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
по курсу «Экономика программной инженерии»

Студент Кривозубов В.О., Соколов Е.М.

Группа ИУ7-83Б

Оценка (баллы) _____

Преподаватель Барышникова М. Ю., Силантьева А. В.

Москва, 2022 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.

Теоретическая часть

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:

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

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:

1) Внешний ввод (EI, транзакция, получающая данные от пользователя) —элементарный процесс, перемещающий данные из внешней среды в приложение. Данные могут поступать с экрана ввода или из другого приложения. Данные могут содержать как управляющую, так и деловую

информацию. Обрабатываемые данные могут соответствовать одному или нескольким внутренним логическим файлам.

2) Внешний вывод (EO, транзакция передающая данные пользователю) —элементарный процесс, перемещающий данные, вычисленные в приложении, во внешнюю среду и связанный с созданием и/или обработ2 кой выходной информации приложения —выходного отчета, документа, экранной формы.
3) Внешний запрос (EQ, интерактивный диалог с пользователем, требующий от него каких-либо действий) —элементарный процесс, состоящий из комбинации «запрос/ответ», не связанный с вычислением производных данных или обновлением внутренних логических файлов (базы данных).
4) Внутренний логический файл(ILF, информация, которая используется во внутренних взаимодействиях системы)—выделяемые пользователем логически связанные группы данных или блоки управляющей информации, которые поддерживаются внутри продукта и обслуживаются через внешние вводы.
5) Внешний интерфейсный файл (EIF, файлы, участвующие во внешних взаимодействиях с другими системами)—выделяемые пользователем логически связанные группы данных или блоки управляющей информации, на которые ссылается продукт, но которые поддерживаются вне продукта.

Расчет скорректированного количества функциональных точек:

$$FP = \text{Общее количество} \cdot (0.65 + 0.01 \cdot \Sigma Fi),$$

где Fi — 14 коэффициентов регулировки сложности.

Для пересчета FP-оценок в LOC-оценки используется количество операторов на один FP для конкретного ЯП (для ассемблера 320).

В модели COCOMO II используются три модели оценки стоимости:

1) Модель композиции приложения –это модель, которая подходит для проектов, созданных с помощью современных

инструментальных средств. Единицей измерения служит объектная точка.

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.

3) Постархитектурная модель –наиболее детализированная модель COCOMO II, которая используется после разработки архитектуры проекта. В состав этой модели включены новые драйверы затрат, новые правила подсчета строк кода, а также новые уравнения.

Модель композиции приложения используется на ранней стадии конструирования ПО, когда:

1) рассматривается макетирование пользовательских интерфейсов
2) оценивается производительность
3) 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.

- NOP = (Объектные точки) * [(100 - %RUSE) / 100] – новые объектные точки;
- PROD – оценка скорости разработки, зависит от опытности команды разработчиков;
- Трудозатраты = NOP / PROD [чел./мес.];
- Время = 3.0 * (Трудозатраты)^(0.33 + 0.2(p - 1.01)).

В модели ранней разработки архитектуры:

- Трудозатраты = 2.45 * EArch * (Размер)^p;

- Размер - количество тысяч строк кода (KLOC);
- EArch = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED;
- Время = 3.0 * (Трудозатраты)^(0.33 + 0.2(p - 1.01)).

На показатель степени в модели COCOMO II влияет пять факторов:

- p = (PREC * FLEX * RESL * TEAM * PMAT) / 100 + 1.01.

Подсчет количества функциональных точек

В ПО имеется 2 внутренних логических файла (ILF)

Первый - для хранения информации о пользователях. Число типов элементов записей (RET) равно 2 (id - число, остальное - строки). Число типов элементов данных (DET) равно 6. Следовательно уровень сложности внутреннего логического файла - низкий.

Второй ILF имеет три элемента данных. Число типов элементов записей равно двум. Уровень сложности низкий.

ПО имеет один внешний интерфейсный файл. Число типов элементов записей (RET) для этого файла равно трем. Число типов элементов данных (DET) внутреннего логического файла будет равно шести. Таким образом, уровень сложности внутреннего логического файла – низкий.

Внешние вводы ПО:

1) Регистрация (мобильное приложение и веб портал). Ссылается на один внутренний логический файл и имеет четыре элемента данных. Уровень сложности - низкий.
2) Оплата штрафа (мобильное приложение и веб-портал). Ссылается на один внешний интерфейсный файл и имеет шесть элементов данных. Уровень сложности низкий.
3) Добавление пользователей в БД (веб-портал). Ссылается на один внутренний логический файл и имеет четыре элемента данных. Уровень сложности низкий.

4) Получение списка штрафов (веб-портал). Ссылается на один внешний интерфейсный файл и имеет семь элементов данных. Уровень сложности низкий.
5) Получение сообщения об успешном или неуспешном оплате штрафа от ГИБДД (веб-портал). Имеет один элемент данных и не ссылается на внутренние файлы. Уровень сложности низкий.
6) Ответ о результате оплаты от платежной системы (веб-портал). Имеет три элемента данных и ссылается на два внутренних файла. Уровень сложности низкий.

Внешние выводы ПО:

1) Вывод сообщения о положительном или отрицательном результате оплаты штрафа. Уровень сложности этого внешнего вывода – низкий, так как он имеет один DET и один FTR (веб портал + мобильное приложение).
2) Запрос на получение списка штрафов. Ссылается на один внешний интерфейсный файл и имеет один элемент данных. Уровень сложности низкий.у
3) Оповещение ГИБДД об оплате штрафа (веб-портал). Ссылается на два внутренних логических файла и имеет восемь элементов данных. Уровень сложности средний.
4) Запрос об оплате штрафа (веб-портал). Имеет три элемента данных и ссылается на два внутренних файла. Уровень сложности низкий.

Ввод данных для метода функциональных точек представлен на рисунке 1. Ввод данных для методики COCOMO II и результаты расчетов представлены на рисунках 2 - 3.

Количество простых экранных форм примем равным 8. Количество модулей, написанных на ЯП третьего уровня – пять (на Java и на JavaScript).

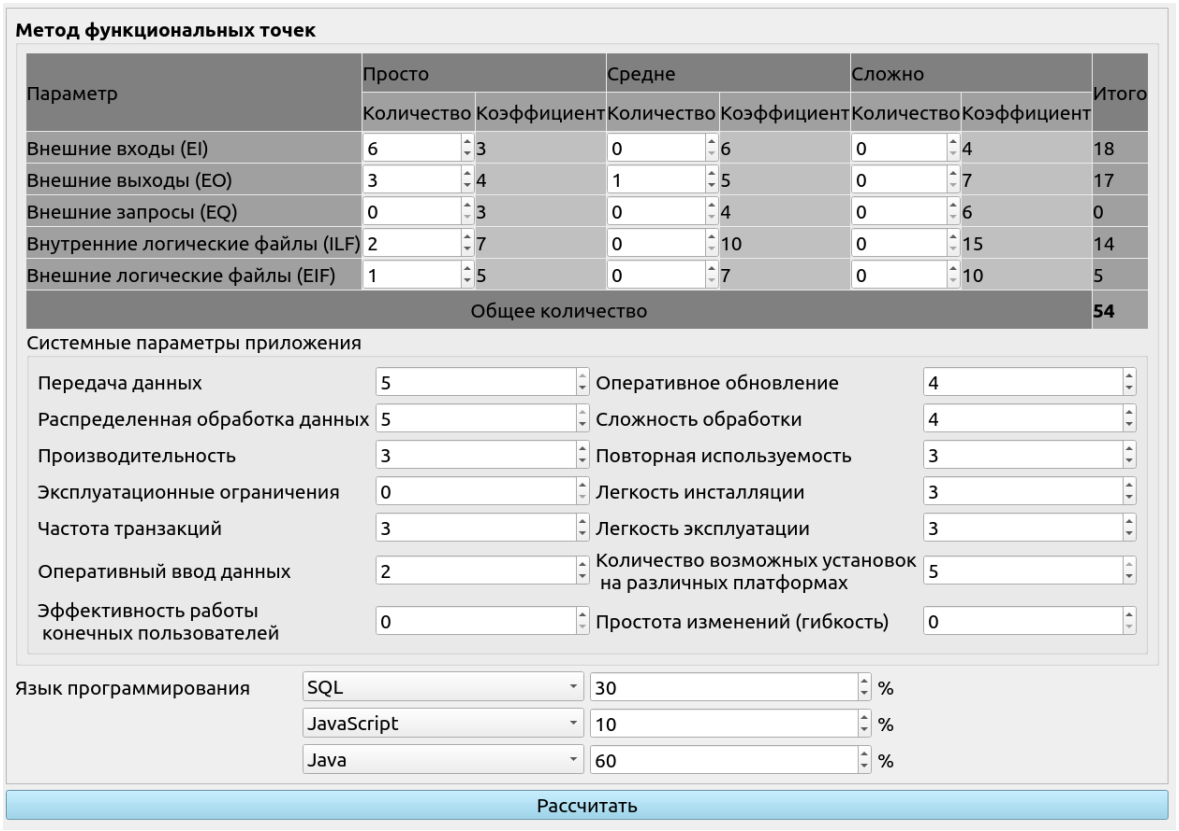

Рис. 1 – Ввод данных для метода функциональных точек

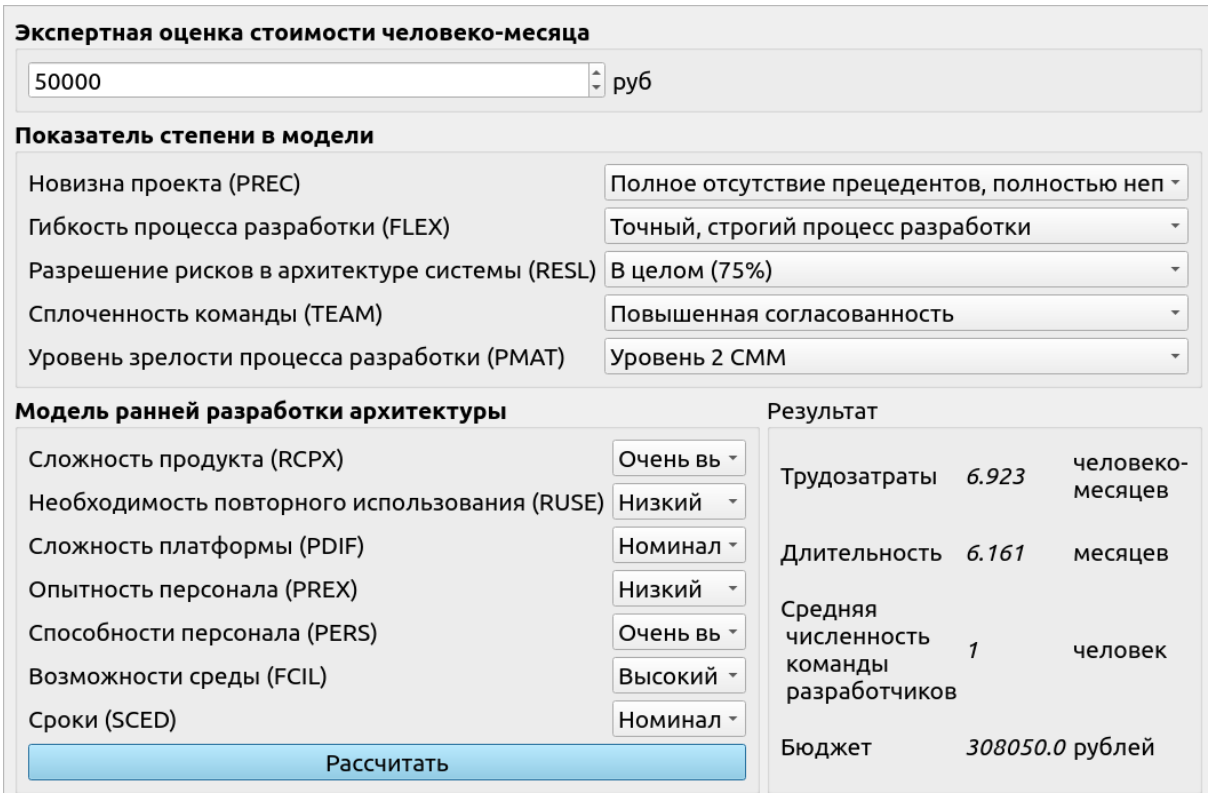

Рис. 2 – Ввод данных для методики COCOMO II (часть 1)

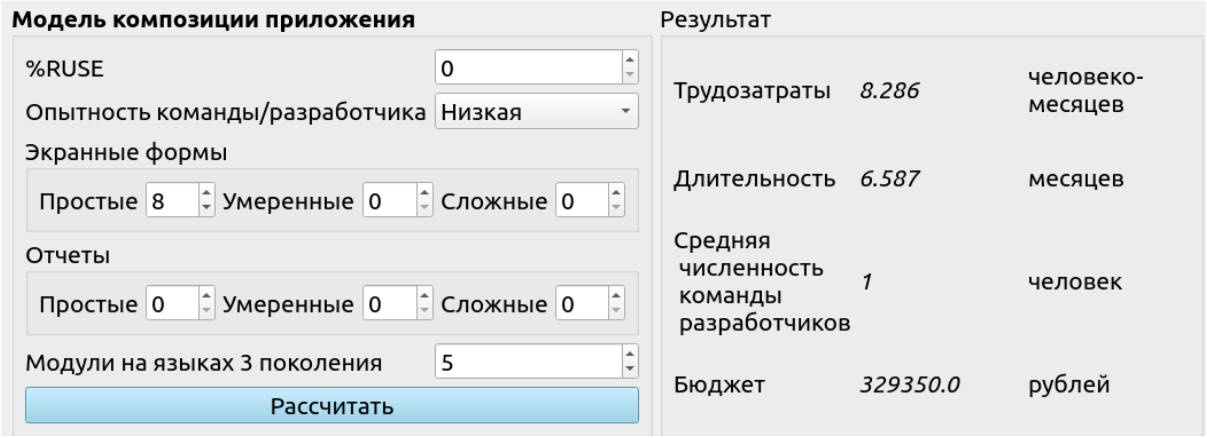

Рис. 3 – Ввод данных для методики COCOMO II (часть 2)

Вывод

Модель COCOMO II позволяет более полно учитывать факторы, влияющих на экономические характеристики производства сложных программных продуктов, а также учитывать уникальные факторы для корректировки экономических характеристик, связанные со специфическим проектом и организацией.

COCOMO II и метод функциональных точек предоставляет возможность оценить объем проекта если собственный опыт аналогичных проектов отсутствует, а экспертное мнение недоступно.