|  |  |
| --- | --- |
|  | **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**

**По дисциплине «Экономика и программная инженерии»**

**НА ТЕМУ:**

***«Оценка параметров программного проекта с использованием метода функциональных точек и модели COCOMO II»***

**Вариант 2**

Студент **ИУ7-86Б** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мансуров В. М.**

Студент **ИУ7-86Б** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Виноградов А. О.**

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

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

**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 – 2

5. Транзакционная нагрузка – 3

6. Интенсивность взаимодействия с пользователем (оперативный ввод данных) – 4

7. Эргономические характеристики, влияющие на эффективность работы конечных пользователей - 1

8. Оперативное обновление – 4

9. Сложность обработки – 4

10. Повторное использование – 0

11. Легкость инсталляции – 1

12. Легкость эксплуатации/администрирования – 2

13. Портируемость – 2

14. Гибкость- 2

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 С#, а третий в объеме 25% кода - на Java.

**Страница авторизации**

****

* + 1. **Внутренний логический файл (ILF):**

1. Таблица БД для пользователя:
   1. DET = 2 (логин, пароль)
   2. RET = 1

Уровень: низкий (7)

1. Локальный файл для запоминания пользователя:
   1. DET = 2 (логин, пароль);
   2. RET = 1

Уровень: низкий (7)

* + 1. **Внешний ввод (EI):**

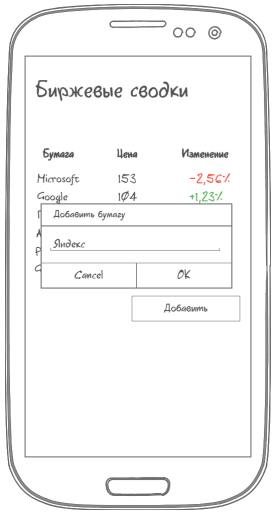
1. Форма авторизации:
   1. DET = 4 (логин, пароль, кнопка, флажок)
   2. FTR = 1

Уровень: низкий (3)

**Итого:**

2 – ILF низкого уровня(7),  
1 – EI низкого уровня(3)

**Страница «Биржевые сводки»**



* + 1. **Внутренний логический файл (ILF):**

1. Таблица БД для бумаги:
   1. DET = 1 (название)
   2. RET = 1

Уровень: низкий (7)

* + 1. **Внешний интерфейсный файл (EIF):**

1. Таблица БД для бумаги во внешнем сервисе:
   1. DET = 3 (название, цена, изменение цены)
   2. RET = 1

Уровень: низкий (5)

**Внешний вывод (EO):**

1. Вывод ценных бумаг:
   1. DET = 3 (название, цена, изменение цены)
   2. FTR = 2 (обращение к внутреннему файлу с именами бумаг, к внешнему с информацией о цене и изменении)

Уровень: низкий (4)

**Внешний ввод (EI):**

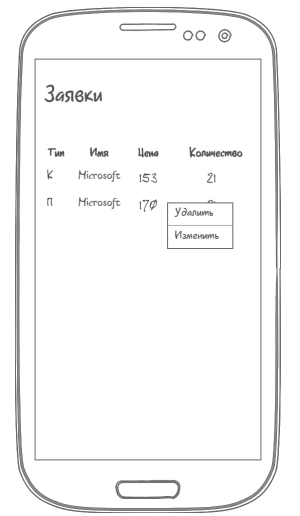
1. Добавление новой бумаги:
   1. DET = 3 (текстовое поле, кнопка Cancel, кнопка OK)
   2. FTR = 1

Уровень: низкий (3)

**Итого:**

1 – ILF низкого уровня (7)  
1 – EIF низкого уровня (5)  
1 – EO низкого уровня (4)  
1 – EI низкого уровня (3)

**Страница «Заявки»**

****

* + 1. **Внутренний логический файл (ILF):**

1. Таблица БД для заявки:
   1. DET = 4 (тип, имя, цена, количество)
   2. RET = 1

Уровень: низкий (7)

**Внешний вывод (EO):**

1. Вывод заявок:
   1. DET = 4 (тип, имя, цена, количество)
   2. FTR = 1 (обращение к внутреннему файлу с заявками)

Уровень: низкий (4)

**Внешний ввод (EI):**

1. Удалить:
   1. DET = 5 (тип, имя, цена, количество, кнопка)
   2. FTR = 1

Уровень: низкий (3)

1. Изменить:
   1. DET = 5 (тип, имя, цена, количество, кнопка)
   2. FTR = 1

Уровень: низкий (3)

**Итого:**

1 – ILF низкого уровня (7)   
1 – EO низкого уровня (4)  
2 – EI низкого уровня (3)

**Страница «Новая заявка»**



* + 1. **Внутренний логический файл (ILF):**

Используется уже учтенная таблица БД «Заявки»

**Внешний ввод (EI):**

1. Удалить:
   1. DET = 5 (бумага, цена, количество, покупка, кнопка)
   2. FTR = 1

Уровень: низкий (3)

**Итого:**

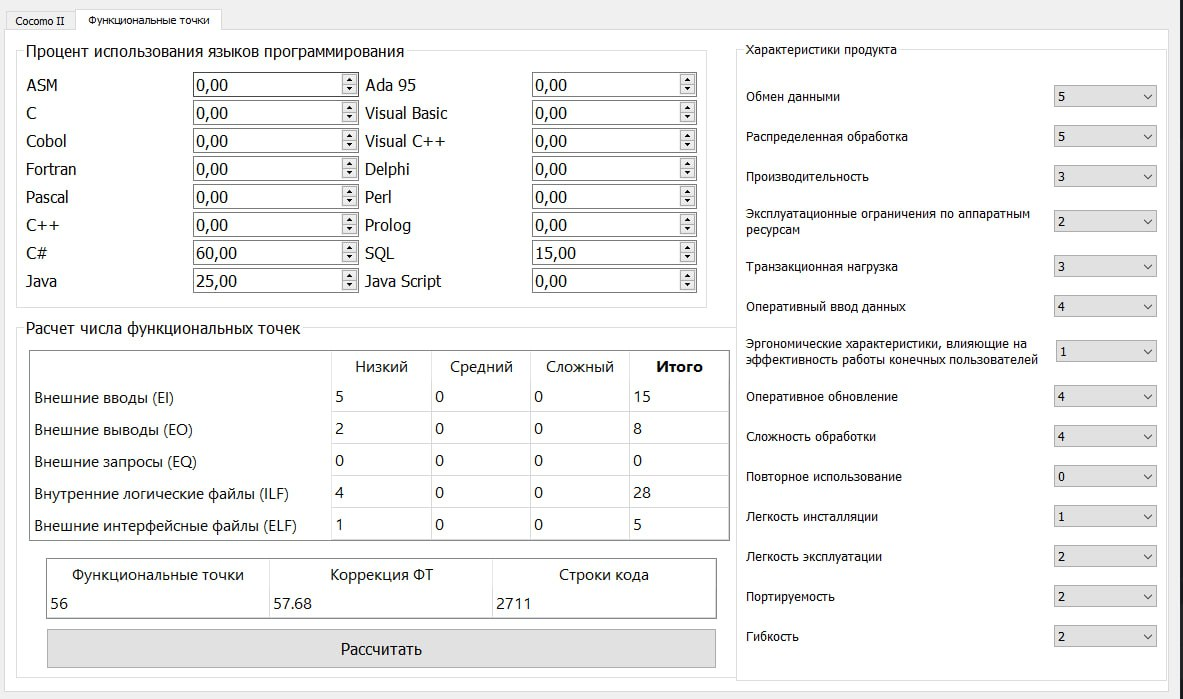
1 – EI низкого уровня (3)

**Расчеты**

Итого по всем страницам:

* 4 ILF низкого уровня
* 1 EIF низкого уровня
* 2 EO низкого уровня
* 5 EI низкого уровня

Далее приведены результаты расчетов числа функциональных точек.



Результаты:

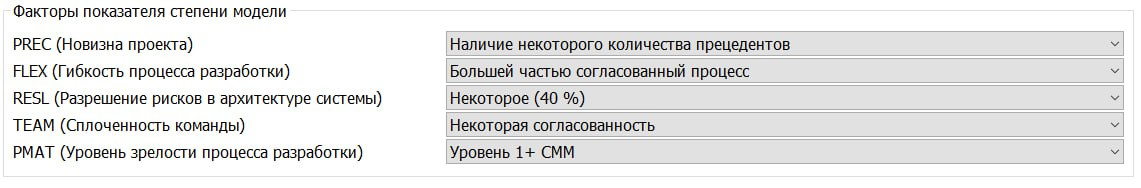
* Количество функциональный точек – 57,7
* Количество строк кода – 2711 LOC

### **Оценка по методике COCOMO 2**

Для реализации проекта была сформирована новая команда разработчиков, у отдельных членов которой имеется некоторый опыт создания систем подобного типа. В целях сплочения команды были проведены определенные мероприятия, что обеспечило на старте проекта приемлемую коммуникацию внутри коллектива. Заказчик не настаивает на жесткой регламентации процесса, однако график реализации проекта довольно жесткий. Несмотря на то, что предметная область является для разработчиков относительно новой, анализу архитектурных рисков было уделено лишь некоторое внимание. Организация только начинает внедрять методы управления проектами и формальные методы оценки качества процесса разработки

**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+ СММ, так как организация только начинает внедрять методы управления проектами и формальные методы оценки качества процесса разработки.



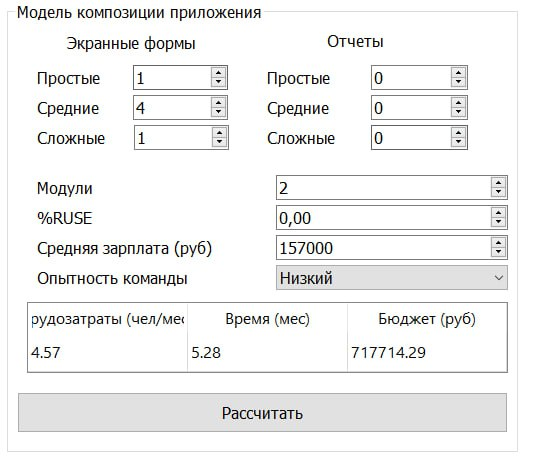
**Модель композиции приложения:**

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

Итого:

1 простая форма, 4 формы средней сложности, 1 сложная форма. В проекте два модуля, написанных на языках третьего поколения (C# и Java).

Ниже приведена оценка трудозатрат и длительности разработки с использованием модели ранней разработки архитектуры.



Результаты:

* Время разработки – 5,28 месяцев
* Трудозатраты – 4,57 человекомесяцев
* Бюджет – 717 714 рублей

Численность команды = 4,57 / 5,28 = 1 разработчик

**Модель ранней разработки:**

### **Показатели проекта:**

### RCPX – очень высокие;

### RUSE – переиспользование не предусматривается;

### PERS – средние способности персонала;

### PREX – низкий опыт разработки подобных систем;

### PDIF – высокая сложность платформы;

### FCIL – очень интенсивное использование инструментальных средств поддержки;

### SCED – жесткий график.

### **Результат**

### 

### Время – 8,65 месяцев

### Трудозатраты – 17,23 человекомесяцев

### Бюджет – 2 705 265 рублей

Численность команды = 17,23 / 8,65 = 2 разработчика

**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.