|  |  |
| --- | --- |
|  | **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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

Студент **ИУ7-86Б** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чепрасов К.М.**

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Барышникова М. Ю.**

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Силантьева А. В.**

*2024 г.*

**1. Описание методов**

**1.1 COCOMO 2**

COCOMO II рассматривает три различные модели оценки стоимости

* Модель композиции приложения
* Модель ранней разработки архитектуры.
* Постархитектурная модель

Время в этой модели считается так:

.

Значение P рассчитывается с учетом 5 показателей по восьмибалльной шкале от низшего (7) до наивысшего (0) уровня. Значения всех показателей суммируются, сумма делится на 100, результат прибавляется к числу 1.01.

### **Модель композиции приложения**

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.

В этой модели сначала считаются новые объектные точки:

.

Затем считаются трудозатраты:

ТРУДОЗАТРАТЫ = NOP/PROD, где PROD – оценка скорости разработки

### **Модель ранней разработки архитектуры**

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 считаются так:

,

где Earch = PeRS\* RCPX \* RUSE \* PDIF \* PREX \* FCIL \* SCED .

Множитель Earch является произведением семи показателей, характеризующих проект и процесс создания ПО, а именно: надежность и уровень сложности разрабатываемой системы (RCPX), повторное использование компонентов (RUSE), сложность платформы разработки (PDIF), возможности персонала (PERS), опыт персонала (PREX), график работ (SCED) и средства поддержки (FCIL). Каждый множитель может быть оценен экспертно, либо его можно вычислить путем комбинирования значений более детализированных показателей, которые используются на постархитектурном уровне.

## **Методика оценки трудоемкости разработки на основе функциональных точек**

**Функциональная точка** — это единица измерения функциональности программного обеспечения. Функциональность программы связана с обработкой информации по запросу пользователя и не зависит от применяемых технических решений. Пользователи — это отправители и целевые получатели данных, ими могут быть как реальные люди, так и смежные интегрированные информационные системы.

Метод функциональных точек позволяет:

* оценивать категории пользовательских бизнес-функций
* разрешить проблему, связанную с трудностью получения LOC – оценок на ранних стадиях жизненного цикла
* определять количество и сложность входных и выходных данных, их структуру, а также внешние интерфейсы, связанные с программной системой

Определение числа функциональных точек является методом количественной оценки ПО, применяемым для измерения функциональных характеристик процессов его разработки и сопровождения независимо от технологии, использованной для его реализации. Трудоемкость вычисляется на основе функциональности разрабатываемой системы, которая, в свою очередь, определяется путем выявления **функциональных типов** — логических групп взаимосвязанных данных, используемых и поддерживаемых приложением, а также **элементарных процессов**, связанных с вводом и выводом информации.

Типы элементарных процессов, используемых в методе функциональных точек:

* **EI** (Внешний ввод) — элементарный процесс, перемещающий данные из внешней среды в приложение.
* **EO** (Внешний вывод) — элементарный процесс, перемещающий данные, вычисленные в приложении, во внешнюю среду.
* **EQ** (Внешний запрос) — элементарный процесс, состоящий из комбинации «запрос/ответ», не связанный с вычислением производных данных или обновлением внутренних логических файлов (базы данных).
* **ILF** (Внутренний логический файл) — выделяемые пользователем логически связанные группы данных или блоки управляющей информации, которые поддерживаются внутри продукта и обслуживаются через внешние вводы.
* **EIF** (Внешний интерфейсный файл) — выделяемые пользователем логически связанные группы данных или блоки управляющей информации, на которые ссылается продукт, но которые поддерживаются вне продукта

Количество транзакционных функциональных типов (входных элементов приложения, выходных элементов приложения и внешних запросов) определяется на основе выявления входных и выходных документов, экранных форм, отчетов, а также по диаграммам классов.

Для каждого выявленного функционального типа (EI, ЕО или EQ) определяется его сложность (низкая, средняя или высокая), которая зависит от количества связанных с этим функциональным типом DET, RET и FTR.

* **FTR** – количество связанных с каждым функциональным типом файлов типа ссылок.
* **DET** – количество связанных с каждым функциональным типом элементарных данных. (количество типов элементов данных)
* **RET** – количество типов элементов записей.

После того, как подсчитаны функциональные типы, определены сложность каждой функции, каждая функция умножается на соответствующий ей параметр, а затем суммируется с целью получения общего количества функциональных точек.

Затем значение корректируются с учетом коэффициентов регулировки сложности.

,

где Fi – 14 коэффициентов регулировки сложности, каждый из которых может принимать значения от 0 до 5. Эти коэффициенты представлены на рисунке 1.

|  |
| --- |
| Рис 1. Коэффициенты регулировки сложности |

Затем FP оценки переводятся в LOC-оценки в соответствии с таблицей, представленной на рисунке 2. В результате мы получаем количество строк кода.

|  |
| --- |
| Рис 2. Пересчет FP-оценок в LOC оценки |

## **Расчет задания по варианту**

**Определение количества строк кода**

Характеристики проекта, полученные из задания:

1. Обмен данными – 5
2. Распределенная обработка – 5
3. Производительность – 3
4. Эксплуатационные ограничения по аппаратным ресурсам – 0
5. Транзакционная нагрузка – 3
6. Интенсивность взаимодействия с пользователем (оперативный ввод данных) – 2
7. Эргономические характеристики, влияющие на эффективность работы конечных пользователей – 0
8. Оперативное обновление – 4
9. Сложность обработки – 4
10. Повторное использование – 3
11. Легкость инсталляции – 0
12. Легкость эксплуатации/администрирования – 3
13. Портируемость – 5
14. Гибкость – 0

Используемые при разработке языки программирования:

1. SQL – 30%
2. JavaScript – 10%
3. Java – 60%

Роли пользователей:

1. Администратор
   * Просмотр всех выплат
   * Просмотр данных пользователей
   * Редактирование пользователей
   * Создание новых пользователей
2. Пользователь
   * Просмотр штрафов
   * Оплата штрафов

Модули:

1. Приложение для мобильного телефона
   * Страница регистрации
     + Логин
     + Пароль
     + Номер водительского удостоверения
     + Номер банковской карты
   * Страница просмотра штрафов
     + Штрафы
     + Кнопка «Оплатить» для каждого штрафа
     + Приходит подтверждение/отказ после нажатия кнопки
2. Веб-портал
   * Тоже, что и «Приложение для мобильного телефона»
   * Панель администратора
3. Модуль регистрации и авторизации
   * Добавление пользователей в базу данных
4. Модуль обмена данными с системой ГИБДД
   * Получает штрафы
     + Номер постановления
     + Дата постановления
     + Имя
     + Фамилия
     + Отчество
     + Сумма штрафа
   * Отсылает подтверждение об удалении или отказ
5. Модуль проведения платёжных транзакций
   * Отправляет запрос
     + Номер карты пользователя
     + Номер счёта ГИБДД
     + Сумма на выполнение проведения оплаты
     + Отсылает успех/провал операции

Показатели проекта:

1. Надёжность и сложность (RCPX) – очень высокий
2. Повторное использование компонентов (RUSE) – очень низкий
3. Возможности персонала (PERS) – очень высокий
4. Опыт персонала (PREX) – низкий
5. Сложность платформы (PDIF) – номинальный
6. Инструментальные средства поддержки (FCIL) – высокий
7. График работ (SCED) – низкий

В данной работе рассматриваются модель композиции приложения и модель ранней разработки архитектуры

1. Новизна проекта (PREC) – почти полное отсутствие прецедентов, в значительной мере непредсказуемый проект
2. Гибкость процесса разработки (FLEX) – точный, строгий процесс разработки
3. Разрешение рисков в архитектуре системы (RESL) – в целом (75%)
4. Сплочённость команды (TEAM) – взаимодействие как в едином целом
5. Уровень развития процесса разработки (PMAT) – уровень 2

### **Расчёт по методу функциональных точек**

**Вычисление EI**

* Регистрация:
  + FTR = 1 (один внутренний логический файл);
  + DET = 4 (логин, пароль, номер водительского удостоверения и номер банковской карты).
* Оплата штрафа
  + FTR = 1 (один внутренний логический файл);
  + DET = 1 (дата оплаты).
* Добавление пользователя администратором;
  + FTR = 1 (один внутренний логический файл);
  + DET = 5 (логин, пароль, тип, номер карты, номер удостоверения).
* Редактирование пользователя администратором;
  + FTR = 1 (один внутренний логический файл);
  + DET = 5 (логин, пароль, тип, номер карты, номер удостоверения).

Уровень сложности – низкий.

**Вычисление EO**

* Вывод списка штрафов
  + FTR = 1 (один внутренний логический файл);
  + DET = 4 (номер, ФИО, дата, сумма).
* Вывод сообщения о статусе оплаты
  + FTR = 1 (один внутренний логический файл);
  + DET = 1 (статус оплаты).
* Вывод информации о пользователе
  + FTR = 1 (один внутренний логический файл);
  + DET = 6 (все поля из базы).

Уровень сложности – низкий.

**Вычисление EQ**

* Запрос на авторизацию
  + FTR = 1 (один внутренний логический файл);
  + DET = 3 (логин, пароль, флажок).
* Запрос на вывод списка
  + FTR = 1 (один внутренний логический файл);
  + DET = 4 (номер, ФИО, дата, сумма).

Уровень сложности – низкий.

**Вычисление ILF**

* ILF
  + RET = 3 (элементы записи);
  + DET = 6 (элементы данных).

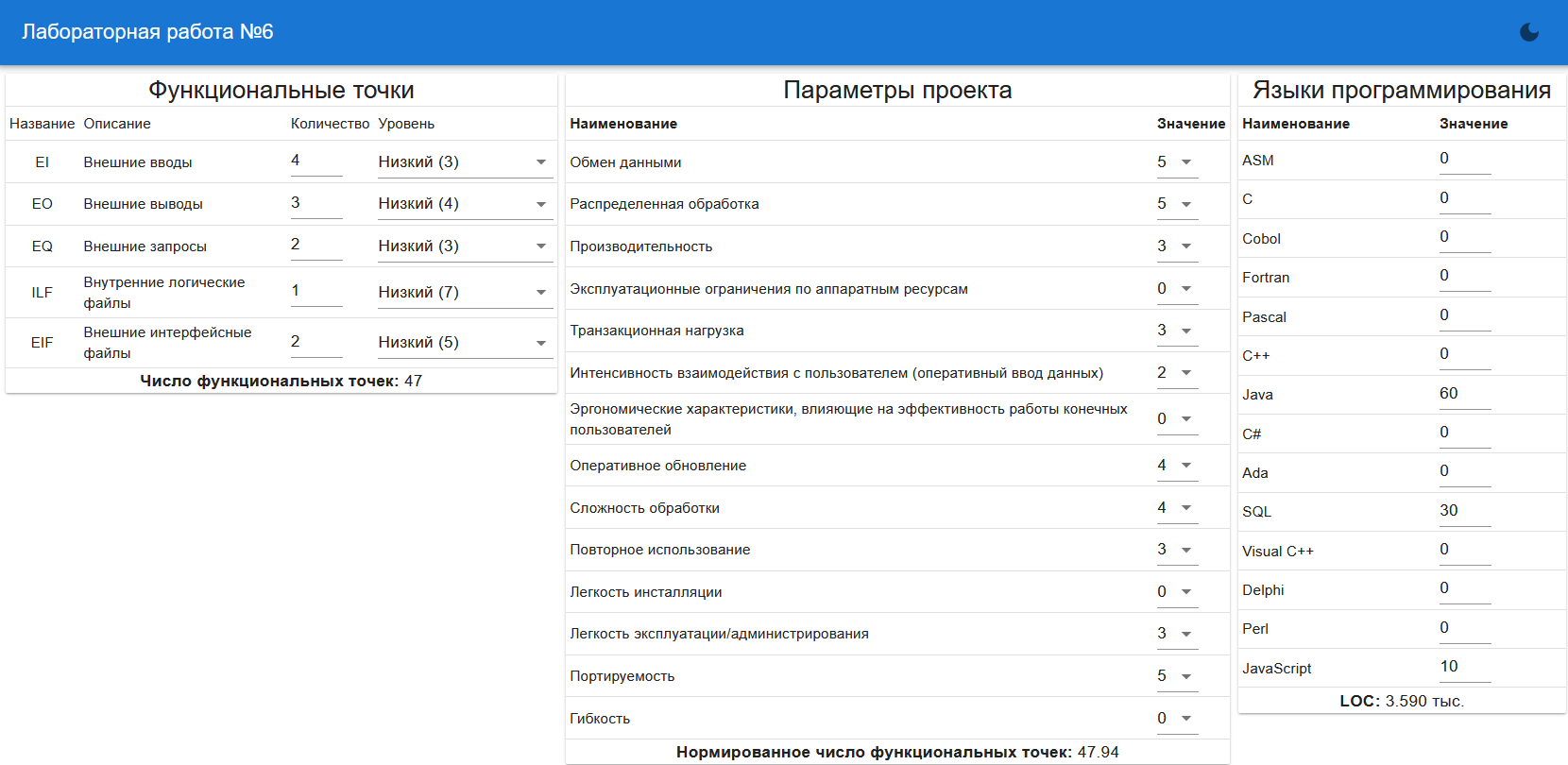
Уровень сложности – низкий.

**Вычисление EIF**

* Веб
  + RET = 3 (элементы записи);
  + DET = 10 (элементы данных).
* Мобильное приложение
  + RET = 3 (элементы записи);
  + DET = 5 (элементы данных).

Уровень сложности – низкий.

**Расчёты для метода функциональных точек**

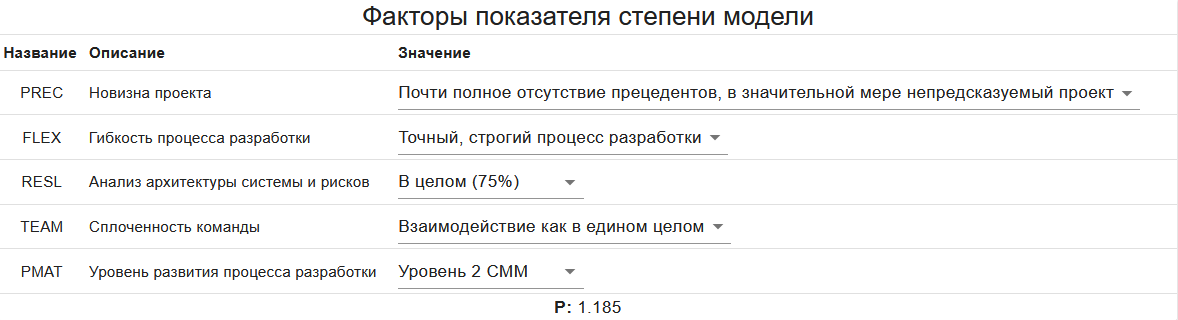


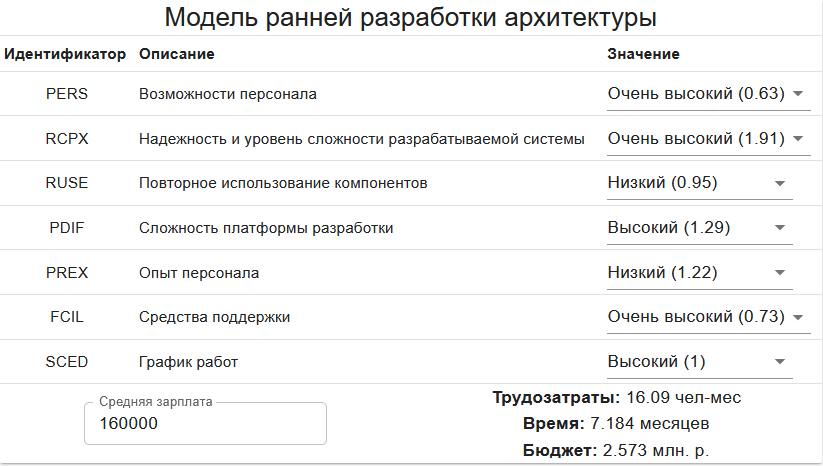
Число функциональных точек – 47

Нормированное число функциональных точек – 47,94

Количество инструкций – 3590

### **Оценка по методике COCOMO 2**

****

****

P=1.185

Трудозатраты – 16 чел-мес

Время – 7 месяцев

Бюджет – 2.57 млн. р.

**Модель композиции приложения:**

Формы:

* Страница регистрации – простая (обращение к БД)
* Страница авторизации – простая (обращение к БД)
* Страница штрафов – средняя (обращение к БД ГИБДД)
* Страница информации о пользователе – простая (обращение к БД)

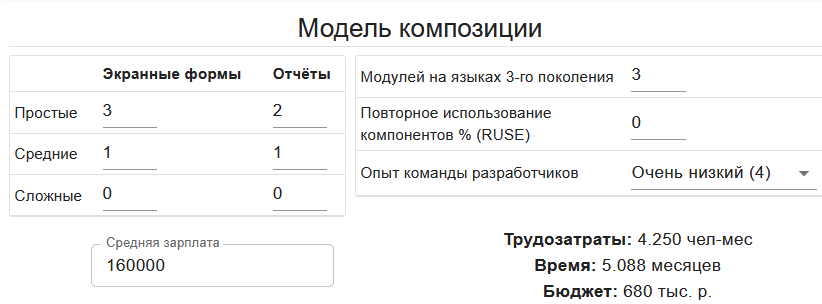
Отчёты:

* Отчёт о пользователях – простой (обращение к БД)
* Отчёт о штрафах – средний (обращение к БД ГИБДД)
* Отчёт о платежах – простой (обращение к БД)

Итог:

* Простые формы – 3
* Средние формы – 1
* Простые отчёты – 2
* Средние отчёты – 1
* Модули на ЯП третьего поколения – 3
* Повторное использование – 0%
* Опыт команды – очень низкий

Результат расчётов:



Трудозатраты – 4.25 чел-мес

Время – 5 месяцев

Бюджет – 680 тыс. р

**Вывод**

В ходе выполнения данной работы был разработан инструмент для определения трудозатрат и времени разработки проекта методом COCOMO2. Также, был выполнен анализ выданного задания, а именно:

* рассчитаны функциональные точки;
* рассчитан показатель степени модели (p);
* были определены факторы, влияющие на показатель степени;
* произведен расчет трудозатрат и времени по модели ранней разработки архитектуры приложения и модели композиции приложения.

В итоге было выяснено, что модель композиции приложения дает намного более оптимистичный прогноз, по сравнению с моделью ранней архитектуры приложения. Связано это с тем, что в модели композиции приложения не учитывается информация о персонале, работающем над проектом в отличие от модели ранней архитектуры. Таким образом, можно предположить, что данная модель дает приближенный результат при условии наивысшей опытности команды и при идеальном протекании работы над проектом.