Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №6 по курсу "Моделирование"

Тема Автомобильный конвейер

Студент Леонов В.В.

Группа ИУ7-76Б

Оценка (баллы) _____

Преподаватель Рудаков И.В.

Москва — 2022 г.

Формальная постановка задачи

Слудует выполнить моделирование процесса выполнения заказов на автомобильном конвейере:

- заказы от клиентов поступают в конструкторское бюро с интервалом 5 ± 1;
- существуют два вида заказов: гражданский и ведомственный, время утверждения которых составляет 10 ± 2 и 15 ± 5, время выполнения составляет 5 ± 1 и 8 ± 3 (требуется дополнительние дооснащение);
- существуют два оператора: регулярный и с более высоким уровнем допуска для обработки ведомственных заказов;
- при попадании заявки в бюро она передается оператору с наименьшим минимально необходимым уровнем допуска;
- после завершения процесса выпуска модели следует выполнить соответствующую сертификацию, время которой составляет 1 ± 2 и 3 ± 2 для гражданских и ведомственных заказов соответственно;
- по статистике 5% гражданских и 2% ведомственных автомобилей имеют дефекты, которые необходимо устранить, отправив автомобиль обратно в сборочное отделение на доработку, после которого будет выполнена повторная сертификация и отправка заказщику.

Будет выполнено моделирование плана на 300 выпущенных автомобилей (30% заказов являются ведомственными).

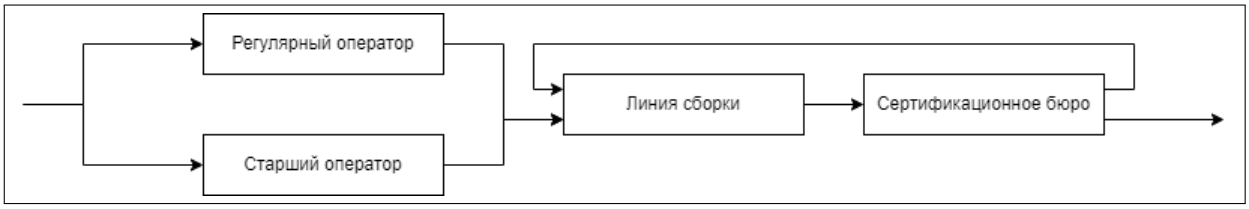

Рисунок 1 – Концептуальная схема

Эндогенные переменные:

- время обработки заказа каждым оператором;
- время выполнения каждого заказа;
- время прохождения сертификации;
- вероятность прохождения сертификации.

Экзогенные переменные:

- количество выпущенных автомобилей каждого из типов;
- количество упущенных заказов на автомобили каждого из типов.

Средства реализации

Язык программирования — Python.
GUI — QT.

Листинг кода

```
1  class Event:
2      def __init__(self, time: float, stage: str, type: str = "None",
           operator: str = "None") -> None:
3          self.time = time
4          # STAGES = ["New", "Generated", "Designed", "Built", "Certifyed",
               "Finished"]
5          self.stage = stage
6          # TYPES = ["Default", "Government"]
7          self.type = type
8          # OPERATORS = ["Middle", "Senior"]
9          self.operator = operator
10
11     def new_stage(self, delta: float, new_stage: str):
12         return Event(self.time + delta, new_stage, self.type,
               self.operator
13
14
15 class Events:
16     def __init__(self) -> None:
17         self.__pool: list[Event] = []
18
19     def append(self, e: Event):
20         i = 0
21         while i < len(self.__pool) and self.__pool[i].time < e.time:
22             i += 1
23         if 0 < i < len(self.__pool):
24             self.__pool.insert(i - 1, e)
25         else:
26             self.__pool.insert(i, e)
27
28     def pop(self, index):
29         return self.__pool.pop(index)
```

```
class Model:
    def __init__(self,
                 generator,
                 operators,
                 builders,
                 certifiers) -> None:
        self.generator = generator
        self.operators = operators
        self.builders = builders
        self.certifiers = certifiers
        self.processed_tasks = {"Default": 0, "Government": 0}
        self.lost = {"Default": 0, "Government": 0}

    def total_processed(self):
        return self.processed_tasks["Default"] +
            self.processed_tasks["Government"]

    def start(self, total_tasks: int, g_rate: int, d_c_rate: int,
        g_c_rate: int):
        events = Events()
        events.append(Event(0, "New"))
        middle_flag = False
        senior_flag = False
        c_rates = {"Default": d_c_rate, "Government": g_c_rate}

        while self.total_processed() < total_tasks:
            e = events.pop(0)

            if e.stage == "New":
                if random.randint(0, 100) >= g_rate:
                    e.type = "Default"
                else:
                    e.type = "Government"
                new_e1 = e.new_stage(0, "Generated")
                events.append(new_e1)

                new_e2 = e.new_stage(self.generator.generate(), "New")
                events.append(new_e2)
```

```
            elif e.stage == "Generated":
                if (e.type == "Default" and middle_flag == False):
                    e.operator = "Middle"
                    middle_flag = True
                elif (e.type == "Default" and senior_flag == False) or \
                    (e.type == "Government" and senior_flag == False):
                    e.operator = "Senior"
                    senior_flag = True
                else:
                    self.lost[e.type] += 1
                    continue
                new_e =
                    e.new_stage(self.operators[e.type].generate(),"Designed")
                events.append(new_e)
            elif e.stage == "Designed":
                if e.operator == "Middle":
                    middle_flag = False
                elif e.operator == "Senior":
                    senior_flag = False
                new_e = e.new_stage(self.builders[e.type].generate(),
                    "Built")
                events.append(new_e)
            elif e.stage == "Built":
                new_e = e.new_stage(self.certifiers[e.type].generate(),
                    "Certifyed")
                events.append(new_e)
            elif e.stage == "Certifyed":
                if random.randint(0, 100) >= c_rates[e.type]:
                    new_e = e.new_stage(0, "Finished")
                else:
                    new_e = e.new_stage(self.builders[e.type].generate(),
                        "Built")
                events.append(new_e)
            elif e.stage == "Finished":
                self.processed_tasks[e.type] += 1

        return self.processed_tasks, self.lost

    class BaseGenerator:
        def __init__(self, loc: float, scale: float = 0) -> None:
            self.loc = loc
            self.scale = scale

        def generate(self) -> float:
            return uniform.rvs(loc=self.loc - self.scale, scale=2 *
                self.scale, size=1)[0]
```

Демонстрация работы программы

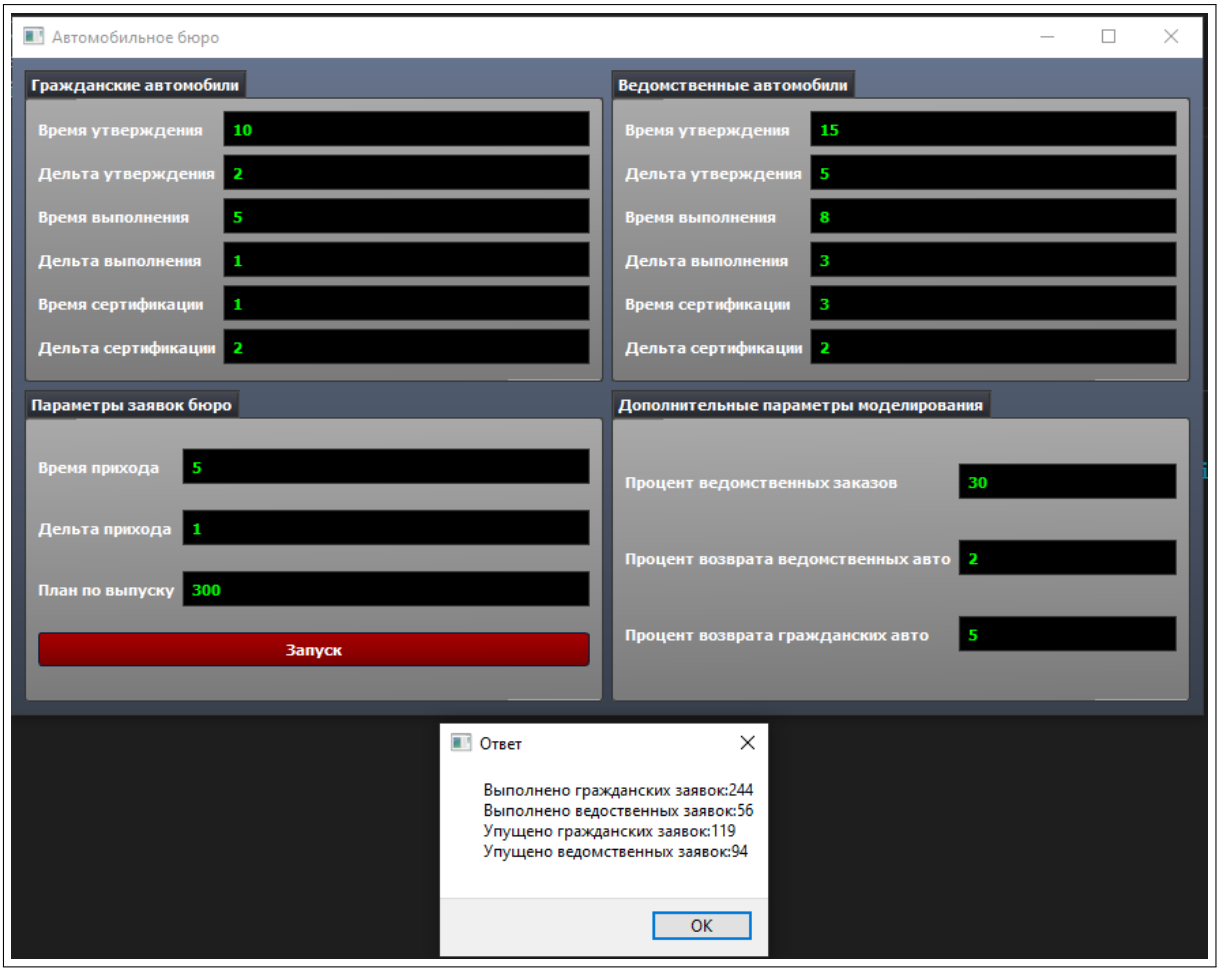

Рисунок 2 – Работа программы

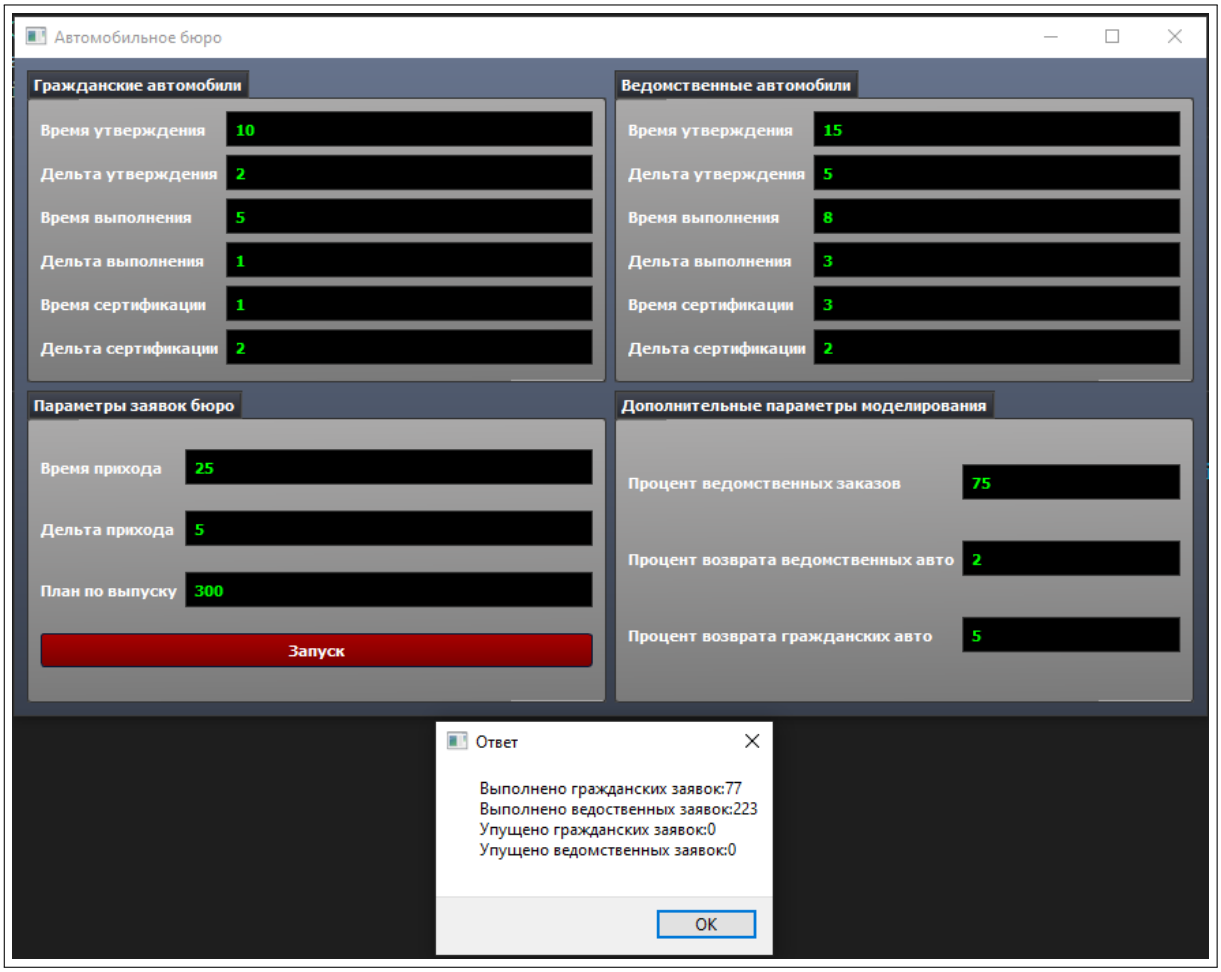

Рисунок 3 – Работа программы