Постановка задачи

Выполнить разработку СОИУ в соответствии с описанием ее функциональности (АИС Портал тестирования) на основе моделей унифицированного процесса (RUP). Написать программу, реализующую фрагмент СОИУ, и реализовать в ней паттерны бизнес-логики (service layer), работы с БД (table data gateway) и gof (заместитель). Для построения диаграмм использовать среду StarUML.

Спецификация основных требований

Перечень требований кандидатов:

Функциональные требования:

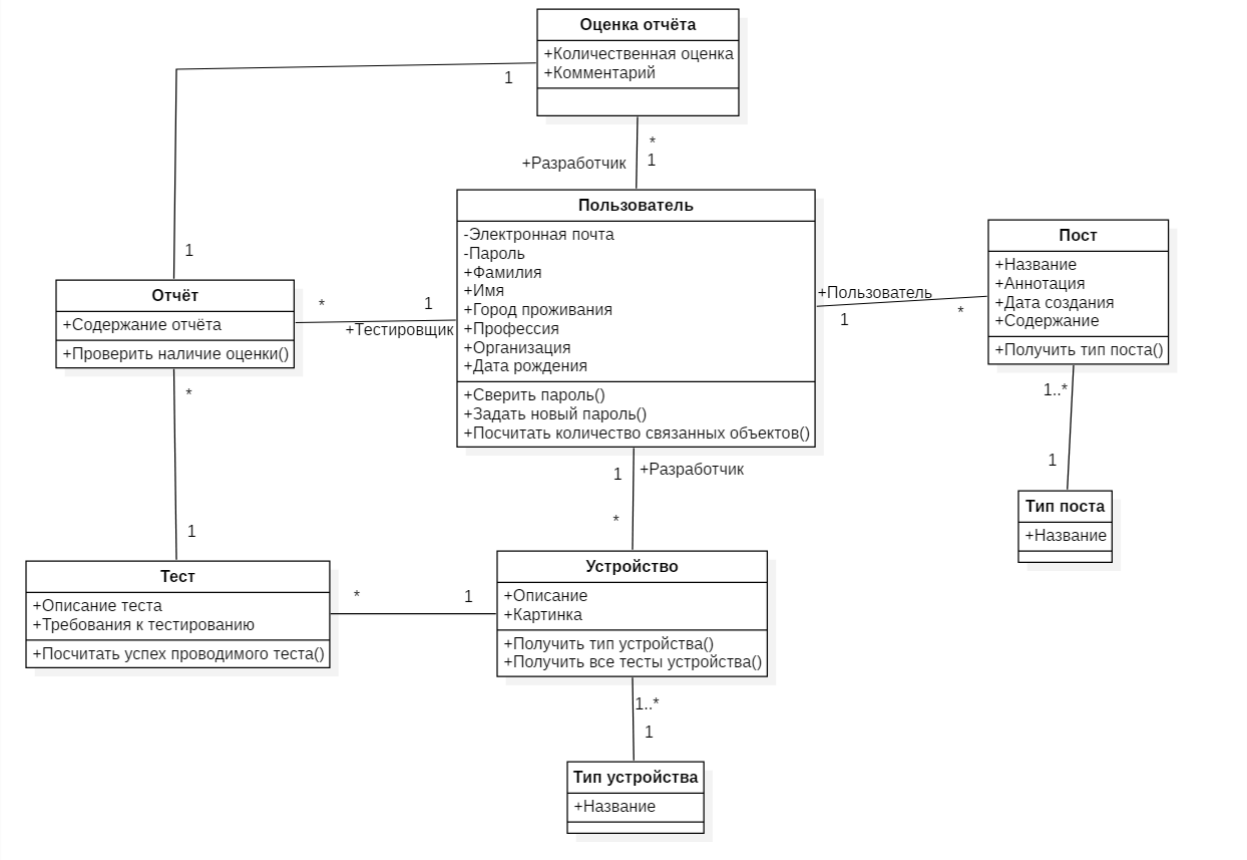
- Разработчики смогут выкладывать свои устройства;
- Разработчики смогут проводить тестирования по своим устройствам;
- Тестировщики смогут принимать участие в тестировании, а разработчики – оценивать их проделанную работу;
- Все пользователи могут писать посты.

Нефункциональные требования:

- Необходимо использовать паттерн бизнес-логики service layer;
- Необходимо использовать паттерн работы с БД table data gateway;
- Необходимо использовать паттерн gof заместитель;
- Система будет доступна через интернет.

Модель предметной области

Диаграмма классов предметной области:



Глоссарий понятий:

- Тестировщик – пользователь системы, который имеет возможности проверять устройства для разработчиков;
- Разработчик – пользователь системы, который имеет некоторые устройства для тестирования;
- Пост (Post) – отдельно взятое сообщение в форуме, социальных сетях или блоге;
- Фреймворк – программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта.

Используемая бизнес-модель – краудсорсинг (crowdsourcing). Она подразумевает использование системы многими пользователями, которые

будут сами наполнять её и работать с нею (примером таких систем можно назвать Википедию).

Выявленные актёры

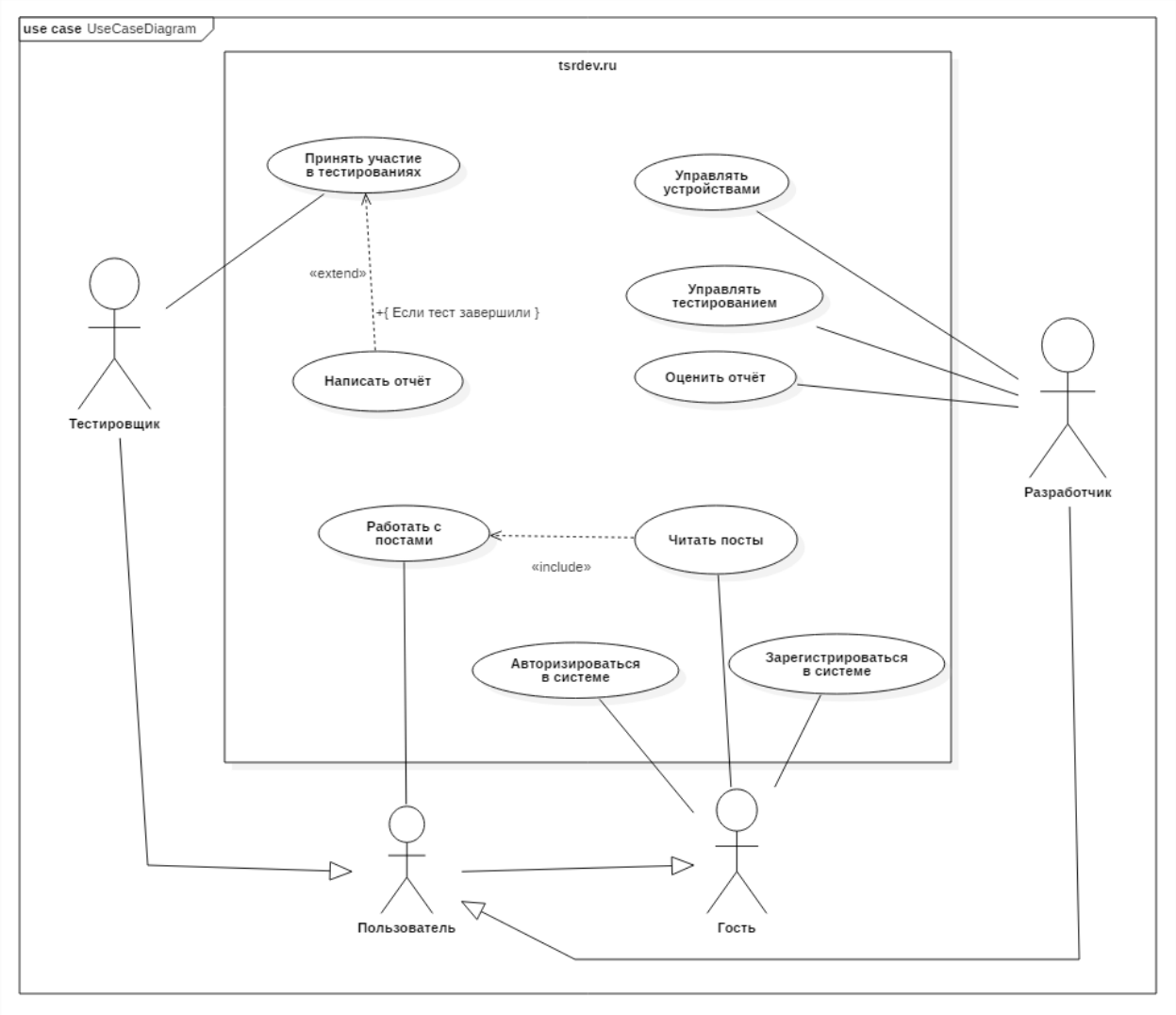
- Гость – единственное не зарегистрированное лицо;
- Пользователь – является стандартным зарегистрированным лицом в системе. Он может писать посты, но не имеет прав в непосредственном модуле тестирования;
- Тестировщик – в системе имеет все права пользователя. Также он участвует в тестированиях и пишет по ним отчёты;
- Разработчик – имеет все права пользователя. Также имеет возможность управлять своими устройствами и тестированиями в системе и оценивать отчёты по тестированиям его устройств.

Выявленные прецеденты, их приоритеты и описание

- Регистрироваться/авторизовываться в системе. Ключевой момент идентификации пользователей в системе для дальнейшей работы актёров в системе;
- Управлять устройствами – добавлять, изменять и удалять объекты типа «Устройство»;
- Управлять тестированиями – добавлять, изменять и удалять объекты типа «Тест»;
- Принимать участие в тестированиях – регистрация в тестировании, которая связывает тестировщика с тестом;
- Написать отчёт – финальный этап тестирования, который требует от тестировщика написать отчёт по проделанной работе;
- Оценить отчёт – разработчик проверяет отчёты тестировщиков и оценивает степень соответствия заданным требованиям;

– Работать с постами – пользователь может добавлять, изменять и удалять свои посты, а также читать чужие посты. Посты будут отсортированы по категориям и дате создания для более удобного поиска;

Диаграмма основных прецедентов



Описание возможной архитектуры

Перечень архитектурно-значимых прецедентов:

Обобщенные механизмы проектирования:

- Механизм хранения – хранение в БД;
- Механизм поддержки системы в режиме онлайн (mod_wsgi);

Системное ПО:

Операционной системой для сервера будет являться Linux Ubuntu 14.04, при разработке будет использоваться ОС Windows 10 с операционной оболочкой bash shell, которое ставится отдельно. Первая ОС удобна для развёртывания системы на веб-сервере, в то время как вторая система позволяет использовать большую часть программных продуктов для разработки.

Варианты используемых компонентов или каркасов:

1. Фреймворк Django – свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

Достоинства:

- Быстрота разработки приложений достаточно высокая;
- Django имеет полную комплектацию для работы с сайтами;
- Большая масштабируемость приложения.

Недостатки:

- Использование шаблона маршрутизации с указанием URL;
- Django слишком монолитный;
- Все базируется на ORM Django.

Последнее означает отсутствие возможности внедрения определенного шаблона проектирования для работы с базами данных, из-за чего данный вариант подходит мало.

2. Фреймворк Flask - фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков – минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

   Достоинства:

   - Возможность использования базовых возможностей для построения веб-приложения, из-за чего она предоставляет возможности для введения определенных паттернов проектирования;
   - Быстрота разработки приложений иногда выше разработки с Django;

   Недостатки:

   - Не подходит для больших проектов, так как усложняются работы с имеющимися инструментами разработки;
3. Фреймворк RubyOnRails – фреймворк, написанный на языке программирования Ruby, реализует архитектурный шаблон Model-View-Controller для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером баз данных.

   Достоинства:

   - Гибкость решений, проектируемых на RoR;
   - Быстрое время разработки.

   Недостатки:

   - Сложность разворачивания проекта на сервере;

- Необходимость изучать новый язык Ruby.

Фреймворк Flask предоставляет не только возможности для разработки веб-приложения, но и имеет особенность подключать только необходимые части. Это позволит проектировать все необходимые условия по курсовой работе. Также он базируется на языке Python, который не нужно изучать с нуля.

Варианты используемых серверов:

1. Apache – Web-сервер с открытым исходным кодом;

   Достоинства:

   - Надежность;
   - Безопасность;
   - Гибкость настройки.

   Недостатки:

   - Отсутствие удобного графического интерфейса администратора;
   - Настройка Apache осуществляется путем редактирования его конфигурационного файла;
   - Отсутствие асинхронности приводит к большим затратам ресурсов системы.
2. Nginx – свободный веб-сервер. В своё время был написан отечественным программистом Игорем Сысоевым для компании Rambler, но поддерживается и развивается до сих пор.

   Достоинства:

- шифрование, сжатие, поддержка многих сайтов на одном IP-адресе и прочие возможности, доступные в большинстве веб-серверов;
- межсистемность, малый размер, простота конфигурации, масштабируемость;
- использование преимущества от асинхронной системы ввода-вывода, что на практике означает экономию ресурсов системы и выгодно сказывается в случае больших нагрузок;

Недостатки:

- Нет встроенной поддержки технологии Web-сокет, хотя при желании Web-сокет можно использовать;
- Появился позднее Apache.

Веб-сервер nginx получает всё большую популярность. Также особенность асинхронной работы с потоками позволяет балансировать нагрузку, и вся СОИУ будет работать значительно стабильнее, так что выбираем его.

Для взаимодействия между сервером и программой воспользуемся стандартом WSGI. WSGI предоставляет простой и универсальный интерфейс между большинством веб-серверов и веб-приложениями или фреймворками. Альтернатив при использовании микрофреймворка Flask и HTTP-сервера nginx нет.

В качестве СУБД возможно использовать PostgreSQL, MySQL и SQLite. Так как система рассчитана на краудсорсинг, то и СУБД должна работать с большим числом данных, из-за чего SQLite не подходит (он предназначен для легковесных баз данных с малым потоком запросов). Из

оставшихся двух PostgreSQL является более современной СУБД с поддержкой разных типов данных, так что в качестве СУБД выбираем её.

Диаграмма подсистем:

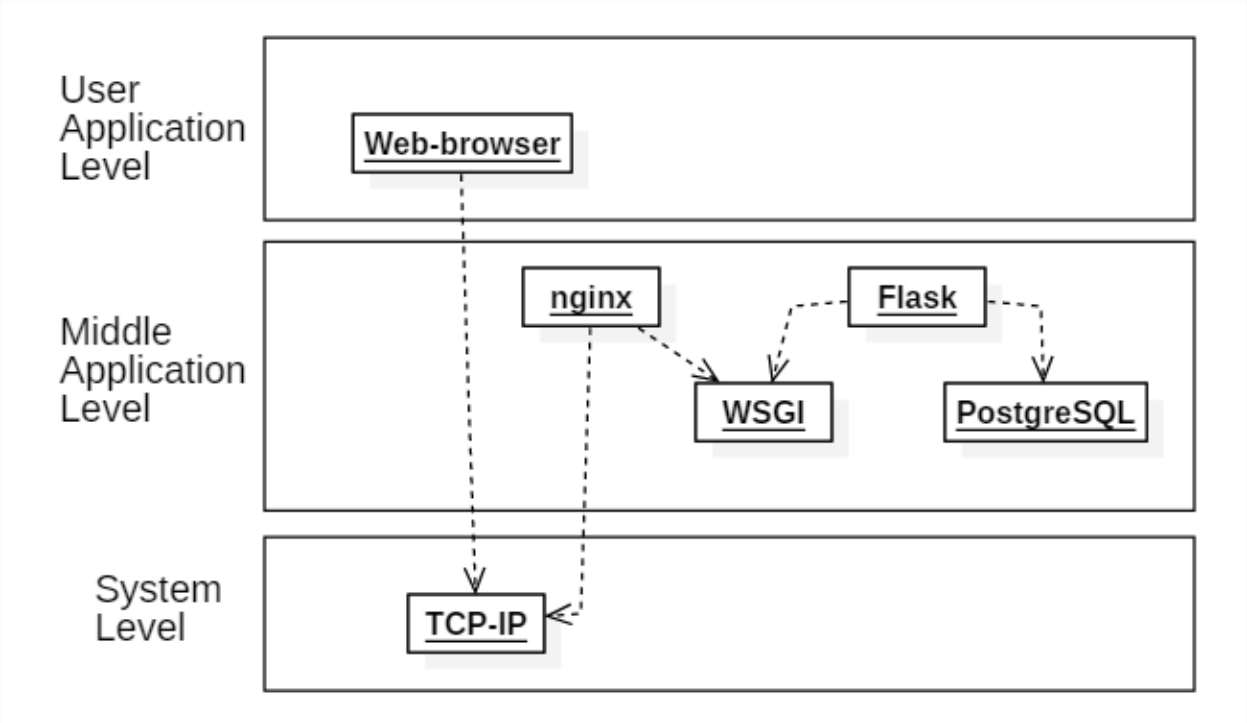
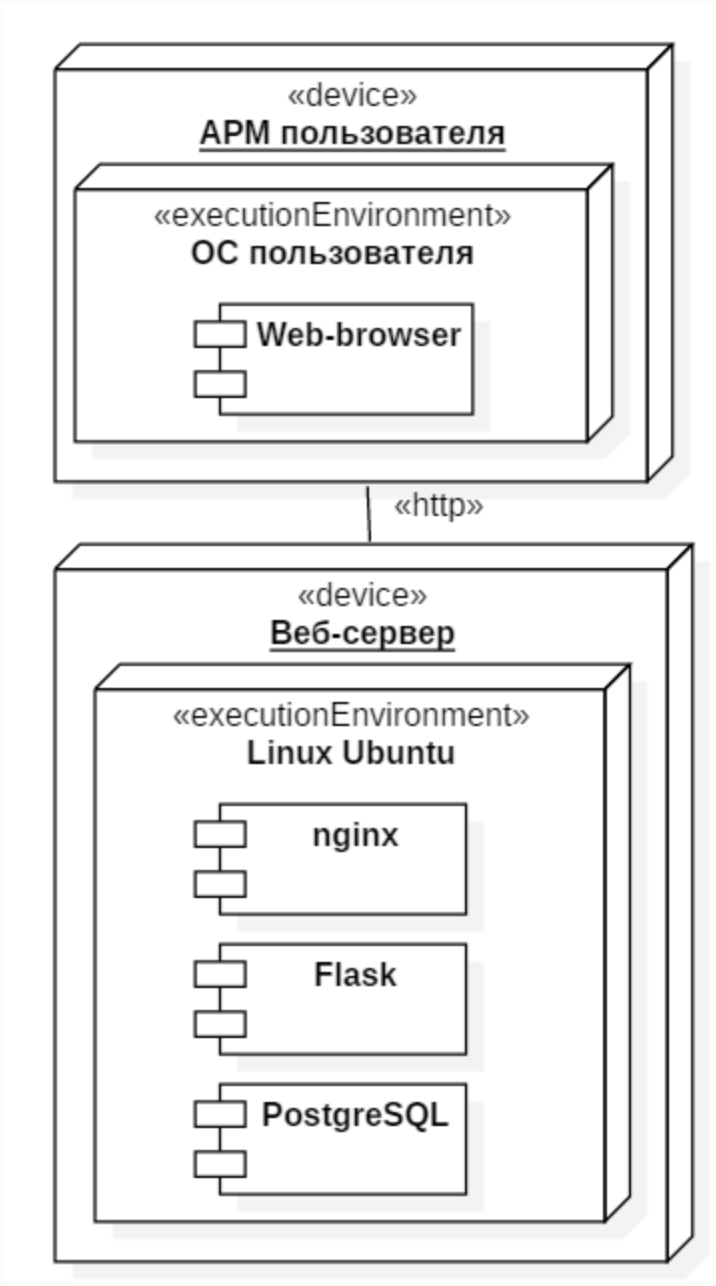


Диаграмма развёртывания:



Перечень критических рисков

Проектные риски:

– Слишком сложный проект, который не будет укладываться во временные рамки. Возможные пути исправления – упрощение спецификации;

Технические риски:

- Микрофреймворк Flask будет плохо работать с PostgreSQL. Для разрешения данного риска достаточно иметь возможность подключить другую СУБД, альтернативой будет MySQL;
- Нарушение логической структуры предметной области (к примеру, разработчик и тестировщик будет являться одним и тем же лицом). Это можно обойти, если использовать чёткое разделение при регистрации.

Перечень экранных форм и их сложность

Перечень форм, которые предполагаются в системе:

1. Форма перехода в определенные части системы – «Посты», «Устройства и Тесты», «Профиль пользователя».
2. Форма регистрации;
3. Форма авторизации;
4. Форма редактирования профиля;
5. Форма выбора поста;
6. Форма просмотра поста;
7. Форма создания поста;
8. Форма редактирования поста;
9. Форма создания или изменения устройства и тестов;
10. Форма выбора или удаления устройств и тестов пользователя;
11. Форма написания отчёта;
12. Форма выбора отчётов по тестированию;
13. Форма оценки отчётов.

Перечень отчётов, которые предполагаются в системе:

1. Отчёт о разработчиках и их устройствах;
2. Отчёт о достижениях тестировщика;
3. Отчёт о достижениях разработчика;
4. Отчёт об отчётах по тестам;
5. Отчёт о тестах устройств и их успеваемости.
6. Списки тестов;
7. Списки отчётов по тестам.

Иным компонентом будет являться только файл настройки сервера для веб-сайта.

Расчёт объектного указателя

Оценка сложности форм представлены в таблице 1.

Таблица 1. Оценка сложности форм.

Обозначение формы	**Количество представлений**	**Количество таблиц данных**	**Оценка сложности**
Ф1	1	0	Простой
Ф2	2	1	Простой
Ф3	1	1	Простой
Ф4	2	1	Простой
Ф5	1	2	Простой
Ф6	0	2	Простой
Ф7	2	2	Простой
Ф8	2	2	Простой
Ф9	4	3	Простой
Ф10	4	3	Простой
Ф11	1	3	Простой
Ф12	2	3	Простой
Ф13	2	4	Простой

Оценка сложности отчётов представлены в таблице 2.

Таблица 2. Оценка сложности отчётов.

Обозначение отчёта	**Количество представлений**	**Количество таблиц данных**	**Оценка сложности**
О1	2	3	Простой
О2	5	8	Сложный
О3	5	8	Сложный
О4	1	4	Простой
О5	2	3	Простой
О6	1	3	Простой
О7	1	4	Простой

Оценка размера проекта в ОР

Оценка размера представлена в таблице 3. Сначала ведётся итоговая оценка размера проекта по типам объекта, используя сумму объектов, помноженную на их весовые коэффициенты. Затем суммируются данные итоговые оценки для оценки размера проекта в ОР.

Таблица 3. Оценка размера ОР.

Тип объекта	**Количество**	**Вес**						**Итого**
		Простой		**Средний**		**Сложный**		
Форма	13	x1	13	x2	0	x3	0	13
Отчёт	7	x2	5	x5	0	x8	2	26
Иной компонент	1					x10	1	10
Объектные указатели (ОР)								49

Расчёт новых объектных указателей NOP производится по следующей формуле:

$$NOP = OP * \frac{100 - \%REUSE}{100}$$, где

%REUSE – процент повторного использования программных продуктов.

Некоторые пары форм (такие как Ф7 и Ф8) имеют однотипный функционал, а также некоторые отчёты (О2 и О3) содержат одинаковые представления (Статистика по постам; тесты, в которых принимали участия итд). Подсчитаем %REUSE:

$\%REUSE = \frac{N_{\text{одинаковыхпар}}}{N_{\text{всехпар}}} * 100$,, где

$N_{\text{одинаковыхпар}}$ – количество пар одинаковых объектов;

$N_{\text{всехпар}}$ – количество возможных пар объектов в целом.

$$N_{\text{всехпар}} = C_{13}^2(\text{пары форм}) + C_7^2(\text{пары отчётов}) = 13 * 6 + 7 * 3 = 99$$

$$\%REUSE = \frac{2}{99} * 100 \approx 2$$

$$NOP = 49 * \frac{100 - 2}{100} = 48{,}02$$

Расчёт затрат

Затраты по модели композиции приложения вычисляются по следующей формуле:

$\text{Затраты} = \frac{NOP}{PROD}[\text{чел} * \text{мес}]$, где

NOP – новые объектные указатели;

PROD – производительность разработки, выраженная в терминах объектных указателей.

PROD подбирается по значениям, представленными в таблице 4.

Таблица 4. Оценка скорости разработки.

Опытность / возможности разработчика	**Зрелость / возможности среды разработки**	**PROD**
Очень низкая	Очень низкая	4
Низкая	Низкая	7
Номинальная	Номинальная	13
Высокая	Высокая	25
Очень высокая	Очень высокая	50

Посчитаем, что опытность и зрелость у нас значения «Номинальная». Тогда:

$$\text{Затраты} = \frac{48{,}02}{13} \approx 3{,}69 \text{ чел} * \text{мес.}$$

Расчёт длительности и стоимости разработки

Стоимость проекта вычисляется по следующей формуле:

$\text{СТОИМОСТЬ} = \text{ЗАТРАТЫ} * \text{РАБКОЭФ}$ где

РАБКОЭФ – среднее значение рабочего коэффициента и составляет 50000 руб/(чел*мес).

$$\text{СТОИМОСТЬ} = 3{,}69 * 50000 = 184\,500 \text{ руб.}$$

Оценка календарного времени (TDEV) вычисляется по следующей формуле:

$TDEV = \left[3{,}0 * (\text{ЗАТРАТЫ})^{0{,}33+0{,}2(B-1{,}01)}\right] * \frac{SCEDPercentage}{100}$ [мес], где

B – показатель степени масштабных факторов;

SCEDPercentage – процент увеличения (уменьшения) номинального графика.

Масштабные факторы описаны в таблице 5.

Таблица 5. Значения масштабных факторов.

Сокращение	**Обозначение**	**Значение фактора**	**Оценка значения фактора**	**Числовое значение**
W1	PREC	Большей частью предсазуемый	Высокий	2
W2	FLEX	Некоторое расслабление в работе	Номинальный	3
W3	RESL	Некоторый	Низкий	5
W4	TEAM	Высокая кооперативность	Очень высокий	1
W5	PMAT	Повторяемый	Номинальный	3

Посчитаем показатель B:

$$B=1{,}01+0{,}01\sum_{i=1}^{5} W_i = 1{,}01+0{,}01*14=1{,}15$$

Положим SCEDPercentage равный 100. Тогда:

$$TDEV = \left[3{,}0 * (3{,}69)^{0{,}33+0{,}2(1{,}15-1{,}01)}\right] * 1 \approx 4{,}8 \text{ месяцев}$$