Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РЕАЛІЗАЦІЯ МЕТОДУ УПРАВЛІННЯ КОМАНДАМИ ДЛЯ ВИЯВЛЕННЯ ФАКТОРІВ ЕФЕКТИВНОСТІ

(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-2

Бобріщ І. Ю.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник зав. каф. Кобилін О.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Бобріщ Ірині Юріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Реалізація методу управління командами для виявлення факторів ефективності

затверджена наказом університету від 16 травня 2022 року № 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 30 травня 2022 р.

3. Вихідні дані до роботи Методології керування проектами, статистичні дані використання методології в компаніях.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналізування існуючих методологій.
2. Виявлення кращої методології серед обраних.
3. Порівняння результатів зі статистичними даними.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, мета роботи, постановка задачі, етапи виконання роботи, результати розрахунків, перспективи роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	18.04.2022	
2	Аналіз завдання, підбір літератури	18.04.22-21.04.22	
3	Аналіз літератури з досліджуваної проблеми	22.04.22-25.04.22	
4	Аналіз необхідних даних	26.04.22-30.04.22	
5	Розрахунки	01.05.22-14.05.22	
6	Аналіз результатів	15.05.22-23.05.22	
7	Оформлення пояснювальної записки	24.05.22-26.05.22	
8	Перевірка на плагіат	27.05.22	
9	Рецензування	28.05.22	
10	Підготовка презентації та доповіді	29.05.22-30.05.22	
11	Занесення роботи в електронний архів	29.05.22	
12	Попередній захист кваліфікаційної роботи	08.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____
(підпис)

Керівник роботи _____зав. каф. Кобилін О.А._
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 45 с., 25 рис., 31 джерело.

МЕТОД ВЕКТОРІВ, МЕТОД ПРІОРИТЕТІВ, МЕТОД САATI, МАТРИЦЯ СПІВВІДНОШЕНЬ.

Об’єктом роботи є різні методи керування командами.

Метою роботи є математичне обґрунтування та порівняння одинадцяти методології керування проєктами, які застосовуються у різних сферах повсякденно. Завдяки результатам можна підібрати методологію під кожен проєкт, задачу, команду тощо.

У роботі було використано методи ієрархічного аналізу, метод векторів пріоритетів, методи попарного порівняння.

У результаті роботи було сформовано порівняння методів між собою на основі математичних даних.

VECTOR METHOD, PRIORITY METHOD, SAATI METHOD, RATIO MATRIX.

The object of work are various methods of team management.

The aim is to analyze the results obtained and classify management methods.

The methods of hierarchical analysis, the method of priority vectors, methods of pairwise comparison were used in the work.

As a result of work the comparison of methods among themselves on the basis of mathematical data was formed.

ЗМІСТ

ВСТУП

Що таке методі керування проєктами? Це набір інструментів та підходів для керування командою, для того, щоб досягти певних результатів. Зараз існує безліч різноманітних методів, які спрямовані на певні цілі: хороша комунікація команди, скорочення терміну розробки продукту, поліпшення якості продукції тощо. Але все це має одну загальну ціль – виконати замовлення в термін згідно технічного завдання [1].

Можна сказати, що кожен метод можна використовувати для будь якого проєкту незалежно від програмного забезпечення, оточення чи суті проєкту.

Розпочинаючи певний проєкт кожен керівник розуміє, що правильно обраний метод керівництва може значно спростити процес розробки продукту, пом’якшити негативні наслідки та розумно використати потенціал співробітників.

Після збору вимог, обговорення бюджету та оцінки проєкту фірма та замовник повинні заключити контракт у якому будуть зазначені терміни виконання роботи, форми взаємодії між замовником та виконавцями, бюджет та інші суттєві пункти. Але головною проблемою стає вибір методу, за яким команда або команди будуть працювати впродовж усього проєкту [2].

Іноді доволі важко визначити метод, який на 100% буде відповідати потребам команди та продукту. У такому випадку керівники намагаються визначити пріоритети, та на основі їх вибрати більш задовільний спосіб. Також, керівник повинен мати чітке уявлення до того, які люди будуть працювати над проєктом, їх власні характеристики та потенціал. Не слід забувати і по перешкоди, які можуть суттєво вплинути на проєкт. Це також важливо враховувати.

Для того, щоб правильно обрати потрібну методологію, яка підійде до конкретного проєкту, потрібно розуміти позитивні та негативні аспекти кожного підходу. Розуміти, які критерії мають більшу вагу ніж інші. Саме

тому була поставлена мета зібрати, проаналізувати та порівняти основні підходи до розробки програмного забезпечення за певними критеріями.

Слід звернути увагу на те, що кожен підхід має свій набір переваг, свою систему та структуру. Тому порівнювати системі доволі складно. Але всі системи можна зв’язати деякими важливими аспектами, такими як:

– гнучкість до змін. Чим вище це значення, тим краще. У сучасному світі потрібно швидко випускати нові продукти спираючись на зміни на ринку та базуючись на потреби користувача;

– широкий спектр можливих проєктів. Методології, які можна використовували лише для маленьких команд, чи тільки для довгострокових проєктів – потрібні, але в рамках однієї фірми логічніше використовувати одну-дві методології, які покривають всі можливі проєкти, та адаптуються під них. Це значно скорочує час адаптування людей, які переходять між проєктами;

– добре працюють на команді, із будь якої кількості людей. Кожен проєкт потребує різної кількості людей і тому це впливає на час, бюджет, розробку, ресурси. Добре, коли методологія враховує різноманітні критерії для врахування всіх змінних [3].

Коли кожна методологія оцінена, порівняна та має математичне значення, тому простіше вибрати. Але ніколи не потрібно забувати, що основним критерієм вибору будуть люди, які будуть працювати у команді, бізнес-потреби проєкту, час та бюджет. Усі ці критерії вкупі допоможуть побудувати систему порівняти методології для визначення ваги, пріоритетності та актуальності використання методології в сучасному світі.

Для порівняння та подальшої оцінки будемо використовувати метод вектору пріоритетів у програмі MS Excel.

Актуальність роботи полягає у тому, щоб оцінити методології з математичного боку. Це дасть більш детальну та повну картину стосовно методології, та способів їх використання.

Слід зазначити, що проектні менеджери не розраховують щось подібне для того, щоб обрати методології для своєї команди. Вони спираються на досвід, команду, тип проекту, тощо. Тому ця робота буде порівнювати результати математичних розрахунків, та реальну ситуацію в фірмах.

1 ОГЛЯД МЕТОДИК

Перш за все слід зазначити, що існує багато способів керувати командою. Не будемо брати до уваги гібриди методів, та будемо аналізувати кожен метод окремо. Будемо розглядати як позитивні так і негативні аспекти.

Почнемо з базової моделі – waterfall.

1.1 Методологія Waterfall

Waterfall, чи каскадна модель – старий спосіб управління проєктами. Цю модель було запропоновано у 1970 роках з ціллю покращення роботи складних процесів. Суть моделі полягає у тому, що кожна ітерація процесу починається тільки після завершення попереднього кроку [4].

Позитивні аспекти:

– дуже проста система. Система не потребує довгого обговорення, дуже легко сприймається;

– не потрібно витрачати багато часу для того, щоб ввести нову людину до процесу;

– на початковому етапі можна прогнозувати результати;

– наявність чіткої та прозорої документації на початку проєкту.

Негативні аспекти:

– не гнучкий алгоритм;

– дуже важко реалізувати суттєві зміни;

– низька стійкість до перешкод;

– на початковому етапі потрібно чітко розуміти кінцевий продукт до деталей;

– доволі обмежений спектр проєктів (цю методологію використовують для короткострокових, чітких і простих проєктів або довгострокових важких проєктів у яких не буде змінюватися технічне завдання).

Слід зазначити, що каскадну модель найчастіше використовують за межами IT-сфери. Наприклад у будівництві, промисловість чи у роботі з контрактами [5].

1.2 Метод критичного шляху

Метод критичного шляху – метод, який дозволяє виявити критичні та некритичні задачі проєкту та контролює терміни їх виконання. Вперше цей метод з'явився у 1950-х роках та використовувався для сільського господарства. Метод було створено для того, щоб знизити витрати на зупинки та перезапуск певних етапів проєкту [6].

Позитивні аспекти:

– висока оптимізація ресурсів. Завдяки тому, що команда бачить які є обмеження за часом і який є резерв, то приймає рішення, які позитивно сприяють на процес;

– мінімізація ризиків. Через те, що у процес вже закладено певний резерв часу команда має змогу виправити недоліки або відреагувати на перешкоди не витрачаючи додаткові часи;

– безпечна розробка. На першому етапі одразу помітно весь план розробки проєкту, час, за який потрібно виконати задачі;

– проєкт виглядає структурованим та зрозумілим [7].

Негативні аспекти:

– на створення детальної схеми витрачається багато часу;

– проєкт складно адаптувати під зміни. На кожному кроці потрібно будет перемалювати схему, та витрачати на це час;

– оцінка кожної задачі є суб'єктивною та через це можуть виникати або вільні часи або їх нестача.

Цей метод найчастіше використовують Oracle, SAP [8].

1.3 Методологія DSDM

Метод розробки динамічних систем – система, яка базується на 8 принципах: люди, практика, ресурси, продукт, принципи, філософія та прагматизм. Ця система має суттєву особливість: у традиційних системах функціонал має фіксоване значення, а ціна, якість та час можуть змінюватись. У DSDM системі фіксованими залишаюсь час, якість та ціна і може змінюватись лише наповнення проєкту. Цей підхід було розроблено в 1994 році для швидкого впровадження у стрімкий процес розробки програмного забезпечення [9].

Позитивні аспекти:

– динамічна розробка. Будь які зміни у проєкті – обернені;

– швидка розробка. Краще розробили хороше зараз, чим ідеальне в кінці;

– команда сама приймає рішення стосовно проєкту. Розробка: реалізація: підхід;

– високі вимоги. Завдяки цьому проєкт розробляють мотивовані професіонали і наприкінці отримуємо продукт, відповідний технічному завданню;

– тестування інтегровано в процес розробки, що значно скорочує витрати на повторне тестування.

Негативні аспекти:

– проблема з конфіденційністю. Забезпечення секретності потребує додаткового тестування, яке, у свою чергу, потребує додаткового часу. Це конфліктує з ідеологією методу (фіксований час, ціна та якість);

– метод потребує постійної комунікації із замовником. Зазвичай це складно організувати, тому виникають затримки, які також впливають на час розробки;

– дуже велика кількість додаткових людей для повної реалізації цієї методології;

– неможливість розробки багаторазового продукту. Для цих проєктів, зазвичай, високі стандарти, а це впливає на час та принцип 20/80.

1.4 Методологія FDD

FDD (Feature driven development – розробка, орієнтована на особливості) – доволі популярна методологія, яку згідно до досліджень [10] використовують понад 11% компаній. Методологія складається з декількох етапів:

– розробка загальної моделі. Команда розробників ділиться на групи, які створюють моделі для окремих завдань. Потім обирається одна із запропонованих моделей або їх поєднання;

– створення списку функцій. Коли команда розробила загальну модель, вона визначає корисні для клієнта функції;

– планування. Тут важливо враховувати навантаження на групу, ризики та інші аспекти, щоб запобігти виникненню критичних проблем;

– дизайн та розробка. На основі даних першого процесу менеджер проєкту обирає групу функцій, які команда повинна реалізувати за певний термін;

– реалізація. Після того як команда розробила та протестувала код та модулі, вона приступає до створення ПЗ. На цей та попередній етап йде 75% зусиль команди розробників.

Позитивні аспекти:

– гнучка оцінка, у якому темпі працює проєкт;

– постійні звіти о роботі на всіх рівнях та етапах дозволяють краще розуміти проєкт;

– проєкт постійно оновлюється;

– краще та швидко знаходяться дефекти та баги.

Негативні аспекти:

– успіх проєкту цілком залежить лише від однієї людини – лідера. Він і розробник, і керівник, і лідер, і аналітик;

– методологія розрахована на довгострокові проєкти і не працює для стартапів.

1.5 Методологія Scrum

Scrum – це метод розробки, який будується навколо ідеї постійного вдосконалення продукту. Найбільш поширена та найпопулярніша методологія в світі.

Цей метод розробки цікавий використанням спринтів – лімітованих відрізків часу, за які команда повинна виконати певний стек задач [11].

Позитивні аспекти:

– дуже гнучка методологія, в якій можна знайти будь яке рішення;

– легка методологія, яка дуже проста;

– однаково добре працює для великих та маленьких команд.

Негативні аспекти:

– дуже важко перебудувати команду на цю методологію;

– на перших етапах складна та незрозуміла;

– майже не комбінується з іншими методами керування;

– для ефективного використання та очікуваних результатів команда повинна дотримуватися усіх правил Scrum розробки [12].

1.6 Методологія Lean

Lean або ощадлива розробка – виробництво, яке передбачає скорочення всіх можливих втрат, економічний та розумний підхід до виробничих ресурсів. Мова не тільки про втрати часу або розтрату фінансів і праці, а й нереалізований творчий потенціал співробітників. Вперше цю систему використала компанія Toyota, а потім її впровадили й багато інших виробництв [13].

Ця розробка має сім суттєвих принципів:

– ліквідувати втрати;

– вбудовувати якість;

– створювати знання;

– відкладати незворотні рішення;

– доставляти швидко;

– поважати людей;

– оптимізувати ціле.

Позитивні аспекти:

– оптимізований підхід дозволяє реалізувати більше функцій за менший час;

– усуває непотрібну діяльність і, як наслідок, знижує витратну частину проєкту;

– надає команді проєкту приймати рішення, що мотивує їх.

Негативні аспекти:

– залежить від задіяної команди та її професіоналізму, що робить його не таким масштабованим, як інші фреймворки;

– результат залежить від якості проєктної документації, помилки якої можуть призвести до помилок у розробки;

– при сплеску попиту може не вистачити ресурсів обслуговування та задоволення всіх незапланованих запитів, і тоді термін виконання збільшується [14].

1.7 Методологія XP

Екстремальне програмування або XP, eXtreme Programming – гнучка методологія розробки програмного забезпечення. Як і в інших agile-методології, вона має особливі інструменти, процеси та ролі.

Мета методики XP – впоратися з вимогами до програмного продукту, що постійно змінюються, і підвищити якість розробки. Тому XP добре підходить для складних та невизначених проєктів [15].

Позитивні аспекти:

– замовник отримує саме той продукт, який йому потрібен, навіть якщо на початку розробки сам не представляє його кінцевий вигляд;

– команда швидко вносить зміни до коду та додає нову функціональність за рахунок простого дизайну коду, частого планування та релізів;

– код завжди працює за рахунок постійного тестування та безперервної інтеграції;

– команда легко підтримує код, так як він написаний за єдиним стандартом і постійно рефакторіться;

– швидкий темп розробки за рахунок парного програмування, відсутності переробок, присутності замовника у команді;

– знижуються ризики, пов’язані з розробкою, т.к. відповідальність за проєкт розподіляється рівномірно та догляд/прихід члена команди не зруйнує процес;

– витрати на розробку нижче, так як команда орієнтована на код, а не на документацію та збори.

Негативні аспекти:

– успіх проєкту залежить від залучення замовника, якого не так просто досягти;

– важко передбачити витрати часу на проєкт, т.к. на початку ніхто не знає повного списку вимог;

– успіх XP сильно залежить від рівня програмістів, методологія працює тільки з senior спеціалістами;

– менеджмент негативно ставиться до парного програмування, не розуміючи, чому він має оплачувати двох програмістів замість одного;

– регулярні зустрічі з програмістами дорого коштують замовникам;

– вимагає занадто сильних культурних змін, щоб не контролювати кожне завдання;

– через брак структури та документації не підходить для великих проєктів;

– гнучкі методології функціонально-орієнтовані, нефункціональні вимоги до якості продукту складно описати у вигляді історій користувача.

1.8 Методологія sixSigma

sixSigma – це методологія управління проєктами, орієнтована на виключення дефекту виробництва як клас. Шість сигм пропонує два підходи залежно від того, чи ви покращуєте процес – DMAIC, чи створюєте новий продукт – DMADV, він же DFSS. Обидва методи можна описати формулою Плануй-Роби-Перевіряй-Виправ [16].

Позитивні аспекти:

– методологія дозволяє зробити всі процеси передбачуваними;

– бізнес-процеси можна описати, виміряти, аналізувати, покращити та контролювати;

– встановлюйте конкретні цілі, які можна підрахувати або виміряти: зменшити витрати на 20%, підняти прибутковість на третину, зменшити виробничий цикл на годину;

– рішення приймаються на основі підтвердженої інформації та статистики, ніж керуючись припущеннями.

Негативні аспекти:

– щоб досягти успіху за допомогою Шести сигм, залучайте весь персонал організації, особливо топ-менеджмент;

– ця методологія не дозволяє розробляти проривні та інноваційні продукти;

– методологія жорстоко обмежує творчість та будь яке відхилення від плану [17].

1.9 Методологія Kanban

Kanban – це популярний підхід до реалізації принципів agile та DevOps при розробці ПЗ. Методика передбачає обговорення продуктивності у режимі реального часу та повну прозорість робочих процесів. Робочі завдання візуально представлені на дошці Kanban, що дозволяє учасникам команди бачити стан кожного завдання у час [18].

Позитивні аспекти:

– гнучкість процесу. На відміну від скраму ця методологія дозволяє зміну планів одразу;

– безперервне постачання продукту замовнику;

– наочність методології допомагає команді швидко виявляти проблемні місця в команді чи розробці;

– скорочення часу циклу. метою методології є скорочення часу розробки і наочним показником є час циклу.

Негативні аспекти:

– не підходить для довгострокового планування. Метод канбан розрахований для досягнення короткострокових цілей. Робота вибудовується вирішенні актуальних завдань, у своїй їх пріоритетність може змінюватися залежно обставин.

– чи не підходить для великих команд. Чим більше людей задіяно в робочому процесі, тим складніше контролювати виконання завдань. Тому найкраще, щоб в одній команді було не більше десяти людей, в ідеалі п’ять [19].

1.10 Методологія RUP

Раціональний уніфікований процес (Rational Unified Process, RUP) – одна із спіральних методології розробки програмного забезпечення. Ітераційна розробка програмного забезпечення в RUP передбачає поділ проєкту на кілька дрібних проєктів, що виконуються послідовно, і кожна ітерація розробки чітко визначена набором цілей, які мають бути досягнуті наприкінці ітерації. Кінцева ітерація передбачає, що набір цілей ітерації повинен точно співпадати з набором цілей, визначених замовником проєкту, тобто всі вимоги повинні бути виконані.

Позитивні аспекти:

– методологія RUP дозволяє справлятися зі змінами вимог, незалежно від того, виходять вони від клієнта або виникають в ході роботи над проєктом;

– RUP наголошує на необхідності точної документації;

– інтеграція вимог відбувається протягом усього процесу розробки, зокрема у фазі побудови.

Негативні аспекти:

– RUP спирається на здатність експертів та професіоналів призначити події певним працівникам, які потім зобов’язані видати заплановані результати у вигляді артефактів;

– інтеграція у процес розробки може негативно позначитися на інший більш фундаментальної діяльності на етапах тестування [20].

1.11 Методологія Crystal

Crystal Clear – це легка гнучка методологія, створена Алістером Коуберном (Cockburn, 2004). Вона призначена для невеликих команд 6-8 осіб для розробки некритичних бізнес-застосунків. Як і всі гнучкі методології створення програм Crystal Clear більше спирається на людей, ніж на процеси та артефакти. Crystal Clear використовує сім методів/практик, три з яких є обов'язковими:

- часте постачання продукту;
- поліпшення через рефлексію;
- особисті комунікації;
- почуття безпеки;
- фокусування;
- простий доступ до експертів;
- якісне технічне оточення.

Позитивні аспекти:

- фокусування на проєкті та меті;
- особиста безпека;
- легкий доступ до досвідчених користувачів, що значно спрощує отримання зворотного зв'язку;
- автоматизовані тести які значно покращують якість продукту та прискорюють розробку.

Негативні аспекти:

- якщо у клієнта немає чіткого розуміння, яким він хоче бачити кінцевий продукт, а ясність настає тільки в процесі розробки (таке відбувається в переважній більшості випадків), то процес перетікає в стандартну бюрократію – доробки продукту можуть тривати нескінченно, доки не вичерпаються фінанси або увага замовника не переключиться на іншу продукцію;

– короткострокове планування, через це випадає цілий фрагмент формування плану розвитку продукту [21].

1.12 Постановка задачі

У сучасному світі безперестану йдуть суперечки – яка методологія керування командами є кращою? Частина людей вважає, що класичні методології як Спіральна, Ієрархічна, каскадна – є кращими. Інші вважають, що гнучкі методології підходять до проєктів краще, мають певні переваги. Кожна із сторін призводить велику кількість різних аргументів, але не обґрунтовує, чому саме певна методологія краща.

Об’єктом роботи є різні методи керування командами.

Метою роботи є математичне обґрунтування та порівняння одинадцяти методології керування проєктами, які застосовуються у різних сферах повсякденно. Завдяки результатам можна підібрати методологію під кожен проєкт, задачу, команду тощо.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів керування;

– проаналізувати критерії порівняння за якими будемо аналізувати методології;

– порівняти критерії між собою;

– порівняти методи керування за критеріями між собою;

– використати метод перемноження векторів;

– проаналізувати отримані результати;

– порівняти отримані результати зі статистикою.

2 АНАЛІЗ КРИТЕРІЇВ ОЦІНКИ МЕТОДОЛОГІЇ

2.1 Попарні порівняння та метод вектору пріоритетів

Для оцінки та порівняння методології будемо використовувати вектор пріоритетів. Але щоб застосовувати цей метод потрібно провести попарне порівняння кожної методології з кожною за окремими критеріями. Також, потрібно визначити, який із критеріїв має більшу «вагу» ніж інші [22].

Матриця парних порівнянь є матрицею, в якій критерій, розташований у рядку, порівнюється з усіма критеріями, зазначеними в стовпцях матриці. Наприклад, якщо критерій №1 важливіший за критерій №2 в 5 разів, то елемент (1, 2) матриці дорівнює 5. Виходячи з цього, головна діагональ матриці завжди заповнена одиницями [23, 24].

Логічно припустити, що якщо критерій №1 важливіший за критерій №2 в 5 разів, а критерій №2 важливіший за критерій №3 в 3 рази, то критерій №1 повинен бути важливішим за критерій №3 рівно в 15 разів. Однак для матриць, що заповнюються реальними людьми, це далеко не завжди так. Це пов’язано з тим, що наповнення матриці суджень здійснюється експертом, який може допустити похибку у визначенні відносної важливості критеріїв з психологічних причин. Одним із завдань методу ієрархій Т. Сааті є прагнення знизити вплив людського фактору на підсумковий смисловий результат. Для визначення ступеня коректності даних у заповненій матриці запроваджено поняття міри узгодженості матриці. В роботі використовуються виключно таблиці, сформовані за методом аналізу ієрархій Т. Сааті щоб зменшити суб’єктивний вплив експерта на результати дослідження [25, 26].

2.2 Критерії оцінки методології

Щоб об’єктивно оцінювати кожну методологію потрібно визначити, за якими саме критеріями и будемо порівнювати методології. Слід брати до уваги, що деякі параметри будуть відсутніми у деяких методів керування.

По-перше згадаємо, що таке метод переваг. Метод переваг – це метод, у якому експерту дають вибір альтернатив і він проводить ранжування цих альтернатив на основі порівняння критеріїв між собою.

По-друге пропонуємо використовувати такі критерії для оцінювання якості методології:

- гнучкість;
- можливість масштабування;
- відмовостійкість;
- складність;
- довжина ітерації;
- довжина розробки [27].

Далі розглянемо кожний із критеріїв.

2.2.1 Гнучкість

Що таке гнучкість для методології та проєкту? Це, по-перше, можливість команди виконувати задачі в умовах невідомості. Найчастіше це відбувається, коли замовник не розуміє до кінця кінцевий продукт та може змінювати технічне завдання впродовж розробки.

По-перше введемо умовні позначки для кожної методології:

- Waterfall – y1;
- Метод критичного шляху – y2;
- DSDM – y3;
- FDD – y4;

– Scrum – y5;
– Lean – y6;
– XP – y7;
– sixSigma – y8;
– Kanban – y9;
– RUP – y10;
– Crystal – y11.

Побудуємо основу для нашої матриці та заповнимо головну діагональ одиницями (рис. 2.1).

	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11
y1	1										
y2		1									
y3			1								
y4				1							
y5					1						
y6						1					
y7							1				
y8								1			
y9									1		
y10										1	
y11											1

Рисунок 2.1 – Основа для матриці попарних порівнянь

Наступним кроком є попарне порівняння методології між собою за критеріями гнучкості. Але слід зазначити, що це абстрактне поняття, тому замість шкали від 1 до 11 будемо використовувати шкалу від 1 до 5 щоб об’єднувати в умовні групи методології с близькими характеристиками.

Отже, методології, які мають найнижчу ступить гнучкості – це Waterfall та Метод критичного шляху. Це через те, що ці моделі не передбачають зміни

у проєкті, і будь які правки призупиняють процес розробки та вимушують команду повністю або частково переробляти проєкт.

Найбільш гнучкими є Scrum, Kanban та XP. Методологія екстремального програмування гнучка через те, що на кожному етапі розробки проєкт в робочому стані. Постійне тестування та прозорий код допомагають дуже швидко переробляти продукт.

Інші методології більш менш схожі між собою по гнучкості, з невеликими відхиленнями, тому ранжування методології має такий вигляд (рис. 2.2).

5	Waterfall	y1
5	Метод критичного шляху	y2
2	DSDM	y3
3	FDD	y4
1	Scrum	y5
2	Lean	y6
1	XP	y7
4	six Sigma	y8
1	Kanban	y9
4	RUP	y10
3	Crystal	y11

Рисунок 2.2 – Ранжування методології по критерію гнучкості

Заповнимо матрицю згідно цього ранжування. Відзначимо, що на перетині методології, які віднесені до однієї групи також будемо ставити одиницю (рис. 2.3).

	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11
y1	1	1	1/4	1/3	1/5	1/4	1/5	1/2	1/5	1/2	1/3
y2	1	1	1/4	1/3	1/5	1/4	1/5	1/2	1/5	1/2	1/3
y3	4	4	1	2	1/2	1	1/2	3	1/2	3	2
y4	3	3	1/2	1	1/3	1/2	1/3	2	1/3	2	1
y5	5	5	2	3	1	2	1	4	1	4	3
y6	4	4	1	2	1/2	1	1/2	3	1/2	3	2
y7	5	5	2	3	1	2	1	4	1	4	3
y8	2	2	1/3	1/2	1/4	1/3	1/4	1	1/4	1	1/2
y9	5	5	2	3	1	2	1	4	1	4	3
y10	2	2	1/3	1/2	1/4	1/3	1/4	1	1/4	1	1/2
y11	3	3	1/2	1	1/3	1/2	1/3	2	1/3	2	1

Рисунок 2.3 – Матриця попарних порівнянь методів керування за параметром гнучкості

Після побудови матриці парних порівнянь проводиться ранжування елементів *(k+1)* рівня ієрархії. Це ранжування здійснюється на підставі вектора пріоритетів $X = (x_1, x_2, \ldots x_6)$, який визначається як головний власний вектор матриці парних порівнянь з рівності:

$$A_n^k \cdot X = \lambda_{max} \cdot X, \tag{2.1}$$

де λ_{max} – Максимальне значення матриці.

Наступним кроком є підрахунок вектору пріоритетів. Для цього необхідно підрахувати суму кожного рядку, потів визначити суму отриманого стовпця. Останнім кроком потрібно поділити кожне значення суми на загальну суму. На рисунку 2.4 приведено результати підрахунків.

Сума	Вектор пріоритетів
4,77	0,025038959
4,77	0,025038959
21,50	0,112937963
14,00	0,073540999
31,00	0,162840784
21,50	0,112937963
31,00	0,162840784
8,42	0,044212148
31,00	0,162840784
8,42	0,044212148
14,00	0,073540999
190,37	1,00

Рисунок 2.4 – Вектор пріоритетів для методів управління за параметром гнучкості

2.2.2 Можливість масштабування

Можливість масштабування методології – це критерій, коли можливо використовувати методологію як для маленьких проєктів так і для великих. Відповідно, можемо використовувати для маленьких та великих команд. Чому цей параметр важливий? Як правило, в рамках однієї компанії прийнято використовувати одну методологію. Але час від часу фірма може отримувати проєкти різного масштабу. Добре, коли методологія працює для великих та маленьких команд. Це спрощує всі процеси в фірмі, та дає змогу брати різноманітні проєкти. Але не всі методології підходять [28].

Наприклад, можна помітити, що окремі процеси методології Scrum можуть значно погіршити показники для великої команди через те, що займають дуже багато часу.

Для попарного порівняння будуть використовуватися лише два значення 1 та 2. Ті методології, які однаково працюють на різних масштабах команди будуть відноситись до категорії 1. Інші – до категорії 2 (рис. 2.5). Такий вибір складу груп базується на інформації та статистиці використання методології в реальних умовах.

1	Waterfall	y1
1	Метод критичног	y2
2	DSDM	y3
2	FDD	y4
2	Scrum	y5
2	Lean	y6
1	XP	y7
2	six Sigma	y8
2	Kanban	y9
1	RUP	y10
2	Crystal	y11

Рисунок 2.5 – Ранжування методології за критерієм масштабування

Заповнимо таблицю згідно новому ранжуванню (рис. 2.6).

	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11
y1	1	1	2	2	2	2	1	2	2	1	2
y2	1	1	2	2	2	2	1	2	2	1	2
y3	1/2	1/2	1	1	1	1	1/2	1	1	1/2	1
y4	1/2	1/2	1	1	1	1	1/2	1	1	1/2	1
y5	1/2	1/2	1	1	1	1	1/2	1	1	1/2	1
y6	1/2	1/2	1	1	1	1	1/2	1	1	1/2	1
y7	1	1	2	2	2	2	1	2	2	1	2
y8	1/2	1/2	1	1	1	1	1/2	1	1	1/2	1
y9	1/2	1/2	1	1	1	1	1/2	1	1	1/2	1
y10	1	1	2	2	2	2	1	2	2	1	2
y11	1/2	1/2	1	1	1	1	1/2	1	1	1/2	1

Рисунок 2.6 – Заповнена таблиця попарного порівняння методології за критерієм масштабування

Розрахуємо вектор пріоритетів за вже відомою нам формулою (2.1) на рисунку 2.7.

Сума	Вектор пріоритетів
18,00	0,133333333
18,00	0,133333333
9,00	0,066666667
9,00	0,066666667
9,00	0,066666667
9,00	0,066666667
18,00	0,133333333
9,00	0,066666667
9,00	0,066666667
18,00	0,133333333
9,00	0,066666667
135,00	1,00

Рисунок 2.7 – Вектор пріоритетів методів управління за критерієм масштабування

Слід зазначити, що використовуються різні шкали для кожного критерію. Але при остаточних підрахунках цей момент не буде впливати на результат, тому що ці значення відносити один одного в рамках кожної таблиці.

2.2.3 Відмовостійкість

Відмовостійкість – важливий критерій в розробці будь якого продукту. Але у сучасному світі неможливо уявити ідеально працюючу систему, яка не видає помилок та не залежить від зовнішніх факторів. так само і з методами управління: існують процеси, які потребують складного обладнання, постійної присутності частини команди або замовника. Розуміємо, що таке не можливо і завжди є ризики, що щось може піїти не так. Тому при виборі моделі управління командами слід враховувати цей фактор.

Отже, для цього критерію буде використовуватись шкала із трьох значень:

– система відмовостійка, команда може працювати в будь якому форматі;

– система має помітні зовнішні проблеми. Проблеми можуть проявлятися в якості продукту, строках постачання, комунікації між членами команди;

– система повністю перестає працювати, зупиняються усі процеси до моменту полагодження.

Перш за все не відмовостійкими системами є Waterfall та метод критичного шляху. Ці методи ітеративні, тобто кожний наступний крок потребує завершення попереднього. Хоча слід зазначити, що у методі критичного шляху закладено деякий запас часу на непередбачувані ситуації. Стосовно методології FDD – це також ітеративний підхід, де на кожному етапі потрібні певні незамінні люди або команди.

Помітні проблеми трапляються у методах керування RUP та Crystal. Якщо у першій методології проблеми з тим, що процес зав’язано на певних людей, то методологія Crystal використовує короткострокове планування, що не дозволяє аналізувати ризики. На усунення проблем буде витрачено багато часу.

Отже, ранжування має такий кінцевий вигляд (рис. 2.8).

3	Waterfall	y1
3	Метод критичного	y2
1	DSDM	y3
3	FDD	y4
1	Scrum	y5
1	Lean	y6
1	XP	y7
1	six Sigma	y8
1	Kanban	y9
2	RUP	y10
2	Crystal	y11

Рисунок 2.8 – Ранжування методології за критерієм відмовостійкості

Заповнимо таблицю попарних порівнянь згідно новому ранжуванню (рис. 2.9).

	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11
y1	1	1	1/3	1	1/3	1/3	1/3	1/3	1/3	1/2	1/2
y2	1	1	1/3	1	1/3	1/3	1/3	1/3	1/3	1/2	1/2
y3	3	3	1	3	1	1	1	1	1	2	2
y4	1	1	1/3	1	1/3	1/3	1/3	1/3	1/3	1/2	1/2
y5	3	3	1	3	1	1	1	1	1	2	2
y6	3	3	1	3	1	1	1	1	1	2	2
y7	3	3	1	3	1	1	1	1	1	2	2
y8	3	3	1	3	1	1	1	1	1	2	2
y9	3	3	1	3	1	1	1	1	1	2	2
y10	2	2	1/2	2	1/2	1/2	1/2	1/2	1/2	1	1
y11	2	2	1/2	2	1/2	1/2	1/2	1/2	1/2	1	1

Рисунок 2.9 – Заповнена матриця попарних порівнянь методів управління за критерієм відмовостійкості

Розрахуємо вектор пріоритетів для критерію відмовостійкості (рис. 2.10).

Сума	Вектор пріоритетів
6,00	0,038961039
6,00	0,038961039
19,00	0,123376623
6,00	0,038961039
19,00	0,123376623
19,00	0,123376623
19,00	0,123376623
19,00	0,123376623
19,00	0,123376623
11,00	0,071428571
11,00	0,071428571
154,00	1,00

Рисунок 2.10 – Вектор пріоритетів методів управління за критерієм відмовостійкості

2.2.4 Низька планка входу

Коли нова команда знайомиться з новою методологією, вона витрачає певний час. Слід розуміти, що зі зміною кожного члена команди наступну людину потрібно навчити працювати в методології. Чим простіша методологія, чим менше артефактів та критеріїв вона має, тим простіше людина розуміє та адаптується.

Найпростішою є каскадна методологія – мінімальна кількість артефактів, зрозумілий шаблон роботи. Щоб зрозуміти процес розробки достатньо просто показати план роботи та указати дедлайни.

Самим складним для розуміння є методології родини agile. Ці методології мають велику кількість додаткових артефактів, нових людей, способів роботи. Час від часу навіть членам команди потрібно нагадувати що потрібно робити. До цих методології відносяться Scrum, Kanbun, Crustal, sixSigma.

Окремо стоїть методологія XP. Сам підхід дуже простий, але потрібно змінювати саму культуру розробки застосунків, написання коду, роботи не в команді, а в парі. З точки зору психології – це важка методологія.

Для ранжування методології за критерієм складності методології будемо використовувати шкалу із п’яти значень. ось такий результат маємо на рисунку 2.11.

1	Waterfall	y1
2	Метод критичного	y2
3	DSDM	y3
3	FDD	y4
5	Scrum	y5
2	Lean	y6
4	XP	y7
5	six Sigma	y8
5	Kanban	y9
1	RUP	y10
5	Crystal	y11

Рисунок 2.11 – Ранжування методології за критерієм складності

Заповнимо матриці попарних порівнянь (рис. 2.12).

	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11
y1	1	2	3	3	5	2	4	5	5	1	5
y2	1/2	1	2	2	4	1	3	4	4	1/2	4
y3	1/3	1/2	1	1	3	1/2	2	3	3	1/3	3
y4	1/3	1/2	1	1	3	1/2	2	3	3	1/3	3
y5	1/5	1/4	1/3	1/3	1	1/4	1/2	1	1	1/5	1
y6	1/2	1	2	2	4	1	3	4	4	1/2	4
y7	1/4	1/3	1/2	1/2		1/3	1	2	2	1/4	2
y8	1/5	1/4	1/3	1/3	1	1/4	1/2	1	1	1/5	1
y9	1/5	1/4	1/3	1/3	1	1/4	1/2	1	1	1/5	1
y10	1	2	3	3	5	2	4	5	5	1	5
y11	1/5	1/4	1/3	1/3	1	1/4	1/2	1	1	1/5	1

Рисунок 2.12 – Заповнена матриця попарних порівнянь методів управління за критерієм складності

Розрахуємо вектор пріоритетів для критерію відмовостійкості (рис. 2.13).

Сума	Вектор пріоритетів
36,00	0,18675105
26,00	0,134875759
17,67	0,091646349
17,67	0,091646349
6,07	0,03147101
26,00	0,134875759
9,17	0,047552351
6,07	0,03147101
6,07	0,03147101
36,00	0,18675105
6,07	0,03147101
192,77	1,00

Рисунок 2.13 – Вектор пріоритетів методів управління за критерієм складності

2.2.5 Довжина ітерації

По-перше, дамо визначення довжини ітерації, або спринту.

Спринт – це невеликий фіксований відрізок часу, коли команда робить якусь обмежену частину проєкту. Наприклад, команда може рухатись

двотижневими спринтами, з кожним спринтом додаючи в проєкт нові можливості.

Але деякі методології не використовують підхід зі спринтами. Наприклад, методологія Waterfall випускає готовий продукт лише наприкінці розробки.

Що стосується гнучких методології , таких як Scrum, Kanban – ці методології випускають продукт кожен спринт. XP методология не прив’язана до системи спринтів, тому продукти випускаються швидко із-зі самої методології та підходу.

Тепер можемо ранжувати наші методології за критерієм ітерації. Будемо використовувати 3 ранги для зручності (рис. 2.14).

3	Waterfall	y1
3	Метод критичного	y2
1	DSDM	y3
2	FDD	y4
1	Scrum	y5
2	Lean	y6
1	XP	y7
2	six Sigma	y8
1	Kanban	y9
2	RUP	y10
2	Crystal	y11

Рисунок 2.14 – Ранжування методології за критерієм довжини ітерації

Заповнимо матриці попарних порівнянь (рис. 2.15).

	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11
y1	1	1	1/3	1/2	1/3	1/2	1/3	1/2	1/3	1/2	1/2
y2	1	1	1/3	1/2	1/3	1/2	1/3	1/2	1/3	1/2	1/2
y3	3	3	1	2	1	2	1	2	1	2	2
y4	2	2	1/2	1	1/2	1	1/2	1	1/2	1	1
y5	3	3	1	2	1	2	1	2	1	2	2
y6	2	2	1/2	1	1/2	1	1/2	1	1/2	1	1
y7	3	3	1	2	1	2	1	2	1	2	2
y8	2	2	1/2	1	1/2	1	1/2	1	1/2	1	1
y9	3	3	1	2	1	2	1	2	1	2	2
y10	2	2	1/2	1	1/2	1	1/2	1	1/2	1	1
y11	2	2	1/2	1	1/2	1	1/2	1	1/2	1	1

Рисунок 2.15 – Заповнена матриця попарних порівнянь методів управління за критерієм довжини ітерації

Визначимо вектор пріоритетів на основі отриманої матриці порівнянь (рис. 2.16).

	Сума	Вектор пріоритетів
?	5,83	0,039771823
?	5,83	0,039771823
	20,00	0,136360537
	11,00	0,074998295
	20,00	0,136360537
	11,00	0,074998295
	20,00	0,136360537
	11,00	0,074998295
	20,00	0,136360537
	11,00	0,074998295
	11,00	0,074998295
	146,67	1,00

Рисунок 2.16 – Вектор пріоритетів методів управління за критерієм довжини ітерації

2.2.6 Довжина розробки

Життєвий цикл програмного забезпечення (ПЗ) – період часу, який починається з моменту прийняття рішення про необхідність створення програмного продукту і закінчується в момент його повного вилучення з експлуатації.

Основні етапи життєвого циклу:

– придбання (дії та завдання замовника, що набуває ПЗ);

– постачання (дії та завдання постачальника, який забезпечує замовника програмним продуктом або послугою);

– розробка (дії та завдання, що виконуються розробником: створення ПЗ, оформлення проектної та експлуатаційної документації, підготовка тестових та навчальних матеріалів тощо);

– експлуатація (дії та завдання оператора – організації, що експлуатує систему);

– супровід (дії та завдання, що їх супроводжує організацією, тобто службою супроводу). Супровід – внесення змін до ПЗ з метою виправлення помилок, підвищення продуктивності або адаптації до умов роботи або вимог, що змінилися.

На основі відомих даних проранжуємо наші методології за критерієм довжини розробки (рис. 2.17).

1	Waterfall	y1
1	Метод критичного	y2
3	DSDM	y3
4	FDD	y4
3	Scrum	y5
1	Lean	y6
3	XP	y7
4	six Sigma	y8
3	Kanban	y9
2	RUP	y10
2	Crystal	y11

Рисунок 2.17 – Ранжування методології за критерієм довжини розробки

Заповнимо матриці попарних порівнянь (рис. 2.18).

	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11
y1	1	1	3	4	3	1	3	4	3	2	2
y2	1	1	3	4	3	1	3	4	3	2	2
y3	1/3	1/3	1	2	1	1/3	1	2	1	1/2	1/2
y4	1/4	1/4	1/2	1	1/2	1/4	1/2	1	1/2	1/3	1/3
y5	1/3	1/3	1	2	1	1/3	1	2	1	1/2	1/2
y6	1	1	3	4	3	1	3	4	3	2	2
y7	1/3	1/3	1	2	1	1/3	1	2	1	1/2	1/2
y8	1/4	1/4	1/2	1	1/2	1/4	1/2	1	1/2	1/3	1/3
y9	1/3	1/3	1	2	1	1/3	1	2	1	1/2	1/2
y10	1/2	1/2	2	3	2	1/2	2	3	2	1	1
y11	1/2	1/2	2	3	2	1/2	2	3	2	1	1

Рисунок 2.18 – Заповнена матриця попарних порівнянь методів управління за критерієм довжини розробки

Визначимо вектор пріоритетів на основі отриманої матриці порівнянь (рис. 2.19).

Сума	Вектор пріоритетів
27,00	0,161841395
27,00	0,161841395
10,00	0,059941258
5,42	0,032468181
10,00	0,059941258
27,00	0,161841395
10,00	0,059941258
5,42	0,032468181
10,00	0,059941258
17,50	0,104897201
17,50	0,104897201
166,83	1,00

Рисунок 2.19 – Вектор пріоритетів методів управління за критерієм довжини розробки

2.3 Порівняння методів оцінок між собою

Введемо умовні позначення, будемо позначати кожну оцінку як x_y, де y це індекс від 1 до 6. Отже:

- гнучкість – x1;
- можливість масштабування – x2;
- відмовостійкість – x3;
- низька планка входу – x4;
- довжина ітерації – x5;
- довжина розробки – x6.

Побудуємо каркас нашої матриці та заповнюємо головну діагональ одиницями (рис. 2.20).

	x1	x2	x3	x4	x5	x6
x1	1					
x2		1				
x3			1			
x4				1		
x5					1	
x6						1

Рисунок 2.20 – Основа матриці порівнянь для методів оцінки методології

Для того, щоб заповнити матрицю значеннями, проаранжуємо критерії по важливості, спираючись на власний досвід.

Зараз проєкти швидко змінюються та адаптуються до зовнішніх умов, тому гнучкі методології розробки дуже важлива. Цей критерій має перший пріоритет [29].

Наступний критерій по важливості – довжина ітерації. Продукт потрібно випускати якомога частіше, тому частота релізів повинна бути високою. Відмовостійкість потрібна для того, щоб у будь яких непередбачуваних обставинах проєкт продовжував працювати. Зовнішні фактори, такі як хвороба, або поломка обладнання не повинні критично сприяти на розробку [30].

Ранжування представлене на рисунку 2.21.

1	Гнучкість - x1;
5	Можливість масштабування - x2;
3	Відмовостійкість - x3;
6	Низька планка входу - x4;
2	Довжина ітерації - x5;
4	Довжина розробки - x6.

Рисунок 2.21 – Ранжування оцінок методології

Спираючись на це ранжування, було заповнено таблицю (рис. 2.22). Можна помітити, що таблиця симетрична відносно головної діагоналі.

	x1	x2	x3	x4	x5	x6
x1	1	5	3	6	2	4
x2	1/5	1	1/3	2	1/4	1/2
x3	1/3	3	1	4	1/2	2
x4	1/6	1/2	1/4	1	1/5	1/3
x5	1/2	4	2	5	1	3
x6	1/4	2	1/2	3	1/3	1

Рисунок 2.22 – Заповнена таблиця

Маючи всі дані тепер можемо порахувати вектор пріоритетів для методів оцінки (рис. 2.23).

Сума	Вектор приоритетів
21,00	0,343417825
4,28	0,070046334
10,83	0,177159989
2,45	0,040065413
15,50	0,253475061
7,08	0,115835377
61,15	1,00

Рисунок 2.23 – Розрахунок вектору пріоритетів

3 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Після того як завершили всі підрахунки, створимо ще одну таблицю для того, щоб виявити найкращу методологію. Для цього розташуємо у стовпець усі методології, а по горизонталі розташуємо критерії оцінки.

Наступним кроком будемо заповнювати осередки таблиці, для цього будемо переносити відповідні вектори пріоритетів від кожного критерію. Останнім кроком перемножимо отриману матрицю на вектор пріоритетів саміх критеріїв порівнянь. На рисунку 3.1 представлені результати.

	x1	x2	x3	x4	x5	x6	
y1	0,025038959	0,133333333	0,038961039	0,18675105	0,039771823	0,161841395	0,061151056
y2	0,025038959	0,133333333	0,038961039	0,134875759	0,039771823	0,161841395	0,059072651
y3	0,112937963	0,066666667	0,123376623	0,091646349	0,136360537	0,059941258	0,110491229
y4	0,073540999	0,066666667	0,038961039	0,091646349	0,074998295	0,032468181	0,063270393
y5	0,162840784	0,066666667	0,123376623	0,03147101	0,136360537	0,059941258	0,125217797
y6	0,112937963	0,066666667	0,123376623	0,134875759	0,074998295	0,161841395	0,108473076
y7	0,162840784	0,133333333	0,123376623	0,047552351	0,136360537	0,059941258	0,130531859
y8	0,044212148	0,066666667	0,123376623	0,03147101	0,074998295	0,032468181	0,065742457
y9	0,162840784	0,066666667	0,123376623	0,03147101	0,136360537	0,059941258	0,125217797
y10	0,044212148	0,133333333	0,071428571	0,18675105	0,074998295	0,104897201	0,075820298
y11	0,073540999	0,066666667	0,071428571	0,03147101	0,074998295	0,104897201	0,075001234
							0,999989848

Рисунок 3.1 – Розрахунок найкращої методології методом перемноження двох матриць

Проаналізуємо отриманий стовпець. Як можна побачити, найбільший результат має сьома методологія, а саме 0,130. За математичним розрахунком отримали, що методологія екстремальної розробки є кращою методологією для отримання якісної роботи. Наступними є Scrum та Kanban з однаковими значеннями в 0,125.

Але, як відомо, більшість компанії використовують саме Scrum та Kanban. Які основні недоліки екстремальної розробки і чому її не обирають у більшості випадків? Менеджери не обирають, тому що розуміють, що перехід до цієї методології буде занадто довгим та складним [31].

Тому можна зробити висновок, що з математичної точки зору оцінка методів пройшла правильно, не враховуючи людські фактори. Рисунок 3.2 ілюструє статистику використання методологій в компаніях.

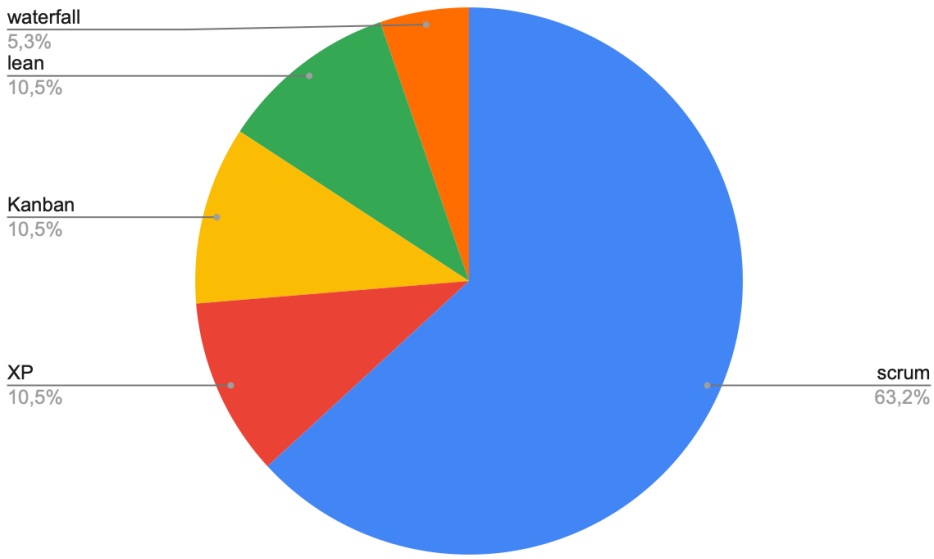

Рисунок 3.2 – Статистика використання методів керування командами

ВИСНОВКИ

У рамках кваліфікаційної роботи було проаналізовано та порівняно одинадцять популярних методологій керування командами. Було виявлено, що методологія екстремального програмування є найкращою з математичної точки зору. Також, у роботі були обрані критерії оцінки для методологій та ці критерії буди порівнянні між собою.

У ході роботи були вирішені такі задачі:

- проведено аналіз існуючих методів керування;
- проаналізовано критерії порівняння за якими будемо аналізувати методології;
- порівняно критерії між собою та побудовано вектор пріоритетів для них;
- порівняно методи керування за критеріями між собою;
- використано метод перемноження векторів, та виявлено найкращу методологію з точки зору математичного аналізу;
- проаналізовано отримані результати, та обґрунтування отриманих даних;
- порівняно отримані результати зі статистикою;
- зроблено висновки стосовно роботи.

Подальшою ціллю цієї роботи є створення власної методології на базі вже існуючих, для того, щоб створити універсальну, легку та просто методологію, яка буде допомагати командам вирішувати задачі різного рівня.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Богомолова, И. С., Гриненко, С. В., Едалова, Е. С., Задорожняя, Е. К., Развадовская, Ю. В., Седова, Т. В., ... & Шевченко, И. К. (2014). Инновационный и проектный менеджмент. *Учебное пособие.–Ростовна-Дону: Изд-во ЮФУ.–2014.–181 с*.

2. Строилова, Э. В. (2013). Проектный менеджмент и реинжиниринг. *Фундаментальные исследования*, *5*(4).

3. Сазерленд, Д. (2016). *Scrum: Революционный метод управления проектами*. " Манн, Иванов и Фербер".

4. Petersen, K., Wohlin, C., & Baca, D. (2009, June). The waterfall model in large-scale development. In *International Conference on Product-Focused Software Process Improvement* (pp. 386-400). Springer, Berlin, Heidelberg.

5. Davie, E. W., & Ratnoff, O. D. (1964). Waterfall sequence for intrinsic blood clotting. *Science*, *145*(3638), 1310-1312.

6. Акимов, В. А., Балашов, В. Г., & Заложнев, А. Ю. (2003). Метод нечеткого критического пути. *Управление большими системами: сборник трудов*, (3), 5-10.

7. Абдулкаримов, Ш. Н. (2016). Сравнительный анализ метода критического пути и метода критической цепи. *Научный журнал*, 58-59.

8. Zafar, I., Nazir, A. K., & Abbas, M. (2017). The impact of agile methodology (DSDM) on software project management. In *Circulation in Computer Science: International Conference on Engineering, Computing and Information Technology (ICECIT 2017)* (pp. 1-6).

9. Coyle, S., & Conboy, K. (2009, May). A study of risk management in DSDM. In *International Conference on Agile Processes and Extreme Programming in Software Engineering* (pp. 142-148). Springer, Berlin, Heidelberg.

10. Seshadri, N., & Winters, J. H. (1994). Two signaling schemes for improving the error performance of frequency division duplex (FDD) transmission

systems using transmitter antenna diversity. *International Journal of Wireless Information Networks*, *1*(1), 49-60.

11. Usman, M., Soomro, T. R., & Brohi, M. N. (2014). Embedding project management into XP, SCRUM and RUP. *European Scientific Journal*, *10*(15).

12. Callegari, D. A., & Bastos, R. M. (2007, March). Project management and software development processes: integrating RUP and PMBOK. In *2007 International Conference on Systems Engineering and Modeling* (pp. 1-8). IEEE.

13. Bertagnolli, F. (2018). *Lean management*. Springer Fachmedien Wiesbaden.

14. Charron, R., Harrington, H. J., Voehl, F., & Wiggin, H. (2014). *The lean management systems handbook* (Vol. 4). CRC Press.

15. Angioni, M., Carboni, D., Pinna, S., Sanna, R., Serra, N., & Soro, A. (2006). Integrating XP project management in development environments. *Journal of Systems Architecture*, *52*(11), 619-626.

16. Хабибуллин, Х. Х. (2011). Lean Six Sigma как методология улучшения бизнес-процессов. *Вестник Казанского юридического института МВД России*, (6), 126-130.

17. Kwak, Y. H., & Anbari, F. T. (2006). Benefits, obstacles, and future of six sigma approach. *Technovation*, *26*(5-6), 708-715.

18. Junior, M. L., & Godinho Filho, M. (2010). Variations of the kanban system: Literature review and classification. *International Journal of Production Economics*, *125*(1), 13-21.

19. Huang, C. C., & Kusiak, A. (1996). Overview of Kanban systems.

20. Таганова, О. А. (2017). Методология разработки корпоративных информационных систем RUP. In *Научные труды Калужского государственного университета имени КЭ Циолковского* (pp. 190-195).

21. Wyllie, P. J. (1977). Crustal anatexis: an experimental review. *Tectonophysics*, *43*(1-2), 41-71.

22. Kou, G., & Lin, C. (2014). A cosine maximization method for the priority vector derivation in AHP. *European Journal of Operational Research*, *235*(1), 225-232.

23. Волокобинский, М. Ю., Пекарская, О. А., & Рази, Д. А. (2016). Принятие решений на основе метода анализа иерархий. *Финансы: теория и практика*, (2 (92)), 33-42.

24. Никул, Е. С. (2012). Алгоритм анализа матриц парных сравнений с помощью вычисления векторов приоритетов. *Известия Южного федерального университета. Технические науки*, *127*(2), 241-247.

25. Саати, Т. Л. (2016). Относительное измерение и его обобщение в принятии решений. Почему парные сравнения являются ключевыми в математике для измерения неосязаемых факторов. *Cloud of science*, *3*(2), 171-262.

26. Саати, Т. (1993). Принятие решений.

27. Брагина, Т. И., & Табунщик, Г. В. (2010). Сравнительный анализ итеративных моделей разработки программного обеспечения. *Радіоелектроніка, інформатика, управління*, (2 (23)), 130-138.

28. Муравьева, Я. И. (2016). Жизненный цикл проекта. *Экономика и социум*, (3), 1888-1893.

29. Заренков, В. А. (2006). Управление проектами.

30. Мазур, И. И., Шапиро, В. Д., Ольдерогге, Н. Г., & Полковников, А. В. (2010). Управление проектами.

31. Heerkens, G. R. (2002). *Project management*. McGraw Hill Professional.