Синтез управляющего конечного автомата

1. Описание задания

В рамках данного задания было необходимо реализовать конечный автомат для симуляции работы стиральной машины. Задача предусматривала создание и настройку состояний автомата, переходов между ними, а также имитацию ошибок в процессе выполнения стирки. Программа должна была корректно реагировать на различные сценарии и переходить в соответствующие состояния в зависимости от результата выполнения определённых этапов стирки.

2. Описание реализации конечного автомата

Конечный автомат был реализован на основе библиотеки transitions, которая предоставляет удобный интерфейс для работы с конечными автоматами в языке Python. Основной класс автомата — WashingMachine, который инкапсулирует в себе состояния и переходы между ними.

```
from transitions import Machine
import random  # Импортируем для имитации возможной ошибки


class WashingMachine(object):
   pass

machine = WashingMachine()

# Определение состояний
states = ['idle', 'filling', 'washing', 'draining', 'defective']

# Определение переходов
transitions = [
    {'trigger': 'start', 'source': 'idle', 'dest': 'filling'},
    {'trigger': 'level_reached', 'source': 'filling', 'dest': 'washing'},
    {'trigger': 'timer_done', 'source': 'washing', 'dest': 'draining'},
    {'trigger': 'timer_failed', 'source': 'washing', 'dest': 'defective'},
    {'trigger': 'level_drained', 'source': 'draining', 'dest': 'idle'},
    {'trigger': 'reset', 'source': 'defective', 'dest': 'idle'},
]

washing_machine = Machine(machine, states=states, transitions=transitions,
initial='idle')

# Имитация залива воды (d1)
def fill_water():
    print("Заливаем воду...")
    machine.level_reached()

def wash():
    print("Стирка...")
    # Имитация  таймера
    if random.random() < 0.5:
```

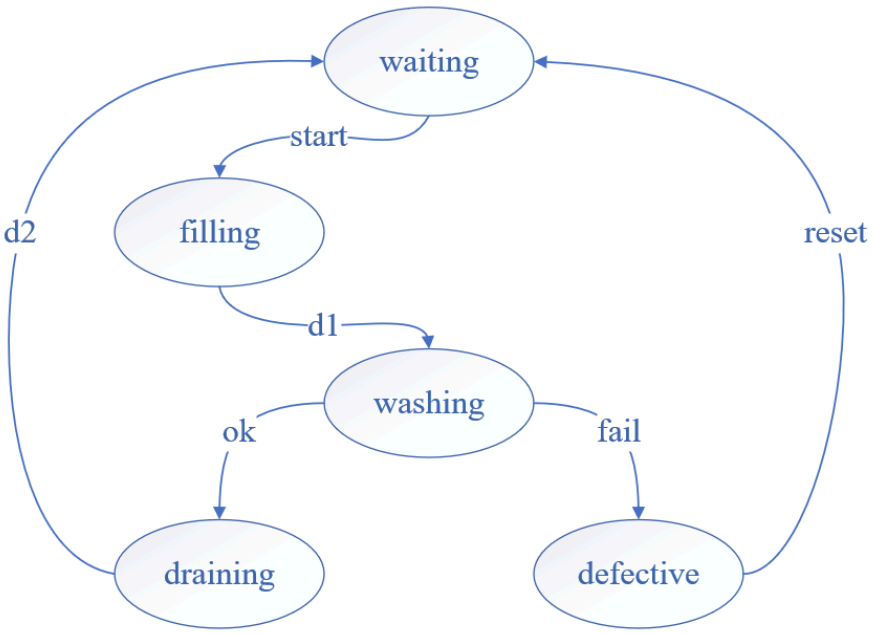
```
            print("Ошибка таймера! Машина переходит в состояние 'defective'")
            machine.timer_failed()
    else:
            print("Таймер успешно завершил работу.")
            machine.timer_done()

# Имитация слива воды (d2)
def drain_water():
    print("Сливаем воду...")
    machine.level_drained()

def simulate_washing():
  print("Начинаем стирку...")
  machine.start()  # Переход в состояние 'filling'
  fill_water()  # Переход в состояние 'washing'
  wash()  # Переход в состояние 'draining' или 'defective'
  if machine.state != 'defective':
      drain_water()
  else:
      print("Требуется ремонт")
      machine.reset()
      print(machine.state)

simulate_washing()
```

3. Диаграмма переходов



4. Вывод

В результате работы конечного автомата была успешно смоделирована работа стиральной машины с возможными сценариями, включая обработку ошибок. Программа корректно переключает состояния в зависимости от хода выполнения этапов стирки. В случае возникновения ошибки, система переходит в дефектное состояние, из которого возможен выход с последующим восстановлением работоспособности автомата.

Данная модель стиральной машины с конечным автоматом может быть легко расширена и адаптирована для моделирования более сложных сценариев, добавления новых состояний и переходов. Она также может быть использована в реальных приложениях для создания систем с детерминированными процессами, которые требуют строгого контроля последовательности выполнения операций.