|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 № 7**

**Вариант 1**

|  |  |
| --- | --- |
| **Тема** “Оценка параметров программного проекта с использованием метода функциональных точек и модели COCOMO II”  **Студент** Сушина А.Д., Романов В.А, Будкин Г.С,  **Группа** ИУ7-81б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Барышникова М. Ю. |  |

Москва.

2021 г

# Цель работы

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

# Ход работы

## 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 коэффицентов 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 |

## COCOMO II

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 числе 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.

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

При разработке ПО 30% кода будет написано на SQL, 10 % - на JavaScript, 60% - на Java.

* + 1. **Вычислим ILF (внутренний логический файл):**

Пользователь

* RET = 2 (строки и номер)
* DET = 6 (id, логин, пароль, тип, регистрационный номер водительского удостоверения, номер банковской карты )
* Уровень сложности — низкий.

Транзакции

* RET = 2 (строки и номера)
* DET = 3 (номер карты, номер счета, сумма оплаты)
* Уровень сложности — низкий.

**Вычислим EIF (внешний интерфейсный файл):**

Штрафы

* RET = 2 (строки и номера)
* DET = 6 (номер постановления, дата постановления, имя, фамилия, отчество нарушителя, сумма штрафа)
* Уровень сложности — низкий.

### Вычислим EI (внешний ввод):

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

### Вычислим EO (Внешний вывод):

Сообщение о результате оплаты

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

### Вычислим EQ (Внешний запрос):

Получение списка штрафов

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

Получение списка пользователей

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

Посчитаем количество строк кода с этими значениями. Для этого заполним поля разработанного приложения и произведем расчет. На рисунке 3 представлен результат:

* Количество функциональных точек — 51
* Количество строк кода — 2742 SLOC

|  |
| --- |
| Рис 3. Расчет количества строк кода |

### **Оценка по методике COCOMO II**

**Определим 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 СММ (чуть выше второго уровня зрелости процессов разработки. ).

**Рассчитаем модель композиции приложения**:

* Простые экранные формы = 8
* Отчеты = 3 (простые) + 1 средний
* Также имеются 5 модулей, написанные на ЯП третьего поколения.
* Повторное использование = 0%
* Опытность команды – низкая

Заполняем необходимые поля в приложении и получаем результат:

* Трудозатраты — 9.857
* Длительность — 6.877
* Численность команды разработчиков - 2
* Бюджет — 591 428

|  |
| --- |
| Рис 4. Результат подсчета для композиционной модели |

**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 )

Заполняем необходимые поля в приложении и получаем результат:

* Трудозатраты — 8.142
* Длительность — 6.417
* Численность команды разработчиков — 2
* Бюджет — 488 4­98

|  |
| --- |
| Рис 5. Результат для модели ранней архитектуры |

# 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 сказать, что данная модель дает идеальный результат при условии наивысшей опытности команды и при идеальном протекании работы над проектом