**Поволжский Государственный Университет Телекоммуникаций и Информатики**

**Кафедра ПОУТС**

Отчет по лабораторной работе №3­­­­

«Технико-экономическое обоснование договорной цены на разработку прикладного программного обеспечения»

Вариант №6

Выполнили: студенты группы ПО-61

Булычев Иван

Проверила: Вержаковская М. А.

Самара 2020

**A screenshot of a cell phone

Description automatically generated**

**Исходные данные для расчетов:**

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Для расчета были взяты данные для варианта 6.

A screenshot of a cell phone

Description automatically generated

**Результаты работы**

Описание файлов, содержащих данные для расчета:

**Файл params.properties:**

Заполняется параметрами для расчета модели по варианту.

Изменение порядка параметров приведет к некорректным расчетам.

**Содержание params.properties:**

Язык\_программирования=10

Срок\_разработки\_(мес.)=16

Размерность\_системы\_определенная\_экспертами=4000

БД\_N=12

БД\_K1=15

БД\_M=11

Количество\_функциональных\_точек=1000

V\_коэффициент\_внешней\_среды=65

Ставка\_программиста(руб)=24000

Отношение\_ставки\_программиста\_к\_системному\_аналитику=0.769230769230769

Отношение\_ставки\_программиста\_к\_техническому\_специалисту=1.428571428571429

Нормотив\_трудоемкости\_опытной\_эксплуатации=0.0095

**Файл languages.csv**

<№> <название языка программирования> <Ассемблер (LOC)> <Показатель LOC на 1 функциональную точку>

Номер должен быть уникальным.

**Файл life\_circle\_stages.csv**

Состоит из таблиц 1.6 и 1.8 Распределение специалистов %

<№> <название этапа жизненного цикла> <Трудозатраты> <Длительность> <Аналитики> <Программисты> <Технические специалисты>

Примеры вышеописанных файлов можно найти по ссылке в приложении.

**Вариант 6**

**Результат работы программы на Python:**

Введите название файла с языками (enter, для выбора по умолчанию - /Users/dstrmv/PycharmProjects/softwareEconomics/src/lab3/languages.csv):>?

л с параметрами для расчета (по умолчанию - /Users/dstrmv/PycharmProjects/softwareEconomics/src/lab3/params.properties):>?

Файл с нормативами трудоемкости разработки ПС относительно размера БД (по умолчанию - /Users/dstrmv/PycharmProjects/softwareEconomics/src/lab3/labor\_categories\_db.csv):>?

л с распределением трудозатрат по жизненным циклам ПС (по умолчанию - /Users/dstrmv/PycharmProjects/softwareEconomics/src/lab3/life\_circle\_stage.csv):>?

Выбранный тип программной системы: ИСС

1.1 Прямой метод определения технико-экономических показателей (метод экспертных оценок)

Норматив производительности труда (строк/человеко-месяц) P = 220

Трудозатраты на разработку системы: 18.181818181818183

Средняя численность специалистов: Z = 2

1.2 Метод определения ТЭП проекта на основе размерности БД программной системы.

Размерность базы данных R = 198000 полей

Норматив трудоемкости разработки ПС: [ 90000 , 200000 ] ϴ‎ = 0.00808

Трудозатраты: 15.9984

Средняя численность специалистов: 1

1.3 Определение технико-экономических показателей функциональных точек

Влияние факторов внешней среды на общее кол-во функциональных точек W = 1.3

Уточненное кол-во функ. точек с учетом факторов внешней среды R(F) = 1300.0

Размерность ПО для Access : 49400.0

Трудозатраты (выбрана ИСС): 19.720330604681727

Средняя численность специалистов: 2

Таблица 1.6 – Распределение тредозатрат и длительности по основным этапам жизненного цикла создания программных систем

Метод Трудозатраты (чм) Деятельность (мес) Исполнителей (чел)

-------------------------------- ------------------- -------------------- --------------------

Прямой метод (экспертных оценок) 18.1818 16 2

На основе размерности БД 15.9984 16 1

Функциональных точек 19.7203 16 2

1.4 Пределение стоимости (договорной цены) на создание ПС

Выбираем исходные данные, полученные при помощи метода: На основе размерности БД 15.9984 1 как наименее затратные

Таблица 1.7 Расчет средней численности сотрудников

Этапы жизненного цикла Численность Zi(чел) Длительность, месяцов Дi

------------------------------------------------- --------------------- --------------------------

Анализ предметной области и разработка требований 255.974 1.6

Проектирование 187.715 4.8

Программирование 296.199 5.6

Тестирование и коплексные испытания 281.572 4

Таблица 1.9 Расчет численности специалистов по этапам жизненного цикла

Этапы жизненного цикла Аналитики Программисты Технические специалисты

------------------------------------------------- ----------- -------------- -------------------------

Анализ предметной области и разработка требований 103 52 103

Проектирование 66 66 57

Программирование 30 193 75

Тестирование и коплексные испытания 43 169 71

Таблица 1.10 Распределение фонда заработной платы по этапам жизненного цикла ПС

Этапы жизненного цикла Аналитики Программисты Техник ФЗП по этапу

------------------------------------------------- ----------- -------------- ----------- --------------

Анализ предметной области и разработка требований 3042461.54 1996800.00 5650285.71 10689547.25

Проектирование 5848615.38 7603200.00 9380571.43 22832386.81

Программирование 3101538.46 25939200.00 14400000.00 43440738.46

Тестирование и коплексные испытания 3175384.62 16224000.00 9737142.86 29136527.47

Фонд оплаты труда на разработку и коплексные испытания системы составляет: 106099200.0 рублей

Срок опытной эксплуатации: 8.0

Численность сотрудников, привлекаемых к опытной эксплуатации: 0.076

Фонд зарплаты сотрудников, привлекаемых к опытной эксплуатации: 163200.0

Общий фонд зарплаты на разработку и внедрение системы составляет: 106262400.0

Таблица 1.11 Смета затрат на разработку и вндрение системы

Наименование статей расходов Сумма (руб)

----------------------------------------------------- -------------

Фонд оплаты труда (ФОТ) 106262400.00

Страховые взносы в ПФР, ФСС, ФОМС, (30%) от ФОТ 31878720.00

Увеличение стоимости основных средств (Notebook) 20000.00

Комунальные услуги и услуги связи (телефон, Интернет) 16000.00

Прочие расходы 8000.00

Итого прямые затраты: 138185120.00

Фонд развития производства (10% от прямых затрат) 13818512.00

Накладные расходы (12 % от прямых затрат) 16582214.40

Всего расходов 168585846.40

НДС (18 % от общей стоимости) 30345452.35

ИТОГО ДОГОВОРНАЯ ЦЕНА 198931298.75

**Приложение**

Актуальные версии исходного кода и исполняемых файлов можно посмотреть по [ссылке](https://github.com/MaximKozlyuk/SoftwareEconomicsPSUTI).

Листинг программы lab3.py

import csv

import math

from tabulate import tabulate

import sys, os

if getattr(sys, 'frozen', False):

# If the application is run as a bundle, the pyInstaller bootloader

# extends the sys module by a flag frozen=True and sets the app

# path into variable \_MEIPASS'.

application\_path = sys.\_MEIPASS

else:

application\_path = os.path.dirname(os.path.abspath(\_\_file\_\_))

application\_path = os.path.dirname(sys.argv[0])

# Обьявление классов и функций

class DefaultPath(object):

def \_\_init\_\_(self, file\_name, default\_path) -> None:

self.default\_path = default\_path

self.file\_name = self.default\_path

if len(file\_name) != 0 and file\_name != "":

self.file\_name = file\_name

super().\_\_init\_\_()

class PropertiesFile(DefaultPath):

default\_properties\_path = application\_path + os.path.sep + "params.properties"

def \_\_init\_\_(self, file\_name) -> None:

super().\_\_init\_\_(file\_name, self.default\_properties\_path)

def read\_properties(self):

p = []

with open(self.file\_name, 'r') as file:

for row in file:

parts = row.split("=")

p.append((parts[0], float(parts[1])))

return p

class LanguagesFile(DefaultPath):

# todo переделать в ./languages.csv протестить работоспособность в виде exe-шника

default\_languages\_path = application\_path + os.path.sep + "languages.csv"

def \_\_init\_\_(self, file\_name) -> None:

self.languages = []

super().\_\_init\_\_(file\_name, self.default\_languages\_path)

def read\_languages(self):

l = []

with open(self.file\_name) as csv\_file:

r = csv.reader(csv\_file)

for row in r:

l.append(Language(int(row[0]), row[1], float(row[2]), float(row[3])))

self.languages = l

return l

def language\_by\_id(self, lang\_id):

for l in self.languages:

if l.lang\_id == lang\_id:

return l

return None

class Language(object):

def \_\_init\_\_(self, lang\_id, name, asm\_loc, loc) -> None:

self.lang\_id = lang\_id

self.name = name

self.asm\_loc = asm\_loc

self.loc = loc

super().\_\_init\_\_()

def \_\_str\_\_(self) -> str:

return self.\_\_class\_\_.\_\_name\_\_ + " " + str(self.lang\_id) + " " + self.name \

+ " " + str(self.asm\_loc) + " " + str(self.loc)

class LaborCategoriesDB(DefaultPath):

default\_labor\_categories\_path = application\_path + os.path.sep + "labor\_categories\_db.csv"

def \_\_init\_\_(self, file\_name) -> None:

self.categories = []

self.size = 0

super().\_\_init\_\_(file\_name, self.default\_labor\_categories\_path)

def labor\_standard\_by\_size(self, db\_size):

for i in self.categories:

if i[0] <= db\_size < i[1]:

return i

return None

def read\_categories(self):

self.categories = []

with open(self.file\_name) as csv\_file:

r = csv.reader(csv\_file)

for row in r:

self.categories.append(

(int(row[0]), int(row[1]), float(row[2]))

)

class Method(object):

all = []

allNames = []

def \_\_init\_\_(self, name, T, Z) -> None:

self.name = name

self.T = T

self.Z = Z

Method.all.append(self)

Method.allNames.append(name)

super().\_\_init\_\_()

@staticmethod

def optimal():

opt = Method.all[0].T

for i in Method.all:

if i.T < opt:

opt = i

return opt

def \_\_str\_\_(self) -> str:

return self.name + " " + str(round(self.T, 4)) + " " + str(math.ceil(self.Z))

class LifeCircleStage(object):

# alpha - трудозатраты, beta - длительность

def \_\_init\_\_(self, lf\_id, name, alpha, beta, analyst, programmers, tech\_stuff) -> None:

self.lf\_id = lf\_id

self.name = name

self.alpha = alpha

self.beta = beta

# процентное распределение специалистов по этапам жизненного цикла

self.analyst = analyst

self.programmers = programmers

self.tech\_stuff = tech\_stuff

super().\_\_init\_\_()

def \_\_str\_\_(self) -> str:

return "Этап ЖЦ " + str(self.lf\_id) + " " + self.name + " a = " + str(self.alpha) + " b = " + str(self.beta) + \

" " + str(self.analyst) + " " + str(self.programmers) + " " + str(self.tech\_stuff)

class LifeCircleStagesFile(DefaultPath):

default\_lc\_path = application\_path + os.path.sep + "life\_circle\_stage.csv"

def \_\_init\_\_(self, file\_name) -> None:

self.lc\_stages = []

self.emp\_by\_lc = []

self.z\_i = []

self.d\_i = []

self.the\_salary\_fund = []

super().\_\_init\_\_(file\_name, self.default\_lc\_path)

def read\_life\_circles(self):

self.lc\_stages = []

with open(self.file\_name, encoding="UTF8") as csv\_file:

r = csv.reader(csv\_file)

for row in r:

self.lc\_stages.append(

LifeCircleStage(int(row[0]), row[1], float(row[2]), float(row[3]),

float(row[4]), float(row[5]), float(row[6]))

)

# Расчитывает и печатает таблицу 1.7

def avg\_emp\_amount(self, T, D):

headers = ["Этапы жизненного цикла", "Численность Zi(чел)", "Длительность, месяцов Дi"]

self.z\_i = []

self.d\_i = []

rows = [headers]

for stage in self.lc\_stages:

self.z\_i.append(math.ceil(stage.alpha \* T / stage.beta \* D))

self.d\_i.append(stage.beta \* D)

rows.append([stage.name, self.z\_i[-1], self.d\_i[-1]])

print(tabulate(rows, headers="firstrow"))

return self.z\_i, self.d\_i

# Расчитывает и печатает таблицу 1.9

def emp\_amount\_by\_life\_circle(self):

self.emp\_by\_lc = []

headers = ["Этапы жизненного цикла", "Аналитики", "Программисты", "Технические специалисты"]

rows = [headers]

for i in range(len(self.lc\_stages)):

self.emp\_by\_lc.append(

[math.ceil(self.lc\_stages[i].analyst \* self.z\_i[i]),

math.ceil(self.lc\_stages[i].programmers \* self.z\_i[i]),

math.ceil(self.lc\_stages[i].tech\_stuff \* self.z\_i[i])]

)

row = [self.lc\_stages[i].name]

row.extend(self.emp\_by\_lc[i])

rows.append(row)

print(tabulate(rows, headers="firstrow"))

# Расчитывает и печатает таблицу 1.10

def salary\_fund(self, programmer\_rate\_, analyst\_ratio, tech\_ratio):

self.the\_salary\_fund = []

headers = ["Этапы жизненного цикла", "Аналитики", "Программисты", "Техник", "ФЗП по этапу"]

rows = [headers]

# определяем месячные ставки:

analyst\_rate = programmer\_rate\_ \* analyst\_ratio

tech\_rate = programmer\_rate \* tech\_ratio

# считаем зарплаты

all\_salary\_sum = 0.0

for i in range(len(self.lc\_stages)):

row = [self.lc\_stages[i].name] # добавляем название этапа жц

self.the\_salary\_fund.append([

analyst\_rate \* self.d\_i[i] \* self.emp\_by\_lc[i][0],

programmer\_rate \* self.d\_i[i] \* self.emp\_by\_lc[i][1],

tech\_rate \* self.d\_i[i] \* self.emp\_by\_lc[i][2]

])

row.extend(self.the\_salary\_fund[-1]) # добавляем расчитанные зп на этапе

stage\_sum = sum(self.the\_salary\_fund[-1])

all\_salary\_sum += stage\_sum

self.the\_salary\_fund.append(stage\_sum)

row.append(stage\_sum) # сумма ФЗП

rows.append(row)

print(tabulate(rows, headers="firstrow", floatfmt=".2f"))

return all\_salary\_sum

# Вывод общей сметы

def estimate(total\_found\_):

total\_project\_cost = 0.0

estimate\_headers = ["Наименование статей расходов", "Сумма (руб)"]

rows = [estimate\_headers]

rows.append(["Фон оплаты труда (ФОТ)", total\_found\_])

total\_project\_cost += total\_found\_

rows.append(["Страховые взносы в ПФР, ФСС, ФОМС, (30%) от ФОТ", total\_found\_ \* 0.3])

total\_project\_cost += total\_found\_ \* 0.3

rows.append(["Увеличение стоимости основных средств (Notebook)", 20000.0])

total\_project\_cost += 20000.0

rows.append(["Комунальные услуги и услуги связи (телефон, Интернет)", 1000 \* deadline])

total\_project\_cost += 1000 \* deadline

rows.append(["Прочие расходы", 500.0 \* deadline])

total\_project\_cost += 500.0 \* deadline

direct\_spent = total\_project\_cost

rows.append(["Итого прямые затраты:", total\_project\_cost])

rows.append(["Фонд развития производства (10% от прямых затрат)", direct\_spent \* 0.1])

total\_project\_cost += direct\_spent \* 0.1

rows.append(["Накладные расходы (12 % от прямых затрат)", direct\_spent \* 0.12])

total\_project\_cost += direct\_spent \* 0.12

rows.append(["Всего расходов", total\_project\_cost])

rows.append(["НДС (18 % от общей стоимости)", total\_project\_cost \* 0.18])

total\_project\_cost += total\_project\_cost \* 0.18

rows.append(["ИТОГО ДОГОВОРНАЯ ЦЕНА", total\_project\_cost])

print(tabulate(rows, headers="firstrow", floatfmt=".2f"))

# Инициализация параметров

languages = LanguagesFile(

input("Введите название файла с языками (enter, для выбора по умолчанию - "

+ LanguagesFile.default\_languages\_path + "):")

)

languages.read\_languages()

propertiesReading = PropertiesFile(

input("Файл с параметрами для расчета (по умолчанию - "

+ PropertiesFile.default\_properties\_path + "):")

)

properties = propertiesReading.read\_properties()

laborCategoriesDB = LaborCategoriesDB(

input("Файл с нормативами трудоемкости разработки ПС относительно размера БД (по умолчанию - "

+ LaborCategoriesDB.default\_labor\_categories\_path + "):")

)

laborCategoriesDB.read\_categories()

lifeCircleStagesFile = LifeCircleStagesFile(

input("Файл с распределением трудозатрат по жизненным циклам ПС (по умолчанию - "

+ LifeCircleStagesFile.default\_lc\_path + "):")

)

lifeCircleStagesFile.read\_life\_circles()

# Язык программирования

language = languages.language\_by\_id(properties[0][1])

# Срок разработки Д (мес.)

deadline = properties[1][1]

# Размерность системы определенная экспертами

system\_size = properties[2][1]

# БД - N

N = properties[3][1]

# БД - K1

K1 = properties[4][1]

# БД - M

M = properties[5][1]

# Кол-во функциональных точек

func\_points = properties[6][1]

# V - коэфф. внешней среды

V = properties[7][1]

# Ставка программиста (руб.)

programmer\_rate = properties[8][1]

# Отношения ставки программиста к:

# системному аналитику

programmerToAnalystSalary = float(properties[9][1])

# техническому специалисту

programmerToTechStuff = float(properties[10][1])

# норматив трудоемкости, при проведении опытной эксплуатации

testing\_labor\_standard = float(properties[11][1])

print("Выбранный тип программной системы: ИСС")

# 1.1

print("\n1.1 Прямой метод определения технико-экономических показателей (метод экспертных оценок)")

P = 0.0

if system\_size < 30000:

P = 220

else:

P = 160

print("Норматив производительности труда (строк/человеко-месяц) P =", P)

T\_1 = system\_size / P

print("Трудозатраты на разработку системы:", T\_1)

Z\_1 = T\_1 / deadline

print("Средняя численность специалистов: Z =", math.ceil(Z\_1))

method1 = Method("Прямой метод (экспертных оценок)", T\_1, Z\_1)

# 1.2

print("\n1.2 Метод определения ТЭП проекта на основе размерности БД программной системы.")

R = 2 \* N \* 5 \* K1 \* 10 \* M

print("Размерность базы данных R =", int(math.ceil(R)), "полей")

laborCategory = laborCategoriesDB.labor\_standard\_by\_size(R)

print("Норматив трудоемкости разработки ПС: [", laborCategory[0], ",", laborCategory[1], "] ϴ‎ =", laborCategory[2])

T\_2 = 0.01 \* R \* laborCategory[2]

print("Трудозатраты: ", T\_2)

Z\_2 = T\_2 / deadline

print("Средняя численность специалистов:", math.ceil(Z\_2))

method2 = Method("На основе размерности БД", T\_2, Z\_2)

# 1.3

print("\n1.3 Определение технико-экономических показателей функциональных точек")

W = 0.65 + (0.01 \* V)

print("Влияние факторов внешней среды на общее кол-во функциональных точек W =", W)

Rf = func\_points \* W

print("Уточненное кол-во функ. точек с учетом факторов внешней среды R(F) =", Rf)

R\_LOC = Rf \* language.loc

print("Размерность ПО для", language.name, ":", R\_LOC)

COCOMO\_A = 3

COCOMO\_E = 1.12

T\_3 = COCOMO\_A \* ((R\_LOC / 1000) \*\* COCOMO\_E) / 12

print("Трудозатраты (выбрана ИСС): ", T\_3)

Z\_3 = T\_3 / deadline

print("Средняя численность специалистов:", math.ceil(Z\_3), "\n")

method3 = Method("Функциональных точек", T\_3, Z\_3)

print("Выводы")

table = [

["Метод", "Трудозатраты (чм)", "Деятельность (мес)", "Исполнителей (чел)"],

[Method.allNames[0], T\_1, deadline, math.ceil(Z\_1)],

[Method.allNames[1], T\_2, deadline, math.ceil(Z\_2)],

[Method.allNames[2], T\_3, deadline, math.ceil(Z\_3)]

]

print(tabulate(table, headers="firstrow"))

optMethod = Method.optimal()

print("\n1.4 Пределение стоимости (договорной цены) на создание ПС")

print("Выбираем исходные данные, полученные при помощи метода:",

optMethod, "как наименее затратные")

print("\nТаблица 1.7 Расчет средней численности сотрудников")

Zi, Di = lifeCircleStagesFile.avg\_emp\_amount(optMethod.T, deadline)

print("\nТаблица 1.9 Расчет численности специалистов по этапам жизненного цикла")

lifeCircleStagesFile.emp\_amount\_by\_life\_circle()

print("\nТаблица 1.10 Распределение фонда заработной платы по этапам жизненного цикла ПС")

dev\_found = round(lifeCircleStagesFile.salary\_fund(programmer\_rate, programmerToAnalystSalary, programmerToTechStuff), 2)

print("\nФонд оплаты труда на разработку и коплексные испытания системы составляет:", dev\_found, "рублей")

# 1.4.2

test\_time = deadline / 2

print("Срок опытной эксплуатации:", test\_time)

testers\_amount = testing\_labor\_standard \* test\_time

# todo сильно меньше единицы

print("Численность сотрудников, привлекаемых к опытной эксплуатации:", testers\_amount)

testers\_salary\_found = round((math.ceil(testers\_amount) \* test\_time \* programmer\_rate \* 0.85), 2)

print("Фонд зарплаты сотрудников, привлекаемых к опытной эксплуатации:", testers\_salary\_found)

total\_found = dev\_found + testers\_salary\_found

print("Общий фонд зарплаты на разработку и внедрение системы составляет:", total\_found)

print("\nТаблица 1.11 Смета затрат на разработку и вндрение системы")

estimate(total\_found)

input('\nPress ENTER to exit')

exit(0)