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

ОТЧЁТ

по лабораторной работе № 7

по курсу «Экономика программной инженерии»

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

Студент: Керимов А. Ш.

Группа: ИУ7-84Б

Вариант: 4

Преподаватель: Барышникова М. Ю.

Москва.
2021 г.

Цель работы

Продолжение знакомства с существующими методиками предварительной оценки параметров программного проекта и практическая оценка затрат по модели COCOMO II

Метод 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.

Информационные характеристики, используемые в методе функциональных точек

- *Количество внешних вводов*. Подсчитываются все вводы пользователя, по которым поступают разные прикладные данные, при этом вводы должны быть отделены от запросов, которые подсчитываются отдельно.
- *Количество внешних выводов*. Подсчитываются все выводы, по которым к пользователю поступают результаты, вычисленные программным приложением. В этом контексте выводы означают отчёты, экраны, распечатки, сообщения об ошибках. Индивидуальные единицы данных внутри отчёта отдельно не подсчитываются
- *Количество внешних запросов*. Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Подсчитываются все запросы — каждый учитывается отдельно

- *Количество внутренних логических* файлов. Подсчитываются все логические файлы (то есть логические группы данных, которые могут быть частью базы данных или отдельным файлом)
- *Количество внешних интерфейсных* файлов. Подсчитываются все логические файлы из других приложений, на которые ссылается данное приложение

Порядок расчёта трудоёмкости разработки ПО

- определение количества функциональных типов приложения
- определение количества связанных с каждым функциональным типом элементарных данных (DET), элементарных записей (RET) и файлов типа ссылок (FTR)
- определение сложности (в зависимости от количества DET, RET и FTR)
- подсчёт количества функциональных точек приложения
- подсчёт количества функциональных точек с учётом общих характеристик системы
- оценка трудоёмкости разработки (с использованием различных статистических данных)

Скорректированное количество функциональных точек

$$FP = \text{Общее количество} \cdot (0{,}65 + 0{,}01 \cdot \sum Fi),$$

где Fi — 14 коэффициентов регулировки сложности. Каждый коэффициент может принимать следующие значения:

0 — нет влияния,

1 — случайное влияние,

2 — небольшое влияние,

3 — среднее влияние,

4 — существенное влияние,

5 — сильное влияние

Определение системных параметров приложения

№	Системный параметр	Описание
1	Передача данных	Сколько средств связи требуется для передачи или обмена информацией с приложением или системой?
2	Распределённая обработка данных	Как обрабатываются распределённые 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	Лёгкость инсталляции	Насколько трудны преобразование и инсталляция приложения?
12	Лёгкость 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?

Пересчёт FP-оценок в LOC-оценки

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

Модель композиции приложения

Данная модель используется на ранней стадии конструирования ПО, когда:

- рассматривается макетирование пользовательских интерфейсов;
- оценивается производительность;
- определяется степень зрелости технологии.

Модель ориентирована на применение объектных точек. Объектная точка — средство косвенного измерения ПО. Подсчёт количества объектных точек производится с учётом количества экранов (как элементов пользовательского интерфейса), отчётов и компонентов, требуемых для построения приложения.

Правила подсчёта объектных точек

- Количество изображений на дисплее (экранных форм). Простые изображения принимаются за 1 объектную точку, изображения умеренной сложности принимаются за 2 точки, очень сложные изображения принято считать за 3 точки.
- Количество представленных отчётов. Для простых отчётов назначаются 2 объектные точки, умеренно сложным отчётам назначаются 5 точек. Написание сложных отчётов оценивается в 8 точек.
- Количество модулей, которые написаны на языках третьего поколения и разработаны в дополнение к коду, написанному на языке программирования четвёртого поколения. Каждый модуль на языке третьего поколения считается за 10 объектных точек.

Формулы

$$NOP = (\text{Объектные точки}) \cdot \left[\frac{100 - \%RUSE}{100}\right] \text{ — новые объектные точки,}$$

$$\text{ТРУДОЗАТРАТЫ} = \frac{NOP}{PROD}\ [\text{чел.} - \text{мес.}],$$

$PROD$ — оценка скорости разработчика.

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.

$$\text{Трудозатраты} = \mathbf{2,45} \cdot \boldsymbol{EArch} \cdot (\text{Размер})^{\boldsymbol{p}},$$

где Трудозатраты (работа) — число человеко-месяцев,

$$EArch = PERS \cdot RPCX \cdot RUSE \cdot PDIF \cdot PREX \cdot FCIL \cdot SCED,$$

Размер — KSLOC (предпочтительно для подсчёта KSLOC предварительно посчитать количество функциональных точек).

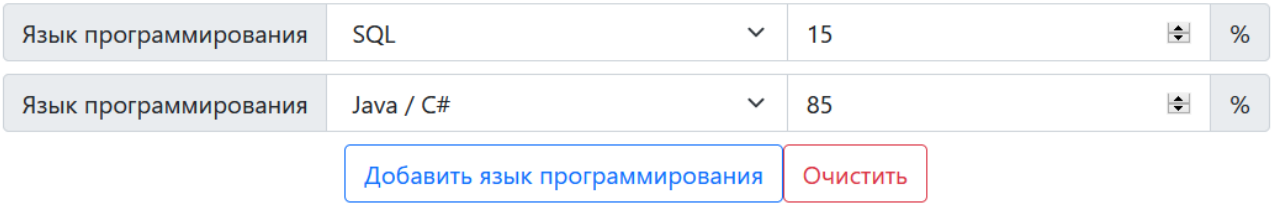
p — показатель степени.

$$\text{Время} = \mathbf{3,0} \cdot (\text{Трудозатраты})^{\mathbf{0,33+0,2}\cdot(\boldsymbol{p}-\mathbf{1.01})},$$

Задание

Компания получила заказ на разработку клиентского мобильного приложения брокерской системы. Программа позволяет просматривать актуальную биржевую информацию, производить сделки и отслеживать их выполнение. Приложение имеет 4 страницы: авторизация, биржевые сводки, заявки, новая заявка.

Разработанное ПО состоит из трёх компонентов. Первый компонент составляет по объёму примерно 15 % программного кода и будет написан на SQL, второй (около 60 % кода) - на C#, а третий в объёме 25 % кода — на Java.



Характеристики продукта:

1. Обмен данными — 5

2. Распределённая 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

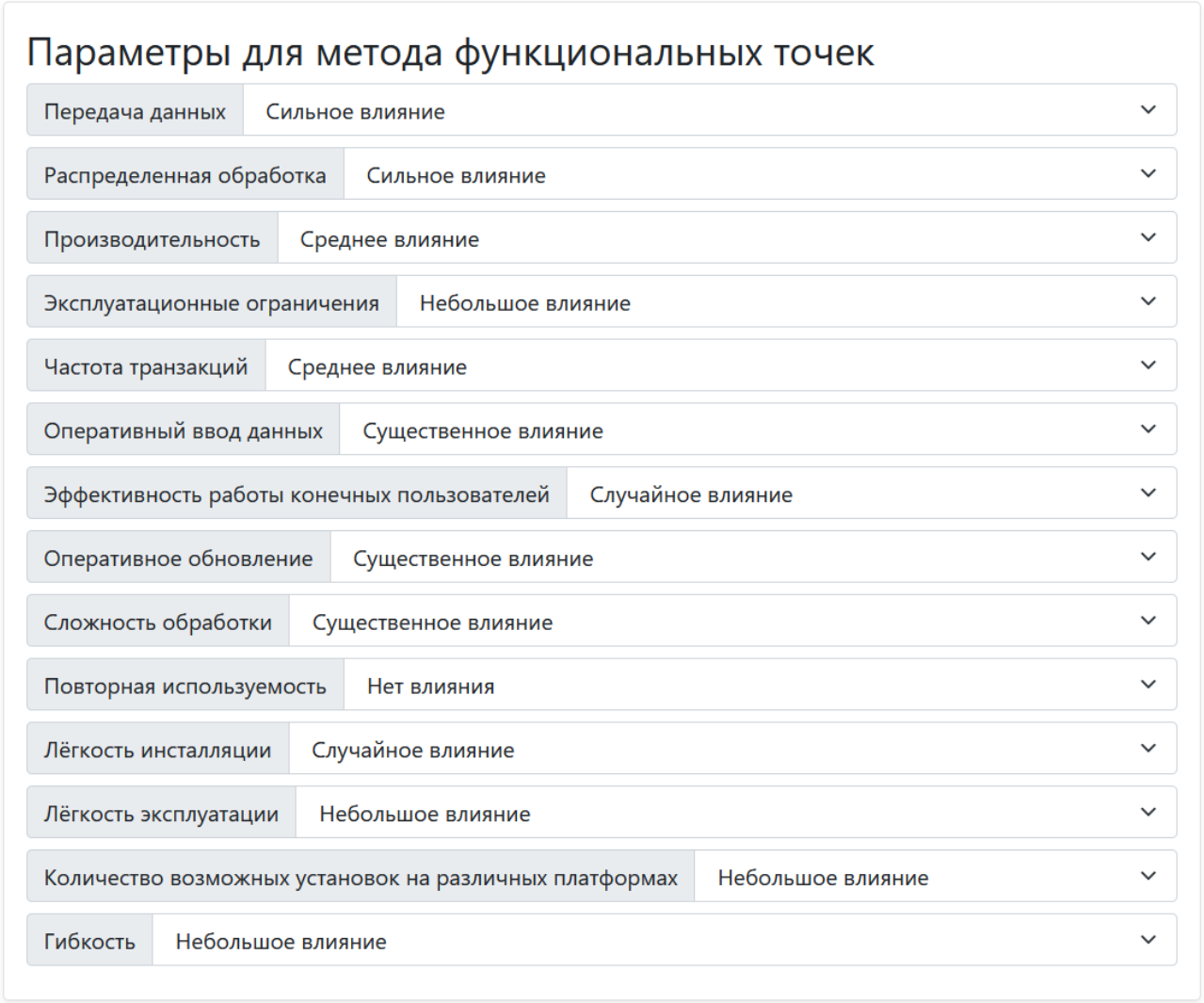
10. Повторное использование — 0

11. Лёгкость инсталляции — 1

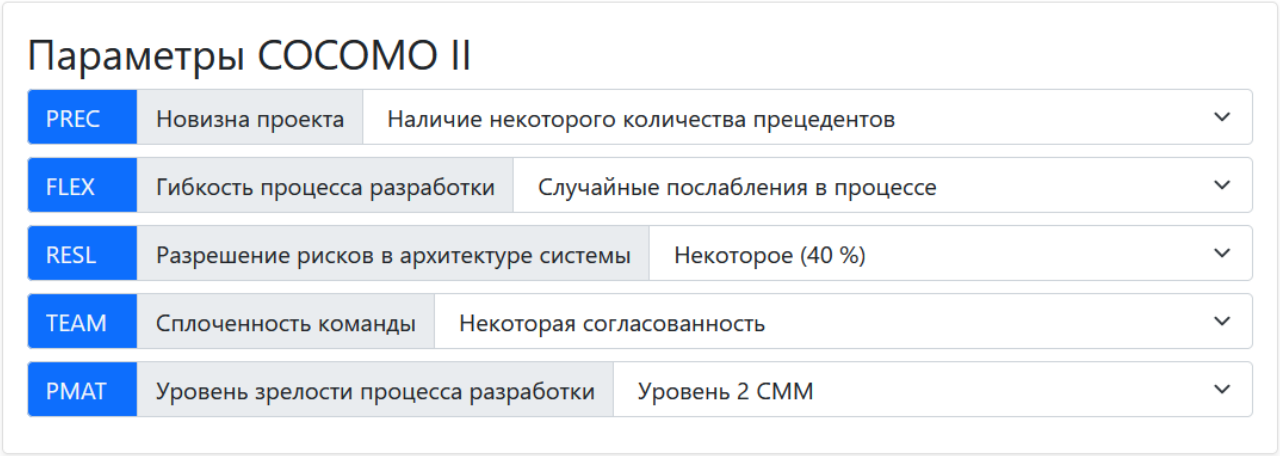
12. Лёгкость эксплуатации/администрирования — 2

13. Портируемость — 2

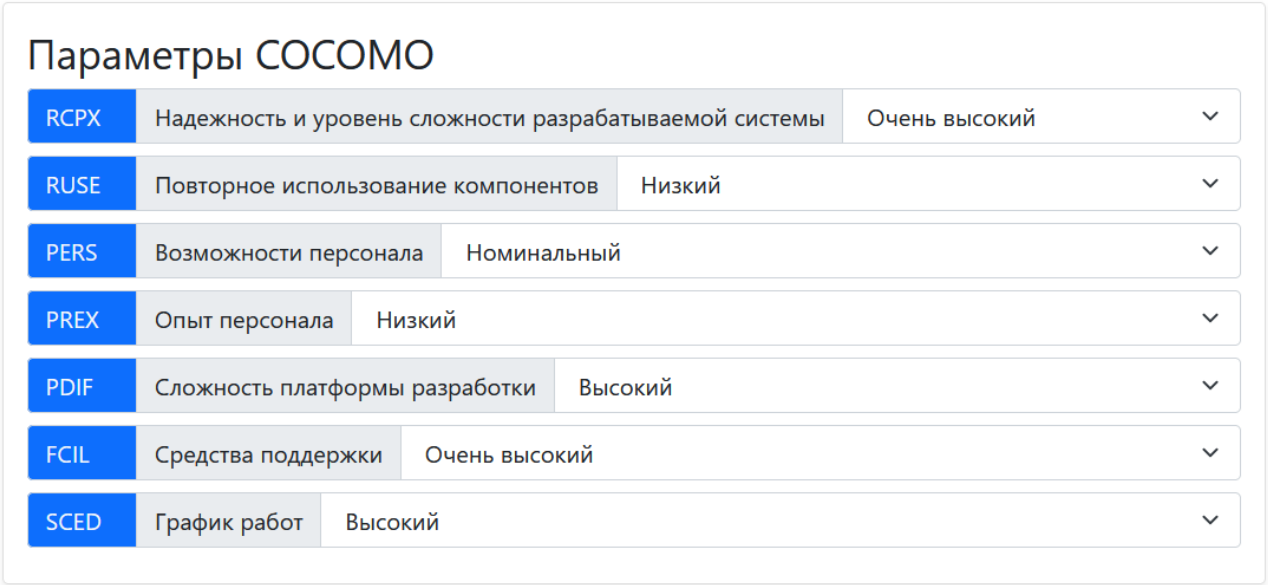
14. Гибкость — 2



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 жёсткой регламентации процесса, однако график реализации проекта довольно жёсткий. 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.



Надёжность и уровень сложности (RCPX) разрабатываемой системы оцениваются как очень высокие, повторного использования компонентов не предусматривается (RUSE). Возможности персонала (PERS) – средние, его опыт работы в разработке систем подобного типа (PREX) низкий. Сложность платформы (PDIF) высокая. Разработка предусматривает очень интенсивное использование инструментальных средств поддержки (FCIL). Заказчик настаивает на жёстком графике (SCED).



Выполнение

Модель композиции приложения

На странице авторизации 1 простая форма (2 текстовых поля и 1 чекбокс): 1 об. т.

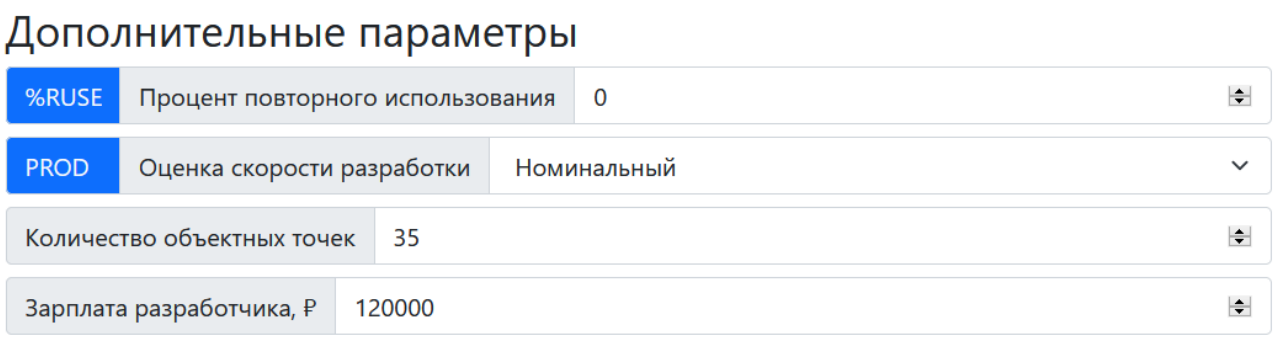
На странице биржи 1 простая форма (только 1 текстовое поле) и 1 отчёт умеренной сложности: 6 об. т.

На странице заявок 1 простая форм (только 1 кнопка) и 1 отчёт умеренной сложности (список из 4 колонок, данные для которых во внешнем ресурсе): 6 об. т.

На странице новой заявки 1 форма умеренной сложности (1 текстовое поле, 2 числовых поля, 1 switch, данные из формы отправляются во внешний ресурс): 2 об. т.

На языках 3-го поколения 2 модуля: 20 об. т.

Итого 35 объектных точек.



Модель композиции приложения

NOP	35
Трудозатраты, чел.-мес.	2.69
Время, мес.	4.34
Количество работников	1
Бюджет	520768.11

Модель ранней разработки архитектуры

1 внутренний файл — запомнить или нет логин и пароль (1 RET — булево значение, 1 DET — чекбокс запомнить или нет логин).

1 внутренний файл — логины и пароли (1 RET — текстовый тип, 2 DET — логин и пароль).

1 внутренний файл — информация о заявках (4 RET — булево значение типа заявки, текстовый имени ценной бумаги, вещественное число цены, целое число количества, 4 DET — тип, имя, цена и количество).

1 внутренний файл — данные об отслеживаемых акциях (1 RET — текстовый тип, 1 DET — имя акции).

1 внешний файл — информация о бирже (2 RET — текстовый для имени и вещественный для цены и изменения, 3 DET — бумага, цена, изменение).

Внешний ввод авторизации $FTR = 2$ — ILF запомнить или нет и ILF с логинами и паролями, $DET = 4$ — логин, пароль, чекбокс «запомнить», кнопка «войти».

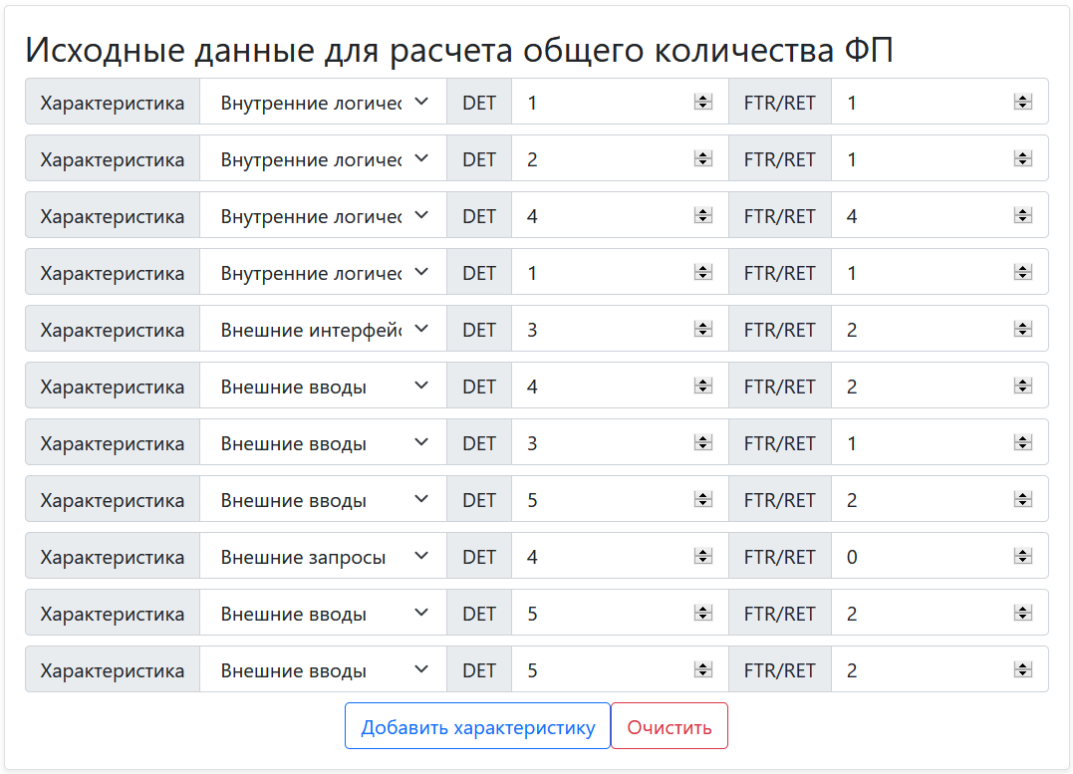
Внешний ввод добавления отслеживаемой акции $FTR = 1$ — ILF отслеживаемых акций, $DET = 3$ — текстовое поле, кнопки «ОК» и «Отменить».

Внешний ввод создания заявки $FTR = 2$ — ILF заявок и EIF с данными о бирже, $DET = 5$ — бумага, цена, количество, тип и кнопка «Оформить».

Внешний запрос на открытие окна добавления $FTR = 0$, $DET = 4$ — кнопки «Добавить», «ОК», «Отменить», поле ввода бумаги.

Внешний ввод удаления $FTR = 2$ — ILF заявок и EIF с данными о бирже, $DET = 5$ — тип, имя, цена, количество, кнопка «Удалить».

Внешний ввод изменения $FTR = 2$ — ILF заявок и EIF с данными о бирже, $DET = 5$ — тип, имя, цена, количество, кнопка «Изменить».



Модель ранней разработки архитектуры

Количество функциональных точек	55.62
Размер кода, KLOC	2.89
Показатель степени	1.22
Трудозатраты, чел.-мес.	18.71
Время, мес.	8.94
Количество работников	2
Бюджет	2145749.62

Выводы

Методика COCOMO II позволяет дать более детальную характеристику на основе большего количества показателей проекта, чем COCOMO.