Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Теория систем

Лабораторная работа №1

Группа: P3324

Выполнил:

Круглов Егор Ильич

Преподаватель:

Русак Алёна Викторовна

Санкт-Петербург

2025г.

Задание

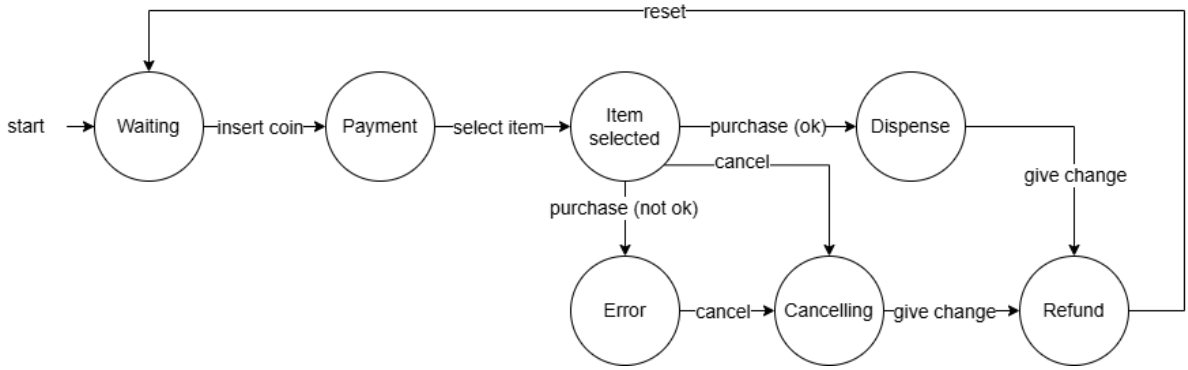
Требуется спроектировать управляющий конечный автомат.

Решение

В качестве конечного автомата был выбран торговый автомат. Опишем его состояния:

- Waiting – ожидание того, что с ним начнут взаимодействовать (вносить деньги)
- Payment – состояние внесения денег в автомат
- Item selected – состояние, когда пользователь выбрал товар
- Dispense – состояние выдачи товара
- Refund – состояние возврата денег (сдачи или при отмене)
- Error – состояние ошибки
- Cancelling – состояние отмены

Диаграмма состояний:



Описание конечного автомата:

Состояния (X): waiting, payment, item_selected, dispense, error, cancelling, refund.

Входные сигналы (U): insert_coin, select_item, confirm_purchase, cancel, give_change, reset.

Выходные сигналы (Y): waiting, payment, item_selected, dispense, error, cancelling, refund.

Функция перехода $\delta: X \times U \to X$ задается таблицей:

Триггер	Исходное состояние	Целевое состояние	Условия/действия
insert_coin	waiting, payment	payment	Пополнение баланса
select_item	payment	item_selected	Выбор товара
confirm_purchase	item_selected	dispense	Успешная покупка (_buy_item)
confirm_purchase	item_selected	error	Ошибка (_show_error)

canel	item_selected, error	cancelling	Отмена, подготовка к возврату
give_change	dispense, cancelling	refund	Возврат остатка
reset	refund	waiting	Возвращение в начальное состояние (_reset)

Код представлен на GitHub - https://github.com/KruglovEgor/System-Theory/tree/main/lab1

```
!pip install transitions
from transitions import Machine

class VendingMachine(object):
    def __init__(self):
        self.balance = 0
        self.selected_item = None
        self.items = {
            'A1': {'name': 'Вода', 'price': 50},
            'A2': {'name': 'Шоколад', 'price': 70}
        }

    def _buy_item(self):
        if self.selected_item in self.items and self.balance >= self.items[self.selected_item]['price']:
            print(f"Покупаем товар {self.selected_item} за {self.items[self.selected_item]['price']}.")
            self.balance -= self.items[self.selected_item]['price']
            return True
        return False

    def _show_error(self):
        if self.selected_item not in self.items:
            print("Нет такого товара!")
        elif self.balance < self.items[self.selected_item]['price']:
            print("Недостаточно денег!")
        else:
            print("Какая-то неизвестная ошибка :(")

    def _reset(self):
        if self.balance > 0:
            print(f"Возвращено денег: {self.balance}")
        self.balance = 0
        self.selected_item = None

    def _insert_coin(self, amount): !
        self.balance += amount
        print(f"Баланс пополнен на {amount}. Текущий баланс: {self.balance}")

    def _select_item(self, item):
        self.selected_item = item
        print(f"Выбран товар: {item}.")


# Инициализация автомата
machine = VendingMachine()
```

```
# Определение состояний и переходов
states = ['waiting', 'payment', 'item_selected', 'dispense', 'refund',
'error', 'cancelling']
transitions = [
    {
        'trigger': 'insert_coin',
        'source': ['waiting', 'payment'],
        'dest': 'payment',
        'after': '_insert_coin'
    },
    {
        'trigger': 'select_item',
        'source': 'payment',
        'dest': 'item_selected',
        'after': '_select_item'},
    {
        'trigger': 'confirm_purchase',
        'source': 'item_selected',
        'dest': 'dispense',
        'conditions': '_buy_item',
        'after': 'give_change'
    },
    {
        'trigger': 'confirm_purchase',
        'source': 'item_selected',
        'dest': 'error',
        'unless': '_buy_item',
        'after': '_show_error'
    },
    {
        'trigger': 'cancel',
         'source': ['item_selected', 'error'],
         'dest': 'cancelling',
         'after': 'give_change'
    },
    {
        'trigger': 'give_change',
        'source': ['dispense', 'cancelling'],
        'dest': 'refund',
        'after': 'reset'
    },
    {'trigger': 'reset',
     'source': 'refund',
     'dest': 'waiting',
     'after': '_reset'}
]

# Создаем конечный автомат
fsm = Machine(model=machine, states=states, transitions=transitions,
initial='waiting')

# Пример использования
print("--- Успешная покупка ---")
machine.insert_coin(50)
machine.select_item('A1')
machine.confirm_purchase()
print(f"Текущее состояние: {machine.state}")

print("\n--- Сценарий ошибки ---")
machine.insert_coin(30)
machine.select_item('A2')
machine.confirm_purchase()
print(f"Текущее состояние: {machine.state}")
```

```
print("\n--- Возврат денег ---")
machine.cancel()
print(f"Текущее состояние: {machine.state}")
```

Вывод

В ходе лабораторной работы был спроектирован и реализован конечный автомат, имитирующий работу торгового аппарата.

1. Конечный автомат был описан:
    - Описание на языке множеств X, U, Y
    - Представлена таблица переходов δ
    - Построена диаграмма состояний
2. Был написан код на языке Python:
    - Корректная обработка 7 состояний и 6 переходов.
    - Интеграция проверок баланса и валидации товаров.
3. Особенности реализации:
    - Разделение логики на методы-колбэки (_buy_item, _show_error).
    - Использование условий (conditions) для ветвления переходов.
4. Результаты тестирования подтвердили:
    - Успешную выдачу товара при достаточном балансе.
    - Блокировку операций при ошибках.
    - Работоспособность механизма возврата денег.