МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт Информационных Технологий

Кафедра Вычислительной Техники

Практические работы
по дисциплине
«Многоагентное моделирование»

Студент группы: ИКБО-04-22 Кликушин В.И.___
(Фамилия студента)

Преподаватель __Гололобов А.А.___
(Фамилия преподавателя)

Москва 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Открытый характер современного информационного общества и глобальной рыночной экономики приводит к ускорению научно-технического прогресса и обострению конкуренции на рынках. Это заставляет предприятия искать новые методы и средства организации и управления, направленные на более качественное и эффективное удовлетворение индивидуальных запросов потребителей. Большинство современных систем характеризуются отсутствием средств своевременной идентификации новых потребностей и возможностей в среде, позволяющих предприятию оперативно принимать эффективные решения по реконфигурации производственных, кадровых, финансовых и других ресурсов. Типичными примерами событий, вызывающих необходимость заново идентифицировать потребности и возможности, являются: появление нового выгодного заказа, для исполнения которого недостаточно собственных ресурсов предприятия, выход из строя части имеющихся ресурсов, а также изменение критериев принятия решений.

Чем выше неопределенность, чем более распределенный характер имеют процессы принятия решения и чем чаще случаются незапланированные события, тем ниже эффективность существующих систем, не способных самостоятельно принимать решения и автоматически перестраиваться под изменения в среде. Кроме того, необходимость модификации схемы принятия решений в традиционных системах оказывается сложной и трудоемкой задачей, которая требует высокой квалификации исполнителей. Это делает разработку и эксплуатацию таких систем крайне дорогостоящими. Соответственно, еще одной актуальной проблемой современности становится рост объемов информации и степени сложности описания систем.

Для решения подобных проблем применяются мультиагентные технологии, в основе которых лежит понятие «агента», которое в последнее время было адаптировано ко многим областям как прикладного и системного программирования, так и к исследованиям в областях искусственного

интеллекта и распределенных интеллектуальных систем.

Сложные системы часто рассматривают как среду действия агентов. С понятием сложных систем связаны следующие фундаментальные идеи, которые непосредственно влияют на функционирование МАС.

• в сложных системах существуют автономные объекты, которые взаимодействуют друг с другом при выполнении своих определенных задач;

• агенты должны иметь возможность реагировать на изменяющиеся условия среды, в которой они функционируют и, возможно, изменять свое поведение на основе полученной информации;

• сложные системы характеризуются возникающими структурами – логически связанными схемами, которые формируется в результате взаимодействия между агентами;

• сложные системы с возникающими структурами часто существуют на грани порядка и хаоса;

• при создании сложных систем на базе агентов имеет смысл рассматривать биологические аналогии, такие как: паразитизм, симбиоз, репродукцию, генетику и естественный отбор (например, компания British Telecom при формировании сети направления звонков использует модель деятельности колонии муравьев).

1 МНОГОАГЕНТНОЕ МОДЕЛИРОВАНИЕ

1.1 Постановка задачи

Необходимо построить модель, описывающую распространение смертельного вируса среди человеческой популяции. С точки зрения реализации, каждый отдельный человек является агентом. Изначально предполагается, что все люди здоровы. Вирус распространяется с заданной эффективностью, более того, уже заражённые люди также могут заражать здоровых людей при контакте. При наличии свободных мест инфицированные люди попадают в больницу, где им вводится вакцина. Такие люди после короткой реабилитации полностью выздоравливают, однако, в будущем могут быть заражены снова. Длительное нахождение вируса в организме приводит к летальному исходу, однако, при небольшом количестве погибших, людей возможно «Вернуть к жизни».

Требования к модели:

- Количество состояний агента – не менее 3;
- Количество параметров агента - не менее 5;
- Размер популяции агентов – не менее 500;
- Возможность динамического изменения параметров – не менее 1;
- Наличие условий в карте агента: не менее 1;
- Наличие графика/ов, для отслеживания динамики изменения состояний популяции.

1.2 Описание этапов выполнения работы

Была создана популяция агентов People, затем была построена диаграмма состояний, задающая поведение агентам (Рисунок 1.2.1).

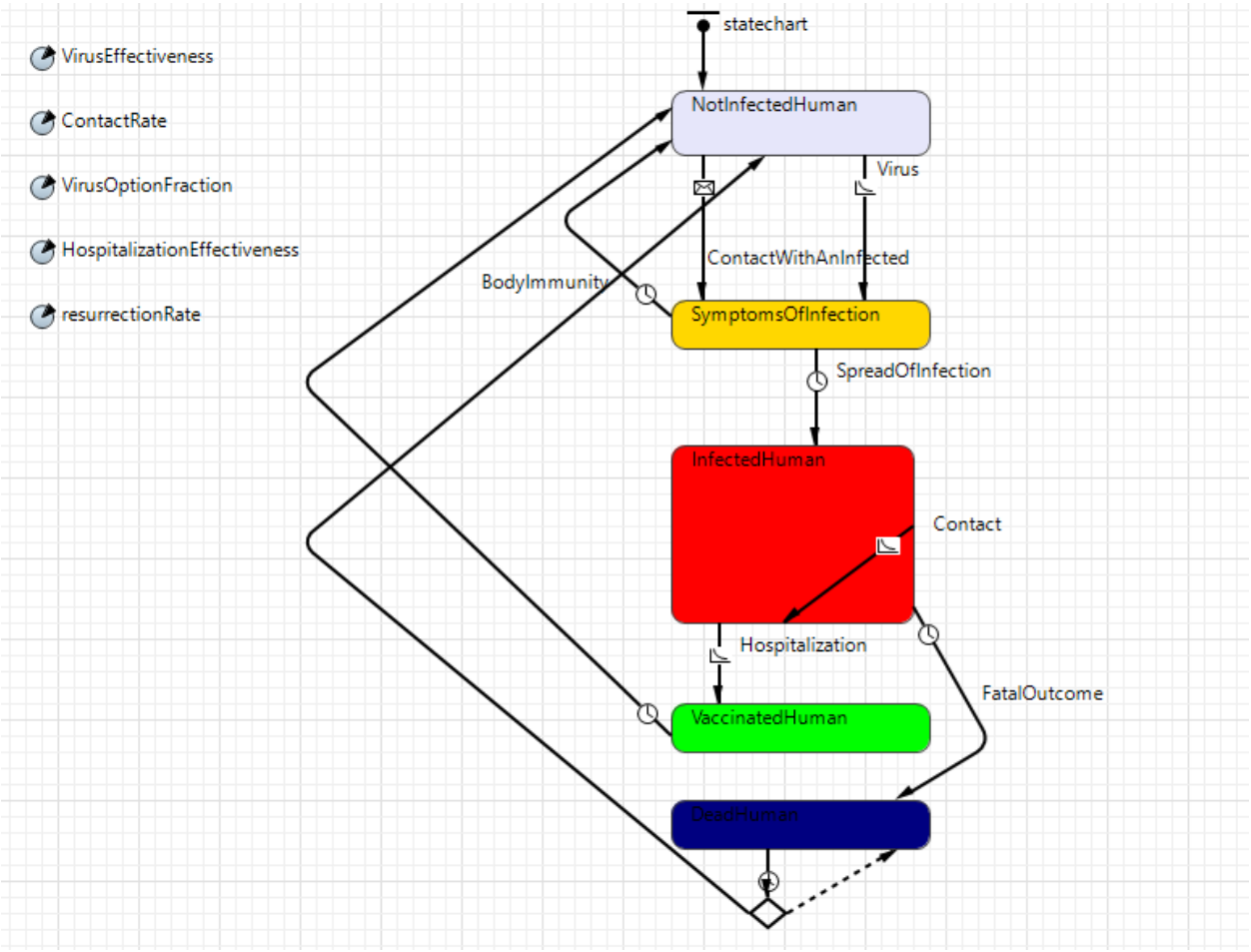

Рисунок 1.2.1 – Диаграмма состояний

Для моделирования переходов между состояниями были заданы следующие параметры:

- MaxRecoveryTime – максимальный срок, при котором организм может самостоятельно справиться с вирусом;
- MaxSpreadOfInfectionTime – время, за которое вирус распространяется в организме;
- DeathRate – среднее количество дней, после которого человек погибает, будучи заражённым;
- VirusEffectiveness – эффективность (заразность) вируса;
- ContactRate – интенсивность контакта между людьми;
- VirusOptionFraction – шанс заразиться при контакте с инфицированным человеком;
- HospitalizationEffectiveness – эффективность госпитализации;

- resurrectionRate – интенсивность «Воскрешения».

Подробнее опишем, как происходят переходы между состояниями.

Здоровый человек получает симптомы вируса, согласно интенсивности распространения (Рисунок 1.2.2).

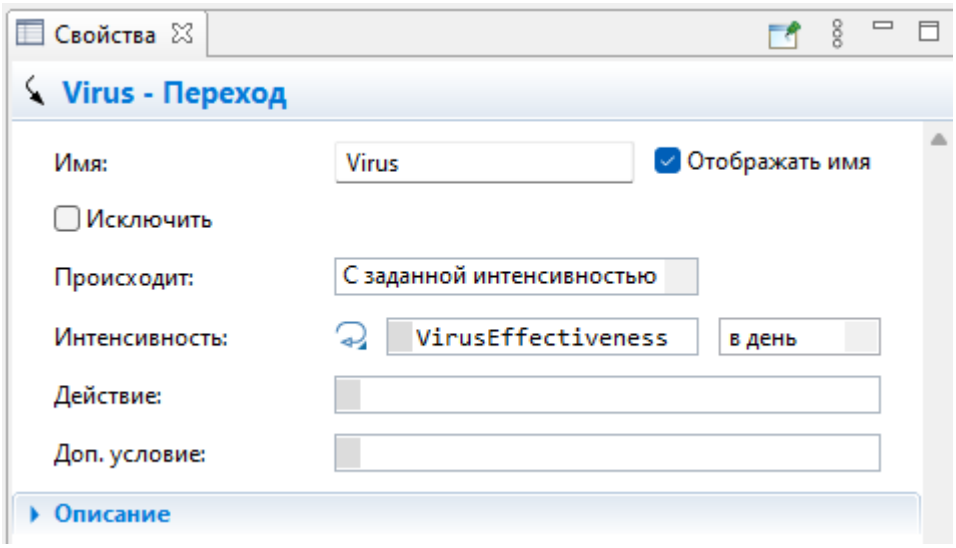

Рисунок 1.2.2 – Переход «virus»

Также, здоровый человек может обрести симптомы вируса вступив в контакт с заражённым (Рисунок 1.2.3).

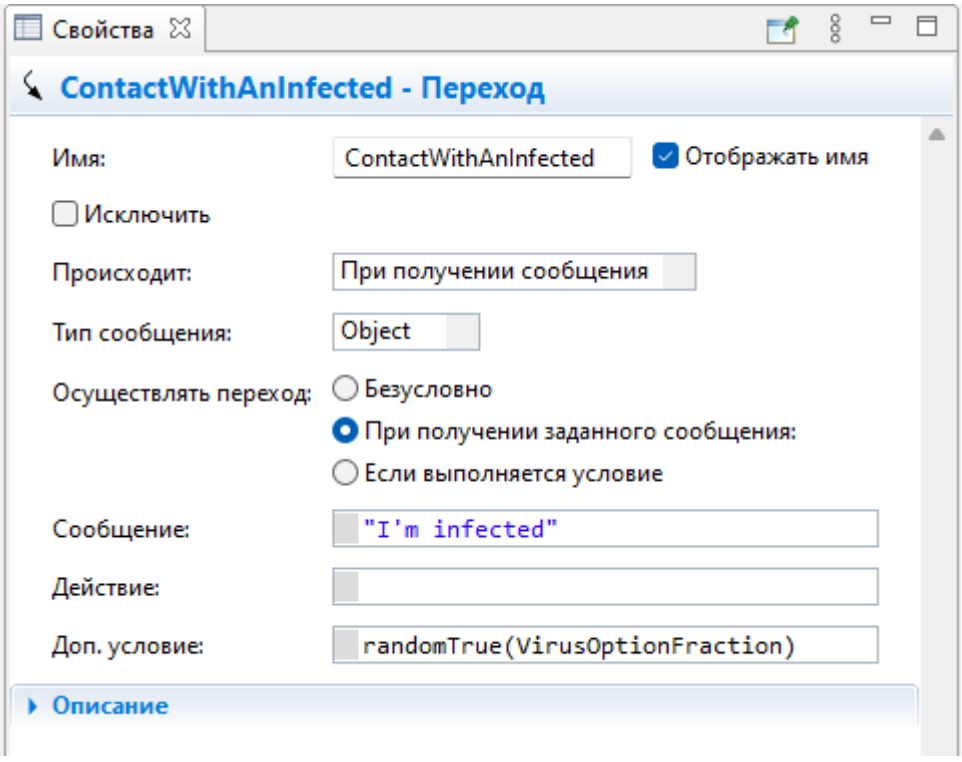

Рисунок 1.2.3 – Переход «ContactWithAnInfected»

На ранних стадиях заражения иммунитет может самостоятельно

справиться с заразой (Рисунок 1.2.4).

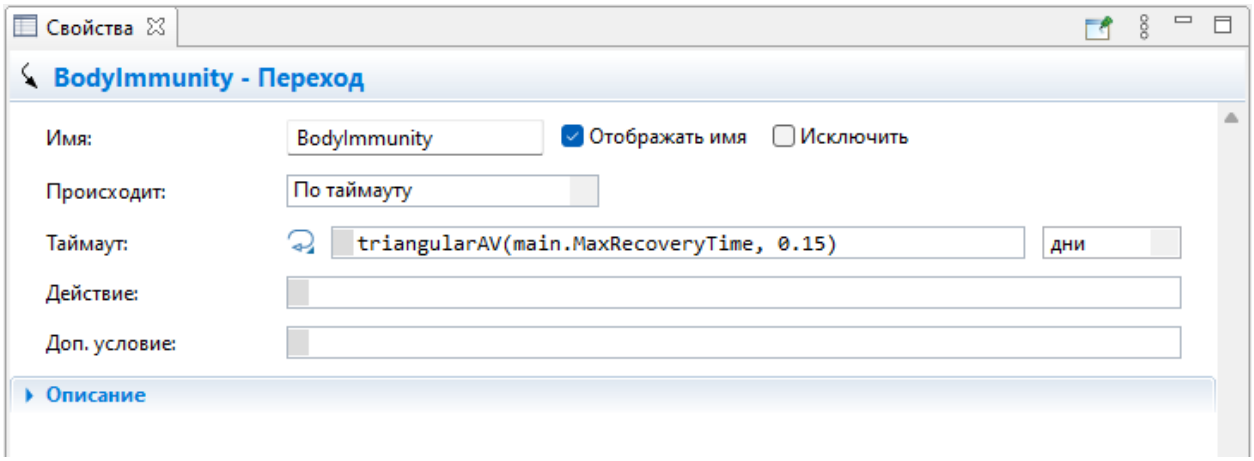

Рисунок 1.2.4 – Переход «BodyImmunity»

Вирус распространяется в организме в течение нескольких дней, состояние человека ухудшается (Рисунок 1.2.5).

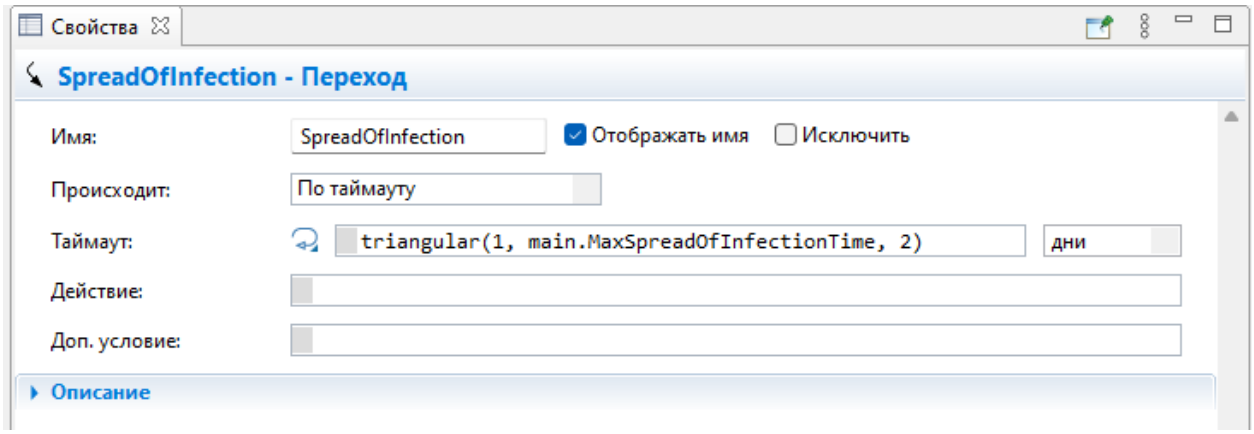

Рисунок 1.2.5 – Переход «SpreadOfInfection»

Инфицированные вступают в контакт со здоровыми людьми, в результате чего последние заражаются (Рисунок 1.2.6).

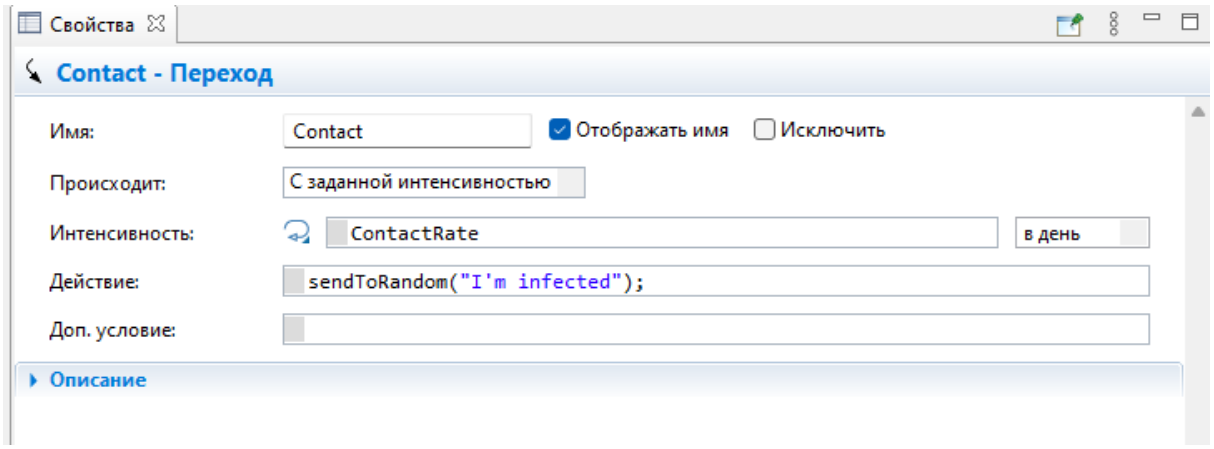

Рисунок 1.2.6 – Переход «Contact»

Часть людей отправляются в больницы на госпитализацию (Рисунок 1.2.7).

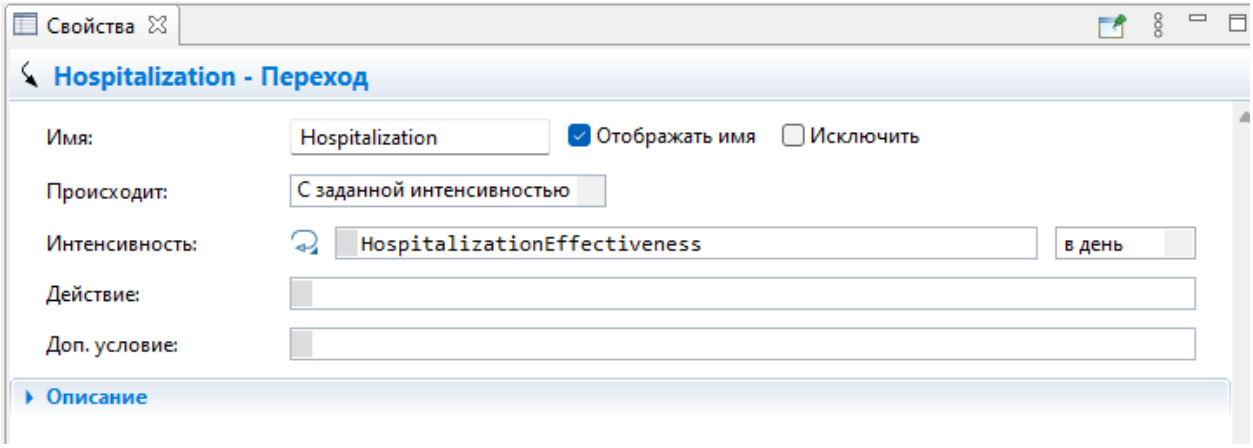

Рисунок 1.2.7 – Переход «Hospitalization»

При отсутствии введённой вакцины спустя некоторое время наступает летальный исход (Рисунок 1.2.8).

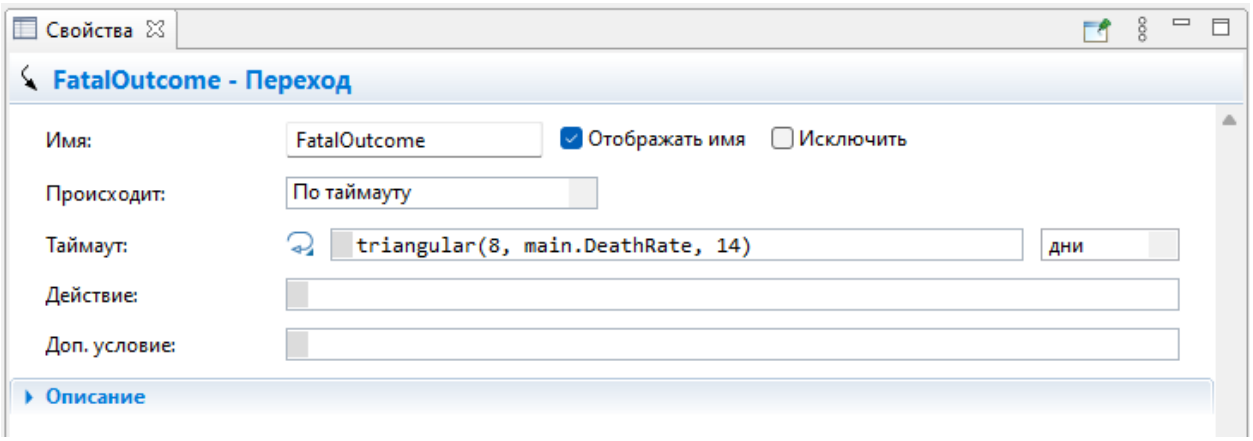

Рисунок 1.2.8 – Переход «FatalOutcome»

Вакцинированный человек полностью выздоравливает и выписывается через несколько дней (Рисунок 1.2.9).

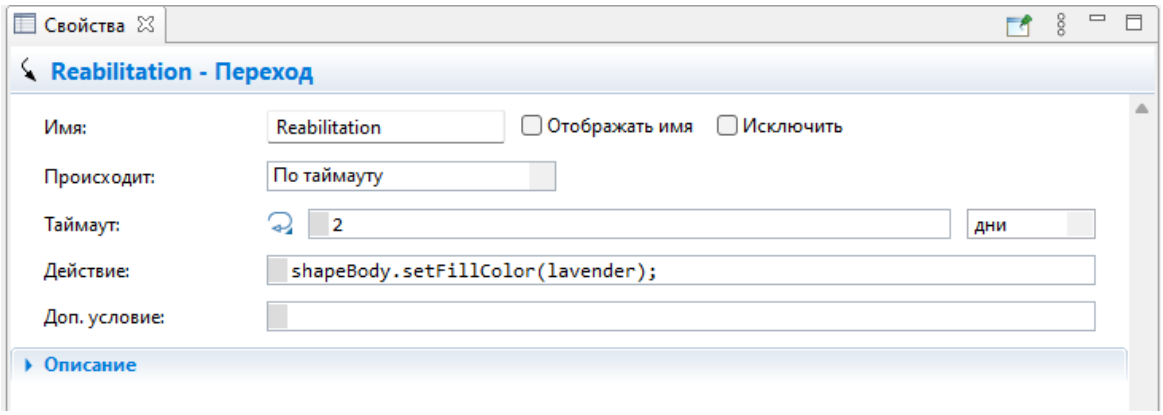

Рисунок 1.2.9 – Переход «Reabilitation»

Человек может воскреснуть, пока количество погибших не слишком велико (Рисунок 1.2.10).

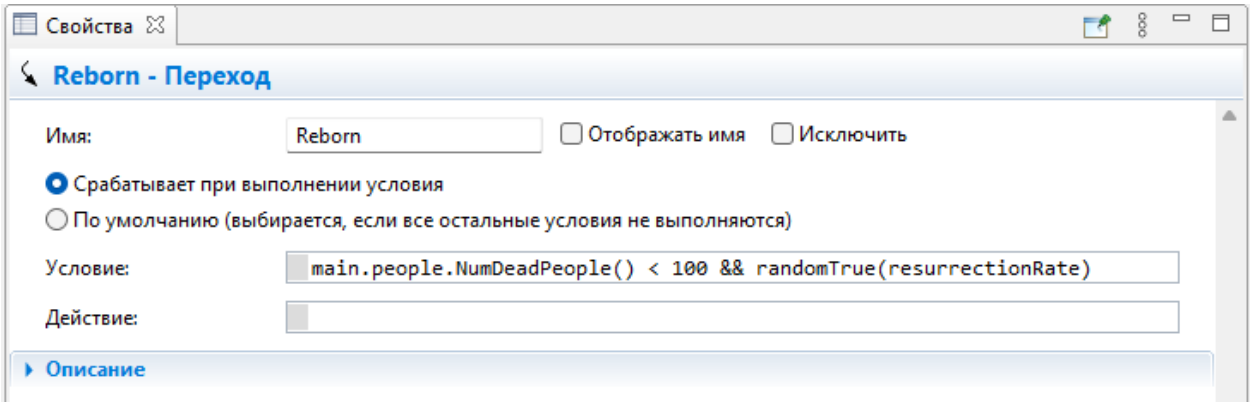

Рисунок 1.2.10 – Переход «Reborn»

Для визуализации динамики системы была добавлена временная диаграмма с накоплением (Рисунок 1.2.11).

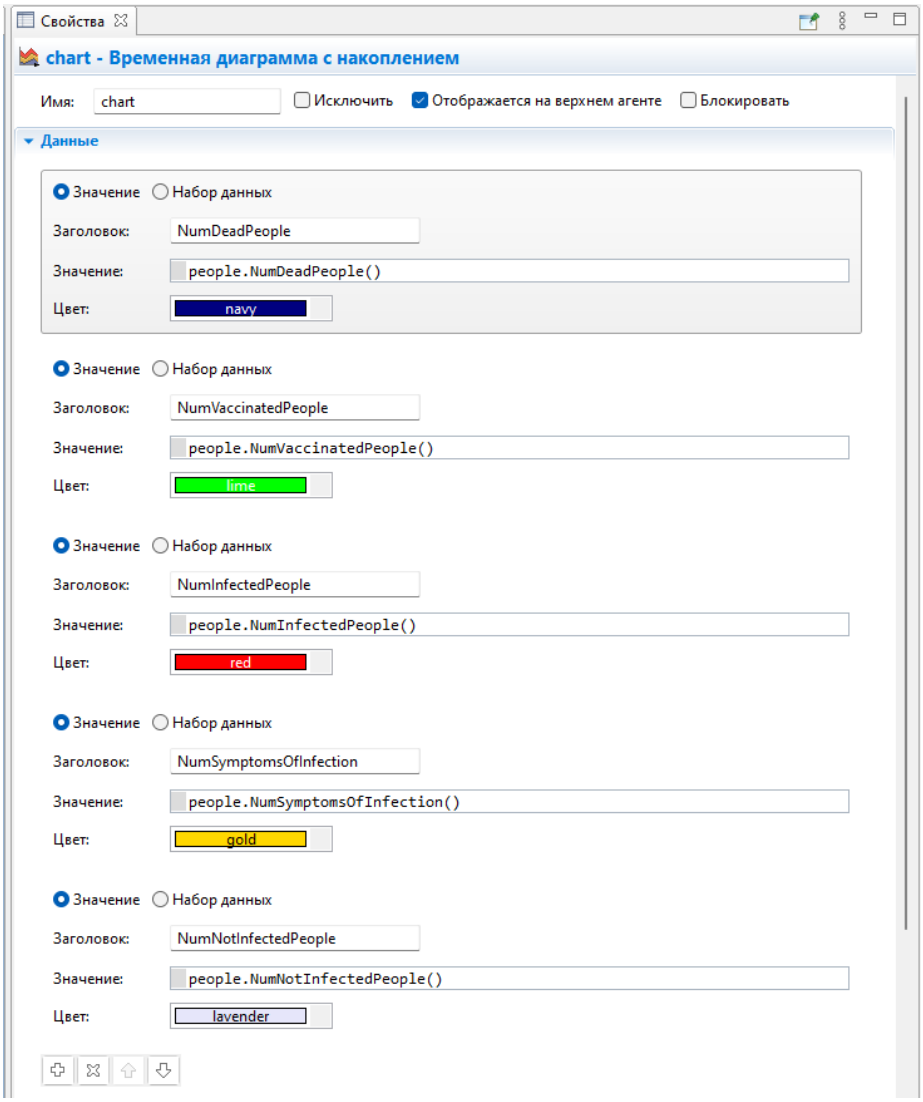

Рисунок 1.2.11 – Временная диаграмма с накоплением

Работа модели представлена на Рисунке 1.2.12.

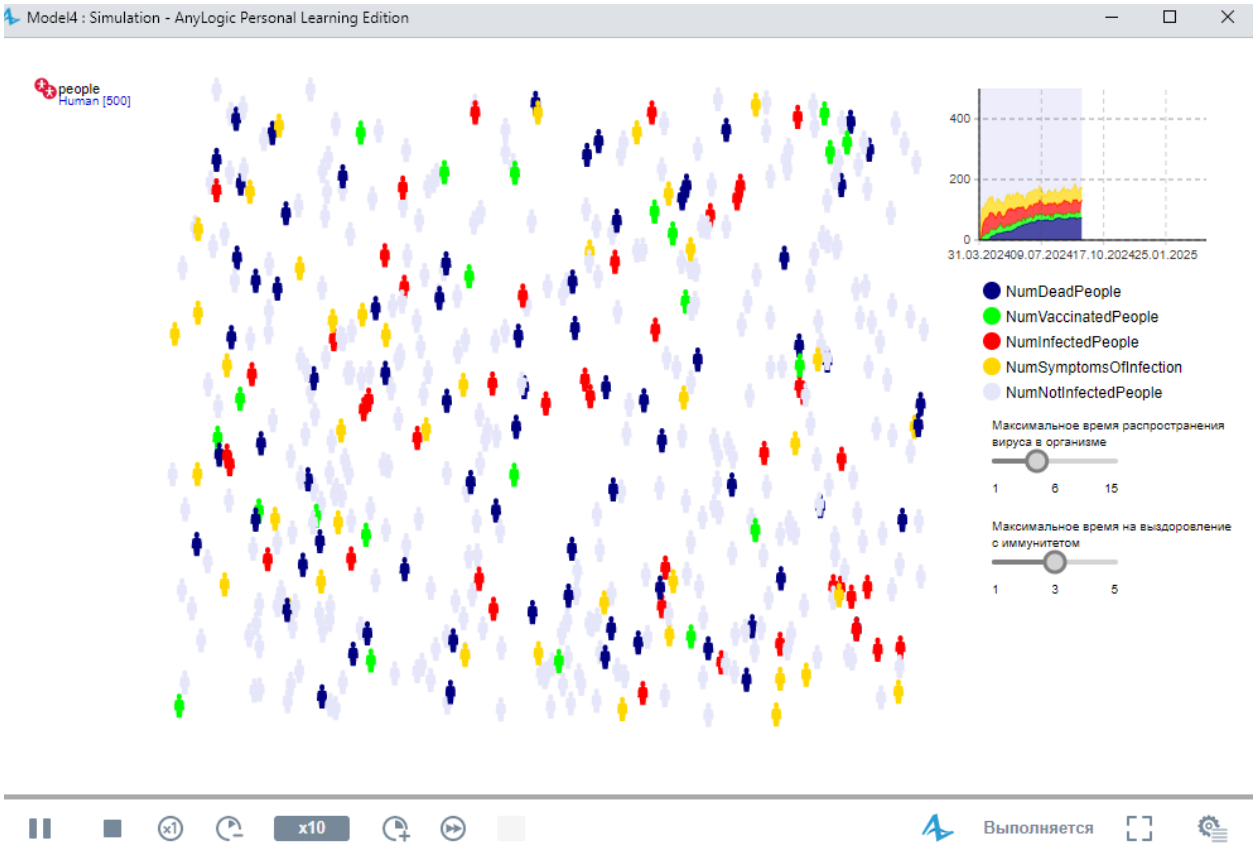

Рисунок 1.2.12 – Симуляция проекта

Изменим динамические параметры и убедимся, что интенсивность распространения вируса понизится (Рисунок 1.2.13).

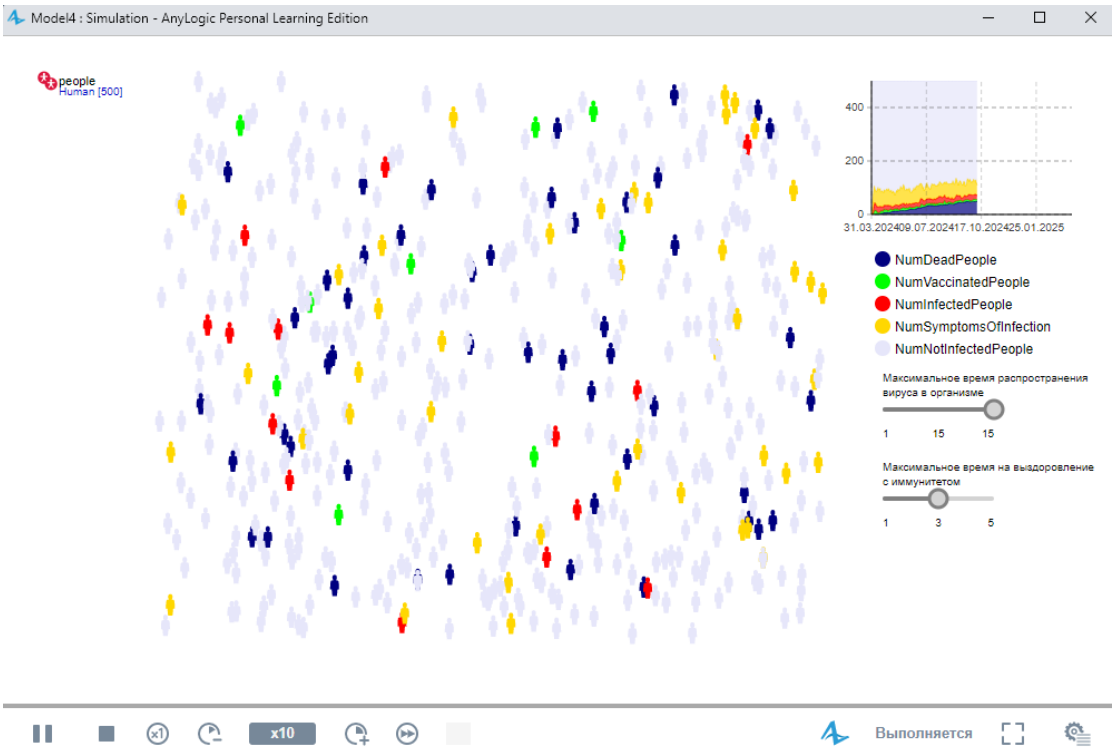

Рисунок 1.2.13 - Симуляция проекта с изменением динамических параметров

2 ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ

2.1 Постановка задачи

Необходимо промоделировать работу аэропорта. Учесть возможность покупки билетов в автоматах. Добавить стойки регистрации на рейсы, пункты досмотра багажа. Предполагается, что пассажиры могут лететь бизнес или эконом классом. В ближайший час осуществляется регистрация и посадка на два рейса: международный и внутри страны.

Требования к модели:

- Количество состояний агента – не имеет значения;
- Количество параметров агента – не имеет значения;
- Размер популяции агентов – не менее 10;
- Возможность динамического изменения параметров – не менее 1;
- Наличие условий в карте агента: не менее 1;
- Наличие картинки и 3D окна, для отслеживания динамики изменения состояний популяции.

2.2 Описание этапов выполнения работы

Изображение модели представлено на Рисунке 2.2.1.

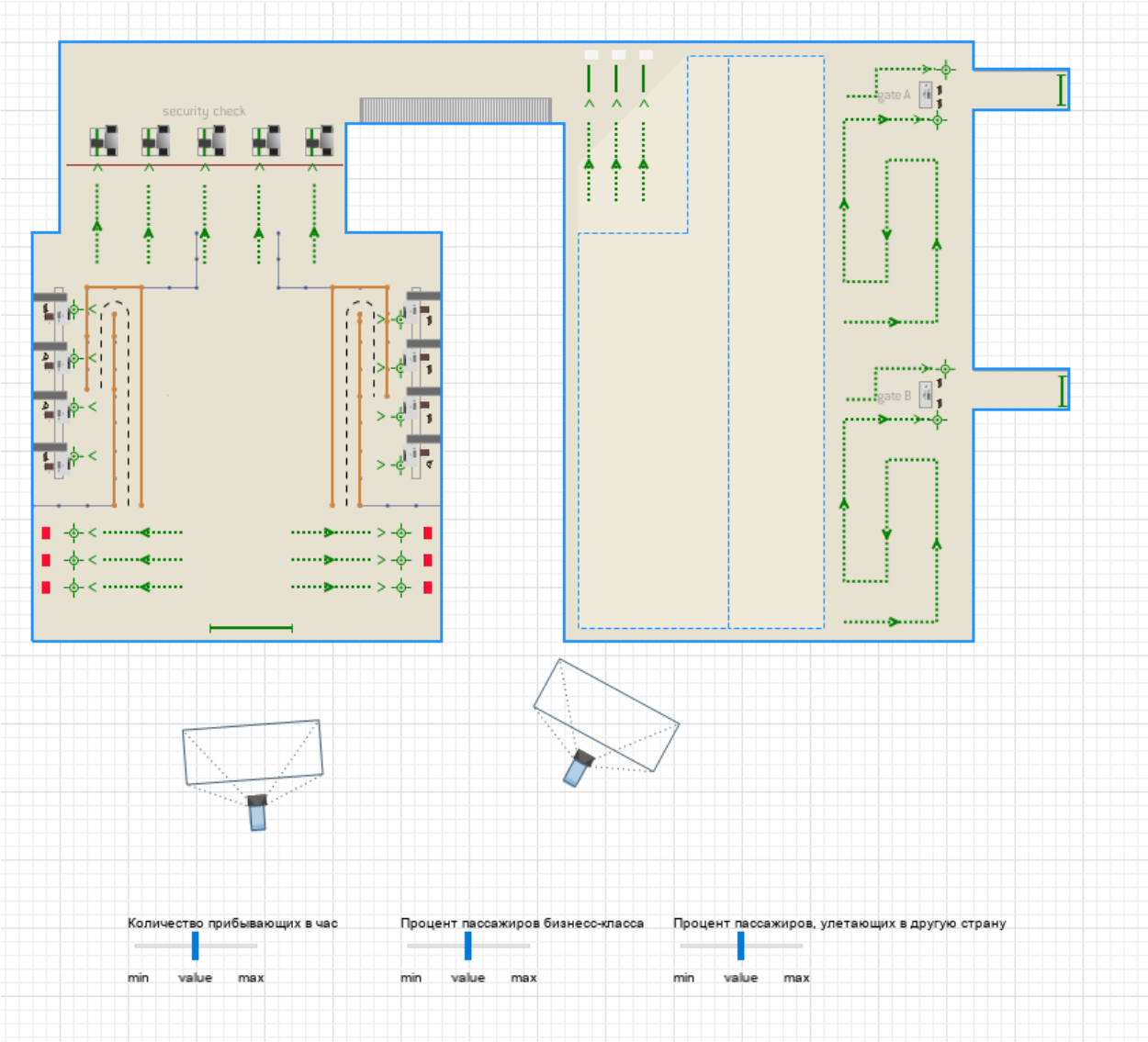

Рисунок 2.2.1 – Модель аэропорта

Диаграмма переходов представлена на Рисунке 2.2.2.

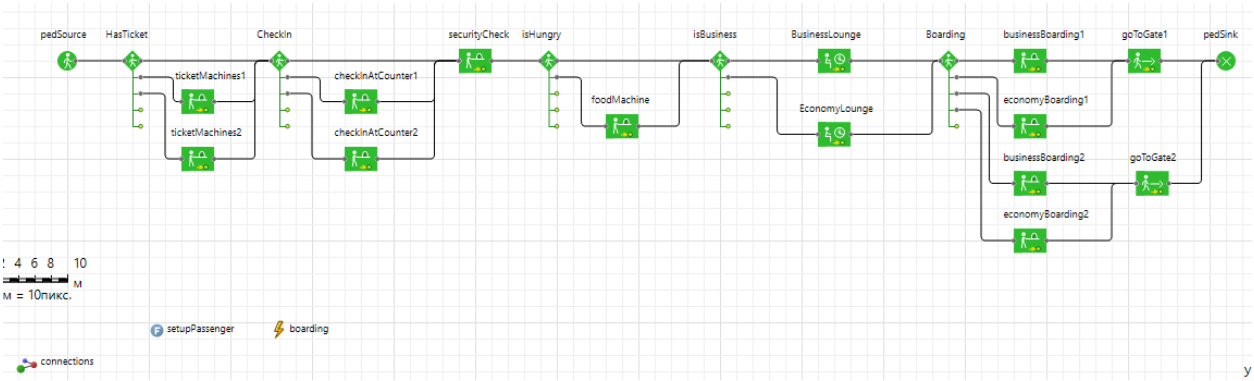

Рисунок 2.2.2 – Диаграмма переходов

Были заданы следующие динамические параметры:

- airportCongestion – загруженность аэропорта. Определяется как количество человек, заходящих в аэропорт в час;

- businessPercentage – процент бизнес-пассажиров;
- outsideTheCountry – процент пассажиров, улетающих в другую страну.

Предполагается, что у 70% процентов пассажиров уже куплен билет и они сразу проходят на стойку регистрации. Остальные 30% покупают билеты в торговых автоматах. Также, 20% пассажиров уже зарегистрированы на рейс, они проходят досмотра багажа и направляются в сторону зала ожидания. Перед входом в зал ожидания стоят торговые автоматы с едой. Зал ожидания разделяется на бизнес-зал и эконом-зал для соответствующего класса пассажира. После объявления на посадку пассажиры встают в очередь на посадку.

Запустим модель (Рисунки 2.2.3 – 2.2.6).

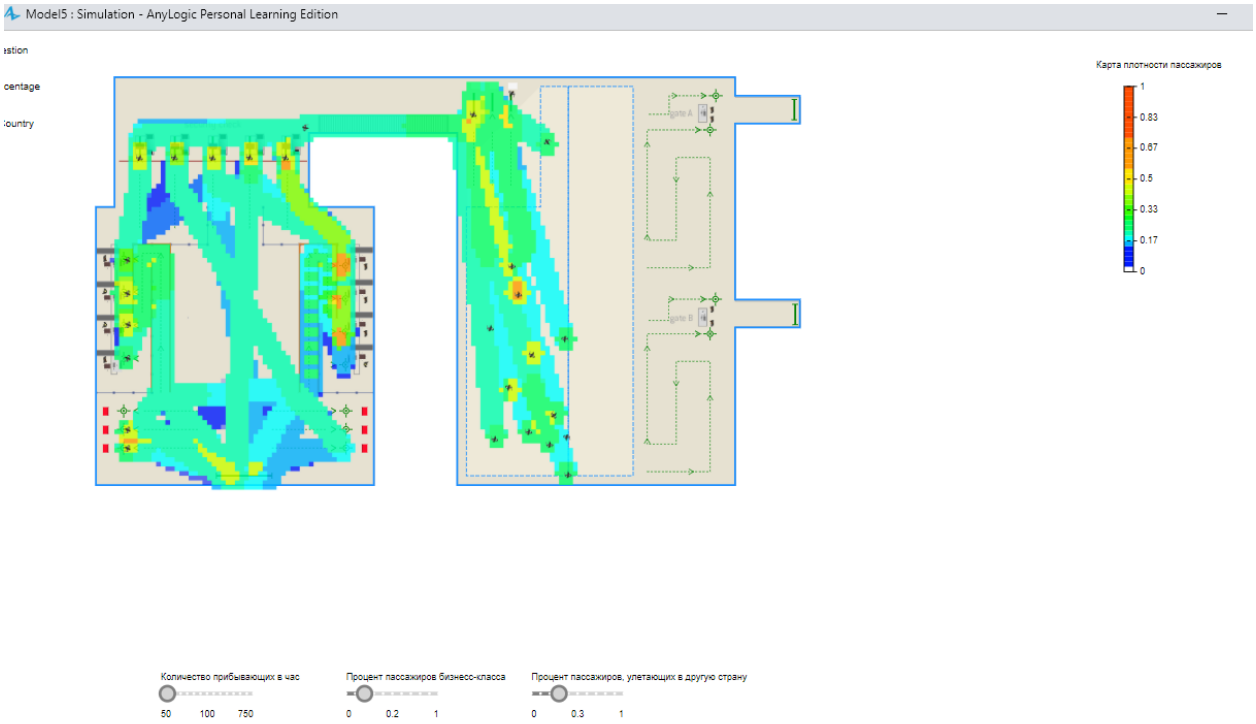

Рисунок 2.2.3 – Запуск модели

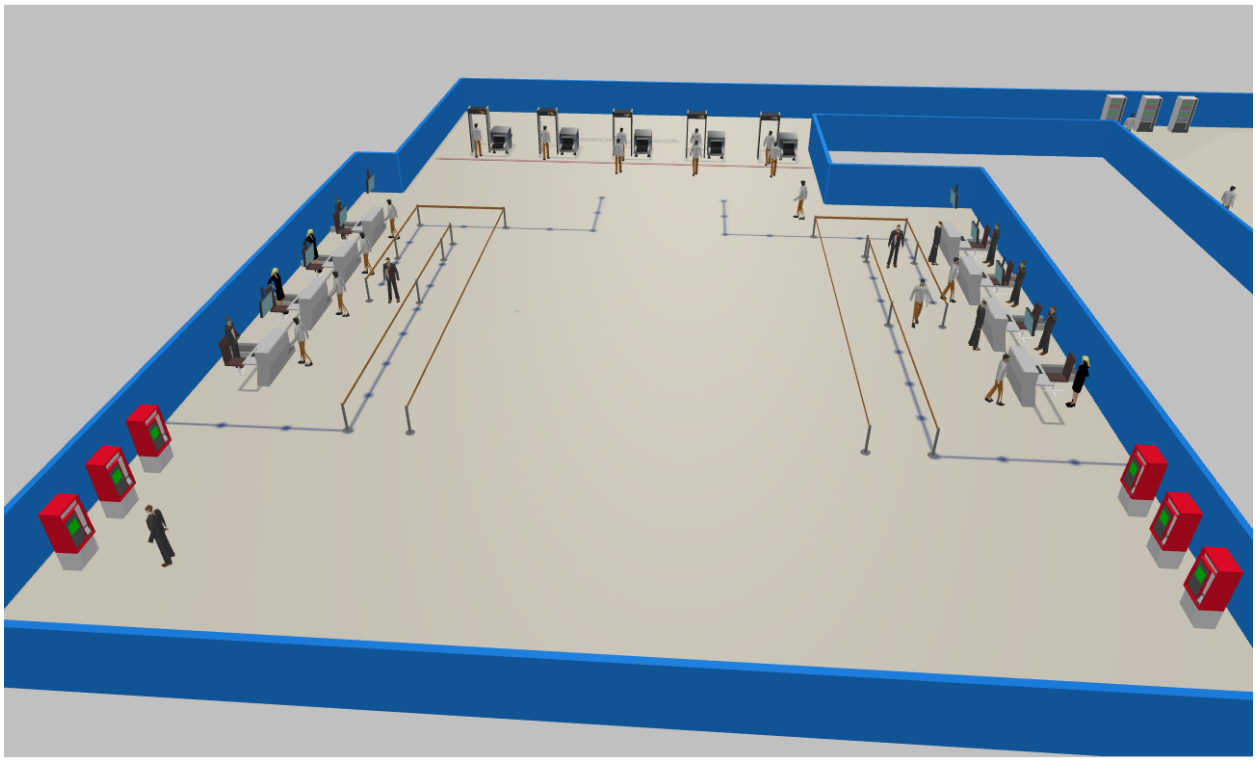

Рисунок 2.2.4 – Камера с первого зала

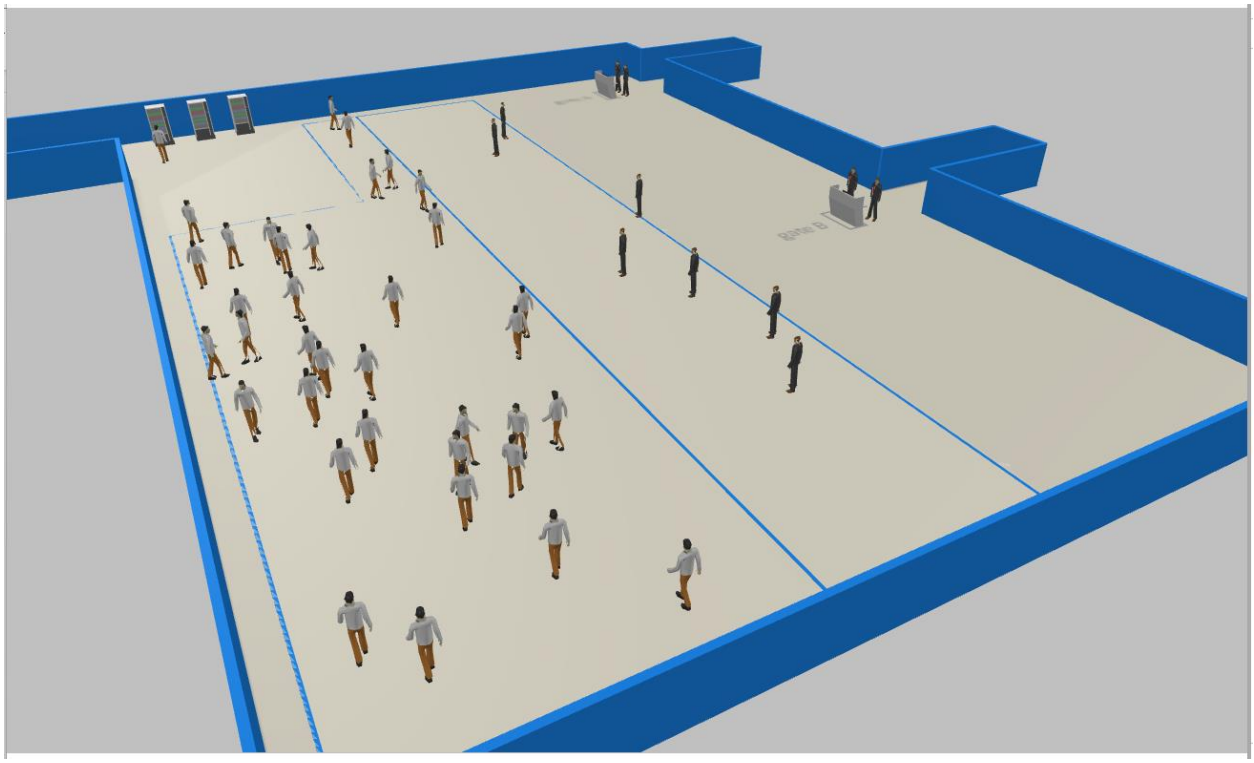

Рисунок 2.2.5 – Камера с зала ожидания

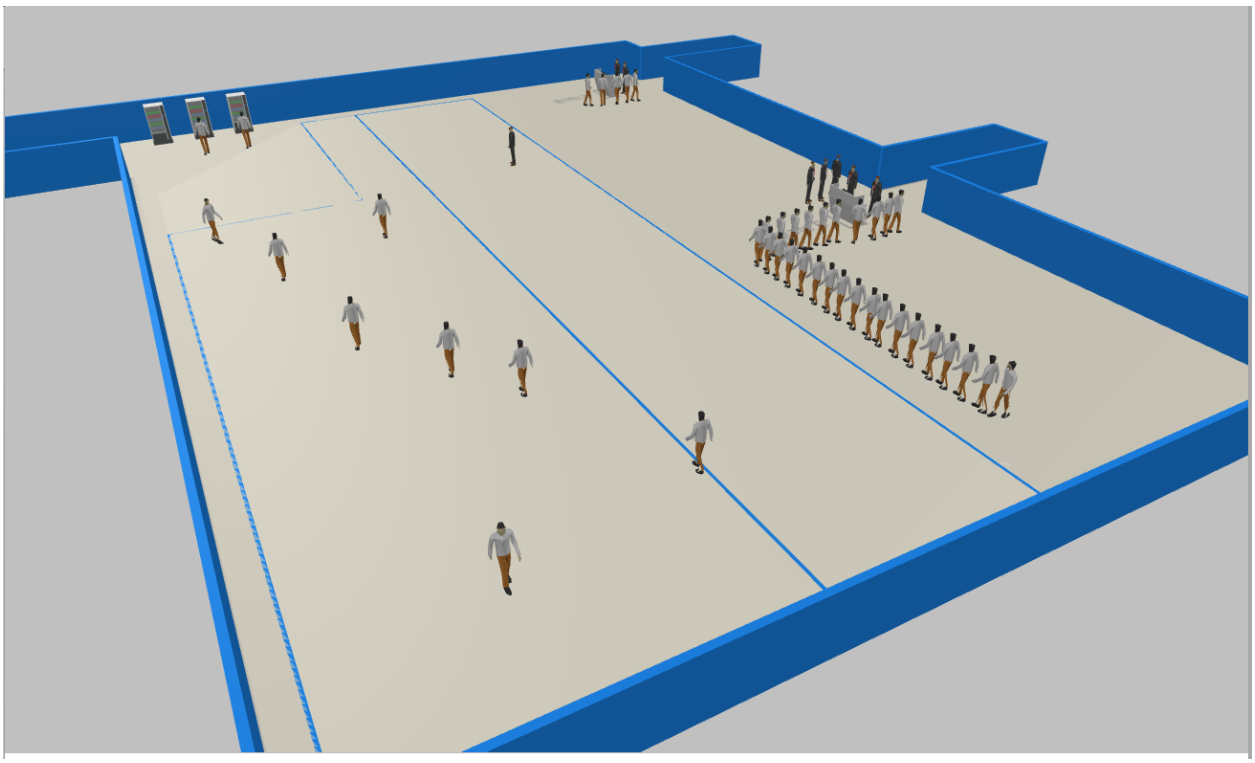

Рисунок 2.2.6 – Объявление на посадку

Как можно заметить, после объявления начала посадки, пассажиры покидают зал ожидания и встают в очереди к нужному выходу. Для моделирования объявления посадки было добавлено событие «boarding» (Рисунок 2.2.7).

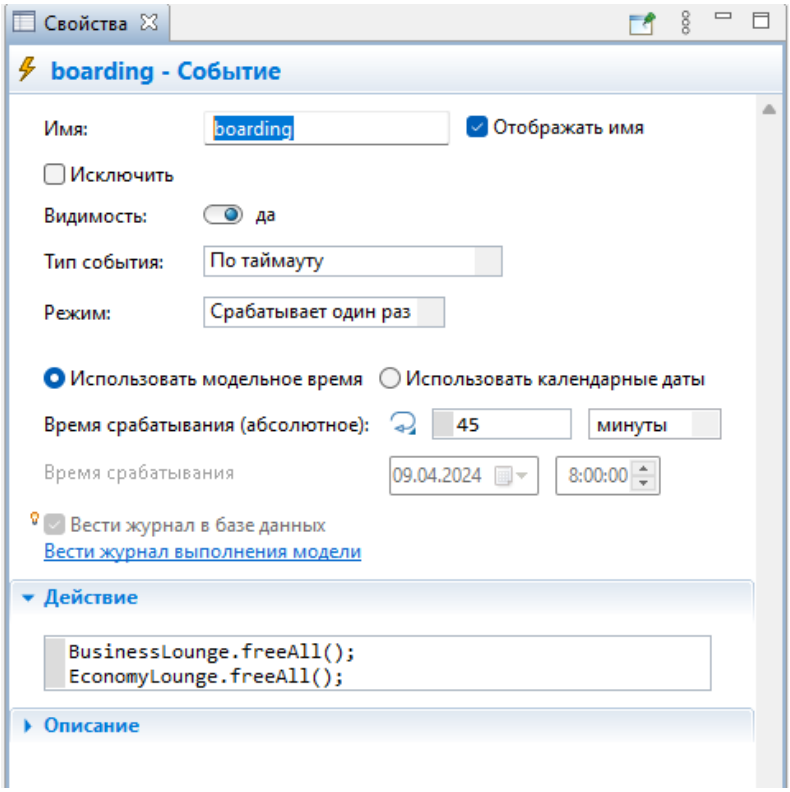

Рисунок 2.2.7 – Событие «boarding»

Изменим значение динамического параметра, отвечающего за процент бизнес-пассажиров, и увидим, что большинство моделей на камерах – люди в костюмах (Рисунок 2.2.8).

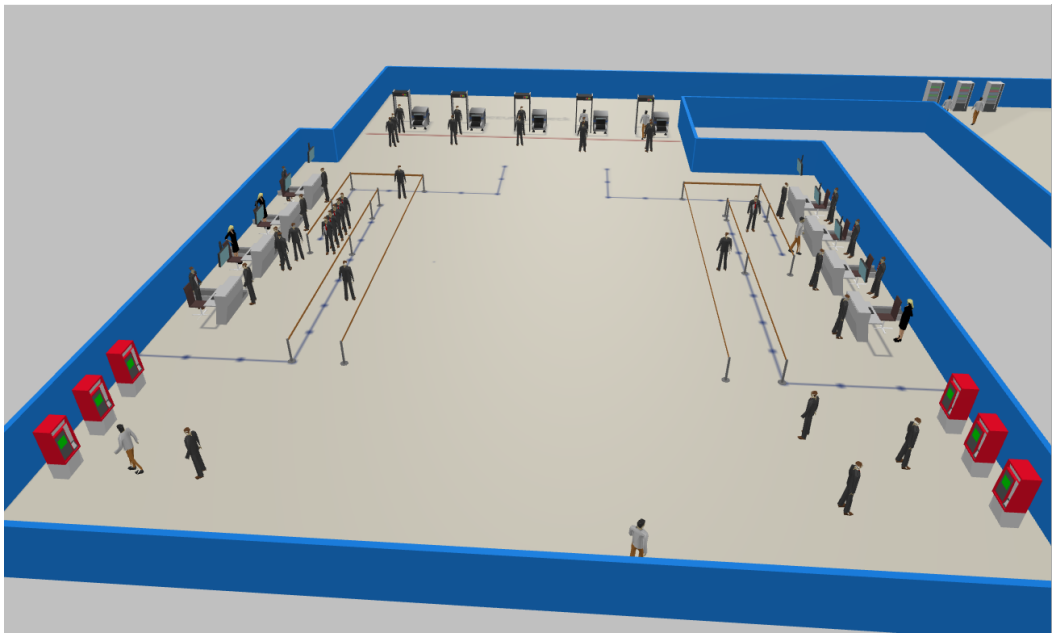

Рисунок 2.2.8 – Запуск модели с изменением динамического параметра

3 СИСТЕМНАЯ ДИНАМИКА

3.1 Постановка задачи

Разработать модель, которая будет описывать распространение на рынке нового товара, учитывать взаимодействие между потенциальными и постоянными клиентами, а также их взаимодействие с маркетинговыми стратегиями, отображать переход из клиентов в постоянных клиентов компании.

Требования к модели:

- Количество накопителей – не менее 4;
- Количество потоков - не менее 3;
- Количество параметров агента – не менее 1;
- Возможность динамического изменения параметров – не менее 1;
- Наличие обратных связей – потоков (циклов);
- Наличие графика/ов, для отслеживания динамики изменения состояний популяции.

3.2 Описание этапов выполнения работы

Были созданы следующие накопители:

- Пользователи;
- Потенциальные клиенты;
- Клиенты;
- Постоянные клиенты.

Добавлены потоки:

- Реклама;
- Продажи;
- Эксплуатация.

Добавлены следующие параметры, влияющие на потоки:

- Спрос рынка;
- Частота показа рекламы;
- Целевая аудитория;
- Бюджет на маркетинг;
- Качество рекламы;
- Сезонность;
- Эффективность рекламы;
- Продажи от рекламы;
- Доля рынка;
- Разовая покупка;
- Частота контактов;
- Продажи от устной рекламы;
- Желание рекомендовать;
- Эффективность акций;
- Срок службы;
- Уровень сервиса;
- Количество продукции;
- Качество продукции;
- Время поставки.

Была построена следующая модель системной динамики (Рисунок 3.2.1).

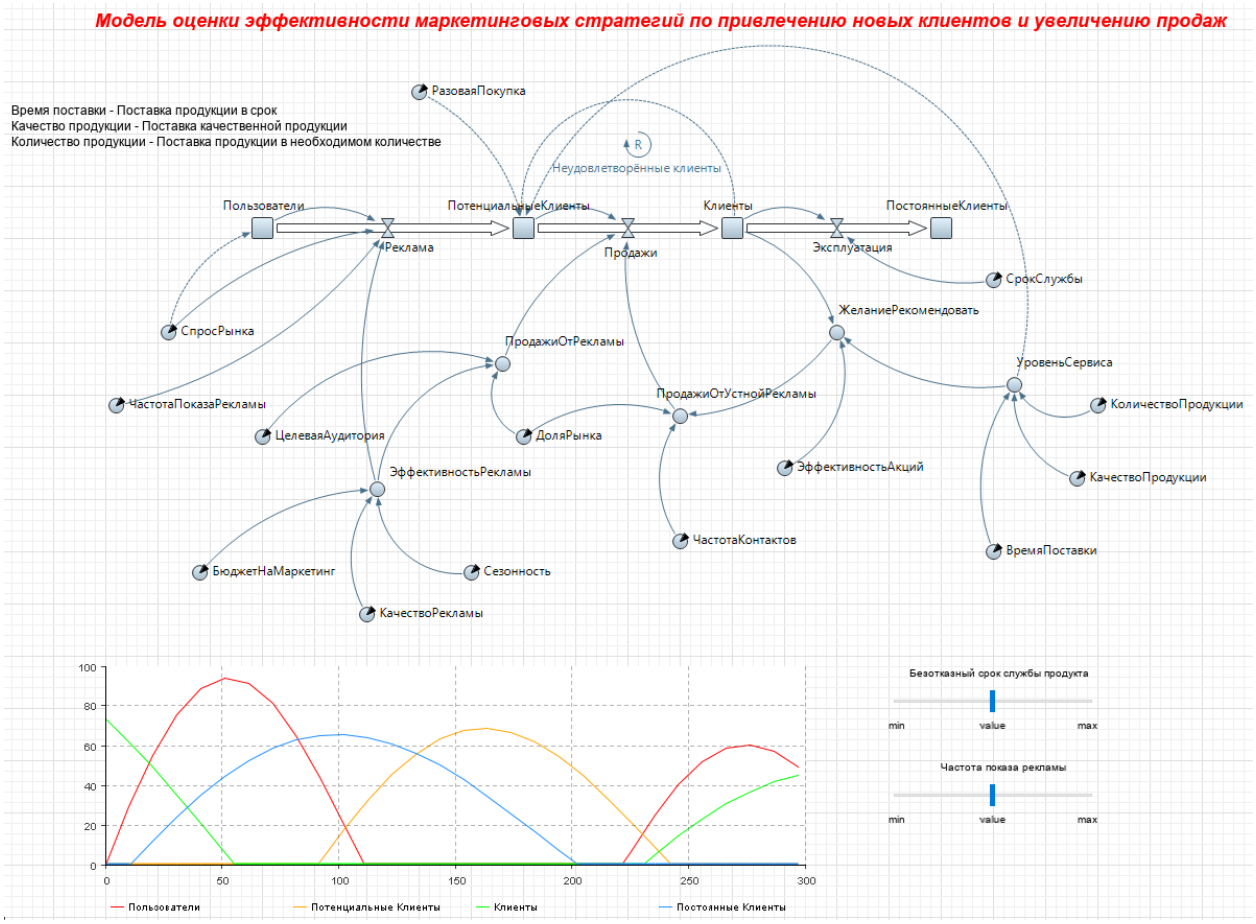

Рисунок 3.2.1 – Модель системной динамики

В роли динамических параметров выступают срок службы, который отвечает за время, после которого клиент становится постоянным клиентов, частота показа рекламы, определяющая интенсивность роста потенциальных, следовательно обычных и постоянных клиентов.

Запустим модель (Рисунок 3.2.2).

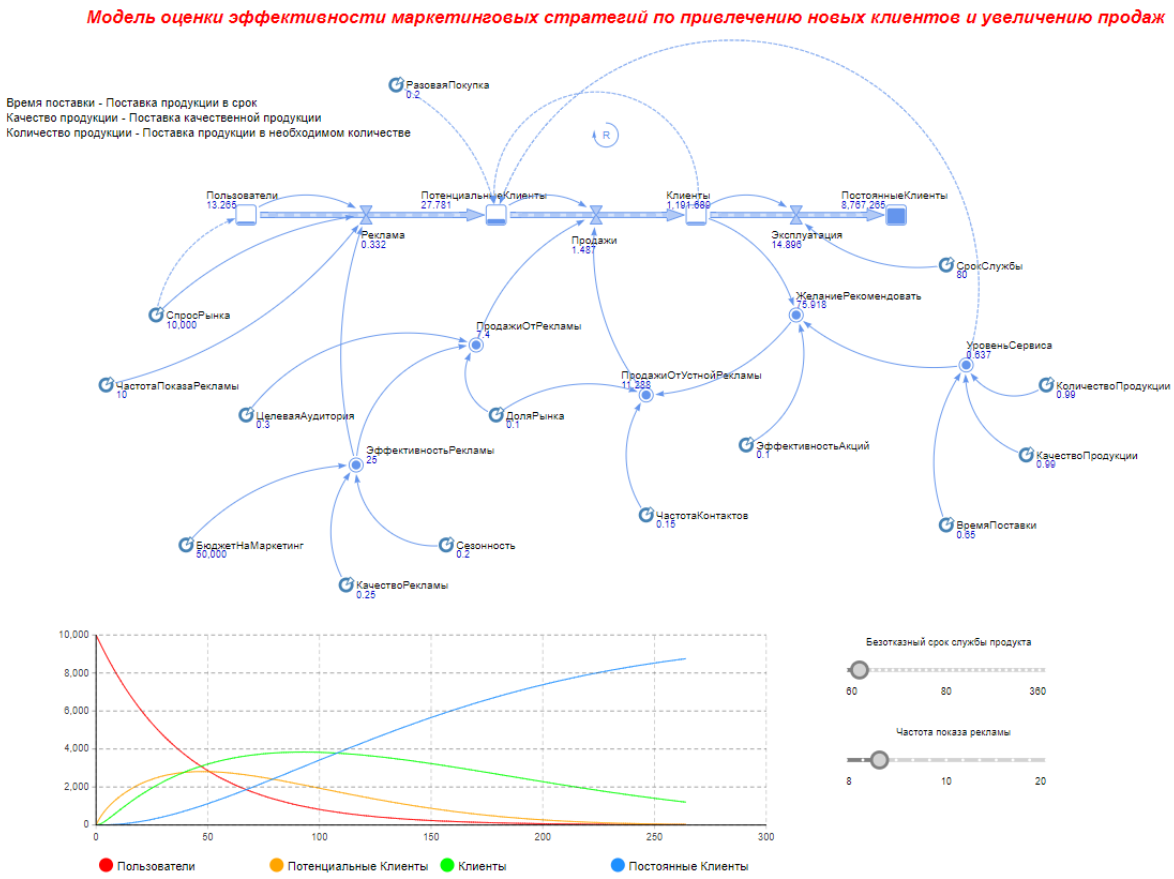

Рисунок 3.2.2 – Запуск модели

Заметим, что примерно спустя 40 дней рекламной кампании количество клиентов превышает количество потенциальных клиентов, и в будущем растет количество постоянных пользователей продукта. Теперь запустим проект, изменив значения динамических параметров (Рисунок 3.2.3).

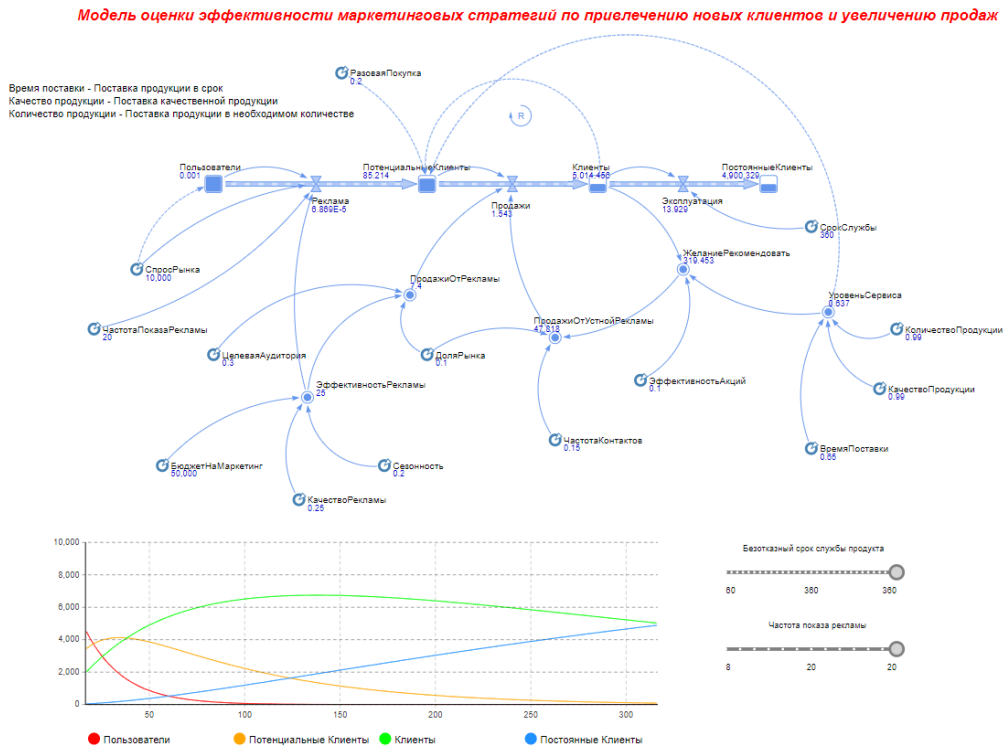

Рисунок 3.2.3 – Запуск проекта с изменением значений динамических параметров

Из графика видно, что с увеличением частоты, интенсивность роста клиентов значительно увеличивается, а с увеличением необходимого срока безотказной работы количество постоянных клиентов увеличивается медленнее, чем в первом запуске.

4 ЗАДАЧА НА ПРОГРАММИРОВАНИЕ

4.1 Постановка задачи

Необходимо создать популяцию агентов и показать ее в среде обитания. Каждый агент является потенциальным покупателем, определенный процент агентов в течение дня становится покупателем (пользователем продукта) и это видно в среде и на графике. Каждый пользователь продукта после покупки продукта может оповестить одного потенциального покупателя, что увеличит процент покупок продукта в день. Продукт портится через несколько дней, и пользователь продукта через определенное время становится вновь потенциальным покупателем, и это отражается в среде обитания и на графике.

4.2 Описание этапов выполнения работы

Для программной реализации был выбран язык Python. Будем отображать окно, в левой части которого находится поле, где будут размещены агенты, а в правой части – график, отражающий количество потенциальных клиентов и пользователей товара. Были заданы следующие константы, отвечающие за параметры модели:

NUM_AGENTS = 100 # Количество агентов

PROB_PURCHASE = 0.08 # Вероятность покупки продукта в день

PROB_REFER = 0.16 # Вероятность оповещения потенциального покупателя после покупки

DAYS_TO_SPOIL = 8 # Количество дней, через которое продукт портится

DAYS_TO_RESET = 5 # Количество дней, через которое пользователь продукта снова становится потенциальным покупателем

OBSERVATION_PERIOD = 90 # Срок наблюдения в днях

Для моделирования работы в течение заданного количества дней была

задана глобальная переменная day_count – текущий день наблюдения.

Популяция агентов хранится в списке agents, сущность одного агента – словарь, в котором хранится тип агента (потенциальный пользователь или пользователь продукта), количество дней с момента покупки (по условию продукт ломается и его владелец через время становится потенциальным пользователем). Запустим программу (Рисунок 4.2.1).

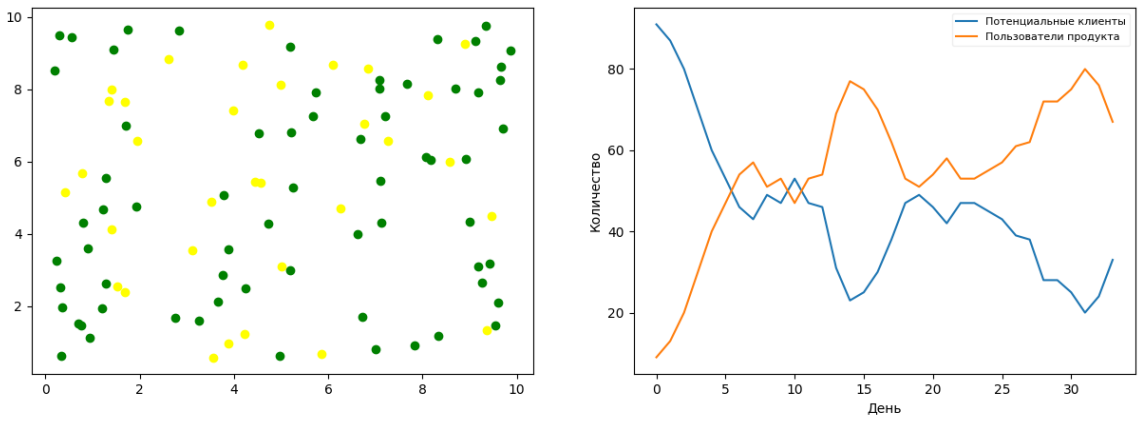

Рисунок 4.2.1 – Запуск программы

Можно заметить, что график симметричен, потому что, количество всех агентов равно количеству потенциальных клиентов вместе с количеством пользователей.

ВЫВОД

В ходе выполнения практических работ я убедился в необходимости проектирования мультиагентных систем, переоценить которую будет трудно. Мультиагентные системы могут применяться как для конструирования и моделирования гибких производственных систем, так и для управления реальными системами производства (логистика), продажи продукции различного назначения (е-коммерции), интеграции и управления знаниями и научной работы. Большое значение в мультиагентном подходе имеет социальный аспект решения современных задач как его концептуальная основа. Такие системы должны постоянно "жить" на сервере предприятия и непрерывно участвовать в решении задач, а не быть запускаемыми от случая к случаю, а для этого – обеспечивать пользователю возможность введения новых данных и компонентов. Наконец, такие системы должны накапливать информацию, извлекать из нее новые знания и в зависимости от этого изменять свое поведение с течением времени.

Я познакомился с программой AnyLogic, которая является лидирующим инструментом имитационного моделирования для бизнеса. Было изучено несколько наиболее распространённых подходов к моделированию, которые сегодня активно применяются в самых различных сферах нашей жизни.

Агентное моделирование позволяет понять поведение индивидов внутри системы, особенности их взаимодействия друг с другом и последствия, которые порождает такое взаимодействие, а также оценить влияние этих последствий на возможное поведение агентов в дальнейшем.

Дискретно-событийное моделирование — подход, соответствующий низкому и среднему уровням абстракции. Термин дискретно-событийное моделирование исторически закрепился за моделированием систем обслуживания потоков объектов некоторой природы: клиентов банка, автомобилей па заправочной станции, телефонных вызовов, пациентов в поликлиниках и т.п. Обслуживание при этом может быть достаточно сложным.

Дискретно-событийный подход широко используется в моделировании бизнес-процессов, производства, логистики, здравоохранения и т.д.

Системная динамика позволяет моделировать сложные системы на высоком уровне абстракции, не принимая в расчет мелкие детали: индивидуальные свойства отдельных продуктов, событий или людей. Такие модели позволяют получить общее представление о системе и хорошо подходят для стратегического планирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Хливнеко Л.В., Пятакович Ф.А., Практика нейросетевого моделирования, Санкт-Петербург:Лань, 2019, -200с.

2. Рыжков Ю.И. Имитационное Моделирование. Авторская имитация систем и сетей с очередями, Санкт-Петербург:Лань, 2019, -112с.

3. Мезенцев К.Н., Мультиагентное моделирование в среде NetLogo, Санкт-Петербург:Лань, 2015, -176с.

4. Тихвинский В.И., Сорокин А.Б. Имитационное моделирование: практикум [Электронный ресурс]: Учебно-методическое пособие / Тихвинский В.И., Сорокин А.Б. – М.: Московский технологический университет (МИРЭА), 2018, –157с.

5. Палей А.Г., Поллак Г.А., Имитационное моделирование. Разработка имитационных моделей средствами iWebsim и AnyLogic, СанктПетербург:Лань, 2019, – 122с.

ПРИЛОЖЕНИЯ

Приложение А — Исходный код практической работы на программирование

Приложение А

Листинг А – Код программной реализации задачи

```
import random
import matplotlib.pyplot as plt
import matplotlib.animation as animation


# Параметры симуляции
NUM_AGENTS = 100  # Количество агентов
PROB_PURCHASE = 0.08  # Вероятность покупки продукта в день
PROB_REFER = 0.16  # Вероятность оповещения потенциального покупателя после
покупки
DAYS_TO_SPOIL = 8  # Количество дней, через которое продукт портится
DAYS_TO_RESET = 5  # Количество дней, через которое пользователь продукта
снова становится потенциальным покупателем
OBSERVATION_PERIOD = 90  # Срок наблюдения в днях


# Размеры прямоугольной области
WIDTH = 10
HEIGHT = 10

# День наблюдения
day_count = 0

# Создание популяции агентов
agents = []
for i in range(NUM_AGENTS):
    agent = {
        "type": "potential",  # Тип агента: "potential" или "user"
        "days_since_purchase": 0,  # Количество дней с момента последней
покупки
    }
    agents.append(agent)

# Инициализация данных для графика
num_potential = []  # Количество потенциальных клиентов в день
num_users = []  # Количество пользователей продукта в день

# Функция обновления для анимации


def update(frame):
    # Обновление агентов

    global day_count

    for agent in agents:
        if agent["type"] == "potential":
            # Потенциальный покупатель может стать покупателем с вероятностью
PROB_PURCHASE
            if random.random() < PROB_PURCHASE:
                agent["type"] = "user"
                agent["days_since_purchase"] = 0
        elif agent["type"] == "user":
            # Пользователь продукта может оповестить потенциального
покупателя с вероятностью PROB_REFER
            if random.random() < PROB_REFER:
                for other_agent in agents:
                    if other_agent["type"] == "potential":
```

Продолжение Листинга А

```
                        other_agent["type"] = "user"
                        other_agent["days_since_purchase"] = 0
                        break
            # Продукт портится через DAYS_TO_SPOIL дней
            agent["days_since_purchase"] += 1
            if agent["days_since_purchase"] >= DAYS_TO_SPOIL:
                agent["type"] = "potential"
            # Пользователь продукта снова становится потенциальным
покупателем через DAYS_TO_RESET дней
            elif agent["days_since_purchase"] >= DAYS_TO_RESET:
                agent["type"] = "potential"

    # Подсчет количества потенциальных клиентов и пользователей в день
    num_potential.append(
        len([agent for agent in agents if agent["type"] == "potential"]))
    num_users.append(
        len([agent for agent in agents if agent["type"] == "user"]))

    # Отображение агентов в прямоугольной области
    ax1.cla()
    for agent in agents:
        if agent["type"] == "potential":
            ax1.scatter(random.random() * WIDTH,
                        random.random() * HEIGHT, c="yellow")
        elif agent["type"] == "user":
            ax1.scatter(random.random() * WIDTH,
                        random.random() * HEIGHT, c="green")

    # Отображение графика
    ax2.cla()
    ax2.plot(num_potential, label="Потенциальные клиенты")
    ax2.plot(num_users, label="Пользователи продукта")
    ax2.set_xlabel("День")
    ax2.set_ylabel("Количество")
    ax2.legend(loc='upper right', fontsize=8, framealpha=0.3)

    day_count += 1
    if day_count >= OBSERVATION_PERIOD:
        plt.close()  # Закрытие графического окна
        exit()  # Завершение программы


# Создание двух осей для области агентов и графика
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))


# Создание анимации
ani = animation.FuncAnimation(fig, update,
interval=10,save_count=OBSERVATION_PERIOD)
plt.show()
```