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”

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

Студенты

Громова В.П. (№ в списке 9)

Лучина Е.Д (№ в списке 13)

Группа ИУ7-81Б

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.

Требования к отчету по лабораторной работе № 7

- Отчет должен содержать:
- Титульный лист (тема, группа, вариант, исполнитель);
- Краткое описание метода функциональных точек и методики COCOMO II и их применения для своего конкретного варианта;
- Детальные расчеты (включая необходимые таблицы, диаграммы);
- Собственную экспертную оценку полезности данной оценки для процесса технико-экономического обоснования программного проекта (в виде выводов с учетом п. 5 задания).

Объем отчета не более 8-12 стр. Отчет предоставляется в электронном виде.

Задание

1. На основе своего варианта задания рассчитать количество функциональных точек для разрабатываемого программного приложения. С этой целью разработать программный инструмент.
2. Произвести оценку трудозатрат и длительности разработки по методике COCOMO II с использованием моделей композиции приложения и ранней разработки архитектуры.
3. Определить среднюю численность команды разработчиков.
4. На основе экспертной оценки стоимости человеко-месяца произвести предварительную оценку бюджета проекта.
5. Дать заключение о применимости метода функциональных точек и модели COCOMO II, а также их сравнение с базовой моделью COCOMO для решения поставленной задачи с учетом своего варианта.

Вариант для выполнения лабораторной работы №7

Примечание: Первый вариант выполняют студенты, имеющие нечетный номер по списку в журнале, второй вариант – студенты с четными номерами.

Постановка задачи

Компания получила заказ на разработку автоматизированной информационной системы оплаты штрафов ГИБДД. Оплата штрафов возможна через веб-интерфейс веб-портала и через приложение для мобильного телефона.

В системе предусмотрено два вида пользователей:

- Пользователь (User) - может просматривать и оплачивать штрафы

- Администратор (Administrator) - доступен просмотр всех выплат, просмотр данных пользователей, их редактирование и создание новых пользователей

Все записи типа «Пользователь» в системе имеют следующие поля:

1. id
2. логин
3. пароль
4. тип
5. регистрационный номер водительского удостоверения
6. номер банковской карты.

Информационная система состоит из следующих модулей:

1. Приложение для мобильного телефона
2. Веб-портал
3. Модуль регистрации и авторизации
4. Модуль обмена данными с системой ГИБДД
5. Модуль проведения платежных транзакций

Приложение для мобильного телефона имеет: страницу регистрации, где пользователь заполняет следующие поля: логин, пароль, номер водительского удостоверения, номер банковской карты; страницу просмотра штрафов, на которой отображаются неоплаченные штрафы, и есть кнопка “оплатить” для каждого штрафа. После нажатия кнопки пользователю приходит ответ о положительном или отрицательном результате выполнения операции.

Веб-портал способен обеспечивать те же функциональные возможности для Пользователя и в дополнение к этому он имеет панель Администратора.

Модуль регистрации и авторизации позволяет добавлять пользователей в базу данных.

Модуль обмена данными с системой ГИБДД предназначен для получения списка штрафов, а также оповещения ГИБДД об оплате штрафа. При отправке сообщения с номером водительского удостоверения информационной системе ГИБДД модуль обмена данными получает список штрафов. При этом каждый штраф описывается следующими полями: номер постановления, дата постановления, имя, фамилия, отчество нарушителя, сумма штрафа. На сообщение об оплате система ГИБДД присылает подтверждение об удаления штрафа из списка неоплаченных или отказ.

Примечание: Система не хранит штрафы в своей базе данных, а получает их из системы ГИБДД при каждом запросе.

Модуль проведения платежных транзакций. Модуль по защищенному протоколу отправляет платежной системе запрос с указанием номера карты пользователя, номера счета ГИБДД и суммы на выполнение проведения оплаты. Система отсылает положительный или отрицательный ответ о результате

выполнения. В отличие от штрафов все операции по оплате сохраняются в базу данных.

Характеристики команды, продукта и проекта

Характеристики продукта:

№	параметр	описание	значение
1	Обмен данными	Несколько внешних интерфейсов, несколько типов коммуникационных протоколов	5
2	Распределенная обработка	Динамическое выполнение процессов в любом подходящем компоненте системы	5
3	Производительность	Время реакции или пропускная способность являются критическими в обычное рабочее время. Не требуется никаких специальных решений относительно использования ресурсов процессора. Время обработки ограничено взаимодействующими системами	3
4	Эксплуатационные ограничения по аппаратным ресурсам	Какие-либо явные или неявные ограничения отсутствуют	0
5	Транзакционная нагрузка	Ожидаются ежедневные пиковые периоды	3
6	Интенсивность взаимодействия с пользователем (оперативный ввод данных)	От 8% до 15% транзакций требуют интерактивного ввода данных	2
7	Эргономические характеристики, влияющие на эффективность работы конечных пользователей	Ни одной из функциональных возможностей таких, как меню, онлайновые подсказки и документация, автоматическое перемещение курсора, скроллинг, удаленная печать и тп	0
8	Оперативное обновление	Онлайновое обновление основных внутренних логических файлов, необходимость специальной защиты от потери данных	4
9	Сложность обработки	Наличие функциональных возможностей: повышенная реакция на внешние воздействия и (или) специальная защита от внешних воздействий, обработка большого количества исключительных ситуаций, поддержка разнородных типов входных/выходных данных	4
10	Повторное использование	Более 10% приложений будут использоваться более чем одним пользователем	3
11	Легкость инсталляции	К установке не предъявляется никаких специальных требований	0

12	Легкость эксплуатации/администрирования	Наличие процедур запуска, копирования и восстановления; минимизируется необходимость в монтировании носителей для резервного копирования;	3
13	Портируемость	Приложение рассчитано на много установок для различных платформ, наличие документации и планов поддержки всех установленных копий приложения.	5
14	Гибкость	Отсутствует поддержка запросов любой сложности к внутренним логическим файлам, управляющая информация не хранится в таблицах, поддерживаемых пользователем в интерактивном режиме	0

При разработке ПО

- 30% кода будет написано на SQL (13 LOC на один оператор);
- 10 % - на JavaScript (56 LOC);
- 60% - на Java (54 LOC).

RCPX	надежность и уровень сложности разрабатываемой системы оцениваются как **очень высокие**	1.91
RUSE	проект не предусматривает специальных усилий на повторное использование компонентов (**низкий**)	0.95
PERS	возможности персонала можно охарактеризовать как **очень высокие**	0.63
PREX	опыт членов команды в данной сфере является скорее **низким**	1.22
PDIF	сложность платформы **средняя**	1.0
FCIL	разработка предусматривает интенсивное использование инструментальных средств поддержки (**высокий**)	0.87
SCED	Учитывая новизну проекта для команды и высокие требования по надежности, заказчик не настаивает на жестком графике (**очень низкий**)	1.43

PREC	Отражает предыдущий опыт организации в реализации проектов данного типа.	3.72 (Наличие некоторого количества прецедентов)
FLEX	Отображает возможность изменения процесса разработки ПО.	5.07 (Точный, строгий процесс разработки)
RESL	Отображает степень детализации анализа рисков, основанного на анализе	4.24 (Степень детализации 60%)

	архитектуры системы.	
TEAM	Отображает степень сплоченности команды и их способность работать совместно.	1.1 (Высокая согласованность)
PMAT	Отображает уровень развития процесса создания ПО в организации-разработчике.	4.68 (Уровень 2 CMM)

К разработке проекта планируется привлечь довольно слаженную команду высокопрофессиональных разработчиков, у которых, однако, практически отсутствует опыт в разработке систем подобного типа. При этом заказчик настаивает на довольно строгом процессе с периодической демонстрацией рабочих продуктов, соответствующих этапам жизненного цикла. Учитывая новизну проекта для команды, на этапе его подготовки был осуществлен относительно детальный анализ рисков, свойственных архитектуре разрабатываемой системы. Организация находится чуть выше второго уровня зрелости процессов разработки.

Надежность и уровень сложности (RCPX) разрабатываемой системы оцениваются как очень высокие, проект не предусматривает специальных усилий на повторное использование компонентов (RUSE). Возможности персонала (PERS) можно охарактеризовать как очень высокие, однако опыт членов команды в данной сфере (PREX) является скорее низким. Сложность платформы (PDIF) средняя. Разработка предусматривает интенсивное использование инструментальных средств поддержки (FCIL). Учитывая новизну проекта для команды и высокие требования по надежности, Заказчик не настаивает на жестком графике (SCED).

Описание метода 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.

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.

Алгоритм подсчета общего количества функциональных точек

1. Подсчитываются функции для каждой категории (вводы, выводы, запросы, структуры данных, интерфейсы)
2. Устанавливаются требования для каждой категории
3. Определяется сложность каждой функции (высокая, средняя, низкая)
4. Каждая функция умножается на соответствующий ей параметр, а затем суммируется с целью получения общего количества функциональных точек

Далее корректируется количество функциональных точек по формуле:

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

где $(0,65 + 0,01 * \Sigma Fi)$ - Нормирующий фактор, Fi — 14 коэффициентов регулировки сложности (приведены в таблице выше). Каждый коэффициент может принимать следующие значения:

- 0 — нет влияния
- 1 — случайное влияние
- 2 — небольшое влияние
- 3 — среднее влияние
- 4 — существенное влияние
- 5 — сильное влияние

В заключение, можно оценить количество строк исходного кода, исходя из выбранного ЯП.

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

Описание метода COCOMO II

Модель композиции приложения

Модель композиции приложения – это одна из трех моделей COCOMO II, которая подходит для проектов, созданных с помощью современных инструментальных средств. Единицей измерения служит объектная точка.

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) x [(100 - %RUSE) /100]

ТРУДОЗАТРАТЫ = NOP /PROD [чел.- мес.]

Время = 3 * (Трудозатраты)(0.33 + 0.2 * (p-1.01))

NOP – новые объектные точки

PROD - оценка скорости разработки

P = 1 - показатель степени

Модель ранней разработки архитектуры

Модель ранней разработки архитектуры (одна из трех моделей COCOMO II). 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 SLOC.

$\text{Трудозатраты} = 2,45 * \text{E}Arch * (\text{Размер})^{p}$,

где Трудозатраты (работа) — число человеко-месяцев;

$EArch = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED$

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).

Размер — KSLOC (предпочтительно для подсчета KSLOC предварительно подсчитать количество функциональных точек)

$$\text{Время} = 3,0 * (\text{Трудозатраты})^{(0.33 + 0.2 * (p - 1.01))}$$

P — показатель степени
Значение показателя степени изменяется от 1.1 до 1.24. Оно зависит

- от того, насколько новаторским является данный проект (PREC),
- от гибкости процесса разработки ПО (FLEX),
- от применяемых процессов управления рисками (RESL),
- от сплоченности команды программистов (TEAM)
- и от уровня управления организацией-разработчиком (PMAT).

Значение показателя степени рассчитывается с учетом этих пяти показателей по пятибалльной шкале от низшего (7 баллов) до наивысшего (0 баллов) уровня
Значения всех пяти показателей суммируются, сумма делится на 100, результат прибавляется к числу 1.01.

	Оценка уровня множителя трудоемкости					
	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PERS	1.62	1.26	1.00	0.83	0.63	0.5
RCPX	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	n/a	0.95	1.00	1.07	1.15	1.24
PDIF	n/a	0.87	1.00	1.29	1.81	2.61
PREX	1.33	1.22	1.00	0.87	0.74	0.62
FCIL	1.30	1.10	1.00	0.87	0.73	0.62
SCED	1.43	1.14	1.00	1.00	1.00	n/a

Достоинства модели COCOMO II

- возможен учет достаточно полной номенклатуры факторов, влияющих на экономические характеристики производства сложных программных продуктов
- метод является достаточно универсальным и может поддерживать различные размеры, режимы и уровни качества продуктов
- фактические данные подбираются в соответствии с реальными проектами и факторами корректировки, которые могут соответствовать конкретному проекту и организации
- прогнозы производственных процессов являются варьируемыми и повторяемыми
- метод позволяет добавлять уникальные факторы для корректировки экономических характеристик, связанные со специфическим проектом и организацией
- возможна высокая степень достоверности калибровки с опорой на предыдущий опыт коллектива специалистов

- результаты прогнозирования сопровождаются обязательной документацией
- модель относительно проста в освоении и применении

Недостатки модели COCOMO II

- все результаты зависят от размера программного продукта: точность оценки размера, оказывает определяющее влияние на точность прогноза трудозатрат, длительности разработки и численности специалистов
- игнорируются требования к характеристикам качества программного продукта
- не учитывается зависимость между интегральными затратами и количеством времени, затрачиваемым на каждом этапе проекта
- игнорируется изменяемость требований к программному продукту в процессе производства
- недостаточно учитывается внешняя среда производства и применения программного продукта
- игнорируются многие особенности, связанные с аппаратным обеспечением проекта

Применение для своего варианта задачи

Метод функциональных точек

Data element type; record element type
Расчет количества функциональных точек:

1. Внутренние логические файлы (ILF)
   К внутренним логическим файлам можно отнести файл, хранящий информацию о зарегистрированных пользователях, а также файл, хранящий информацию об операциях по оплате штрафов.
   Таким образом для разрабатываемой системы предусматривается 2 логических файла:
   1) DET: 6 штук (id пользователя, логин, пароль, тип, регистрационный номер водительского удостоверения, номер банковской карты); RET: 1 штука, так как можно использовать один тип записей для всех перечисленных DET.
   2) DET: 3 штуки (номер карты пользователя, номер счета, сумма); RET: 2 штуки, так как номер карты и номер счета - строки, а сумму можно записать в виде числа.
2. Внешние интерфейсные файлы (EIF)
   К внешним интерфейсным файлам можно отнести 3 описанных в задании модуля, к которым обращаются мобильное приложение и веб-портал.

1) Модуль регистрации и авторизации: DET: 6, RET:1 - аналогично первому логическому файлу.
2) Модуль обмена данными с системой ГИБДД: DET: 7 (номер постановления, дата постановления, имя, фамилия, отчество нарушителя, сумма штрафа, а также ответное сообщение), RET: 2 (сумма штрафа может записывать числом, остальное - строками).
3) Модуль проведения платежных транзакций: DET: 4 (аналогично второму логическому файлу, а также ответное сообщение), RET: 2.

3. Внешние вводы (EI)

   К внешним вводам можно отнести страницу регистрации, кнопку оплаты штрафа, которые нужны как в мобильном приложении, так и на веб-портале, а также необходима панель редактирования и добавления информации для администратора, которая есть только на веб-портале. Тогда:

   1) Регистрация x2: DET: 5 (логин, пароль, номер водительского удостоверения, номер банковской карты, а также кнопка регистрации), FTR: 1 - кнопка ссылается на первый логический файл.
   2) Оплата штрафа x2: DET: 8 (6 полей пользователя - номер постановления, дата постановления, фамилия, имя, отчество, сумма штрафа, 1 кнопка «оплатить» и уведомление об удалении штрафа), FTR: 2 - кнопка ссылается на оба логических файлов.
   3) Панель администратора x1: 8 (6 полей пользователя - аналогично первому логическому файлу, 2 кнопки: «добавить», «редактировать»), FTR: 1 - ссылка на первый логический файл.

4. Внешние выводы (EO)

   К внешним выводам можно отнести успешную или неуспешную транзакцию: DET: 4 (3 поля - номер карты пользователя, номер счета ГИБДД, сумма; и уведомление), FTR: 1 - ссылка на модуль проведения платежных транзакций.

5. Внешние запросы (EQ)

   1) Вывод штрафов: DET: 6 (6 полей пользователя - номер постановления, дата постановления, фамилия, имя, отчество, сумма штрафа), FTR: 1 - ссылается на модуль обмена данными системы ГИБДД.
   2) Вывод данных для администратора: DET: 6 (6 полей пользователя - id, логин, пароль, тип, права, банковская карта), FTR: 1 - ссылка на первый логический файл.

Таблица. Ненормированное количество функциональных точек.

тип элементарных процессов	кол-во	ранг	ненорм. FP

Внешние вводы (EI)	5	6	30
Внешние выводы (EO)	2	4	8
Внешние запросы (EQ)	3	4	12
Внутренние логические файлы (ILF)	2	7	14
Внешние интерфейсные файлы (EIF)	3	5	15
Итого			79

Нормированное количество функциональных точек:

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

ΣFi =37

Нормирующий фактор: 0.65 + 0.1 * 37 = 1.02

$FP = 79$ *1.02 = 80.58

Количество строк исходного кода, исходя из того, что программу необходимо разработать с использованием языка программирования Java (0.6), SQL (0.3), JS(0.1):

(0.6 * 53 + 0.3 * 13 + 0.1 * 56) * 80.58 = (31.8 + 3.9 + 5.6) * 80.58 = 3 328 LOC.

Модель COCOMO II - композия приложения

Подсчет 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 отчетов. Для простых отчетов назначаются 2 объектные точки, умеренно сложным отчетам назначаются 5 точек. Написание сложных отчетов оценивается в 8 точек.
- количество модулей, которые написаны на языках третьего поколения и разработаны в дополнение к коду, написанному на языке программирования четвертого поколения. Каждый модуль на языке третьего поколения считается за 10 объектных точек.

Результаты расчета:

- регистрация: 2 точки (среднее) x2
- вывод штрафов: 3 точки (сложное) x2
- уведомление: 1 точка (простое) x2
- вывод данных пользователей: 3 точки (сложное) x2

- форма администратора 3 точки (сложное)

Получаем 9 * 2 + 3= 18 + 3 = 21 объектных точек.

За модуль на языке третьего поколения (Java) + 10 точек.

Всего: 21 + 10 = 31 объектных точек.

RUSE - Низкий (примем за 10 %),

NOP = 31 x [(100 - 10) /100] = 31* 0.9 = 27.9.

Возможности персонала можно охарактеризовать как очень высокие, а опыт членов команды в данной сфере является скорее низким, поэтому оценка скорости разработки будем считать номинальной . PROD = 13.

Тогда:

ТРУДОЗАТРАТЫ = NOP /PROD [чел.- мес.] = 27.9 / 13 = 2.15 ч-м

А Время = 3 * (2.15)^(0.33 + 0.2 * (1-1.01)) = 3.856 месяцев

(NOP – новые объектные точки, PROD - оценка скорости разработки)

Модель сосото II - ранней разработки архитектуры

Степень

$$p = (PREC + FLEX + RESL + TEAM + PMAT) / 100 + 1.01 =$$

$$= (3.72 + 5.07 + 4.24 + 1.1 + 4.68) / 100 + 1.01 = 18.81/100 + 1.01 =$$

Размер возьмем из метода функциональных точек:

Размер = 3 328 LOC = 3.328 KLOC.

$$EArch = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED =$$

$$= 1.91 * 0.95 * 0.63 * 1.22 * 1 * 0.87 * 1.43 = 1.735.$$

$$\text{Трудозатраты} = 2.45 * 1.735 * (3.328)^{1.1981} = 17.95.$$

$$\text{Время} = 3.0 * (17.59)^{(0.33 + 0.2 * (1.1981 - 1.01))} = 8.6.$$

Для выполнения проекта потребуется 2 разработчика (17.95 / 8.6).

Предварительная оценка бюджета: 150 000 * 2 * 8.6 = 2 580 000 рублей.

Программный расчет

Модель COCOMO II

Параметры для метода функциональных точек

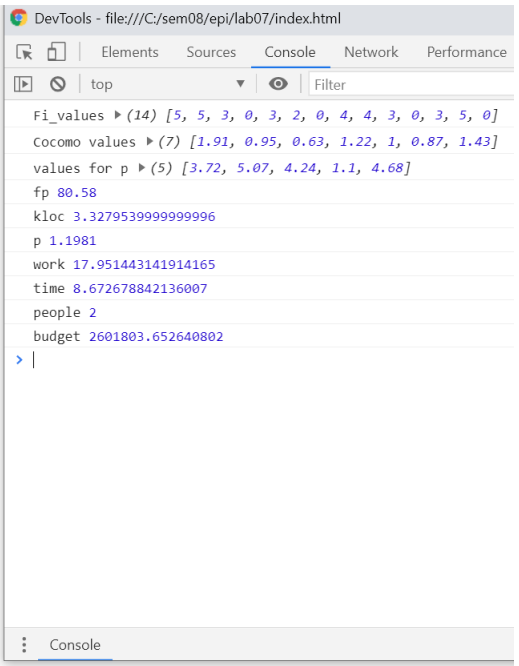
Обмен данными: сильное влияние
Распределенная обработка: сильное влияние
Производительность: среднее влияние
Эксплуатационные ограничения по аппаратным ресурсам: нет влияния
Транзакционная нагрузка: среднее влияние
Интенсивность взаимодействия с пользователем (оперативный ввод данных): небольшое влияние
Эргономические характеристики, влияющие на эффективность работы конечных пользователей: нет влияния
Оперативное обновление: существенное влияние
Сложность обработки: существенное влияние
Повторное использование: среднее влияние
Легкость инсталляции: нет влияния
Легкость эксплуатации/администрирования: среднее влияние
Портируемость: сильное влияние
Гибкость: нет влияния

Параментры COCOMO

Надежность и уровень сложности разрабатываемой системы (RCPX): Очень высокий
Повторное использование компонентов (RUSE): Низкий
Возможности персонала (PERS): Очень высокий
Опыт персонала (PREX): Низкий
Сложность платформы разработки (PDIF): Номинальный
Средства поддержки (FCIL): Высокий
График работ (SCED): Очень низкий

Параметры для расчета показателя COCOMO 2

Новизна проекта (PREC): Наличие некоторого количества прецедентов
Гибкость процесса разработки (FLEX): Точный, строгий процесс разработки
Разрешение рисков в архитектуре системы (RESL): Частое (60 %)
Сплоченность команды (TEAM): Высокая согласованность
Уровень зрелости процесса разработки (PMAT): Уровень 2 CMM
calculate

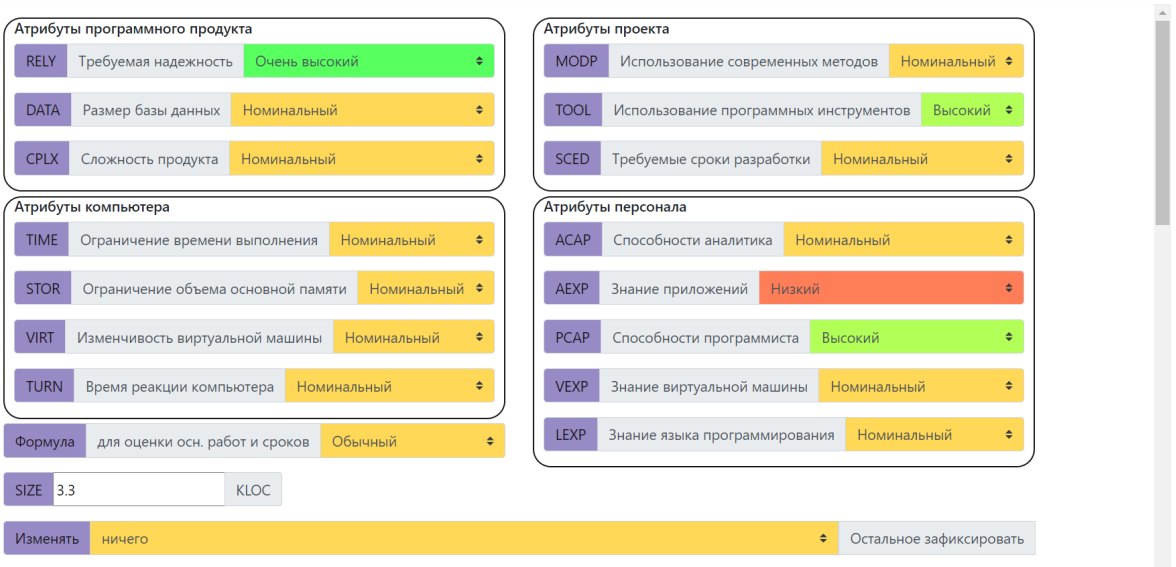


Трудозатраты: 17.95 ч-м.

Время: 8.67 месяцев.

Бюджет: 2 601 803 рублей.

Модель COCOMO



PM: 14.12392279032189; TM: 6.837904576540705, RELY = 1.4, TOOL = 0.91, AEXP = 1.15, PCAP = 0.86, 3.3 KLOC, Обычный вариант

Распределение работ и времени по стадиям жизненного цикла

Вид деятельности	Трудозатраты (чм)	Время (м)	Кол-во сотрудников (Work/Time)
Планирование и определение требований	1.1299138232257513	2.4616456475546538	0
Проектирование продукта	2.5423061022579403	2.4616456475546538	1
Детальное проектирование	3.5309806975804725	1.2308228237773269	3
Кодирование и тестирование отдельных модулей	3.6722199254836916	1.2308228237773269	3
Интеграция и тестирование	4.378416064999786	1.9146132814313976	2
Итого	15.253836613547643	9.299550224095357	

Предположительный бюджет

Анализ требований (4%)	79947
Проектирование продукта (12%)	239841
Программирование (44%)	879418
Тестирование (6%)	119921
Верификация и аттестация (14%)	279815
Канцелярия проекта (7%)	1399073
Управление конфигурацией и обеспечение качества (7%)	139907
Создание руководств (6%)	119921
Непредвиденные риски(+20%)	399735
Итого	2398412

Трудозатраты: 15.25 ч-м.

Время: 9.3 месяцев.

Бюджет: 2 398 412 рублей.

Вывод

Модель COCOMO II позволяет учитывать переиспользование продукта, а также новизну проекта, возможные риски, гибкость проекта. Все это возможно благодаря учету значений новых драйверов при расчете времени и трудозатрат проекта. В сравнении с результатами расчетов по модели COCOMO, можно наблюдать меньшее время выполнение проекта при больших трудозатратах, что возможно при слаженной команде и хорошем планировании. Такие показатели не учитывались в первом варианте модели.

Исходный код разработанного инструмента для расчета:
https://github.com/IamLena/epi/tree/main/lab07.
Скачать файлы index.html и main.js в одну папку, открыть в браузере index.html
Выбрать параметры проекта и нажать на кнопку calculate