

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
**«Кузбасский государственный технический университет имени Т.Ф. Горбачева»**

Кафедра автомобильных перевозок

Составители  
О. С. Семенова  
Ю. Н. Семенов

## **МОДЕЛИРОВАНИЕ ТРАНСПОРТНЫХ СИСТЕМ**

Методические указания к практическим занятиям  
для магистрантов всех форм обучения

Рекомендованы учебно-методической комиссией направления  
подготовки 23.04.01 «Технология транспортных процессов»  
в качестве электронного издания  
для использования в учебном процессе

Кемерово 2016

## Рецензенты

Стенина Н. А. – кандидат технических наук, доцент кафедры автомобильных перевозок

Воронов Ю. Е. – доктор технических наук, профессор кафедры автомобильных дорог, председатель учебно-методической комиссии направления подготовки 23.04.01 «Технология транспортных процессов»

**Семенов Юрий Николаевич**

**Семенова Ольга Сергеевна**

**Моделирование транспортных систем:** методические указания к практическим занятиям [Электронный ресурс]: для магистрантов направления 23.04.01 «Технология транспортных процессов», образовательная программа «Организация и управление транспортными процессами», всех форм обучения / сост.: Ю. Н. Семенов, О. С. Семенова; КузГТУ. – Кемерово, 2016.

Приведенные методические указания к практическим занятиям по курсу «Моделирование транспортных систем» позволяют расширить знания, полученные в ходе аудиторных занятий; способствуют закреплению теоретических положений; развивают навыки по их практическому применению.

© КузГТУ, 2016

© Семенов О. С.,  
Семенова Ю. И.,  
составление, 2016

**СОДЕРЖАНИЕ**

<u>Общие положения .....</u>	<u>4</u>
<u>Практическая работа №1.</u>	
<u>Построение дискретно-событийной модели. Сбор статистики по работе блоков диаграммы процесса. Оптимизация.....</u>	<u>5</u>
<u>Практическая работа №2.</u>	
<u>Создание и оптимизация пешеходной модели.....</u>	<u>9</u>
<u>Практическая работа №3.</u>	
<u>Моделирование движения автомобилей и пешеходов на участке УДС .....</u>	<u>14</u>
<u>Практическая работа №4.</u>	
<u>Использование системной динамики для моделирования транспортных систем.....</u>	<u>23</u>
<u>Список литературы.....</u>	<u>28</u>

## **ОБЩИЕ ПОЛОЖЕНИЯ**

«Моделирование транспортных систем» является дисциплиной, формирующей у студентов общее представление о моделировании транспортных систем, что позволит применять разделы прикладной математики в научно-практических исследованиях.

## Практическая работа №1

### Построение дискретно-событийной модели.

#### Сбор статистики по работе блоков диаграммы процесса. Оптимизация

**Цель работы:** Построение, анализ и оптимизация дискретно-событийной модели.

#### Теоретические положения

Дискретно-событийное моделирование обязано своим рождением Дж. Гордону, который в начале 1960-х спроектировал и реализовал на IBM систему дискретно-событийного программирования GPSS (Global Purpose Simulation System).

Основной объект в этой системе – пассивный транзакт (заявка на обслуживание), который может определенным образом представлять собой работников, клиентов, покупателей, детали, сырье, документы, сигналы и т. п. «Перемещаясь» по модели, транзакты становятся в очереди к одноканальным и многоканальным устройствам, захватывают и освобождают эти устройства, расщепляются, уничтожаются и т. д. Таким образом, дискретно-событийную модель можно рассматривать как глобальную схему обслуживания заявок.

Основными компонентами системы массового обслуживания являются (рисунок 1):

- входной поток поступающих требований (или заявок на обслуживание), который задается на основе вероятностного закона распределения моментов поступления заявок требований;
- дисциплина очереди, принцип, на основе которого подключаются требования к очереди на обслуживание;
- механизм обслуживания определяется продолжительностью процедуры обслуживания и количеством требований, удовлетворяемых в результате.

Для построения дискретно-событийной модели используются стандартные блоки:

- *Создание агентов.* Обычно используется в качестве начальной точки потока агентов.
- *Уничтожение поступивших агентов.* Обычно используется в качестве конечной точки потока агентов.
- *Задержка агентов на заданный период времени.*
- *Очередь агентов, ожидающих приема объектами, следующими за данным в потоковой диаграмме*
- *Ресурсы* – это специальный объект *Библиотеки моделирования процессов*, который может потребоваться агенту для выполнения какой-то задачи. В каждый момент времени ресурс может быть занят только одним агентом.

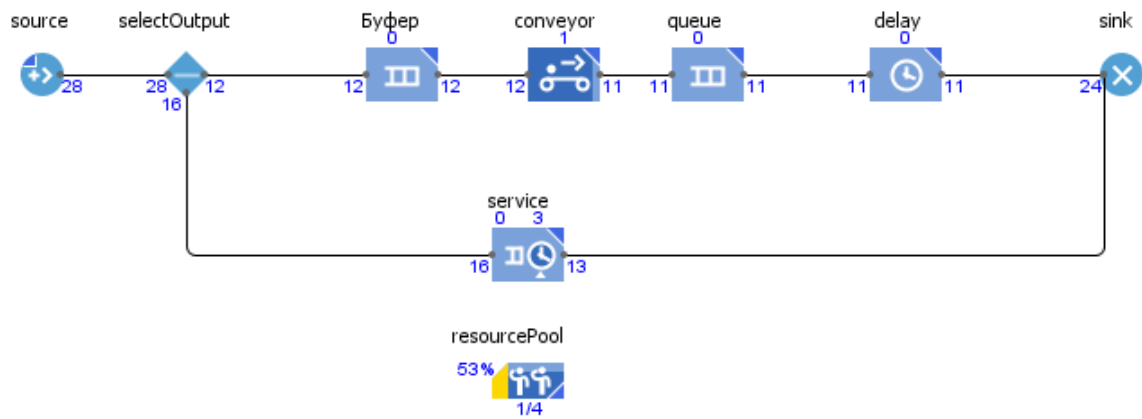


Рисунок 1 – Основные объекты дискретно-событийной модели

Логику модели можно задать на языке Java:

```
public class Main extends Agent
{
    public
    int КолКлерков;
    public int _КолКлерков_DefaultValue_xjal() {
        final Main self = this;
        return 4;
    }

    public void set_КолКлерков( int КолКлерков ) {
        if (КолКлерков == this.КолКлерков) {
            return;
        }
        int _oldValue_xjal = this.КолКлерков;
        this.КолКлерков = КолКлерков;
        onChange_КолКлерков_xjal( _oldValue_xjal );
        onChange();
    }

    protected void onChange_КолКлерков() {
        onChange_КолКлерков_xjal( КолКлерков );
    }

    protected void onChange_КолКлерков_xjal( int oldValue ) {
        int index;
        {
            com.anylogic.libraries.processmodeling.ResourcePool<Клерк> self = re-
sourcePool;
            int _value;
            _value = КолКлерков;
            resourcePool.set_capacity( _value );
        }
    }

    public
    double ИнтенсивностьПрихода;
    public double _ИнтенсивностьПрихода_DefaultValue_xjal() {
        final Main self = this;
        return
        0.5 ; }

    public void set_ИнтенсивностьПрихода( double ИнтенсивностьПрихода ) {
```

```

if (ИнтенсивностьПрихода == this.ИнтенсивностьПрихода) {
    return;
}
double _oldValue_xjal = this.ИнтенсивностьПрихода;
this.ИнтенсивностьПрихода = ИнтенсивностьПрихода;
onChange_ИнтенсивностьПрихода_xjal( _oldValue_xjal );
onChange();
}
}

```

Для сбора статистики по работе блоков диаграммы процесса используются объекты Библиотеки моделирования процессов, которые самостоятельно производят сбор основной статистики (рисунок 2).

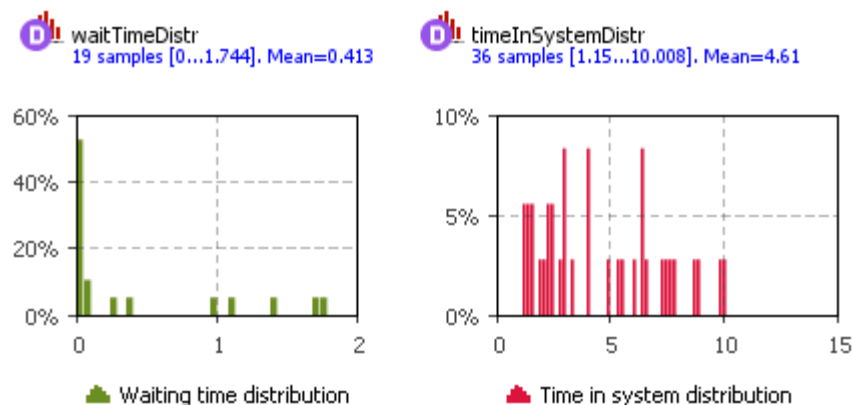


Рисунок 2 – Гистограммы для сбора статистики

Оптимизация дискретно-событийной модели производится в несколько этапов:

- Добавление оптимизируемого параметра для агента верхнего уровня;
- Добавление целевой функции;
- Создание эксперимента (настройка эксперимента, задание модельного времени, создание интерфейса).

### Задания к практической работе № 1

1. Смоделировать зону погрузки/разгрузки транспортных средств. Количество погрузочно-разгрузочных механизмов – 1 шт., количество платформ для погрузки/разгрузки – 1 шт.
2. Добавить в модель 1 платформу для разгрузки ТС в ручную, 3 платформы для разгрузки ТС механизированным способом.
3. Создать 3D анимацию погрузки ТС, добавить камеры.
4. Собрать статистику.
5. Провести оптимизацию зоны погрузки/разгрузки транспортных средств. Определить оптимальное количество разгрузочных платформ при различной интенсивности прибытия автомобилей.

Сроки контроля – 5 неделя семестра.

Форма контроля – ответы на контрольные вопросы, отчёт по ПРН<sup>№1</sup>.

## **Требования к отчёту по практической работе №1**

Отчёт представляется в электронном виде, сохраняется на компьютере до защиты практической работы. Отчёт должен содержать:

1. Представление модели в формальном виде, с указанием переменных входа/выхода, внутренней логики модели, целевой функции и ограничений (для оптимизационных моделей);
2. Описание основных блоков потоковой диаграммы, используемых в модели;
3. Потоковую диаграмму;
4. Анимацию модели;
5. Блоки сбора статистики по работе блоков диаграммы процесса;
6. Оптимизационный эксперимент и результаты оптимизации модели (для оптимизационных моделей);
7. Анализ результатов оптимизации модели.

### **Контрольные вопросы:**

1. Основные понятия теории моделирования.
2. Системный подход в моделировании систем.
3. Классификация систем массового обслуживания и их основные характеристики.
4. Инструментальные средства и языки моделирования.
5. Что такое модель, с какой целью ее применяют?
6. В чем отличие структурного и классического подхода к моделированию систем?
7. В чем смысл таких характеристик модели, как управляемость и адаптивность?
8. Какие бывают виды моделирования?
9. В чем суть имитационного моделирования, каковы его преимущества и недостатки?
10. Из каких основных блоков состоит имитационная систем?
11. На чём основываются решения в реальных бизнес-ситуациях?
12. Что такое формализация модели?
13. Что такое условная оптимизация?
14. Для чего предназначен анализ "Что-если"?
15. Приведите примеры статических и динамических моделей.
16. Что такое материальные модели?
17. Что такое информационная модель?
18. Что такое математическая модель? Приведите примеры.
19. Можно ли назвать поясняющий чертеж к задаче моделью? Поясните ответ.
20. По какому признаку модели делятся на статические и динамические?



## Практическая работа №2

### Создание и оптимизация пешеходной модели

**Цель работы:** Построение, анализ и оптимизация пешеходной модели.

#### Теоретические положения

В моделях, созданных с помощью объектов *Пешеходной библиотеки*, пешеходы движутся в непрерывном пространстве, реагируя на различные виды препятствий в виде стен, различных областей и других пешеходов.

Составляющие модели движения пешеходов:

- *Среда*. В неё включаются объекты физической среды - стены, различные области, сервисы, очереди и т.д. Объект среды задается специальным графическим элементом разметки, у которого задаются параметры объекта среды. Ресурсы (сервисы) также являются объектами среды.
- *Поведение*. Задается блок-схемой.

Основным объектом библиотеки является пешеход. Пешеход задается с помощью объекта типа *Ped*. Пешеход «обитает» в заданном физическом пространстве (моделируемой среде) и передвигается согласно заданным правилам.

Тип пешехода унаследован от типа агента *Agent*, поэтому пешеходы перемещаются по блок-схеме так же, как агенты. Блок-схемы пешеходных моделей строятся с помощью объектов, содержащихся в *Пешеходной библиотеке*. В библиотеке есть объекты не только для создания пешеходов, но и для управления потоком пешеходов.

Этапы создания пешеходной модели:

1. Добавление рисунка-плана моделируемого пространства (помещения, здания) (рисунок 3).

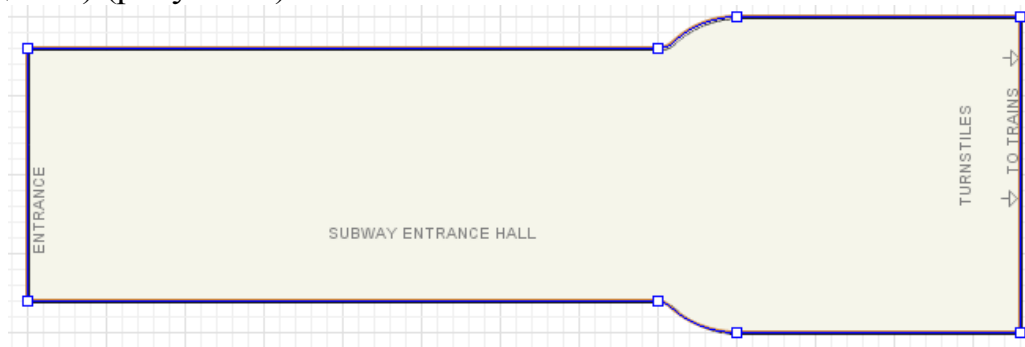


Рисунок 3 – План моделируемого пространства

2. Поверх стен на этом плане рисуются стены с помощью специальных элементов разметки пространства.

3. Создание диаграммы процесса, предназначенной для задания логики перемещения пешеходов внутри здания (рисунок 4), задание свойств и обработка событий:

```
public class Main extends Agent
{
```

```

    public
double    ЗначениеИнтенсивности;
    public double _ЗначениеИнтенсивности_DefaultValue_xjal() {
        final Main self = this;
        return
2500
    ;
    }
    public void set_ЗначениеИнтенсивности( double ЗначениеИнтенсивности ) {
        if (ЗначениеИнтенсивности == this.ЗначениеИнтенсивности) {
            return;
        }
        double _oldValue_xjal = this.ЗначениеИнтенсивности;
        this.ЗначениеИнтенсивности = ЗначениеИнтенсивности;
        onChange_ЗначениеИнтенсивности_xjal( _oldValue_xjal );
        onChange();
    }
    protected void onChange_ЗначениеИнтенсивности() {
        onChange_ЗначениеИнтенсивности_xjal( ЗначениеИнтенсивности );
    }

    @AnyLogicInternalCodegenAPI
    protected void onChange_ЗначениеИнтенсивности_xjal( double oldValue ) {
        int index;
        {
            com.anylogic.libraries.pedestrian.PedSource<Pedestrian> self = ped-
Source;
            double _value;
            _value = ЗначениеИнтенсивности
;
            pedSource.set_rate( _value );
        }
    }
    public void setParametersToDefaultValues() {
        super.setParametersToDefaultValues();
        ЗначениеИнтенсивности = _ЗначениеИнтенсивности_DefaultValue_xjal();
    }

    @Override
    public boolean setParameter(String _name_xjal, Object _value_xjal, boolean
_callOnChange_xjal) {
        switch ( _name_xjal ) {
            case «ЗначениеИнтенсивности»:
                if ( _callOnChange_xjal ) {
                    set_ЗначениеИнтенсивности( ((Number) _value_xjal).doubleValue() );
                } else {
                    ЗначениеИнтенсивности = ((Number) _value_xjal).doubleValue();
                }
                return true;
            default:
                return super.setParameter( _name_xjal, _value_xjal, _callOnChange_xjal
);
        }
    }

    public <T> T getParameter(String _name_xjal) {
        Object _result_xjal;
        switch ( _name_xjal ) {
            case «ЗначениеИнтенсивности»: _result_xjal = ЗначениеИнтенсивности;
            break;
            default: _result_xjal = super.getParameter( _name_xjal ); break;
        }
        return (T) _result_xjal;
    }

```

```

    private static String[] _parameterNames_xjal;

    @Override
    public String[] getParameterNames() {
        String[] result = _parameterNames_xjal;
        if (result == null) {
            List<String> list = new ArrayList<>( Arrays.asList( su-
per.getParameterNames() ) );
            list.add( «ЗначениеИнтенсивности» );
            result = list.toArray( new String[ list.size() ] );
            _parameterNames_xjal = result;
        }
        return result;
    }

    public
double
variable;
    @AnyLogicInternalCodegenAPI
    private static Map<String, IElementDescriptor> elementDescriptors_xjal =
null;

    @AnyLogicInternalCodegenAPI
    @Override
    public Map<String, IElementDescriptor> getElementDescriptors() {
        if (elementDescriptors_xjal == null) {
            elementDescriptors_xjal = createEl-
ementDescriptors( super.getElementDescriptors(), Main.class );
        }
        return elementDescriptors_xjal;
    }

    @AnyLogicCustomProposalPriority(type = AnyLogicCustomProposalPriori-
ty.Type.STATIC_ELEMENT)
    public static final Scale scale = new Scale( 10.0 );

    @Override
    public Scale getScale() {
        return scale;
    }

    public EventTimeout _data_autoUpdateEvent_xjal = new EventTimeout(this);
    public EventTimeout _datal_autoUpdateEvent_xjal = new EventTimeout(this);

    public EventTimeout _chart1_autoUpdateEvent_xjal = new
EventTimeout(this);

    public String getNameOf( EventTimeout _e ) {
        if ( _e == _data_autoUpdateEvent_xjal ) return «data auto update event»;
        if ( _e == _datal_autoUpdateEvent_xjal ) return «datal auto update
event»;
        if ( _e == _chart1_autoUpdateEvent_xjal ) return «chart1 auto update
event»;
        return super.getNameOf( _e );
    }

    public EventTimeout.Mode getModeOf( EventTimeout _e ) {
        if ( _e == _data_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
        if ( _e == _datal_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
        if ( _e == _chart1_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
        return super.getModeOf( _e );
    }
}

```

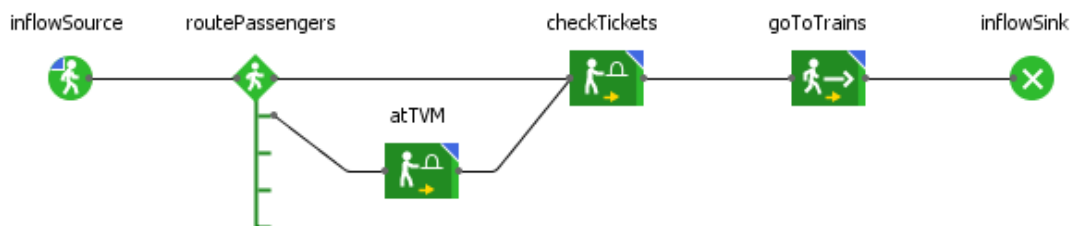


Рисунок 4 – Диаграмма процесса

4. Отображение карты плотности пешеходов (рисунок 5).

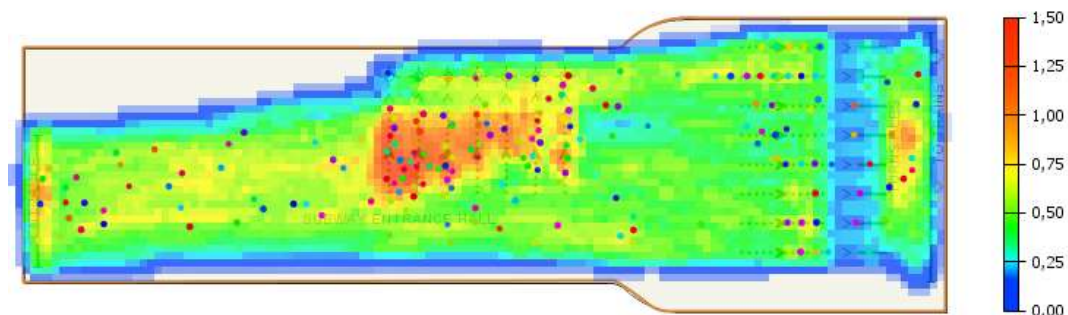


Рисунок 5 – Карта плотности пешеходов

5. Добавление блоков для сбора статистики (диаграмм, гистограмм и т. д.).
6. Создание эксперимента: задание оптимизируемого параметра, целевой функции и проведение оптимизации.

### Задание к практической работе № 2

1. Создайте пешеходную модель, описывающую движение людей на вокзале. Для улучшения наглядности разработайте меню и области просмотра.
2. Постройте круговую диаграмму **времени, проведенного в системе пешеходами-служащими и пешеходами-женщинами** (можно выбрать другие типы пешеходов, присутствующие в модели).
3. Постройте гистограмму **среднего времени, проведенного пешеходом в очереди к кассе**.
4. Проведите оптимизацию модели. Определите интенсивность, при которой потери времени пешеходов в системе будут **минимальны**. В качестве ограничения установить **размер очереди у касс** равным 5 пешеходам.

Сроки контроля – 9 неделя семестра.

Форма контроля – ответы на контрольные вопросы, отчёт по ПР№2.

### Требования к отчёту по практической работе №2

Отчёт представляется в электронном виде, сохраняется на компьютере до защиты практической работы. Отчёт должен содержать:

1. Представление модели в формальном виде, с указанием переменных входа/выхода, внутренней логики модели, целевой функции и ограничений (для оптимизационных моделей);
2. Описание основных блоков потоковой диаграммы, используемых в модели;
3. Потоковую диаграмму;
4. Анимацию модели;
5. Блоки сбора статистики по работе блоков диаграммы процесса;
6. Оптимизационный эксперимент и результаты оптимизации модели (для оптимизационных моделей);
7. Анализ результатов оптимизации модели.

### **Контрольные вопросы:**

1. Основные компоненты реализации дискретно-событийной модели.
2. Диаграмма процесса.
3. Компоненты дискретно-событийной модели.
4. Имитационное моделирование системы массового обслуживания.
5. Оптимизация модели.
6. Основные компоненты системы массового обслуживания.
7. Опишите цикл функционирования системы массового обслуживания.
8. Опишите основные характеристики систем массового обслуживания.
9. Опишите параметры систем с одним устройством обслуживания.
10. Опишите параметры многоканальных системы массового обслуживания.
11. Основные блоки потоковой диаграммы.
12. Отслеживание состояния объектов при моделировании.
13. Применение дискретно-событийного моделирования.
14. Для чего применяются динамические значения параметров в окне презентации модели?
15. Как запустить модель на выполнение?
16. Что такое виртуальное время?
17. Как переключиться из режима виртуального времени в реальное?
18. Как изменить скорость выполнения модели?
19. Как изменяются режимы отрисовки изображения?
20. В чем смысл эксперимента в программе имитационного моделирования?
21. Какие типы экспериментов поддерживаются программой имитационного моделирования?
22. Как изменить текущие значения переменных и параметров модели при ее выполнении?
23. Как показать график изменения переменной модели?

### Практическая работа №3

#### Моделирование движения автомобилей и пешеходов на участке УДС

**Цель работы:** Построение, анализ и оптимизация модели движения автомобилей и пешеходов на участке УДС.

#### Теоретические положения

Движение автомобилей можно смоделировать с помощью Библиотеки моделирования процессов или с помощью Библиотеки дорожного движения.

#### *Моделирование с помощью Библиотеки моделирования процессов*

В первую очередь строится диаграмма состояний для светофорного объекта, добавляются переменные и задаются их первоначальные значения (рисунок 6). Для каждого блока диаграммы состояний прописываются действия при входе и действия при выходе.

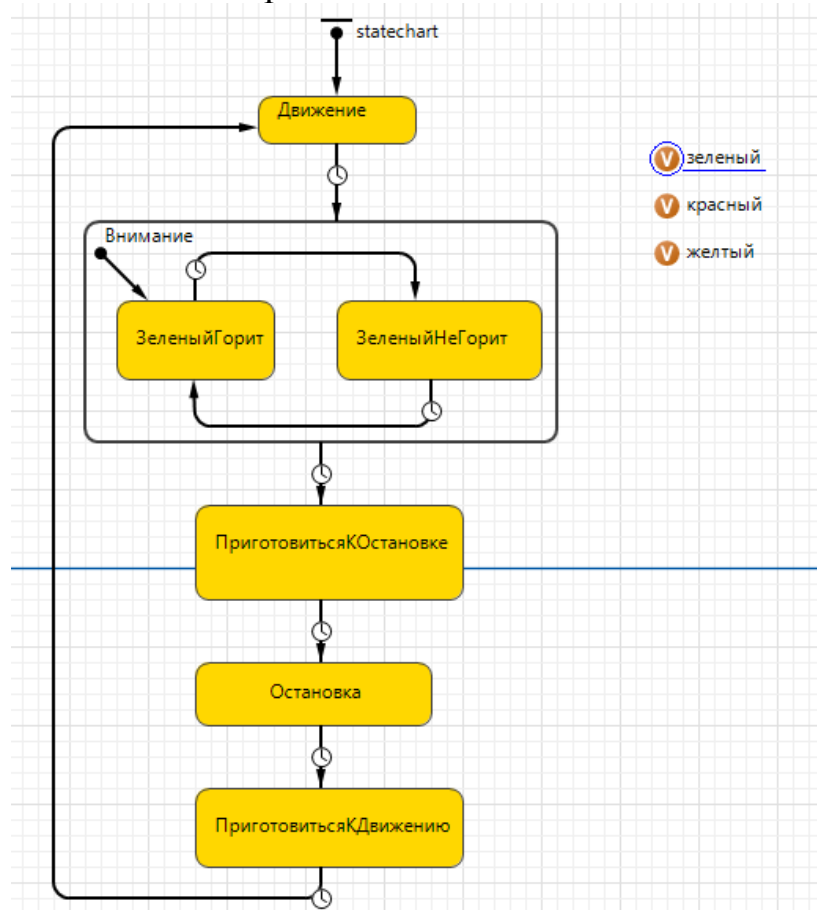


Рисунок 6 – Диаграмма состояний

Затем строится потоковая диаграмма для моделирования движения транспортных средств, настраиваются свойства каждого объекта, входящего в неё (рисунок 7).

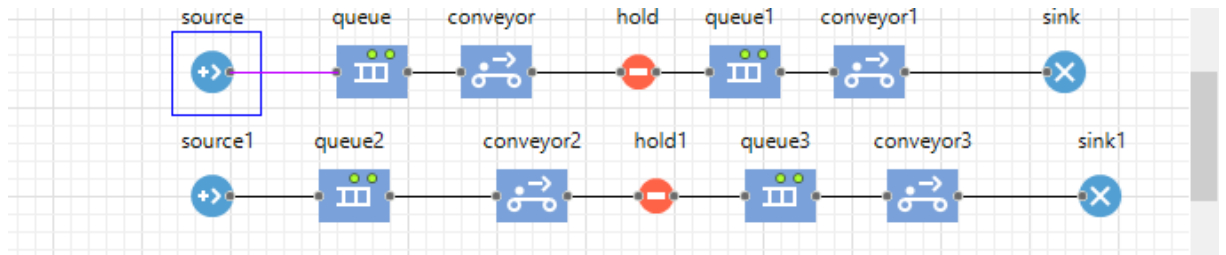


Рисунок 7 – Потокковая диаграмма

После этого строится диаграмма состояний для пешеходного светофора (рисунок 8). Для безопасной работы пешеходного перехода необходимо синхронизировать срабатывания диаграмм состояний для пешеходного светофора и транспортного светофора так, чтобы всегда, когда пешеходный светофор находится в состояниях ДвижениеПешеходов или Мигает, транспортный светофор обязательно находился бы в состоянии Остановка.

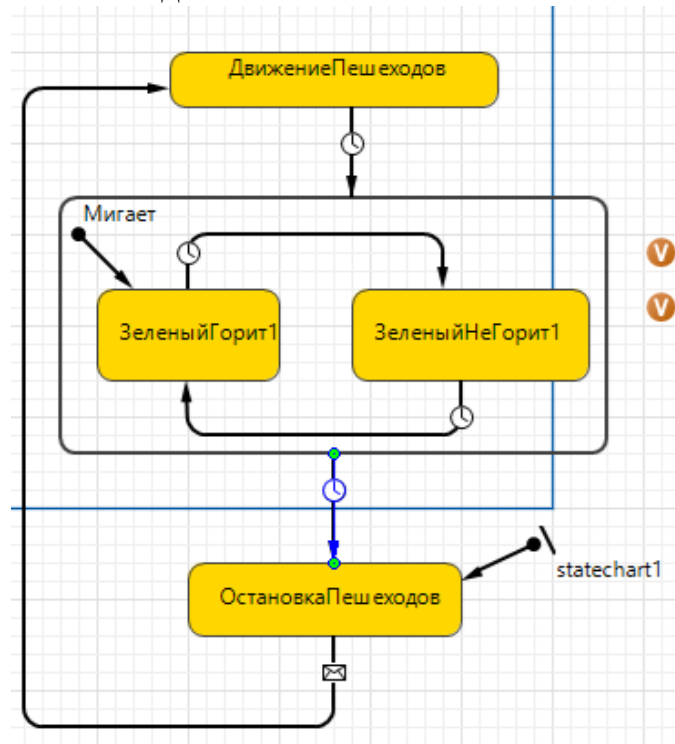


Рисунок 8 – Диаграмма состояний

Для этого можно подобрать подходящие таймауты срабатывания переходов диаграмм состояний, но при каждом изменении модели, придется эти таймауты подбирать заново. Лучше всего синхронизировать диаграммы состояний, посылая специальные разрешающие сигналы из одной диаграммы состояний в другую, используя методы `receiveMessage ()` или `fireEvent ()`.

Затем добавляется потокковая диаграмма движения пешеходов.

Автоматизированное светофорное регулирование, применение табло вызывного пешеходного (ТВП), переключение на ночной режим и т.д. достигается с помощью изменения диаграммы состояния светофорных объектов

(рисунок 9) и добавления дополнительных объектов в модель, например кнопок, надписей и т. д. (рисунок 10).

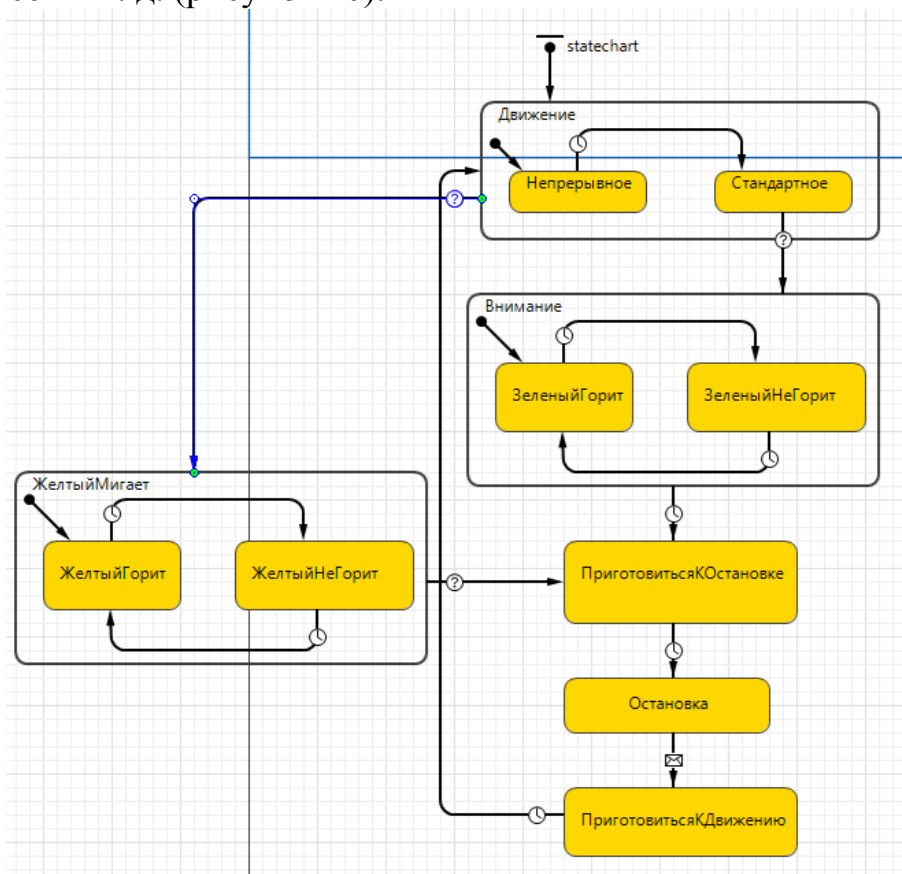


Рисунок 9 – Диаграмма состояний

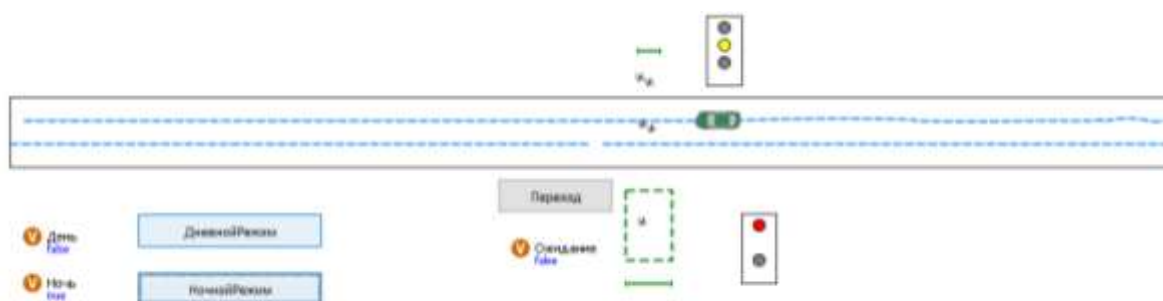


Рисунок 10 – Добавление ТВП и переключения в дневной/ночной режимы

### **Моделирование с помощью Библиотеки дорожного движения**

Библиотека дорожного движения позволяет моделировать и визуализировать движение потоков автотранспортных средств. С ее помощью можно промоделировать как движение транспортных средств на автомагистрали, так и на УДС, перевозку грузов, парковки и т. д. С помощью этой библиотеки можно моделировать крупномасштабные транспортные системы.

Основным объектом библиотеки является класс Car, который реализует движение, торможение, ускорение, перестроение на другую полосу и обнаружение столкновений. Автомобиль в библиотеке является пассивной заявкой.

Экземпляры класса Car создаются объектом CarSource, после чего они следуют по маршруту, задаваемому объектом CarMove To, при этом их дви-



жением управляет водитель – объект, реализующий интерфейс Driver. При создании автомобиля объектом CarSource используется заданный по умолчанию водитель класса Standard Driver.

Элементы, используемые для создания, продвижения и удаления автомобилей из модели:

- *Объект CarSource.* Создает транспортные средства, помещает их на одну из дорог и вставляет заявку типа Car в диаграмму процесса, задающую автотрафик. С объекта CarSource обычно начинается диаграмма процесса дорожного трафика.

- *Объект CarDispose.* Удаляет транспортные средства из модели.

- *Объект CarMove To.* Объект CarMove To управляет движением автомобиля. Для каждого автомобиля-заявки объект CarMove To высчитывает маршрут от текущего местоположения до заданной точки назначения и направляет автомобиль по этому маршруту. При достижении автомобилем точки назначения, он может как остановиться, так и продолжить движение без заданного маршрута. Пока автомобиль находится под управлением объекта CarMove To, им можно управлять с помощью методов класса Car. Объект CarMove To предоставляет в распоряжение ряд параметров, позволяющих задавать код, который будет выполняться в ключевые моменты движения автомобиля.

Задание длины агента Car осуществляется с помощью вызова пользовательской функции Длина Авто (). В тело функции записывается код, позволяющий с помощью функции uniform сгенерировать равномерно распределенную в интервале (0, 10) случайную величину rnd (тип данных double). Затем, с помощью оператора условия if проверяем, в какой диапазон попадает случайное число rnd. Если rnd попадает в диапазон от [0, 3], то функция Длина Авто () возвращает значение 10, если rnd попадает в диапазон от (3, 5), то функция Длина Авто () возвращает значение 6, во всех остальных случаях функция возвращает значение 5. В зависимости от состава транспортного потока можно настроить другие значения интервалов.

Задание фигуры анимации агента Car осуществляется с помощью пользовательской функции, которая возвращает одну из 7-и фигур анимации (bus\_2, truck, lorry, tractor, car1, car2, car3 – имена 3D объектов, находящихся на диаграмме класса Main):

```
double l = car.getLength();
int rnd = uniform_discr( l, 2 );
if( l == 10 ) {
    if( rnd == 1 ) return bus_2;
    if( rnd == 2 ) return truck;
}
if( l == 6 ) return lorry;
rnd = uniform_discr( l, 4 );
switch( rnd ) {
    case 1: return tractor;
    case 2: return car1;
    case 3: return car2;
    case 4: return car3;
}
return null;
```

Объект Road Network задает и отображает сеть дорог на анимации во время выполнения модели. Этот объект должен присутствовать в любой модели дорожного движения.

Создание сети дорог осуществляется в следующем порядке:

- Добавляем на диаграмму класса Main линии или дуги, изменяем толщину и высоту ( $Z = 0$ ). Соединяем.
- Группируем все нарисованные и соединенные участки дорог в 2 направлениях.
- Задаём свойства объекта Road Network.

Элементы, используемые для создания, продвижения и удаления поездов из модели находятся в Желенодорожной библиотеке:

*Объект Train Source.* Этот блок создает поезд, помещает его на один из путей ж/д узла, и вставляет агента-поезд в диаграмму процесса движения поезда. Новый поезд помещается на путь, и на этом пути должно быть свободное место для всех вагонов (при создании поезд должен полностью находиться на пути).

*Объект Train Dispose.* Удаляет поезда из модели.

*Объект Train Move To.* Единственный объект, который управляет движением поезда. Поезд может перемещаться только тогда, когда он находится в объекте Train Move To. Точка назначения на пути задается либо как расстояние от начала пути (в метрах), либо графически с помощью элемента Точка ж/д пути.

В данной модели необходимо блокировать движение ТС при появлении поезда. Достигается это с помощью диаграммы состояний для шлагбаума и настройки соответствующих свойств объектов, находящихся на диаграмме класса Main. Логике модели можно задать на языке Java:

```
public Statechart ДиаграммаСостояний = new Statechart( this, (short)1 );
public String getNameOf( Statechart _s ) {
    if(_s == this.ДиаграммаСостояний) return «ДиаграммаСостояний»;
    return super.getNameOf( _s );
}
public void executeActionOf( Statechart _s ) {
    if( _s == this.ДиаграммаСостояний ) {
        enterState( Открыт, true );
        return;
    }
    super.executeActionOf( _s );
}

@AnyLogicCustomProposalPriority(type = AnyLogicCustomProposalPriority.Type.STATIC_ELEMENT)
public static final short Открыт = 0;
@AnyLogicCustomProposalPriority(type = AnyLogicCustomProposalPriority.Type.STATIC_ELEMENT)
public static final short Закрывается = 1;
@AnyLogicCustomProposalPriority(type = AnyLogicCustomProposalPriority.Type.STATIC_ELEMENT)
public static final short Закрыт = 2;
@AnyLogicCustomProposalPriority(type = AnyLogicCustomProposalPriority.Type.STATIC_ELEMENT)
public static final short Открывается = 3;
```

```

protected static final short _STATECHART_ELEMENT_NEXT_ID_xjal = 4;
public boolean inState( short _state ) {
    switch( _state ) {
        case Открыт:
        case Закрывается:
        case Закрыт:
        case Открывается:
            return ДиаграммаСостояний.isStateActive( _state );
        default: return super.inState( _state );
    }
}

@Override
public String getNameOfState( short _state ) {
    switch( _state ) {
        case Открыт: return «Открыт»;
        case Закрывается: return «Закрывается»;
        case Закрыт: return «Закрыт»;
        case Открывается: return «Открывается»;
        default: return super.getNameOfState( _state );
    }
}

public void enterState( short _state, boolean _destination ) {
    switch( _state ) {
        case Открыт:
            logToDBEnterState(ДиаграммаСостояний, _state);
            ДиаграммаСостояний.setActiveState_xjal( Открыт );
            {
hold1.unblock();
hold2.unblock();
; }
            transition.start();
            return;
        case Закрывается:
            logToDBEnterState(ДиаграммаСостояний, _state);
            ДиаграммаСостояний.setActiveState_xjal( Закрывается );
            {
hold1.block();
hold2.block();
; }
            ClosingCompleted.start();
            return;
        case Закрыт:
            logToDBEnterState(ДиаграммаСостояний, _state);
            ДиаграммаСостояний.setActiveState_xjal( Закрыт );
            transition2.start();
            return;
        case Открывается:
            logToDBEnterState(ДиаграммаСостояний, _state);
            ДиаграммаСостояний.setActiveState_xjal( Открывается );
            OpeningCompleted.start();
            return;
        default:
            super.enterState( _state, _destination );
            return;
    }
}

@Override
@AnyLogicInternalCodegenAPI
public void exitState( short _state, Transition _t, boolean _source,
Statechart _statechart ) {
    switch( _state ) {

```

```

    case Открыт:
        logToDBExitState(ДиаграммаСостояний, _state);
        logToDB(ДиаграммаСостояний, _t, _state);
        if ( !_source || _t != transition) transition.cancel();
        return;
    case Закрывается:
        logToDBExitState(ДиаграммаСостояний, _state);
        logToDB(ДиаграммаСостояний, _t, _state);
        if ( !_source || _t != ClosingCompleted) ClosingCompleted.cancel();
        return;
    case Закрыт:
        logToDBExitState(ДиаграммаСостояний, _state);
        logToDB(ДиаграммаСостояний, _t, _state);
        if ( !_source || _t != transition2) transition2.cancel();
        return;
    case Открывается:
        logToDBExitState(ДиаграммаСостояний, _state);
        logToDB(ДиаграммаСостояний, _t, _state);
        if ( !_source || _t != OpeningCompleted) OpeningCompleted.cancel();
        return;
    default:
        super.exitState( _state, _t, _source, _statechart);
        return;
}
}

```

Для задания положения шлагбаума необходимо разместить на диаграмме класса Main функцию ПоложениеШлагбаума(), которая бы проверяла активное состояние Диаграммы Состояний с помощью функции getActiveSimpleState (), а в зависимости от активного состояния Диаграммы Состояний возвращала бы угол поворота шлагбаума. Примерный вид функции:

```

switch( ДиаграммаСостояний.getActiveSimpleState() ) {
case Открыт: return PI/2;
case Закрывается: return PI / 4;
case Закрыт: return 0;
case Открывается: return PI / 6;
}
return 0;

```

### Задание к практической работе № 3

1. Построить модель движения автомобилей через заданный перекрёсток.
2. Установить возможность изменения времени горения красного и зеленого сигнала светофора.
3. Построить модель движения пешеходов через заданный перекрёсток. Пешеходы должны двигаться в 2-х направлениях.
4. Установить на пешеходном переходе кнопку Переход и запрограммировать её.
5. Добавить кнопки перехода на ночной режим и дневной режим. Измените презентацию модели таким образом, чтобы рядом со светофором высвечивалось время в секундах, оставшееся до смены сигнала.

6. Создать модель дорожно-транспортной развязки с железнодорожным переездом.

Сроки контроля – 13 неделя семестра.

Форма контроля – ответы на контрольные вопросы, отчёт по ПР№3.

### **Требования к отчёту по практическим работам №3**

Отчёт представляется в электронном виде, сохраняется на компьютере до защиты практической работы. Отчёт должен содержать:

1. Представление модели в формальном виде, с указанием переменных входа/выхода, внутренней логики модели, целевой функции и ограничений (для оптимизационных моделей);
2. Описание основных блоков потоковой диаграммы, используемых в модели;
3. Потоковую диаграмму;
4. Анимацию модели;
5. Блоки сбора статистики по работе блоков диаграммы процесса;
6. Оптимизационный эксперимент и результаты оптимизации модели (для оптимизационных моделей);
7. Анализ результатов оптимизации модели.

### **Контрольные вопросы:**

1. Агенты и их типы.
2. Создание агентов. Параметры типа Агент.
3. Влияние поведения децентрализованных агентов на поведение всей системы в целом.
4. Область применения агентного моделирования
5. Агентная синхронизация.
6. Состояния, подвижность и анимация агентов.
7. Основные блоки потоковой диаграммы.
8. Отслеживание состояния объектов при моделировании.
9. Для чего применяются динамические значения параметров в окне презентации модели?
10. Как запустить модель на выполнение?
11. Что такое виртуальное время?
12. Как переключиться из режима виртуального времени в реальное?
13. Как изменить скорость выполнения модели?
14. В чем смысл эксперимента в программе имитационного моделирования?
15. Какие типы экспериментов поддерживаются программой имитационного моделирования?
16. Как изменить текущие значения переменных и параметров модели при ее выполнении?
17. Как показать график изменения переменной модели?

18. Агентные связи и их взаимодействие друг с другом.
19. Динамическое создание и уничтожение агентов.
20. Поддержка агентного моделирования в AnyLogic.

## Практическая работа №4

### Использование системной динамики для моделирования транспортных систем

**Цель работы:** Изучение основ системной динамики

#### Теоретические положения

Системная динамика – направление в изучении сложных систем, исследующее их поведение во времени и в зависимости от структуры элементов системы и взаимодействия между ними. В том числе: причинно-следственных связей, петель обратных связей, задержек реакции, влияния среды и других.

Для изучения основ системной динамики часто используют модель распространения нового продукта по Бассу, которая описывает процесс распространения изначально никому не известного продукта. Для того чтобы люди начали его приобретать, он рекламируется в СМИ, посредством «сарафанного радио», специалистами. В итоге определенная доля людей приобретает продукт под воздействием рекламы. Также люди приобретают продукт в результате общения с теми, кто этот продукт уже приобрел. Процесс приобретения нового продукта под влиянием убеждения его владельцев чем-то похож на распространение эпидемии.

Описание модели в терминах системной динамики производится следующим образом:

1. Определение ключевых переменных модели, задание их свойств:

```
public class Main extends Agent
{
    public
    double ЧисленностьНаселения;
    public double _ЧисленностьНаселения_DefaultValue_xjal() {
        final Main self = this;
        return
100000
    ;
    }
    public void set_ЧисленностьНаселения( double ЧисленностьНаселения ) {
        if (ЧисленностьНаселения == this.ЧисленностьНаселения) {
            return;
        }
        double _oldValue_xjal = this.ЧисленностьНаселения;
        this.ЧисленностьНаселения = ЧисленностьНаселения;
        onChange_ЧисленностьНаселения_xjal( _oldValue_xjal );
        onChange();
    }
    protected void onChange_ЧисленностьНаселения() {
        onChange_ЧисленностьНаселения_xjal( ЧисленностьНаселения );
    }

    @AnyLogicInternalCodegenAPI
    protected void onChange_ЧисленностьНаселения_xjal( double oldValue ) {
    }
    public
```

```

double ЧастотаОбщения;
public double _ЧастотаОбщения_DefaultValue_xjal() {
    final Main self = this;
    return
100
;
}
public void set_ЧастотаОбщения( double ЧастотаОбщения ) {
    if (ЧастотаОбщения == this.ЧастотаОбщения) {
        return;
    }
    double _oldValue_xjal = this.ЧастотаОбщения;
    this.ЧастотаОбщения = ЧастотаОбщения;
    onChange_ЧастотаОбщения_xjal( _oldValue_xjal );
    onChange();
}
protected void onChange_ЧастотаОбщения() {
    onChange_ЧастотаОбщения_xjal( ЧастотаОбщения );
}

@AnyLogicInternalCodegenAPI
protected void onChange_ЧастотаОбщения_xjal( double oldValue ) {
}
public
double ЭффективностьРекламы;
public double _ЭффективностьРекламы_DefaultValue_xjal() {
    final Main self = this;
    return 0.011 ;
}
public void set_ЭффективностьРекламы( double ЭффективностьРекламы ) {
    if (ЭффективностьРекламы == this.ЭффективностьРекламы) {
        return;
    }
    double _oldValue_xjal = this.ЭффективностьРекламы;
    this.ЭффективностьРекламы = ЭффективностьРекламы;
    onChange_ЭффективностьРекламы_xjal( _oldValue_xjal );
    onChange();
}

```

2. Определение, каким образом переменные модели влияют друг на друга.

3. Создание потоковой диаграммы модели. При создании потоковой диаграммы необходимо учесть, какие переменные должны быть представлены накопителями, какие потоками, а какие – динамическими переменными.

Переменные, которые накапливают значения с течением времени (накопители):

- численность потребителей продукта
- численность потенциальных потребителей продукта.

Процесс приобретения продукта является потоком.

Системно-динамическое представление модели показано на рисунке 4.

4. Накопители обозначаются прямоугольниками, поток – вентилем, а динамические переменные – кружками. Стрелки обозначают причинно-следственные зависимости в модели.



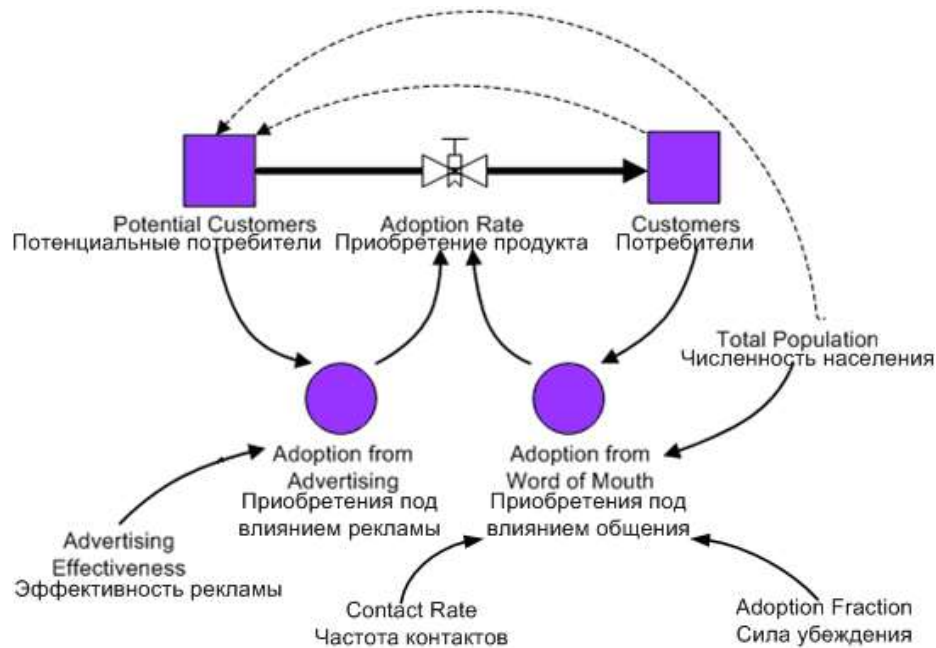


Рисунок 11 – Системно-динамическое представление модели

Исследование динамики обеих составляющих потока продаж осуществляется в процессе имитации (рисунок 12).

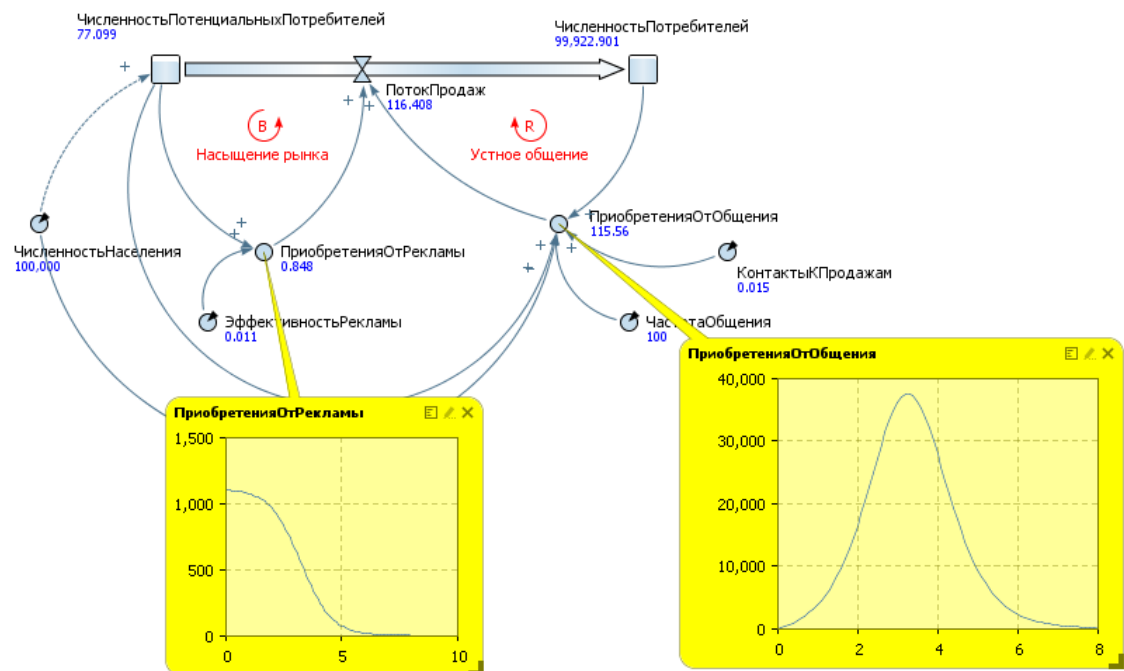


Рисунок 12 – Имитация

Анализ модели позволяет сделать вывод, что при внедрении нового продукта на рынок, когда число потребителей равно нулю, реклама будет являться единственным источником продаж. Наибольший рекламный эффект отмечается в начале процесса распространения продукта; он неуклонно падает по мере уменьшения численности потенциальных потребителей.

Приобретения товара в результате общения максимально на 3 году продвижения товара. Затем происходит уменьшение количества продаваемого товара за счёт снижения количества потенциальных потребителей, которые переходят в число потребителей.

#### **Задание к практической работе №4**

1. Создать модель распространения любой продукции (автомобилей, транспортных карт и т. д.).
2. Создать модель функционирования транспортной системы.

Сроки контроля – 17 неделя семестра.

Форма контроля – ответы на контрольные вопросы, отчёт по ПР№4.

#### **Требования к отчёту по практической работе №4**

Отчёт представляется в электронном виде, сохраняется на компьютере до защиты практической работы. Отчёт должен содержать:

1. Представление модели в формальном виде, с указанием переменных входа/выхода, внутренней логики модели, целевой функции и ограничений (для оптимизационных моделей);
2. Описание основных блоков потоковой диаграммы, используемых в модели;
3. Потоковую диаграмму;
4. Анимацию модели;
5. Блоки сбора статистики по работе блоков диаграммы процесса;
6. Оптимизационный эксперимент и результаты оптимизации модели (для оптимизационных моделей);
7. Анализ результатов оптимизации модели.

#### **Контрольные вопросы:**

1. Элементы модели системной динамики.
2. Принципы построения модели.
3. Область применения.
4. Анализ системы. Описание модели в терминах системной динамики.
5. Определение ключевых переменных модели.
6. Определение, каким образом переменные модели влияют друг на друга.
7. Создание потоковой диаграммы модели.
8. Приведите варианты использования динамических моделей.
9. Приведите основные принципы при моделировании динамических систем.
10. Для чего предназначены параметры?
11. Для чего предназначены переменные?
12. В чем заключаются отличия динамической системы от модели системной динамики?
13. Создание накопителей и потока.
14. Опишите варианты использования модели системной динамики.

15. Опишите варианты использования объекта "накопитель".
16. Опишите варианты использования объекта "поток".
17. Опишите варианты использования объекта "динамическая переменная".
18. Опишите принципы создания связей между переменными.
19. Элементы системно-динамической модели.
20. Создание динамических переменных.
21. Создание связей в динамической модели.
22. Полярность связей в динамической модели.
23. Создание компенсирующего цикла с обратной связью.
24. Создание усиливающего цикла с обратной связью.
25. Настройка запуска модели. Задание остановки модели по прошествии n единиц модельного времени.

## Список литературы

1. Боев, В. Д. Концептуальное проектирование систем в Anylogic 7 и GPSS World. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 556 с. – Режим доступа: [http://biblioclub.ru/index.php?page=book\\_red&id=428950](http://biblioclub.ru/index.php?page=book_red&id=428950). – Загл. с экрана. (06.06.2016)
2. Волкова, В. Н. Моделирование систем : Подходы и методы: учебное пособие. – Санкт-Петербург : Издательство Политехнического университета, 2013. – 568 с. – Режим доступа: [http://biblioclub.ru/index.php?page=book\\_red&id=362986](http://biblioclub.ru/index.php?page=book_red&id=362986). – Загл. с экрана. (06.06.2016)
3. Кудряшов, В. С. Моделирование систем: учебное пособие. – Воронеж : Воронежский государственный университет инженерных технологий, 2012. – 208 с. – Режим доступа: [http://biblioclub.ru/index.php?page=book\\_red&id=141980](http://biblioclub.ru/index.php?page=book_red&id=141980). – Загл. с экрана. (06.06.2016)
4. Чикуров, Н. Г. Моделирование систем и процессов : учебное пособие для студентов вузов, обучающихся по направлению подготовки "Автоматизация технологических процессов и производств (машиностроение)" / Н. Г. Чикуров. – Москва : РИОР, 2013. – 398 с.
4. Афонин, В.В. Моделирование систем: учебно-практическое пособие. – Москва : Интернет-Университет Информационных Технологий, 2011. – 232 с. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=232979>. – Загл. с экрана. (02.03.2016)
6. Беликова, Н.А. Математическое моделирование: учебное пособие, Ч. 2. – Москва : Самарский государственный архитектурно-строительный университет, 2009. – 66 с. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=144941>. – Загл. с экрана. (02.03.2016)
7. Голубева, Н.В. Математическое моделирование систем и процессов. – Санкт-Петербург : Лань, 2013. – 192 с. – Режим доступа: [http://e.lanbook.com/books/element.php?pl1\\_id=4862](http://e.lanbook.com/books/element.php?pl1_id=4862). – Загл. с экрана. (11.03.2016)
8. Данилов, Н.Н. Математическое моделирование: учебное пособие. – Кемерово : Кемеровский государственный университет, 2014. – 98 с. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=278827>. – Загл. с экрана. (02.03.2016)
9. Мур, Д.Х. Экономическое моделирование в Microsoft Excel / Дж. Мур, Л. Уэдерфорд. – Москва : Вильямс, 2004. – 1024 с.
10. Петров, А.В. Моделирование процессов и систем. – Санкт-Петербург : Лань, 2015. – 288 с. – Режим доступа: [http://e.lanbook.com/books/element.php?pl1\\_id=68472](http://e.lanbook.com/books/element.php?pl1_id=68472). – Загл. с экрана. (02.03.2016)
11. Стариков, А.В. Экономико-математическое и компьютерное моделирование: учебное пособие. – Воронеж : Воронежская государственная лесотехническая академия, 2008. – 133 с. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=143139>. – Загл. с экрана. (02.03.2016)
12. Салмина, Н.Ю. Имитационное моделирование: учебное пособие. – Томск : Эль Контент, 2012. – 90 с. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=208690>. – Загл. с экрана. (09.03.2016)
13. Яхнеева, И.В. Моделирование и проектирование систем поставок в условиях риска. – Москва : БИБЛИО-ГЛОБУС, 2013. – 176 с. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=229658>. – Загл. с экрана. (02.03.2016)