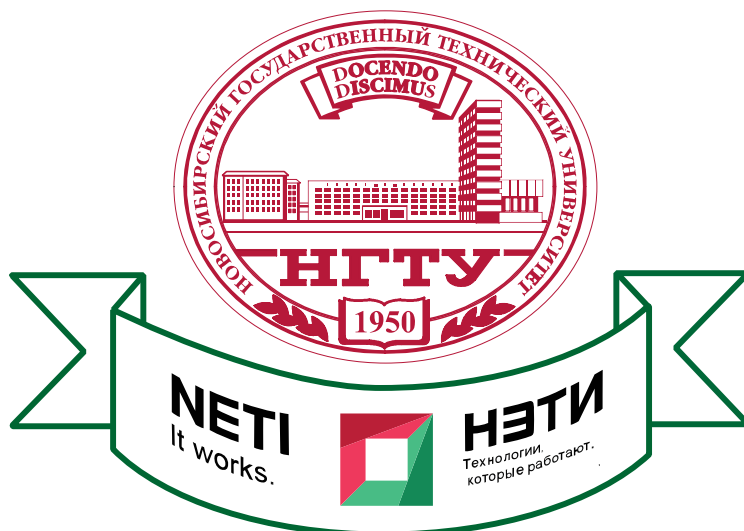


Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

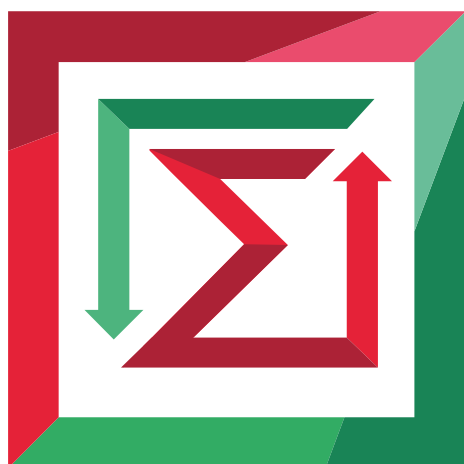
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Теоретической и прикладной информатики

Лабораторная работа № 3
по дисциплине «Компьютерное моделирование»

МОДЕЛИРОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ЦЕПЯМИ ПОСТАВОК



Факультет:	ПМИ
Группа:	ПМИ-02
Студент:	Сидоров Даниил, Дюков Богдан
Преподаватель:	Карманов Виталий Сергеевич

Новосибирск

2026

1. Описание системы

Задача реализации поставок товара заключается в том, чтобы произвести и поставить конечному потребителю некоторый продукт: фабрика производит, а другие три звена цепи поставок продвигают товар, пока он не достигает конечного потребителя в конце системы поставок (модель цепи поставок основана на известной деловой игре BeerGame [10, 11]).

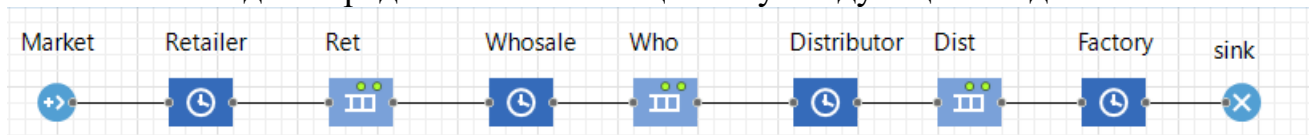
2. Цели работы

Построить и исследовать модель системы управления цепями поставок.

3. Описание выполненных действий

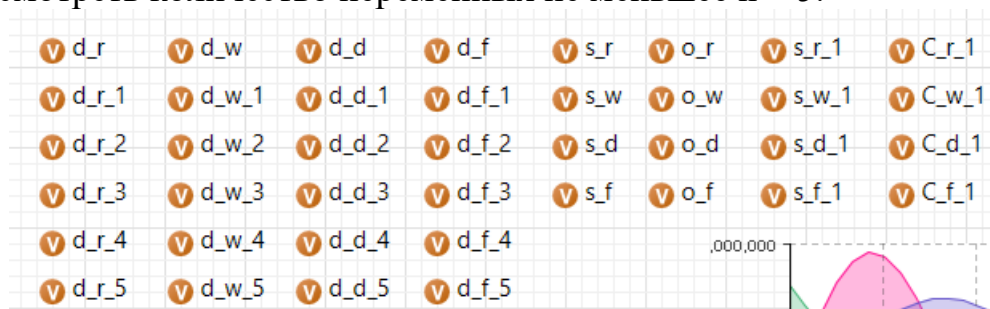
Создание диаграммы процессов

Наша модель представляет собой цепочку следующего вида:

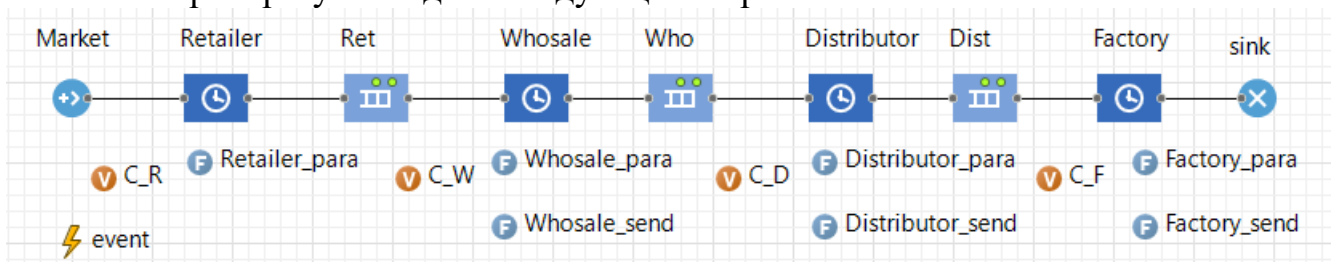


Наименование	Расшифровка
Market	Рынок
Retailer	Розничный торговец
Ret	Промежуточная очередь
Whosale	Оптовый поставщик
Who	Промежуточная очередь
Distributor	Дистрибьютор
Dist	Промежуточная очередь
Factory	Фабрика
sink	Выход

Согласно алгоритму № 2 объем заказа определяется через скользящее арифметическое среднее порядка h . Требуется для каждого участника цепи предусмотреть количество переменных не меньшее $h = 5$.



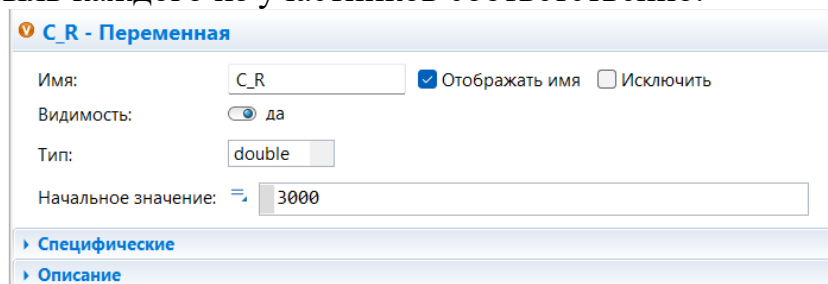
Преобразуем модель следующим образом:



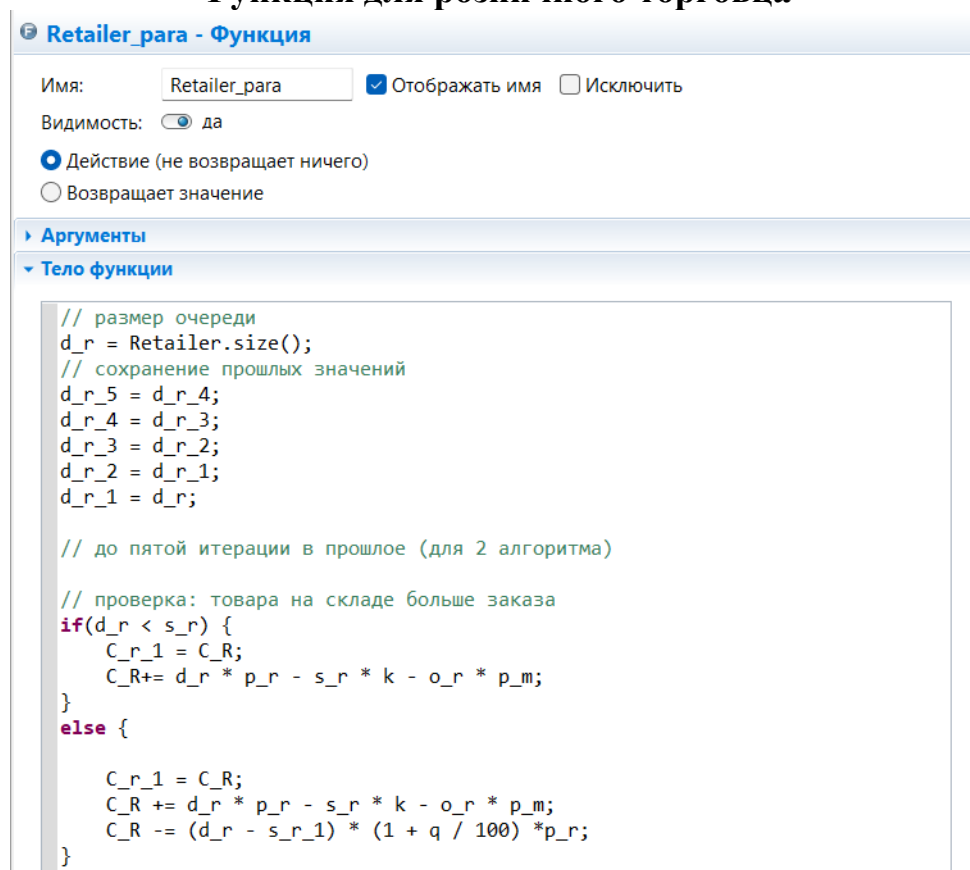
Необходимо оценить объем поставок на основе анализа истории закупок и текущего состояния запасов. В этом случае уравнение управления запасами выглядит следующим образом:

$$s_i^p = s_{i-1}^p + P_i^p - d_i^p$$

Добавим в модель переменные прибыли, учитывая, что C_R , C_W , C_D , C_F – прибыль каждого из участников соответственно.



Функция для розничного торговца



```
// размер очереди
d_r = Retailer.size();
// сохранение прошлых значений
d_r_5 = d_r_4;
d_r_4 = d_r_3;
d_r_3 = d_r_2;
d_r_2 = d_r_1;
d_r_1 = d_r;

// до пятой итерации в прошлое (для 2 алгоритма)

// проверка: товара на складе больше заказа
if(d_r < s_r) {
    C_r_1 = C_R;
    C_R += d_r * p_r - s_r * k - o_r * p_m;
}
else {
    C_r_1 = C_R;
    C_R += d_r * p_r - s_r * k - o_r * p_m;
    C_R -= (d_r - s_r_1) * (1 + q / 100) * p_r;
}
```

Данная функция проверяет товар на складе и определяет прибыль розничного торговца. Каждую единицу времени (в рассматриваемой модели – одна неделя) происходят поставки, продажи, закупки и производится выплата за хранение. Финансовый результат вычисляется по следующей формуле (на примере розничного торговца):

$$C_i^r = C_{i-1}^r + d_i^r p_i^m - s_i^r k - o_i^r p_i^r$$

Если на складе не хватает товара, то штраф за перекупку определяется по формуле:

$$C_i^r = C_{i-1}^r - (d_i^r - s_{i-1}^r) \left(1 + \frac{q}{100}\right) p_i^r$$

Задача оптимизации цепи имеет вид (для N периодов взаимодействия цепи поставок):

$$\sum_{i=1}^N (C_i^p + C_i^d + C_i^w + C_i^r) \rightarrow \max$$

Функция для оптового поставщика

Whosale_para - Функция

Имя:
☒ Отображать имя
☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)
☐ Возвращает значение

Аргументы

Тело функции

```

//размер заказа дистрибьютору
d_w = Whosale.size();
// сохранение прошлых значений
d_w_5 = d_w_4;
d_w_4 = d_w_3;
d_w_3 = d_w_2;
d_w_2 = d_w_1;
d_w_1 = d_w;

// до пятой итерации в прошлое (для 2 алгоритма)

// проверка: товара на складе больше заказа
if(d_w < s_w) {
    C_w_1 = C_W;
    C_W += d_w * p_w - s_w * k - o_w * p_m;
}
else {
    C_w_1 = C_W;
    C_W += d_w * p_w - s_w * k - o_w * p_m;
    C_W -= (d_w - s_w_1) * (1 + q / 100) * p_w;
}

```

Эта функция проверяет товар на складе и высчитывает прибыль оптового поставщика.

Whosale_send - Функция

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)
 ☐ Возвращает значение

Аргументы

Тело функции

```

if(mod == 0) {
    // h = 3
    s_r_1 = s_r;
    s_r -= d_r;
    o_r = ((d_r - s_r) + (d_r_1 - s_r) +
    (d_r_2 - s_r) + (d_r_3 - s_r)) / 3;

    if(o_r < 0) {
        o_r = 0;
    }
    s_r += o_r;
}
else {
    // h = 5
    s_r_1 = s_r;
    s_r -= d_r;
    o_r = ((d_r - s_r) + (d_r_1 - s_r) + (d_r_2 - s_r) +
    (d_r_3 - s_r) + (d_r_4 - s_r) + (d_r_5 - s_r)) / 5;

    if(o_r < 0) {
        o_r = 0;
    }
    s_r += o_r;
}

```

Данная функция реализует алгоритм № 2. Объем заказа определяется через скользящее арифметическое среднее порядка h .

Функция для дистрибьютора

Distributor_para - Функция

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)
 ☐ Возвращает значение

Аргументы

Тело функции

```

//размер заказа дистрибьютору
d_d = Distributor.size();
// сохранение прошлых значений
d_d_5 = d_d_4;
d_d_4 = d_d_3;
d_d_3 = d_d_2;
d_d_2 = d_d_1;
d_d_1 = d_d;

// до пятой итерации в прошлое (для 2 алгоритма)

// проверка: товара на складе больше заказа
if(d_d < s_d) {
    C_D_1 = C_D;
    C_D += d_d * p_d - s_d * k - o_d * p_m;
}
else {
    C_D_1 = C_D;
    C_D += d_d * p_d - s_d * k - o_d * p_m;
    C_D -= (d_d - s_d_1) * (1 + q / 100) * p_d;
}

```

Эта функция проверяет товар на складе и высчитывает прибыль дистрибьютора.

Distributor_send - Функция

Имя:
☒ Отображать имя
☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)
☐ Возвращает значение

Аргументы

Тело функции

```

if(mod == 0) {
    // h = 3
    s_w_1 = s_w;
    s_w -= d_w;
    o_w = ((d_w - s_w) + (d_w_1 - s_w) +
(d_w_2 - s_w) + (d_w_3 - s_w)) / 3;

    if(o_w < 0) {
        o_w = 0;
    }
    s_w += o_w;
}
else {
    // h = 5
    s_w_1 = s_w;
    s_w -= d_w;
    o_w = ((d_w - s_w) + (d_w_1 - s_w) + (d_w_2 - s_w) +
(d_w_3 - s_w) + (d_w_4 - s_w) + (d_w_5 - s_w)) / 5;

    if(o_w < 0) {
        o_w = 0;
    }
    s_w += o_w;
}

```

Данная функция реализует алгоритм № 2. Объем заказа определяется через скользящее арифметическое среднее порядка h .

Функция для фабрики

Factory_para - Функция

Аргументы

Тело функции

```
// размер очереди
d_f = Factory.size();
// сохранение прошлых значений
d_f_5 = d_f_4;
d_f_4 = d_f_3;
d_f_3 = d_f_2;
d_f_2 = d_f_1;
d_f_1 = d_f;

// до пятой итерации в прошлое (для 2 алгоритма)

if(mod == 0) {
    // h = 3
    s_f_1 = s_f;
    s_f -= d_f;
    o_f = ((d_f - s_f) + (d_f_1 - s_f) +
           (d_f_2 - s_f) + (d_f_3 - s_f)) / 3;
    if(o_f < 0) {
        o_f = 0;
    }
    s_f += o_f;
}
else {
    // h = 5
    s_f -= d_f;
    o_f = ((d_f - s_f) + (d_f_1 - s_f) + (d_f_2 - s_f) +
           (d_f_3 - s_f) + (d_f_4 - s_f) + (d_f_5 - s_f)) / 5;
    if(o_f < 0) {
        o_f = 0;
    }
    s_f += o_f;
}
if(d_f < s_f) {
    C_f_1 = C_F;
    C_F += d_f * p_f - s_f * k - o_f * p_m;
}
else {
    C_f_1 = C_F;
    C_F += d_f * p_f - s_f * k - o_f * p_m;
    C_F -= (d_f - s_f_1) * (1 + q / 100) * p_f;
}
```

В этой функции также учитывается количество используемых для вычислений заказов h .

Factory_send - Функция

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)

☐ Возвращает значение

Аргументы

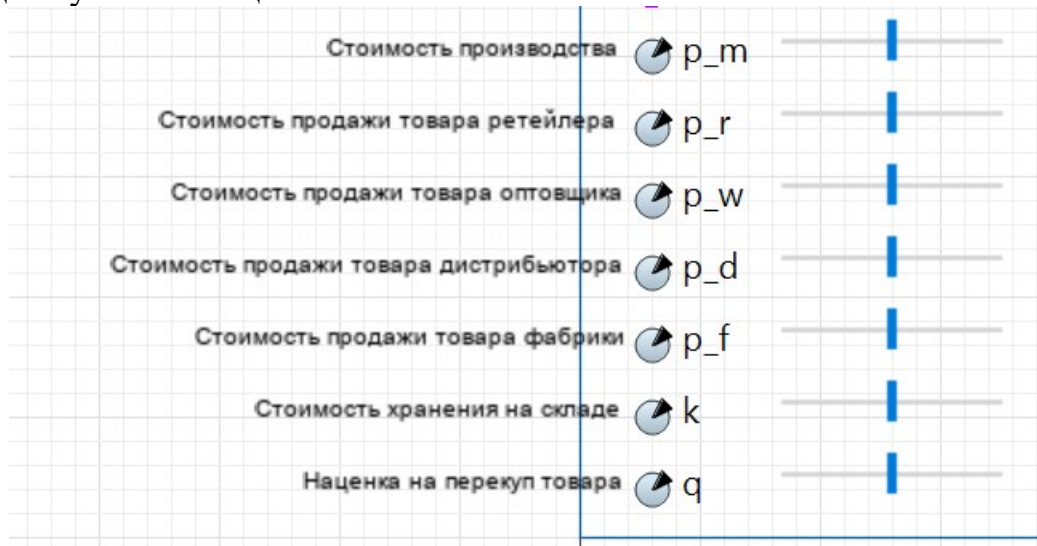
Тело функции

```
if(mod == 0) {
    s_d_1 = s_d;
    s_d -= d_d;
    o_d = ((d_d - s_d) + (d_d_1 - s_d) +
           (d_d_2 - s_d) + (d_d_3 - s_d)) / 3;
    if(o_d < 0) {
        o_d = 0;
    }
    s_d += o_d;
}
else {
    s_d_1 = s_d;
    s_d -= d_d;
    o_d = ((d_d - s_d) + (d_d_1 - s_d) + (d_d_2 - s_d) +
           (d_d_3 - s_d) + (d_d_4 - s_d) + (d_d_5 - s_d)) / 5;
    if(o_d < 0) {
        o_d = 0;
    }
    s_d += o_d;
}
```

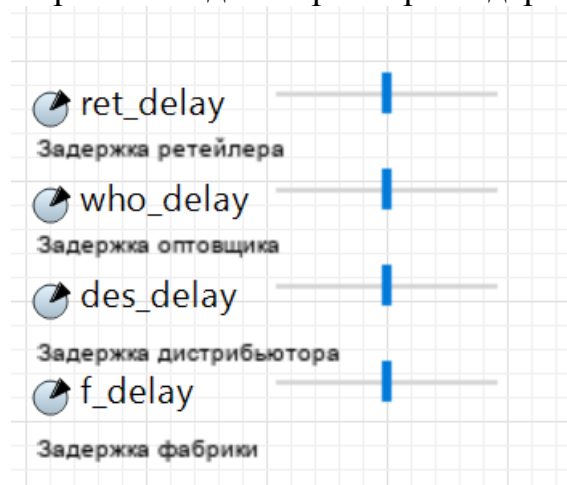
Данная функция интерпретирует алгоритм № 2. Объем заказа определяется через скользящее арифметическое среднее порядка h . Mod – переменная режима, позволяющая выбрать значение h (ноль соответствует $h = 3$; единица соответствует $h = 5$).



Установим параметры, позволяющие регулировать стоимость заказа для каждого участника цепи поставок.



Аналогичным образом введем параметры задержки.



Событие о прибытии заказа (event) изменяется согласно стандартному дискретному распределению.

event - Событие

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

Тип события:

Режим:

☒ Использовать модельное время ☐ Использовать календарные даты

Время первого срабатывания (абс.): секунды

Время срабатывания:

Период: секунды

☒ Вести журнал в базе данных
[Вести журнал выполнения модели](#)

Действие

Описание

Установим участникам цепи поставок необходимые параметры.

Market - Source

Имя: ☒ Отображать имя ☐ Исключить

Прибывают согласно:

Местоположение прибытия:



Retailer - Delay

Имя: ☒ Отображать имя ☐ Исключить

Тип задержки: ☒ Определенное время
☐ До вызова функции stopDelay()

Время задержки: секунды

Максимальная вместимость: ☒

Место агентов:  

Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☒

Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

🕒 Whosale - Delay

Имя: ☒ Отображать имя ☐ Исключить

Тип задержки: ☒ Определенное время
☐ До вызова функции stopDelay()

Время задержки:

Максимальная вместимость: ☒

Место агентов:  

▼ Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☒

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

🕒 Distributor - Delay

Имя: ☒ Отображать имя ☐ Исключить

Тип задержки: ☒ Определенное время
☐ До вызова функции stopDelay()

Время задержки:

Максимальная вместимость: ☒

Место агентов:  

▼ Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☒

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

⌚

Factory - Delay

Имя:

Factory

☒ Отображать имя
 ☐ Исключить

Тип задержки:

☒ Определенное время
☐ До вызова функции stopDelay()

Время задержки:

секунды

Максимальная вместимость:

☒

Место агентов:

▼ Специфические

Выталкивать агентов:

☐

Вернуть агента в исходную точку:

☒

Включить сбор статистики:

☐

▼ Действия

При входе:

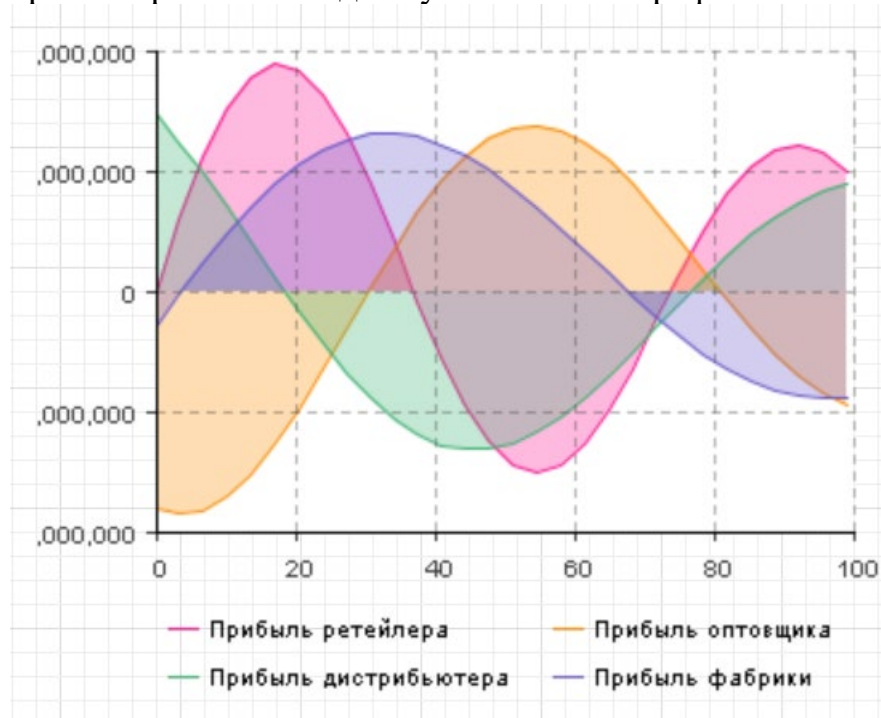
При подходе к выходу:

При выходе:

При извлечении:

Создание анимации модели

Отобразим прибыль каждого участника на графиках.




plot - Временной график

Данные

☒ Значение ☐ Набор данных

Заголовок: Прибыль ретейлера

Значение: C_R

Стиль маркера: 

Толщина линии:  1 pt

Цвет: deepPink

☒ Значение ☐ Набор данных

Заголовок: Прибыль оптовщика

Значение: C_W

Стиль маркера: 

Толщина линии:  1 pt

Цвет: darkOrange

☒ Значение ☐ Набор данных

Заголовок: Прибыль дистрибьютера

Значение: C_D

Стиль маркера: 

Толщина линии:  1 pt

Цвет: mediumSeaGr.

☒ Значение ☐ Набор данных

Заголовок: Прибыль фабрики

Значение: C_F

Стиль маркера: 

Толщина линии:  1 pt

Цвет: slateBlue



Масштаб

Временной диапазон:

100

единицы мод. времени

Вертикальная шкала:

☒ Авто
 ☐ Фиксированный

От:

0

До:

100000000

Внешний вид

Метки по оси X:

Снизу

Метки по оси Y:

Слева

Формат временной оси:

Единицы модельного времени

Цвет фона:

Нет заливки

Цвет границы:

Нет линии

Цвет меток:

darkGray

Цвет сетки:

darkGray

☒ Рисовать линию
 ☒ Заливка области под линией

Интерполяция:

☒ Линейный
 ☐ Ступенчатая

Отразим простейшей анимацией процесс моделирования. Для этого создадим тип агента, который находится в библиотеке моделирования процессов.

Расположим элементы следующим образом, свяжем участников пути и укажем нового агента Product:

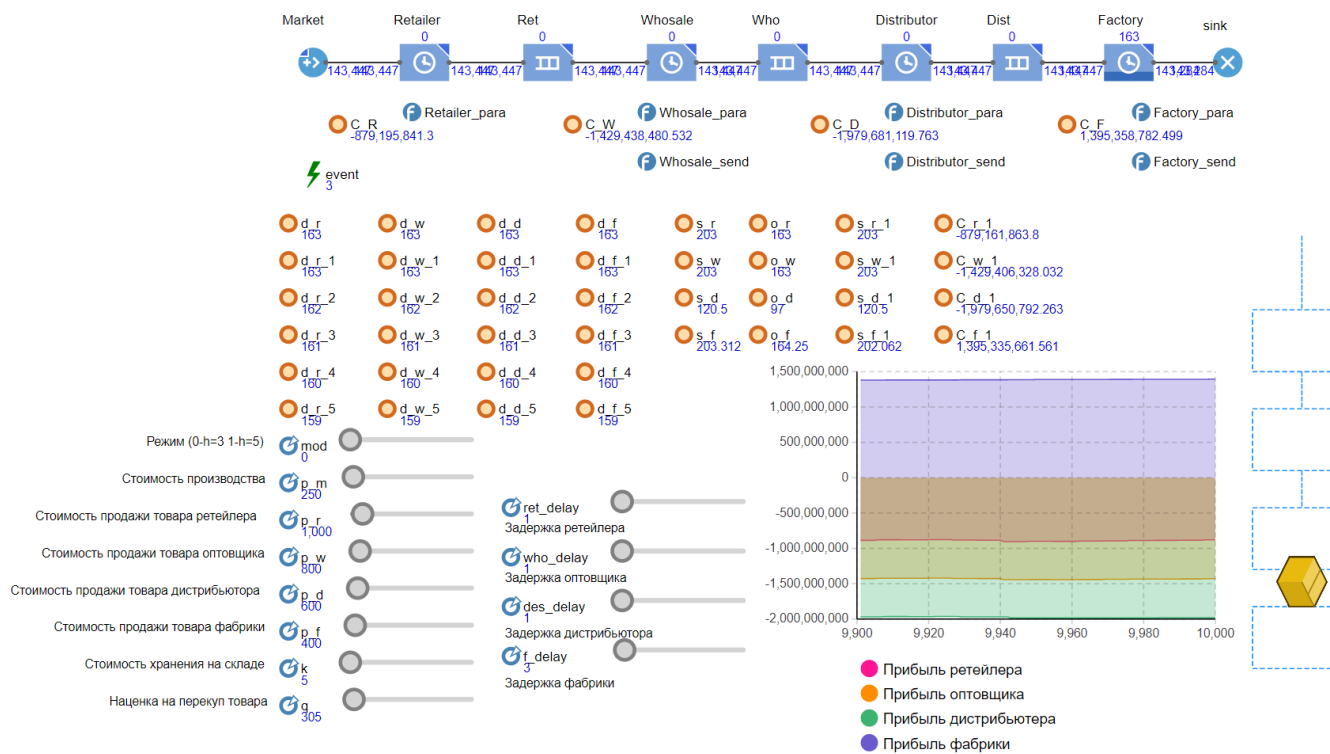
4. Полученные результаты и их анализ

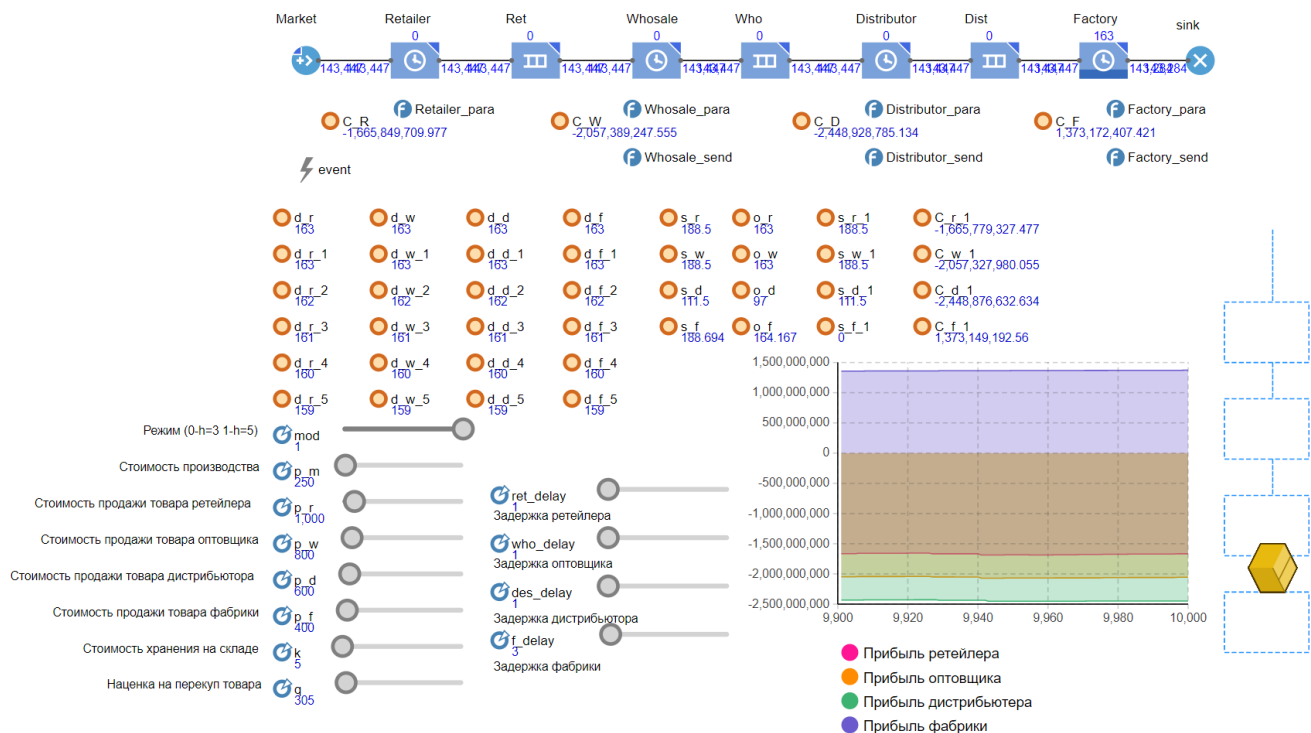
Исследуем модель при $h = 3$ и $h = 5$:

Зададим следующие параметры:

- Стоимость производства = 250 д.е.
- Стоимость продажи товара ретейлером = 1000 д.е.
- Стоимость продажи товара оптовщиком = 800 д.е.
- Стоимость продажи товара дистрибьютором = 600 д.е.
- Стоимость продажи товара фабрикой = 400 д.е.
- Стоимость хранения на складе = 250 д.е.
- Наценка на перекуп товара = 305 д.е.
- Задержка ретейлера = 1 нед.
- Задержка оптового поставщика = 1 нед.
- Задержка дистрибьютора = 1 нед.
- Задержка фабрики = 3 нед.

h	Прибыль розничного продавца, д.е.	Прибыль оптового поставщика, д.е.	Прибыль дистрибьютора, д.е.	Прибыль фабрики, д.е.
3	-879195841.3	-1429438480.532	-1979681119.763	1395358782.499
5	-1665849709.977	-2057389247.555	-2448928785.134	1373172407.421

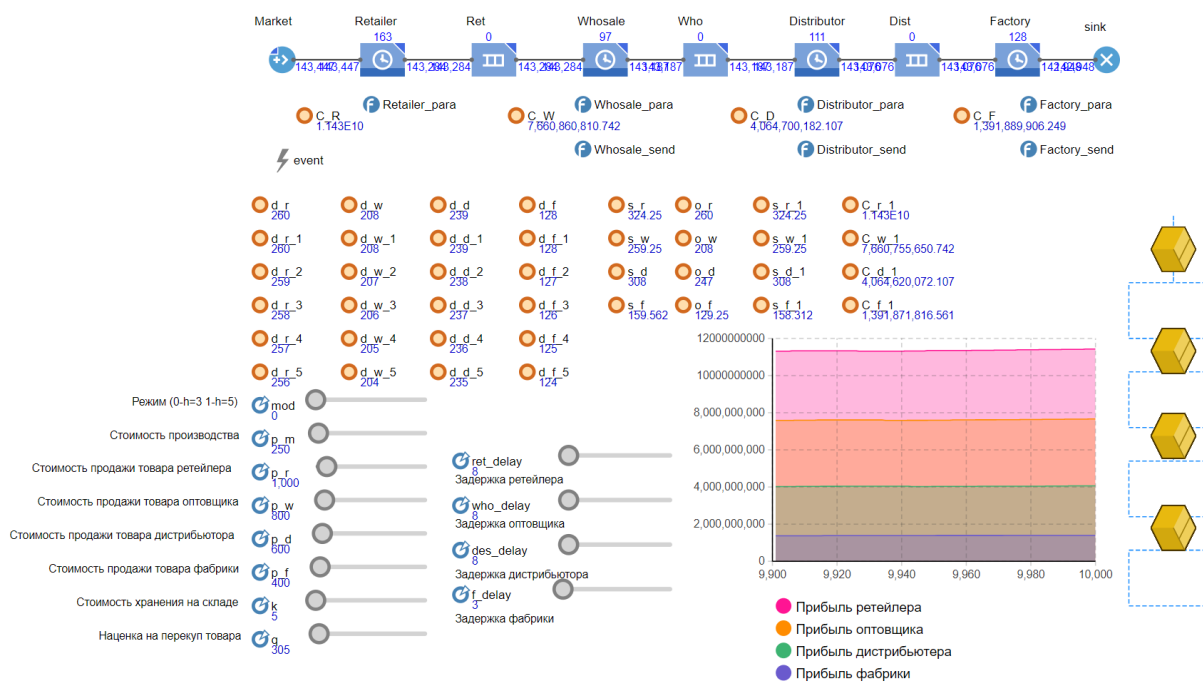


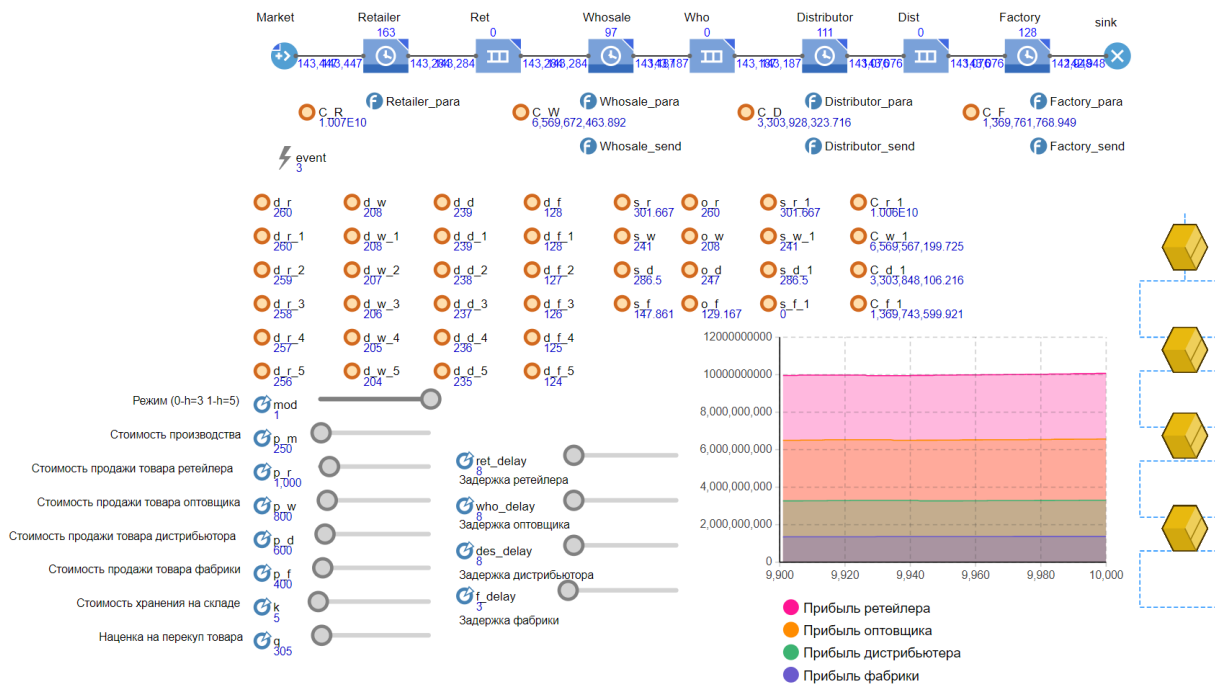


Видим, что все узлы, кроме фабрики работают в минус, изменим параметры, что изменить ситуацию:

- Задержка ретейлера = 8 нед.
- Задержка оптового поставщика = 8 нед.
- Задержка дистрибьютора = 8 нед.
- Задержка фабрики = 3 нед.

h	Прибыль розничного продавца, д.е.	Прибыль оптового поставщика, д.е.	Прибыль дистрибьютора, д.е.	Прибыль фабрики, д.е.
3	11430000000	7660860810.742	4064700182.107	1391889906.249
5	10070000000	6569672463.892	3303928323.716	1369761768.949





Видим, что при $h=3$ и $h=5$ с текущими задержками все узлы начинают работать в плюс. При $h=5$ ситуация немного хуже, так как прибыль узлов меньше.

5. Вывод

В итоге построена модель функционирования сети поставок, учитывающая характер случайного спроса и фактор задержки поставок. Определены объемы запасов в материальном и стоимостном выражении.