ФЕДЕРАЛЬНОЕ АГЕНСТВО ЖЕЛЕЗНОДОРОЖНОГО ТАРНСПОРТА

Федерально государственное бюджетное образовательное учреждение
высшего образования
«Иркутский государственный университет путей и сообщения»
(ФГБОУ ВО ИрГУПС)
Факультет «Управление на транспорте и информационнные технологии»
Кафедра «Информационные системы и защита информации»

ОТЧЕТ ПО ПРАКТИКЕ

Производственная - научно-исследовательская работа

НП.430200.090404.000.ПЗ

Выполнил:
студент группы ПИм.1-16-1, Арляпов С.В.
Шифр: 1621345

«___»_____20__г.

Проверил:
ст. пр. Звонков И.В.
_____

«___»_____20__г.

Иркутск 2018

Содержание

Задание на практику

Разработать модуляцию работы системы (группы) лифтов на языке высокого уравня общего назначения, используя тривиальный алгоритм обработки запросов.

В ходе практики должны быть освоены компетенции:

- способность заниматься научными исследованиями;
- способность анализировать и оценивать уровни своих компетенций в сочетании со способностью и готовностью к саморегулированию дальнейшего образования и профессиональной мобильности;
- способность анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями.

Введение

Автоматизированные системы управления активно приходят в повседневную жизнь человечества. Сначала, это были системы для управления производственным процессом на крупных предприятиях, теперь данные системы решают и бытовые задачи. Одной из таких задач является доставка человека с одного этажа на другой. Данная задача достаточно подробно описана в книге С.В. Васильева [1], там имеются абстрактная модель, логическая модель и логический вывод. С предложенным логическим выводом справилась бы система автоматического доказательства теорем А.А. Ларионова [2], но реализация всей системы лифтов на позитивно образованных формулах требует больших трудозатрат и будет носить чисто исследовательский характер.

Построить имитационную модель полностью на логическом выводе достаточно сложно, поэтому следует упростить задачу. А значит необходимо разбить модель на две части: систему взаимодействующих объектов и аппарат принятия решений. Таким образом, разработку обоих частей можно вести независимо дуг от друга. Поскольку при разработке имитационной модели можно сразу не иметь готовую логическую часть, а только ту, что будет содержать основные правила и реализовывать первый рассмотренный подход. Получив стабильную модель системы взаимодействия объектов, можно начать разработку и тестирование более сложных алгоритмов логического вывода. В дальнейшем можно будет усложнять имитационную модель, добавлять дополнительные условия и факторы, тем самым приближая модель к реальным условиям.

Основной решаемой задачей является реализация имитационной модели и алгоритма принятия решений. Которые являются системой разделённой на два блока, реализация которых может вестись на разных языках программирования, например: блок имитации на языке питон, сильной стороной которого является большое количество фреймворков, а логический блок на языке пролог, который поддерживает декларативное программирование. Коммуникация этих двух блоков реализуется через интерфейс, что даёт возможность разрабатывать один блок независимо от другого. Предлагаемая модель позволяет не только вести раздельную разработку блоков, а заменять один блок на другой, например блок с реализацией на другом языке. В последствии планируется заменить логический блок на систему автоматического доказательства теорем [2], где поиск логического вывода будет оптимизирован, а имитационный блок на реальную систему лифтов.

1 Теоретическая часть

SimPy - это Python фреймворк процессо-ориентированной дискретно-событийно системы моделирования. Его диспетчеры событий основаны на функциях-генераторах Python.

Также они могут использоваться для создания асинхронных сетей или для реализации мультиагентных систем (с как моделируемым, так и реальным взаимодействием).

Процессы в SimPy - это просто Python генераторы, которые используются для моделирования активных компонентов, например, таких как покупатели, транспортные средства или агенты. SymPy также обеспечивает различные виды общих ресурсов для моделирования точек с ограниченной пропускной способностью (например, серверов, касс, тоннелей). Начиная с версии 3.1, SimPy также будет обеспечить возможности мониторинга для помощи в сборе статистических данных о ресурсах и процессах.

SymPy представляет собой открытую библиотеку символьных вычислений на языке Python. Цель SymPy - стать полнофункциональной системой компьютерной алгебры (CAS), при этом сохраняя код максимально понятным и легко расширяемым. SymPy полностью написан на Python и не требует сторонних библиотек.

SymPy можно использовать не только как модуль, но и как отдельную программу. Программа удобна для экспериментов или для обучения. Она использует стандартный терминал IPython, но с уже включенными в нее важными модулями SymPy и определенными переменными x, y, z.

2 Реализация програмнного решения

Данная симуляция системы управления группой лифтов с ограниченным количеством кабин и несколькими людьми, которые приходят для попадания с одного этажа на другой. В данной системе используется ресурс для моделирования ограниченного количества кабин. Он также определяет процесс выбора кабины и перевозку ей человека.

Когда человек появляется рядом с шахтой лифта, он вызывает первую из свободных кабин. Как только он его кабина забирает, время ожидания человека прекращается. Он, наконец, добирается до нужного этажа и уходит.

После старта системы начинается генерация людей, они появляются после случайного интервала времени, пока продолжается симуляция.

```
import simpy
import random
import datetime
import os
LOGFILE = 'project.log'
T_INTER = 5
SIM_TIME = 40
NUM_FLOORS = 30
def info(message, env = None):
    now = datetime.datetime.now()
    time = 'null'
    if env is not None :
        time = '%.2f' % (env.now)
    file = open(LOGFILE, 'a')
    #   file.write('%s %s\n' % (NOW.strftime("%Y-%m-%d %H:%M"), message))
    file.write('%s [%s:%s] %s\n'
               % (now.strftime("%d %b %Y %X"), os.getpid(), time, message)
                  )
    file.close()
class Man:
    def __init__(self, ind):
        self.name = 'Man %d' % ind
        self.floor = random.randint(0, NUM_FLOORS // 2)
        self.target = self.floor
        while (self.target == self.floor):
            self.target = random.randint(0, NUM_FLOORS)
        info('%s appears on floor \'%s\' with target \'%s\''
             % (self.name, self.floor, self.target), env)
class Cabine(object):
    def __init__(self):
        self.state = 0
        self.floor = 0
class Elevator(object):
    def __init__(self, env, num_machines):
        self.env = env
        self.cab = simpy.Resource(env, num_machines)
        self.cabs = [Cabine() for i in range(num_machines)]
        self.n_cabs = num_machines

    def get_free_cab(self):
        for i in range(self.n_cabs):
            if (self.cabs[i].state == 0):
                return i
    def moving(self, man):
        cid = self.get_free_cab()
```

```
            self.cabs[cid].state = 1
            moving_time = abs(man.target - man.floor)
            waiting = abs(man.floor - self.cabs[cid].floor) + moving_time
            yield self.env.timeout(waiting)
            self.cabs[cid].state = 0
            info('%s has reached by Cab %d' % (man.name, cid), env)

def call(env, ind, elev):
    man = Man(ind)
    with elev.cab.request() as request:
        yield request
        info('%s is moving' % (man.name), env)
        yield env.process(elev.moving(man))
def people_generator(env):
    elev = Elevator(env, 2);
    i = 0
    env.process(call(env, i, elev))
    while True:
        yield env.timeout(random.randint(T_INTER - 2, T_INTER + 2))
        i += 1
        env.process(call(env, i, elev))
info('Start session')
env = simpy.Environment()
env.process(people_generator(env))
env.run(until=SIM_TIME)
info('Session end')
```

При том, что данная программа является многопоточной, стоит отметить тот факт, что здесь используются общие ресурсы. Эти ресурсы могут использоваться для ограничения количества процессов, использующих их одновременно. Процесс должен запрашивать право использования ресурса. Как только право на использование больше не требуется, оно должно быть выпущено. Данная система смоделирована как ресурс с ограниченным количеством кабин. Люди прибывают к кнопке вызова кабины и просят выполнить перевозку на другой этаж. Если все кабины заняты, человек должен ждать, пока один из пассажиров не закончит поездку и не освободит кабину.

3 Пример работы реализации

Данный пример иллюстрирует работу группы лифтов, где их количество равно 2, в здании с 30 этажами. В ходе работы системы появится люди в случайные моменты времени на случайных этажах этажах, и у каждого человека целью является добратьна на другой этаж.

Изначально первая кабина находится на первом этаже e_0, а вторя на втором e_1. Ниже приведён лог показывающий работу системы:

```
19 May 2018 16:00:58 [51:null] Session end
23 May 2018 21:41:15 [23:null] Start session
23 May 2018 21:41:15 [23:0] Man 0 appears on floor '0' with target '24'
23 May 2018 21:41:15 [23:0] Man 0 is moving
23 May 2018 21:41:15 [23:12] Man 1 appears on floor '14' with target '5'
23 May 2018 21:41:15 [23:12] Man 1 is moving
23 May 2018 21:41:15 [23:24] Man 0 has reached by Cab 0
23 May 2018 21:41:15 [23:24] Man 2 appears on floor '2' with target '10'
23 May 2018 21:41:15 [23:24] Man 2 is moving
23 May 2018 21:41:15 [23:33] Man 3 appears on floor '3' with target '1'
23 May 2018 21:41:15 [23:34] Man 2 has reached by Cab 0
23 May 2018 21:41:15 [23:34] Man 3 is moving
23 May 2018 21:41:15 [23:35] Man 1 has reached by Cab 1
23 May 2018 21:41:15 [23:39] Man 3 has reached by Cab 0
23 May 2018 21:41:15 [23:null] Session end
9 May 2018 16:00:58 [51:null] Session end
```

Изучив выше изложенный журнал, можно увидеть, что завремя симуляции появилось 4 человека, каждый человек доставлен.

Заключение

В результате прохождения практики была разработана программная реализация на языке Python с использованием библиотеки SimPy, который же в свою очередь существенно упрощает процесс моделирования. Данная реализация модулирует процесс работы и управления группы лифтов.

На данном этапе реализован простейший алгоритм управления, следующим шагом будет разработка интеллектуальной обработки запросов, то есть реализация логического блока. Если удаться сделать логический блок отдельной самостоятельной программой, то разрабатываемая система станет более гибкой в вопросах разработки и настройки.

А также получены практические навыки во владении культурой мышления, выстраивании логики рассуждений и высказываний, основанных на интерпретации данных, интегрированных их разных областей науки и техники, выносе суждения на основании неполных данных.

Литература

[1] С. Н. Васильев. Интеллектуальное управление динамическими системами / С. Н. Васильев, А. К. Жерлов, Е. А. Федосов, Б. Е. Федунов - М.. Физико-математическая литература, 2000. - 352 с.

[2] А. А. Ларионов. Программные технологии для эффективного поиска логического вывода в исчислении позитивно-образованных формул / А. А. Ларионов, Е. А. Черкашин – Иркутск : Изд-во ИГУ, 2013. – 104 с.