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»

Лабораторная работа по дисциплине «Экономика программной инженерии»

Тема Оценка параметров программного проекта с использованием метода функциональных точек и модели COCOMO II

Студенты Светличная А.А. (ИУ7-83Б), Шабанова А.В. (ИУ7-83Б), Кириченко С. П. (ИУ7-81Б)

Преподаватель Барышникова М.Ю.

Москва — 2024 г.

Модель композиции приложения в COCOMO II

Данная модель используется на ранней стадии конструирования ПО, когда:

— рассматривается макетирование пользовательских интерфейсов;

— оценивается производительность;

— определяется степень зрелости технологии.

Модель ориентирована на применение объектных точек. Объектная точка — средство косвенного измерения ПО. Подсчет количества объектных точек производится с учетом количества экранов (как элементов пользовательского интерфейса), отчетов и компонентов, требуемых для построения приложения.

Модель композиции приложения следует рассматривать как некую рабочую гипотезу, основанную на концепции продукта.

Правила подсчета объектных точек:

— простые изображения принимаются за 1 объектную точку, изображения умеренной сложности принимаются за 2 точки, очень сложные изображения принято считать за 3 точки;

— для простых отчетов назначаются 2 объектные точки, умеренно сложным отчетам назначаются 5 точек, написание сложных отчетов оценивается в 8 точек;

— каждый модуль на языке третьего поколения считается за 10 объектных точек.

Новые объектные точки NOP определяются по следующей формуле:

$$\text{NOP} = \text{Объектные точки} \cdot \frac{100 - \%RUSE}{100}, \tag{1}$$

где %RUSE — процент повторного использования кода программы.

Трудозатраты вычисляются по следующей формуле:

$$\text{Трудозатраты} = \frac{NOP}{PROD}, \tag{2}$$

где PROD — оценка скорости разработки.

Длительность выполнения проекта на уровне композиции приложения определяется по следующей формуле:

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

где p — показатель степени.

Значение показателя степени рассчитывается с учетом факторов, влияющих на показатель степени по следующей формуле:

$$\text{p} = \frac{(PREC + FLEX + RESL + TEAM + PMAT)}{100} + 1.01. \tag{4}$$

Применение модели

Из макета интерфейса:

1. для страницы «Авторизация»:
   - одна форма средней сложности (авторизация);
2. для страницы «Биржевые сводки»:
   - одна сложная форма (таблица биржевых сводок);
   - одна форма средней сложности (форма ввода);
3. для страницы «Заявки»:
   - одна форма средней сложности (таблица заявок);
   - одна простая форма (кнопки «Удалить» и «Изменить»);
4. для страницы «Новая заявка»:
   - одна форма средней сложности (форма добавления заявки).

Итого:

— 1 простая форма;

— 4 формы средней сложности;

— 1 сложная форма.

В проекте два модуля, написанных на языках третьего поколения (C# и Java). Повторного использования компонентов не предусматривается (RUSE). Опытность команды — низкая.

Факторы, влияющие на показатель степени

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

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

— Разрешение рисков в архитектуре системы (RESL) — некоторое (40 %), так как анализу архитектурных рисков было уделено лишь некоторое внимание;

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

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

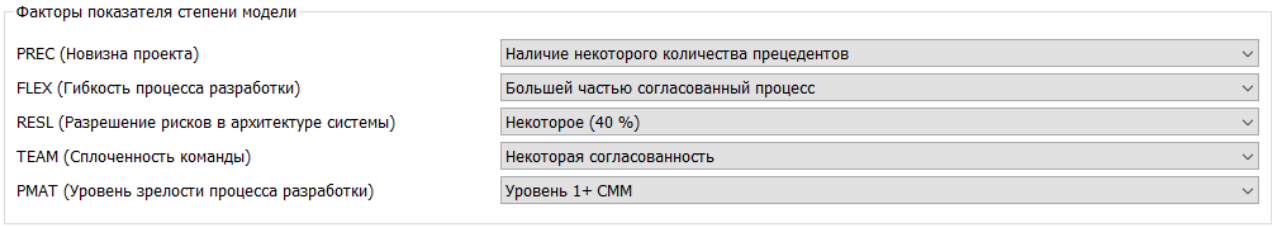

Рисунок 1 – Факторы показателя степени модели

$$\mathrm{p} = \frac{(3.72 + 2.03 + 5.65 + 3.29 + 6.24)}{100} + 1.01 = 1.2193. \tag{5}$$

Результаты

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

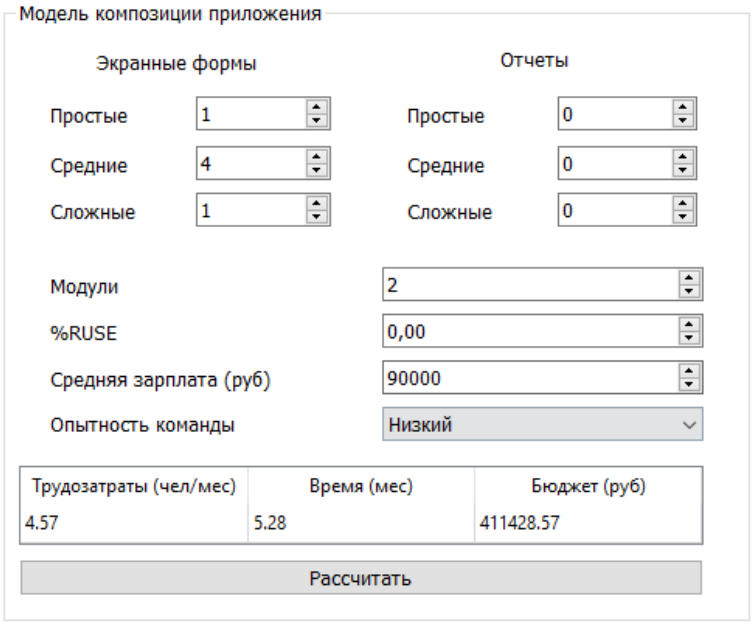

Рисунок 2 – Результаты расчетов по модели композиции приложения

Средняя численность команды определяется по следующей формуле:

$$\text{Численность команды} = \ = \frac{4.57}{5.28} = 1. \tag{6}$$

Предварительная оценка бюджета проводится по следующей формуле:

$$\text{Бюджет} = \text{Трудозатраты} \cdot \text{Средняя зарплата} = 4.57 \cdot 90000 = 411300. \tag{7}$$

Метод функциональных точек

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, получающая данные от пользователя);
- **внешний вывод** (EO, транзакция передающая данные пользователю);
- **внешний запрос** (EQ, интерактивный диалог с пользователем, требующий от него каких-либо действий);
- **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** — это уникальное распознаваемое пользователем, нерекурсивное (неповторяющееся) поле данных;
- **RET** — идентифицируемая пользователем логическая группа данных внутри ILF или EIF;
- **FTR** — это тип файла, на который ссылается транзакция.

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.

Характеристики команды, продукта и проекта

Разработанное ПО состоит из трех компонентов. Первый компонент составляет по объему примерно 15% программного кода и будет написан на SQL, второй (около 60% кода) — на C#, а третий в объеме 25% кода — на Java.

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.

Страница «Авторизация»

Рисунок 3 – 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 (ILF):

— 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.

Внешний ввод (EI):

— запоминание данных пользователя:

  – DET = 4 (логин, пароль, флажок, кнопка);
  – FTR = 1 (локальный файл для данных одного пользователя).

  Уровень: низкий.

Внешний запрос (EQ):

— авторизация:

  – DET = 4 (логин, пароль, флажок, кнопка);
  – FTR = 1 (локальный файл для данных одного пользователя).

  Уровень: низкий.

Итого по странице «Авторизация»:

— 2 ILF низкого уровня — - данные из полей логин и пароль используются во внутренних взаимодействиях системы;

— 1 EI низкого уровня — по кнопке происходит транзакция получения данных от пользователя;

— 1 EQ низкого уровня — форма авторизации является в целом интерактивным диалогом.

Страница «Биржевые сводки»

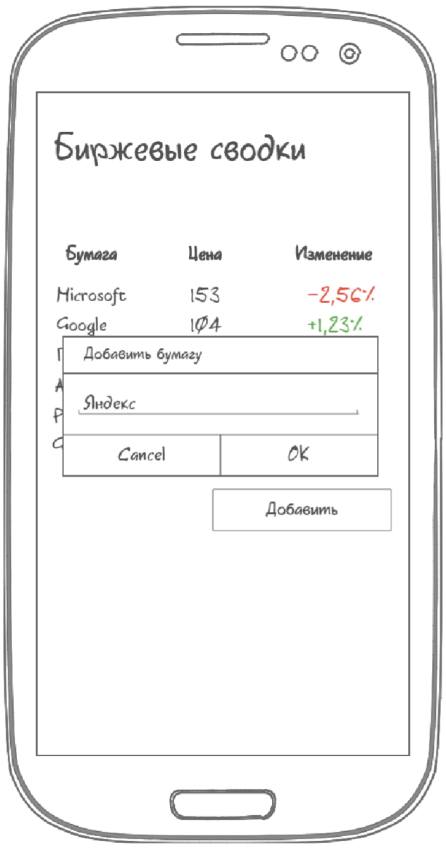

Рисунок 4 – Страница «Биржевые сводки»

Биржевые сводки отражают текущую ситуацию на бирже. 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 (ILF):

— таблица ценных бумаг с полем имени:

  – DET = 1 (имя);
  – RET = 1 (имя является строкой).

Уровень: низкий.

Внешний интерфейсный файл (EIF):

— информация по цене и изменению о ценных бумагах:

  – DET = 3 (имя, цена, изменение);
  – RET = 2 (имя — строка, цена и изменение — вещественный тип).

Уровень: низкий.

Внешний ввод (EI):

— 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 (EO):

— 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.

Итого по странице «Биржевые сводки»:

— 1 ILF низкого уровня — - данные из поля добавить бумагу будут использоваться во внутреннем взаимодействии системы;

— 1 EIF низкого уровня — данные по акциям получаются из внешней системы;

— 1 EI низкого уровня — транзакция передачи данных по кнопке OK в приложение;

— 1 EO низкого уровня — таблица, которая обновляется по кнопке добавить является транзакцией передачи данных пользователю.

Страница «Заявки»

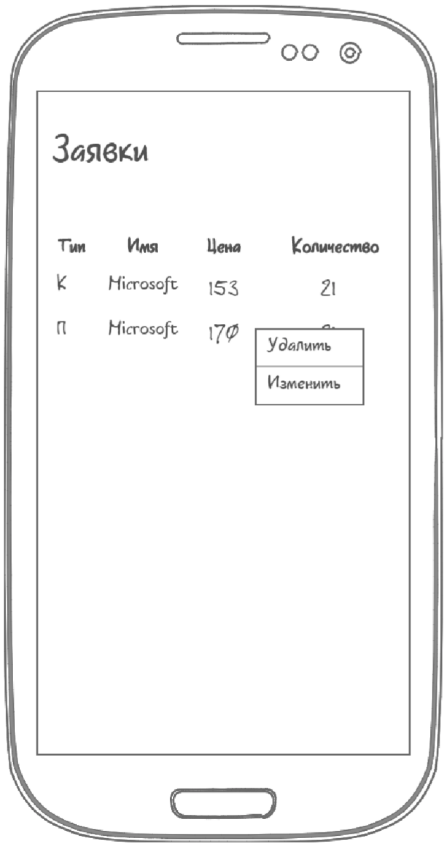

Рисунок 5 – Страница «Заявки»

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 (ILF):

— таблица текущих заявок:

- DET = 4 (тип, имя, цена, количество);
- RET = 4 (тип — логический, имя — строка, цена — вещественное число, количество — целое число).

Уровень: низкий.

Внешний ввод (EI):

— удаление заявки:

- DET = 5 (тип, имя, цена, количество, кнопка);
- FTR = 1 (обращение к внутреннему логическому файлу).

Уровень: низкий.

— изменение заявки:

- DET = 5 (тип, имя, цена, количество, кнопка);
- FTR = 1 (обращение к внутреннему логическому файлу).

Уровень: низкий.

Внешний вывод (EO):

— вывод информации о заявках:

- DET = 4 (тип, имя, цена, количество);
- FTR = 1 (обращение к внутреннему логическому файлу).

Уровень: низкий.

Итого по странице «Заявки»:

— 1 ILF низкого уровня — информация описывающая заявку будет использоваться во внутренних взаимодействиях системы;

— 2 EI низкого уровня — кнопки удалить и изменить описывают транзакции получения данных от пользователя;

— 1 EO низкого уровня — таблица является данными, которые передаются пользователю.

Страница «Новая заявка»

Рисунок 6 – 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 (ILF):

— Используется та же таблица, что и для страницы «Заявки».

Внешний ввод (EI):

— создание новой заявки:

  – DET = 5 (имя, цена, количество, флаг покупки и кнопка);
  – FTR = 1 (обращение к внутреннему логическому файлу).

  Уровень: низкий.

Итого по странице «Новая заявка»:

— 1 EI низкого уровня — по кнопке оформить происходит транзакция получения данных от пользователя.

Расчеты

Итого функциональных типов по всем страницам:

— 5 EI низкого уровня;

— 2 EO низкого уровня;

— 1 EQ низкого уровня;

— 4 ILF низкого уровня;

— 1 EIF низкого уровня.

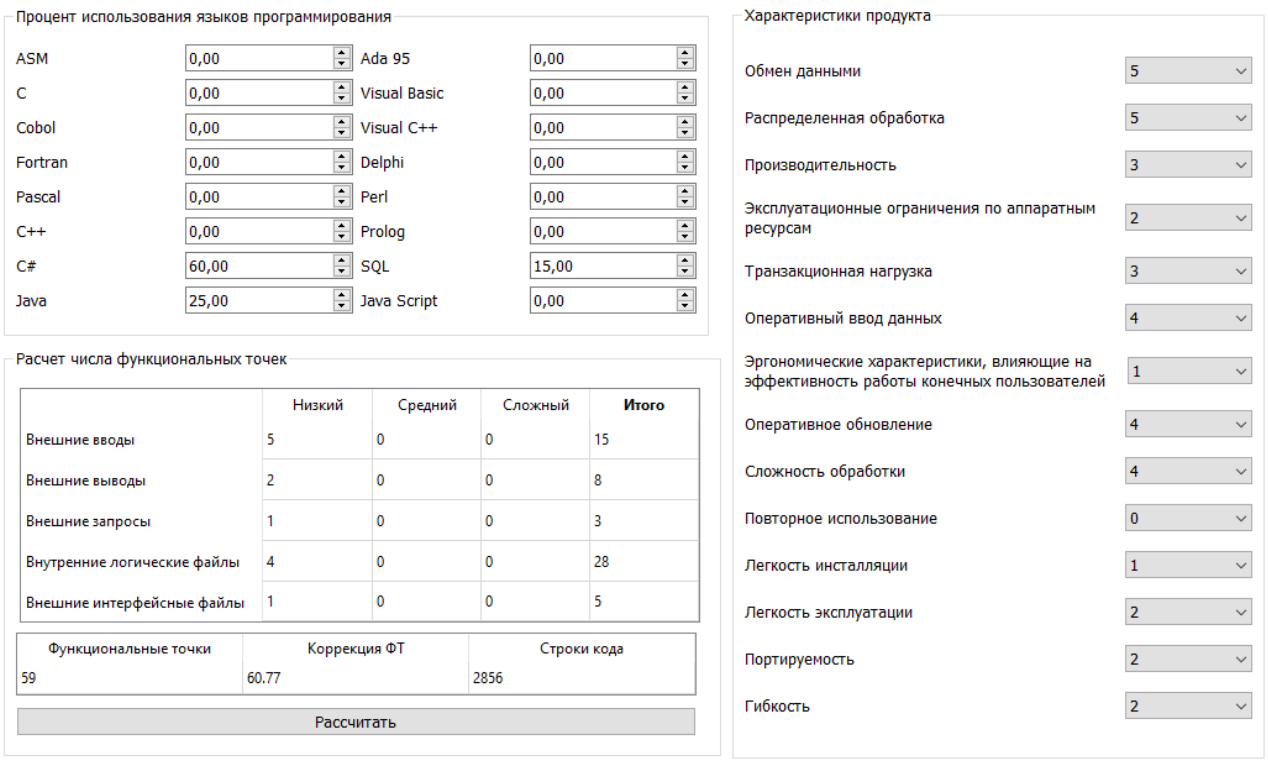

Рисунок 7 – Результаты расчетов по методу функциональных точек

Были получены следующие результаты:

— первоначальное число функциональных точек = 59;

— скорректированное число функциональных точек = 60.77;

— с учетом соотношения языков программирования проект будет состоять из 2856 строк кода.

Модель ранней разработки архитектуры в COCOMO II

Эта модель применяется для получения приблизительных оценок проектных затрат периода выполнения проекта перед тем как будет определена архитектура в целом. В этом случае используется небольшой набор новых драйверов затрат и новых уравнений оценки. В качестве единиц измерения используются функциональные точки либо KSLOC.

Трудозатраты вычисляются по следующей формуле:

$$\text{Трудозатраты} = 2.45 \cdot \text{EArch} \cdot \text{Размер}^{p}, \tag{8}$$

где Размер — KSLOC, а EArch определяется по следующей формуле:

$$\text{EArch} = PERS \cdot RCPX \cdot RUSE \cdot PDIF \cdot PREX \cdot FCIL \cdot SCED. \tag{9}$$

Длительность выполнения проекта определяется по следующей формуле:

$$\text{Время} = 3 \cdot \text{Трудозатраты}^{(0.33+0.2\cdot(p-1.01))}, \tag{10}$$

где p — показатель степени.

Значение показателя степени рассчитывается с учетом факторов, влияющих на показатель степени по следующей формуле:

$$\text{p} = \frac{(PREC + FLEX + RESL + TEAM + PMAT)}{100} + 1.01. \tag{11}$$

Применение модели

Надежность и уровень сложности (RCPX) разрабатываемой системы оцениваются как очень высокие, повторного использования компонентов не предусматривается (RUSE). Возможности персонала (PERS) — средние, его опыт работы в разработке систем подобного типа (PREX) низкий. Сложность платформы (PDIF) высокая. Разработка предусматривает очень интенсивное использование инструментальных средств поддержки (FCIL). Заказчик настаивает на жестком графике (SCED).

Результаты

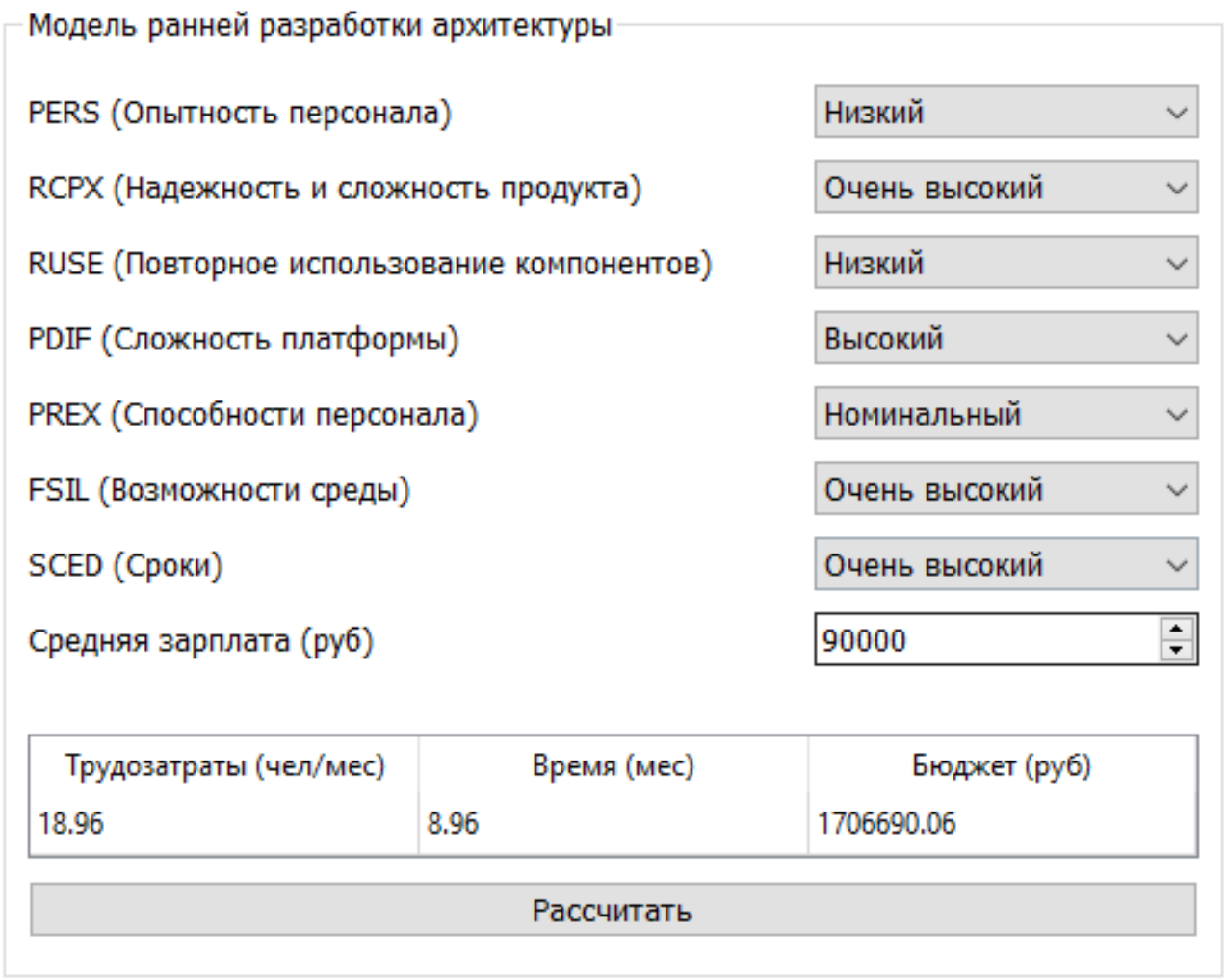

Рисунок 8 – Результаты расчетов по модели ранней разработки архитектуры

Средняя численность команды определяется по следующей формуле:

$$\text{Численность команды} = \quad = \frac{18.96}{8.96} = 3. \tag{12}$$

Предварительная оценка бюджета проводится по следующей формуле:

$$\text{Бюджет} = \text{Трудозатраты} \cdot \text{Средняя зарплата} = 18.96 \cdot 90000 = 1706400. \tag{13}$$

Выводы

В ходе выполнения работы был разработан инструмент для определения трудозатрат и времени разработки проекта методом COCOMO II. Также в ходе выполнения:

— был рассчитан показатель степени;

— были определены трудозатраты и длительность выполнения проекта по модели композиции приложения;

— методом функциональных точек вычислено количество строк кода проекта;

— были определены трудозатраты и длительность выполнения проекта по модели ранней разработки архитектуры.

В итоге было выяснено, что модель композиции приложения дает более оптимистичный прогноз, по сравнению с моделью ранней архитектуры приложения.