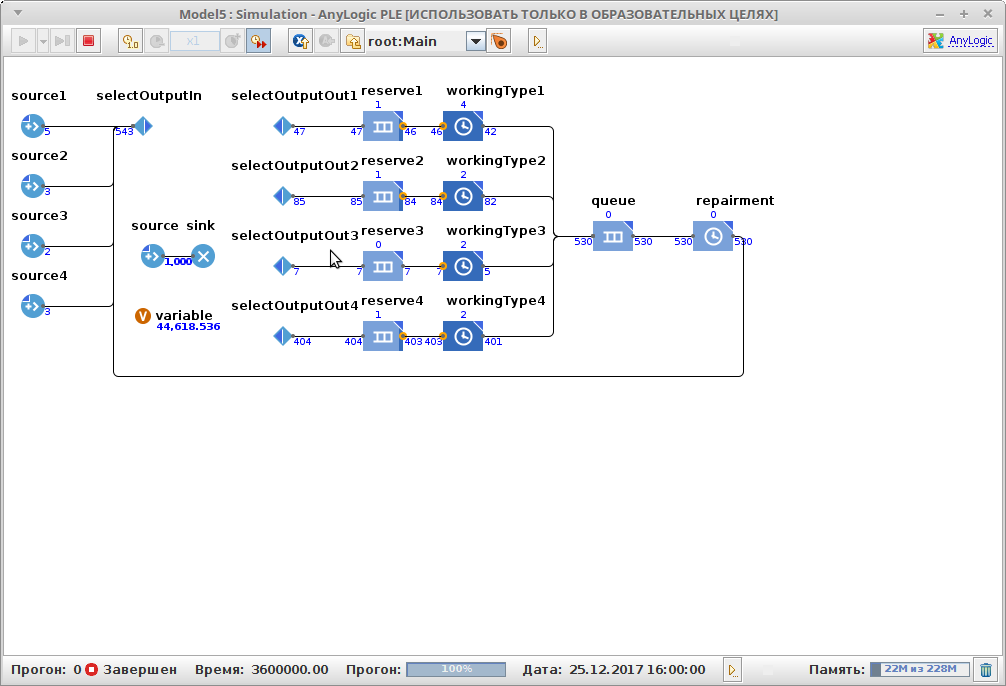


Рисунок 1 – Графическое изображение разработанной модели

Модель содержит четыре источника средств связи разных типов, которые вводят в систему определённое заранее заданное количество деталей в момент запуска. Команды запуска представлены на рисунке 2.

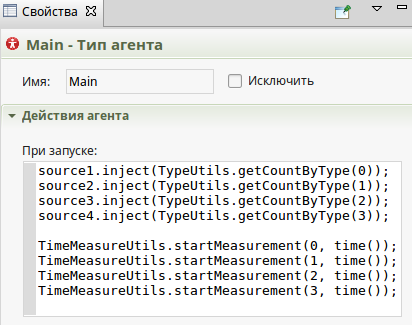


Рисунок 2 – Команды ввода средств связи в систему и старт измерений времени

Здесь также упомянуты классы TimeMeasureUtils и TypeUtils, содержимое которого представлено ниже.

На рисунке 3 представлены свойства источника заявок.

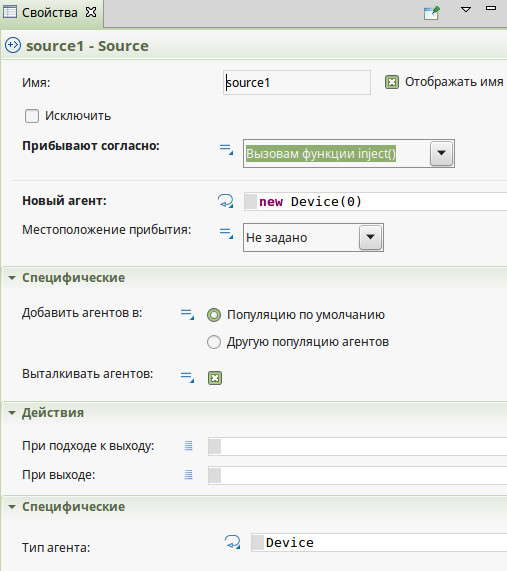


Рисунок 3 – Свойства источника заявок

Типом агента является класс Device, который описан ниже.

После генерации устройство переходит на блок выбора, который осуществляет его распределение по типу. Свойства этого блока представлены на рисунке 4.

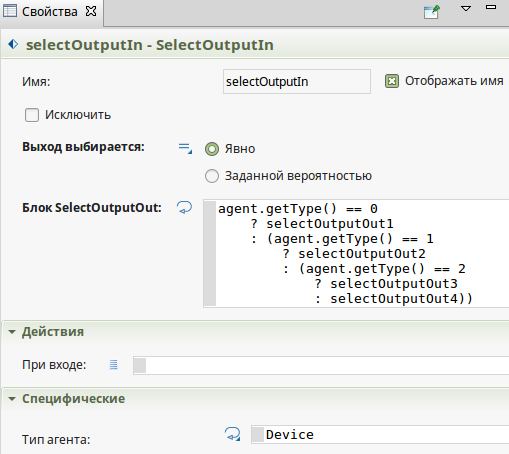


Рисунок 4 – Свойства блока выбора

После распределения каждое из устройств попадает в резерв своего типа. Время пребывания в резерве фиксируется и определённым образом влияет на значение выручки (variable). Свойства блока резерва представлены на рисунке 5.

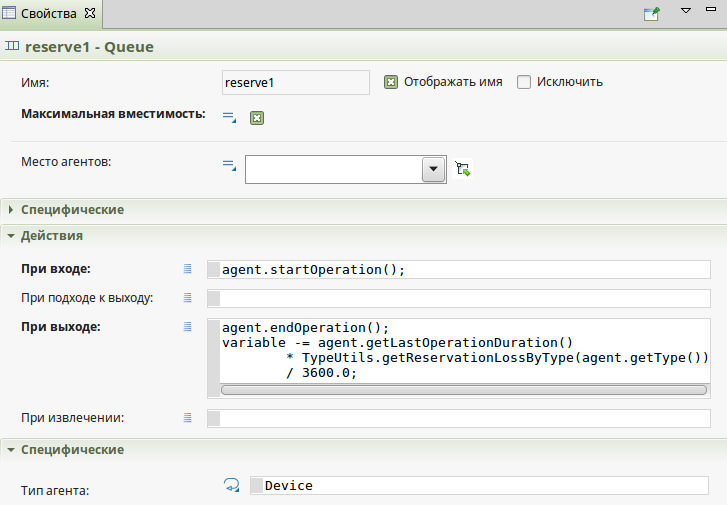


Рисунок 5 – Свойства блока резерва

Если есть место в блоке работы, устройство переходит в него. Время пребывания в состоянии работы фиксируется и используется при расчёте прибыли. Также фиксируется состояние самого блока (количество работающих устройств) и время пребывания в этом состоянии. Эти данные также используются в расчёте прибыли. Действия при входе и выходе из блока представлены на рисунке 6.

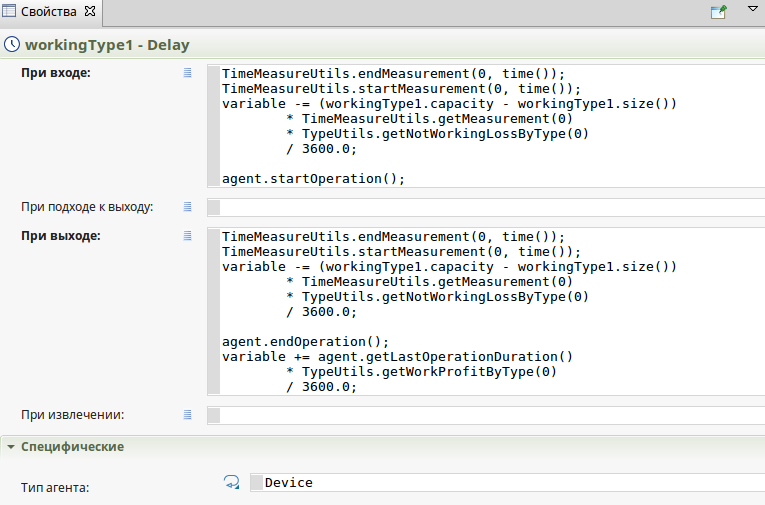
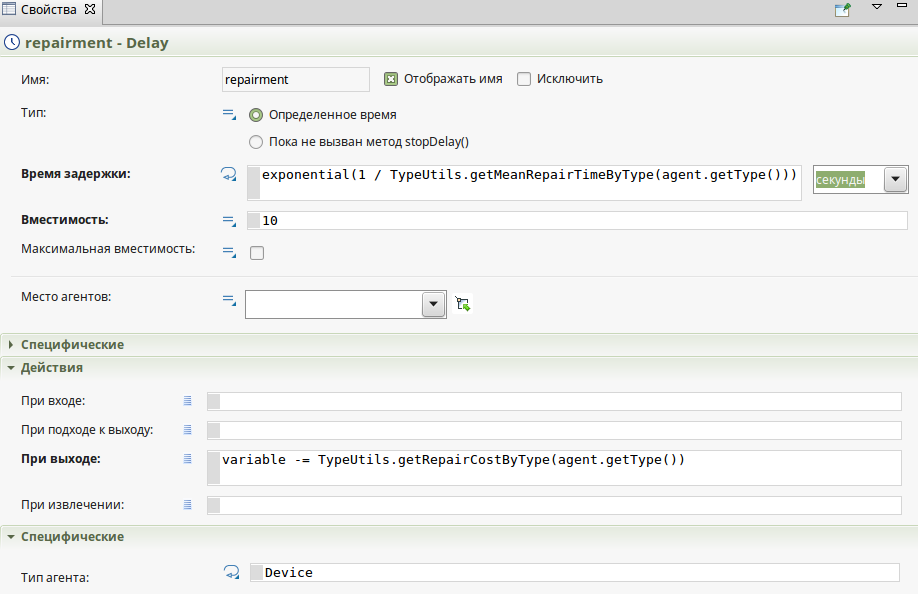


Рисунок 6 – Действия при входе и выходе из блока работы устройства

После завершения работы устройство переходит в очередь и ремонт. Свойства блока ремонта представлены на рисунке 7.



Риснуок 7 – Свойства блока ремонта

После ремонта устройство снова попадает на блок распределения.

Также на схеме присутствует счётчик шагов (представленный источником и поглотителем), который выполняет подсчёт прибыли после завершения последнего шага. Действия при завершении моделирования представлены на рисунке 8.

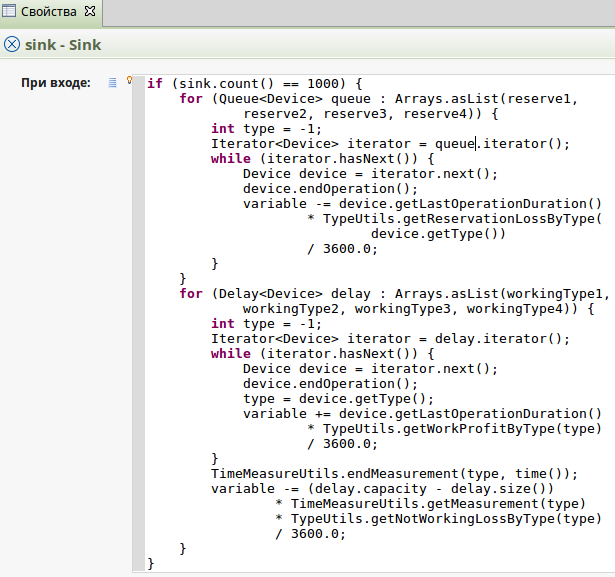


Рисунок 8 – Действия при завершении моделирования

Также для работы системы разработаны три класса: Device, TimeMeasureUtils и TypeUtils.

Класс Device описывает агент-устройство и содержит номер типа а также поля для сохранения времени начала и конца операции.

Текст класса Device:

public class Device extends Agent implements Serializable {

private static final long serialVersionUID = 1L;

private int type;

private double operationTimeStart;

private double operationTimeEnd;

public Device(int type) {

this.type = type;

}

public void startOperation() {

operationTimeStart = time();

}

public void endOperation() {

operationTimeEnd = time();

}

public int getType() {

return type;

}

public double getLastOperationDuration() {

return operationTimeEnd - operationTimeStart;

}

}

Класс TimeMeasureUtils необходим для сохранения времени длительности состояния блока работы. Он содержит массив измерений времени в котором каждый блок имеет свою ячейку.

Текст класса TimeMeasureUtils:

public class TimeMeasureUtils implements Serializable {

private static double[] measurements = new double[4];

public static void startMeasurement(int i, double time) {

measurements[i] = time;

}

public static void endMeasurement(int i, double time) {

measurements[i] = time - measurements[i];

}

public static double getMeasurement(int i) {

return measurements[i];

}

private TimeMeasureUtils() { }

}

Класс TypeUtils содержит параметры времени и стоимости различных операций и состояний для каждого типа устройства. Создан для удобства, чтобы можно было хранить параметры типов в одном месте.

Текст класса TypeUtils:

public class TypeUtils {

private static Map<Integer, Integer> mapCountByType = new HashMap<>();

private static Map<Integer, Integer> mapNeedToWorkByType = new HashMap<>();

private static Map<Integer, Double> mapMeanFailTimeByType = new HashMap<>();

private static Map<Integer, Double> mapMeanRepairTimeByType = new HashMap<>();

private static Map<Integer, Double> mapWorkProfitByType = new HashMap<>();

private static Map<Integer, Double> mapRepairCostByType = new HashMap<>();

private static Map<Integer, Double> mapNotWorkingLossByType = new HashMap<>();

private static Map<Integer, Double> mapReservationLossByType = new HashMap<>();

static {

mapCountByType.put(0, 5);

mapCountByType.put(1, 3);

mapCountByType.put(2, 2);

mapCountByType.put(3, 3);

mapNeedToWorkByType.put(0, 4);

mapNeedToWorkByType.put(1, 2);

mapNeedToWorkByType.put(2, 2);

mapNeedToWorkByType.put(3, 2);

mapMeanFailTimeByType.put(0, 345600.0);

mapMeanFailTimeByType.put(1, 64800.0);

mapMeanFailTimeByType.put(2, 1209600.0);

mapMeanFailTimeByType.put(3, 18000.0);

mapMeanRepairTimeByType.put(0, 7200.0);

mapMeanRepairTimeByType.put(1, 1800.0);

mapMeanRepairTimeByType.put(2, 3600.0);

mapMeanRepairTimeByType.put(3, 1200.0);

mapWorkProfitByType.put(0, 500.0);

mapWorkProfitByType.put(1, 200.0);

mapWorkProfitByType.put(2, 400.0);

mapWorkProfitByType.put(3, 50.0);

mapRepairCostByType.put(0, 50.0);

mapRepairCostByType.put(1, 10.0);

mapRepairCostByType.put(2, 10.0);

mapRepairCostByType.put(3, 5.0);

mapNotWorkingLossByType.put(0, 30.0);

mapNotWorkingLossByType.put(1, 10.0);

mapNotWorkingLossByType.put(2, 50.0);

mapNotWorkingLossByType.put(3, 10.0);

mapReservationLossByType.put(0, 5.0);

mapReservationLossByType.put(1, 15.0);

mapReservationLossByType.put(2, 10.0);

mapReservationLossByType.put(3, 10.0);

}

public static int getCountByType(int type) {

return mapCountByType.get(type);

}

public static int getNeedToWorkByType(int type) {

return mapNeedToWorkByType.get(type);

}

public static double getMeanFailTimeByType(int type) {

return mapMeanFailTimeByType.get(type);

}

public static double getMeanRepairTimeByType(int type) {

return mapMeanRepairTimeByType.get(type);

}

public static double getWorkProfitByType(int type) {

return mapWorkProfitByType.get(type);

}

public static double getRepairCostByType(int type) {

return mapRepairCostByType.get(type);

}

public static double getNotWorkingLossByType(int type) {

return mapNotWorkingLossByType.get(type);

}

public static double getReservationLossByType(int type) {

return mapReservationLossByType.get(type);

}

private TypeUtils() { }

}

1. **Проведение эксперимента**

В ходе эксперимента были зафиксированы все параметры, кроме исследуемых. Зафиксированные значения количество типов: первый – 5, второй – 3, третий – 2, четвёртый – 3. Фиксированное значение количества мастеров – 5.

В ходе первого эксперимента менялось значение количества устройств первого типа, и отслеживалась прибыль. Результаты представлены в таблице 1.

Таблица 1 – Влияние количества устройств первого типа на прибыль

|  |  |
| --- | --- |
| Количества устройств первого типа | Прибыль |
| 3 | -1227765 |
| 4 | -229998 |
| 5 | 44385 |
| 6 | 68646 |
| 7 | 174146 |
| 8 | 169146 |
| 9 | 164146 |

Максимальная прибыль наблюдается при наличии 7 устройств, после чего значение начинает медленно убывать. Это объясняется тем, что убытки от отсутствия устройства на дежурстве гораздо больше, чем от содержания устройства в резерве, а устройство отказывает достаточно часто.

В ходе второго эксперимента менялось значение количества устройств второго типа, и отслеживалась прибыль. Результаты представлены в таблице 2.

Таблица 2 – Влияние количества устройств второго типа на прибыль

|  |  |
| --- | --- |
| Количества устройств второго типа | Прибыль |
| 1 | -333781 |
| 2 | 31485 |
| 3 | 44385 |
| 4 | -74572 |

Максимальная прибыль наблюдается при наличии 3 устройств, так как при наличии одного устройства появляются убытки при отсутствии необходимого количества на дежурстве, а при наличии четырёх устройств как минимум одно стоит в резерве и на него расходуются средства.

В ходе третьего эксперимента менялось значение количества устройств третьего типа, и отслеживалась прибыль. Результаты представлены в таблице 3.

Таблица 3 – Влияние количества устройств третьего типа на прибыль

|  |  |
| --- | --- |
| Количества устройств третьего типа | Прибыль |
| 1 | -480526 |
| 2 | 44385 |
| 3 | 65248 |
| 4 | 55248 |

Максимальная прибыль наблюдается при наличии 3 устройств, после чего значение начинает медленно убывать.

В ходе четвёртого эксперимента менялось значение количества устройств четвёртого типа, и отслеживалась прибыль. Результаты представлены в таблице 4.

Таблица 4 – Влияние количества устройств четвёртого типа на прибыль

|  |  |
| --- | --- |
| Количества устройств четвёртого типа | Прибыль |
| 1 | -971344 |
| 2 | -225410 |
| 3 | 44385 |
| 4 | -229496 |

Максимальная прибыль наблюдается при наличии 3 устройств. При увеличении или уменьшении прибыль резко уменьшается, что связано с большими затратами как при отсутствии на дежурстве устройства, так и при его содержании в резерве.

В ходе пятого эксперимента наблюдалось влияние количества мастеров на прибыль. Результаты представлены в таблице 5.

Таблица 5 – Влияние количества мастеров на прибыль

|  |  |  |
| --- | --- | --- |
| Количество мастеров | Количество ремонтов | Прибыль |
| 1 | 520 | 99330 |
| 2 | 524 | 19986 |
| 3 | 530 | 44386 |
| 4 | 530 | 44385 |

Результаты также сильно зависят от типов устройств, отданных на ремонт. Увеличение прибыли объясняется отсутствием убытков, если устройство не стоит в резерве, а находится в ремонте. Уменьшение – увеличением убытков от отсутствия устройства на дежурстве. С ростом количества мастеров прибыль стабилизируется, так как поломанные устройства не ожидают очереди на ремонт.

**Вывод:**

В ходе выполнения лабораторных работ закреплены навыки работы со средой имитационного моделирования Anylogic. Построена модель функционирования системы связи. Проведено исследование влияния количества устройств каждого типа и количества мастеров на прибыль. Убытки от количества устройств происходят из-за дороговизны содержания их в резерве и отсутствия на дежурстве. Также и убытки из-за количетсва мастеров. Определены показатели, при которых достигается максимальная прибыль.