МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___» _____2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Онлайнова система управління розробкою програмних проектів

Виконавець: Барановський А.М,

Керівник: Ткаченко В.Г.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 "Комп'ютерна інженерія"
(шифр, найменування)

Освітньо професійна програма «Системне програмування»
Форма навчання заочна

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Литвиненко О. Є.

«_____»_____2020 р.

ЗАВДАННЯ
на виконання дипломного проєкту
Барановського Артема Миколайовича
(прізвище, ім'я, по батькові)

1.Тема роботи: "Онлайнова система управління розробкою програмних проектів"
затверджена наказом ректора від "21" грудня 2020 року № 2523 /ст.

2. Термін виконання роботи: з 11.01.2021 до 28.02.2021
3.Вихідні дані до проєкту (роботи): системна програмна документація, документація до програмного забезпечення управління проєктами

4.Зміст пояснювальної записки (перелік питань, що підлягають розробці):

5.Перелік обов'язкового графічного матеріалу:

6. Календарний план

№ п/п	Етапи виконання дипломного проєкту	Термін виконання етапів	Примітка
1	Провести аналіз літератури за темою дипломного проєкту та аналіз існуючих систем.	11.01.21 12.01.21	
2	Оформлення розділу 1	13.01.21- 14.01.21	
3	Оформлення розділу 2	15.01.21- 16.01.21	
4	Оформлення розділу 3	17.01.21- 28.01.21	
5	Редагування та друк пояснювальної записки, графічного матеріалу	29.01.21- 01.02.21	
6	Проходження нормо-контролю, перепліт пояснювальної записки.	02.02.21- 14.02.21	
7	Розробка тексту доповіді, підготовка презентації	15.02.21- 17.02.21	
8	Отримання відгуку керівника, рецензії.	18.02.21- 19.02.21	
9	Захист дипломного проєкту на кафедру	21.02.21- 23.02.21	

7. Дата видачі завдання «_11_» січня 2021 р.

Керівник дипломного проєкту_____Ткаченко В.Г.
(підпис)

Завдання прийняв до виконання_____Барановський А.М.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту “Онлайнова система управління розробкою програмних проектів”: 70 с., 22 рис., 26 літературних джерел, 1 додаток.

РОЗРОБКА ПЗ, ПРОЄКТ ПЗ, КОНТЕНТ ПЗ, АНАЛІЗ ПРОЄКТУ, УПРАВЛІННЯ ПРОЄКТУВАННЯМ, ТЕСТУВАННЯ

Мета дослідження – вивчення методологічної різноманітності групової динаміки в умовах розробки *web*-додатків.

Об'єкт дослідження – методи групової розробки *web*-додатків.

Предмет дослідження – програмний система управління *web*-проектами.

Прогнози припущення щодо розвитку об’єкта дослідження – створення робочого зразка програми та використання його при розробці програмних проєктів.

Результати дипломного проєктування рекомендується використовувати при розробці нових програмних засобів управління проєктами.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ВСТУП

На сьогодні існує велика кількість різноманітних проектів, а також компаній, які розглядають можливість їх завершення. Для кожної компанії дуже важливо довести ці проекти до кінця з мінімальними затримками. Для цього потрібно ретельно підходити до планування використання робітників та інших ресурсів і своєчасно реагувати на зміни в процесі.

На сьогоднішній день необхідність моніторингу та оцінки проекту стикається з кожною компанією, що розробляє програмне забезпечення, незалежно від розміру та зрілості. Однак ця проблема особливо гостра для невеликих команд, стартапів та компаній з молодими менеджерами проектів. Через відсутність великого накопиченого досвіду в управлінні командою дуже часто виникають затримки у впровадженні. А це, в свою чергу, призводить до затримки проекту в цілому і як наслідок невдоволення замовників. Це негативно відображає репутацію компанії та ставлення до неї її потенційних споживачів.

Моніторинг процесів і завдань, що здійснюються під час реалізації проекту, а також їх оцінка дозволяє своєчасно реагувати на негативні зміни під час їх реалізації. Коли керівники проектів приймають важливі рішення без перевірених даних, це немовби вони роблять удар у темряві. Їх рішення ґрунтуються на дуже незначних доказах або взагалі відсутніх, тому міра може бути не дуже ефективною і може бути лише марною тратою ресурсів та часу. Основними цілями проекту є забезпечення досягнення цілей проекту. Оскільки цілі проекту полягають у зусиллях, графіку та якості, фокус моніторингу зосереджений саме на цих аспектах. Для проекту можуть проводитися різні рівні моніторингу. Три основні рівні моніторингу - це рівень діяльності, звіт про стан та аналіз етапів.

Актуальність даної теми полягає в тому, що створення автоматизованої системи моніторингу та оцінки програмних проектів дозволить командам, що не є економними та мають обмежений бюджет, ефективно планувати використання ресурсів, що впливатиме на задоволеність споживачів.

Запропонована система програм є альтернативою існуючій. Вона сприяє значному збільшенню швидкості прийняття рішень щодо використання ресурсів, відображаючи та підраховуючи основні показники стану проекту.

РОЗДІЛ 1
АНАЛІЗ ПРОЦЕСУ РЕАЛІЗАЦІЇ ПРОЄКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1.Вступ до управління проектами

У своїй сучасній формі управління проектами бере свій початок на початку 1950-х років, хоча його коріння сягають останніх років 19 століття. Оскільки компанії визнали переваги організації проекту та визнали необхідність у спілкуванні та координації між відділами та професіями, розроблений чітко визначений метод управління проектами.

Сьогодні в багатьох організаціях не працюють штатні керівники проектів. Насправді загальноприйнятою практикою є об'єднання команди проектів для задоволення конкретних потреб, що, як правило, передбачає виготовлення кінцевого продукту чи послуги, що приносить користь або змінює організацію. Кінцевий результат може бути матеріальним або нематеріальним.

Щоб досягти успіху в цьому кінцевому результаті - це управління проектами. По суті, управління проектами зосереджується на плануванні та контролі всього, що пов'язано з досягненням кінцевого результату - і це процес, який кожна людина в проектній команді повинна охопити, зрозуміти та виконати, незалежно від рівня досвіду. Приклад діяльності IT-організації, яка займається розробкою програмного забезпечення, можна розділити на такі розділи:

- Розвиток;
- Управління.

Проект - це частина запланованої роботи або діяльності, яка закінчується протягом певного періоду і призначена для досягнення певної мети.

Основними характеристиками проекту є:

- Працюючи для досягнення мети;
- Одинарна визначувана мета, кінцевий пункт або результат;
- Унікальний;
- Тимчасова діяльність;
- Перерізати організаційні лінії;
- Залучати незнайомство.

Повна процедура розробки програмного забезпечення від аналізу до тестування та технічного обслуговування, виконана відповідно до методологій виконання, у заздалегідь визначений діапазон часу для досягнення очікуваного продукту, який називається Проект програмного забезпечення.

На відміну від автомобіля, ви не можете торкатися програмного забезпечення, ні відчувати його, ні бачити. Майже всі програмні продукти ретельно підігнані відповідно до вимог замовника. Дуже важливим є те, що основна технологія змінюється і прогресує так часто і швидко, що досвід одного продукту може не застосовуватися до іншого. Всі ці обмеження спричиняють ризик для прогресу програмування, тому дуже важливо ефективно управляти програмними проектами.

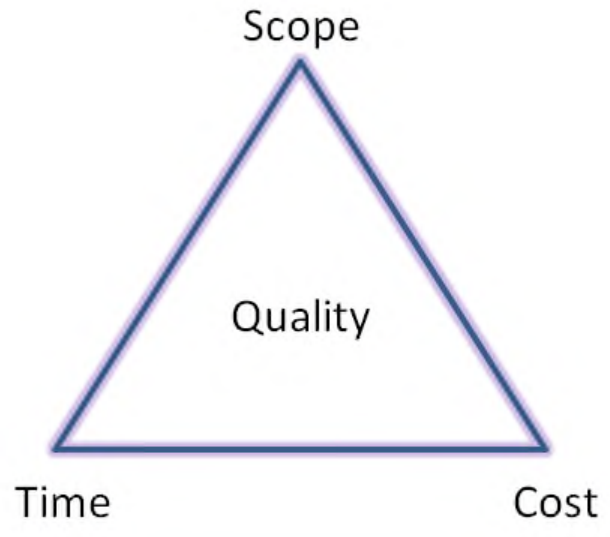


Рис. 1. 1 Трикутник управління проектами

На рисунку 1.1 показано потрійні обмеження для програмних проектів. Його початок був неясним, але він застосовувався з 1950-х років.

Він стверджує, що:

- Зміни в одному обмеженні вимагають змін у інших для компенсації.
- Керівник проекту може вибирати між обмеженнями.
- Якість роботи обмежується фінансовим планом проекту, термінами виконання та особливостями.

Це фундаментальна частина організації програмного забезпечення, щоб забезпечити якісний результат проекту, зберігаючи витрати в межах обмежень витрат замовника та реалізовуючи проект відповідно до графіка.

Є кілька компонентів, як внутрішніх, так і зовнішніх, які можуть вплинути на потрійний трикутник вимог. Будь-який із трьох факторів може серйозно вплинути на інші два. Тому управління програмним проектом є важливим для включення вимог користувача поряд із планом витрат та часом.

Наприклад, проект можна закінчити швидше, збільшивши план витрат або скорочуючи функції. Аналогічним чином, збільшення списку функцій може вимагати еквівалентного збільшення бюджету та розкладу. Скорочення бюджету без коригування розкладу або функцій призведе до зниження якості.

Практично кажучи, однак торгівля між обмеженнями не завжди можлива. Наприклад, кидання грошей (і працівників) на повністю укомплектований проект може уповільнити його. Більше того, у погано реалізованих проектах часто можна збільшити бюджет, графік або обсяг, не впливаючи негативно на якість.

Цей трикутник використовується для аналізу програмних проектів. Регулярно зловживають, щоб охарактеризувати досягнення як надання необхідних функцій, з розумною якістю, у складеному плані витрат та графіку.

Трикутник управління проектами явно недостатній як модель успіху проекту, оскільки він виключає основні показники успіху, включаючи вплив на зацікавлені сторони, навчання та задоволеність клієнтів.

Зовсім недавно трикутник управління проектами поступився місцем діаманту управління проектами - з витратами, часом, обсягом та якістю, як чотири вершини, і очікуваннями споживачів як центральна тема.

Немає двох клієнтів, які мають однакові очікування. Ви повинні чітко запитати про очікування кожного клієнта. Якщо ви не знаєте, які ці очікування, ви не сподіваєтесь їх задовольнити.

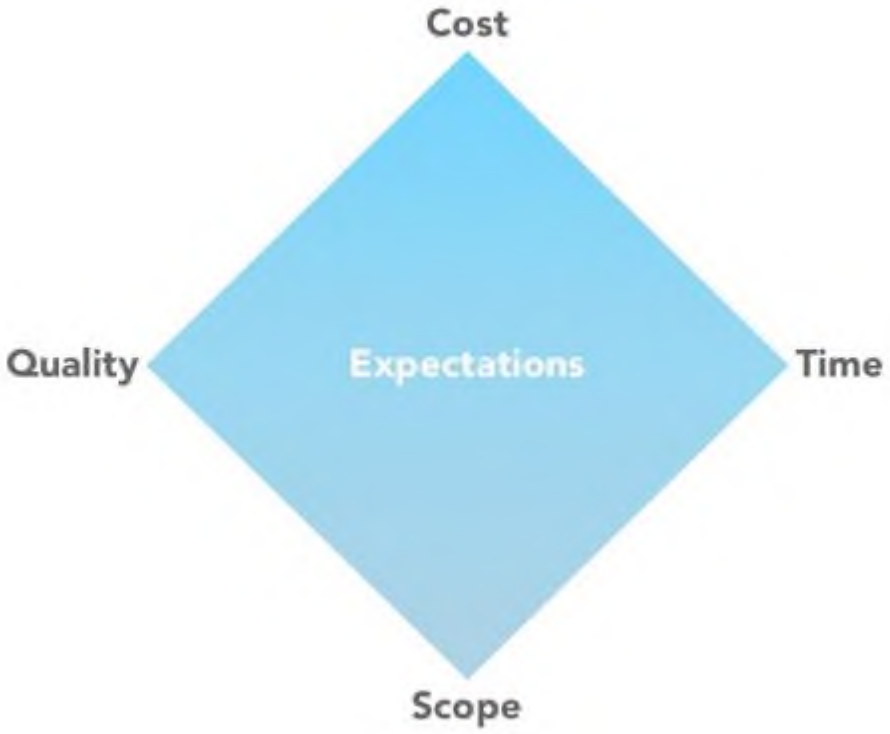


Рис. 1. 2 Алмаз управління проектами

Людина, яка бере на себе відповідальність за виконання програмного проекту, називається менеджером програмного проекту. Ця особа добре знає всі етапи SDLC, через які пройде програмне забезпечення. Керівник проекту може ніколи не брати безпосередньої участі у створенні кінцевого продукту, але він керує та контролює діяльність, пов'язану з виробництвом.

Керівник проекту - це особа, яка несе загальну відповідальність за успішне ініціювання, планування, проектування, виконання, моніторинг, контроль та закриття проекту. Будівництво, нафтохімія, архітектура, інформаційні технології та багато різних галузей промисловості, які виробляють продукцію та послуги, використовують цю посаду.

Керівник проекту повинен володіти комбінацією навичок, включаючи здатність задавати проникливі питання, виявляти невказані припущення та вирішувати конфлікти, а також більш загальні навички управління.

Ключовим серед обов’язків керівника проекту є усвідомлення того, що ризик безпосередньо впливає на ймовірність успіху, і що цей ризик повинен вимірюватися як офіційно, так і неофіційно протягом усього періоду проекту.

Ризики виникають через невизначеність, і успішний керівник проекту - це той, хто зосереджує увагу на цьому як на своїй головній проблемі. Більшість питань, які впливають на проект, так чи інакше призводять до ризику. Хороший менеджер проектів може значно зменшити ризик, часто дотримуючись політики відкритого спілкування, забезпечуючи можливість кожному значному учаснику висловити свою думку та занепокоєння.

Керівник проекту - це людина, яка відповідає за прийняття рішень, як великих, так і малих. Керівник проекту повинен переконатися, що вони контролюють ризик та мінімізують невизначеність. Кожне рішення, яке приймає керівник проекту, має безпосередньо приносити користь їх проекту.

Керівник проекту уважно стежить за процесом розробки, виконує та готує різні плани, організовує необхідні та адекватні ресурси, робить можливим спілкування між усіма членами команди для вирішення питань вартості, активів, часу, плану витрат, якості та задоволеності споживачів.

Існує кілька обов'язків, які ця людина повинна виконувати:

Проект контролю

- Налаштування та визначення особливостей проекту
- Зробіть необхідний крок, щоб вийти або уникнути проблем
- Моніторинг просування та виконання
- Аналіз ризиків на кожному етапі
- Моніторинг управлінської діяльності
- Виступати представником проектної компанії

Нагляд за людьми

- Встановлення ієрархії звітності тощо.
- Управління персоналом
- Зв'язок з партнерами
- Виступати в ролі керівника проекту

1.1. Фази проекту та життєвий цикл проекту

Project Management Institute, Inc. (PMI) визначає управління проектами як "застосування знань, умінь, інструментів та методів до широкого кола діяльності з метою задоволення вимог конкретного проекту". Процес керування та управління проектом від початку до кінця можна розділити на 5 основних фаз:

- Зачаття та ініціація;
- Визначення та планування;
- Виконання;
- Продуктивність та контроль;
- Закрити.

Ідея проекту буде ретельно вивчена, щоб визначити, чи є це вигідним для організації чи ні. На цьому етапі команда, яка приймає рішення, визначить, чи реально реалізувати проект.

Етапи ініціації проекту можуть включати наступне:

- Проведення техніко-економічного обґрунтування;
- Визначення сфери застосування;
- Визначення результатів;
- Визначення зацікавлених сторін проекту;
- Розробка бізнес-кейсу.

План проекту, статут проекту та / або обсяг проекту можуть бути написані в письмовій формі, в яких викладається робота, яку потрібно виконати. На цьому етапі команда повинна розставити пріоритети проекту, розрахувати бюджет і графік і визначити, які ресурси потрібні.

Етапи етапу планування проекту можуть включати наступне:

- Створення плану проекту;
- Створення документів робочого циклу абообробляти карти;
- Оцінка бюджету та створення фінансового плану;
- Збір ресурсів;
- Передбачення ризиків та потенційних якісних перешкод.

З чітким визначенням проекту та набором детальних планів проекту, ви тепер готові перейти до фази виконання проекту. Це фаза, на якій результати

фізично будуються та представляються замовнику для прийняття. Поки будується кожен кінцевий результат, здійснюється набір процесів управління для моніторингу та контролю результатів, що виводяться проектом. Ці процеси включають управління часом, витратами, якістю, змінами, ризиками, проблемами, постачальниками, клієнтами та комунікацією. Після того, як всі результати будуть виготовлені, і замовник прийняв остаточне рішення, проект готовий до закриття.

Етапи виконання етапу проекту можуть включати наступне:

- Створення завдань та організація робочих процесів
- Інструктаж членів команди про завдання
- Спілкування з членами команди, клієнтами та вищим керівництвом
- Моніторинг якості роботи
- Управління бюджетом

Менеджери проектів порівнюватимуть стан проекту та прогрес із фактичним планом, оскільки ресурси виконують заплановану роботу. На цьому етапі менеджерам проектів, можливо, доведеться коригувати графіки або робити те, що необхідно, щоб тримати проект на шляху.

Закриття проекту передбачає надання кінцевих результатів замовнику, передачу проектної документації бізнесу, розірвання контрактів з постачальником, звільнення проектних ресурсів та повідомлення про закриття проекту всім зацікавленим сторонам. Останнім кроком, що залишився, є проведення Огляду після впровадження, щоб визначити рівень успіху проекту та відзначити будь-які уроки, отримані для майбутніх проектів.

Етапи до фази закриття проекту можуть включати наступне:

- Аналіз ефективності проекту
- Аналіз ефективності роботи команди
- Документування закриття проекту
- Проведення оглядів після впровадження
- Облік використаного та невикористаного бюджету

1.2. Визначення моніторингу проекту

Моніторинг проекту стосується процесу відстеження всіх показників, пов’язаних з проектом, включаючи виконання команди та тривалість завдання, виявлення потенційних проблем та вжиття коригувальних заходів, необхідних для забезпечення того, щоб проект був у межах обсягу, бюджету та дотримання визначених термінів. Простіше кажучи, моніторинг проекту контролює всі завдання та стежить за проектною діяльністю, щоб переконатися, що ви реалізовуєте проект за планом.

Основним завданням менеджерів проектів з моніторингу проекту є отримання уявлення про реалізацію проекту, щоб вони могли визначити, чи потрібні якісь дії для забезпечення виконання цілей проекту.

Оскільки цілі проекту пов'язані із зусиллями, графіком та якістю, моніторинг зосереджується на цих аспектах. Для проекту можуть проводитися різні рівні моніторингу. Три основні рівні моніторингу - це рівень діяльності, звіт про стан та етап. Вимірювання, проведені для проекту, використовуються для моніторингу.

Моніторинг рівня активності гарантує, що кожна подія в детальному розкладі була зроблена правильно та протягом часу. Цей тип моніторингу можна проводити щодня на засіданнях команди проекту або керівником проекту, перевіряючи стан усіх завдань, які планується виконати на цей день. Виконане завдання часто позначено як 100% виконане в детальному розкладі - воно використовується такими інструментами, як Microsoft Project, щоб відстежувати відсоток виконання загального проекту або завдання вищого рівня. Цей моніторинг повинен забезпечити, щоб проект продовжував працювати у відповідності до запланованого графіку.

Звіти про стан часто готуються щотижня, щоб розповісти, що сталося, і що потрібно зробити. Звіти про стан зазвичай містять короткий опис дій, успішно виконаних з моменту останнього звіту про стан, будь-яких затриманих подій, будь-яких проблем у проекті, які потребують уваги, і якщо все станеться наступного тижня. Знову ж таки, мета - забезпечити продовження проекту згідно із запланованим графіком.

Етап аналізується на кожному етапі або кожні кілька тижнів, якщо кроки занадто віддалені та складніші. Аналіз фактичних та прогнозованих зусиль та графіку часто включається в аналіз етапів. Якщо відхилення значне, це може означати, що проект може зіткнутися з труднощами і може не відповідати своїм цілям.

У цій ситуації керівники проектів розуміють причини змін і, якщо потрібно, застосовують коригувальні та профілактичні дії. Також можна повідомити про дефекти, які виявляються різними завданнями контролю якості, а також про кількість виправлених дефектів. Цей звіт відстежує хід проекту стосовно усіх цілей.

Рис. 1. 3 Основні процеси управління проектами

Рисунок 1.3 показує основні процеси управління проектами та людей, які відповідають за ці завдання.

Моніторинг приносить користь ключовим учасникам розвитку громади наступними способами:

Для виконавців проектів моніторинг може покращити управління. Моніторинг прогресу щодо визначених цілей, менеджер проекту може оцінити, що працює, а що ні, а звідти може визначити, які зміни слід внести до проекту.

Цей потік дає можливість покращити спосіб, що робиться в проектній організації.

Для компаній, незалежно від того, виконують проект чи підтримують його через партнерство чи фінансування, моніторинг може використовуватися для демонстрації прогресу внутрішньому управлінню та зовнішнім зацікавленим сторонам. Внутрішні вимірювані результати можуть виправдати продовження фінансування та пояснити рентабельність інвестицій, спрямованих на розвиток громади, для керівників та акціонерів. Зовні результати моніторингу можуть продемонструвати прихильність та компетентність у розвитку громади, і таким чином допомогти компанії зберегти свою соціальну ліцензію на діяльність. Це змушує компанії приймати обґрунтовані рішення щодо великих проектів, що здійснюються, і знати, куди інвестувати.

Для членів громади, які беруть участь у моніторингу, є можливість впливати на розробку та виконання проектів розвитку громади. Більше того, забезпечуючи зворотний зв’язок щодо того, чи досягають програми цілей відповідно до потреб та бажань громади, моніторинг є потужним механізмом підзвітності.

1.3. Оцінка проекту та його основні характеристики

Процес, інтегрований із процесом розробки програмного забезпечення, може допомогти проектам створити надійні та реалістичні плани для виконання зобов'язань та реалізації вимог проекту, які називаються оцінкою програмного забезпечення.

Крім того, він може підтримувати іншу управлінську діяльність, надаючи своєчасне планування та точну інформацію. Для ефективного управління точна оцінка різних заходів є абсолютно необхідною. За правильної оцінки менеджери можуть ефективніше та ефективніше контролювати проект і керувати ним.

У ідеальному світі оцінка повинна бути складена з використанням наступних десяти кроків:

1. Встановити оціночний обсяг і мету;

Визначте та задокументуйте очікування. Якщо всі учасники зрозуміють обсяг і мету кошторису, ПМ матиме не лише базу, на якій можна виміряти вплив

майбутніх змін. Прем'єр-міністр також усуне непорозуміння між проектною групою та пояснить суперечливі припущення щодо того, що очікується.

Документування специфікацій заявок, включаючи технічні деталі, зовнішні залежності та бізнес-вимоги, надає цінну інформацію для оцінки ресурсів, необхідних для завершення проекту. Чим детальніше технічні характеристики, тим краще. Тільки тоді, коли ці вимоги відомі та зрозумілі, ПМ може визначити реальні витрати на розробку.

Цитата повинна розглядатися як живий документ. Коли дані змінюються або стає доступною нова інформація, вона повинна бути задокументована та включена до кошторису, щоб забезпечити цілісність проекту.

2. Встановити технічну базову лінію, основи та припущення;

Щоб створити точну технічну базову лінію, спочатку потрібно визначити функціональність, яку потрібно оцінити. Визначте обмеження, пов’язані із додатком та проектом, та визначте, яку функціональність необхідно розробити порівняно з функціями, які можна розгорнути або повторно використати через COTS.

Основні правила - це стислі твердження, що описують основу оцінки. "Ця оцінка містить лише функції a, b і c, ніяких витрат, пов'язаних з поїздками" є прикладом основного правила.

Припущення - це припущення, що описують невідомі змінні, що впливають на оцінку. "Ця оцінка передбачає, що розробник програмного забезпечення використовуватиме систему розробки X" - це приклад припущення. Пізніше в проекті, якщо ви підтвердите, що буде використана "Система розробки X", це перевірене припущення буде перероблено як основне правило.

Принципи та припущення складають основу оцінки, і хоча вони є попередніми на ранніх стадіях оцінки і, отже, піддаються невизначеності, вони повинні бути надійними та задокументованими.

3. Збір даних;

Була отримана певна основна інформація для забезпечення узгодженої оцінки проекту. Не всі дані надходять з одного джерела і недоступні одночасно.

Коли збирається нова інформація, ви вже маєте організовану та ретельну систему документації.

Загальну форму слід адаптувати до кожного з параметрів, які не мають значення для поточної оцінки. Загалом, чим менше запитань вам доведеться задати щодо своїх джерел, тим вони будуть щасливішими.

4. Розмір програмного забезпечення;

Розмір, як правило, є найважливішим (але, звичайно, не єдиним) фактором витрат і часу. Загальний обсяг програмного проекту визначається шляхом визначення не тільки кількості нового програмного забезпечення, яке потрібно розробити, але також кількості існуючого COTS та іншого програмного забезпечення, яке буде інтегровано в нову систему. На додаток до оцінки розміру товару, ви повинні оцінити будь-які переробки, необхідні для розробки товару. Вони зазвичай виражаються у вигляді рядків вихідного коду (SLOC) або функціональних точок, хоча є й інші можливі одиниці виміру. Для визначення загальної невизначеності передбачуваний розмір слід подавати як найменш вірогідну маржу.

5. Підготувати базовий кошторис;

Бюджет і графік походять з кошторисів. Отже, якщо оцінка є неправильною, отримані графіки та бюджети, ймовірно, будуть неточними. З огляду на важливість завдання оцінки, розробники, які хочуть вдосконалити свої навички оцінки програмного забезпечення, повинні розуміти та застосовувати деякі основні процедури. Перш за все, для розміру програмного забезпечення та підготовки кошторисів слід використовувати кваліфікованого, досвідченого та кваліфікованого персоналу. По-друге, дуже важливо, щоб вони були забезпечені правильними технологіями та інструментами. По-третє, керівник проекту повинен визначити та впровадити зрілий, задокументований та повторюваний процес оцінки.

Оцінка базової лінії може бути побудована з використанням різноманітних підходів, включаючи вгадування (що не рекомендується), виключне використання наявних даних про продуктивність, підхід знизу вгору, експертні оцінки та моделі витрат.

6. Кількісна оцінка ризиків та аналіз ризиків;

Багато подій відбувається під час розробки програмного забезпечення. Ризик характеризується втратою часу чи якості, грошей, контролю, розуміння тощо. Втрата, пов'язана з ризиком, називається наслідком ризику.

Прем'єр-міністр повинен мати уявлення про ймовірність того, що подія відбудеться. Імовірність ризику, виміряна від 0 (неможливо) до 1 (безпека), називається ймовірністю ризику. Якщо ймовірність ризику дорівнює 1, ризик називається проблемою, оскільки це станеться безпечно.

Для кожного ризику нам потрібно визначити, що може зробити ПМ, щоб мінімізувати або уникнути впливу події. Контроль ризику включає ряд заходів для зменшення або усунення ризику.

Управління ризиками допомагає вам виявити та вирішити потенційні загрози проекту, чи то через внутрішні проблеми чи умови, чи зовнішні фактори, які ви не можете контролювати. Проблеми, пов’язані з розмірами та оцінкою програмного забезпечення, можуть мати суттєвий негативний вплив. Ключове слово тут потенційно означає, що якщо проблеми передбачувані та своєчасно усуваються їх причини, вплив можна пом'якшити. Процес управління ризиками є засобом для цього.

7. Підтвердження оцінки та огляд;

Існує багато способів перевірити оцінку. Необхідно оцінити як процес створення кошторису, так і саму кошторис. В ідеалі перевірку повинен робити той, хто особисто не брав участі у складі кошторису і може розглянути його об’єктивно. Аналітик, який підтверджує кошторис, повинен використовувати різні методи, інструменти та окремо зібрані дані, що використовуються в кошторисі, що підлягає аудиту.

Переглядаючи оцінку, ПМ повинен оцінити припущення, зроблені в процесі оцінки. Переконайтеся, що прийняті принципи послідовно застосовуються протягом оцінки. Нижчі витрати та ризики, пов’язані з винятковими вимогами, були занижені або недооцінені, тоді як оцінки продуктивності могли бути завищеними. Повзуче збільшення вимог, можливо, призвело до більшої невизначеності, ніж було враховано в початковій оцінці.

8. Скласти план проекту;

Процес створення плану проекту включає оцінку та розподіл витрат та графіків для функціональної та орієнтованої на завдання структури проекту. Такі моделі, як SEER Client для Microsoft, запускають цю функцію автоматично. Вісім найважливіших фаз розробки програмного забезпечення:

- Приймання;
- Придбання;
- Після розгортання;
- Тест коду та одиниці;
- Концепція;
- Дизайн;
- Інтеграція;
- Вимога.

9. Оцінка документа та вивчені уроки;

Кожного разу, коли ПМ робить пропозицію і знову в кінці розробки програмного забезпечення, ПМ повинен задокументувати відповідну інформацію, яка складає кошторис, та записати уроки, які Ви вивчили. Таким чином, прем'єр-міністр має докази того, що його процес є дійсним і що він зробив оцінку добросовісно, і він матиме фактичні результати, за допомогою яких можна відкалібрувати ваші моделі оцінки. Обов’язково задокументуйте відсутність або неповну інформацію, а також ризики, проблеми та проблеми, спричинені процесом, а також будь-які ускладнення, що мали місце. Крім того, задокументуйте всі ключові рішення, прийняті під час розробки ціни, результати та вплив ваших дій. Нарешті, опишіть і задокументуйте динаміку, яка відбулася під час процесу. Сюди входять взаємодія вашої команди оцінювачів, інтерфейси з вашими клієнтами, і компроміси, які ви зробили при вирішенні проблем, виявлених під час процесу. Моделі витрат, засновані на фактичних витратах попередніх проектів, можуть бути відкалібровані та продемонстровані їх точність шляхом порівняння вартості ваших поточних кошторисів із попередніми даними

проекту та фактичної вартості вашого завершеного проекту, тим самим коригуючи параметри використання моделі для покращення майбутньої точності.

10. Відстежуйте проект протягом розвитку.

Оцінка розміру, вартості та часу програмного забезпечення має бути постійним процесом. Попередні оцінки можуть знадобитися для завершення роботи або започаткування процесу розробки, або вам може знадобитися провести аналіз витрат-вигод або рентабельності інвестицій (ROI) для оцінки прибутковості проекту.

Попередні оцінки є найскладнішими в розробці та найменш точними через неповний характер наявної інформації та інші обговорювані фактори.

Оцінка вартості програмного забезпечення є складним процесом, але необхідною частиною успішної розробки програмного забезпечення.

Звукова оцінка починається зі структури розподілу робіт (WBS). WBS - це перелік завдань, які, якщо їх виконати, дадуть кінцевий продукт. Спосіб розбиття роботи диктує, як це буде зроблено. Існує багато способів розкласти проект на завдання. Проект може бути розбитий за ознаками, за фазою проекту (завдання вимог, завдання проектування, завдання програмування, завдання контролю якості тощо) або за деякою їх комбінацією. В ідеалі, WBS повинен відображати шлях розробки попередніх проектів.

Після створення WBS команда повинна скласти оцінку зусиль, необхідних для виконання кожного завдання. Найточнішими є ті оцінки, які спираються на попередній досвід. Члени команди повинні переглянути попередні результати проекту та з’ясувати, скільки часу тривало виконання подібних завдань у попередніх проектах. Джерела затримок у минулому слід враховувати при складанні поточних оцінок.

Оцінка проекту може включати супровід:

- Розмір програмного забезпечення
- Зусилля
- Час
- Вартість

Розмір програмного забезпечення може бути оцінений шляхом обчислення кількості функціональних точок або через KLOC (кілограмний рядок коду) у програмному забезпеченні. Точки функцій змінюються залежно від вимог користувача або програмного забезпечення, а рядки коду залежать від практики кодування.

Менеджери оцінюють зусилля з точки зору потреб персоналу та людських годин, необхідних для виробництва програмного забезпечення. Для оцінки зусиль повинен бути відомий розмір програмного забезпечення. Це можна отримати з історичних даних організації, досвід менеджерів або розмір програмного забезпечення можна перетворити на зусилля, використовуючи деякі стандартні формули.

Після оцінки зусиль і розміру можна оцінити час, необхідний для виготовлення програмного забезпечення. Необхідні зусилля розділені на підкатегорії як взаємозалежність різних компонентів програмного забезпечення відповідно до необхідних специфікацій. Задачі програмного забезпечення WBS поділяють на менші заходи, завдання чи події.

Завдання плануються щоденно або на календарні місяці. Сума часу, необхідного для виконання всіх завдань у годинах або днях, - це загальний час, витрачений на завершення проекту.

Це можна вважати найскладнішим з усіх, оскільки воно залежить від більшої кількості елементів, ніж будь-який з попередніх. Для оцінки вартості проекту необхідно врахувати:

- Залучені подорожі;
- Навчання та підтримка;
- Якість програмного забезпечення;
- Кваліфікований персонал зі спеціальними навичками;
- Розмір програмного забезпечення;
- Апаратне забезпечення;
- Спілкування;
- Додаткові ліцензії, програмне забезпечення чи інструменти.

Керівник проекту може оцінити ці фактори, використовуючи наступні методи:

- Розкладання;
- Емпірична оцінка.

Техніка розкладання передбачає програмне забезпечення як продукт різного складу. Є дві основні моделі:

- Рядок коду (LOC);
- Точки функцій (FP).

Техніка емпіричного оцінювання використовує емпірично отримані формули для оцінки. Ці формули базуються на FP або LOC.

Оцінка функціональних точок проводиться від імені ряду функціональних точок у програмному продукті.

Оцінка LOC проводиться від імені ряду рядків кодів у програмному продукті.

Наприклад, в методі емпіричної оцінки можна використовувати модель конструктивних витрат (COCOMO) або модель Putnam.

Модель Putnam - це експериментальна модель оцінки зусиль програмного забезпечення. Перша робота Лоуренса Х. Патнама, розповсюджена в 1978 р., Розглядається як перша робота в галузі програмного моделювання процесів. Як група, емпіричні моделі працюють, збираючи дані програмного проекту та пристосовуючи криву до даних. Оцінки майбутніх зусиль робляться шляхом надання розміру та обчислення відповідних зусиль за допомогою рівняння, яке відповідає вихідним даним (зазвичай з певною помилкою).

Створена Лоуренсом Путнамом-старшим модель Путнам описує зусилля та час, необхідні для завершення проекту програмного забезпечення певного розміру. SLIM (Software LIfecycle Management) - це ім’я, надане Путнамом власним наборам інструментів, розробленим його компанією QSM, Inc. на основі його моделі. Це одна з найперших із цих типів розроблених моделей і є однією з найбільш широко використовуваних. Тісно пов’язаними параметричними моделями програмного забезпечення є COCOMO, Параметричний огляд інформації для калькуляції та оцінки - Програмне забезпечення (PRICE-S) та

Оцінка та оцінка програмного забезпечення - Модель оцінки програмного забезпечення (SEER-SEM).

COCOMO розроблений Barry W. Boehm. Він поділяє програмний продукт на три категорії програмного забезпечення: органічне, здвоєне та вбудоване.

Basic COCOMO корисний для швидкого оцінювання витрат на програмне забезпечення. Однак це не враховує відмінності в якості персоналу, апаратних обмеженнях та досвіді, використанні сучасних інструментів та методів.

1.4. Інструменти управління проектами

Ризик та невизначеність зростають багаторазово щодо розміру проекту, навіть коли проект розробляється відповідно до встановлених методологій.

Доступні інструменти, які допомагають ефективно керувати проектами. Найпоширенішими є:

- WBS
- Діаграма Ганта
- Діаграма PERT
- Гістограма ресурсу
- Аналіз критичного шляху

Структура розподілу робіт - це ключовий проект, який організовує роботу команди у керовані секції.

Звід знань про управління проектами [1] визначає структуру розподілу робіт як "орієнтовану на результат ієрархічну декомпозицію роботи, що виконується командою проекту".

Структура розподілу робіт візуально визначає обсяг на керовані фрагменти, які команда проекту може зрозуміти, оскільки кожен рівень структури розподілу робіт надає подальше визначення та деталі.

З цим визначенням зрозуміло, що WBS надає однозначну заяву про цілі та результати роботи, яку потрібно виконати. Він являє собою чіткий опис обсягу проекту, результатів та результатів - “що” проекту. WBS - це не опис процесів, які виконуються для виконання проекту ... і не стосується графіка, який визначає, як і коли будуть виготовлені результати, а, скоріше, конкретно обмежується описом та деталізацією результатів або обсягу проекту. WBS є основоположним

компонентом управління проектами, і як такий є критично важливим фактором для інших процесів управління проектами та результатів, таких як визначення діяльності, діаграми мережевих графіків проектів, графіки проектів та програм, звіти про ефективність, аналіз ризиків та реагування, інструменти контролю, або організація проекту.

Елементом структури розподілу робіт може бути товар, дані, послуга або будь-яка їх комбінація. WBS також надає необхідну структуру для детальної оцінки витрат та контролю, а також надає вказівки щодо розробки та контролю графіка [2].

Команда проекту створює структуру розподілу робіт за проектом, визначаючи основні функціональні результати та розподіляючи ці результати на менші системи та допоміжні результати. Ці допоміжні результати розкладаються далі, поки не може бути призначена одна особа. На цьому рівні визначаються та групуються конкретні робочі пакети, необхідні для виготовлення матеріалів, що подаються.

Робочий пакет представляє перелік завдань або "завдань" для створення конкретної одиниці роботи. Якщо ви бачили детальні графіки проектів, тоді ви визнаєте завдання в робочому пакеті як "матеріал", який люди повинні виконати до певного часу та на певному рівні зусиль.

З точки зору витрат, ці робочі пакети, як правило, групуються та призначаються певному відділу для виготовлення роботи. Ці підрозділи, або рахунки витрат, визначені в структурі розподілу організації і їм виділяється бюджет для отримання конкретних результатів.

Інтегруючи рахунки витрат із структурної структури організації та структури роботи проекту, вся організація може відстежувати фінансовий прогрес на додаток до результатів проекту.

Приклад WBS, показаний на малюнку 1.4.

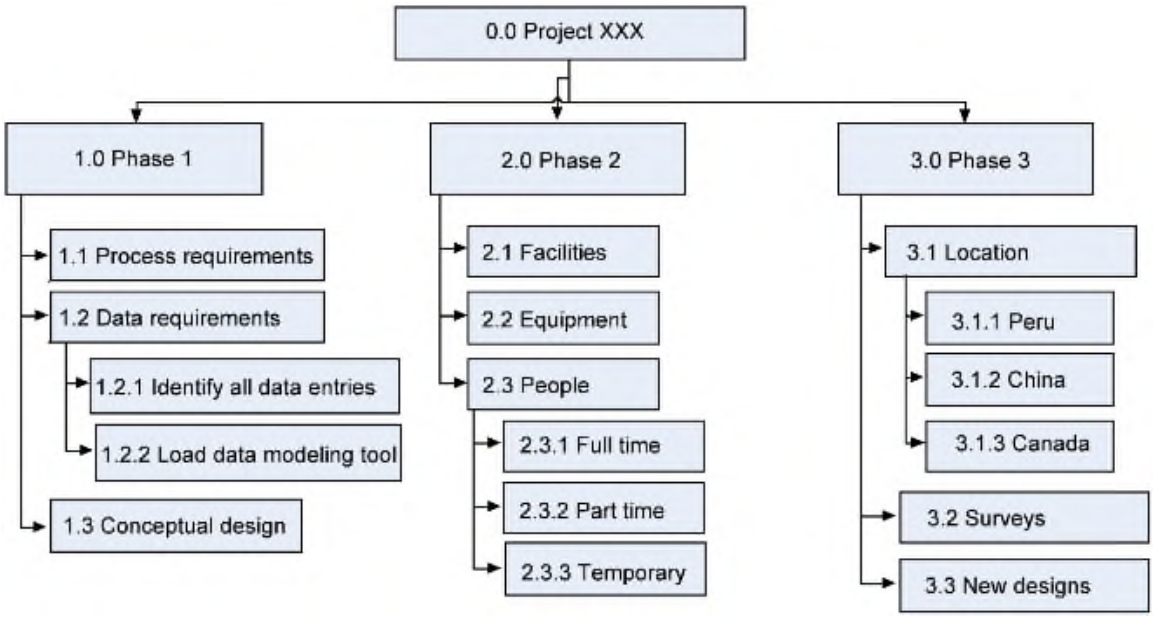


Рис. 1. 4 Приклад WBS

Деякі ключові характеристики високоякісних структур розподілу робіт (PMI, 2006) викладені нижче:

- Центральним атрибутом WBS є те, що він "орієнтований на результати" [3].
- Додатковим ключовим атрибутом WBS є те, що це «ієрархічна декомпозиція твору».
- Правило 100% [4] є одним з найважливіших принципів, що керує розробкою, розкладанням та оцінкою СРБ.
- WBS може бути представлений різними способами, включаючи графічний, текстовий або табличний подання. Форму представництва слід вибирати виходячи з потреб конкретного проекту.

Ключові документи для початку розробки WBS включають:

- Статут проекту
- Постановка проблеми проекту або визначення сфери застосування
- Застосовувана документація договору або угоди
- Існуюча практика управління проектами

Діаграми Ганта - це горизонтальна стовпчаста діаграма, яка була розроблена Генрі Ганттом у 1917 році як інструмент контролю виробництва.

Він представляє графік проекту щодо періодів часу. Діаграма Ганта будується з горизонтальною віссю, що представляє загальний проміжок часу

проекту, розбитий на кроки (наприклад, дні, тижні або місяці) і вертикальною віссю, що представляє завдання, з яких складається проект (наприклад, якщо проект оснащує ваш комп’ютер новим програмним забезпеченням, основними завданнями, що беруть участь, можуть бути: проведення досліджень, вибір програмного забезпечення, встановлення програмного забезпечення). Горизонтальні смуги різної довжини представляють послідовності, терміни та проміжок часу для кожного завдання. Використовуючи той самий приклад, ви б поставили "провести дослідження" у верхній частині вертикальної осі і намалювали на графіку смужку, яка представляє кількість часу, який ви очікуєте витратити на дослідження, а потім введіть інші завдання нижче першого та репрезентативні смуги в ті моменти часу, коли ви плануєте їх виконати. Проміжки стовпчиків можуть перекриватися, як, наприклад, ви можете проводити дослідження та вибирати програмне забезпечення протягом того самого проміжку часу. У міру завершення проекту можуть бути додані вторинні стовпчики, наконечники стріл або затемнені смуги для позначення виконаних завдань або частин виконаних завдань. Вертикальна лінія використовується для подання дати звіту.

Діаграми Ганта дають чітку ілюстрацію стану проекту, але одна проблема з ними полягає в тому, що вони не вказують залежностей завдань - ви не можете сказати, як одне завдання, що відстає від графіка, впливає на інші завдання.

Приклад діаграми Ганта, зображеної на рисунку 1.5.

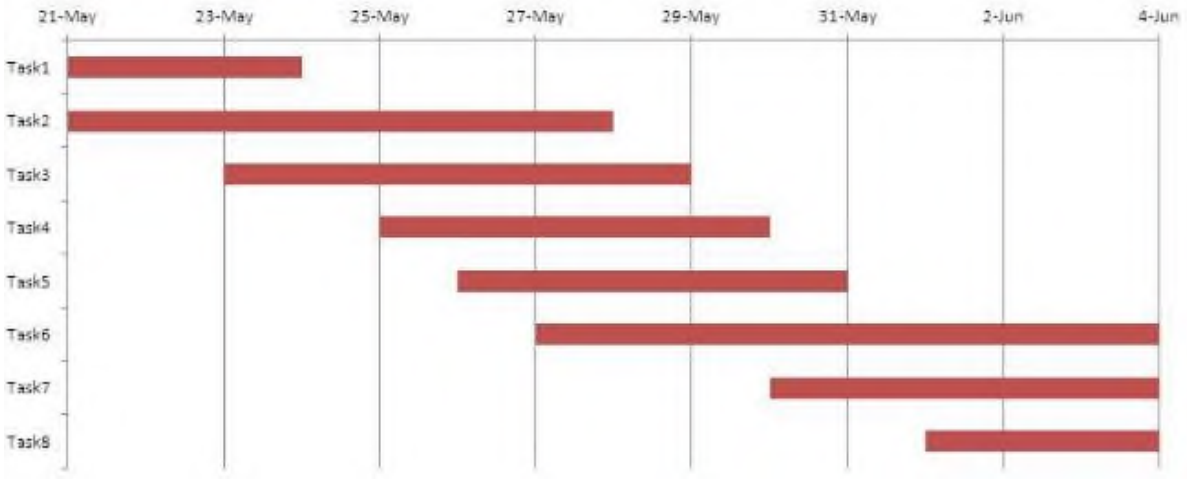


Рис. 1. 5 Діаграма Ганта

Діаграма PERT - це інструмент управління проектами, який використовується для планування, організації та координації завдань у рамках проекту. PERT розшифровується як "Техніка оцінки програм", методологія, розроблена ВМС США в 1950-х роках для управління підводною ракетною програмою "Поларіс". Він здатний графічно представляти основні події проекту як паралельно, так і послідовно. Події, що відбуваються одна за одною, показують залежність пізнішої події від попередньої.

Події відображаються як нумеровані вузли. Вони з'єднані стрілками з маркованими напрямками, що відображають послідовність завдань у проекті. Напрямок стрілок на лініях вказує на послідовність завдань.

Завдання, яке необхідно виконати в послідовності, називається залежним або послідовним завданням. Завдання, які не залежать від завершення одного для початку іншого, і можуть виконуватися одночасно. Ці завдання називаються паралельними або паралельними. Завдання, які потрібно виконувати послідовно, але не вимагають ресурсів або часу виконання, вважаються залежними від події. Вони представлені пунктирними лініями зі стрілками і називаються фіктивними видами діяльності.

Приклад діаграми PERT, показаний на рисунку 1.6.

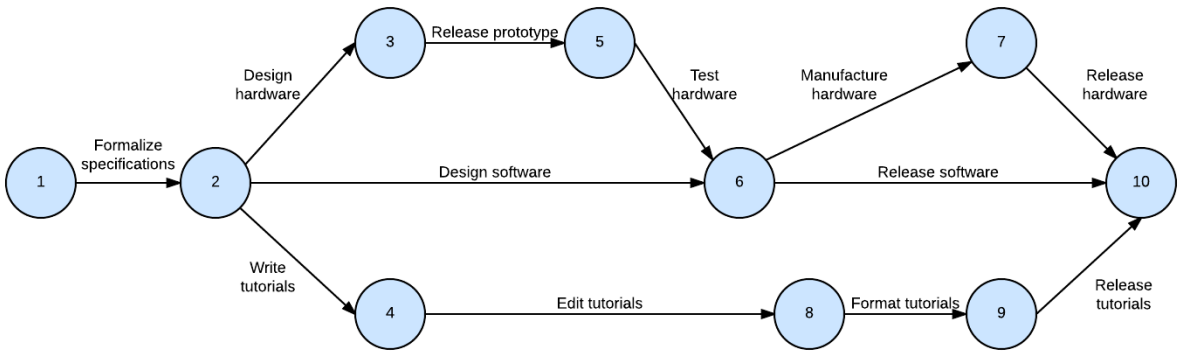


Рис. 1. 6 Діаграма PERT

Аналіз критичного шляху - це техніка, яка визначає діяльність, необхідну для виконання завдання, включаючи визначення часу, необхідного для виконання кожної діяльності, та взаємозв’язків між діями. Також відомий як метод критичного шляху, аналіз критичного шляху допомагає передбачити, чи проект завершиться вчасно.

Аналіз критичного шляху (CPA) допомагає реорганізувати проект перед початком та в міру його прогресу. CPA допомагає підтримувати завершення проекту в строгому порядку і гарантує, що результати завершаться вчасно. Наприкінці 1950-х років Джеймс Келлі-молодший з Ремінгтона Ренда та Морган Уокер з Дюпона розробили техніку управління проектами, яка називається методом критичного шляху (CPM). CPM допомагає у плануванні проектної діяльності та допомагає керівникам проектів вирішувати складні та невідкладні завдання та проекти з великою кількістю видів діяльності.

CPA виявляє та визначає критичні та некритичні завдання стосовно бізнес-процесу чи робочого плану та обсягу плаваючого потоку, пов'язаного з кожною діяльністю, щоб запобігти затримкам розкладу та вузьким місцям у процесі. CPA є ключовим компонентом у скороченні термінів проекту та контролі витрат, щоб запобігти перевищенню бюджету проекту.

Результати аналізу критичного шляху визначають критичний шлях, послідовний набір пов’язаних та важливих кроків, які складають план роботи, як правило, з нульовим провисанням. CPA вивчає всі варіанти зменшення часу, необхідного для виконання важливих етапів у робочому плані.

При використанні методу критичного шляху деякі дії не можуть розпочатися, поки інші не будуть закінчені, і їх потрібно виконувати послідовно. Ці заходи відомі як послідовні заходи. Наприклад, коли ви готуєте недільну вечерю, шинку не можна подавати до обіду, поки вона не буде ретельно приготована і вийнята з духовки.

CPM діагностує час, необхідний для завершення діяльності, залежності від діяльності, етапи, результати та те, як кожна діяльність пов’язана з попередньою та наступною діяльністю. Метод критичного шляху діагностує проектну діяльність, використовуючи кола та стрілки, щоб зобразити зв’язок між діями та тривалістю кожної діяльності. Ряд дій називається шляхом, а найдовший шлях на діаграмі - критичний шлях. Критичний шлях - це шлях, який виправдовує остаточну часову шкалу проекту.

CPM планує та організовує ресурси, визначає пріоритети завдань, кошториси та мінімізує час проекту, підтримує оцінку витрат та оцінку проекту

та допомагає надавати керівництво проектом. Метод критичного шляху оцінює конкретні завдання, які мають бути завершені, оцінює, де може відбуватися паралельна діяльність, виявляє найшвидший час для завершення проекту, визначає потреби у ресурсах, класифікує послідовність дій та забезпечує планування проекту.

Гістограма ресурсу - це інструмент, який часто використовується командою з управління проектами або як засіб візуального представлення команди та всіх зацікавлених сторін. Якщо говорити конкретно, гістограма ресурсу - це спеціально гістограма, яка використовується для відображення конкретного періоду часу, протягом якого заплановано працювати над певним ресурсом протягом заздалегідь визначеного та конкретного періоду часу. Гістограми ресурсів можуть також містити порівняльну ознаку доступності ресурсів, що використовується для порівняння з метою контрасту. Гістограми ресурсів - справді зручні інструменти, які можна використовувати для команди управління проектами та / або керівника групи управління проектами, оскільки вони дозволяють швидко і легко переглянути на одній сторінці саме те, які ресурси доступні,

Гістограма ресурсів дозволяє одночасно розглядати окремі ресурси в графіку та діаграмі Ганта. Переглядаючи ці два дисплеї одночасно, ми можемо приймати розумні рішення щодо використання ресурсів. Гістограма ресурсу показує обсяг використання та доступність ресурсу, а діаграма Ганта - дії, над якими планується працювати ресурс.

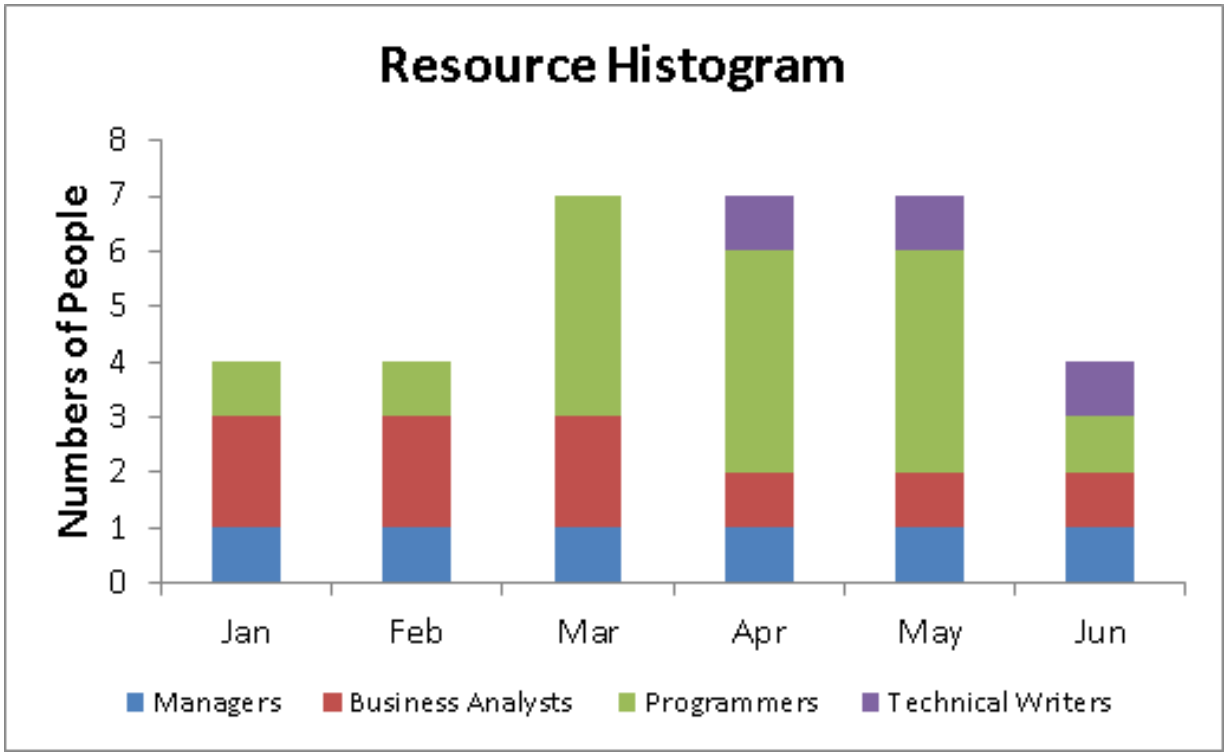


Рис. 1. 7 Гістограма ресурсу

Ці інструменти корисні для розпізнавання взаємозалежних завдань у проекті. Це також допомагає з'ясувати найкоротший шлях або критичний шлях для успішного завершення проекту. Як і діаграма PERT, кожній події відводиться певний часовий проміжок. Цей інструмент показує залежність події за умови, що подія може перейти до наступної, лише якщо попередня завершена.

Події розміщені відповідно до їх максимально раннього часу початку. Шлях між початковим і кінцевим вузлом є критичним шляхом, який неможливо додатково зменшити, і всі події потрібно виконувати в однаковому порядку.

Основні відмінності між діаграмою Ганта та діаграмою Перта, наведені в таблиці 1.1.

Таблиця 1. 1. Порівняння діаграми Ганта та діаграми Перта

Діаграма Ганта	**Діаграма Перта**
Представлено у вигляді гістограми	Представляється з блок-схемою (або мережевою діаграмою)
Використовується для невеликих проектів	Використовується для великих та складних проектів

Вкажіть точну тривалість часу та відсоток завершеності	Потрібно передбачити час
Неможливо відобразити взаємопов’язані завдання, які залежать одне від одного	Має численні взаємопов’язані мережі незалежних завдань

1.5. Висновки

Управління проектами - це процес та діяльність з планування, організації, мотивації та контролю ресурсів, процедур та протоколів для досягнення конкретних цілей у наукових або повсякденних проблемах, а основною метою управління проектами є проект. Для забезпечення належного управління та розподілу завдань між працівниками програмним організаціям необхідно розробити власні системи моніторингу та оцінки. Повний аналіз предметної області на основі отриманої інформації було призначено інструментам моніторингу та оцінки, які найбільш успішно впливають на проект.

РОЗДІЛ 2
ІСНУЮЧІ ПРОГРАМНІ ЗАСОБИ ДЛЯ МОНІТОРИНГУ ТА ОЦІНКИ ПРОГРАМНИХ ПРОЕКТІВ

2.1. Workzone

- це проста у використанні програма для управління проектами та співпраці з документами, розміщена у хмарі. Він існує приблизно з 2000 року і вважається одним із найбільш зрілих та перевірених боями програмного забезпечення для онлайн-управління проектами на ринку. Робоча зона дозволить всім знати, що відбувається з вашими проектами, в одному перегляді. Швидко визначте, які проекти відстають і чому. Розгорніть, щоб побачити додаткові деталі, включаючи підзадачі та взаємозалежності [5].

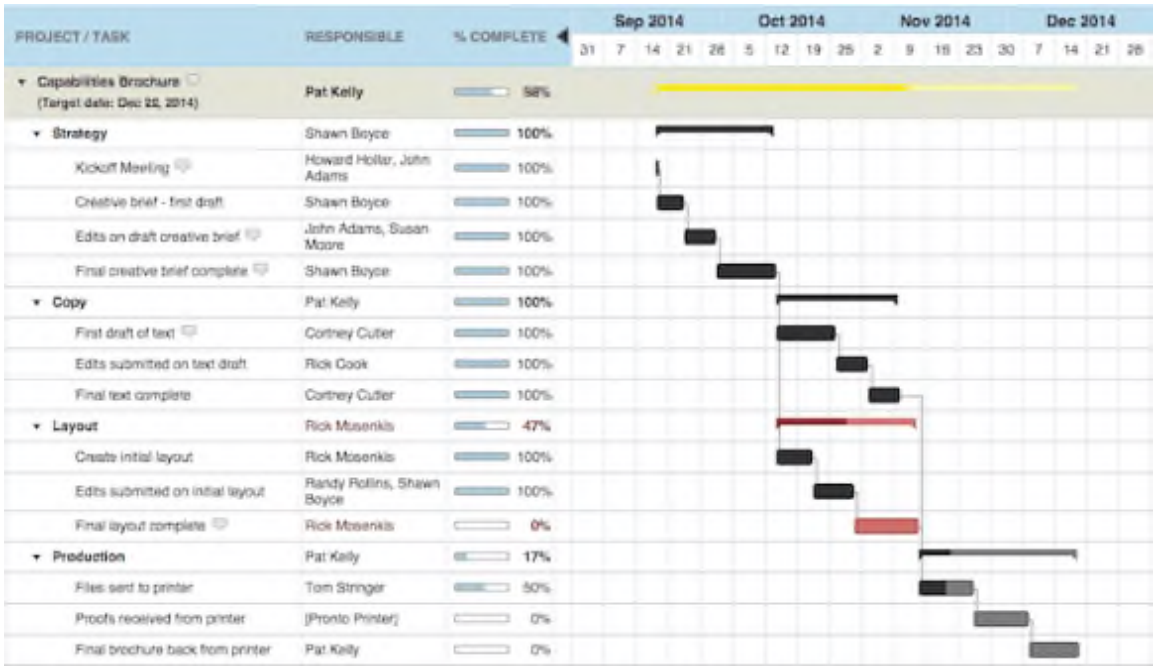


Рисунок 2. 1 Головне вікно робочої зони

Робочу зону створили досвідчені спеціалісти з реклами, які хотіли допомогти агенціям та творчим колективам зрозуміти всю свою роботу.

Основні особливості:

- Персоналізовані, індивідуальні списки справ
- Створюйте підзадачі та залежності від завдань
- Встановіть дозволи певним користувачам (включаючи клієнтів) на доступ до проектів, завдань та файлів
- Швидко переглядайте статуси за допомогою діаграм Ганта
- Доручити людям більше одного завдання

Workzone має безліч потужних функцій, але простіший у використанні, ніж складніші інструменти. Робоча зона призначена для використання всією командою.

Переваги Workzone:

- Інформаційна панель проекту надає портфоліо всіх проектів команди
- Персоналізовані списки справ дозволяють членам команди відстежувати шлях (і їх можна автоматично отримувати щодня по електронній пошті)
- Діліться документами спеціально для певних проектів
- Налаштуйте власні форми прийому та функції звітування
- Встановіть рівні дозволу за проектом та документом, щоб кожна людина бачила лише те, що підходить
- Завдяки Workzone ви також отримаєте першокласну, цілеспрямовану підтримку клієнтів та підтримку, що допоможе всій вашій команді швидко розпочати роботу на правій нозі.

Вартість: $ 24- $ 44 на місяць

2.1. Простий проект

Easy Projects - це набір програмного забезпечення для управління проектами, розроблене компанією Logic Software, що базується в Торонто. Як приклад програмного забезпечення для управління проектами в Інтернеті, Easy Projects розроблено для забезпечення управління мережевими проектами в Інтернеті через браузер, а не за допомогою відстеження вручну чи програмного забезпечення для настільних ПК [6].

Компанія Easy Projects відповідає своїй назві для наших адміністраторів та користувачів. Він має просту та універсальну функціональність, яка може відповідати практично будь-яким нашим потребам. Після того, як ми визначили, як продукт найкраще працюватиме для нас, і налаштували його, ми були дуже задоволені тим, наскільки добре ми рухалися вперед. Користувачі можуть покращити свій робочий процес за допомогою шаблонів, ми знайшли креативне використання існуючих функціональних можливостей для вирішення проблемних проблем, і компанія продовжує прогресувати завдяки новим функціям та вдосконаленням продуктів.

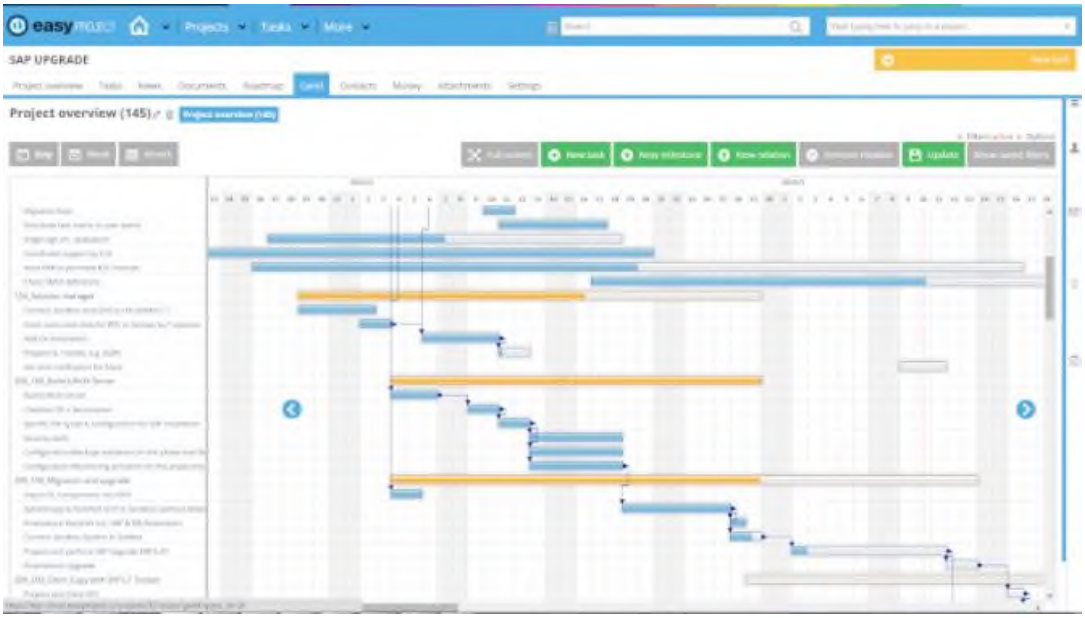


Рисунок 2. 2 Головне вікно простого проекту

Easy Project - це просте, наочне та інтуїтивно зрозуміле програмне забезпечення для управління проектами.

Основні особливості:

- Діаграма Ганта;
- Управління ризиками;
- Управління портфелем;
- Шаблони проектів.

2.2. **Програмне забезпечення для управління двома планами**

Простий проект заснований на WBS, методах Ганта, Agile, зароблений вартості та інших найкращих практиках роботи з ПМ. Він сумісний з модулями

для ресурсів, фінансів, довідкової служби та CRM. Незабаром з'являться інші чудові плагіни.

Команда Easy Software пропонує вам професійні послуги - впровадження, підтримку, регулярні вебінари та електронне навчання [7].

З переповненості програмного забезпечення з відкритим кодом виділяється 2-Plan. Система має три симбіотичні програми - усі безкоштовні:

2-Plan Team, веб-інструмент управління проектами з безліччю варіантів хостингу. Працюйте 2-gether, дошка управління завданнями на основі Scrum для проектів однієї команди. Усі ці варіанти безкоштовні, але ви можете оплатити додаткові розширення.

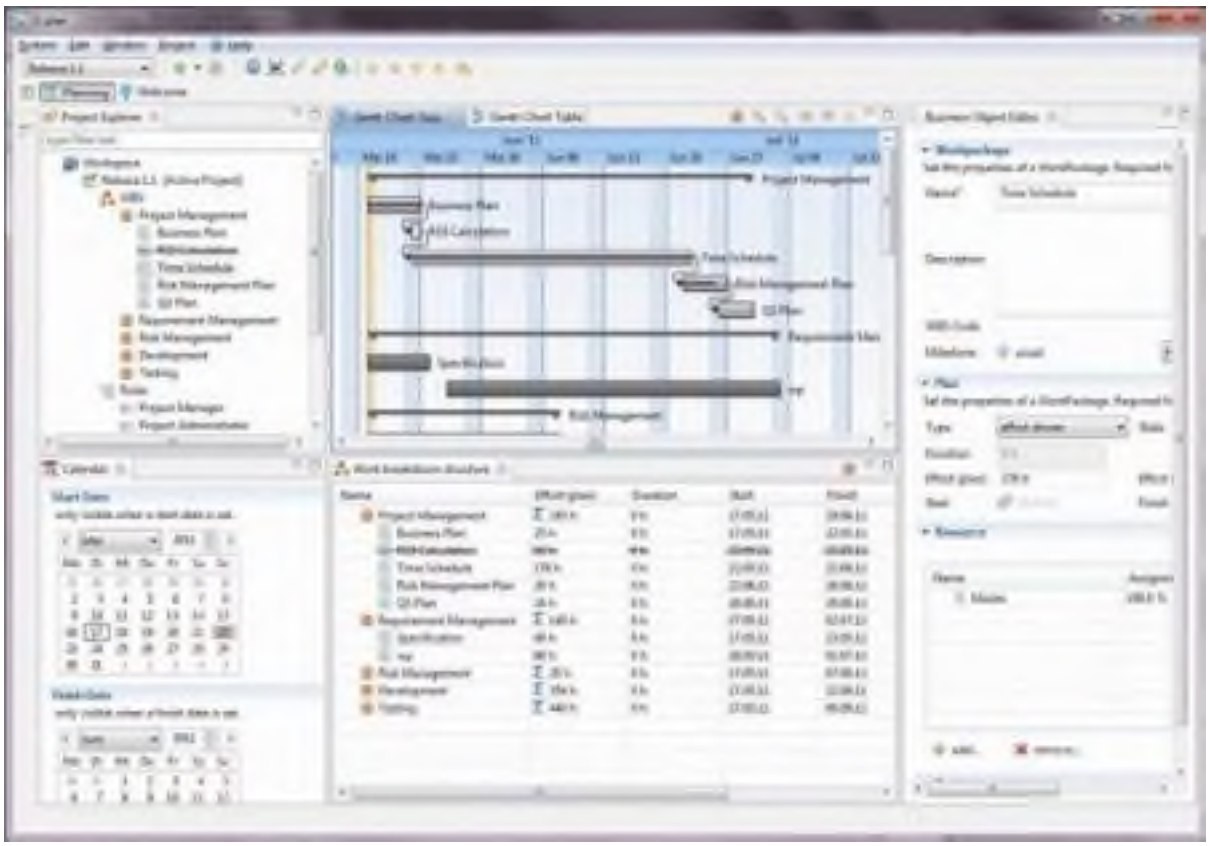


Рисунок 2. 3 Головне вікно 2-плану програмного забезпечення для управління проектами

Плюси

2-план - це махінація, коли справа стосується особливостей. Лише у настільній версії менеджери проектів можуть створювати анімовані графічні WBS, розробляти етапи проекту, реалізовувати планування зверху вниз і знизу вгору, а також будувати системи управління проектами.

Команда 2-Plan дозволяє легко координувати роботу з командами поза містечками та відстежувати час, витрачений на виконання завдань, - і вона легко інтегрується з робочим столом 2-Plan. Робота 2-Gether подібна до Trello тим, що вона використовує систему Kanban, але вона також має можливість розширюватися в більшу робочу діаграму.

Мінуси

2-Plan пропонує безліч функцій, які швидко можуть стати непосильними для команд, у яких мало часу на їх обширні посібники. Крім того, безкоштовний варіант може бути обмежуючим. Наприклад, робота вдвох дозволяє компаніям безкоштовно використовувати дві дошки, а розмір команди обмежений трьома.

Програмне забезпечення для управління двома проектами - це корпоративна версія, яка може вам знадобитися, якщо ваша організація більша, і ви одночасно обробляєте кілька проектів. На рисунку 2.4 показано два проекти з перспективою 2-плану високого класу. Це поставляється з серйозною кривою навчання. Розгляньте 2-план управління проектами на рівні підприємства.

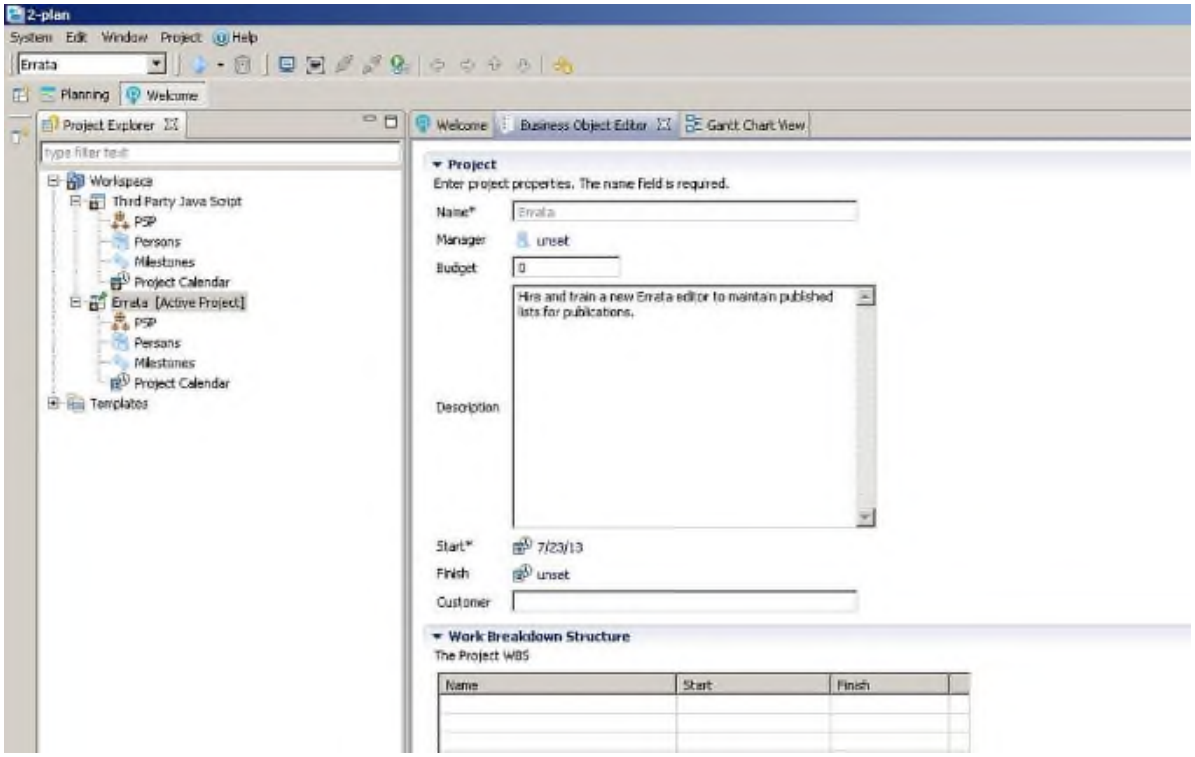
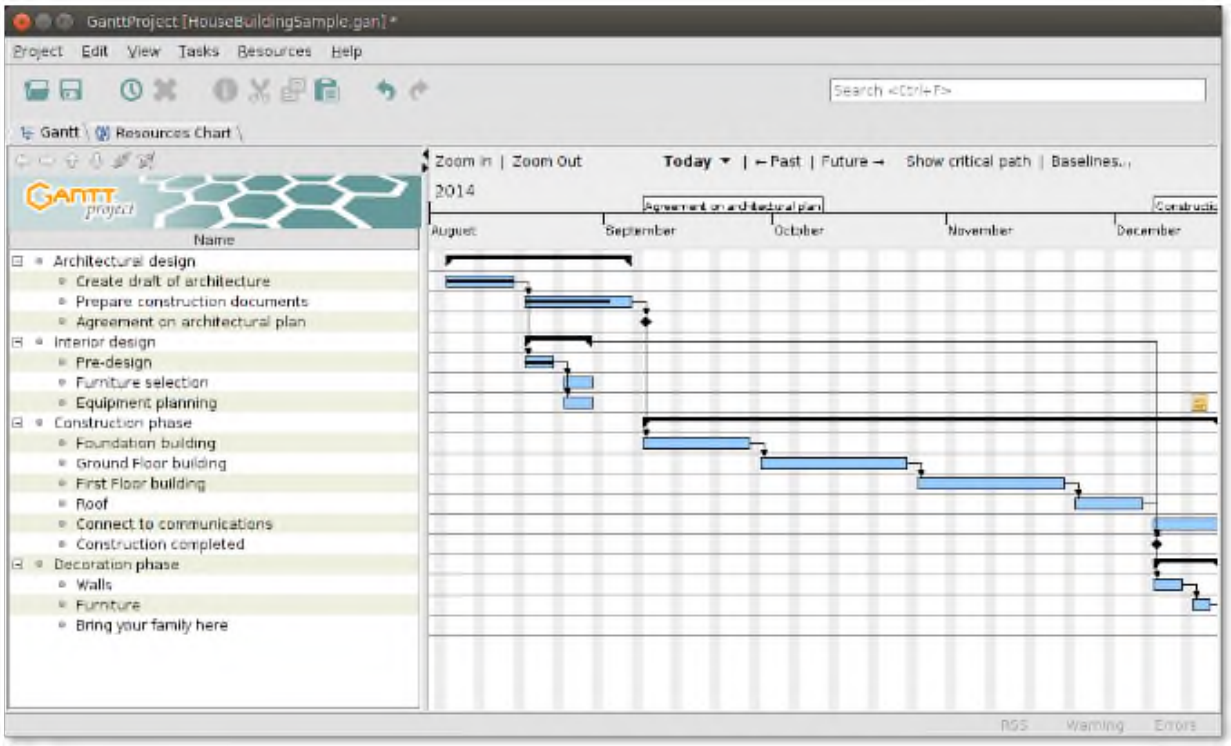


Рисунок 2. 4 2-план підтримує кілька проектів.

2.3.Проект Ганта

GanttProject - це ще один безкоштовний інструмент планування та управління проектами з відкритим кодом. Рецензенти порівняли цю надзвичайно

важку програму з Microsoft Project - як з точки зору пропонованих функцій, так і складності. Ця система може генерувати діаграми Ганта та PERT, створювати звіти у форматах HTML та PDF, а також пропонує різноманітні інструменти планування та управління часом. Він стверджує, що слава - це простота, і якщо ви вкажете під час встановлення, як показано на малюнку C, він відкриє файли MS Project. Він безкоштовний і підтримує версії для Windows, Mac OS X та Linux. Налаштувати вигляд Ганта, показаний на малюнку D, не так просто, як це могло б бути, але досить просто. Залежності легко створювати та відображати, подібно до MS Project та ProjectLibre. Варіанти звітування вражають, але він не підтримує функції оцінки вартості.



Малюнок 2.5 Головне вікно GanttProject

Плюси

Немає обмежень щодо того, що ви можете робити з GanttProject. Платформа управління дозволяє користувачам швидко створювати структурований графік для будь-якого проекту. Він пропонує призначення завдань та реалізацію етапу. Програмне забезпечення з відкритим кодом також дозволяє менеджерам проектів виявляти проблемні області в робочому процесі, щоб компанії могли встановлювати цілі щодо вдосконалення.

Мінуси

Я не рекомендував би GanttProject людям, які не знайомі з програмним забезпеченням для управління проектами. Багато хто вважає це надзвичайним - і підтримка здебільшого залишається на його форумах.

2.5.Бітрікс24

Bitrix24 - це система управління проектами, абсолютно безкоштовна для необмежених користувачів. Це обмежує обсяг пам'яті до 5 ГБ на місяць, з можливістю оновлення до 39 доларів на місяць. Особливості конкурують із функціями поточного програмного забезпечення PM: Basecamp [8].

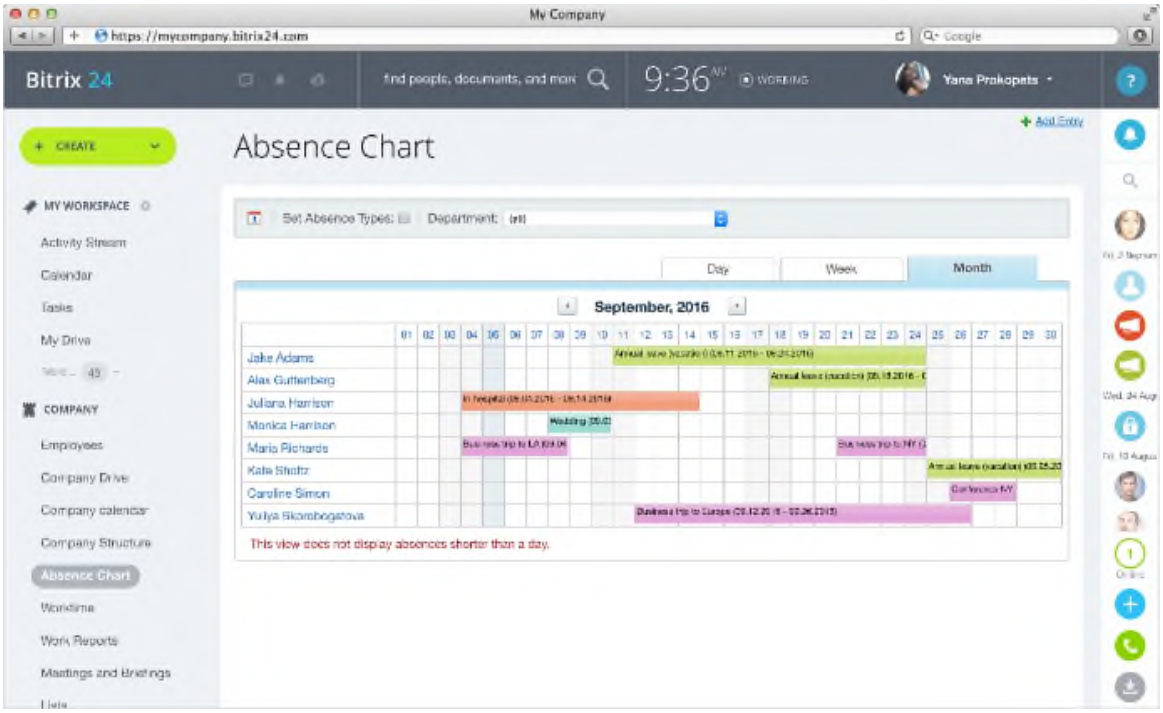


Рисунок 2.6 Головне вікно Bitrix24

Плюси

Користувачі можуть вибрати, використовувати Bitrix24 у хмарі або самостійно розміщувати на власному сервері компанії. Особливості PM - надзвичайні: Bitrix24 пропонує діаграми Ганта, багаторівневі варіанти завдань, відстеження часу та управління, а також навіть планування робочого навантаження співробітників.

Bitrix24 також полегшує спілкування в режимі реального часу за допомогою групового чату, відеоконференцій та обміну миттєвими повідомленнями. Він також діє як альтернатива DropBox - безкоштовна версія пропонує 5 ГБ хмарного сховища для зручного обміну документами.

Крім того, останні оновлення включають:

- Інструмент планування робочого навантаження працівника, який дозволяє менеджерам планувати певну кількість годин для виконання завдання, а потім порівнювати його з кількістю фактичних годин, проведених тими, кому було призначено завдання.
- Можливість створювати шаблони завдань, що містять підзадачі та контрольні списки.

Мінуси

Bitrix24 має на меті вирішити низку ділових проблем, і користувачі можуть легко переповнитись між його функціональними можливостями управління проектами та CRM-інструментами. З огляду на це, більшість скарг, як правило, походять від другої частини програмного забезпечення, ніж до першої.

Bitrix24 поєднує в собі соціальну інтрамережу, завдання управління проектами та CRM. Як якщо б ви включили Salesforce до списку завдань проекту, але хотіли чогось простішого. Великою частиною Bitrix24 є можливість зберігати всі розмови та оновлення компанії в одному місці, разом із вашими проектами. Вони мають власну та хмарну версії, але є обмеження щодо зберігання. Незважаючи на те, що є безкоштовний варіант, з великою командою ви захочете оновити не лише основи.

2.6.ProjectLibre

ProjectLibre - це програмне забезпечення для управління проектами, альтернатива Microsoft Project. Це також нова оновлена версія. Він отримав нагороду InfoWorld "Найкраще з відкритим кодом". ProjectLibre сумісний з файлами Microsoft Project 2003, 2007 та 2010 [9].

Це вилка від OpenProj, безкоштовного настільного додатку з відкритим кодом, призначеного для досить тісного наслідування старого Microsoft Project. Це робить розумну роботу, імітуючи макет та інтерфейс Project, тому, якщо ви знайомі з ним, ви зможете завантажити його та почати роботу з дуже невеликою кривою навчання; легко створити просту діаграму Ганта зі звичайним робочим процесом; створення відступної структури розподілу робіт (WBS), встановлення тривалості, створення посилань та призначення ресурсів. Він навіть відкриває

файли .mpp, хоча не може зберегти їх назад до того ж формату, замість цього зберігає як XML-файли, що дратує. Все це звучить добре, але коли ви намагаєтесь його використовувати, ви не можете не побажати, що ви просто використовували Microsoft Project; це просто дуже незграбний, потворний і прискіпливий у використанні. Так, це безкоштовно,

ProjectLibre (раніше відомий як OpenProj) - це програмне забезпечення для управління проектами з відкритим кодом, яке відкрито виставляє рахунки як заміну MS Project. (Нахабно!) Сумісний із MS Project 2003, 2007 та 2010, а також Linux, MacOS та Windows, тому охоплює майже всіх. Він також підтримує кілька мов.

На рисунку 2.7 показано його схожість з MS Project. Створення залежностей та ресурсів є простим та інтуїтивно зрозумілим (здебільшого). Незалежно від того, орієнтуєтесь ви на введення даних чи інтерфейс, ви знайдете проекти, прості в налаштуванні та обслуговуванні. Подання Ганта обчислює та відображає шлях до вашого проекту візуально за замовчуванням. Тривалість та залежності завдань працюють, як очікувалося, і здебільшого знайти варіанти, які ви найбільше використовуєте, буде легко.

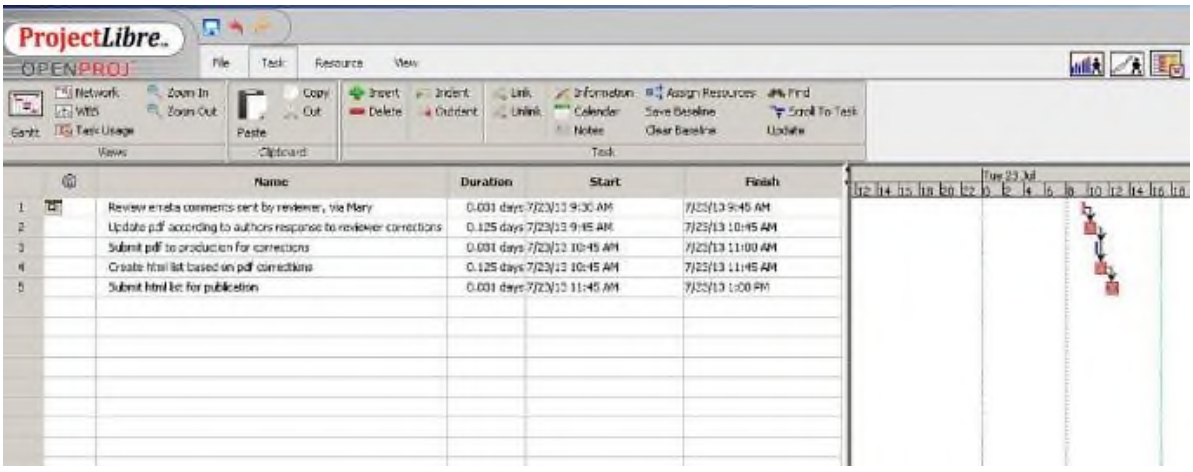


Рисунок 2.7 Введення завдань проекту у поданні WBS

Користувач може легко роздрукувати або переглянути це вікно, і ви можете надрукувати файл у форматі PDF, але ви, мабуть, будете більше покладатися на подання звітів, як показано на малюнку В. За замовчуванням розумні та легко налаштовуються. Результатом є низька кількість звітів, яку ви можете використовувати для документування свого проекту або обміну з іншими.

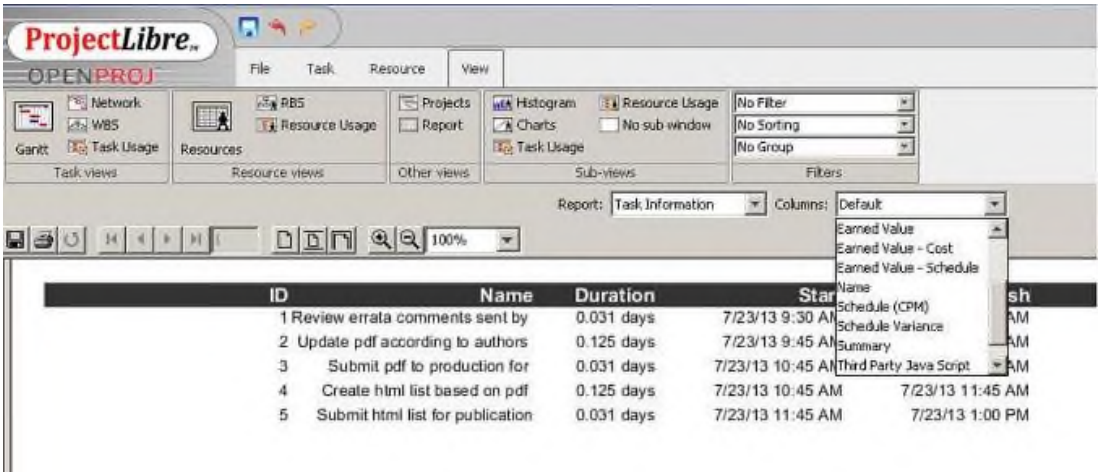


Рисунок 2.8 Прості звіти легко налаштувати.

Ця програма відповідає своєму ажіотажу. Єдиним негативним моментом є відсутність користувацької документації, але він має активну Інтернет-спільноту.

2.7.Відкрити робочий стіл

Open Workbench - це надійний, зрілий інструмент для планування та управління проектами. Він відповідає і підтримує основні ідеали управління проектами, одночасно подаючи інформацію інтуїтивно зрозуміло та легко для вивчення. Десятки тисяч менеджерів проектів у всьому світі використовують Open Workbench для планування та виконання складних проектів.

Усі проекти протягом свого життєвого циклу виконують низку завдань (або етапів). Використовуючи Open Workbench, ці критичні завдання або етапи стають більш керованими, що робить проекти більш вірогідними. Open Workbench дозволяє менеджерам проектів створювати структури розподілу робіт (WBS) із завданнями та етапами, встановлювати базові схеми, планувати плани проектів із залежностями, призначати ресурси завданням, планувати роботу над завданнями протягом певного періоду, коригувати графік, коли фактична робота реєструється , зв’язати головний і підпроекти, запланувати ресурси між ними та провести аналіз зароблено вартості [10].

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ КОНТЕНТОМ ДЛЯ СПРОЩЕННЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Описання системи керування контентом проєктів

Інструменти реалізації

Для реалізації прототипу ми вирішили використовувати .NET Framework 4.7.2 та C # завдяки попередньому досвіду та знайомству. C # дозволив нам швидко розробити суворий, корисний і надійний користувальницький інтерфейс для компонента Network Proxy рішення. Хоча це може бути повільніше, ніж мови низького рівня з ручним управлінням пам'яттю, як C ++, переваги наявності величезного досвіду роботи з ним переважають усі недоліки.

В якості IDE ми обрали спільноту Microsoft Visual Studio 2017. Це безкоштовна IDE, яка є його дуже модульною та спеціально розробленою з урахуванням програмування на C #.

Впровадження послуг

Для того, щоб протестувати створений прототип на практиці, ми створили низку невеликих сервісів, які певною мірою відтворювали SOA реального рішення. Це було досягнуто спочатку впровадженням базового постачальника послуг, який дозволяв створювати, читати (шукати), оновлювати та видаляти послуги зі списку пошуку.

Після цього ми створили кілька невеликих сервісів із наступними функціями та залежностями:

Сервіс 1 - періодично викликає іншу службу для обчислення середнього значення діапазону випадкових значень

Послуга 2 - виконує обчислення середнього значення діапазону значень. Використовує одну іншу службу для обчислення суми, а іншу для поділу.

Послуга 3 - обчислює суму діапазону значень.

Послуга 4 - обчислює фактор між двома значеннями.

Усі створені послуги попередньо зареєстровані в нашому службі пошуку. Ідея полягає в тому, що періодично служба 1 генерує серію випадкових цілих чисел і робить запит ро службі 2 для їх обчислення. Послуга 2 виконує це, синхронно викликаючи службу 3 для обчислення суми випадкових чисел, а потім подає запит до послуги 4 для обчислення коефіцієнта з ділення між сумою (з послуги 3) та підрахунком предметів.

Це дуже базовий приклад, але він дозволяє нам відтворити SOA з метою тестування прототипу.Рис. 3.1 Діаграма послідовності послуг, що обчислює середнє значення4.1 показана схема послідовності алгоритму послуг, описаного вище.

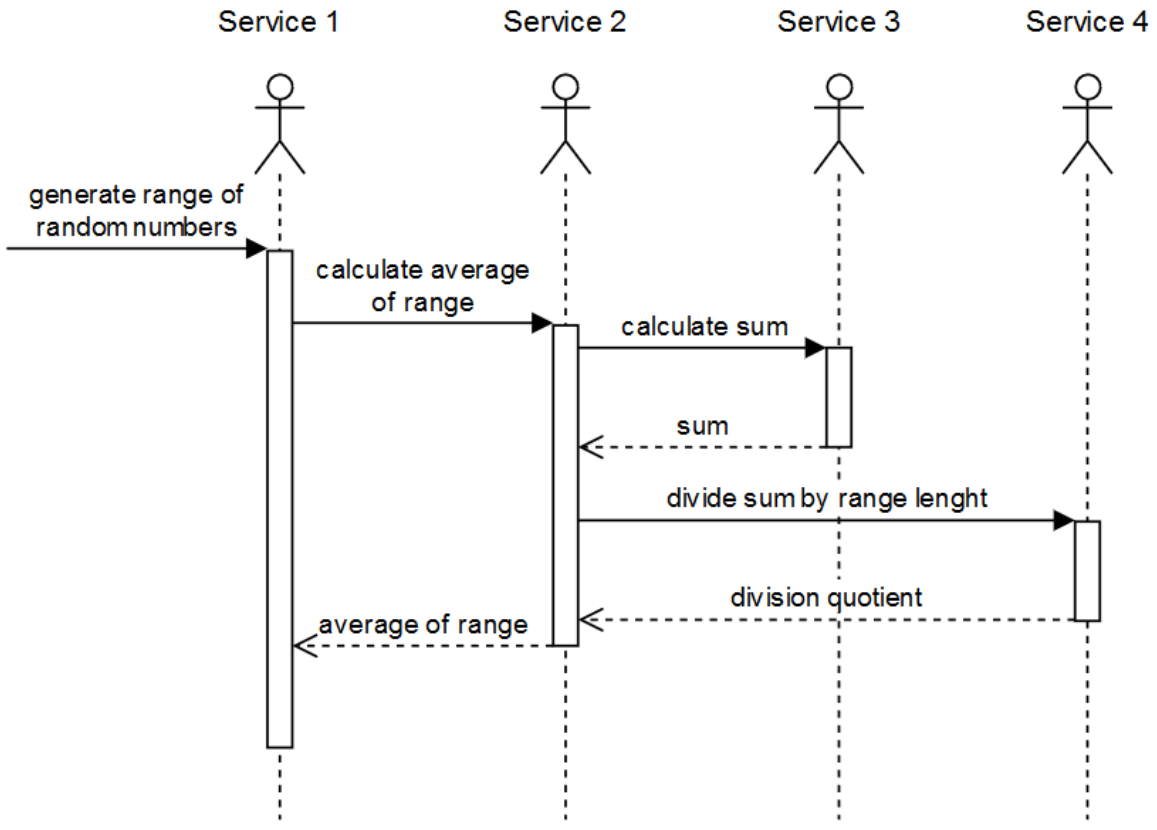


Рис. 3.1 Діаграма послідовності послуг, що обчислює середнє значення

Файли конфігурації

Прототип розроблений з урахуванням того випадку, коли він не знає сам по собі, який вид послуг присутній у системі або який потрібен, як і справжня мережа в Інтернеті.

На жаль, варіант використання нашої системи, який є аналізом трафіку та маніпуляціями, неможливий без вказівки хоча б деяких служб та їх взаємозв'язків. Як мінімум нам потрібно вказати точки входу служб у систему, щоб ми могли відфільтрувати їх від іншого перехопленого мережевого трафіку та визначити, кому він належить.

Разом із точкою входу нам також потрібно вказати назву служби для кожної точки входу. Це забезпечить деяку ясність, коли ми будемо візуалізувати змодельоване мережеве середовище та відображати, як поводяться запити на послуги в системі.

Було прийнято рішення використовувати XML для синтаксису файлів, оскільки він є читабельним і зрозумілим для людини, а також є витратним. Для прототипу ми створили два файли: servdef.xml та globalconf.xml

Файл визначення служб

Файл конфігурації служби з назвою servdef.xml використовується для вказівки точок входу служби в рішення для моделювання та їх назв.

В якості точки входу вказується порт tcp. Оскільки передбачається, що все рішення розміщується на одному комп'ютері, немає необхідності вказувати IP-адресу.

Ім'я служби є довільним рядковим значенням і може мати будь-яку довжину, а може і не вказуватися взагалі. У цьому випадку кожному сервісу присвоюється загальна назва «Сервіс N», де N - номер служби у колекції неназваних послуг, починаючи з 1.

Дерево xml із файлу servdef.xml можна попередньо переглянути Рис. 3.2 Конфігурація служби XML-структура дерева3.2:

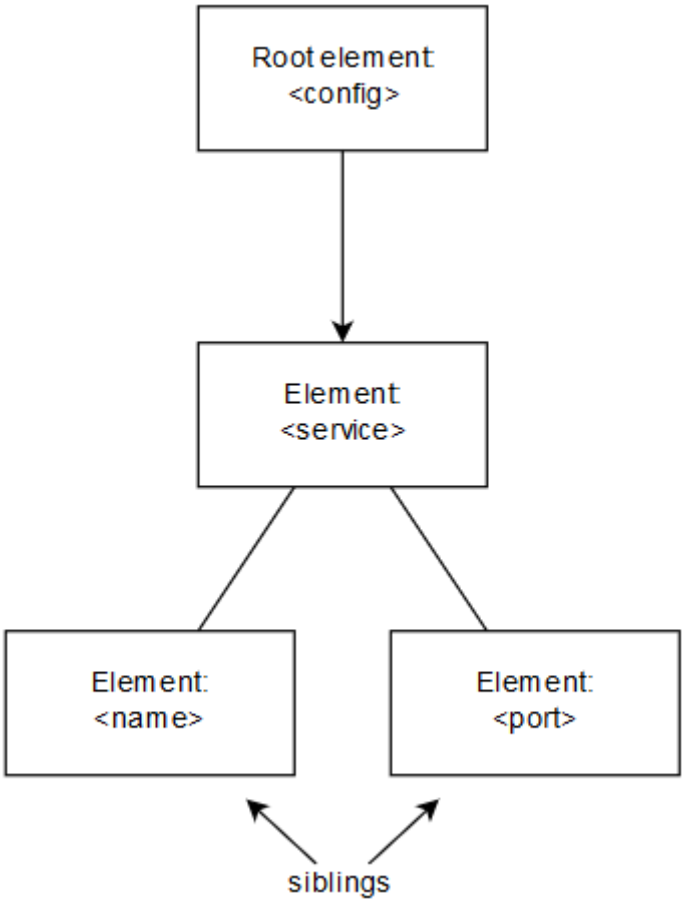


Рис. 3.2 Конфігурація служби XML-структура дерева

Файл конфігурації глобальних налаштувань

Файл конфігурації глобальних налаштувань містить усі параметри, що визначають поведінку модельованого середовища. Він побудований з трьох розділів: загальні налаштування, налаштування послуги та налаштування дзвінків.

Глобальні настройки - це параметри, які впливають на все середовище. В даний час вони мають лише один параметр - швидкість передачі пакетів, який імітує передачу даних через мережу. Налаштування послуг та налаштування дзвінків також дозволяють змінювати швидкість передачі даних, але вони застосовуються більш суворо, ніж загальні налаштування. Дерево конфігурації xml файлу можна переглянути нижчеРис. 3.3 Деревоподібна структура файлу глобальних налаштувань4.5:

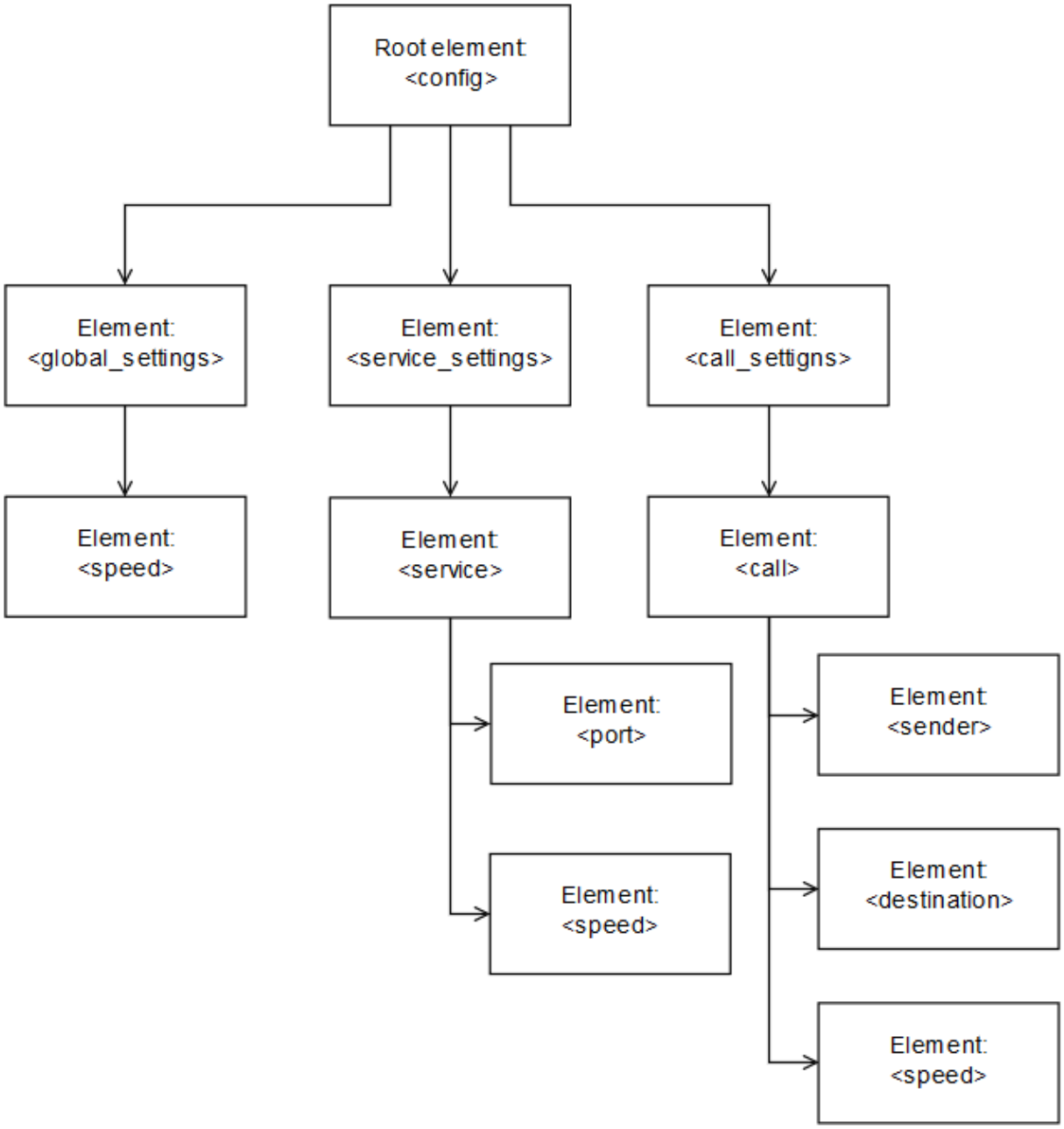


Рис. 3.3 Деревоподібна структура файлу глобальних налаштувань

Прототип графічного інтерфейсу

Для того, щоб використати цей прототип і забезпечити простий доступ до інформації, зібраної мережевим проксі, ми створили спеціальний графічний додаток для користувацького інтерфейсу.

Він використовує Windows Presentation Foundation (WPF) як основу свого графічного подання. Для кращого стилю використовується бібліотека управління MahApps WPF, яка надає широкий спектр компонентів та стилів.

Знімок екрана інтерфейсу створеної програми можна побачити нижче на Рис. 3.6 Головний екран прототипу3.6:

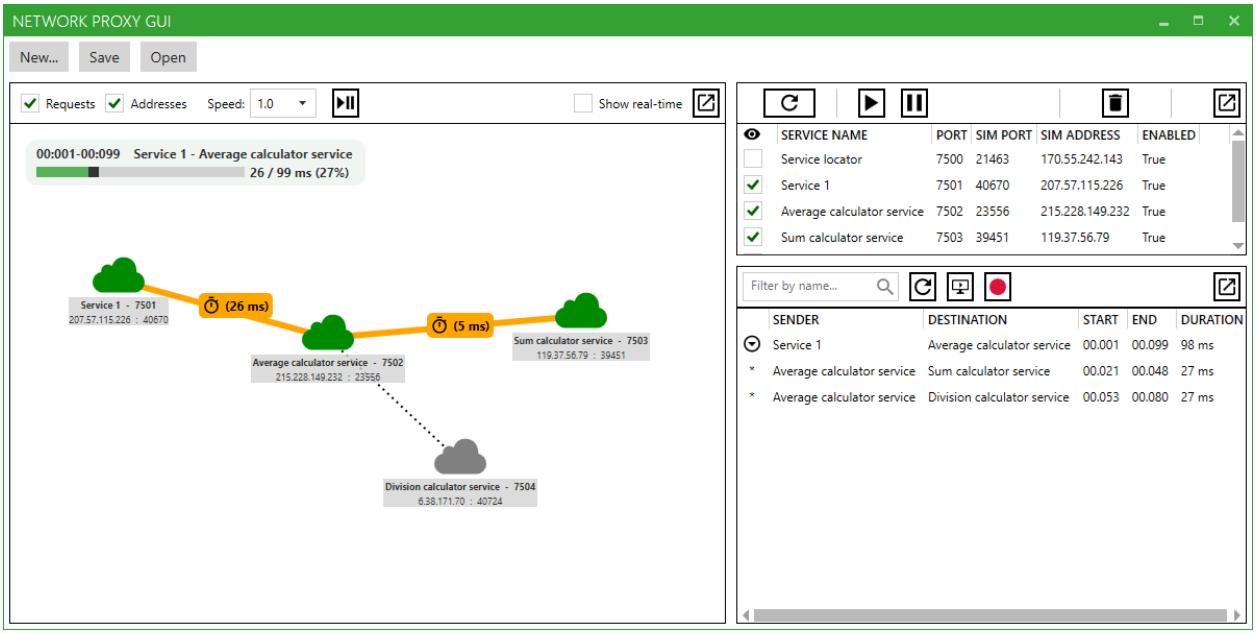


Рис. 3.6 Головний екран прототипу

Основний екран програми, показаний на Рис. 3.6 Головний екран прототипу, складається з трьох областей:

Сформована карта послуг та попередній перегляд зв'язку зліва

Список зареєстрованих служб та інформація про їхні симуляційні адреси вгорі праворуч

Історія запитів, оброблених мережевим проксі внизу праворуч

Три області складають основну частину ознак прототипу. Кожен з них детально описаний у наступних підрозділах.

Карта довкілля

Карта довкілля є головною особливістю прототипу і її можна побачити на Рис. 3.7 Імітаційна карта взаємовідносин із середовищем та послугами4.7. Він забезпечує візуалізацію всіх служб, які були зареєстровані у файлі конфігурації. Послуги, які взаємодіють між собою, динамічно з'єднуються лінією, коли таке спілкування відбувається вперше після запуску програми. Це так, оскільки, хоча

ми можемо вказати параметри мережі взаємодії служб, ми не знаємо, чи відбудеться ця взаємодія коли-небудь.

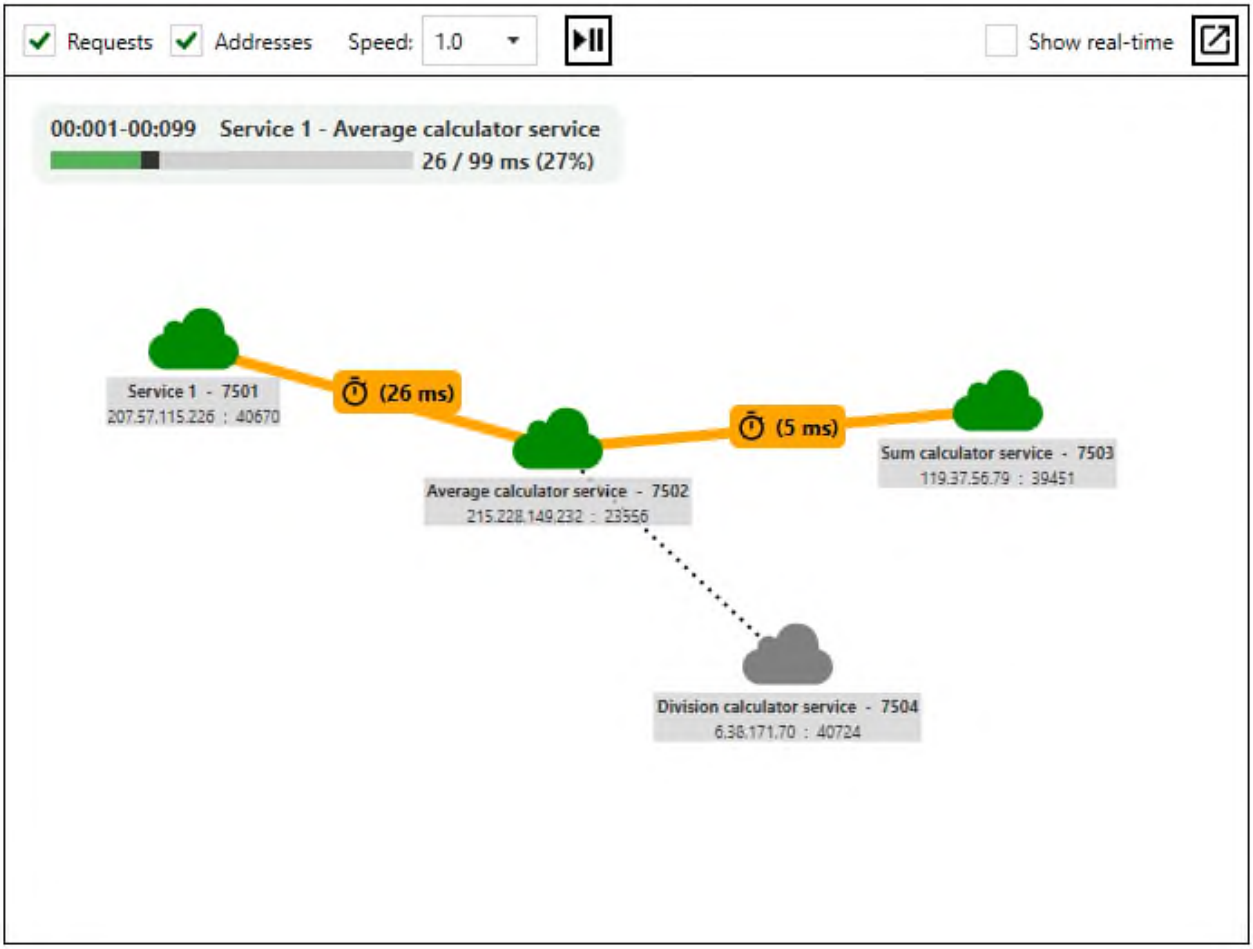


Рис. 3.7 Імітаційна карта взаємовідносин із середовищем та послугами

Кожна служба, представлена в інтерфейсі хмарою, відображає короткий опис відповідної інформації про неї, а саме її ім'я, мережевий порт, що використовується для взаємодії з мережевим проксі-сервером, та сформована комбінація ip: порт для внутрішнього відображення мережевого проксі-сервера, яка необхідна для імітації розподіленого спілкування.

Завдяки нашій здатності відстежувати прогрес спілкування на всіх етапах, ми можемо точно визначити, коли служби взаємодіють між собою та чи чекає хтось із них на відповідь. Це дозволяє виділити послуги різними кольорами, залежно від того, чи вони зараз перебувають у процесі обчислення. Це також дозволяє нам виділити зв’язок між службами (пунктирний чорний - в режимі

очікування, густий помаранчевий - надсилання запиту або очікування відповіді) і найголовніше, візуально відображати, скільки часу використовувалося посилання. Приклад, показаний наРис. 3.7 Імітаційна карта взаємовідносин із середовищем та послугами досить простий і складається лише з чотирьох служб плюс прихований постачальник послуг, але в сценарії, коли використовується кілька десятків служб, ця візуалізація може бути корисною для розуміння того, як різні служби взаємодіють між собою під час розрахунків.

Усі дані, використані для цього подання, побудовані на кількох ключових мітках часу, зроблених у вирішальні моменти. Ми вже говорили про них раніше в попередньому розділі. Порівняно низька кількість необхідних даних дозволила нам записати ці дані, зберегти їх у реальному часі, а потім забезпечити візуальне представлення, яке насправді можна переглядати із звичайною або зменшеною швидкістю.

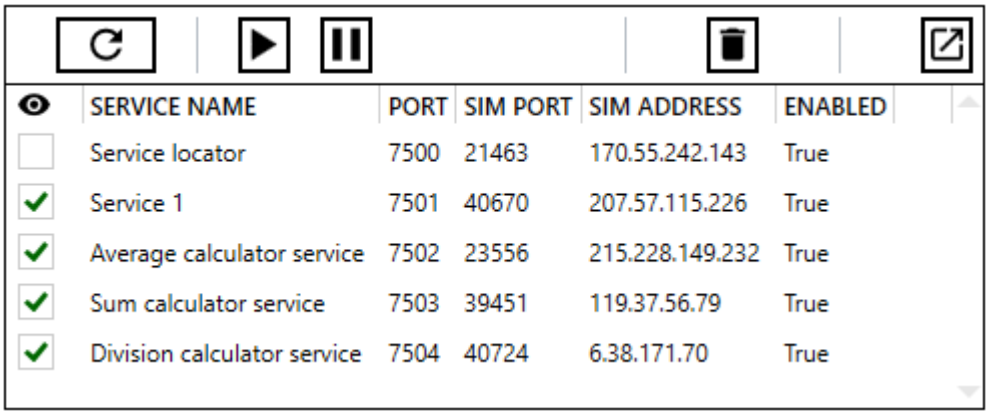
Хоча ми вирішили не реалізовувати симуляційне відтворення швидше, ніж його фактична швидкість, можливість відтворювати його зі зниженою швидкістю, у сто разів повільнішою, дозволяє краще зрозуміти, що насправді відбувається в системі, якщо операції виконуються швидко швидкість.

Список послуг

Список послуг показує служби, які були зареєстровані в програмі через файли конфігурації, та відображає їх реальну та створену для моделювання мережеву інформацію (Рис. 3.8 Перегляд списку послуг прототипів4.8). За допомогою цього меню ми можемо ввімкнути або вимкнути відображення послуги на карті середовища. Якщо це зробити, це буде просто виключено для візуалізації, що може допомогти зменшити надмірність деяких служб. Якщо проміжна служба, яка служить зв'язком між двома іншими, відключена, натомість лінія відображатиметься так, ніби ці дві служби безпосередньо взаємодіють між собою.

Цей вигляд також дозволяє зупиняти або зупиняти служби, імітуючи їх перезапуск, несправний збій або вимкнення. Якщо послугу тут вимкнено,

мережевий проксі буде діяти так, ніби насправді недоступний. Сама служба продовжуватиме працювати нормально, але всі спроби зв'язатися з нею або з неї будуть перервані та оброблятись так, ніби вони не змогли дістатися до пункту призначення.



SERVICE NAME	PORT	SIM PORT	SIM ADDRESS	ENABLED
Service locator	7500	21463	170.55.242.143	True
Service 1	7501	40670	207.57.115.226	True
Average calculator service	7502	23556	215.228.149.232	True
Sum calculator service	7503	39451	119.37.56.79	True
Division calculator service	7504	40724	6.38.171.70	True

Рис. 3.8 Перегляд списку послуг прототипів

Незважаючи на те, що цей перегляд забезпечує короткий список відповідної інформації, подвійне клацання на послузі відкриває нове вікно, яке називається оглядом послуги, з додатковою інформацією, що відображається в ньому (Рис. 3.9 Знімок екрана огляду послуг3.9).

Поряд з інформацією, яку ви можете бачити у списку послуг, у цьому вікні також відображається список вхідних або вихідних запитів до послуги. Ця інформація може відображатися у двох поданнях. Спочатку надається загальний перелік служб, які зробили запит на цю послугу, кількість дзвінків від кожної з них та середній час відповіді на запит. Другий вигляд схожий на перегляд історії запитів (який обговорюватиметься незабаром після цього), але замість цього він фільтрується виключно для запитів, де поточна послуга є відправником або призначенням.

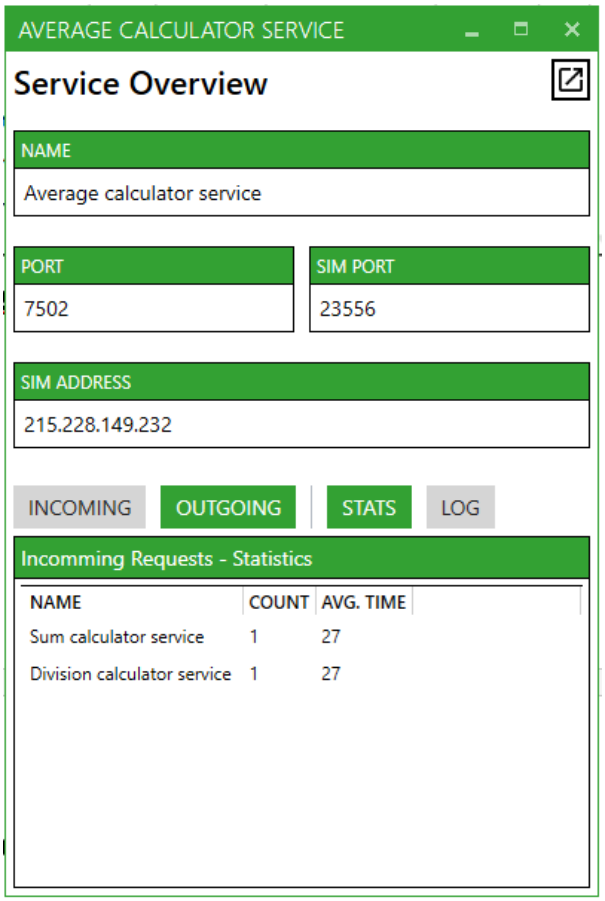


Рис. 3.9 Знімок екрана огляду послуг

Історія запитів

Історія запитів - це список запитів, які виконувались через мережевий проксі зареєстрованими службами. Цей список показує, хто був відправником запиту, хто був пунктом призначення цього запиту, скільки часу знадобилося для отримання відповіді відправника та, що найголовніше, якщо сталася ланцюжок послуг, на яких це вплинуло (Рис. 3.10 Скріншот історії запитів3.10).

Запити, подані між двома службами, відображаються в такому вигляді. Але якщо одна служба робить запит на другу службу, а друга служба робить запит на третю службу, перш ніж він надав відповідь на першу послугу, тоді ми вважаємо, що сталася ланцюжок послуг. Початковий запит, який ініціював ланцюговий запит, відображається стрілкою в списку історії запитів, а його “дочірні” запити відображаються із маленькою зірочкою.

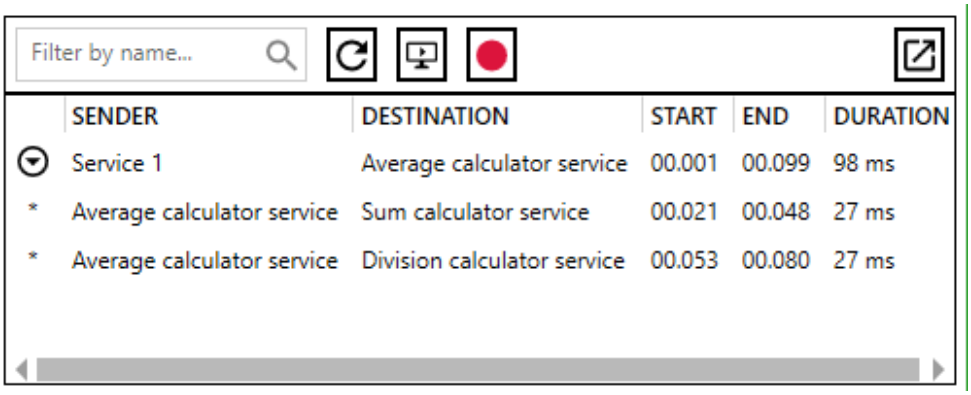


Рис. 3.10 Скріншот історії запитів

Цей вигляд також надає функцію тимчасового вимкнення запису будь-яких нових запитів та запуску відтворення конкретного запиту для його перегляду на карті середовища.

Двічі клацніть на будь-який запит зі списку, відкриється нове вікно з інформацією про вибраний запит (Рис. 3.11 Знімок екрана огляду запиту4.11). Він містить детальну інформацію про проміжок часу цього запиту, подібно до того, як це було зображено в попередньому розділі.

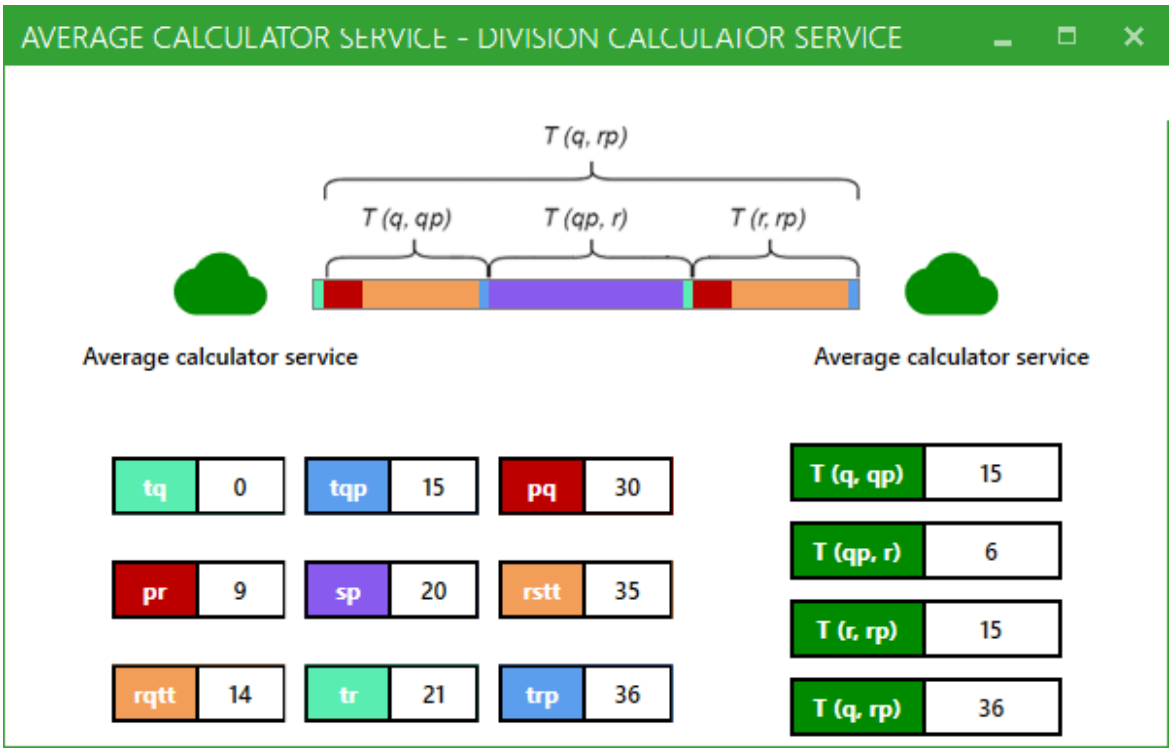


Рис. 3.11 Знімок екрана огляду запиту

Хронологія в середині екрана містить детальну розбивку того, що сталося під час запиту, і скільки часу знадобилося для того, щоб відбулися події. У нижній частині екрана та ж інформація відображається більш організовано.

3.3. Висновки до розділу

Завдяки створеному прототипу імітаційного засобу SOA ми вирішуємо проблему імітації транспортного шару SOA.

Створений прототип дозволяє нам моделювати, управляти, візуалізувати та аналізувати зв'язок між кількома службами в SOA, імітуючи розподілену мережу, при цьому всі необхідні компоненти запускаються на одній машині.

З усіх можливостей, які надає нам повний контроль транспортного рівня, ми застосували механізм затримки повідомлень, коли вони передаються між службами, і суворий механізм реєстрації та візуалізації того, як служби взаємодіють між собою.

Створений прототип можна вважати доказом концепції та частиною рішення, що дозволяє нам моделювати середовище спілкування між кількома службами, не вимагаючи встановлення їх на фізично різних машинах, зберігаючи при цьому основи архітектури SOA, такі як динамічне розташування служби та зчеплення.

ВИСНОВКИ

Дипломний проєкт містить базову інформацію щодо створення засобу управління проєктами програмного забезпечення. Було описано всі етапи розробки, що були пройдені, в яких вирішувались питання аналізу, планування, проєктування та реалізації.

Поставлена задача, яка полягала в розробці системи керування контентом для розробки програмного забезпечення, була вирішена, та реалізовано вирішення проблем, таких як:

– відсутність документування задач проєкту;

– відсутність обговорення різних тем, пов'язаних з проєктом;

– неможливість спостереження за ходом розвитку проєкту тобто немає графіків виконання проєкту, історії зміни вимог.

В результаті цього проекту ми досягли низки таких цілей:

Проведено аналіз області архітектури, орієнтованої на послуги, її основних концепцій та принципів проектування

Проведено аналіз основних переваг та недоліків SOA, зокрема вказано на складність розуміння основних концепцій, способів експлуатації та переваг впровадження

Проаналізовано та вказано на найпопулярніші технології реалізації SOA, що ще більше покращує наше розуміння теми

Розроблене рішення, яке дозволяє записувати та контролювати весь мережевий зв’язок між службами, при цьому всі служби розміщуються на одному комп’ютері або навіть на одній сесії користувача

Створено прототип рішення та успішно перевірено його основну ідею запису поточного часу на різних етапах спілкування, щоб забезпечити візуальне представлення всієї системи для користувача. У той же час створене рішення може бути певною мірою використано для профілювання продуктивності послуг як "чорні ящики"

Створений прототип може бути використаний у майбутньому для створення належного комерційного рішення, що забезпечує: освітню цінність, що дозволяє краще зрозуміти основні ідеї SOA з чітко доступними видимими відгуками; підвищена стабільність завдяки можливості виконувати інтеграційні тести між службами в імітованому середовищі, подібному до реального.

Подальша робота повинна бути спрямована на дослідження та розробку методів виявлення асинхронного спілкування зі службами або спілкування з великою кількістю одночасних запитів. На цей час прототип показує найкращі та правильні показники, якщо споживчий запит на послугу виконується в одному екземплярі та під контролем.

Ідея може також розвиватися в подальшому до інструментів налагодження для звичайної розробки, які будуть зосереджені на візуалізації потоку потоків у додатку між його компонентами. По суті, формула для розрахунку часу може бути використана таким же чином, тоді як техніка візуалізації залишається незмінною.