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»

Вариант 1

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

Студент: **Нгуен Ф. С. , Нгуен Т. Т.**

Группа: ИУ7И – 86Б

Оценка (баллы): Барышникова М.Ю.

Силантьева А.В.

2021г.

Цель работы: Целью лабораторной работы является продолжение знакомства с существующими методиками предварительной оценки параметров программного проекта и практическая оценка затрат по модели COCOMO II.

1. Анализ

1.1. 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, EO 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 коэффицентов регулировки сложности:

$$FP = \text{Общееколичество} * (0.65 + 0.01 * \sum Fi),$$

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

№	Системный параметр	Описание
1	Передача данных	Сколько средств связи требуется для передачи или обмена информацией с приложением или системой?
2	Распределенная обработка данных	Как обрабатываются распределенные данные и функции обработки?
3	Производительность	Нуждается ли пользователь в фиксации времени ответа или производительности?
4	Эксплуатационные ограничения	Насколько сильны эксплуатационные ограничения и каков объем специальных усилий на их преодоление?
5	Частота транзакций	Как часто выполняются транзакции (каждый день, каждую неделю, каждый месяц) ?
6	Оперативный ввод данных	Какой процент информации надо вводить в режиме онлайн?
7	Эффективность работы конечных пользователей	Приложение проектировалось для обеспечения эффективной работы конечного пользователя?
8	Оперативное обновление	Как много внутренних файлов обновляется в онлайновой транзакции?
9	Сложность обработки	Выполняет ли приложение интенсивную логическую или математическую обработку?
10	Повторная используемость	Приложение разрабатывалось для удовлетворения требований одного или многих пользователей?
11	Легкость инсталляции	Насколько трудны преобразование и инсталляция приложения?
12	Легкость эксплуатации	Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления?
13	Количество возможных установок на различных платформах	Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций?
14	Простота изменений (гибкость)	Была ли спроектирована, разработана и поддержана в приложении простота изменений?

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.

Язык программирования	Количество операторов на один FP
Ассемблер	320
C	128
Кобол	106
Фортран	106
Паскаль	90
C++	53
Java / C#	53
Ada 95	49
Visual Basic 6	24
Visual C++	34
Delphi Pascal	29
Perl	21
Prolog	54

Рис 2. Пересчет FP-оценок в LOC оценки

1.2. COCOMO II

COCOMO II рассматривает три различные модели оценки стоимости

- Модель композиции приложения
- Модель ранней разработки архитектуры.
- Пост архитектурная модель

Время в этой модели считается так:

$$\text{Время} = 3.0 * (\text{Трудозатраты})^{(0.33+0.2*(p-1.01))}$$

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

1.2.1. 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:

$$NOP = (\text{Объектныеточки}) * [(100 - \%RUSE)/100]$$

Затем считаются трудозатраты:

$$\text{ТРУДОЗАТРАТЫ} = NOP/PROD$$

где PROD – оценка скорости разработки

1.2.2. 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:

$$\text{Трудозатраты} = 2.45 * EArch * (\text{Размер})^{p},$$

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 экспертом, либо его можно вычислить путем комбинирования значений более детализированных показателей, которые используются на пост архитектурном уровне.

2. Задание

2.1. 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 – 3
12. Легкость эксплуатации/администрирования – 3
13. Портируемость – 5
14. Гибкость – 0

Внутренний логический файл ILF = 2:

Пользователь

- RET = 2 (строки и номер)
- DET = 6 (id, логин, пароль, тип, регистрационный номер водительского удостоверения, номер банковской карты)
- Уровень сложности: низкий (Page 14 leksi)

Транзакции

- RET = 2 (строки и номер)
- DET = 3 (номер карты, номер счета, сумма транзакции)
- Уровень сложности: низкий

Внешний интерфейсный файл EIF = 1:

Штрафы

- RET = 2 (строки и номер)
- DET = 6 (номер постановления, даат постановления, имя, фамилия, отчество, сумма штрафа)
- Уровень сложности: низкий

Внешний ввод EI = 5:

Регистрация (мобильное приложение и веб портал)

- FTR = 1 (пользователь)
- DET = 5 (элементы данных: логин, пароль, номер водительского удостоверения, номер банковской карты, кнопка)
- Уровень сложности: низкий.

Оплатить штраф (мобильное приложение и веб портал)

- FTR = 1 (штраф)
- DET = 7 (элементы данных: номер постановления, дата постановления, имя, фамилия, отчество нарушителя, сумма штрафа, кнопка)
- уровень сложности: низкий

Добавление пользователей в бд (веб-портал)

- FTR = 1 (пользователь)
- DET = 5 (элементы данных: логин, пароль, номер водительского удостоверения, номер банковской карты, кнопка)
- уровень сложности: низкий

Внешний вывод EO = 1:

Сообщение о результате оплаты (мобильное приложение и веб портал)

- FTR = 1 (транзакция)
- DET = 3 (элементы данных: номер карты, сумма, результат)
- уровень сложности: низкий

Внешний запрос EQ = 3:

Получение списка штрафов (мобильное приложение и веб портал)

- FTR = 1 (штраф)
- DET = 6 (элементы данных: номер постановления, дата постановления, имя, фамилия, отчество нарушителя, сумма штрафа)
- уровень сложности: низкий

Получение списка пользователей (веб портал)

- FTR = 1 (пользователь)
- DET = 5 (элементы данных: логин, пароль, номер водительского удостоверения, номер банковской карты, кнопка)
- уровень сложности: низкий

Количество функциональных точек:

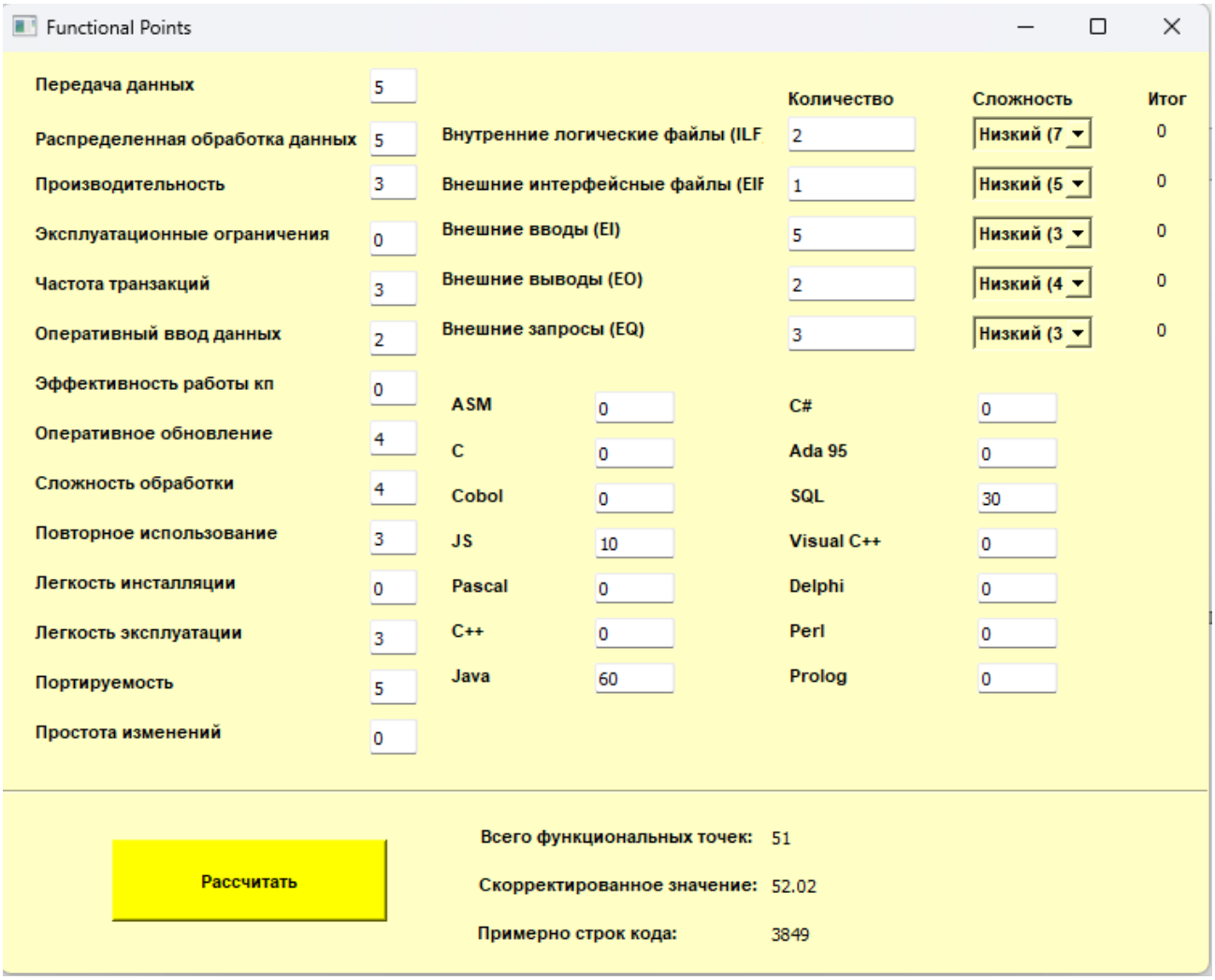
Характеристика	Количество	Уровень сложности	Итого
EI	5	3	15
EO	2	4	8
EQ	3	3	9
ILF	2	7	14
EIF	1	5	5

⇨ Количество функциональных точек: 51

Коэффицентов регулировки сложности:

$$FP = \text{Общее кол.} * \left(0.65 + 0.01 * \sum Fi\right) = 51 * (0.65 + 0.01 * 37) = 52.02$$

Пересчет FP-оценок в LOC-оценки:



2.2. Оценка по методике COCOMO II

Показатели проекта:

- Новизна проекта (PREC) – Почти полное отсутствие прецедентов, в значительной мере непредсказуемый проект (т.к. практически отсутствует опыт в разработке систем подобного типа).

- Гибкость процесса разработки (FLEX) – Некоторые послабления в процессе (довольно строгом процессе с периодической демонстрацией рабочих продуктов, соответствующих этапам жизненного цикла.).
- Разрешение рисков в архитектуре системы (RESL) – почти полное (детальный анализ рисков).
- Сплоченность команды (TEAM) – довольно слаженная (повышенная согласованность)
- Уровень развития процесса разработки (PMAT) – Уровень 2 CMM (чуть выше второго уровня зрелости процессов разработки.).

Модель композиции приложения:

- Простые экранные формы = 8
- Отчеты = 0
- Также имеются 5 модулей, написанные на ЯП третьего поколения.
- Повторное использование отсутствует.
- Опытность команды – низкая

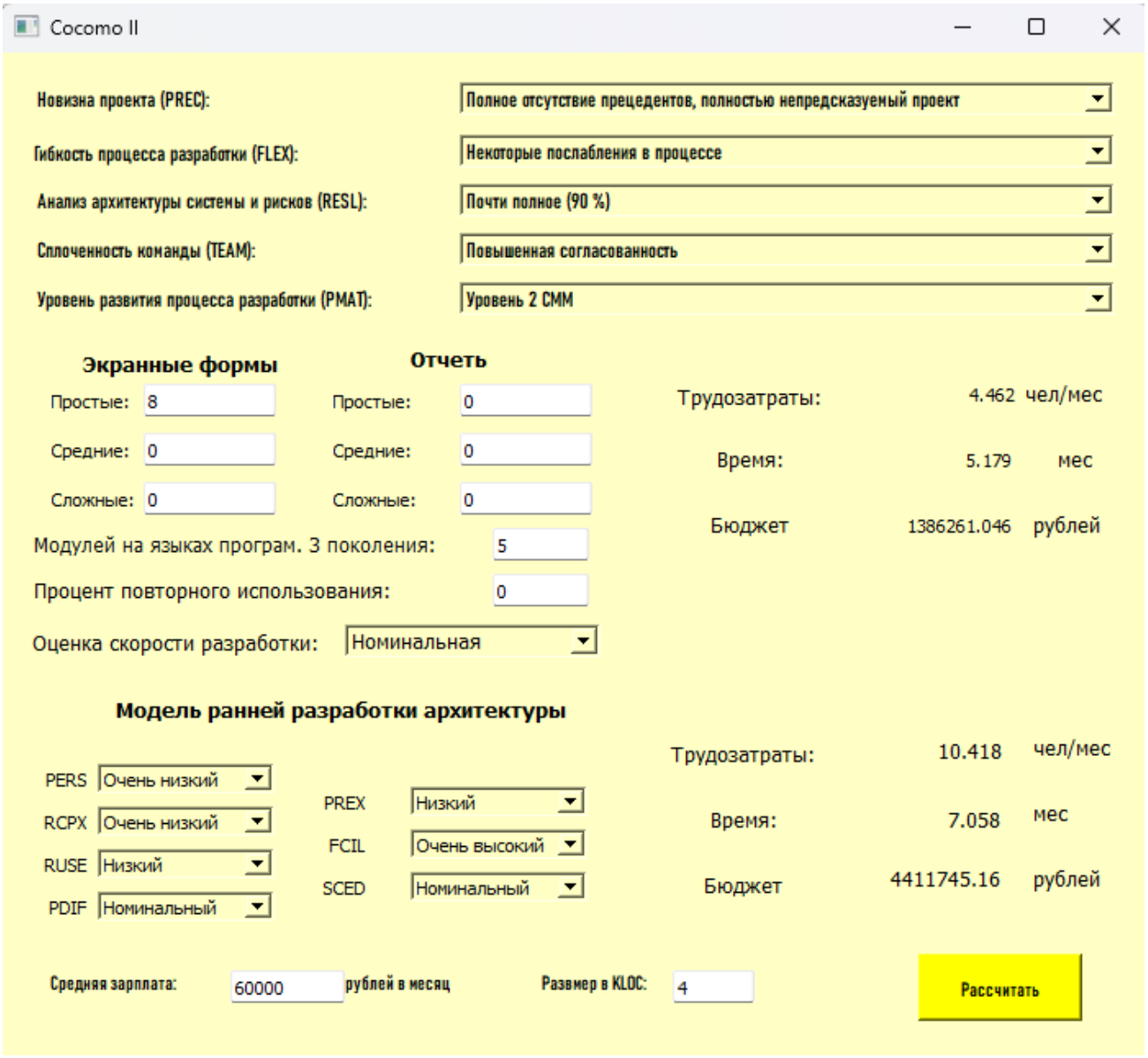
Результат – см. *Рис. 2.1.*

Рассчитаем модель ранней разработки архитектуры:

- PERS (возможности персонала) – очень высокий (Возможности персонала можно охарактеризовать как очень высокие)
- RCPX (надежность и уровень сложности разрабатываемой системы) – очень высокий (Надежность и уровень сложности разрабатываемой системы оцениваются как очень высокие)
- RUSE (повторное использование компонентов) – низкий (проект не предусматривает специальных усилий на повторное использование компонентов)

- PDIF (сложность платформы разработки) – номинальный (Сложность платформы (PDIF) средняя)
- PREX (опыт персонал) – низкий (опыт членов команды в данной сфере является скорее низким)
- FCIL (средства поддержки) – очень высокий (интенсивное использование инструментальных средств поддержки)
- SCED (график работ) – номинальный (Заказчик не настаивает на жестком графике)

Результат – см. *Рис. 2.1.*


Cocomo II
Новизна проекта (PREC):
Полное отсутствие прецедентов, полностью непредсказуемый проект
Гибкость процесса разработки (FLEX):
Некоторые послабления в процессе
Анализ архитектуры системы и рисков (RESL):
Почти полное (90 %)
Сплоченность команды (TEAM):
Повышенная согласованность
Уровень развития процесса разработки (PMAT):
Уровень 2 CMM
Экранные формы
Простые: 8
Средние: 0
Сложные: 0
Отчеть
Простые: 0
Средние: 0
Сложные: 0
Модулей на языках програм. 3 поколения: 5
Процент повторного использования: 0
Оценка скорости разработки: Номинальная
Трудозатраты: 4.462 чел/мес
Время: 5.179 мес
Бюджет 1386261.046 рублей
Модель ранней разработки архитектуры
PERS Очень низкий
RCPX Очень низкий
RUSE Низкий
PDIF Номинальный
PREX Низкий
FCIL Очень высокий
SCED Номинальный
Трудозатраты: 10.418 чел/мес
Время: 7.058 мес
Бюджет 4411745.16 рублей
Средняя зарплата: 60000 рублей в месяц
Развмер в KLOC: 4
Рассчитать

Вывод

Методика функциональных точек позволяет без серьезных трудозатрат оценить количество строк в проекте еще до начала его реализации. Методика COCOMO II позволяет оценить сроки и стоимость разработки проекта на разных его этапах. Чем больше информации известно о проекте, тем выше будет точность данной оценки.

По результатам расчетов можно заметить, что разность между результатами расчетов по разным моделям очень велика. Модель композиции приложения показала существенно меньшие трудозатрат на выполнение проекта, чем модель с ранней архитектуры приложения. Планируемый бюджет оказался намного меньше. Связано это с тем, что в модели композиции приложения не учитывается информация о персонале, работающем над проектом в отличие от модели ранней архитектуры.

Исходя из этих данных, можно утверждать, что данная модель дает наилучший результат в рамках условия, когда команда имеет наивысшую опытность, а также учитывая идеальное протекание работы над проектом.