

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО
Физико-механический институт
Высшая школа прикладной математики и вычислительной физики

Отчёт по лабораторной работе №3

по дисциплине

"Автоматизация научных исследований"

Генерация диаграмм UML

Выполнил студент:
Лапина Ольга Константиновна
Группа 5040102/50201

Санкт-Петербург
2025

Содержание

1. Задание.....	3
2. Описание предметной области и промпты.....	4
2.1. Описание предметной области.....	4
2.2. Промпты.....	6
A1. Промпт 1 (минимальный).....	6
A2. Промпт 2 (базовый).....	7
A3. Промпт 3 (расширенный).....	7
A4. Промпт 4 (продвинутый).....	7
3. Результаты генераций диаграммы вариантов использования.....	10
3.1. Промпт 1.....	10
3.2. Промпт 2.....	14
3.3. Промпт 3.....	18
3.4. Промпт 4.....	21
4. Сравнительный анализ результатов.....	26
Таблица 1: Анализ результатов генераций.....	26
5. Выводы.....	27

1. Задание

Цель работы

Разработать и проанализировать диаграмму UML (а или б) для информационной системы по выбору студента.

Задание

1. Часть а. Диаграмма вариантов использования

- Выбрать предметную область для проектирования информационной системы
- Определить основных действующих лиц системы
- Выявить основные функциональные требования
- Построить диаграмму вариантов использования с учетом следующих требований:
 - Отобразить все действующие лица
 - Определить основные и вспомогательные варианты использования
 - Установить отношения между вариантами использования (include, extend)
 - Добавить границу системы
 - Обеспечить логическую целостность диаграммы

2. Часть б. Диаграмма классов

- На основе выбранной предметной области разработать диаграмму классов
- Определить основные классы системы
- Установить иерархию наследования
- Определить типы связей между классами
- Указать атрибуты классов с типами данных
- Добавить кратности ассоциаций

Требования к выполнению

- Использовать PlantUML для генерации диаграмм
- Соблюдать стандарты UML
- Обеспечить логическую целостность диаграмм
- Предоставить комментарии к ключевым элементам диаграмм
- Проверить работоспособность сгенерированного кода
- Провести исследование для 1-2 моделей ИИ

Критерии оценки

- Корректность синтаксиса PlantUML
- Соответствие стандартам UML
- Полнота отображения функциональных требований
- Качество визуализации
- Логическая целостность диаграмм
- Обоснованность выбора предметной области

2. Описание предметной области и промпты

Диаграмма вариантов использования

2.1. Описание предметной области

В качестве объекта для исследований возьмём предметную область "Скриптовые языки для решения научных задач", для которой нам нужно построить диаграмму вариантов использования.

В качестве действующих лиц (акторов) выступают скриптовые языки программирования: Python, R, Julia, Bash, JavaScript, Lua.

Python используется для задач:

1. **Анализ данных и машинное обучение:** Python широко используется в этих областях благодаря библиотекам NumPy, pandas, scikit-learn, TensorFlow и PyTorch.
2. **Статистический анализ:** для проверки гипотез, регрессионного анализа, работы с распределениями и временных рядов используются библиотеки statsmodels, scipy.stats и pandas. Поддерживаются как базовые методы (описательная статистика, t-тесты), так и сложные модели (ARIMA, байесовские подходы).
3. **Визуализация данных:** библиотеки Matplotlib, Seaborn и Plotly позволяют создавать графики и диаграммы.
4. **Символьные вычисления:** библиотека SymPy поддерживает алгебраические преобразования, решение уравнений, дифференцирование и интегрирование.
5. **Обработка больших данных:** Python применяется с инструментами вроде Apache Spark и Dask.
6. **Автоматизация научного workflow:** Python используется для создания пайплайнов, интеграции инструментов и управления экспериментами.
7. **Численное моделирование:** библиотеки SciPy и NumPy предоставляют инструменты для численных расчётов.
8. **Автоматизация рутинных действий в GUI-приложениях:** возможно с помощью библиотек вроде PyAutoGUI или Selenium, а также с помощью разработки собственных модулей расширения.

9. **Разработка веб-интерфейсов для научных инструментов:** реализуется с помощью фреймворков и библиотек (например, Django и Flask)

R используется для задач:

1. **Статистический анализ:** R разработан специально для статистических вычислений и анализа данных.
2. **Анализ данных и машинное обучение:** язык поддерживает множество методов анализа и машинного обучения через пакеты вроде caret и randomForest.
3. **Визуализация данных:** пакеты ggplot2 и lattice предоставляют мощные инструменты для создания графиков.

Julia используется для задач:

1. **Численное моделирование:** Julia оптимизирована для численных расчётов и часто используется в научных вычислениях.
2. **Обработка больших данных:** язык подходит для работы с большими объёмами данных благодаря своей скорости и возможностям параллелизации.
3. **Анализ данных и машинное обучение:** Julia поддерживает анализ данных через библиотеки DataFrames и другие инструменты.

Bash используется для задач:

1. **Автоматизация научного workflow:** Bash часто используется для автоматизации рутинных задач, управления скриптами и интеграции различных инструментов в научных проектах.

JavaScript используется для задач:

1. **Визуализация данных:** библиотеки D3.js и Chart.js позволяют создавать интерактивные графики и диаграммы.
2. **Разработка веб-интерфейсов для научных инструментов:** JavaScript используется для создания интерактивных веб-приложений и интерфейсов.

3. **Автоматизация рутинных действий в GUI-приложениях:** возможно через автоматизацию браузера (например, с помощью Selenium) или создание расширений для браузеров.

Lua используется для задач:

1. **Автоматизация научного workflow:** Lua часто встраивается в другие системы для расширения функциональности (например, в игры, сетевые устройства, редакторы).
2. **Автоматизация рутинных действий в GUI-приложениях:** Lua может использоваться для создания макросов и автоматизации в приложениях, поддерживающих этот язык.

Важной характеристикой системы является то, что несколько скриптовых языков могут решать одни и те же научные задачи, что показывает их конкуренцию или нишевое разделение. Например:

- Задачу «Анализ данных и машинное обучение» выполняют Python, R и Julia.
- Задачу «Статистический анализ» выполняют Python, R.
- Задачу «Автоматизация научного workflow» выполняют Python, Bash и Lua.
- Задачу «Визуализация данных» выполняют Python, R и JavaScript.
- Задачу «Численное моделирование» выполняют Python и Julia.
- Задачу «Автоматизация рутинных действий в GUI-приложениях» выполняют Python, JavaScript и Lua.

2.2. Промпты

A1. Промпт 1 (минимальный)

Сгенерируй PlantUML код для диаграммы вариантов использования для области: “Скриптовые языки для решения научных задач”. В качестве действующих лиц рассматривай сами скриптовые языки программирования. Укажи действующие лица, перечисли основные случаи использования.

A2. Промпт 2 (базовый)

Сгенерируй PlantUML код для диаграммы вариантов использования (use case diagram) для области: "Скриптовые языки для решения научных задач". Укажи действующие лица: Python, R, Julia, Bash, JavaScript, Lua. Перечисли основные случаи использования (научные задачи), такие как анализ данных и машинное обучение, визуализация данных, статистический анализ, численное моделирование, символьные вычисления, обработка больших данных, автоматизация научного workflow, автоматизация рутинных действий в GUI-приложениях, разработка веб-интерфейсов для научных инструментов и др.

A3. Промпт 3 (расширенный)

Сгенерируй PlantUML код для диаграммы вариантов использования, которая моделирует предметную область "Скриптовые языки для решения научных задач". Укажи действующие лица, в качестве которых выступают языки программирования и перечисли случаи использования. Для Python: Анализ данных и машинное обучение; Статистический анализ; Визуализация данных; Символьные вычисления; Обработка больших данных; Автоматизация научного workflow; Численное моделирование; Автоматизация рутинных действий в GUI-приложениях, Разработка веб-интерфейсов для научных инструментов. Для R: Статистический анализ; Анализ данных и машинное обучение; Визуализация данных. Для Julia: Численное моделирование; Обработка больших данных; Анализ данных и машинное обучение. Для Bash: Автоматизация научного workflow. Для JavaScript: Визуализация данных; Разработка веб-интерфейсов для научных инструментов; Автоматизация рутинных действий в GUI-приложениях. Для Lua: Автоматизация научного workflow; Автоматизация рутинных действий в GUI-приложениях. Включи границу экосистемы, которая будет называться "Научные задачи".

A4. Промпт 4 (продвинутый)

Сгенерируй PlantUML код для диаграммы вариантов использования, которая моделирует предметную область "Скриптовые языки для решения научных задач".

Требования к диаграмме:

1. Используй последний стандарт UML

2. Граница экосистемы:
 - a. Название: "Научные задачи"
 - b. Включает все варианты использования
3. Действующие лица (языки программирования):
 - a. Python, R, Julia, Bash, JavaScript, Lua
 - b. Расположи их слева и справа от границы экосистемы для лучшей читаемости
4. Варианты использования - сгруппируй их логически:
 - a. Основные научные задачи:
 - Анализ данных и машинное обучение
 - Статистический анализ
 - Численное моделирование
 - Обработка больших данных
 - Символьные вычисления
 - b. Визуализация и интерфейсы:
 - Визуализация данных
 - Разработка веб-интерфейсов для научных инструментов
 - c. Автоматизация:
 - Автоматизация научного workflow
 - Автоматизация рутинных действий в GUI-приложениях
5. Связи между акторами и вариантами использования:
 - a. Python -> [Анализ данных и машинное обучение, Статистический анализ, Визуализация данных, Символьные вычисления, Обработка больших данных, Автоматизация научного workflow, Численное моделирование, Автоматизация рутинных действий в GUI-приложениях, Разработка веб-интерфейсов для научных инструментов]
 - b. R -> [Статистический анализ, Анализ данных и машинное обучение, Визуализация данных]
 - c. Julia -> [Численное моделирование, Обработка больших данных, Анализ данных и машинное обучение]
 - d. Bash -> [Автоматизация научного workflow]
 - e. JavaScript -> [Визуализация данных, Разработка веб-интерфейсов для научных инструментов, Автоматизация рутинных действий в GUI-приложениях]
 - f. Lua -> [Автоматизация научного workflow, Автоматизация рутинных действий в GUI-приложениях]

6. Визуальные улучшения:

- a. Размести варианты использования логическими группами внутри границы экосистемы
- b. Обеспечь достаточное расстояние между элементами для читаемости
- c. Добавь заголовок диаграммы

Сгенерируй полный код PlantUML с учетом всех этих требований.

3. Результаты генераций диаграммы вариантов использования

3.1. Промпт 1.

1. DeepSeek

Код PlantUML:

```
@startuml
!pragma useIcon false

left to right direction

actor "Python" as py
actor "R" as r
actor "Julia" as jl
actor "MATLAB" as mat
actor "Bash/Shell" as sh

rectangle "Научные вычисления и анализ данных" {
    usecase "Быстрое прототипирование\налгоритмов" as uc1
    usecase "Статистический анализ\ни визуализация данных" as uc2
    usecase "Численное моделирование\ни симуляции" as uc3
    usecase "Символьные вычисления\ни манипуляция формулами" as uc4
    usecase "Интерактивное исследование\нданных (ноутбуки)" as uc5
    usecase "Машинное обучение\ни Data Science" as uc6
    usecase "Обработка и очистка\псырых данных" as uc7
    usecase "Автоматизация научных\прасчетов и pipeline" as uc8
    usecase "Интеграция с C/Fortran\нбиблиотеками" as uc9
    usecase "Параллельные вычисления\ни HPC" as uc10
}

' Связи Python
py --> uc1
py --> uc2
py --> uc3
py --> uc5
py --> uc6
py --> uc7
py --> uc8
py --> uc9
py --> uc10

' Связи R
r --> uc1
r --> uc2
r --> uc5
r --> uc6
r --> uc7
r --> uc8
r --> uc9

' Связи Julia
jl --> uc1
jl --> uc2
```

```
jl --> uc3
jl --> uc5
jl --> uc6
jl --> uc8
jl --> uc9
jl --> uc10

' Связи MATLAB
mat --> uc1
mat --> uc2
mat --> uc3
mat --> uc5
mat --> uc8
mat --> uc10

' Связи Bash/Shell
sh --> uc7
sh --> uc8

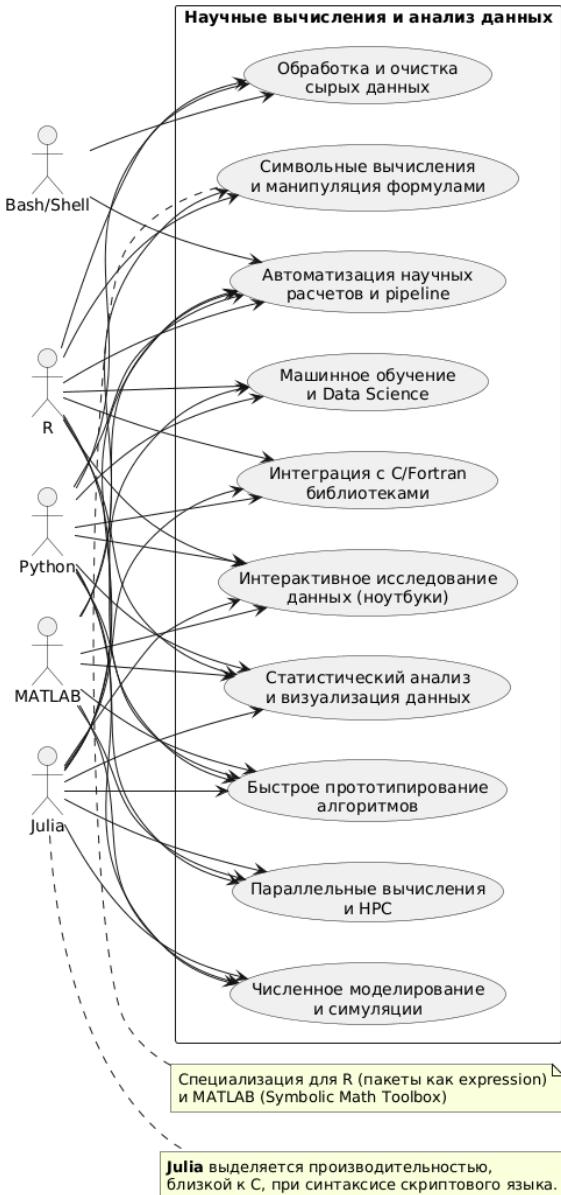
' Специфические случаи использования
r --> uc4
mat --> uc4

note top of uc4
Специализация для R (пакеты как expression)
и MATLAB (Symbolic Math Toolbox)
end note

note right of jl
<b>Julia</b> выделяется производительностью,
близкой к C, при синтаксисе скриптового языка.
end note

@enduml
```

Диаграмма:



Анализ: Код отработал с первого запроса. Однако включен MATLAB, который не является скриптовым языком. Указаны задачи, которые не предполагались при описании предметной области, что произошло за счёт достаточно общего запроса. Неверные стрелки по стандарту UML от акторов к вариантам использования.

2. ChatGPT

Код PlantUML:

```
@startuml
skinparam actorStyle awesome

title Скриптовые языки для решения научных задач

' --- Действующие лица (языки) ---
actor Python
```

```

actor R
actor "Julia" as Julia
actor "MATLAB (Script)" as Matlab
actor "Bash" as Bash

' --- Случаи использования ---
usecase "Научные вычисления\n(численные методы,\nалгоритмы
оптимизации)" as UC_Calc
usecase "Статистический анализ\nи моделирование" as UC_Stats
usecase "Визуализация данных\n(графики, диаграммы)" as UC_Vis
usecase "Обработка больших данных\nи автоматизация" as UC_Data
usecase "Машинное обучение" as UC_ML
usecase "Построение\nпрототипов алгоритмов" as UC_Proto
usecase "Автоматизация экспериментов\nи запуск вычислительных
задач" as UC_Auto

' --- Связи между языками и вариантами использования ---
' Python
Python --> UC_Calc
Python --> UC_Stats
Python --> UC_Vis
Python --> UC_Data
Python --> UC_ML
Python --> UC_Proto
Python --> UC_Auto

' R
R --> UC_Stats
R --> UC_Vis
R --> UC_ML
R --> UC_Proto

' Julia
Julia --> UC_Calc
Julia --> UC_ML
Julia --> UC_Proto
Julia --> UC_Vis

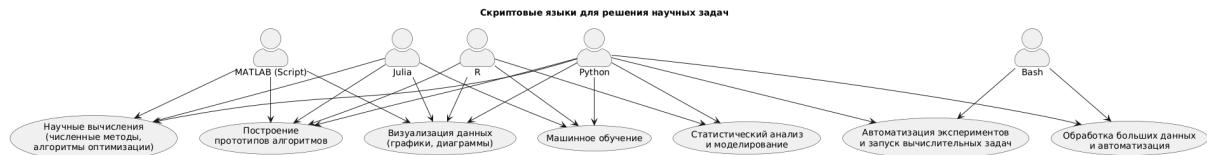
' MATLAB
Matlab --> UC_Calc
Matlab --> UC_Vis
Matlab --> UC_Proto

' Bash
Bash --> UC_Auto
Bash --> UC_Data

@enduml

```

Диаграмма:



Анализ: Код отработал с первого запроса. Однако включен MATLAB, который не является скриптовым языком. Указаны задачи, которые не предполагались при описании предметной области, что произошло за счёт достаточно общего запроса. Неверные стрелки по стандарту UML от акторов к вариантам использования. Также неверное использование обозначений для акторов.

3.2. Промпт 2.

1. DeepSeek

Код PlantUML:

```
@startuml
!theme plain

actor "Python" as py
actor "R" as R
actor "Julia" as jl
actor "Bash" as sh
actor "JavaScript" as js
actor "Lua" as lua

rectangle "Научные задачи" {
    (Анализ данных и машинное обучение) as uc1
    (Статистический анализ) as uc2
    (Визуализация данных) as uc3
    (Численное моделирование) as uc4
    (Символьные вычисления) as uc5
    (Обработка больших данных) as uc6
    (Автоматизация научного workflow) as uc7
    (Автоматизация рутинных действий в GUI) as uc8
    (Разработка веб-интерфейсов\пдля научных инструментов) as uc9
}

' Связи для Python
py --> uc1
py --> uc2
py --> uc3
py --> uc4
py --> uc5
py --> uc6
py --> uc7
py --> uc8 : например, с PyAutoGUI
py --> uc9 : например, с Flask/Django

' Связи для R
```

```
R --> uc1
R --> uc2
R --> uc3
R --> uc4
R --> uc6
R --> uc7

' Связи для Julia
jl --> uc1
jl --> uc2
jl --> uc3
jl --> uc4
jl --> uc5
jl --> uc6
jl --> uc7

' Связи для Bash
sh --> uc7

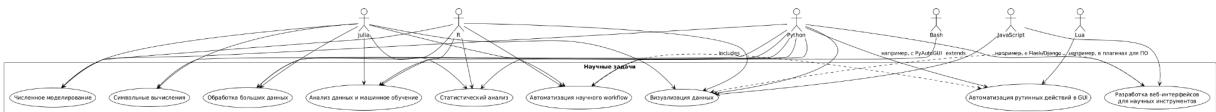
' Связи для JavaScript
js --> uc3
js --> uc9

' Связи для Lua
lua --> uc8 : например, в плагинах для ПО

' Отношения между Use Cases (необязательно, но показывает зависимости)
uc7 .> uc8 : extends
uc9 .> uc3 : includes

@enduml
```

Диаграмма:



Анализ: Отработал с первого раза. Отобразил все запрошенные языки программирования. Есть две стрелки `include` которые непонятно откуда берут своё начало. Стрелки сильно пересекаются и сложно (практически невозможно) определить какие скриптовые языки программирования какие задачи решают. Некоторых связей недостает, например, для языка JavaScript нет стрелки к задаче по Автоматизации рутинных действий в GUI. Используются неверные стрелки (по стандарту UML) от акторов к вариантам использования.

2. ChatGPT

Код PlantUML:

```

@startuml
title Скриптовые языки для решения научных задач

left to right direction

' -----
' Акторы (скриптовые языки)
' -----

actor Python
actor R
actor Julia
actor Bash
actor JavaScript
actor Lua

' -----
' Варианты использования
' -----

usecase "Анализ данных и \nмашинное обучение" as UC_Analytics_ML
usecase "Визуализация данных" as UC_Visualization
usecase "Статистический анализ" as UC_Statistics
usecase "Численное моделирование" as UC_NumericalModeling
usecase "Символьные вычисления" as UC_Symbolic
usecase "Обработка больших данных" as UC_BigData
usecase "Автоматизация научного\nc workflow" as UC_SciWorkflow
usecase "Автоматизация рутинных\ndействий в GUI-приложениях" as UC_GUIAutomation
usecase "Разработка веб-интерфейсов\ndля научных инструментов" as UC_WebUI
usecase "Обработка текстов и файлов" as UC_TextFile
usecase "Интеграция с существующими\nc научными библиотеками" as UC_Integration

' -----
' Связи акторов с use case
' -----


' Python
Python --> UC_Analytics_ML
Python --> UC_Visualization
Python --> UC_Statistics
Python --> UC_NumericalModeling
Python --> UC_Symbolic
Python --> UC_BigData
Python --> UC_SciWorkflow
Python --> UC_GUIAutomation
Python --> UC_WebUI
Python --> UC_TextFile
Python --> UC_Integration

' R
R --> UC_Analysis
R --> UC_Visualization

```

```
R --> UC_Statistics
R --> UC_BigData
R --> UC_Integration

' Julia
Julia --> UC_Analytics_ML
Julia --> UC_NumericalModeling
Julia --> UC_Symbolic
Julia --> UC_BigData
Julia --> UC_Visualization
Julia --> UC_Integration

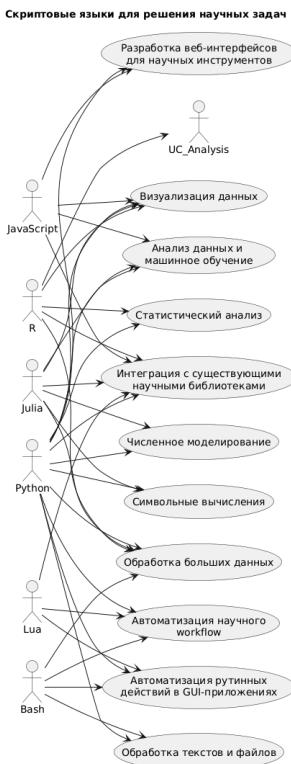
' Bash
Bash --> UC_SciWorkflow
Bash --> UC_TextFile
Bash --> UC_BigData
Bash --> UC_GUIAutomation

' JavaScript
JavaScript --> UC_WebUI
JavaScript --> UC_Visualization
JavaScript --> UC_Analytics_ML
JavaScript --> UC_Integration

' Lua
Lua --> UC_GUIAutomation
Lua --> UC_SciWorkflow
Lua --> UC_Integration

@enduml
```

Диаграмма:



Анализ: Отработал с первого раза. Отобразил все запрошенные языки программирования. Стрелки сильно пересекаются и сложно (практически невозможно) определить какие скриптовые языки программирования какие задачи решают. Некоторых связей недостает и также есть лишние стрелки. Используются неверные стрелки (по стандарту UML) от акторов к вариантам использования. Также есть ошибка в виде актора UC_Analysis, которого не должно быть.

3.3. Промпт 3.

1. DeepSeek

Код PlantUML:

```

@startuml
left to right direction

actor "Python" as Py
actor "R" as R
actor "Julia" as Jl
actor "Bash" as Sh
actor "JavaScript" as Js
actor "Lua" as Lua

rectangle "Научные задачи" {
    (Анализ данных и машинное обучение) as ML
    (Статистический анализ) as Stats
    (Визуализация данных) as Viz
}

```

```

(Символьные вычисления) as Symb
(Обработка больших данных) as BigData
(Автоматизация научного workflow) as Workflow
(Численное моделирование) as Modeling
(Автоматизация рутинных действий в GUI-приложениях) as GUIAuto
    (Разработка веб-интерфейсов для научных инструментов)
as WebDev
}

Py --> ML
Py --> Stats
Py --> Viz
Py --> Symb
Py --> BigData
Py --> Workflow
Py --> Modeling
Py --> GUIAuto
Py --> WebDev

R --> Stats
R --> ML
R --> Viz

Jl --> Modeling
Jl --> BigData
Jl --> ML

Sh --> Workflow

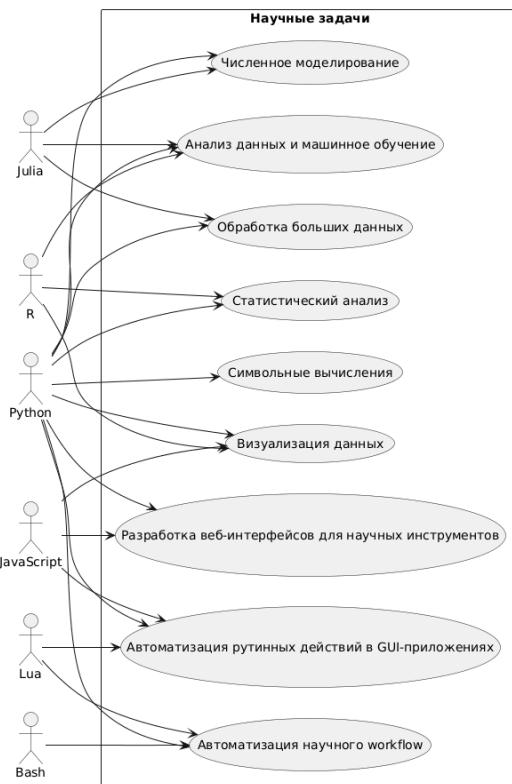
Js --> Viz
Js --> WebDev
Js --> GUIAuto

Lua --> Workflow
Lua --> GUIAuto

@enduml

```

Диаграмма:



Анализ: Диаграмма отрисована с первого раза. Хорошо видны связи, есть пересечения, но легко можно проследить кто с чем связан. Однако используются неверные стрелки (по стандартам UML) от акторов к вариантам использования.

2. ChatGPT

Код PlantUML:

```

@startuml
left to right direction
skinparam actorStyle awesome

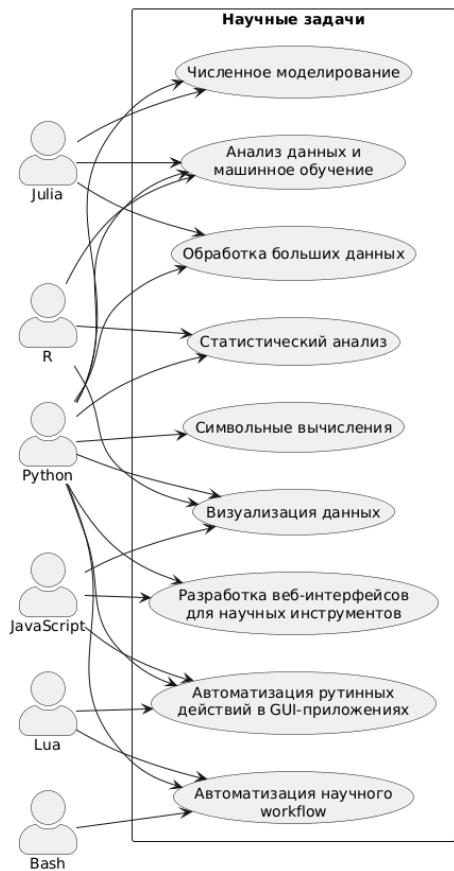
actor Python
actor R
actor Julia
actor Bash
actor JavaScript
actor Lua

rectangle "Научные задачи" {
    usecase "Анализ данных и\пмашинное обучение" as UC_ML
    usecase "Статистический анализ" as UC_Stat
    usecase "Визуализация данных" as UC_Vis
    usecase "Символьные вычисления" as UC_Sym
    usecase "Обработка больших данных" as UC_BigData
    usecase "Автоматизация научного\nc workflow" as UC_Workflow
    usecase "Численное моделирование" as UC_Num
    usecase "Автоматизация рутинных\ndействий в GUI-приложениях" as UC_GUI
}

```

```
usecase "Разработка веб-интерфейсов\пдля научных  
инструментов" as UC_Web  
}  
  
' --- Python ---  
Python --> UC_DL  
Python --> UC_Stat  
Python --> UC_Vis  
Python --> UC_Sym  
Python --> UC_BigData  
Python --> UC_Workflow  
Python --> UC_Num  
Python --> UC_GUI  
Python --> UC_Web  
  
' --- R ---  
R --> UC_Stat  
R --> UC_DL  
R --> UC_Vis  
  
' --- Julia ---  
Julia --> UC_Num  
Julia --> UC_BigData  
Julia --> UC_DL  
  
' --- Bash ---  
Bash --> UC_Workflow  
  
' --- JavaScript ---  
JavaScript --> UC_Vis  
JavaScript --> UC_Web  
JavaScript --> UC_GUI  
  
' --- Lua ---  
Lua --> UC_Workflow  
Lua --> UC_GUI  
  
enduml
```

Диаграмма:



Анализ: Диаграмма отрисована с первого раза. Хорошо видны связи, есть пересечения, но легко можно проследить кто с чем связан. Однако используются неверные стрелки (по стандартам UML) от акторов к вариантам использования. Также неверное использование обозначений для акторов.

3.4. Промпт 4.

1. DeepSeek

Код PlantUML:

```

@startuml
left to right direction

title Диаграмма вариантов использования: Скриптовые языки для решения научных задач

actor "Python" as py
actor "R" as r
actor "Julia" as jul
actor "Bash" as bash
actor "JavaScript" as js
actor "Lua" as lua

rectangle "Научные задачи" {
    Численное моделирование
    Анализ данных и машинное обучение
    Обработка больших данных
    Статистический анализ
    Символьные вычисления
    Визуализация данных
    Разработка веб-интерфейсов для научных инструментов
    Автоматизация рутинных действий в GUI-приложениях
    Автоматизация научного workflow
}
    
```

```
usecase "Анализ данных и машинное обучение" as uc1
usecase "Статистический анализ" as uc2
usecase "Численное моделирование" as uc3
usecase "Обработка больших данных" as uc4
usecase "Символьные вычисления" as uc5

usecase "Визуализация данных" as uc6
usecase "Разработка веб-интерфейсов\nдля научных инструментов" as uc7

usecase "Автоматизация научного workflow" as uc8
usecase "Автоматизация рутинных действий\nв GUI-приложениях" as uc9
}

' Связи для Python
py --> uc1
py --> uc2
py --> uc3
py --> uc4
py --> uc5
py --> uc6
py --> uc7
py --> uc8
py --> uc9

' Связи для R
r --> uc1
r --> uc2
r --> uc6

' Связи для Julia
jul --> uc1
jul --> uc3
jul --> uc4

' Связи для Bash
bash --> uc8

' Связи для JavaScript
js --> uc6
js --> uc7
js --> uc9

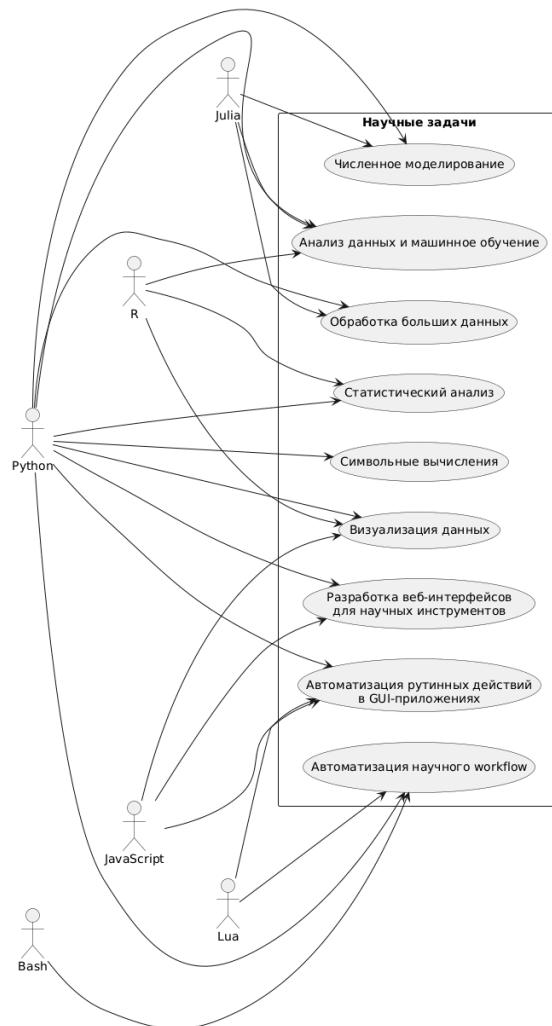
' Связи для Lua
lua --> uc8
lua --> uc9

' Размещение акторов слева и справа
py -[hidden]- r
r -[hidden]- jul
bash -[hidden]- js
js -[hidden]- lua
```

```
@enduml
```

Диаграмма:

Диаграмма вариантов использования: Скриптовые языки для решения научных задач



Анализ: Отрисовался с первого раза. Однако не выполнил требование про расположение акторов по разные стороны от вариантов использования. Визуально связи читаемые, однако визуально нагромождены. Используются неверные стрелки (по стандартам UML) от акторов к вариантам использования.

2. ChatGPT

Код PlantUML:

```
@startuml
' Настройка направления для большей читаемости
left to right direction
skinparam packageStyle rectangle
```

```

' Акторы - языки программирования, слева и справа
actor "Python" as Python
actor "R" as R
actor "Julia" as Julia
actor "Bash" as Bash
actor "JavaScript" as JS
actor "Lua" as Lua

rectangle "Научные задачи" {
    ' --- Основные научные задачи ---
    package "Основные задачи" {
        (Анализ данных и машинное\обучение) as UC_DataML
        (Статистический анализ) as UC_Stats
        (Численное моделирование) as UC_NumModel
        (Обработка больших данных) as UC_BigData
        (Символьные вычисления) as UC_Symbolic
    }

    ' --- Визуализация и интерфейсы ---
    package "Визуализация / Интерфейсы" {
        (Визуализация данных) as UC_Visualization
        (Веб-интерфейсы для\научных инструментов) as UC_WebUI
    }

    ' --- Автоматизация ---
    package "Автоматизация" {
        (Автоматизация научного workflow) as UC_AutoWorkflow
        (Автоматизация рутинных действий\в GUI-приложениях) as
UC_AutoGUI
    }
}

' Связи акторов и use-cases
Python --> UC_DataML
Python --> UC_Stats
Python --> UC_Visualization
Python --> UC_Symbolic
Python --> UC_BigData
Python --> UC_AutoWorkflow
Python --> UC_NumModel
Python --> UC_AutoGUI
Python --> UC_WebUI

R --> UC_Stats
R --> UC_DataML
R --> UC_Visualization

Julia --> UC_NumModel
Julia --> UC_BigData
Julia --> UC_DataML

Bash --> UC_AutoWorkflow

```

```
JS --> UC_Visualization
```

```
JS --> UC_WebUI
```

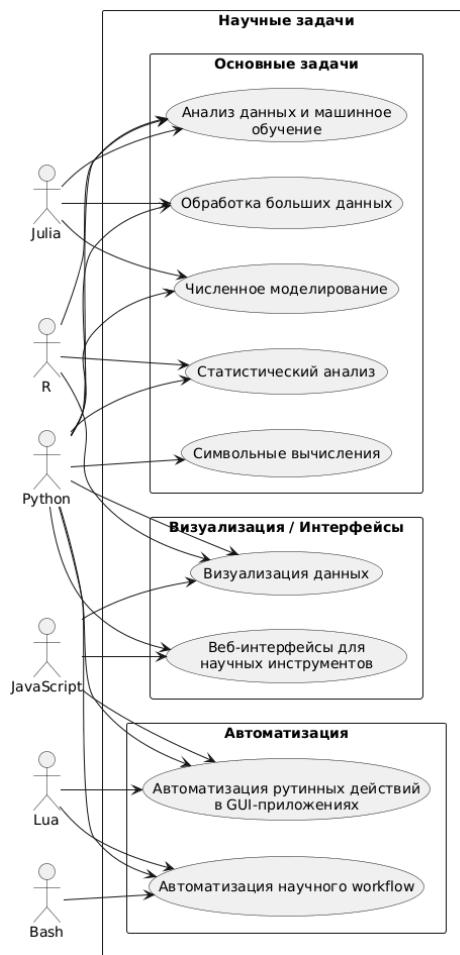
```
JS --> UC_AutoGUI
```

```
Lua --> UC_AutoWorkflow
```

```
Lua --> UC_AutoGUI
```

```
@enduml
```

Диаграмма:



Анализ: Отрисовался с первого раза. Однако не выполнил требование про расположение акторов по разные стороны от вариантов использования. Визуально связи читаемые, однако визуально нагромождены. Явно выделены задачи по группам, как и было запрошено. Не добавлено название диаграммы. Используются неверные стрелки (по стандартам UML) от акторов к вариантам использования.

4. Сравнительный анализ результатов

Таблица 1: Анализ результатов генераций

Модель	Работоспособность кода	Ошибки	Соответствие логике UML	Сложность и стиль	Соответствие содержанию ТЗ
Промпт 1					
DeepSeek	Высокая	Нет	Частичное	Низкое качество визуализации	Плохо соответствует
ChatGPT	Высокая	Нет	Низкое	Хорошее качество визуализации	Плохо соответствует
Промпт 2					
DeepSeek	Высокая	Нет	Частичное	Низкое качество визуализации	Практически соответствует
ChatGPT	Высокая	Лишний актор	Частичное	Низкое качество визуализации	Практически соответствует
Промпт 3					
DeepSeek	Высокая	Нет	Почти полное	Хорошее качество визуализации	Соответствует
ChatGPT	Высокая	Нет	Низкое	Среднее качество визуализации	Соответствует
Промпт 4					
DeepSeek	Высокая	Нет	Почти полное	Среднее качество визуализации	Практически соответствует
ChatGPT	Высокая	Нет	Почти полное	Среднее качество визуализации	Практически соответствует

5. Выводы

На основе анализа таблицы с оценками можно сделать следующие выводы:

1. Из двух моделей: DeepSeek и ChatGPT на конкретно этой задаче лучше отработала модель DeepSeek. Он не использовал неправильные обозначения для акторов, у него лучше соответствие со стандартом UML.
2. Обе модели проигнорировали требование в промпте 4 о расположении акторов относительно вариантов использования. Также с большим текстом диаграммы получились более нагромождёнными, чем, например, в промпте 3 с меньшей детализацией по визуализации.
3. По совокупности факторов лучше всего отработал Промпт 3.
4. Лучшей парой модель-промпта является: DeepDeek с промптом 3.

Все диаграммы можно посмотреть на Яндекс диске по ссылке:
<https://disk.yandex.ru/d/MHXgAO2V-vHILQ>