ИУ5-32Б,

Колесников Никита

Отчет по лабораторным работам 1-6

Цель: Создание системы оценки рисков и расчета надежности для IT-проектов

Разобьем задачу на подзадачи удовлетворяющие условия для каждой из 6 лабораторных работ.

1 Лаба
Реализация выбора значимых критериев для многокритериального принятия решения (выбора критичных компонентов ИТ-проекта).
По заданному списку критериев собрать с пользователя:

1. *текущая оценка значимости критерия по 5-бальной шкале*
2. *желаемая оценка значимости критерия по 5-бальной шкале*
3. *вес критерия (сумма всех весов = 100%)*
4. *уровень отсечения (принцип Парето, инженерный принцип)*

Провести расчет взвешенной оценки, ранжирование критериев и отсечение значимых по заданному уровню. Изучить проект на наличие ранее определенных значимых критериев

2 Лаба
Реализация группового многокритериального выбора критичных компонентов ИТ-проекта.
По заданному списку компонентов провести оценку по значимых критериям из Лабы 1 по 5-бальной шкале со стороны заказчика и со стороны исполнителя.Составить бинарную матрицу и распределить компоненты на группы по их важности.

3 Лаба
Провести проверку согласованности решения по принципу Парето (построить матрицы попарного сравнения и определить коэф. согласованности.)
Определить группы критичных компонентов у заказчика и исполнителя.
Определить обобщенный согласованный список критичных компонентов.
Реализовать определение и сохранение критичных компонентов в зависимости от стадии IT-продукта

4 Лаба
Реализация решения системы линейных алгебраических уравнений, описывающих функционирование критичных компонентов ИТ-проекта. Метод динамики средних.
По заданным графам состояний критичных компонентов полученных в Лабе 2 -3 собрать с пользователя:

1. *Кол-во объектов в компоненте (если выбрано множество)*
2. *Интенсивности перехода между состояниями*
3. *Ущерб, приносящий одним объектом, находящимся в неработоспособном состоянии (по каждому не работоспособному состоянию)*

Рассчитать по заданным системам уравнений:

1. *Кол-во объектов, находящихся в каждом состоянии (если выбрано множество)*
2. *Вероятность нахождения объекта в каждом состоянии (если выбран один объект)*
3. *Общая надежность компонента*
4. *Совокупный ущерб компонента*

5 Лаба
Расчет рисков и надежности компонентов не подчиняющихся системе линейных алгебраических уравнений

6 Лаба

Реализация решения системы линейных алгебраических уравнений, описывающих взаимодействие критичных компонентов ИТ-проекта.

По заданным графам состояний критичных компонентов полученных в Лабе 2 собрать с пользователя:

1. *Выбор методики расчета (множество объектов в компоненте или один)*
2. *Кол-во объектов в компоненте (если выбрано множество)*
3. *Интенсивности перехода между состояниями*
4. *Уравнения управляющих воздействий (равенства между интенсивностями переходов)*
5. *Ущерб, приносящий одним объектом, находящимся в неработоспособном состоянии (по каждому не работоспособному состоянию)*

Рассчитать по заданным системам уравнений:

1. *Параметры системы*
2. *Общая надежность проекта*
3. *Совокупный ущерб проекта*

Текст программы

```
import sys
#Проверк наличия файла
import os.path
#Решение системы уравнений
from sympy import symbols, Eq, nsolve
#Вывод изображения
from matplotlib import pyplot as plt
from matplotlib import image as mpimg
#Математический корень
from math import sqrt
#Математический факториал
from math import factorial
#Математический експонент
from math import exp
#Класс компонент
class component:
    importance=0
    group_executor=0
    group_customer=0
    group_difference=0
    group=0
    def __init__(self,name, group):
        self.name = name
        self.group=group
    def __str__(self):
        return f"{self.name}, {self.importance}, {self.group_executor},
{self.group_customer},{self.group_difference} {self.group}"
#Класс критерий
class criteria:
    Y=0
    K=0
    def __init__(self, name,Y,K):
        self.name = name
        self.Y = Y
        self.K = K
    def __str__(self):
        return f"{self.name}  {self.Y} {self.K}"
#Нахождение, создание или изменения файла для критичных критериев, а так же
запись в массив крит критериев
def file_criteria():
    if os.path.exists("crit_criteria")==1:
        change=str(input("Файл с критическими критериями уже создан, желаете
```

```
его поменять?"))
        if change=="1" or change=="Да" or change=="да" or change=="Yes" or
change=="yes":
            list_criteria=create_list_criteria()
            list_crit_criteria=set_crit_criteria(list_criteria)
        else:
            print("Работаем с иходными данными")
            f_criteria = open('crit_criteria', 'r',encoding='utf-8')
            list_crit_criteria=[]
            for crit in f_criteria:
                X=criteria(crit,0,0)
                list_crit_criteria.append(X)
            f_criteria.close()
        if len(list_crit_criteria)==0:
            print("Файл с иходными данными пуст, его нужно заполнить")
            list_criteria=create_list_criteria()
            list_crit_criteria=set_crit_criteria(list_criteria)
    else:
        print("Создаем файл с критическимим критериями")
        list_criteria=create_list_criteria()
        list_crit_criteria=set_crit_criteria(list_criteria)
    return list_crit_criteria
#Нахождение или создание файла для критичных компонентов,  а так же запись в
массив крит компонентов
def file_components(name, list_crit_criteria):
    if os.path.exists(name)==1:
        change=str(input("Файл с критическими компонентами уже создан,
желаете его поменять? "))
        if change=="1" or change=="Да" or change=="да" or change=="Yes" or
change=="yes":
            list_components=create_list_components()
            list_crit_components=crit_comp(list_components,
list_crit_criteria,name)
            write_crit_comp(list_crit_components, name)
        else:
            print("Работаем с иходными данными\n")
            f_components = open(name, 'r',encoding='utf-8')
            list_crit_components=[]
            for comp in f_components:
                X=component(comp,1)
                list_crit_components.append(X)
            f_components.close()
            if len(list_crit_components)==0:
                print("Файл с иходными данными пуст, его нужно заполнить\n")
                list_components=create_list_components()
                list_crit_components=crit_comp(list_components,
list_crit_criteria,name)
                write_crit_comp(list_crit_components, name)
    else:
        print("Создаем файл с критическимим компонентами")
        list_components=create_list_components()
        list_crit_components=crit_comp(list_components,
list_crit_criteria,name)
        write_crit_comp(list_crit_components, name)
    return list_crit_components
#Считывание из файла компонентов
def create_list_components():
    f_component = open('components', 'r',encoding='utf-8')
    list_components=[]
    for comp in f_component:
        X=component(comp,0)
        list_components.append(X)
    f_component.close()
    return list_components
```

```
#Считывание из файла критериев
def create_list_criteria():
    sum_Y=0
    count_criteria=0
    list_criteria=[]
    f_criteria = open('criteria', 'r',encoding='utf-8')
    for crit in f_criteria:
        print("Рассмотрим критерий: ",crit)
        count_criteria+=1
        K_D=int(input("Задайте текущую оценку отказа критерия при отсутствии
мероприятий по управлению рисками(от 0-5, где 0 - значительный ущерб; 5 -
незначительный ущерб или ущерб отсутствует): "))
        while K_D>5 or K_D<0:
            K_D=int(input("Значение K(D) задан неверно! Измените параметр. Он
может принимать значение только 0-5. Попробуйте еще раз. "))
        K_S=int(input("Задайте желаемое значение оценки отказа критерия с
учетом возможным мероприятий по управлению рисками(от 0-5, где 0 -
значительный ущерб; 5 - незначительный ущерб или ущерб отсутствует): "))
        while K_D>5 or K_D<0:
            K_S=int(input("Значение K(S) задан неверно! Измените параметр. Он
может принимать значение только 0-5. Попробуйте еще раз. "))
        Y=int(input("Задайте относительный уровень важности критерия(сумма по
всем критериям должна равняться 100%): "))
        sum_Y+=Y
        while sum_Y>100:
            sum_Y-=Y
            Y=int(input("Сумма значений Y > 100%! Измените параметр Y "))
            sum_Y+=Y
        K=Y*(K_S-K_D)
        while K<0:
            print("Значения K(D) и K(S) заданы неверно! Измените эти
параметры")
            K_D=int(input())
            K_S=int(input())
            K=Y*(K_S-K_D)
        X=criteria(crit,Y,K)
        list_criteria.append(X)
    f_criteria.close()
    return list_criteria
#Отсечение неважных критериев
def cut_off(list, perc):
    sum_Y=0
    numb=0
    for i in range(len(list)):
        if (sum_Y+list[i].Y) <= perc:
            sum_Y+=list[i].Y
        else:
            numb=i
            break
    return list[:i]
#Выбор процента отсечения и сортировка
def set_crit_criteria(list_criteria):
    persent=int(input("Задайте процент отсечения значимых критериев: "))
    while persent>100 or persent<0:
        persent=int(input("Выбран некорректный процент! Он должен принимать
значение 1-100. Попробуйте еще раз "))
    list_criteria.sort(key=lambda x: x.K, reverse=True)
    list_crit_criteria=cut_off(list_criteria,persent)
    write_crit_criteria(list_crit_criteria)
    return list_crit_criteria
#Запись крит. критериев в файл
def write_crit_criteria(list_crit_criteria):
    f_crit_criteria = open('crit_criteria', 'w',encoding='utf-8')
    for i in range(len(list_crit_criteria)):
```

```
        f_crit_criteria.write(list_crit_criteria[i].name)
    f_crit_criteria.close()
#Оценка значимости компонентов
def evo_comp(list_com, list_cri):
    evaluation_components=[]
    for i in range(len(list_com)):
        evaluation_criteria=[list_com[i].name[:-2]]
        print("Оценка важности компонента: ",list_com[i].name)
        for j in range(len(list_cri)):
            print("Оцените значимость компонета по критерию(от 0-5, 0 -
незначительный ущерб; 5 - максимальный ущерб): ",list_cri[j].name)
            x=int(input())
            while x<0 or x>5:
                x=int(input("Некоректный ввод! Попробуйте снова "))
            evaluation_criteria.append(x)
        evaluation_components.append(evaluation_criteria)
    return evaluation_components
#Создание бинарной таблицы оценки занчимости компонентов
def bin_evaluation_component(list_com, list_cri):
    list_evaluation_components=evo_comp(list_com,list_cri)
    bin_evaluation_component=[]
    for i in range(len(list_evaluation_components)):
        bin_evaluation_component1=[]
        bin_evaluation_component1.append(list_com[i].name[:-2])
        max_evaluation_component1=max(list_evaluation_components[i][1:])
        for j in range(len(list_evaluation_components)):
            max_evaluation_component2=max(list_evaluation_components[j][1:])
            if max_evaluation_component1>max_evaluation_component2 and
all(list_evaluation_components[i][k]>=list_evaluation_components[j][k]  for k
in range(1,len(list_evaluation_components[i]))):
                bin_evaluation_component1.append(1)
            else:
                bin_evaluation_component1.append(0)
        bin_evaluation_component.append(bin_evaluation_component1)
    #print(*bin_evaluation_component)
    return bin_evaluation_component
#Определение суммы 0 для каждого компонента
def sum_bin_eva(bin_evaluation_component):
    list_sum_bin_eva=[0]*(len(bin_evaluation_component))
    list_sum_bin_eva.insert(0,"Сумма")
    for i in range(len(bin_evaluation_component)):
        for j in range(1,len(bin_evaluation_component)+1):
            if bin_evaluation_component[i][j]==0:
                list_sum_bin_eva[j]+=1
    #print(list_sum_bin_eva)
    return list_sum_bin_eva
#Выставление важности для каждого сомпанента
def set_importance(list_components,list_sum_bin_eva):
    for i in range(len(list_components)):
        list_components[i].importance=list_sum_bin_eva[i]
#Выставление № группы для исполнителя
def set_group_executor(list_components,set_sum_bin_eva):
    group=1
    for i in range(len(set_sum_bin_eva)):
        for j in range(len(list_components)):
            if list_components[j].importance==set_sum_bin_eva[i]:
                list_components[j].group_executor=group
        group+=1
#Выставление № группы для заказчика
def set_group_customer(list_components,set_sum_bin_eva):
    group=1
    for i in range(len(set_sum_bin_eva)):
        for j in range(len(list_components)):
            if list_components[j].importance==set_sum_bin_eva[i]:
```

```
                list_components[j].group_customer=group
        group+=1
#Изменение номеров групп в множестве
def change_set_group(list_components,set_sum_bin_eva):
    group=1
    for i in range(len(set_sum_bin_eva)):
        if set_sum_bin_eva[i]!=group:
            k=set_sum_bin_eva[i]-group
            set_sum_bin_eva[i]=group
        group+=1
#Вывод критичных компонентов для исполнителя
def print_crit_comp_executor(list_components,set_sum_bin_eva):
    for i in range(len(set_sum_bin_eva)):
        print("\nГруппа №",set_sum_bin_eva[i], ": ")
        for j in range(len(list_components)):
            if (set_sum_bin_eva[i])==(list_components[j].group_executor):
                print(list_components[j].name)
#Вывод критичных компонентов для заказчика
def print_crit_comp_customer(list_components,set_sum_bin_eva):
    for i in range(len(set_sum_bin_eva)):
        print("\nГруппа №",set_sum_bin_eva[i], ": ")
        for j in range(len(list_components)):
            if (set_sum_bin_eva[i])==(list_components[j].group_customer):
                print(list_components[j].name)
#Основная функция определения критических компонентов для исполнителя
def crit_comp_executor(list_components,list_crit_criteria):

list_sum_bin_eva=sum_bin_eva(bin_evaluation_component(list_components,list_cr
it_criteria))
    list_set_sum_bin_eva=sorted(set(list_sum_bin_eva[1:]))
    set_importance(list_components,list_sum_bin_eva[1:])
    set_group_executor(list_components,list_set_sum_bin_eva[::-1])
    change_set_group(list_components,list_set_sum_bin_eva)
    print("Распределение компонентов по группам для исполнителя")
    print_crit_comp_executor(list_components,list_set_sum_bin_eva)
#Основная функция определения критических компонентов для заказчика
def crit_comp_customer(list_components,list_crit_criteria):

list_sum_bin_eva=sum_bin_eva(bin_evaluation_component(list_components,list_cr
it_criteria))
    list_set_sum_bin_eva=sorted(set(list_sum_bin_eva[1:]))
    set_importance(list_components,list_sum_bin_eva[1:])
    set_group_customer(list_components,list_set_sum_bin_eva[::-1])
    change_set_group(list_components,list_set_sum_bin_eva)
    print("Распределение компонентов по группам для заказчика")
    print_crit_comp_customer(list_components,list_set_sum_bin_eva)
#Оценка согласованности выбора критичный комапнентов
def agreement_crit_comp(list_components,list_crit_criteria,name):
    persent_agree=80
    max_group=0
    for i in range(len(list_components)):

max_group_comp=max(list_components[i].group_executor,list_components[i].group
_customer)
        list_components[i].group_difference=max_group_comp-
min(list components[i].group executor,list components[i].group customer)
        max_group=max(max_group,max_group_comp)
    group_agree=(max_group*(100-persent_agree))/100
    f=0
    for i in range(len(list_components)):
        if (list_components[i].group_executor==1 or
list_components[i].group_customer==1) and
list_components[i].group_difference>group_agree :
            print(list_components[i].name)
```

```
            f=1
        elif list_components[i].group_executor==1 or
list_components[i].group_customer==1:
            list_components[i].group=1
    if f==1:
        #print("Ошибка согласования для данных компонентов, советуем более
детально обсудить данную тему между заказчиком и исполнителем!")
        #crit_comp(list_components,list_crit_criteria,name)
        sys.exit("Ошибка согласования для данных компонентов, советуем более
детально обсудить данную тему между заказчиком и исполнителем! Дальнейшая
работа невозможна!")
#Согласование по Поретто
# def agreement_crit_comp(list1, list2):
#     list_agree=[]
#     for i in range(len(list1)):
#         list_agree_comp=[]
#         list_agree_comp.append(list1[i].name)
#         for j in range(1,len(list[i])):
#             if list1[i][j]==list2[i][j]:
#                 list_agree_comp.append(1)
#             else:
#                 list_agree_comp.append(0)
#         list_agree.append(list_agree_comp)
#     return list_agree

# def check_agree(list_agree):
#     persent_agree=80
#     cnt_agree=(len(list_agree)*persent_agree)/100
#     f=0
#     for i in range(len(list_agree)):
#         sum=0
#         for j in range(1,len(list_agree[i])):
#             sum+=list_agree[i][j]
#         if sum<cnt_agree:
#             print(list_agree[i][j-1])
#             f=1
#     if f==1:
#         print("Ошибка согласования для данных компонентов, советуем более
детально обсудить данную тему между заказчиком и исполнителем!")
#     return f

#Запись критичных критериев в файл
def write_crit_comp(list_components, name):
    list_crit_components=[]
    f_crit_comp=open(name,"w", encoding='utf-8')
    for comp in list_components:
        if comp.group==1:
            f_crit_comp.write(comp.name)
            list_crit_components.append(comp)
    f_crit_comp.close()
    return list_crit_components
#Основная функция определения критичных компонентов
def crit_comp(list_components,list_crit_criteria,name):
    post=int(input("Вы являетесь исполнителем(1) или заказчиком(2) "))
    while post!=1 and post!=2:
        post=int(input("Введено неправильное значение, попробуйте еще раз "))
    if post==1:
        crit_comp_executor(list_components,list_crit_criteria)
        change=str(input("Спасибо за вашу оценку! Теперь нам нужно узнать
мнение заказчика.\nГотовы начать? "))
        if change=="1" or change=="Да" or change=="да" or change=="Yes" or
change=="yes":
            crit_comp_customer(list_components,list_crit_criteria)
        else:
```

```
            sys.exit("Для оценки соглассованности необходимо мнение
заказчика")
    elif post==2:
        crit_comp_customer(list_components,list_crit_criteria)
        change=str(input("Спасибо за вашу оценку! Теперь нам нужно узнать
мнение исполнителя.\nГотовы начать? "))
        if change=="1" or change=="Да" or change=="да" or change=="Yes" or
change=="yes":
            crit_comp_executor(list_components,list_crit_criteria)
        else:
            sys.exit("Для оценки соглассованности необходимо мнение
заказчика")
    agreement_crit_comp(list_components,list_crit_criteria,name)
    list_crit_components=write_crit_comp(list_components, name)
    return list_crit_components
#Вывод критичных компонентов
def print_crit_components(list_crit_companies):
    for i in range(len(list_crit_companies)):
        print(list_crit_companies[i].name)
#Вывод критичных критериев
def print_crit_criteria(list_crit_criteria):
    for i in range(len(list_crit_criteria)):
        print(list_crit_criteria[i].name)
#Вывод изображения
def print_image(name):
    name_model="Модель компонента: "+name
    plt.title(name_model)
    name_image="graphs/"+name+".png"
    image = mpimg.imread(name_image)
    plt.axis('off')
    plt.imshow(image)
    plt.show()
#Вывод информации о модели
def print_ifo_model_and_cnt(name):
    file_name="probability/"+name+".txt"
    f=open(file_name,'r',encoding='utf-8')
    cnt=0
    for i in f:
        print(i)
        cnt+=1
    f.close()
    return cnt
#Расчет вероятности безотказной работы
def P(numerical_solution,N,method):
    if method==1:
        sum_m=0
        for i in range(len(numerical_solution)):
            sum_m+=numerical_solution[i]
        P=numerical_solution[0]/sum_m
    elif method==2:
        P=numerical_solution[0]
    return P
#Расчет совокупного риска компонента
def R(list_C, numerical_solution, N,method):
    R=0
    if method==1:
        for i in range(1,len(list_C)):
            R+=list_C[i]*sqrt(numerical_solution[i]*(1-
(numerical_solution[i]/N)))
    elif method==2:
        for i in range(1,len(list_C)):
            R+=list_C[i]*numerical_solution[i]
    return R
#Введение C для компонента
```

```
def c_conpanent(x):
    c_conpanent=[]
    for i in range(x):
        print("Введите выгоду, которую привносит единица работоспособного
объекта(C0) или потери, которые привносит объект при нахождении в
неработоспособном состоянии(C1,C2,...). Введите значение C", i)
        c_conpanent.append(float(input()))
    return c_conpanent
#Введение l для компонента бюджетного обеспечения (budget support).
def l_component_BS():
    list_l=[]
    list_l0=[0,0]
    list_l1=[0,0]
    list_l0[1]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 1-ое. "))
    list_l1[0]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 0-ое. "))
    list_l.append(list_l0)
    list_l.append(list_l1)
    return list_l
#Введение l для компонента контрактная среда (contract environment).
def l_component_CE():
    list_l=[]
    list_l0=[0,0,0]
    list_l1=[0,0,0]
    list_l2=[0,0,0]
    list_l0[1]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 1-ое. "))
    list_l0[2]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 2-ое. "))
    list_l1[0]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 0-ое. "))
    list_l1[2]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 2-ое. "))
    list_l2[1]=float(input("Введите интенсивность перехода системы из 2-го
состояния в 1-ое. "))
    list_l.append(list_l0)
    list_l.append(list_l1)
    list_l.append(list_l2)
    return list_l
#Введение l для компонента факторы среды предприятия (environmental factors
of the enterprise).
def l_component_EFE():
    list_l=[]
    list_l0=[0,0,0]
    list_l1=[0,0,0]
    list_l2=[0,0,0]
    list_l0[1]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 1-ое. "))
    list_l0[2]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 2-ое. "))
    list_l1[2]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 2-ое. "))
    list_l2[0]=float(input("Введите интенсивность перехода системы из 2-го
состояния в 0-ое." ))
    list_l.append(list_l0)
    list_l.append(list_l1)
    list_l.append(list_l2)
    return list_l
#Введение l для компонента содержание проекта (project content).
def l_component_PC():
    list_l=[]
    list_l0=[0,0,0,0]
    list_l1=[0,0,0,0]
```

```
    list_l2=[0,0,0,0]
    list_l3=[0,0,0,0]
    list_l0[1]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 1-ое. "))
    list_l1[2]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 2-ое. "))
    list_l1[3]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 3-е. "))
    list_l2[0]=float(input("Введите интенсивность перехода системы из 2-го
состояния в 0-ое. "))
    list_l3[0]=float(input("Введите интенсивность перехода системы из 3-го
состояния в 0-ое. "))
    list_l.append(list_l0)
    list_l.append(list_l1)
    list_l.append(list_l2)
    list_l.append(list_l3)
    return list_l
#Введение l для компонента человеко-ресурсное обеспечение (human resource
support).
def l_component_HRS():
    list_l=[]
    list_l0=[0,0,0,0,0]
    list_l1=[0,0,0,0,0]
    list_l2=[0,0,0,0,0]
    list_l3=[0,0,0,0,0]
    list_l4=[0,0,0,0,0]
    list_l0[1]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 1-ое. "))
    list_l0[2]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 2-ое. "))
    list_l0[3]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 3-ое. "))
    list_l0[4]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 4-ое. "))
    list_l1[0]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 0-ое. "))
    list_l1[3]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 3-е. "))
    list_l1[4]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 4-ое. "))
    list_l2[0]=float(input("Введите интенсивность перехода системы из 2-го
состояния в 0-ое. "))
    list_l3[0]=float(input("Введите интенсивность перехода системы из 3-го
состояния в 0-ое. "))
    list_l3[1]=float(input("Введите интенсивность перехода системы из 3-го
состояния в 1-ое. "))
    list_l3[4]=float(input("Введите интенсивность перехода системы из 3-го
состояния в 4-ое. "))
    list_l4[0]=float(input("Введите интенсивность перехода системы из 4-го
состояния в 0-ое. "))
    list_l.append(list_l0)
    list_l.append(list_l1)
    list_l.append(list_l2)
    list_l.append(list_l3)
    list_l.append(list_l4)
    return list_l
#Введение l для компонента аппаратно-технический комплекс HTC(Hardware and
technical complex)..
def l_component_HTC():
    list_l=[]
    list_l0=[0,0,0,0,0]
    list_l1=[0,0,0,0,0]
    list_l2=[0,0,0,0,0]
    list_l3=[0,0,0,0,0]
```

```
    list_l4=[0,0,0,0,0]
    list_l0[1]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 1-ое. "))
    list_l0[2]=float(input("Введите интенсивность перехода системы из 0-го
состояния в 2-ое. "))
    list_l1[0]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 0-ое. "))
    list_l1[2]=float(input("Введите интенсивность перехода системы из 1-го
состояния в 2-ое. "))
    list_l2[3]=float(input("Введите интенсивность перехода системы из 2-го
состояния в 3-е. "))
    list_l2[4]=float(input("Введите интенсивность перехода системы из 2-го
состояния в 4-ое. "))
    list_l3[1]=float(input("Введите интенсивность перехода системы из 3-го
состояния в 1-ое. "))
    list_l3[4]=float(input("Введите интенсивность перехода системы из 3-го
состояния в 4-ое. "))
    list_l4[0]=float(input("Введите интенсивность перехода системы из 4-го
состояния в 0-ое. "))
    list_l.append(list_l0)
    list_l.append(list_l1)
    list_l.append(list_l2)
    list_l.append(list_l3)
    list_l.append(list_l4)
    return list_l
#В зависимости от метода мы считаем либо x=m, либо x=p. Но система не менят
вид, а меняеся только обозначение переменных

#Решение системы уравнений для компонента бюджетного обеспечения (budget
support).
def solution_component_BS(N,list_l,list_c):
    x0, x1= symbols('x0 x1')
    equations = [
        Eq(-list_l[0][1]*x0+list_l[1][0]*x1, 0),
        Eq(-list_l[1][0]*x1+list_l[0][1]*x0, 0),
        Eq(x0+x1, N)
    ]
    initial_guess = [list_c[0], list_c[0]]
    numerical_solution = nsolve(equations, (x0, x1), initial_guess)
    return numerical_solution
#Решение системы уравнений для компонента контрактная среда (contract
environment).
def solution_component_CE(N,list_l,list_c):
    x0, x1, x2= symbols('x0 x1 x2')
    equations = [
        Eq(-(list_l[0][1]+list_l[0][2])*x0+list_l[1][0]*x1, 0),
        Eq(-(list_l[1][0]+list_l[1][2])*x1+list_l[0][1]*x0+list_l[2][1]*x2,
0),
        Eq(-list_l[2][1]*x2+list_l[1][2]*x1+list_l[0][2]*x0, 0),
        Eq(x0+x1+x2, N)
    ]
    initial_guess = [list_c[0], list_c[0],list_c[0]]
    numerical_solution = nsolve(equations, (x0, x1, x2), initial_guess)
    return numerical_solution
#Решение системы уравнений для компонента факторы среды предприятия
(environmental factors of the enterprise).
def solution_component_EFE(N,list_l,list_c):
    x0, x1, x2= symbols('x0 x1 x2')
    equations = [
        Eq(-(list_l[0][1]+list_l[0][2])*x0+list_l[2][0]*x2, 0),
        Eq(-list_l[1][2]*x1+list_l[0][1]*x0, 0),
        Eq(-list_l[2][0]*x2+list_l[0][2]*x0+list_l[1][2]*x1, 0),
        Eq(x0+x1+x2, N)
    ]
```

```
    initial_guess = [list_c[0], list_c[0],list_c[0]]
    numerical_solution = nsolve(equations, (x0, x1, x2), initial_guess)
    return numerical_solution
#Решение системы уравнений для компонента содержание проекта (project
content).
def solution_component_PC(N,list_l,list_c):
    x0, x1, x2, x3= symbols('x0 x1 x2 x3')
    equations = [
        Eq(-list_l[0][1]*x0+list_l[2][0]*x2+list_l[3][0]*x3, 0),
        Eq(-(list_l[1][2]+list_l[1][3])*x1+list_l[0][1]*x0, 0),
        Eq(-list_l[2][0]*x2+list_l[1][2]*x1, 0),
        Eq(-list_l[3][0]*x3+list_l[1][3]*x1, 0),
        Eq(x0+x1+x2+x3, N)
    ]
    initial_guess = [list_c[0], list_c[0], list_c[0], list_c[0]]
    numerical_solution = nsolve(equations, (x0, x1, x2, x3), initial_guess)
    return numerical_solution
#Решение системы уравнений для компонента аппаратно-технический комплекс
HTC(Hardware and technical complex).
def solution_component_HTC(N,list_l,list_c):
    x0, x1, x2, x3, x4 = symbols('x0 x1 x2 x3 x4')
    equations = [
        Eq(-x0*(list_l[0][1]+list_l[0][2])+x1*list_l[1][0]+x4*list_l[4][0],
0),
        Eq(-x1*(list_l[1][0]+list_l[1][2])+x0*list_l[0][1]+x3*list_l[3][1],
0),
        Eq(-x2*(list_l[2][3]+list_l[2][4])+x0*list_l[0][2]+x1*list_l[1][2],
0),
        Eq(-x3*(list_l[3][1]+list_l[3][4])+x2*list_l[2][3], 0),
        Eq(-x4*list_l[4][0]+x2*list_l[2][4]+x3*list_l[3][4], 0),
        Eq(x0+x1+x2+x3+x4, N)
    ]
    initial_guess = [list_c[0], list_c[0], list_c[0], list_c[0], list_c[0]]
    numerical_solution = nsolve(equations, (x0, x1, x2, x3, x4),
initial_guess)
    return numerical_solution
#Решение системы уравнений для компонента человеко-ресурсное обеспечение
(human resource support).
def solution_component_HRS(N,list_l,list_c,):
    x0, x1, x2, x3, x4 = symbols('x0 x1 x2 x3 x4')
    equations = [
        Eq(-
(list_l[0][1]+list_l[0][2]+list_l[0][3]+list_l[0][4])*x0+x1*list_l[1][0]+x2*l
ist_l[2][0]+x3*list_l[3][0]+x4*list_l[4][0], 0),
        Eq(-
(list_l[1][0]+list_l[1][3]+list_l[1][4])*x1+x0*list_l[0][1]+x3*list_l[3][1],
0),
        Eq(-x2*list_l[2][0]+x0*list_l[0][2], 0),
        Eq(-
(list_l[3][0]+list_l[3][1]+list_l[3][4])*x3+x0*list_l[0][3]+x1*list_l[1][3],
0),
        Eq(-x4*list_l[4][0]+x0*list_l[0][4]+x1*list_l[1][4]+x3*list_l[3][4],
0),
        Eq(x0+x1+x2+x3+x4, N)
    ]
    initial_guess = [list_c[0], list_c[0], list_c[0], list_c[0], list_c[0]]
    numerical_solution = nsolve(equations, (x0, x1, x2, x3, x4),
initial_guess)
    return numerical_solution
#Проверка системы
def check_solution(method,numerical_solution):
    if method==1:
        for i in range(len(numerical_solution)):
            if not((numerical_solution[i]//1)==int(numerical_solution[i]) and
```

```
numerical_solution[i]%1==0):
                print("Система имеет дробные или отрицательные результаты.
Измените параметры l")
                return 1
            else:
                numerical_solution[i]=int(numerical_solution[i])
    elif method==2:
        for i in range(len(numerical_solution)):
            if numerical_solution[i]<0:
                print("Система имеет отрицательные результаты. Измените
параметры l")
                return 1
            else:
                numerical_solution[i]=float(numerical_solution[i])
    return 0
#Полная реализация определения P и R для компонента аппаратно-технический
комплекс HTC(Hardware and technical complex).
def component_HTC(N,list_c, method):
    list_l=l_component_HTC()
    numerical_solution=solution_component_HTC(N,list_l,list_c)
    f=check_solution(method,numerical_solution)
    if f==1:
        component_HTC(N,list_c,method)
    elif f==0:
        print("P=",P(numerical_solution,N,method)*100,'%')
        print("R=",R(list_c,numerical_solution,N,method))
#Полная реализация определения P и R для компонента бюджетного обеспечения
(budget support).
def component_BS(N,list_c, method):
    list_l=l_component_BS()
    numerical_solution=solution_component_BS(N,list_l,list_c)
    f=check_solution(method,numerical_solution)
    if f==1:
        component_BS(N,list_c,method)
    elif f==0:
        print("P=",P(numerical_solution,N,method)*100,'%')
        print("R=",R(list_c,numerical_solution,N,method))
#Полная реализация определения P и R для компонента контрактная среда
(contract environment).
def component_CE(N,list_c, method):
    list_l=l_component_CE()
    numerical_solution=solution_component_CE(N,list_l,list_c)
    f=check_solution(method,numerical_solution)
    if f==1:
        component_CE(N,list_c,method)
    elif f==0:
        print("P=",P(numerical_solution,N,method)*100,'%')
        print("R=",R(list_c,numerical_solution,N,method))
#Полная реализация определения P и R  для компонента факторы среды
предприятия (environmental factors of the enterprise).
def component_EFE(N,list_c, method):
    list_l=l_component_EFE()
    numerical_solution=solution_component_EFE(N,list_l,list_c)
    f=check_solution(method,numerical_solution)
    if f==1:
        component EFE(N,list c,method)
    elif f==0:
        print("P=",P(numerical_solution,N,method)*100,'%')
        print("R=",R(list_c,numerical_solution,N,method))
#Полная реализация определения P и R компонента человеко-ресурсное
обеспечение (human resource support).
def component_HRS(N,list_c, method):
    list_l=l_component_HRS()
    numerical_solution=solution_component_HRS(N,list_l,list_c)
```

```
    f=check_solution(method,numerical_solution)
    if f==1:
        component_HRS(N,list_c,method)
    elif f==0:
        print("P=",P(numerical_solution,N,method)*100,'%')
        print("R=",R(list_c,numerical_solution,N,method))
#Реализация определения P и R компонентов ПО
def component_PO():
    l=float(input("Введите l "))
    M=float(input("Введите M "))
    m=int(input("Введите m "))
    n=int(input("Введите n "))
    C1=float(input("Введите C1 "))
    C2=float(input("Введите C2 "))
    p=l/M
    p0=P0(m,n,p)
    print("P=",p0)
    ls=Ls(m,n,p)
    R=ls*C1+Lq(m,ls)*C2
    print("R=",R)
def P0(m,n,p):
    x1=((factorial(m))/(factorial(m-n)))*(p**n)
    x2=0
    for k in range(0,n+1):
        x2+=((factorial(m))/(factorial(m-k)))*(p**k)
    P0=x1/x2
    # x1=0
    # for k in range(0,n+1):
    #     x1+=((n*p*(1/n))**k)/factorial(k)
    # P0=1/x1
    return P0
def Ls(m,n,p):
    x=0
    for k in range(0,n):
        x+=((factorial(m)/factorial(m-k-1))*(p**k))
    Ls=m-(1/G(m,n,p))*x
    return Ls
def G(m,n,p):
    G=0
    for k in range(0,n-1):
        G+=((factorial(m)/factorial(m-k))*(p**k))
    return G
def Lq(m,Ls):
    Lq=m-Ls
    return Lq
#Определение вызываемой функции по названию компонента
def component_name(comp,N,list_c,method):
    match comp:
        case "Методическая и правовая часть проекта":
            component_EFE(N,list_c,method)
        case "Бюджетное обеспечение проекта":
            component_BS(N,list_c,method)
        case "Активы процессов организации проекта":
            component_EFE(N,list_c,method)
        case "Факторы среды предприятия":
            component EFE(N,list c,method)
        case "Контрактная среда проекта":
            component_CE(N,list_c,method)
        case "Содержание проекта":
            component_PC(N,list_c,method)
        case "Человеко-ресурсное обеспечение":
            component_HRS(N,list_c,method)
        case "Организационное обеспечение АС":
            component_EFE(N,list_c,method)
```

```
        case "Техническое обеспечение АС":
            component_HTC(N,list_c,method)
#Расчет надежности и риска
def assessment(list_crit_components):
    for i in range(len(list_crit_components)):
        N = 0
        comp = list_crit_components[i].name[:-1]
        print_image(comp)
        cnt = print_ifo_model_and_cnt(comp)
        if comp == "Специализированное программное обеспечение (СПО)" or
comp=="Общее программное обеспечение (ОПО)" or comp=="Программное обеспечение
для разработки и тестирования":
            component_PO()
        else:
            if comp == "Бюджетное обеспечение проекта":
                N = 1
            else:
                print("Выберите метод рассчета: 1-метод динамики средних, 2-
классический метод")
                while True:
                    method = int(input())
                    if method == 1:
                        N = int(input("Введите N (Нормируещее условие) "))
                        break
                    elif method == 2:
                        N = 1
                        break
                    else:
                        print("Неверный ввод. Попробейте снова")
            list_c = c_conpanent(cnt)
            component_name(comp, N, list_c, method)
#Основная функция
def main():
    #component_PO()
    list_crit_criteria = file_criteria()
    print("Желаете посмотреть критические критерии?")
    change=str(input())
    if change=="1" or change=="Да" or change=="да" or change=="Yes" or
change=="yes":
        print_crit_criteria(list_crit_criteria)

    print("На какой фазе проекта вы хотите определить критичные
компоненты(назовите номер)?")
    phase = int(input())
    name = "crit_comp_phase№" + str(phase)

    list_crit_components = file_components(name, list_crit_criteria)
    print("Желаете посмотреть критичные компоненты?")
    change=str(input())
    if change=="1" or change=="Да" or change=="да" or change=="Yes" or
change=="yes":
        print_crit_components(list_crit_components)

    print("Желаете рассчитать риск и надежность критичных компонентов?")
    change = str(input())
    if change == "1" or change == "Да" or change == "да" or change == "Yes"
or change == "yes":
        assessment(list_crit_components)
    print("Всего хорошего ;)")

if __name__ == "__main__":
    main()
```

Анализ результатов

```
Качество технической и сопроводительной документации на АС.
Соответствие процессов реализации и артефактов формальным активам процессов организаци
Соответствие системы/проекта правовым нормам и стандартам
Взаимное соответствие системы/проекта контрактным условиям и обязательствам
Качество организации взаимодействия заинтересованных сторон и проекта/системы
Удобство (эргономика) системы
Качество реализованных функциональных требований системы
Качество реализованных внутренних нефункциональных требований системы
Качество реализованных внешних нефункциональных требований системы
Адаптивность системы
Качество технической поддержки системы
Качество оборудования АС
Затраты на реализацию системы/проекта
Прямые выгоды от реализации АСУП
Косвенные выгоды от реализации АСУП
Сроки реализации системы/проекта
Общая безопасность АСУП или компании (в т.ч. информационная и юридическая)
Влияние на репутацию компании (в т.ч. внутренняя репутация)
Внешние проекты и системы
Влияние на аспекты экологии и устойчивого развития
Влияние на социальные и кадровые аспекты деятельности предприятия
```

```
1  Качество технической и сопроводительной документации на АС.
2  Соответствие процессов реализации и артефактов формальным активам процессов организации.
3  Соответствие системы/проекта правовым нормам и стандартам
4
```

```
1  Методическая и правовая часть проекта
2  Бюджетное обеспечение проекта
3  Активы процессов организации проекта
4  Факторы среды предприятия
5  Контрактная среда проекта
6  Содержание проекта
7  Человеко-ресурсное обеспечение
8  Организационное обеспечение АС
9  Аппаратно-технический комплекс
10 Специализированное программное обеспечение (СПО)
11 Общее программное обеспечение (ОПО)
12 Программное обеспечение для разработки и тестирования
```

```
1  Человеко-ресурсное обеспечение
2
```