|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных Технологий

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**Отчёт**

по практическому занятию №6

по дисциплине «Моделирование программных систем»

Выполнил студент группы Андрусенко Л. Д.

Гришин А. В.

|  |  |
| --- | --- |
| **Проверил:** | Образцов В.М. |

Работа выполнена «\_\_» \_\_\_\_\_\_\_202\_\_ г

«Зачтено» «\_\_» \_\_\_\_\_\_\_202\_\_ г.

Москва 2024

**Задание**

**Цель работы:** получение экспериментальной модели процессора. **Постановка задачи:** Построить модель работы процессора используя сети Петри. Использовать в качестве инструмента имитационного моделирования – Anylogic 8 PLE (бесплатная версия).

**Выполнение задания**

Компьютер работает в мультипрограммном режиме обработки пакетов, т.е. одновременно начинает обработку нескольких (вплоть до установленного максимума k = 4) заданий, но не может начать обработку новых заданий, пока не будет выполнен этот пакет. В пакете каждое задание имеет собственное время выполнения, которое покидает центральный процессор (ЦП) по его истечении. В системе существует три класса приоритетов: высший приоритет имеют задания типа 3, средний – задания типа 2, низший – задания типа 1. Когда ЦП завершает выполнение последнего задания в пакете, он сначала обращается к заданиям из очереди класса 3 и берет на выполнение как можно больше заданий, вплоть до указанного максимума k. Если в очереди класса 3 было меньше k заданий, ЦП принимает как можно больше заданий из очереди класса 2, чтобы сумма заданий класса 1 и класса 2 составила максимальный размер пакета k. Если в пакете еще остается место, ЦП переходит к очереди класса 1. Процессор не может начать обработку любых поступающих заданий, пока не завершит выполнение всех заданий в текущем пакете. Установленное время обслуживания задания класса i равномерно распределено между константами ai и bi. Для каждого класса заданий действует отдельный процесс поступления, т.е. интервалы времени между поступлениями двух последовательных заданий класса i экспоненциально распределены со средним значением ri. Построим модель работы системы со следующими значениями параметров (табл. 1).

*Таблица 1*

|  |  |  |  |
| --- | --- | --- | --- |
| i | ri | ai | bi |
| 1 | 5 | 4 | 8 |
| 2 | 1.6 | 1 | 2 |
| 3 | 0.2 | 0.5 | 1 |

Изменим имя агента на Entity (заявка). Откажемся от предложения выбрать анимацию агента (позже создадим свою). Добавим агенту один параметр priority вещественного типа.

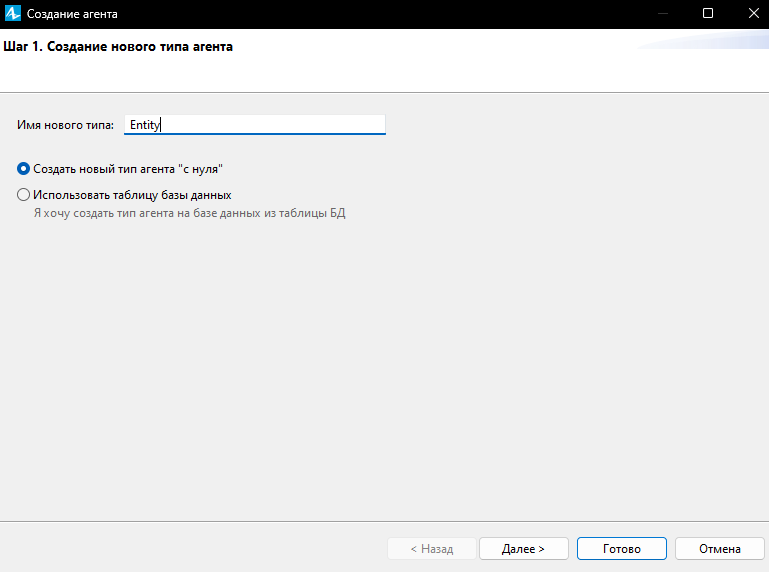


Рисунок 1 — Создание агента

Агент Entity появился в структуре проекта как класс модели. Щелкнем по нему. Раскроется закладка окна графического редактора заявки. Добавим в него элемент oval из палитры Презентация и настроим его свойства. Выберем Цвет заливки – silver, радиус – 10.

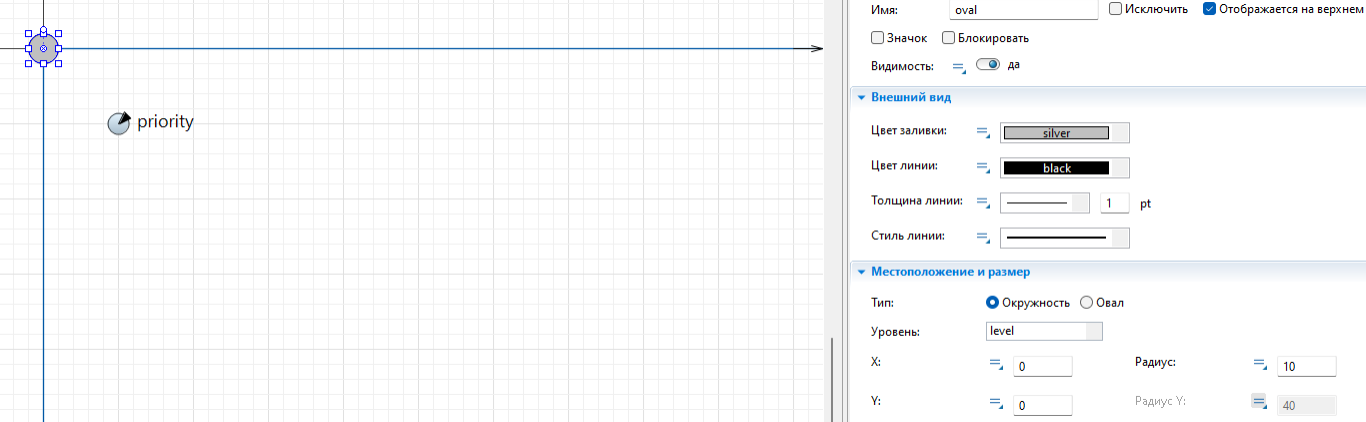
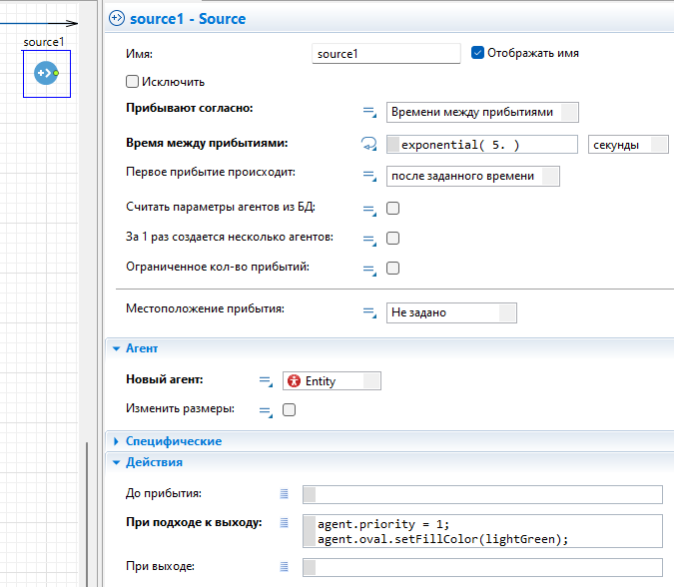


Рисунок 2 – Добавление oval

Создадим источник заявок первого типа (с наименьшим приоритетом). Для этого поместим в модель элемент source (источник). Переименуем в source1. Настроим его свойства. В качестве новой заявки будет фигурировать агент Entity. При подходе к выходу из источника заявке будет назначаться приоритет, при этом кружок-презентация заявки будет светло-зеленого цвета.



# Рисунок 3 – Настройка свойств источника заявок с низшим приоритетом

Аналогично создадим источники заявок с приоритетом 2 и 3. Назовем их, соответственно, source2и source3. В поле свойств Время между прибытиями для источника source2 запишем exponential(1.6), а для источника source3: exponential(.2). Внесем соответствующие изменения в код в поле Действия – При подходек выходу. Из source2будут выходить кружки синего цвета, а из source3 – красного.

# 

# Рисунок 4 – Настройка свойств источника заявок со средним приоритетом

# Рисунок 5 – Настройка свойств источника заявок с высшим приоритетом

# Внесем в модель три параметра целого типа. Назовем их count\_1, count\_2, count\_3. В них будут сохраняться количество заявок соответствующего типа в пакете. Для контроля общегочислазаявок в пакете (их должно быть ровно4) понадобитсяещеодин параметр – pack.

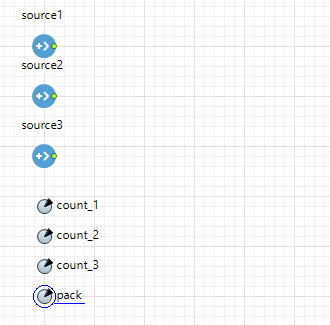


Рисунок 6 – Внесение параметров

Раскроем список элементов Разметка пространства библиотеки процессов. Перетащим в модель два элемента Прямоугольный узел. В свойствах первого узла (node) Аттракторы выберем количество и расположение точек,в которые будут помещаться презентации заявок (кружки), находящихся в очереди. Для второго узла(node1) установим Количество аттракторов 4 и сетку 1х4.

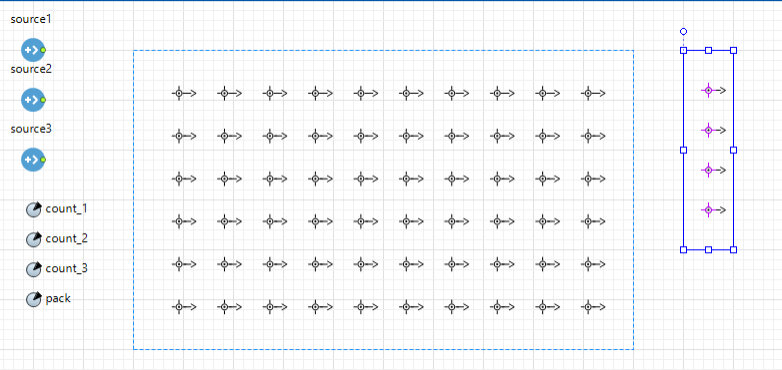


Рисунок 7 – Создание прямоугольной области

Добавим в модель элементы queue (очередь), hold (захват), delay (задержка), sink (сток). Далее последовательно соединим их. Ко входу очереди подключим источники заданий.

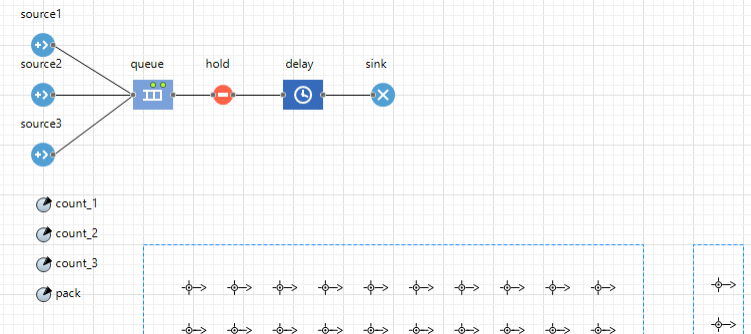


Рисунок 8 – Добавление элементов в модель

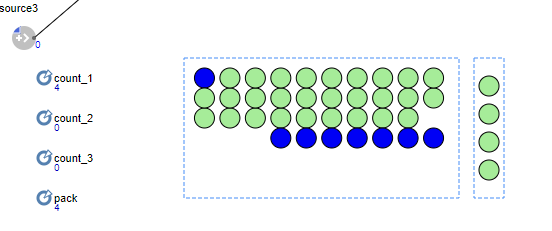


Рисунок 9 – Запуск модели

На работающей модели видно, как изменяется размер очереди. Заявки сортируются и подсчитываются по признаку приоритета, затем обслуживаются процессором и покидают его в полном соответствии с заданными значениями параметров. Для анализа динамических свойств модели организуем сбор данных о работе очереди и процессора. Для этого в свойстве элемента delay Специфические – Включить сброс статистики.

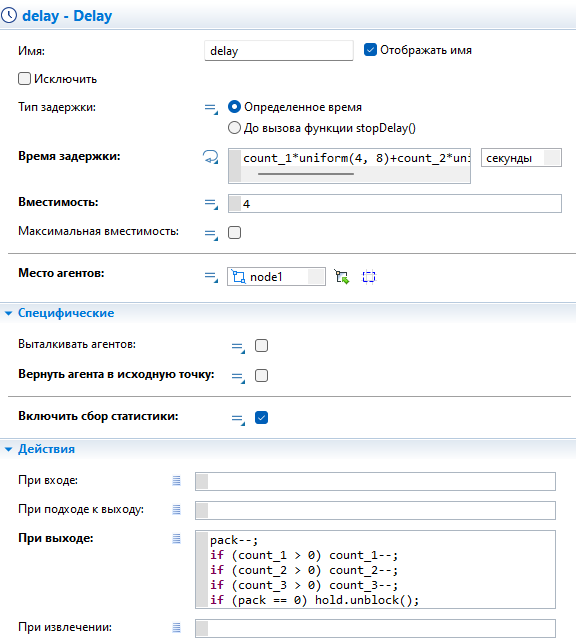


Рисунок 10 – Настройка сброса статистики

# Для определения времени, потраченного процессором на выполнение задания, понадобятся блоки TimeMeasureStart и TimeMeasureEnd. Вставим эти блоки перед hold и после delay.

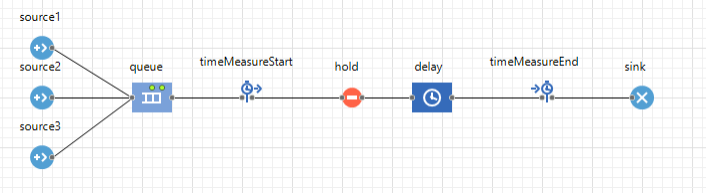


Рисунок 11 – Положение блоков TimeMeasureStart и TimeMeasureEnd

для определения времени выполнения задания

Ниже элемента node1 поместим в модуль Гистограмму из палитры Статистика. В свойствах гистограммы укажем, что на ней будут отображаться двет екущие характеристики распределения (tmEnd.distribution) времени обработки пакета.

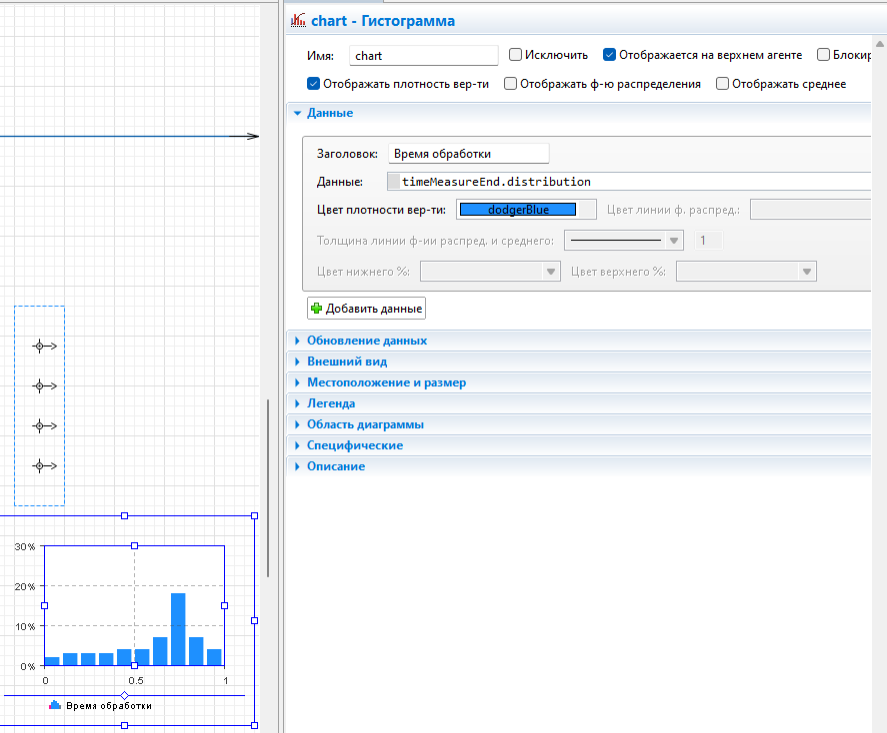


Рисунок11 – Гистограмма времениобработки пакета

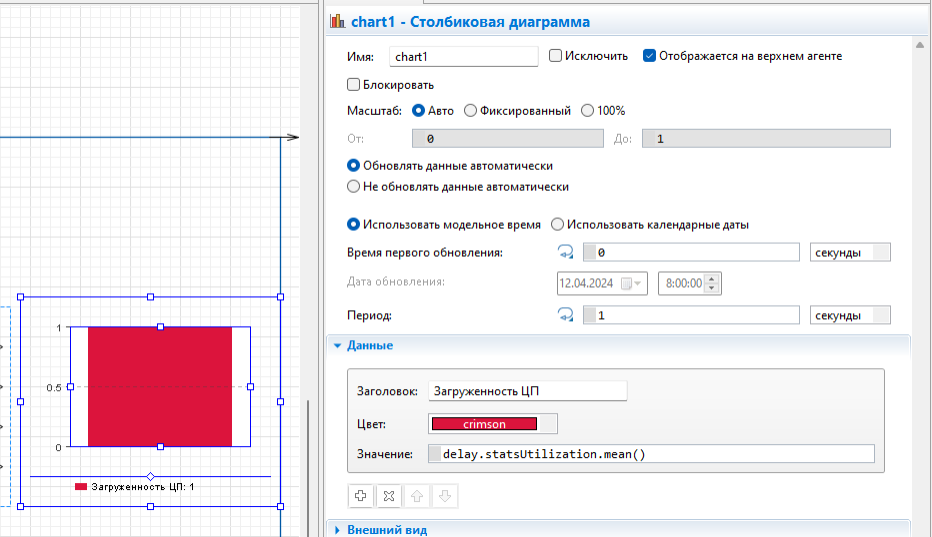
Правее элемента node1 поместим Столбиковую диаграмму. В окне свойств диаграммы отредактируем свойство Данные. Данный элемент будет показывать текущую среднюю загруженность процессора.

Рисунок 12 – Диаграмма занятости процессора

Внесем в модель еще одну столбиковую диаграмму и разметим ее под очередью. В свойствах диаграммы добавим один элемент данных.

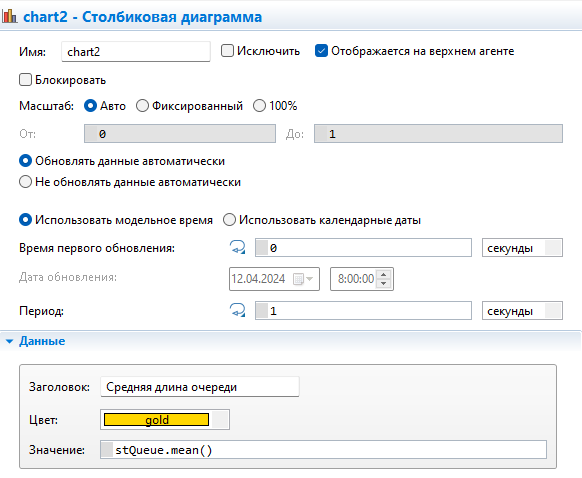


Рисунок 13 – Настройка свойств диаграммы очереди

# Запустим модель.

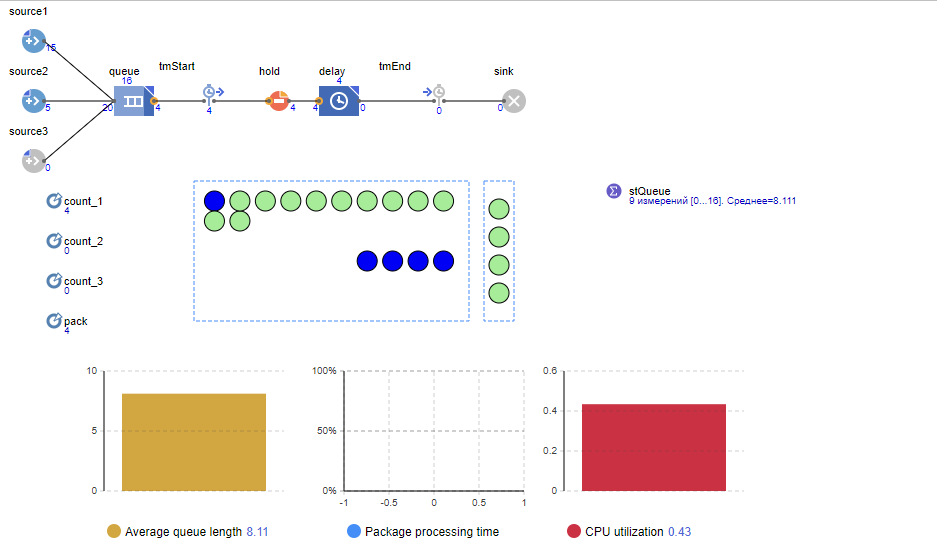


Рисунок 14 – Запуск модели

**Вывод**

В результате выполнения данной практической работы была успешно построена модель работы процессора с использованием сетей Петри в имитационной среде Anylogic 8 PLE. Модель позволяет демонстрировать процесс обработки заявок разного приоритета, анализировать динамические характеристики системы, такие как время обработки пакета, загруженность процессора и длина очереди.

Выполнение работы позволило приобрести навыки построения имитационных моделей с использованием сетей Петри, освоить основные элементы и инструменты среды Anylogic для моделирования дискретных систем, а также закрепить знания о приоритетном обслуживании заявок в мультипрограммной системе. Полученная модель может быть использована для исследования и оптимизации работы процессора в различных условиях.

# Список источников

1. Илья Григорьев . Anylogic за три дня Практическое пособие по имитационному моделированию 2017. – режим доступа: URL: [Моделирование программных систем 1/1 [II.23-24]\_54: Лекции Образцов (mirea.ru)](https://online-edu.mirea.ru/mod/folder/view.php?id=571354)