Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №7 по курсу «Экономика программной инженерии»

Тема Оценка параметров программного проекта с использованием модели COCOMO II

Студент Гурова Н. А., Малышев И. Н., Николаев С. С.

Группа ИУ7-84Б

Оценка (баллы)

Преподаватель Барышникова М. Ю., Силантьева А. В.

Москва — 2024 г.

Модель композиции приложения в COCOMO II

Описание модели

Для оценки стоимости проекта на ранней стадии конструирования (создание прототипов, макетирование пользовательских интерфейсов, анализ осуществимости, оценка производительности, определение степени зрелости технологии) используется модель композиции приложения. Для такой модели характерны грубые входные данные, оценки низкой точности, приблизительные требования и архитектура на уровне концепции.

В качестве единицы измерения используется **объектная точка** — средство косвенного измерения программного обеспечения. Определение числа объектных точек производится с учетом количества экранов (как элементов пользовательского интерфейса), отчетов и компонентов, требуемых для построения приложения, в соответствии со следующими правилами:

— простые экранные формы принимаются за одну объектную точку, умеренной сложности — за две объектные точки, сложные — за три объектные точки;

— простые отчеты принимаются за две объектные точки, умеренной сложности — за пять объектных точек, сложные — за восемь объектных точек;

— модули, написанные на языках программирования третьего поколения, считаются за десять объектных точек.

Новые объектные точки NOP определяются по следующей формуле:

$$\text{NOP} = \text{Объектные точки} \cdot \frac{(100 - \%\text{RUSE})}{100},$$

где %RUSE — процент повторного использования кода программы.

Трудозатраты вычисляются так:

$$\text{Трудозатраты} = \frac{\text{NOP}}{\text{PROD}},$$

где PROD — оценка скорости разработки, определяемая из опытности или возможности разработчика или команды или из зрелости или возможности среды разработки.

Время разработки получается в соответствии с формулой:

$$\text{Время} = 3 \cdot \text{Трудозатраты}^{0.33+0.2\cdot(\text{p}-1.01)},$$

где p — показатель степени. Значение показателя степени рассчитывается с учетом факторов, влияющих на показатель степени:

$$\text{p} = \frac{(\text{PREC} + \text{FLEX} + \text{RESL} + \text{TEAM} + \text{PMAT})}{100} + 1.01.$$

Применение

Из макета интерфейса:

— для страницы «Авторизация»:

  — одна форма средней сложности;

— для страницы «Биржевые сводки»:

  — одна сложная форма (таблица биржевых сводок);

  — одна форма средней сложности (форма ввода);

— для страницы «Заявки»:

  — одна форма средней сложности (таблица заявок);

  — одна простая форма (кнопки «Удалить» и «Изменить»);

— для страницы «Новая заявка»:

  — одна форма средней сложности (форма добавления заявки).

Итого:

— 1 простая форма;

— 4 формы средней сложности;

— 1 сложная форма.

Разработанное приложение состоит из трех компонентов: первый компонент написан на SQL, второй — на C#, а третий — на Java. Поэтому в проекте два модуля, написанных на языках третьего поколения (C# и Java).

Из условия задания:

— %RUSE = 0 %;

— опытность команды — низкая.

Факторы, влияющие на показатель степени

Из условия задания:

— у отдельных членов команды имеется некоторый опыт создания систем подобного типа, поэтому новизна проекта (PREC) — наличие некоторого количества прецедентов;

— заказчик не настаивает на жесткой регламентации процесса, однако график реализации проекта довольно жесткий, поэтому гибкость процесса разработки (FLEX) — большей частью согласованный процесс;

— анализу архитектурных рисков было уделено лишь некоторое внимание, поэтому разрешение рисков в архитектуре системы (RESL) — некоторое (40 %);

— в целях сплочения команды были проведены определенные мероприятия, что обеспечило на старте проекта приемлемую коммуникацию внутри коллектива, поэтому сплоченность команды (TEAM) — некоторая согласованность;

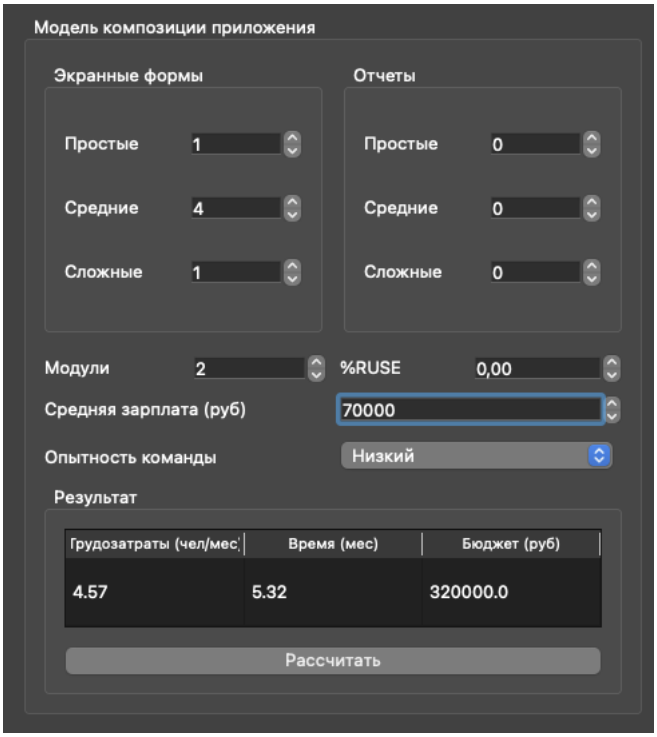
— организация только начинает внедрять методы управления проектами и формальные методы оценки качества процесса разработки, поэтому уровень зрелости процесса разработки (PMAT) — уровень 1 CMM.

Тогда имеем:

$$p = \frac{(3.72 + 2.03 + 5.65 + 3.29 + 7)}{100} + 1.01 = 1.2269.$$

Результат

На рисунке показана оценка трудозатрат и длительности разработки с использованием модели композиции приложения.



Средняя численность команды определяется по следующей формуле:

$$\text{Численность команды} = \frac{\text{Трудозатраты}}{\text{Время}} = \frac{4.57}{5.29} = 1 \text{ работник}.$$

Предварительная оценка бюджета для средней зарплаты 70 000 рублей проводится по следующей формуле:

$$\text{Бюджет} = \text{Трудозатраты} \cdot \text{Средняя зарплата} = 4.57 \cdot 70000 = 320000 \text{ рублей}.$$

Метод функциональных точек

Описание

Функциональная точка — это единица измерения функциональности программного обеспечения.

Пользователи — это отправители и целевые получатели данных, ими могут быть как реальные люди, так и смежные интегрированные информационные системы.

Функциональные типы — логических группы взаимосвязанных данных, используемых и поддерживаемых приложением:

- внешний ввод (EI) — транзакция получения данных от пользователя;
- внешний вывод (EO) — транзакция перечи данных пользователю;
- внешний запрос (EQ) — интерактивный диалог с пользователем, требующий от него каких-либо действий и не связанный с вычислением производных данных или обновлением внутренних логических файлов (базы данных);
- внутренний логический файл (ILF) — информация, которая используется во внутренних взаимодействиях системы;
- внешний интерфейсный файл (EIF) —файлы, участвующие во внешних взаимодействиях с другими системами.

Для оценки сложности функциональных типов используются следующие характеристики (их количество):

- DET (data element type) — это уникальное распознаваемое пользователем, неповторяющееся поле данных;
- RET (record element type) — идентифицируемая пользователем логическая группа данных внутри ILF или EIF;
- FTR (file type referenced) — это тип файла, на который ссылается транзакция.

Оценка числа функциональных точек происходит по алгоритму:

1. определение функциональных типов приложения и их количества;
2. определение количества связанных с каждым функциональным типом элементарных данных (DET), элементарных записей (RET) и файлов типа ссылок (FTR)
3. оценка сложности каждого типа по соответствующим таблицам и определение количества функциональных типов каждой сложности;
4. подсчет количества функциональных точек приложения, путем умножения числа количества функциональных типов каждой сложности на соответстсвующие коэффициенты и сложения полученных результатов;
5. проведение корректировки числа функциональных точек с учетом общих характеристик системы в соответствии с формулой:

$$\mathrm{AFP} = \mathrm{FP} \cdot (0.65 + 0.01 \cdot \sum_{i=1}^{14} F_i),$$

где FP — число функциональных точек, полученное на предыдущем шаге,

F_i — i-ый коэффициент регулировки сложности, принимающий значения от 0 до 5 (нет влияния, случайное, небольшое, среднее, важное, основное), которое выбирается по соответствующей таблице.

Из числа функциональных точек выводится количество строк кода по таблицам соответствия их числа (число строк кода на одну функциональную точку), построенных на основе статистических данных в зависимости от языка программирования.

Применение

Постановка задачи

Компания получила заказ на разработку клиентского мобильного приложения брокерской системы. Программа позволяет просматривать актуальную биржевую информацию, производить сделки и отслеживать их выполнение. Приложение имеет 4 страницы: авторизация, биржевые сводки, заявки, новая заявка.

Страница «Авторизация»

На данной странице осуществляется ввод логина и пароля пользователя для входа в систему. Страница содержит **два поля ввода** и **одну командную кнопку**, а также **флажок-переключатель**, который активируется при необходимости запоминания параметров авторизации.

По данной странице можно выделить следующий набор функциональных типов:

- внутренние логические файлы:
  - таблица пользователей бд с полями логина и пароля:
    DET = 2 — логин, пароль;
    RET = 1 — логин и пароль являются строками;
    Уровень: низкий;
  - локальный файл для хранения данных одного пользователя:
    DET = 2 — логин, пароль;
    RET = 1 — логин и пароль являются строками;
    Уровень: низкий;
- внешник логический файл: отсутствует;
- внешний ввод: запоминание данных пользователя:
  DET = 4 — логин, пароль, флажок, кнопка;
  FTR = 1 — локальный файл для данных одного пользователя.
  Уровень: низкий;

— внешний вывод: отсутствует;

— внешний запрос: авторизация:

DET = 4 — логин, пароль, флажок, кнопка;
FTR = 1 — локальный файл для данных одного пользователя.
Уровень: низкий.

Итого по странице:

— 2 ILF низкого уровня;

— 1 EI низкого уровня;

— 1 EQ низкого уровня.

Страница «Биржевые сводки»

Страница содержит **таблицу**, **кнопку** «Добавить» и диалоговое окно с **одним полем для ввода** и **двумя командными кнопками**.

Таблица содержит три колонки: Ценная бумага (имя бумаги), Цена (цена за одну ценную бумагу), Изменение (изменение цены бумаги со времени последнего закрытия биржи). Кнопка «Добавить» вызывает **диалоговое окно** для добавления новой бумаги (окно состоит из поля ввода и кнопок OK, Cancel).

По данной странице можно выделить следующий набор функциональных типов:

— внутренний логический файл: таблица ценных бумаг с полем имени:

DET = 1 — имя;
RET = 1 — имя — строка;
Уровень: низкий.

— внешний логический файл: информация по цене и изменению о ценных бумагах;

DET = 3 — имя, цена, изменение;
RET = 1 — имя — строка, цена и изменение — вещественный тип;
Уровень: низкий.

— внешний ввод: добавление новой бумаги:

DET = 3 — текстовое поле для имени, кнопка «Cancel», кнопка «OK»;
FTR = 1 — добавление записи во внутренний логический файл.
Уровень: низкий.

— внешний вывод информации по ценным бумагам:

DET = 3 — имя, цена, изменение;
FTR = 2 — обращение к внутреннему файлу с именами бумаг и к внешнему с информацией о цене и изменении;
Уровень: низкий;

— внешний запрос: отсутсвует.

Итого по странице:

— 1 ILF низкого уровня;

— 1 EIF низкого уровня;

— 1 EI низкого уровня;

— 1 EO низкого уровня.

Страница «Заявки»

Заявки содержат **таблицу**, отображающую текущие (еще не выполненные) заявки на покупку или продажу ценных бумаг. Таблица содержит четыре поля: Тип (покупка/продажа), Имя бумаги, Цена, по которой готовы покупаться/продаваться бумаги, Количество бумаг для покупки/продажи. При нажатии на любую строку таблицы появляется контекстное меню с возможностью **удалить** или **изменить заявку**.

По данной странице можно выделить следующий набор функциональных типов:

— внутренний логический файл: таблица текущих заявок:

DET = 4 — тип, имя, цена, количество;
RET = 4 — тип — логический, имя — строка, цена — вещественное

число; количество — целое число;
Уровень: низкий.

— внешний логический файл: отсутствует

— внешний ввод:

  – удаление:
    DET = 5 — тип, имя, цена, количество, кнопка;
    FTR = 1 — обращение к внутреннему логическому файлу;
    Уровень: низкий;

  – изменение:
    DET = 5 — тип, имя, цена, количество, кнопка;
    FTR = 1 — обращение к внутреннему логическому файлу;
    Уровень: низкий;

— внешний вывод информации о заявках:
  DET = 4 — тип, имя, цена, количество;
  FTR = 1 — обращение к внутреннему логическому файлу;
  Уровень: низкий;

— внешний запрос: отсутсвует.

— 1 ILF низкого уровня;

— 2 EI низкого уровня;

— 1 EO низкого уровня.

Страница «Новая заявка»

Страница позволяет оформить заявку на покупку или продажу ценной бумаги. Страница состоит из **4 полей**: Бумага (имя бумаги), Цена (цена, по которой необходимо купить/продать бумагу), Покупка (булева переменная в значение true обозначает покупку, false – продажа) и кнопки «Оформить» - для подтверждения оформления заявки.

По данной странице можно выделить следующий набор функциональных типов:

— внутренний логический файл: тот же, что и на странице заявок;

— внешний логический файл: отсутствует;

— внешний ввод: создание новой заявки:

DET = 5 — поля ввода имя, цена, количество, флаг покупки и кнопка;
FTR = 1 — обращение к внутреннему логическому файлу;
Уровень: низкий;

— внешний вывод: отсутствует;

— внешний запрос: отсутствует.

— 1 EI низкого уровня.

Расчеты

Всего функциональных типов:

— 5 EI низкого уровня;

— 2 EO низкого уровня;

— 1 EQ низкого уровня;

— 4 ILF низкого уровня;

— 1 EIF низкого уровня.

Для уточнения числа функциональных точек используются следующие значения характеристики продукта:

— Обмен данными — 5

— Распределенная обработка — 5

— Производительность — 3

— Эксплуатационные ограничения по аппаратным ресурсам — 2

— Транзакционная нагрузка — 3

— Интенсивность взаимодействия с пользователем (оперативный ввод данных) — 4

— Эргономические характеристики, влияющие на эффективность работы конечных пользователей — 1

— Оперативное обновление — 4

— Сложность обработки — 4

— Повторное использование — 0

— Легкость инсталляции — 1

— Легкость эксплуатации/администрирования — 2

— Портируемость — 2

— Гибкость — 2

Результаты расчетов числа функциональных точек представлены на рисунке.

Таким образом, число функциональных точек:

— первоначальное = 59;

— скорректированное = 60.77.

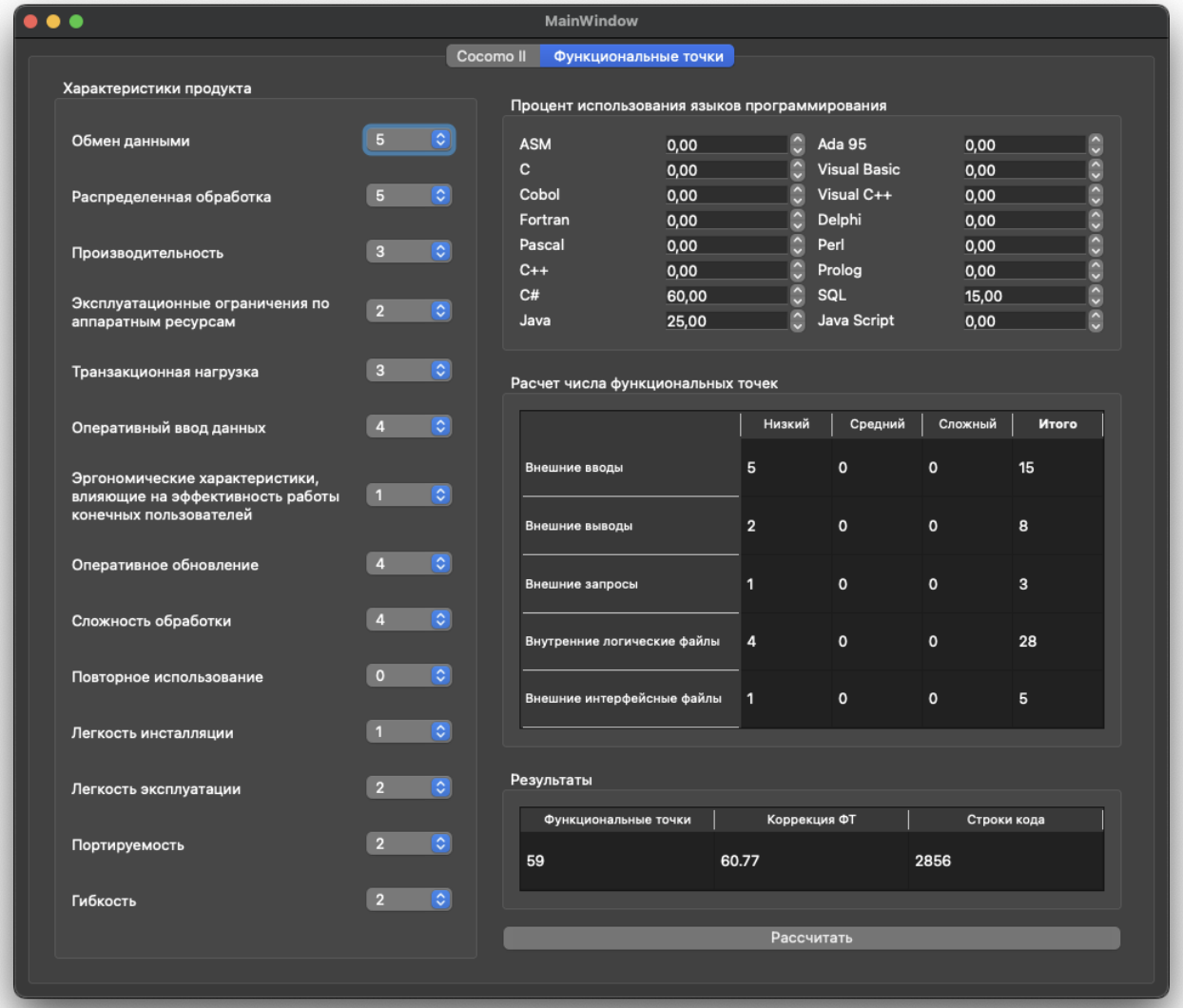
Процентное соотношения языков программирования:

— SQL — 15%;

— C# — 60%;

— Java — 25%.

С учетом данного соотношения с использованием таблиц соответствия числа функциональных точек строкам кода получаем, что проект будет состоять по данной оценке из 2856 строк кода.



Модель ранней разработки архитектуры в COCOMO=

Описание модели

Для получения приблизительных оценок проектных затрат периода выполнения проекта перед тем, как будет определена архитектура в целом, применяется модель ранней разработки архитектуры. Для такой модели характерны оценки умеренной точности и ясно понимаемые особенности проекта, требования и архитектура. В качестве единиц измерения используются функциональные точки или KSLOC.

Трудозатраты вычисляются так:

$$\text{Трудозатраты} = 2.45 \cdot \text{EArch} \cdot \text{Размер}^{p},$$

где Размер = KSLOC, EArch определяется через произведение множителей трудоемкости:

$$\text{EArch} = \text{PERS} \cdot \text{RCPX} \cdot \text{RUSE} \cdot \text{PDIF} \cdot \text{PREX} \cdot \text{FCIL} \cdot \text{SCED}.$$

Время разработки получается в соответствии с формулой:

$$\text{Время} = 3 \cdot \text{Трудозатраты}^{0.33+0.2\cdot(\text{p}-1.01)},$$

где p — показатель степени. Значение показателя степени рассчитывается с учетом факторов, влияющих на показатель степени:

$$\text{p} = \frac{(\text{PREC} + \text{FLEX} + \text{RESL} + \text{TEAM} + \text{PMAT})}{100} + 1.01.$$

Применение

Из условия задания:

— надежность и сложность продукта (RCPX) — очень высокие;

— повторное использование компонентов (RUSE) — низкий;

— опытность персонала (PERS) — низкая;

— способности персонала (PREX) — номинальные;

— сложность платформы (PDIF) — высокая;

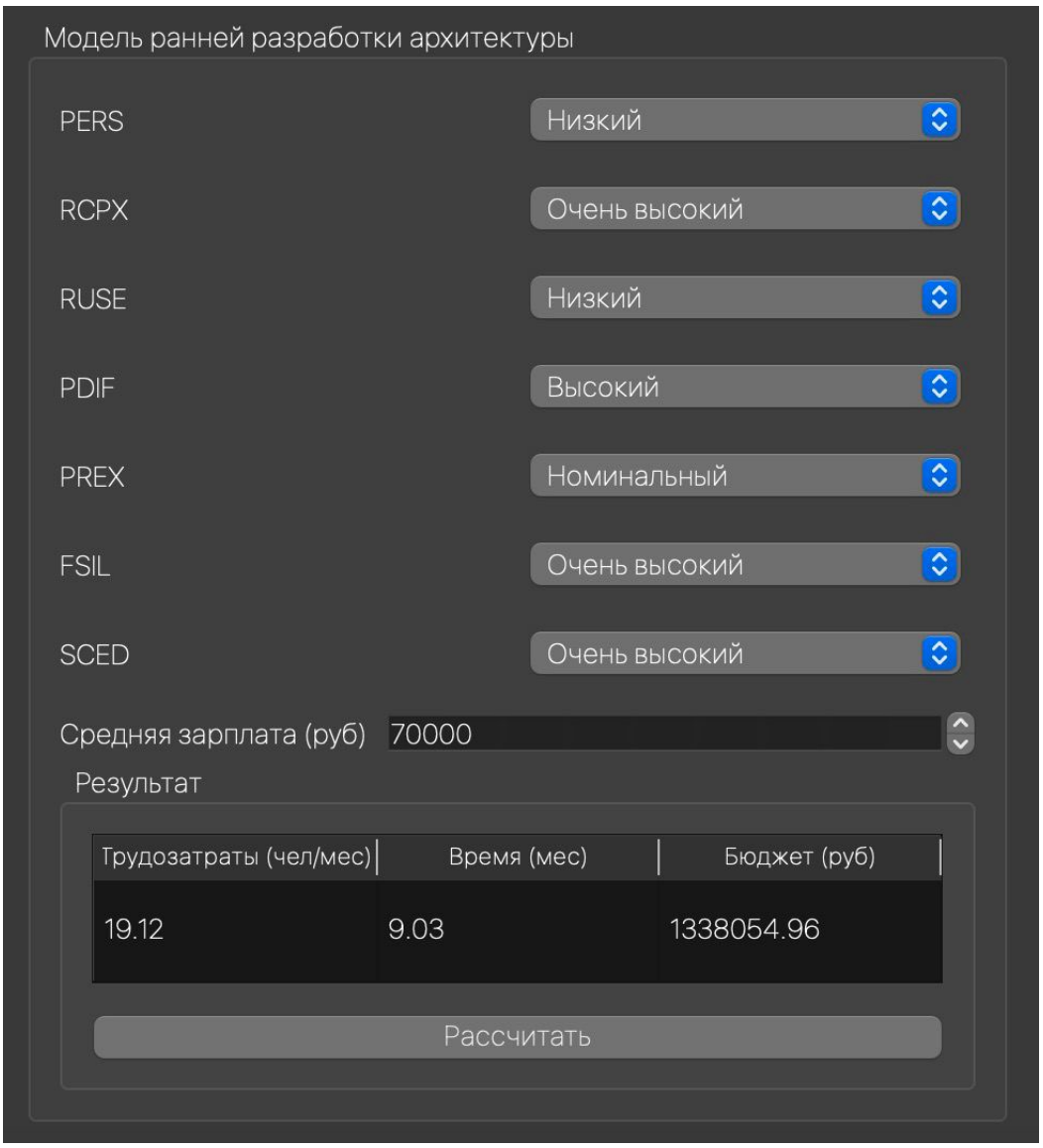
— возможности среды (FCIL) — очень высокие;

— сроки (SCED) — очень высокие.

Результат

На рисунке представлена оценка трудозатрат и длительности разработки с использованием модели ранней разработки архитектуры.

Средняя численность команды определяется по следующей формуле:



$$\text{Численность команды} = \frac{\text{Трудозатраты}}{\text{Время}} = \frac{19.2}{9.03} = 3 \text{ работника.}$$

Предварительная оценка бюджета для средней зарплаты 70 000 рублей проводится по следующей формуле:

$$\text{Бюджет} = \text{Трудозатраты} \cdot \text{Средняя зарплата} = 19.12 \cdot 70000 = 1338054.96 \text{ рублей.}$$

Вывод

В ходе выполнения данной работы была освоена методология оценки параметров проекта COCOMO2 и разработан программный инструмент для её применения. Выполнен анализ выданного задания:

— рассчитаны функциональные точки и показатель степени модели (p);

— были определены факторы, влияющие на показатель степени;

— рассчитаны трудозатраты и времени по моделям ранней разработки архитектуры приложения и композиции приложения.

По модели композиции приложения прогноз более благоприятный, чем в модели ранней архитектуры приложения.

Методология COCOMO2 является более сложной по сравнению с COCOMO, но позволяет более тонко настраивать параметры плана, что даёт более точный и детальный прогноз.