ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Пояснювальна записка

до дипломної роботи

МАГІСТРА

на тему: "УДОСКОНАЛЕННЯ ПРОЦЕСІВ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ШЛЯХОМ МІНІМІЗАЦІЇ РИЗИКІВ"

Виконала:
студентка 2 року навчання
групи 8.04.126.010.20.1
спеціальності 126 "Інформаційні системи та технології"
Винокурова Вероніка Костянтинівна

Керівник:
д.пед.н., проф.
Гризун Людмила Едуардівна

Харків – 2021 рік

РЕФЕРАТ

Дипломна робота містить 65 сторінок, 18 рисунків, 5 таблиць, список літератури з 37 найменувань.

Метою роботи є побудова та впровадження моделі мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення.

Основна мета дослідження полягає в оптимізації процесів розробки програмного забезпечення та впровадження плану реагування на ризики шляхом їх мінімізації.

Об'єкт дослідження - процес проєктування програмного забезпечення.

Предмет дослідження -мінімізація ризикових збитків для удосконалення процесів проєктування програмного забезпечення

Теоретичною значимістю роботи є систематизація наявних знань про методологію ризик-менеджменту, розширення знань про інструментарій різних методів управління ризиком, а також складання універсального алгоритму управління ризиками на проєкті.

Практична значущість роботи даної роботи полягає в рішенні важливості впровадження ризику менеджменту в процес планування розробки програмного забезпечення для ранньої ідентифікації та аналізу можливих проблем на проєкті.

Наукова новизна досліджень полягає в обґрунтуванні важливості аналізу ризиків та впровадження плану реагування на ризики відповідно до властивостей проєкту, який допоможе зменшити вірогідність появи негативних факторів на проєкті та створити сприятливі умови на проєкті які позначаються на продуктивності команди на краще.

КЛЮЧОВІ СЛОВА: ПРОЄКТНИЙ МЕНЕДЖМЕНТ, МЕНЕДЖЕР, УПРАВЛІННЯ, РИЗИК, РИЗИК МЕНЕДЖМЕНТ, ОПТИМІЗАЦІЯ, ПЛАН РЕАГУВАННЯ.

ABSTRACT

Thesis contains 65 pages, 18 figures, 5 tables, bibliography of 37 titles.

The aim of the work is to build and implement a model of minimizing risk losses to improve software design processes.

The main goal of the study is to optimize software development processes and implement a risk response plan by minimizing them.

The object of research is the software design process.

The subject of research -minimization of risk losses to improve software design processes

The theoretical significance of the work is the systematization of existing knowledge about the methodology of risk management, expanding knowledge about the tools of different methods of risk management, as well as compiling a universal risk management algorithm for the project.

The practical significance of this work is to address the importance of implementing risk management in the planning process of software development for early identification and analysis of possible problems in the project.

The scientific novelty of the research is to substantiate the importance of risk analysis and implementation of a risk response plan in accordance with the properties of the project, which will help reduce the likelihood of negative factors on the project and create favorable conditions for the project.

KEY WORDS: PROJECT MANAGEMENT, MANAGER, MANAGEMENT, RISK, RISK MANAGEMENT, OPTIMIZATION, RESPONSE PLAN.

ЗМІСТ

ВСТУП

Протягом попередніх 30 років сформувалася нова культура управлінської діяльності - управління проєктами. Це сталося, тому що в теперішній час переважна більшість бізнесу є проєктно-орієнтованим. До нього відноситься вся інноваційна, інвестиційна сфери, штучний інтелект, консалтинг, інжиніринг і т.д. Для виробництв вищевказаного типу надзвичайно актуальним є професійне управління проєктами.

До недавнього часу під поняттям «проєкт» розумівся комплект проєктно-кошторисної документації на створення будівель, споруд або технічних пристроїв. У сучасному професійному управлінні з поняттям проєкту зв'язується процес здійснення комплексу цілеспрямованих заходів щодо створення нового продукту або послуг в рамках встановлених бюджету, часу і якості.

Багато IT компаній реалізують свою діяльність у формі проєктів. Проте навіть в не самих складних проєктах виникають проблеми, які можуть проявлятися в незадоволеності замовника і керівництва компанії, зриві термінів, конфліктах всередині команди, що реалізує проєкт, та в іншому. Розв'язання цих проблем є важливою складовою діяльності менеджера проєкту і має спиратися на відповідні методології, інструментальні засоби та стандарти.

Сьогодні актуальною є проблема ефективного управління ризиками. Основу будь-якої управлінської діяльності складають прийняті органами управління одноособово чи колегіально, і які спрямовані на досягнення певної мети, яка стоїть перед організацією. Одним з основних завдань, які виконують у рамках управління IT-проєктами, є управління ризиками проєктної діяльності або управління ризиками проєкту.

На сьогодні більшість IT-компанії, зазвичай, стають конкурентоспроможними тільки за рахунок інноваційної діяльності, яка за своєю сутністю пов'язана з різними ризиками, тобто ймовірністю виникнення збитків або недоотримання прибутків порівняно з прогнозованими. Тому ризик є одночасно як причиною можливих збитків, так і джерелом потенційних прибутків. Також ризик – важлива складова процесу управління, неврахування якого призводить до вироблення, аналізу й прийняття не обґрунтованих і малоефективних управлінських рішень. Основне завдання управління ризиками – не відмовитись від ризику як такого взагалі, а приймати ризикові рішення, ґрунтуючись на об'єктивних критеріях і допустимих втратах. Прийняті керівником проєкту ризик-

орієнтовані рішення часто призводять до більш ефективної реалізації програмних проєктів, від яких отримують свою вигоду як замовники і розробники програмного забезпечення, так і його безпосередні користувачі

Це завдання відокремлюється від інших функцій управління ІТ-проєктами, що актуалізує пошук ефективних засобів мінімізації ризиків та збитків від них для удосконалення процесів проєктування програмного забезпечення.

Об’єкт дослідження - процес проєктування програмного забезпечення.

Предмет дослідження -мінімізація ризикових збитків для удосконалення процесів проєктування програмного забезпечення

Метою роботи є побудова та впровадження моделі мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення.

Відповідно до мети сформульовано такі завдання дослідження.

1. Проаналізувати сучасні підходи до проєктування програмного забезпечення та виявити проблеми його проєктування.

2. Вивчити ризики, що виникають при проєктуванні програмного забезпечення, та сучасні методи управління ними.

3. Побудувати математичну модель задачі мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення.

4. Впровадити побудовану модель мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення на підприємстві.

5. Спланувати заходи для зниження ризиків та ризикових збитків та експериментально перевірити їх ефективність.

Робота складається із вступу, трьох розділів, висновків та списку використаних джерел.

1. ТЕОРЕТИЧНІ ЗАСАДИ УДОСКОНАЛЕННЯ ПРОЦЕСІВ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Аналіз сучасних методологій проєктування програмного забезпечення

Напрямок проєктного менеджменту початок формування в останні десятиліття, стало загальновизнаною методологією здійснення проєктів і перетворилося на невід'ємну частину ведення бізнесу і загальнолюдської культури. Управління проєктами є новою економікою, що розвивається областю знань і практики в управлінні, має життєво важливе постійно зростаюче значення в розвитку економіки та суспільства.

Відповідно до джерела [1], проєкт - це обмежена в часі, ресурсах та вимогах якості унікальна сукупність процесів, направлена на досягнення унікальних цілей та завдань для створення нової цінності. Методологія управління проєктами - це методологія об'єднання, планування, керівництва, координації трудових, фінансових, і матеріально - технічних ресурсів, спрямованих на ефективне досягнення його цілей шляхом застосування сучасних методів, техніки і технологій управління, для досягнення певних результатів за складом і обсягом робіт, вартості , часу і якості.

Переважна більшість реалізованих компаніями проєктів є надзвичайно складними з точки зору прийняття і виконання управлінських рішень. Як відомо, успіх таких проєктів залежить не тільки від правильності стратегічних рішень, прийнятих на початкових стадіях, а й значною мірою визначається їх обгрунтованістю і оперативністю на наступних етапах реалізації. Тому питання впровадження прогресивних підходів, методів і систем в практику управління проєктами підприємства набувають безперечну важливість і актуальність.

Управління проєктами - одна з найбільш швидко управлінських дисциплін нашого часу. В умовах сучасної економіки, коли конкуренція у всіх областях зросла, здається, до межі, а тривалість життя окремих товарів обчислюються місяцями і навіть тижнями, застосування технологій управління проєктами є необхідним не тільки для процвітання, але і для виживання майже кожного комерційного підприємства.

Підходи до розробки програмного забезпечення визначають успіх проєкту, адже без правильно підібраної методології складно досягти стабільності у роботі продукту, безпеки та стійкості функціональних особливостей. Проєктний підхід

до управління вже довів свою ефективність на практиці і застосовується провідними світовими компаніями, його успішно використовують IBM, Motorola, Boeing, Intel і безліч інших.

У програмного забезпечення, як у живої істоти є свій життєвий цикл. Життєвий цикл програмного забезпечення, відповідно до джерела [17], (Software Development Life Cycle, SDLC) – умовна схема, що включає фази створення програмного забезпечення. Цикл розробки пропонує шаблон, використання якого полегшує проєктування, створення та випуск якісного програмного забезпечення. На виході має вийти економічно вигідний продукт, який відповідає вимогам замовника. Моделі життєвого циклу багато в чому зумовлюють і методології розробки програмного забезпечення.

Для повнішої характеристики питання були розглянуті роботи [3, 4]. Нижче описані етапи циклу відповідно до цих джерел, починаючи з моменту, коли замовник сформулював свої ідеї виконавцю.

Грамотне планування функціональності майбутнього продукту і аналіз вимог грають ключову роль для всього проєкту. За цей етап несе відповідальність менеджер проєкту, так як саме він відповідає за успіх всього процесу розробки. На цьому етапі важливо максимально точно та однозначно визначити вимоги до проєкту і для команди проєкту, і для бізнесу. Аналітики допомагають визначити кінцеві цілі та завдання роботи.

Після планування настає черга UX / UI дизайнерів - фахівців, які проєктують інтерфейси. Дизайнери займаються вивченням поведінки користувачів і вибудовуванням зрозумілого людині інтерфейсу. Візуальний вигляд продукту - також результат роботи дизайнерів. Надалі йде розробка. Розробники йдуть однією з методологій - для компанії це в основному Agile. Ця методологія передбачає гнучкий ітеративний підхід - тобто розробники діють послідовно, розділяючи проєкт на більш дрібні завдання. Ітерації в Agile називаються спринті, і в один спринт входять роботи по всіх напрямках: планування, дизайн, розробка, тестування.

Потім настає фаза тестування. Фахівці з тестування виконують різні види тестування: модульне, інтеграційне, тестування інтерфейсу і інші види залежно від мети. Ця категорія фахівців повинна прийти до кінцевого висновку, що в продукті немає помилок і він готовий до релізу. Після цього продукт можна впроваджувати і інтегрувати зі стороннім програмним забезпеченням. Процес розробки на цьому не закінчується - він триває, поки не будуть внесені

доопрацювання.

Готовий продукт може потребувати додаткової підтримки, будь то додаткові питання з приводу роботи продукту від клієнтів, або необхідність внести зміни в уже закладені функції, - фахівці служби підтримки завжди готові прийти на допомогу. Цілями супроводу є підганяння системи з урахуванням зміни середовища і відповідно до бажання користувачів, а також підтримка працездатності системи програмного забезпечення якомога довше.

Розробка програмного забезпечення закінчується із передачею програмного забезпечення клієнту. Готова програма має бути такою, яку хотів мати клієнт. Однак, програмне забезпечення має розвиватися далі. У ході роботи виявляються аномалії, змінюється робоче середовище, виникають нові вимоги.

Все це необхідно знати і вміти застосовувати, для того щоб розробляти ПО. Основною причиною більшості провалів програмних проєктів є саме застосування конкретних методів управління розробкою.

Модель розробки програмного забезпечення описує, які стадії життєвого циклу воно проходить і що відбувається на кожній з них, відповідно до джерела [19].

Існує багато моделей розробки програмного забезпечення, наприклад:

1. Code and fix - модель кодування і усунення помилок;
2. Waterfall Model - каскадна модель, або «водоспад»;
3. V-model - V - образна модель, розробка через тестування;
4. Incremental Model - інкрементна модель;
5. Iterative Model - ітеративна (або ітераційна) модель;
6. Spiral Model - спіральна модель;
7. Chaos model - модель хаосу;
8. Prototype Model - прототипна модель.

З цих моделей найбільш популярні шість основних: каскадний, V-образна, інкрементна, спіральна і гнучка. Розберемо їх докладніше.

Методологія Waterfall (каскадна модель, «водоспад») - сама «стара» з усіх. Вперше вона була закладена американським ученим в галузі інформатики Уінстоном Уокером Ройсом в 1970 році у відповідь на потребу управління все більш ускладняються процесом розробки програмного забезпечення. З того часу вона набула широкого поширення, особливо в сфері програмного забезпечення.

Ця модель вперше формалізувала структуру етапів розробки програмного забезпечення. Вона підтримує стратегію одноразового проходу етапів розробки

програмного забезпечення. Каскадна модель базується на повному формулюванні вимог на початку життєвого циклу. До їх уточнення чи зміни наступних кроках життєвого циклу повернення немає.

Методологія Waterfall ділиться на окремі етапи. Відповідно до джерела [22] спочатку необхідно зібрати і проаналізувати вимоги, потім розробити рішення (і підхід), впровадити рішення і виправити проблеми, якщо вони з'явилися. На всіх етапах моделі виконуються допоміжні та організаційні процеси та роботи, що включають управління проєктом, оцінку та управління якістю, верифікацію та атестацію, менеджмент конфігурації, розробку документації. У результаті завершення кроків формується проміжні продукти розробки, які можуть змінюватися наступних кроках. На рис. 1.1 наведені етапи розробки за використанням каскадної моделі.

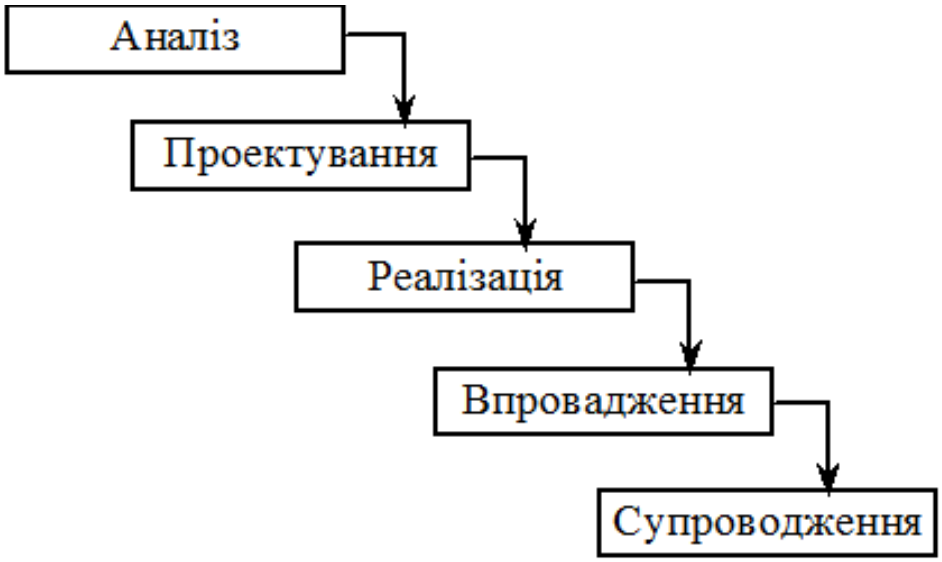

Рис. 1.1. Етапи каскадної моделі

Витрачаючи час на ранніх стадіях розвитку проєкту, менеджери створюють умови для своєчасного виконання вимог. Це дозволяє заощадити час і сили на виправлення недоліків і вирішення проблем в подальшому. Waterfall підійде для коротких нескладних проєктів, проєктів з чітко встановленими вимогами, змінювати вимоги до продукту не планується, проєктів, в яких змінюються ресурси, залежні від докладної документації.

Перевагами Waterfall моделі було розглянуто у роботі [23]:

1. Простота використання - цю модель просто зрозуміти і

використовувати. Розподіл на етапи досить інтуїтивно, його просто освоїти незалежно від досвіду;

2. Структура - жорсткість методології Waterfall - одночасно і недолік, і явна перевага. Чіткий поділ на етапи дозволяє організувати і розподілити роботу;

3. Документація - оскільки багато уваги приділяється збору і розуміння вимог, модель Waterfall в значній мірі спирається на документацію. Завдяки цьому нових ресурсів простіше влитися в проєкт і почати роботу над ним.

Серед недоліків Waterfall моделі слід відзначити такі:

1. Проєкт залежить від конкретних виконавців;

2. Виконавці працюють за чітким планом;

3. Вартість та тривалість проєкту заздалегідь розраховують та затверджують, а під час роботи їх не змінюють;

4. Вимоги не змінюються під час роботи.

V-образна модель (розробка через тестування) - це вдосконалена каскадна модель, в якій замовник з командою програмістів одночасно складають вимоги до системи і описують, як будуть тестувати її на кожному етапі. Історія цієї моделі починається в 1980-х. Ця модель найкраще підійде для проєктів, в яких важлива надійність і ціна помилки дуже висока. Наприклад, при розробці подушок безпеки для автомобілів або систем спостереження за пацієнтами в клініках.

Основне призначення V-подібної моделі – забезпечення планування тестування системи на ранніх стадіях проєкту. V-подібна модель підтримує стратегію одноразового послідовного виконання етапів розробки життєвого циклу та базується на попередньо повному формуванні вимог.

На етапі аналізу вимог до системи розробляється план введення в дію та забезпечення приймання.

Далі виконується проєктування системи - проєктування системної архітектури та аналіз вимог до програмних засобів. На цьому етапі розробляється план кваліфікаційних випробувань системи.

На етапі проєктування програмних засобів виконується проєктування програмної архітектури і технічне проєктування програмних засобів.

На етапі складання та кваліфікаційних випробувань програмних засобів здійснюється складання програмних засобів у функціонально пов'язані групи та кваліфікаційні випробування даних груп.

Потім йде етап складання та кваліфікаційних випробувань системи виконується повне складання системи та її кваліфікаційні випробування. Метою

цього етапу є підтвердження результатів етапу проєктування системи.

На етапі введення в дію та забезпечення приймання здійснюються приймальні випробування, що мають на меті перевірку користувачем відповідності системи вихідним вимогам. На рис. 1.2 наведені етапи розробки за використанням V-подібної моделі.

Рис. 1.2. Етапи V-подібної моделі

Переваги V-подібної моделі було розглянуто у роботі [28]:

1. Кількість помилок в архітектурі програмного забезпечення зводиться до мінімуму;
2. Велика роль приділяється верифікації та атестації програмного продукту, починаючи з ранніх стадій його розробки, усі дії плануються;
3. Передбачається верифікація не лише самого програмного продукту, а й усіх отриманих внутрішніх та зовнішніх даних;
4. Хід виконання робіт легко відстежується, т.к. перехід на наступний етап здійснюється лише після завершення всіх робіт на попередньому.

До негативних рис V-подібної моделі відносять такі недоліки:

1. Недостатня гнучкість моделі;
2. Власне створення програми відбувається на етапі написання коду,

тобто вже в середині процесу розробки;

3. Недостатній аналіз ризиків;

4. Немає роботи з паралельними подіями та можливості динамічного внесення змін.

Застосовується також інкрементна модель, що є методом, в якому проєкт проєктується, реалізується і тестується інкрементного (тобто кожен раз з невеликими доповненнями) до самого закінчення розробки. Це включає в себе як розробку, так і подальшу підтримку продукту. Він вважається закінченим в той час, коли задовольняє всім вимогам.

Інкрементна стратегія розробки є запланованим поліпшенням продукту в процесі його життєвого циклу. При використанні інкрементної моделі життєвого циклу здійснюється процес одноразової часткової реалізації всієї системи та наступного за ним повільного нарощування її функціональних можливостей або характеристик якості в послідовно реалізованих прототипах (інкрементах). Застосування моделі прискорює процес створення системи за рахунок принципу компонування зі стандартних блоків, що дозволяє в цілому зменшити витрати на розробку системи і контролювати зміну вимог.

На ранніх етапах життєвого циклу виконується проєктування системи загалом. На цих етапах визначаються інкременти та реалізовані ними функції. Кожен інкремент потім проходить через інші етапи життєвого циклу. Виконання даних етапів відповідає каскадній моделі життєвого циклу і може бути розподілено згідно з календарним графіком. На рис. 1.3 наведені етапи розробки за використанням інкрементної моделі.

Рис. 1.3 Етапи інкрементної моделі

Перевагами інкрементної моделі вважають такі риси, відповідно роботі [27].

1. Робоча програма виходить на ранній стадії життєвого циклу продукту, одержання функціонального продукту після реалізації кожного інкременту;
2. Гнучкість. Змінити масштаби і вимоги проєкту щодо менш затратно, незначний час розробки кожного інкременту, що спрощує роботу з потребами замовника;
3. Невеликі ітерації спрощують тестування і внесення правок, можливість вирівнювання графіка розподілу робочої сили за допомогою розподілу за часом обсягу роботи над проєктом;
4. Простіше ідентифікувати ризики, впоратися з ними, розподіл ризику між кількома невеликими інкрементами, що скорочує його загальну величину, можливість перегляду ризиків;
5. Кожна ітерація - проста в управлінні контрольна точка проєкту, можливості підтримки постійного прогресу продукту, розробників та технологій у ході виконання проєкту;
6. Запобігання формуванню громіздких переліків вимог, стабілізація вимог під час створення певного інкременту, за рахунок короткої тривалості створення інкременту, включення в процес користувачів та можливості відсунення не важливих змін на наступні інкременти.

Недоліками інкрементної моделі є:

1. Вимоги мають бути сформульовані заздалегідь. Оскільки створення деяких модулів буде завершено значно раніше за інші, виникає потреба в чітко визначених інтерфейсах;
2. Потрібне ретельне планування та проєктування. Керівництво має дбати про розподіл роботи, а технічний персонал повинен дотримуватись субординації у відносинах між співробітниками;
3. Може виникнути тенденція до відтягування розв'язання тяжкої проблеми на майбутнє з метою продемонструвати керівництву успіх, досягнутий на ранніх етапах розробки.

Спіральна модель життєвого циклу розробки програмних засобів і систем поєднує переваги інших видів моделей. Крім того, до неї включено аналіз та управління ризиками, процеси підтримки та управління, передбачені можливості використання Базова концепція спіральної моделі полягає в наступному. Кожен цикл розробки (ітерація) є набором операцій, що відповідає крокам у каскадній

моделі. При цьому враховуються кожен компонент продукту або системи та кожен рівень складності, починаючи із загального аналізу вимог і до програмування кожного компонента. На рис. 1.4 наведено основні етапи спіральної моделі.

Модель поділена на чотири квадранти. Спираючись на роботу [31], у кожен квадрант моделі входять основні та допоміжні дії з розробки продукту чи системи.

У квадранті 1 – аналіз вимог, альтернативних варіантів та обмежень.

У квадранті 2 – оцінка альтернативних варіантів, ідентифікація та вирішення ризиків.

У квадрант 3 - розробка продукту поточного рівня.

У квадранті 4 - планування наступної фази.

Переваги спіральної методології:

1. Найкращий аналіз ризиків;
2. Хороша документація процесу розробки;
3. Гнучкість – можливість внесення змін та додавання нової функціональності навіть на відносно пізніх етапах;
4. Раннє створення робочих прототипів.

До негативних рис моделі слід віднести такі риси:

1. Може бути досить дорогою у використанні;
2. Управління ризиками вимагає залучення висококласних фахівців;
3. Успіх процесу великою мірою залежить від стадії аналізу ризиків;
4. Не підходить для невеликих проєктів.

Однією з розповсюджених моделей є Agile (гнучка методологія) - методологія управління с акцентом на розробці програмного забезпечення. З'явилася вона як результат незастосовності методології Waterfall в рамках складних проєктів.

Хоча ідеї, властиві Agile, вже давно використовуються в сфері розробки програмного забезпечення, формально методологія з'явилася лише в 2001 році, коли кілька представників з IT випустили Agile-маніфест.

Agile повністю протилежна методології Waterfall за підходом і ідеології. Сама назва з англійської мови перекладається як «Гнучкий», а це значить, що в управлінні використовується швидкий і гнучкий підхід. На рис. 1.5 наведено етапи Agile моделі.

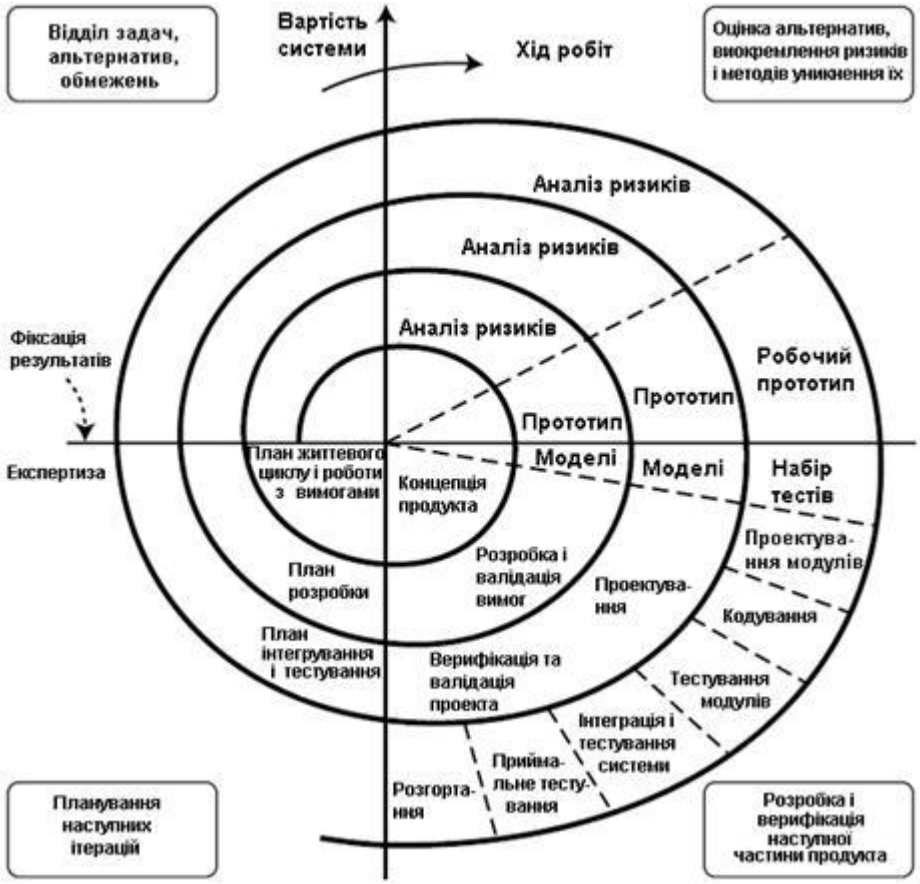

Рис. 1.4. Етапи спіральної моделі

Рис. 1.5. Етапи Agile моделі

Переваги Agile методології було розглянуто у роботі [33]. Вони включають

в себе:

1. Гнучкість і свобода - оскільки тут не потрібно чітко позначати етапи і робити упор на вимогах, у виконавців проєкту з'являється можливість експериментувати і вносити зміни поступово. Саме тому Agile відмінно підходить творчим проєктам;
2. Знижений ризик - методологія Agile передбачає регулярне отримання зворотного зв'язку від зацікавлених учасників і подальше внесення змін. Це значно скорочує ризик провалу проєкту, так як потрібні ресурси залучені в процес.

Недоліками Agile методології є:

1. Слабка увага до документації. У разі виникнення конфліктних ситуацій аргументів на свій захист у виконавця просто не буде;
2. Проблеми із реалізацією комплексних продуктів. Якщо на початку розробки було закладено обмеження — на пізніх етапах виправляти їх буде складно;
3. Імовірність відмови клієнтів у процесі розробки. Якщо замовник не має чіткого розуміння, що він хоче бачити на виході — виконувати його вимоги дуже складно;
4. Вплив ієрархії. Керівники приймають рішення, оскільки всі процеси циклу реалізуються паралельно. Важливість рядових програмістів мінімальна.

Підсумовуючи аналіз даної методології, зауважимо, що:

1. Гнучкість підходу Agile дозволяє адаптувати його до проєктів різного типу;
2. Методологія найкраще працює в випадках коли ви не впевнені, яким повинен бути кінцевий результат, але маєте загальне уявлення про продукт;
3. Коли проєкт потрібно швидко підлаштовувати під зміни;
4. Якщо взаємодія і комунікація - ваші сильні сторони, а планування - немає.

Існує велика кількість методологій управління проєктами, кожна з яких має свої завдання та можливості. Вони є незамінними інструментами, які допоможуть вам у реалізації успішних проєктів із високою цінністю. І хоча кожен підхід дає багато переваг, слід зазначити, що методології не є універсальними. Ефективність методу варіюватиметься залежно від галузі та конкретного проєкту. У свою чергу, їх сильні сторони та переваги працюють найкраще, коли ви застосовуєте їх правильно, дотримуючись типу проєкту та його вимог. Це включає тип команди, зміст проєкту, цілі результатів, а також можливості управління проєктом.

Усім відомо що управління проєктами часто стикається з невизначеностями, які перебувають за межами його розуміння. Це означає, що важливо вибрати підходящу методологію, яка може доповнити та покращити проєкт. Проте на жаль, жодна з цих методологій не може бути ідеальною, кожна з них потребує адаптації під проєкт, кожна має свої ризики при застосуванні. Тому виявлення та аналіз ризиків у тій чи іншій методології, визначення ступеню впливу ризику на проєкт та вибір рішення для усунення ризику знизить кількість несприятливих моментів на проєкті та значно підвищить продуктивність команди.

1.2. Проблеми процесів проєктуванні програмного забезпечення за різними методологіями

Як зазначалось вище, сьогодні існує багато різноманітних методологій побудови процесу розробки ПЗ, і кожна з них має свої переваги та недоліки, сфери застосування, в яких певні з них найбільш ефективні. Всі ці методології мають на меті поліпшення виробничого процесу, який дозволив би найбільш ефективно і якісно виробляти програмні продукти. Проте слід враховувати, що при розробці ПЗ за будь-якої обраної методології існують свої проблеми, з якими зіштовхнеться команда під час роботи. Дуже важливо заздалегідь знати про ці проблеми і спиратися на них при виборі методології для конкретного проєкту. Це допоможе вибрати найбільш оптимальний варіант методології розробки програмного забезпечення для вашого випадку. Згідно з джерелом [15], існують загальні проблеми розробки програмного забезпечення, які були визначені для наступних видів методологій.

Зокрема, проблемами при розробці програмного забезпечення за методологією Waterfall є така низка проблем.

Підвищений ризик якщо ви виявите помилку або вам знадобиться внести зміни, доведеться починати проєкт спочатку, це великий ризик. А це означає, що ви і зовсім можете не завершити проєкт вчасно. Висока ймовірність виявлення критичних проблем вже на завершальному етапі розробки, причому їхнє усунення на етапі готового продукту обходиться занадто дорого.

Складність першого етапу є великою проблемою - весь підхід Waterfall залежить від того, наскільки правильно ви зрозумієте і проаналізуєте вимоги. Якщо вам не вдасться зробити це або якщо вимоги зміняться, доведеться починати спочатку. Тому ця методологія управління не підходить складним довгостроковим

проєктам. Витрати часу на ведення докладної документації, яка, до того ж, може бути не завжди зрозумілою замовнику, та викликати у нього питання. Необхідні кваліфіковані бізнес-аналітики, здатні сформулювати прийнятне для продуктивної роботи технічного завдання, відсутня можливість для маневру, якщо в процесі розробки з'ясувалося, що продукт не відповідає вимогам ринку.

Проблеми при розробці програмного забезпечення за V-образною методологією наведено нижче.

В основі моделі лежить послідовна лінійна структура, у результаті кожна спроба повернутися на одну або дві фази назад. Це є великою проблемою тому, щоб виправити будь-яку проблему або недолік, призведе до значного збільшення витрат і збою в графіку. А саме для кожної фази створюються результативні дані, які після його завершення вважаються замороженими. Також важливим фактором є те, що модель не розрахована на динамічні зміни в вимогах протягом усього життєвого циклу, так як отримані дані "заморожуються", всі вимоги повинні бути відомі на початку життєвого циклу. Через це виникає необхідність у жорсткому управлінні та контролі, оскільки в моделі не передбачена можливість модифікації вимог. Модель заснована на документації, отже, кількість документів може бути надмірною. В результаті цієї проблеми користувачі не можуть переконатися як розроблений продукт до закінчення всього процесу розробки. Вони не можуть оцінити якість, якщо не можна побачити готовий продукт розробки. Тобто у користувача немає можливості поступово звикнути до системи. Процес навчання відбувається наприкінці життєвого циклу, коли вже запущено в експлуатацію.

Кожна фаза є передумовою для виконання наступних дій, що перетворює такий метод на ризикований вибір для систем, що не мають аналогів, оскільки він не піддається гнучкому моделюванню. А саме для кожної фази створюються результативні дані, які після його завершення вважаються замороженими. Також важливим фактором є те, що модель не розрахована на динамічні зміни в вимогах протягом усього життєвого циклу, так як отримані дані "заморожуються", всі вимоги повинні бути відомі на початку життєвого циклу. Через це виникає необхідність у жорсткому управлінні та контролі, оскільки в моделі не передбачена можливість модифікації вимог. Модель заснована на документації, отже, кількість документів може бути надмірною. Через це може витрачатись занадто багато часу на ведення документації. Також слід враховувати що весь програмний продукт розробляється за один раз та немає можливості розбити систему на частини.

Проблеми при розробці програмного забезпечення за інкрементною методологією розглянуто нижче.

Через те що кожна фаза ітерації нерухома з'являється необхідність чітко визначених інтерфейсів між модулями. Ця проблема пов'язана з різними термінами їх створення та складністю формального аналізу та перевірки окремих інкрементів.

Також при розробці програмного забезпечення за використанням інкрементної методології слід враховувати що можуть виникнути проблеми щодо архітектури системи, так як не всі вимоги зібрані заздалегідь для всього життєвого циклу програмного забезпечення, непередбаченість ітерацій у межах кожного інкременту моделі, також є можливість змін у технологіях робіт, що може порушити графік робіт.

Інкрементна модель використовується вкупі з чіткими і зрозумілими вимогами, які впроваджуються пофазно. Існує можливість поточної зміни вимог до системи, які вже реалізовані у попередніх інкрементах. Це може негативно вплинути на розробку продукту та терміни закінчення роботи над продуктом. Це додає необхідність хорошого планування та проєктування, грамотного розподілу роботи. Використання на етапі аналізу загальних цілей замість повністю сформульованих вимог може виявитися незручним для керівництва.

Проблеми при розробці програмного забезпечення за спіральною методологією наведено нижче.

Основна проблема спірального циклу – це визначення моменту переходу наступного етапу. Для її вирішення необхідно запровадити тимчасові обмеження на кожен із етапів життєвого циклу. Інакше процес розробки може перетворитися на нескінченне вдосконалення вже зробленого. При такому підході корисно дотримуватися принципу «краще ворог хорошого». Тому завершення ітерації має виконуватися строго відповідно до плану, навіть якщо не вся запланована робота закінчена. Планування робіт зазвичай проводиться на основі статистичних даних, отриманих у попередніх проєктах, та особистого досвіду розробників.

Іншою проблемою є висока вартість моделі за рахунок вартості та додаткових часових витрат на планування, визначення цілей, виконання аналізу ризиків та прототипування під час проходження кожного циклу спіралі. А саме невиправдано висока вартість моделі для проєктів, що мають низький рівень ризику або невеликі розміри.

Проблеми при розробці програмного забезпечення за методологією Agile

розглянуто нижче.

Для деяких програмних продуктів розробники не можуть повною мірою кількісно оцінити необхідні зусилля, особливо на початку життєвого циклу розробки великих продуктів. Команди, які мають досвіду гнучкої розробки, бояться цих невідомих. Цей страх призводить до розчарування, поганих практик і часто до поганих рішень. Більше регламентований каскадний процес (або процес водоспаду) дозволяє легко визначити кількість зусиль, часу і вартості постачання кінцевого продукту.

Оскільки вимоги до програмного забезпечення уточнюються якраз під час розробки, документація не надто докладна. Це проблема. Це означає, що якщо до команди приєдналися нові члени, вони не знатимуть подробиць, деяких особливостей або як вони мають діяти. Це створює непорозуміння та труднощі. Цей метод не вимагає детального планування для початку роботи та передбачає, що потреби замовника постійно змінюються. Такі недостатні вихідні дані можуть обмежити використання Agile. Потім, якщо зворотний зв'язок замовника не зрозумілий, розробник може сфокусуватися на розробці у неправильному напрямку. Крім того, є небезпека неконтрольованого розширення меж проєкту, і продукт, що постійно змінюється, стає нескінченним.

Проаналізувавши проблеми кожної моделі розробки програмного забезпечення, можна зробити висновок, що застосовність тієї чи іншої моделі розробки програмного забезпечення суттєво залежить від характеру вимог, що висуваються до проєктованої системи. Наприклад, якщо вимоги не можуть бути визначені заздалегідь, а в ході робіт часто змінюватимуться, то найбільш прийнятною є модель прототипування і спіральна модель.

Оскільки при використанні низки моделей успіх реалізації проєкту залежить від ступеня злагодженості роботи єдиної команди розробників та користувачів, то вже на початкових етапах роботи, обираючи модель життєвого циклу, слід отримати повне уявлення про характеристики колективу користувачів як комплексний фактор, що впливає на вибір моделі.

Найважливішу роль при виборі моделі життєвого циклу відіграють тип передбачуваного проєкту та ризики, пов'язані з виконанням проєкту. З наведених міркувань випливає, що процедура вибору моделі життєвого циклу повинна здійснюватися з урахуванням розгляду не окремих критеріїв, які комплексу.

Для адекватної оцінки цієї ситуації виникає потреба у залученні експертів та подальшої обробки отриманих експертних оцінок, і, відповідно, представляється

актуальним завданням створення відповідної інформаційної системи, яка б у комплексі охоплювала питання застосування різних моделей життєвого циклу в конкретних ситуаціях, охарактеризованих експертами.

Аналізуючи проблеми кожного методу можна обрати той, який найбільш підходить для проєкту. У табл. 1.1 наведена застосовність різних моделей розробки програмного забезпечення в залежності від характеристик набору вимог до проєктованої системи.

Таблиця 1.1

Характеристика набору вимог моделей розробки програмного забезпечення

Характеристика набору вимог	Модель				
	Каскадна	V-подібна	Інкрементна	Спіральна	Agile
Чи є вимоги легко визначними та/або добре відомими?	Так	Так	Ні	Ні	Ні
Чи можуть бути вимоги наперед визначені?	Так	Так	Так	Ні	Так
Чи часто змінюватимуться вимоги?	Ні	Ні	Ні	Так	Так
Чи потрібно демонструвати вимоги з метою їхнього визначення?	Ні	Ні	Ні	Так	Так
Чи потрібна для демонстрації можливостей перевірка концепції?	Ні	Ні	Ні	Так	Так
Чи вимоги відображатимуть складність системи?	Ні	Ні	Так	Так	Ні
Чи відображають вимоги ранньому етапі функціональні	Ні	Ні	Так	Так	Так

властивості системи?					

Часто в гонитві за прибутком ключові проблеми розробки програмного забезпечення йдуть на другий план — терміни перемагають у боротьбі за високу якість програмних рішень. Швидкості немає місця бути в процесі проєктування та планування такого складного підходу до розробки програмного забезпечення. Як вже було зазначено, існує велика кількість ризиків, з якими може зіткнутися команда, незалежно від того, яка методологія обрана, яка особливість проєкту, яка кількість співробітників і навіть які терміни завершення проєкту. Однією із складових удосконалення процесу проєктування ПЗ є ризик менеджмент, який дозволяє заздалегідь спланувати роботу в команді, ґрунтуючись на передбачуваних ризиках і контролювати процес розробки, ґрунтуючись на раніше виявлених проблемах. Таку процедуру найчастіше виконує менеджер проєкту, і це є невід'ємною частиною процесу розробки програмного забезпечення.

1.3. Ризик менеджмент як складова удосконалення процесу проєктування програмного забезпечення

В інженерії програмного забезпечення існують ризики його розроблення незалежно від того, як проєкт та його реалізації були підкріплені фінансово і матеріально-технічним забезпеченням, нормативно-правовими актами і професійною підтримкою його виконавців. Зазвичай, під ризиком в інженерії програмного забезпечення розуміють ситуацію, яка може призвести до втрати очікуваного прибутку, або подію, яка може поставити під загрозу успіх реалізації програмного проєкту.

Управління ризиками – невід'ємна складова ефективного управління будь-яким програмним проєктом, керівник якого повинен ефективно планувати, управляти та здійснювати постійний контроль за всіма етапами виконання завдань проєкту. Керівник проєкту має вміти прогнозувати потенційні прибутки від успішної реалізації проєкту та оцінювати можливі втрати від прийняття необгрунтованих рішень, а також прагнути до зменшення збитків при настанні ризикових подій.

На сьогодні більшість IT-компанії, зазвичай, стають конкурентоспроможними тільки за рахунок інноваційної діяльності, яка за своєю сутністю пов'язана з різними ризиками, тобто ймовірністю виникнення збитків або недоотримання прибутків порівняно з прогнозованими. Тому ризик є одночасно як причиною можливих збитків, так і джерелом потенційних прибутків. Також ризик – важлива складова процесу управління, неврахування якого призводить до вироблення, аналізу й прийняття необгрунтованих і малоефективних управлінських рішень. Основне завдання управління ризиками – не відмовитись від ризику як такого взагалі, а приймати ризикові рішення, ґрунтуючись на об'єктивних критеріях і допустимих втратах. Прийняті керівником проєкту ризик-орієнтовані рішення часто призводять до більш ефективної реалізації програмних проєктів, від яких отримують свою вигоду як замовники і розробники програмного забезпечення, так і його безпосередні користувачі.

Проблеми, які з'являються під час розроблення програмного забезпечення в різний час були розглянуті в багатьох наукових публікаціях, наприклад [4; 8]. Проте, у своїх дослідженнях і публікаціях автори мало приділяють уваги аналізу причин появи ризиків, які необхідно враховувати у процесі розроблення програмного забезпечення.

Питання управління ризиками розроблення програмного забезпечення широко розглянуто в різних наукових джерелах, починаючи з нормативних документів Міністерства оборони США, в роботах, написаних з використанням цих даних [7], і завершуючи виданнями для широкого загалу. Наприклад, в роботі [8] процес організації ризик-менеджменту розглянуто як проєкт всередині програмного проєкту. Автори стверджують, що цей проєкт має свої етапи життєвого циклу і управління ним потрібно проводити паралельно з управління програмним проєктом. У роботі хоча матеріал й викладено дещо спрощено, проте його головна ідея – ризик-менеджмент складна, але необхідна складова життєвого циклу будь-якого програмного проєкту.

Відповідно до джерела [7], під ризиками розуміють появу негативних ситуацій імовірнісного характеру, які істотно впливають на результати реалізації програмного проєкту, відображають втрати або збитки від недостатньої якості виконуваних процедур чи завдань проєкту. Ці втрати та збитки викликані неточностями при формуванні наборів вимог до програмного забезпечення, недоліками обґрунтування термінів і бюджету проєкту, не виявленими дефектами на етапах конструювання програмного забезпечення та тестування його

компонент, а також під час неправильної його експлуатації. Тут важливо розуміти, що ризик – це ймовірна подія, яка може відбутися з різними наслідками, а може й оминути від напасти програмний проєкт чи навіть продукти його реалізації.

З найбільш поширеної точки зору, кожен ризик у певному сенсі пропорційний як очікуваним втратам, які можуть бути заподіяні ризиковою подією, так і ймовірністю цієї події. Відмінності у визначеннях ризику залежить від контексту втрат, їх оцінки та вимірі.

Ризики мають відношення до майбутнього, їх слід відрізняти від проблем та труднощів, які мають місце нині. Мета управління ризиками — максимізувати їх позитивний вплив, але при цьому мінімізувати пов'язані з ними негативні фактори. Управління ризиками допомагає досягти компромісів між небезпеками та можливостями. Новий інструментарій, нові технології, вимоги користувачів, що зростають, зростають загрози для інформаційної безпеки, плинність кадрів — все це здатне спричинити зміни в IT-проєкті і змусити приймати рішення в умовах невизначеності.

Ризики виникнення потенційних загроз і небезпек існують практично в усіх програмних проєктах, але не завжди вони відбуваються та наносять шкоду програмному забезпеченню. Зазвичай, проявлена небезпека перетворюється на проблему як поточну, так і майбутню. Прогнозування потенційних загроз і небезпек у багатьох випадках – це передбачення прояву деяких ризикових подій, що, як правило, мають негативно вплинути на хід реалізації програмного проєкту та на його остаточні результати – продукти проєкту. У такому контексті ризик виникнення потенційної небезпеки розглядають як прояв деякої випадкової події, яка має ймовірнісний характер. На рис. 1.6 відображено план управління ризиками, який буде розглянуто для удосконалення процесу проєктування програмного забезпечення.

Рис. 1.6. Управління ризиками проєкту

Хоча ризики розроблення програмного забезпечення й поділено на декілька основних категорій, однак, у кожному конкретному випадку можуть бути додані й інші типи ризиків, які не розглянуто у цьому дослідженні. Отже, до основних категорій ризиків відповідно до джерела [7] належать:

Технологічні ризики, пов'язані з незнанням технологій, які заплановано використовувати працівниками для розроблення програмного забезпечення, або з низькою апробацією й відпрацьованістю цих технології в колективі виконавців проєкту.

Ризики наявних суперечностей у вимогах до програмного забезпечення, пов'язані з виявленням суперечностей у вимогах замовника на етапі декомпозиції користувацьких вимог у системні вимоги, або на етапі конструювання програмного коду чи інтеграції продуктів проєкту.

Ризики, пов'язані з низькою кваліфікацією персоналу, тобто керівник проєкту повинен знати можливості своїх працівників і, за потреби, організувати їх навчання ще до початку реалізації програмного проєкту, щоб не витрачати час на ліквідацію помилок у вже розробленому програмному забезпеченню.

Ризики поганої взаємодії між замовником програмного забезпечення і його виконавцем, пов'язані з відсутністю комунікації між їхніми керівниками або їх представниками. Недостатнє обговорення користувацьких вимог до програмного забезпечення або його архітектури може негативно позначитися на його майбутньому функціоналі та, як наслідок, на якість самого програмного забезпечення.

Ризики неефективного планування етапів реалізації проєкту – можуть бути пов'язані з відсутністю навиків планування процесу виконання завдань проєкту як його керівником, так і їх виконавцями, якщо вони готують інформацію про терміни виконання багатьох робіт і завдань проєкту.

Ризики недостатності системи контролю за реалізацією завдань проєкту – обумовлені значною кількістю особливостей у області проєктного менеджменту при розробленні програмного забезпечення, коли складно передбачити і врахувати всі можливі ситуації, які можуть виникнути під час реалізації станів програмного проєкту.

Ризики появи нових користувацьких вимог до програмного забезпечення – виникають в процесі його розроблення, коли у замовника з'являються все нові й нові вимоги, які відсувають терміни виконання конкретних завдань проєкту і ускладнюють можливість оцінювати їх виконання.

Ризики наявних суперечностей у вимогах до програмного забезпечення, пов'язані з виявленням суперечностей у вимогах замовника на етапі декомпозиції користувацьких вимог у системні вимоги, або на етапі конструювання програмного коду чи інтеграції продуктів проєкту.

Ризики неправильно сформульованих системних вимог – виникають тоді, коли на самому початку реалізації проєкту були некоректно сформульовані характеристики цільової системи, для якої розробляють програмне забезпечення: програмне оточення (операційна система, встановлені компоненти, сервіси і т.п.) або вимоги до апаратної частини (частота процесора, об'єм жорсткого диска, обсяг оперативної пам'яті і т.п.);

Ризики нездатності команди впоратися зі складністю розроблення програмного забезпечення – виникають тоді, коли програмне забезпечення може бути настільки складним, що команда його розробників просто не може з ним впоратися;

Ризик низької продуктивності праці команди розробників програмного забезпечення – обумовлені значною тривалістю реалізації етапів програмного проєкту. Часто на самому початку реалізації проєкту це призводить до значних втрат часу, який складно буде надолужити на наступних етапах його реалізації. При цьому доводиться або переносити терміни виконання завдань проєкту, або працювати в більш динамічному режимі на дещо пізніх етапах реалізації проєкту;

Ризик зміни працівників програмного проєкту – виникають тоді, коли проєкт покидають ключові його виконавці, які максимально володіють

інформацією щодо особливостей етапів його реалізації;

Ризики неправильно сформульованих системних вимог – виникають тоді, коли на самому початку реалізації проєкту були некоректно сформульовані характеристики цільової системи, для якої розробляють програмне забезпечення: програмне оточення

Ризики розкрадання продуктів проєкту – виникають тоді, коли розробники програмного забезпечення, йдучи з проєкту, забирають з собою деякі його складові, що розроблялися ними, і, трохи модифікувавши їх, можуть продати або використовувати в інших проєктах, наприклад, у конкурентів;

Ризики порушення закону про авторське право – виникають тоді, коли виконавці проєкту використовують без відома його керівника чужі продукти проєкту (програмні коди, алгоритми або бібліотеки тощо), які захищені законом про авторське право, але їх офіційно не було придбано або їх використання не узгоджене з автором.

До спекулятивних ризиків розроблення програмного забезпечення віднести ризики фінансових обмежень – можуть виникнути як з вини керівника проєкту, який планував його терміни і бюджет реалізації, так і з інших причин, наприклад, низької платоспроможності замовника програмного забезпечення;

На рис. 1.7 наведена коротка класифікація ризиків проєкту відповідно до джерела [20].

Рис. 1.7. Класифікація ризиків

Сьогодні у століття інформаційних технологій використання різних інформаційних систем набуває масового характеру. У той же час некеровані системи можуть спричинити дуже серйозні ризики, з якими впоратися самостійно практично неможливо. Ризик менеджмент IT рішень - це той варіант, коли впоратися з труднощами, що виникли, можна структуровано і грамотно. У цьому враховується все: отримана інформація з документів, висновки результатів інтерв'ювання, співбесід, нарад, тренінгів, аналізів фінансової складової компанії. Тому саме ризик-менеджмент являється невід'ємною складовою для удосконалення процесу проєктування програмного забезпечення. Завдяки ризик менеджменту можна досягти це максимізації ймовірності сприятливих умов при проєктуванні та мінімізація ймовірності виникнення ризиків.

Висновки до розділу 1

У розділі 1 було проаналізовано сучасні підходи до проєктування програмного забезпечення. Були виявлені переваги та недоліки кожного з цих методів. Встановлено, що усі методології мають загальні проблеми при

проєктуванні програмного забезпечення, а саме підвищений ризик якщо знадобиться внести зміни, необхідність у жорсткому управлінні та контролі, проблеми щодо проєктування архітектури системи заздалегідь, підвищення вартості проєкту за рахунок додаткових часів на планування, прототипування, документування та визначення цілей проєкту.

Було розглянуто сутність поняття ризик, наведено його класифікацію, а саме виділено зовнішні та внутрішні, технічні та нетехнічні, передбачувані та непередбачувані, юридичні, управлінські, фінансові, маркетингові та ризики персоналу.

2. ШЛЯХИ МІНІМІЗАЦІЇ РИЗИКІВ ПРИ ПРОЄКТУВАННІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Сучасні методи ідентифікації ризиків

Перш за все, для оцінки ризику і прийняття пов'язаних з ним рішень необхідно зібрати вихідну інформацію про суб'єкта, що несе ризик. Цей основний етап називається ідентифікацією ризику, яка збирає інформацію про структуру об'єкта і виявляє небезпеки.

Ідентифікація – це визначення ризиків, засноване на визначенні факторів, що їх продукують, а також документальне оформлення параметрів цих ризиків. Якісний та кількісний аналіз причин виникнення та можливості негативних наслідків необхідні для формування оціночної процедури. Планування реагування на знайдені ризики передбачає створення комплексу заходів, спрямованих на зниження негативного впливу ризиків на параметри та результати проєкту. Але чільне місце у цій системі займають саме моніторинг ризиків та контроль над ними – вони здійснюються протягом усього життєвого циклу проєкту.

Існує безліч способів, які допоможуть отримати інформацію про характеристики індивідуальних ризиків, притаманних тому чи іншому виду діяльності. Тому для вирішення цієї проблеми рекомендується використовувати набір методів. Задокументовані ризики та їх характеристики, відображені в реєстрі ризиків, дозволяють досліджувати причини та можливі суміжні ризики, які можуть взаємовпливи та взаємозамінювати один одного залежно від параметрів оточення діяльності. Розмір реєстру ризиків буде залежати від масштабу діяльності, але якщо організація має на меті підвищення якості своєї діяльності та створення якомога якіснішого продукту або послуги, то реєстр ризиків повинен бути максимально повним, незалежно від ймовірності та значення подій ризику.

Розуміючи реалії ситуації в галузі ІТ, абсолютно очевидно, що керувати всіма можливими ризиками є малоймовірним, оскільки це потребує великих фінансових та кадрових витрат. Тому обов'язковою умовою для реєстру ризиків є наявність у ньому пріоритетів.

Пріоритет ризику - це параметр, якими ідентифікується найбільш важливі та значущі з них на даний момент часу. Саме важливість коректного визначення

пріоритетів ризиків, серед їх загального числа, може забезпечити надійні «тили» значущих процесів та проєктів, що проводяться в організації.

Тому важливою частиною організації діяльності у цій сфері є створення спеціальних програм з управління та виявлення нових ризиків. Такі програми реалізуються ризик-менеджером департаменту управління ризиками компанії. Яким би не був план управління ризиками, він має свій власний бюджет і бюджет, так що витрати на виявлення ризиків не перевищують заподіяну ними шкоду.

Існує безліч способів класифікації та, відповідно, ідентифікації ризиків проєкту. Цей матеріал детально розглянуто у джерелі [25]. Частина з них є строго формальними та математичними (наприклад Метод Монте-Карло), частина з них менш формальна та доступна для застосування у широких аудиторіях фахівців, незалежно від рівня попередньої спеціальної підготовки до діяльності з аналізу ризиків (Метод Brainstorming), деякі з них застосовні у всіх напрямках під час роботи в IT галузі, а окремі, є вузькоспрямований для конкретних процесів та доменів.

На вибір конкретного методу або комплексу методів роботи з ризиками впливають різні фактори, такі як доступність кваліфікованих ресурсів, характер і ступінь невизначеності даних та інформації, складність методу його застосування. Перед тим, як вибрати той чи інший метод, необхідно провести дослідження, метою якого є обґрунтування вибору конкретних методів ідентифікації ризику із зазначенням їх прийнятності, придатності та застосування у заданих умовах функціонування. Необхідно забезпечити відповідність використовуваних методів та вихідних даних для об'єднання результатів різних аналізів при роботі над великими проєктами.

Розглянемо найбільш популярні та добре себе зарекомендували на практиці методи, які, при невеликому засвоєнні методичного матеріалу, зможе застосувати кожен спеціаліст у своїй роботі. Нижче описані методи були виявлені та розглянуті у джерелах [28, 29].

1. Brainstorming

Мозковий штурм є найпростішим і найпоширенішим методом ідентифікації, який існує досить. Метою мозкового штурму є створення докладного списку всіх можливих, хай навіть найфантастичніших і малоймовірних, на перший погляд, ризиків проєкту чи процесу. Важливо, що на зборах, присвячених самому Мозговому штурму, не розробляється реєстр ризиків. Мета зборів – розробити список ризиків. У зборах бере участь від 5 до 12 осіб (члени команди з

процесу/проєкту, схильному до ризикового впливу, запрошені експерти з розглянутої області та суміжних областей, зацікавлені стейкхолдери. Учасники зборів виявляють та ідентифікують усі можливі ризики, які, на їхню власну думку при цьому не допускається обговорення висунутих ідей, а група експертів озвучує будь-які ідеї, які надалі повинні бути детально і ретельно проаналізовані, структуровані з подальшим відображенням ризиків.

Плюси методу – відносна швидкість, реалізованість, легка досяжність кінцевого результату, зниження ефекту придушення членів групи провідною особистістю. Недоліком порівняно з методом мозкового штурму є нижча соціальна орієнтованість методики карток кроуфорду. Методики цілком життєздатні і мають, за умови правильного використання та модерування, практичну значимість і цінність.

2. Метод Delphi

Другий за популярністю метод, який використовується під час роботи з ризиками, метод Delphi. Більш детально його було розглянуто у роботі [33]. Він був розроблений під час розпалу холодної війни, у 50-х роках у США, групою експертів, які представляють одну з корпорацій, що працюють на урядові структури. Цей метод дозволяє проаналізувати ризики кілька разів, систематизувати їх, автоматично відчуваючи незначні другого план. Консенсус та список ризиків виходить через кілька ітерацій цього процесу

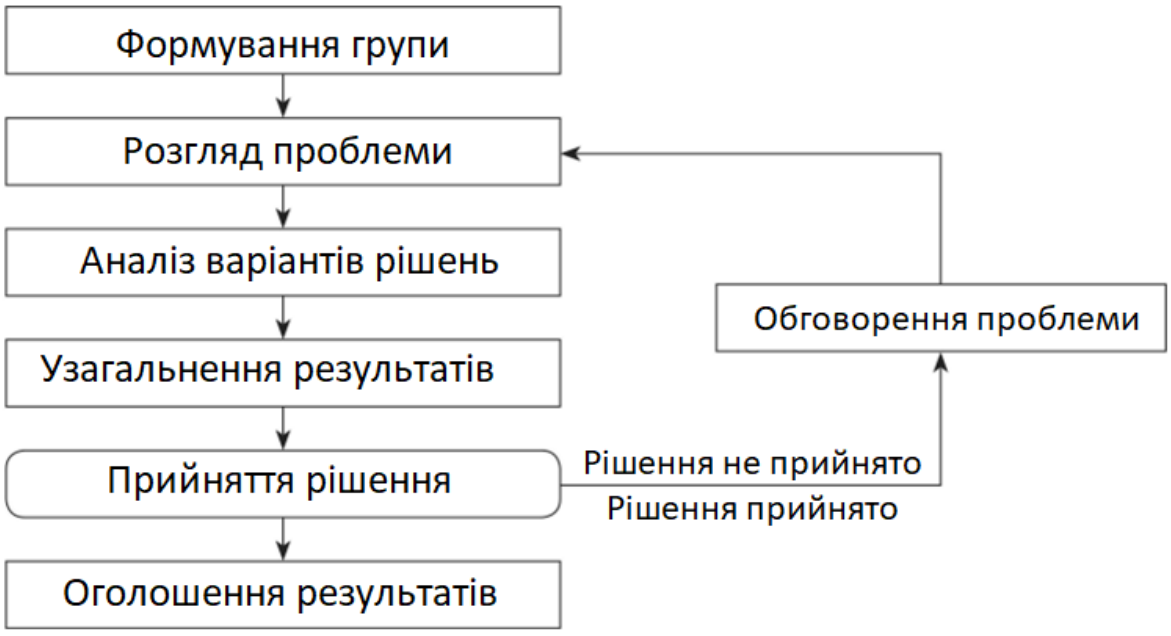

Рис. 2.1. Метод Delphi

У методі Deiphi виключається тиск з боку колег і страх незручного становища при висловлюванні ідеї. Основними переваги способу є виключення можливості домінування однієї особи, спосіб може проводитися дистанційно, наприклад, через електронну пошту, нівелюється можливість «не зрілої» оцінки. Недоліки методу: вимагає участі кожного члена групи, займає багато часу, високе завантаження лягатиме на плечі ведучого та адміністративний персонал. На рис. 2.1 схематично зображена процедура ідентифікації ризиків за методом Delphi.

3. Ідентифікація основних причин

Ідентифікація основних причин (ІОП) - це окремий метод, має чітко сформульований алгоритм (проти методами Brainstorming чи Delphi). ІЗП це комплексний підхід, що використовується при ідентифікації ризиків.

Суть даного методу, як описано у джерелі [17], полягає у докладному розгляді всіх можливих ризиків, які, за своєю суттю, є наслідком певної діяльності/ї та побудовою причинно-наслідкових зв'язків. За допомогою зафіксованих закономірностей і стає можливим виявити основні та головні причини ризиків, галузі та активності, в яких вони виникають, враховуючи різноманітні суміжні процеси, які впливають на виникнення ризиків. Після того, як причини виявлені та зафіксовані, необхідно ухвалити рішення про те, що і яким чином необхідно коригувати та виправляти.

Безперечною перевагою методу ідентифікації основних причин є можливість його проведення без додаткового залучення дорогого ресурсу додаткових експертів, окремо взятим фахівцем, що робить цей метод менш «експертним» (з точки зору кількості учасників) порівняно з розглянутими раніше методами, але не менш ефективним, та швидким.

Недоліками можна вважати необхідність наявності певної документарної бази, на основі якої можна було б побудувати аналіз ідентифікації та виявлення ризиків.

4. SWOT аналіз

SWOT (SWOT, акронім - Strengths [переваги], Weaknesses [недоліки], Opportunities [можливості] та Threats [загрози]). Термін "SWOT" введений у Гарварді в 1963 році, професором економіки Кеннетом Ендрюсом.

Мета проведення цього виду аналізу – оцінити можливості та оточення «ризикового» проєкту чи процесу. На сьогоднішній день ця методика набула дуже широкого поширення у різноманітних галузях бізнесу при проведенні консалтингових та управлінських досліджень за рахунок своєї привабливої суб'єктивності та легко інтерпретованості результатів, виконаних конкретними експертами.

Переваги і недоліки - це чинники внутрішнього середовища процесу чи проєкту, що впливають виникнення чи самі породжують ризики. Можливості та загрози - це фактори зовнішнього середовища, що впливають на об'єкт і призводять до можливого виникнення ризиків.

Перевагами цього виду аналізу є: універсальність застосування, гнучкість використання, широта використання, легка адаптованість.

Слабкими сторонами SWOT аналізу: поверхневість факторів, що оцінюються, тільки якісний опис факторів, суб'єктивність.

Завдання SWOT-аналізу — дати структурований опис ситуації, щодо якої потрібно ухвалити будь-яке рішення.

Висновки, зроблені на його основі, носять описовий характер без рекомендацій та розстановки пріоритетів, що призводить до того, що даний аналіз, сам по собі, не можна вважати самодостатнім, а часом навіть дуже шкідливим та «отруйним». На рис. 2.2 наведено принцип SWOT аналізу.

Рис. 2.2. Принцип SWOT аналізу

5. Метод номінальних груп

Метод номінальної групи - це груповий процес, що включає ідентифікацію проблеми, генерація рішень та прийняття рішень. Його можна використовувати в групах різного розміру, які хочуть швидко ухвалити рішення, наприклад, шляхом голосування, але хочуть, щоб думка кожного бралася до уваги. Різниця у методі підрахунку. Спочатку кожен член групи пропонує своє бачення рішення з короткими поясненнями. Потім рішення, що повторюються, видаляються зі списку всіх рішень, і учасники переходять до ранжування рішень: 1, 2, 3, 4 і так далі.

6. Метод експертних оцінок

Цей метод застосовується у випадках відсутності або недостатнього обсягу інформації. Відповідно до джерела [34], він забезпечує виділення певних груп ризиків і отримання оцінки ступеня ризику на підставі знань спеціалістів І науковців, їхнього вміння узагальнювати власний і світовий досвід досліджень із певної проблематики. Ідея і зміст методу полягає у побудові раціональної процедури інтуїтивно-логічного мислення людини з кількісними методами оцінювання і оброблення отриманих результатів. При цьому висновки експертів,

зроблені на базі науково-практичного досвіду, приймаються як вирішення проблеми.

За допомогою методів експертних оцінок розв'язують задачі як якісного аналізу ризиків, такі кількісного. Розрізняють методи індивідуальних і колективних експертних оцінок. Кожний із методів має свої особливості, але спільними для всіх є процедури, які забезпечують їх реалізацію: вибір експертної групи; проведення експертизи; оброблення експертної інформації; верифікація результатів експертизи. Порівняно з іншими методами визначення ступеня ризику методу експертних оцінок більше властивий суб'єктивний характер.

Основна перевага методу експертних оцінок полягає в можливості використовувати досвід експертів для аналізу проєкту та обліку впливу різноманітних якісних чинників.

Перевагою експертного аналізу є відсутність необхідності в точних початкових даних і дорогих програмних засобах, можливість здійснювати оцінку до розрахунку ефективності проєкту, а також простота розрахунку.

Основні недоліки – складність залучення незалежних експертів і суб'єктивність оцінки. Порівняльна характеристика методів ідентифікації ризиків наведена у табл. 2.1.

Таблиця 2.1

Порівняльна характеристика методів ідентифікації ризиків

Метод ідентифікації	Переваги	Недоліки
Мозковий штурм	Сприяє взаємодії усієї команди розробки. Швидкий. Недорогий.	Може проявитися переважання одного члена команди, наприклад ліда чи менеджера проєкту. Можна зосереджуватися тільки в конкретних модулях програми. Можливий відхід від реальних ризиків
Метод Delphi	Може проводитися дистанційно (через електронну пошту або в онлайн зв'язках), що зараз дуже актуально. Виключається проблема ранньої оцінки. Вимагає участі кожного члена команди розробки.	Займає багато часу. Високе завантаження менеджера проєкту чи бізнес аналітика.
Метод номінальних	Зменшується ефект домінуючою особистості. Забезпечує взаємодію	Високе завантаження менеджера проєкту чи бізнес аналітика.

груп	команди. Дає упорядкований список ризиків що можуть виникнути під час розробки.	Потребує багато часу.
Картки Кроуфорда	Швидкий. Легко реалізується. Повинен брати участь кожен у команді. Виробляється велика кількість ідей по реалізації. Можна проводити з групами більше звичайного розміру.	Менше взаємодія між командою розробки програмного забезпечення.
Метод експертних оцінок	Використовується минулий досвід розробки проєкту та проблеми які вже виникали.	Експерт може бути упередженим. Потребує багато часу. Може бути затратно фінансово через те що експерт - людина з досвідом, витрачає час на цю роботу, а час роботи дуже високооплатний.
Контрольні списки	Конкретний і упорядкований Легко використовувати упередженість	Може не містити конкретних елементів для даного проєкту
Метод аналогії	Використовує минулий досвід для виключення проблем в майбутньому подібні проєкти містять багато схожих рис	Потребує багато часу. легко отримати результати, яких не слід цього випадку. Аналогія може бути некоректної

Закінчення таблиці 2.1

Метод SWOT аналізу	Дозволяє виявити всі фактори, що впливають на організацію, а саме дозволяє узагальнити і зіставити дані абсолютно різного типу і призначення.	Відсутність динаміки в часі

2.2. Методи управління ризиками

Після розпізнавання ризику його потрібно проаналізувати, щоб потім обрати правильний план реагування на ризик. Відповідно до джерела [21], існують такі методи аналізу ризиків.

Якісний аналіз ризиків – процес подання якісного аналізу ідентифікації ризиків і визначення ризиків, що вимагають швидкого реагування.

Така оцінка ризиків визначає ступінь важливості ризику й обирає спосіб реагування. Доступність супровідної інформації допомагає легше розставити пріоритети для різних категорій ризиків.

Відповідно до джерела [23], яке було розглянуто для детального аналізу, завдання якісного аналізу ризиків полягає у:

1. Виявленні та ідентифікації можливих видів ризиків;
2. Дослідженні причин виникнення ідентифікованих ризиків і наслідків їх дій;
3. Встановленні потенційних меж окремих видів ризиків;
4. Наданні вартісної оцінки можливих втрат від прояву ризиків;
5. Розробленні системи заходів щодо зменшення та уникнення ризиків.

Також існує кількісний аналіз ризиків, який визначає ймовірність їх виникнення і вплив наслідків ризиків на проєкт, що допомагає групі менеджменту проєкту правильно приймати рішення і уникати невизначеностей. Кількісний аналіз передбачає визначення окремих ризиків та ризику проєкту в цілому у конкретних числових показниках.

Кількісне аналізування проєктних ризиків передбачає:

1. Вибір системи показників для оцінювання ризиків;
2. Обґрунтування і вибір методів кількісного оцінювання ризиків;
3. Формування інформаційної бази для кількісного аналізу ризиків;
4. Побудову економіко-математичних моделей для оцінювання альтернативних варіантів рішень;
5. Вибір підмножини пріоритетних (ефективних, оптимальних) рішень.

Під час кількісного аналізування ризику встановлюється співвідношення між ступенем ризику і очікуваним економічним результатом. На ступінь допустимого ризику впливають параметри економічної системи (власний капітал, основні фонди, обігові кошти, фінансовий стан тощо), причому у момент обґрунтування рішення вони відомі, а також величина очікуваних збитків або прибутків. Очевидно, що чим адекватними є моделі, які описують ризикові ситуації, і досконалі методи визначення кількісних оцінок ризиків, тим меншими стає значення чинника невизначеності. На рис. 2.3 зображена узагальнена блок-схема комплексної оцінки ризиків.

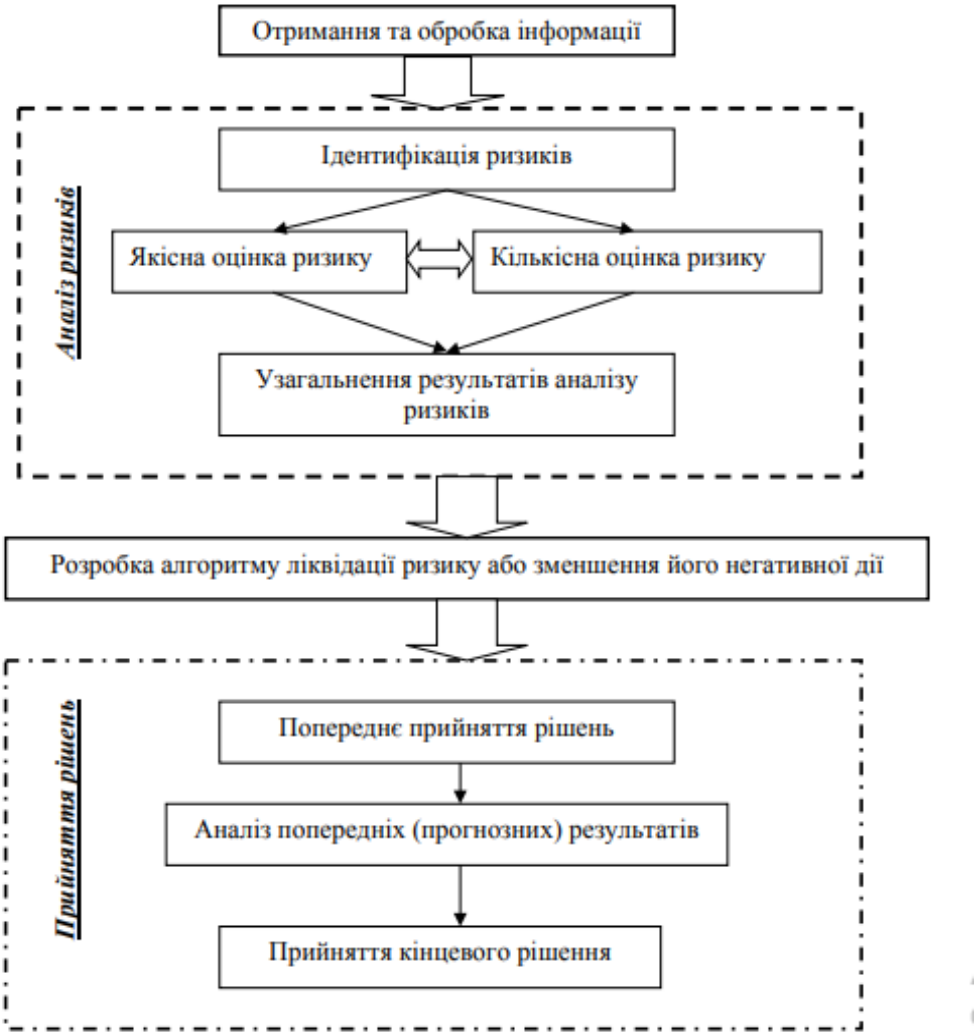

Рис. 2.3. Блок-схема комплексної оцінки ризиків

Існує багато методів для аналізу ризиків. Відповідно до джерела [18], розглянемо сучасні методи кількісного аналізу ризиків:

1. Визначення параметрів граничного рівня.

Застосовується для визначення групи параметрів (показників), що характеризують ступінь стійкості проєкту щодо можливих змін його реалізації. Граничним вважається таке значення параметра, при якому чистий прибуток дорівнює нулю. Основний показник групи – точка беззбитковості.

Значення точки беззбитковості обов'язково розраховується під час створення нового підприємства, модернізації виробничих потужностей, запровадження створення нового виду продукції. Значення точки беззбитковості обов'язково розраховується під час створення нового підприємства, модернізації

виробничих потужностей, запровадження створення нового виду продукції. Головна мета розрахунку точки беззбитковості полягає у знаходженні обсягів реалізації, необхідних для відшкодування витрат, а основою для аналізу беззбитковості є дані бухгалтерської звітності.

2. Оптимізаційні методи.

Забезпечують пошук найкращих (оптимальних) варіантів функціонування економічних систем і процесів при раціональному використанні обмежених ресурсів, тобто таких значень керуючих факторів, при яких досягається максимум (мінімум) функції мети (цільової функції) в області допустимих рішень. Показниками функції мети обсяги виробництва та реалізації продукції, витрати на виробництво, прибуток, рівень задоволення потреб тощо.

Область допустимих рішень формується з урахуванням фінансових, трудових, технологічних, ринкових обмежень витрат ресурсів. Розрізняють завдання оптимізації з однією та кількома функціями мети. Методи оптимізації застосовують при відборі проєктів з безлічі, що забезпечують певний рівень ризику, що не перевищує встановленого рівня.

3. Теорії ігор та статистичних рішень.

Теорія ігор використовується для дослідження конфліктних ситуацій, в яких зіштовхуються супротивні сторони, кожна з яких переслідує свою мету, причому результат довільної дії кожної із сторін залежить від того, які заходи буде вживати супротивник.

Теорія ігор — це теорія математичного моделювання прийняття рішень в умовах конфлікту, завдання якої полягає у виробленні ефективної поведінки учасників конфлікту. Якщо невизначеність пов'язана не із свідомими діями супротивника, а з не проінформованістю про умови, в яких потрібно приймати рішення, то використовують апарат теорії статистичних рішень (ігри з природою).

4. Метод ставки дисконту з поправкою на ризик.

Цей метод дозволяє, збільшуючи безризикову ставку відсотку на величину надбавки за ризик, врахувати фактори ризику при розрахунку ефективності проєкту. Так, у випадку інноваційних проєктів надбавка за ризик може досягати 10-20% .

Його ідея - коригування базової норми дисконту, яка вважається безризиковою або мінімально прийнятною (наприклад, ставка дохідності по державним цінним паперам, гранична або середня вартість капіталу для підприємства).

Коригування здійснюється шляхом додавання обсягу необхідної премії за ризик, після чого виконується розрахунок критерії вефективності проєкту (NPV,IRR) за знову отриманою таким чином нормою. Рішення приймається згідно з правилом обраного критерію. У загальному випадку чим більше ризик, що асоціюється з проєктом, тим вище має бути обсяг премії, яка може визначатися за внутрішньо фірмовими процедурами, експертним шляхом або за формальними методиками.

5. Дерево рішень.

Для побудови "дерева рішень" аналітик визначає склад і тривалість фаз життєвого циклу проєкту; виділяє ключові події, які можуть вплинути на подальший розвиток проєкту, та можливий час їх настання; аналітик обирає всі можливі рішення, які можуть бути прийнятими в результаті настання кожної із подій, та визначає ймовірність кожного із них; останнім етапом аналізу даних для побудови "дерева рішень" є встановлення вартості кожного етапу здійснення проєкту (вартості робіт між ключовими подіями) в поточних цінах.

На основі даних будується "дерево рішень". Його вузли представляють ключові події, а стрілки, що їх поєднують – перелік робіт по реалізації проєкту. Крім того, приводиться інформація відносно часу, вартості робіт і ймовірності розвитку того чи іншого рішення.

В результаті побудови дерева рішень визначається ймовірність кожного сценарію розвитку проєкту, а також чистий приведений дохід (ЧПД) по кожному сценарію та по проєкту в цілому.

6. Метод достовірних еквівалентів.

Метод достовірних еквівалентів (коефіцієнтів достовірності) на відміну від попереднього дозволяє здійснити коригування не норми дисконту, а очікуваних значень потоку – шляхом введення спеціально знищуючих коефіцієнтів αt для кожного періоду реалізації проєкту. Цей метод дуже гарно розкрито у роботі [28]. Отже, єдиний метод, який би повністю відповідав вимогам до оцінки проєктних ризиків, виокремити важко і потрібно використовувати їх комбінацію. Основними методами в рамках кількісного аналізу проєктів є: аналіз чутливості проєкту, сценарний підхід, імітаційне моделювання.

Вибір методу залежить від багатьох факторів, як-от: масштабу проєкту, його складності, можливості використання інформації, оточення, у якому реалізується інвестиційний проєкт та ін. Однак наявність характеристик певних методів оцінки ризиків, виділення їх переваг та недоліків дозволяє орієнтуватися та вибрати

найбільш привабливий для певного інвестиційного проєкту. Керівництво функціональних підрозділів може використовувати дослідження з метою ідентифікації ризиків діяльності підприємства та інвестиційних проєктів.

Кінцева мета аналізу ризиків полягає в розробці заходів, які дозволяють знизити ризик проєкту, а також в урахуванні відповідних ним витрат.

Спираючись на результати робіт [31, 32], були виявлені основні результати, які свідчать про якісно проведений аналіз ризиків:

1. Ранжування загального ризику проєкту;
2. Список ризиків по пріоритету;
3. Список ризиків для додаткового аналізу та управління;
4. Тренди в результатах якісного аналізу ризику.

Ранжування ризику може означати, що загальний ризик проєкту щодо інших проєктів може бути високий або низький. Можна порівнювати ризики різних проєктів по відношенню один до одного. Ризики можуть бути розбиті за пріоритетом, за різною кількістю критеріїв. Це включає рейтинг: високий, низький, середній або рівень ієрархічної структури робіт.

Ризики, що потрапляють в категорію високих або середніх, мають бути головними кандидатами для подальшого аналізу, включаючи кількісний аналіз ризиків, і для подальших дій з управління ризиками. При повторенні аналізу проявляється тенденція - тренд в результатах аналізу. Такий тренд може зробити відгуки на ризик або подальший аналіз більш-менш терміновим і важливим.

2.3. Математична модель задачі мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення

Одним із методів ризик-менеджменту та аналізу є розв’язання задачі мінімізації ризикових збитків, яке здійснюється після ідентифікації ризиків. Така задача відноситься до задач лінійного програмування.

Побудуємо математичну модель задачі мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення.

У загальному вигляді задача лінійного програмування формулюється таким чином. Знайти вектор $x = (x_1, x_2, \ldots x_n)$, який забезпечує екстремум (максимум чи мінімум) цільової функції $f(x)$ за умови, що змінні вектору x належать деякій області G:

$$\begin{cases} f(\boldsymbol{x}) => extr \\ \boldsymbol{x} \in G \end{cases}$$

Залежно від виду функції f (x) і області G розрізняють такі види математичного програмування: квадратичне програмування, опукле програмування, цілочисельне програмування, лінійне програмування та інші.

Лінійне програмування характеризується тим, що:

- цільова функція f (x) є лінійною функцією змінних x_1 , x_2 , ... x_n;

- область G визначається системою лінійних рівнянь і нерівностей, які задають обмеження для змінних x_1 , x_2 , ... x_n.

Вектор x = (x_1 , x_2 , ... x_n), компоненти якого задовольняють обмеженням, називають планом, або припустимим рішенням задачі лінійного програмування.

Припустиме рішення, що забезпечує екстремум (максимум чи мінімум) цільової функції є оптимальним планом задачі:

f (x*) = max(min) f(x), де x* = (x_1 *, x_2 *, ... x_n *)

Виходячи із загального формулювання задачі лінійного програмування, задача мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення має бути сформульована у такий спосіб.

Нехай вектор x складається із компонентів, що є ресурсами x_1 , x_2 , ... x_n кту.

Для обчислення ризиків Ri, згідно із джерелами [30], застосуємо формулу:

$$R_i = \sum_{i=1}^{n} P_i C_i \text{ , де}$$

Pi – вірогідності виникнення ризику для і-го ресурсу

C_i – вартість ризикових збитків на одиницю і-го ресурсу

і – кількість ресурсів та вірогідних загроз для кожного з них.

Необхідно визначити, скільки одиниць кожного ресурсу x_i необхідно виділити на проєкті, для того щоб вартість компенсації наслідків ризиків (ризикових збитків) була мінімальною:

$$F(x) = \sum_{i=1}^{n} R_i x_i = r_1 x_1 + \cdots + r_n x_n \rightarrow min$$

При цьому має виконуватись система обмежень щодо наявності запасів ресурсів та їх співвідношення, як формуються для кожного конкретного проєкту, виходячи із його масштабів та особливостей, а також на основі аналізу ідентифікованих рисків. Система обмежень формуються у вигляді системи нерівностей.

Для розв'язання задачі мінімізації ризикових збитків як задачі лінійного програмування, в якому здійснюється скерований рух по опорних планах до знаходження оптимального плану звичайно застосовується симплекс-метод. Це

ітеративний процес спрямованого розв'язання системи рівнянь по кроках, який починається з опорного рішення і в пошуках кращого варіанта рухається по кутових точках області припустимих рішень, що покращують значення цільової функції доти, поки цільова функція не досягне оптимального значення. Для автоматизації розв’язання такої задачі симплекс-методом доцільно застосувати процедуру *Пошук рішення* пакету аналізу у MS Excel.

Висновки до розділу 2

У другому розділі було розглянуто сучасні методи ідентифікації ризику, а саме метод Delphi, метод карт Кроуфорд, SWOT аналіз та метод експертної оцінки, який й був використаний для ідентифікації ризиків на проєкті. Було розглянуто ризик менеджмент як невід'ємну складову удосконалення процесу проєктування програмного забезпечення та сучасні методи управління ризиками, а саме: визначення параметрів граничного рівня, теорія статистичних рішень, метод ставки дисконту, метод достовірних еквівалентів та оптимізаційний метод, який у подальшому буде використано для мінімізації ризикових збитків. Також у другому розділі було побудовано математичну модель задачі мінімізації ризикових збитків для удосконалення процесів проєктування програмного забезпечення у загальному вигляді.

3. ВПРОВАДЖЕННЯ УДОСКОНАЛЕННЯ ПРОЦЕСІВ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ШЛЯХОМ МІНІМІЗАЦІЇ РИЗИКІВ

3.1. Ідентифікація ризиків проєкту на підприємстві

Для аналізу ризиків був розглянутий проєкт в IT компанії IT-Solution, яка надає послуги для різних етапів діяльності компанії, починаючи від підготовки до відкриття бізнесу до розширення і зміни сфери його діяльності. IT-Solution створює мобільні рішення, якими користується щодня безліч людей. Такий проєкт і був досліджений у ході роботи над дипломним проєктом з точки зору управління ризиками.

Проєкт Team Stream - це мобільне застосунок для онлайн занять йогою. Онлайн платформи для тренувань в наш час дуже актуальні, адже можна займатися спортом і проводити тренування не виходячи з дому. Основний напрямок занять - це йога, але існують також інші направлення. Ця платформа має велику перспективу подальшого розвитку для клієнтів. Великою перевагою є те, що команда може легко отримувати зворотний зв'язок від клієнтів. Це дозволить дізнатися про недоліки і слабкі сторони системи та варіанти її поліпшення.

Нижче перераховано основний функціонал застосунку для тренера, який має аккаунт.

1. Реєстрація облікового запису та заповнення даних профілю, опис про себе, створення тарифних планів та умов їх використання;
2. Складання графіка онлайн занять та додавання клієнтів на них;
3. Проведення онлайн конференцій з кількома людьми у групі, а також і з однією людиною персонально;
4. Спілкування з клієнтами, можливість підтримувати зв'язок;
5. Можливість завантажувати тренування для офлайн доступу, можливість створювати добірки з таких відео та згрупувати їх за темами.

Користувачу платформи надаються такі функції.

1. Реєстрація акаунту та підписка на тренерів, які цікавлять;
2. Відображення стрічки з усіма відео тренерів та перегляд їхньої персональної інформації (сторінок);
3. Перегляд власного графіка занять та графіка тренера, можливість додавати собі тренування у календар;
4. Поїздка онлайн занять як у групі, так і наодинці з тренерів з

відеозв'язку;

5. Онлайн чат з тренером;

6. Можливість переглядати добірки тренера з відео для офлайн переглядів у вільний час.

Схарактеризуємо основні проблеми роботи на кті. Через великий обсяг роботи, вимог, що часто змінюються, і поганої комунікації в команді процес розробки значно затягується. Команда постійно не встигає розробити функціонал в оговорені сроки, витрачає занадто много часу на непотрібні процеси. Через це виникають непередбачені ризики, реагування на які не було сплановано заздалегідь, виникають проблеми із замовником через зірвані терміни, перевитрати бюджету, втрату прибутку та негативний відгук від клієнтів через пропущені баги поспіхом перед релізами, через тиск із боку замовника. Все це у купі дуже негативно впливає на проєкт.

До недоліків роботи слід віднести те, що проєкт розроблявся з використанням методології Waterfall. Це зумовило такі складнощі: через те що на кті часто вносили зміні вже під час розробки, було дуже складно до них адаптуватись, тестування проводилось на останніх етапах розробки великих модулей, дуже важко вносити зміни в документацію та доповнювати її, дуже часто команда брала занадто багато задач та не вкладалась у терміни через те що обсяг роботи був дуже великий.

Спираючись на теоретичний матеріал з розділу 1, було проаналізовано поточний процес розробки додатку та кінцеві цілі на проєкті та визначено, рекомендований до використання ітеративний метод розробки програмного забезпечення. Він полягає в тому, що цикл розділений на більш дрібні легко створювані модулі. Кожен модуль проходить через фази визначення вимог, проєктування, кодування, впровадження та тестування. Даний метод дуже підходить до цього проєкту тому що процедура розробки по інкрементної моделі передбачає випуск на першому великому етапі продукту в базовій функціональності, а потім вже послідовне додавання нових функцій, так званих «інкрементів».

В інкрементній моделі повні вимоги до системи поділяються на різні складання. Термінологія часто використовується для опису поетапного складання ПЗ, що дуже актуально для проєкту так як він активно розробляється. Є кілька циклів розробки, і разом вони становлять життєвий цикл «мульти-водоспад». Цикл розділений на дрібніші модулі, що легко створюються. Кожен модуль

проходить через фази визначення вимог, проєктування, кодування, впровадження та тестування. Саме це допоможе команді не загубити які етапи є, який етап зараз та який буде наступним Процедура розробки по інкрементної моделі передбачає випуск першому великому етапі продукту у базової функціональності, та був вже послідовне додавання нових функцій, про «інкрементів». Для ідентифікації ризиків проєктування означеного мобільного застосунку було залучено комбінацію двох методів - метод експертної оцінки та SWOT аналіз.

Як раніше вже обговорювалося, на проєкті є провідні розробники з великим досвідом роботи в даній галузі розробки. На додаток до цього вони беруть активну участь у розробці поточного проєкту із самого початку. Тому їхня думка є об'єктивною на проєкті і проведений SWOT аналіз буде дуже показовим, оскільки його проводять досвідчені експерти, а саме розробник з досвідом роботи більше 4х років, менеджер проєкту та бізнес-аналітик.

Проведений SWOT аналіз на підприємстві дав наступний результат, який представлено на рис. 3.1.

Рис. 3.1. SWOT аналіз проєкту Team Stream

Виходячи з результатів SWOT аналізу проєктування мобільного застосунку, було ідентифіковано такі ризики на проєкті.

При розробці програмного забезпечення дуже важливо відразу правильно розпланувати завдання та погодити їх з замовником. Залежно від цього буде формуватися бюджет на розробку. Через погане планування може виникнути проблема з перевитратою бюджету, а саме час може бути витрачено на не пріоритетний функціонал, неправильно викладені вимоги, зайву комунікацію і інше. Також бюджет виділений на команду розподіляється не завжди правильно всередині неї - іноді катастрофічно мало часу приділяється на якісне документування вимог, тестування або ж налагодження додатка, перевірку коду, рефакторинг і виправлення технічних проблем.

Непорозуміння з замовником є великою проблемою при розробці програмного забезпечення. Такі проблеми можуть виникати внаслідок недостатньої комунікації, або ж іноді навпаки через надмірну комунікації, коли на мітинги витрачається занадто багато часу вони втрачають свою ефективність. Також однією з причин може бути неправильно задокументовані вимоги або в цілому їх відсутність.

Однією з найважливіших проблем на проєкті є погана якість коду через відсутність правильно спланованого тестування. Так як бюджет розподіляється невірно на проєкті, на тестування виділяється мало часу. Як результат, тестування покриває код не повністю і через це пропущено велика кількість проблем, які потрапляють до кінцевого користувача. Користувачі ж в свою чергу можуть залишати негативні відгуки, які видно замовнику.

Запуск програми з запізненням також є поширеною проблемою при розробці програмного забезпечення. Пов'язано це з тим що часто команда не вкладається в раніше узгоджені терміни, спочатку дає неправильну оцінку за часом на розробку або ж не закладає час на непередбачені ситуації. Також бувають варіанти коли перед релізом поповнюється список функціоналу який повинен бути випущений і цей функціонал раніше не узгоджений. Через це випуск програми може бути перенесений / відкладений.

На проєкті працює не надто велика кількість кваліфікованих співробітників. Всі вони є досвідченими і висококваліфікованими. Проте виникають непередбачувані ситуації, коли співробітник відсутній на робочому місці. Це викликає проблему передачі його завдання іншим розробникам, що призводить до перевитрат часу та затримки строків розробки ПО.

Ідентифіковані ризики та ресурси проєкту, на які можуть вплинути означені ризики, наведено у табл. 3.1.

Таблиця 3.1

Ідентифіковані ризики та ресурси проєкту

Ідентифікатор ризику	Ризик	Ресурс (x)
R_1	Невірна оцінка часових меж проєкту	Час на оцінювання робіт
R_2	Недостатня взаємодія із замовником	Час на взаємодію с замовником на етапі розробки
R_3	Недоліки тестування	Час на планування тестування
R_4	Зміна вимог до підзадач проєкту у процесі його реалізації	Час на документування вимог до підзадач
R_5	Недостатня кількість кваліфікованих розробників	Час роботи команди розробників

Після ідентифікації ризиків було оцінено вірогідність P_i їх виникнення на проєкті для кожного із ресурсів x_i та вартість C_i ризикових збитків на одиницю кожного ресурсу. Для цього був використано метод експертної оцінки ризиків, розглянутий у розділі 2.

В ролі експертів виступали Менеджер проєкту, Замовник, Бізнес Аналітик, Менеджер с тестування, Провідний розробник та Дизайнер. Саме ці співробітники мають великий досвід роботи та працюють на працюють на проєкті з самого початку, тому вони зможуть об’єктивно оцінити.

Кожен з них висловив свою думку щодо вірогідності виникнення кожного з ризиків. Потім ці результати було порівняно, обчислено середньоарифметичне оцінок вірогідностей, яке було використано для формування цільової функції, що виражає загальну вартість ризикових збитків.

У такий же спосіб було оцінено вартість C_i ризикових збитків на одиницю кожного ресурсу.

На рис. 3.2 наведено результати оцінки кожного з експертів вірогідності P_i виникнення ризиків на проєкті для кожного із ресурсів x_i та вартості C_i ризикових збитків на одиницю кожного ресурсу.

	Вероґідність виникнення					Вартість компенсації				
експерт	R1	R2	R3	R4	R5	R1	R2	R3	R4	R5
1	0,25	0,22	0,75	0,5	0,5	18	36	24	10	50
2	0,45	0,75	0,23	0,6	0,1	15	25	36	11	47
3	0,24	0,55	0,5	0,75	0,2	28	14	30	29	36
4	0,6	0,5	0,33	0,6	0,4	13	34	21	14	25
5	0,2	0,2	0,7	0,88	0,3	25	16	40	13	40
	0,348	0,444	0,502	0,666	0,3	19,8	25	30,2	15,4	39,6

Рис. 3.2. Результати оцінки кожного з експертів щодо вірогідності P_i виникнення ризиків на проєкті для кожного із ресурсів x_i та вартості C_i ризикових збитків на одиницю кожного ресурсу

В подальшому ці дані будуть використані для розрахунків коефіцієнтів капіталовкладення на одиницю ресурсу для побудови цільової функції задачі мінімізації ризикових збитків (див. п. 2.3).

3.2. Впровадження моделі мінімізації ризикових збитків для проєкту на підприємстві

Для мінімізації ризикових збитків для проєкту на підприємстві було впроваджено математичну модель задачі мінімізації ризикових збитків, побудовану у п. 2.3.

Відповідно до результатів експертної оцінки (Рис. 3.2), було розраховано коефіцієнти ризикових збитків за формулою:

$$R_i = \sum_{i=1}^{n} P_i C_i \text{ , де}$$

Pi – вірогідності виникнення ризику для і-го ресурсу

C_i – вартість ризикових збитків на одиницю і-го ресурсу

і – кількість ресурсів та вірогідних загроз для кожного з них.

Були отримані наступні результати для коефіцієнтів:

$R_1 = 0.3 * 20 = 6$

$R_2 = 0.4 * 25 = 10$

$R_3 = 0.5 * 30 = 15$

$R_4 = 0.7 * 15 = 10.5$

$R_5 = 0.3 * 40 = 12$

Тоді задача мінімізації ризикових збитків для удосконалення процесів проєктування нашого додатку на підприємстві полягає у наступному.

Визначити, скільки одиниць кожного ресурсу x_i необхідно виділити на проєкті, для того щоб вартість компенсації наслідків ризиків (ризикових збитків) була мінімальною:

$$F(x) = \sum_{i=5}^{n} R_i x_i = 6x_1 + 10x_2 + 15x_3 + 10.5x_4 + 12x_5 \rightarrow min$$

При цьому має виконуватись така система обмежень:

1. $x_1 , x_2 , x_3 , x_4 , x_5 \geq 1$
2. $x_1 + x_2 + x_3 + x_4 + x_5 \leq 700$
3. $x_1 + x_2 + x_3 + x_4 + x_5 \geq 500$
4. $x_1 + x_2 + x_3 + x_4 \leq x_5$
5. $x_3 \geq x_1$
6. $x_2 \geq 15$

Система обмежень побудована, виходячи із аналізу наявності запасів ресурсів на кті та на основі їх економічного та фізичного сенсу.

Усі ресурси повинні бути позитивними (обмеження 1).

На проєкт виділяється обмежений бюджет на розробку, що відбивається на загальному часі роботи команди, а саме на усі ресурси в сумі не може бути витрачено понад 700 годин (обмеження 2).

Водночас запланований об'єм робіт не може бути виконаний менш ніж за 500 годин (обмеження 3).

На суто розробку ПЗ має бути відведено більше часу ніж на проєктування, планування, спілкування із замовником і тестування разом (обмеження 4).

Оскільки проблемою даного проєкту є недолік тестування та надлишок спілкування із замовником, раціональним рішенням було ввести обмеження на те, що на тестування часу має бути витрачено більше часу, ніж на проміжні обговорення вимог із замовником (обмеження 5).

На спілкування із замовником заздалегідь спланова‌но виділити не менше 15 годин (обмеження 6).

Для автоматизації розв’язання такої задачі мінімізації ризикових збитків симплекс-методом було застосовано процедуру *Пошук рішення* пакету аналізу у середовищі MS Excel.

Для цього у відповідні комірки робочого аркушу було введено:

числові значення Pi (вірогідності виникнення ризику для і-го ресурсу), C_i (вартість ризикових збитків на одиницю і-го ресурсу) та числові значення системи обмежень;

формули для обчислення коефіцієнтів цільової функції Ri;

формула цільової функції із посиланням на комірки коефіцієнтів Ri та на пусті комірки, що відіграють роль шуканих ресурсів x_i.

Далі було застосовано процедуру Пошук рішення пакету аналізу MS Excel на рис. 3.3.

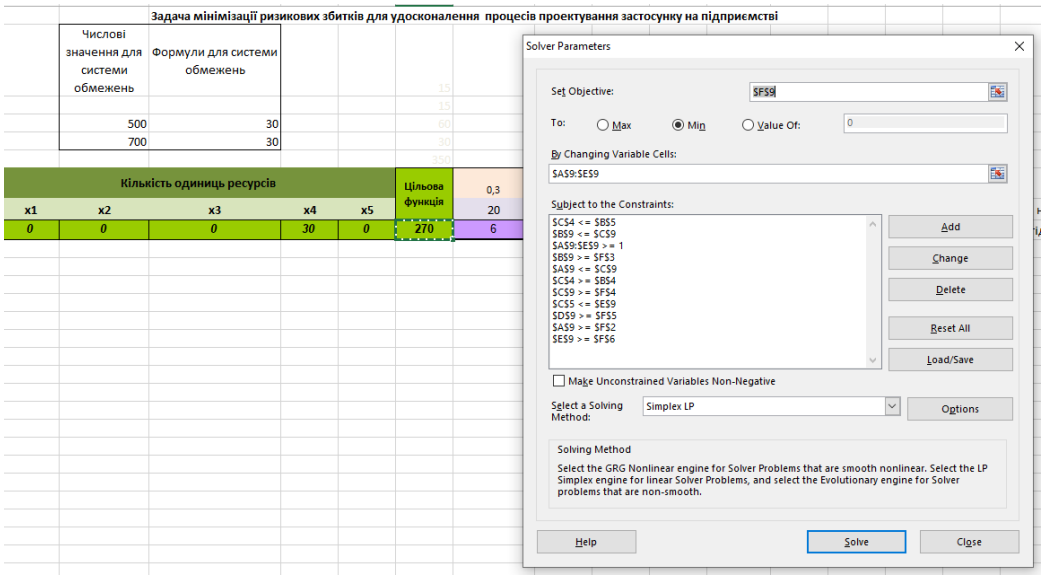

Рис. 3.3. Пошук рішення пакету аналізу MS Excel

В результаті було отримано, скільки одиниць кожного ресурсу має бути виділено, для того щоб ризикові збитки були мінімальні. Отримані результати розв'язання задачі мінімізації наведено на рис. 3.4.

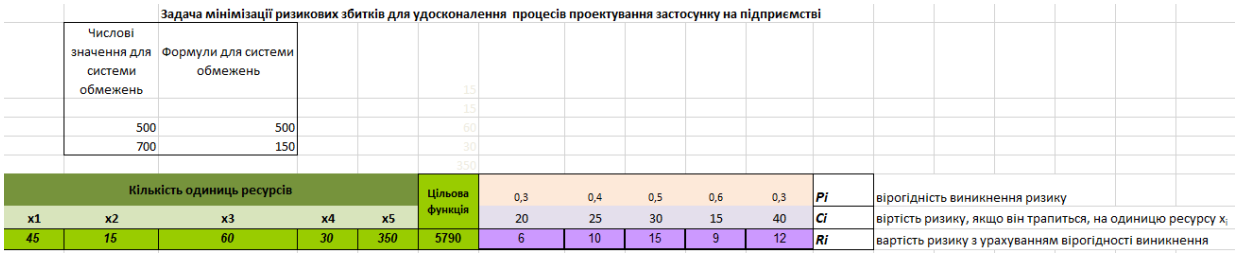

Рис. 3.4. Результати розв'язання задачі мінімізації вартості ризикових збитків

Результати розв'язання задачі мінімізації відображають, як потрібно

розподілити ресурси на проєкті для того щоб мінімізувати вартість ризикових збитків, якщо ризики виникнуть:

$x_1 = 45$ год.

$x_2 = 15$ год.

$x_3 = 60$ год.

$x_4 = 30$ год.

$x_5 = 350$ год.

Цільова функція при цьому дорівнює 5790 умовних одиниць.

3.3. Планування та впровадження заходів для зниження ризиків

На підставі аналізу можна вирішити, чи вимагають ризики реагування. Наприклад, деякі ризики вимагають реагування в плані проєкту, в той час як деякі вимагають лише моніторингу на проєкті, а деякі взагалі не вимагають відповіді. Для мінімізації вірогідності виникнення ризиків та зниження вартості ризикових збитків при розробці програмного забезпечення необхідно побудувати план реагування на ризик. План допоможе знизити ймовірність виникнення ризиків та уникнути критичних наслідків на проєкті під час розробки.

Неправильна оцінка проєкту. Коли складається оцінка проєкту, буває, що вона не виправдовує очікувань. Команда може вибрати тривалість ітерації проєкту, стек технологій та інші чинники. Між клієнтом і командою часто виникають розбіжності, що призводить до збільшення тривалості завдання, витрат, через які у клієнта закінчуються гроші і він не може завершити проєкт.

Для мінімізації цього ризику:

1. Необхідно в першу чергу виконувати тільки найважливіші завдання;
2. Додати час розробникам для вивчення і зниження ризиків в частинах нового проєкту;
3. Додати розробникам час на ознайомлення з функціоналом до проведення оцінювання;
4. Додати час на планування роботи та обговорення її оцінки;
5. Додати передбачуваний період для команди розробників протягом тижня для завдання, що виходить за рамки проєкту;
6. В управлінні проєктами конус невизначеності описує розвиток невизначеності в кращому випадку під час проєкту. На початку проєктів, мало що відомо про продукт або результати роботи, тому оцінки схильні до великої

невизначеності.

Для того щоб відстежувати роботу в команді, кількість витраченого часу і швидкість виконання роботи рекомендується використання автоматизованої системи для управління проєктами та завданнями, такі як Jira, Trello, Asana і інші. Більш детально розглянемо Jira, так як в даний час це найактуальніша і найпопулярніша система, яка полегшує і спрощує роботу команди.

JIRA - платформа для управління проєктами, завданнями і відстеження помилок. Платформа призначена, в першу чергу, для розробників і ведення agile-проєктів. Jira доступна в веб-версії і у вигляді десктопного додатка.

Платформа JIRA являє собою гнучкий інструмент, в якому компанії можуть адаптувати і змінювати під свої потреби функціональний зовнішній вигляд сервісу. Для кожного окремого проєкту адміністратор Jira може визначити тип завдань з унікальними складовими елементами, прив'язати набір статусів і інше. До кожному проєкту можна визначити права доступу для учасників.

Структура JIRA складається з трьох елементів: проєкт, завдання і підзадачі. Проєкт - основний елемент платформи, в якому зберігаються завдання і інформація по роботі над програмою. Користувачі мають можливість створити проєкт з нулі або використовувати готовий шаблон. Для відстеження ходи роботи над проєктом автоматично створюється дорожня карта. Дорожня карта проєкту являє собою ієрархічну структуру, що дозволяє планувати робочий процес в різних часових перспективах, відстежувати процес виконання завдань і систематизувати роботу декількох команд над одним проєктом.

Завдання JIRA - це структуровані інструменти для управління проєктом. У завданнях міститься інформація про необхідні дії, фіксується час для її виконання, встановлюється виконавець, прикріплюються додаткові файли. Користувач може отримувати повідомлення про внесених змінах завдання, вести журнал виконання, створювати підзадачі і залишати коментарі.

Особливості JIRA:

1. Kanban-дошка - допомагає команді забезпечити прозорість роботи над проєктом, оптимізувати робочий процес, розподілити завдання з беклога (список завдань). Приклад такої дошки зображений на рис. 3.5.

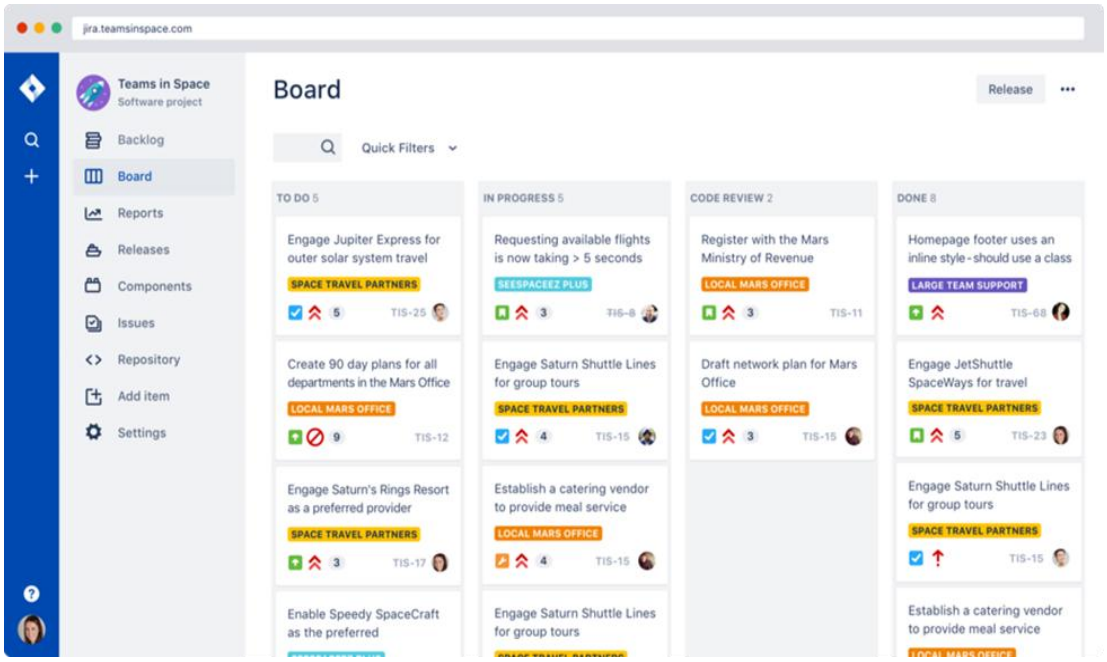

Рис. 3.5. Kanban дошка

2. Scrum-дошка - дозволяє управляти складним проєктом, об'єднати команди з різних напрямків розробки продукту для досягнень однієї мети;

3. Прив'язка програмного коду до завдань за допомогою Bitbucket і спільна над ним;

4. Ведення документації, протоколів та інших документів за допомогою Confluence;

5. Спільна робота - обмін інформацією по проєкту, спільне вирішення питань і звернення за допомогою до колег;

6. Звітність в JIRA - звіти формуються за допомогою віджетів на панелі дашборда і можуть містити інформацію про проєкт в цілому або про окремі його елементи. Звіти візуалізуються в графіки та діаграми;

7. Підтримка інтеграцій з безліччю інструментів для розробки і інших сервісів.

Ризики поганої взаємодії між замовником ПЗ і його виконавцем під час розробки програмного забезпечення. Вони пов'язані з відсутністю комунікації між керівниками або їх представниками. Недостатнє обговорення користувацьких вимог до ПЗ або його архітектури до та навіть після старту процесу реалізації може негативно позначитися на його майбутньому функціоналі та, як наслідок, на якість самого ПЗ. Більш того щоб команда розробників закінчила вчасно проєкт

необхідно досить часто спілкуватися з клієнтом або менеджером проєкту для уточнення деталей по проєкту навіть вже на етапі реалізації. Якщо цього не робити, то є ризик непорозуміння з обох сторін і збільшення тривалості ітерацій. Але варто враховувати, що надмірна комунікація про свою ефективність прирівнюється до недокомунікації. Якщо витрачати спілкування занадто багато часу, його потім може бути замало для виконання основного запланованого обсягу роботи. Тому важливо максимально точно для поточного проєкту підібрати оптимальну кількість часу на початкову та проміжну комунікацію із замовником

Для мінімізації цього ризику:

1. Необхідно чітко узгодити, коли клієнт і замовник зможуть провести Тестування користувача (User Acceptance Testing);
2. Важливо обумовити допустимий час відповіді, якщо у кожної зі сторін є проблема або питання по проєкту;
3. Узгодити часові межі для комунікації;
4. Потрібно визначати чіткий вибір цілей і пріоритетів проєкту.

Код низької якості та недоліки тестування. Низька якість коду - одна з найпоширеніших проблем в розробці і одна з найбільших проблем клієнта. Найчастіше замовник не розуміє код і не може визначити його якість. До завершення розробки може з'ясуватися, що у додатки не якісно розроблений код, проблеми в роботі і відсутність грамотного тестування.

Як мінімізувати цей ризик:

1. Розробникам важливо слідувати розробленим стандартам коду;
2. Клієнт може найняти менеджера проєкту або технічного директора, який може перевіряти якість коду та контролювати команду розробників;
3. Дотримання системи Стандартів Якості Коду (англ. Clear Coding Standards and Guidelines);
4. Тестування після кожної ітерації коду.

Найбільш важливу роль грає правильне тестування системи. Більш детально розглянемо вид тестування який базується на ризиках - Risk Based Testing. Це в основному тестування, яке проводиться для проєкту на основі ризиків. Тестування на основі ризику використовує ризик, щоб визначити пріоритети та підкреслити відповідні тести під час виконання тесту. Простіше кажучи – ризик – це ймовірність настання небажаного результату. Цей результат також пов’язаний із впливом. Оскільки може не вистачити часу на тестування всіх функціональних можливостей, тестування на основі ризиків передбачає тестування

функціональних можливостей, які мають найбільший вплив та ймовірність збою.

Тестування на основі ризику — це ідея, що ми можемо організувати наші зусилля з тестування таким чином, щоб зменшити залишковий рівень ризику продукту під час розгортання системи.

1. Тестування на основі ризиків починається на початку проєкту, визначає ризики для якості системи та використовує ці знання про ризики для планування, специфікації, підготовки та виконання тестування;
2. Тестування на основі ризику включає як пом’якшення – тестування, щоб надати можливість зменшити ймовірність дефектів, особливо дефектів, що мають значний вплив, – так і тестування на випадок непередбачених обставин, щоб визначити обхідні шляхи, щоб зробити дефекти, які проходять повз нас, менш болючими;
3. Тестування на основі ризиків також передбачає вимірювання того, наскільки добре ми працюємо у пошуку та усуненні дефектів у критичних зонах;
4. Тестування на основі ризику також може включати використання аналізу ризиків для виявлення можливостей для усунення або запобігання дефектів за допомогою тестування, а також для того, щоб допомогти нам вибрати, які тестові дії виконувати.

Метою тестування на основі ризику практично не може бути – проєкт без ризику. Те, що ми можемо отримати від тестування на основі ризиків, — це провести тестування з використанням найкращих практик з управління ризиками, щоб досягти результату проєкту, який урівноважує ризики з якістю, функціями, бюджетом і графіком.

Часті зміни у вимогах вже після початку виконання підзадачі. Вносячи зміни, необхідно спочатку проаналізувати як це вплине на поточний стан проєкту, скільки зусиль потребує і чи існує ризик затримки. Завдяки аналізу, ви зможете грамотно розподілити обов'язки, внести зміни в пріоритети і надати клієнту точну інформацію про те, що може (або не може) бути виконано. Нерозділена реакція в такому випадку буде означати ваше беззастережне прийняття тих чи інших змін. Це поширена практика в процесі розробки ПЗ. Саме тому дуже важливо всім учасникам проєкту усвідомлювати наслідки змін і спільно йти на деякі компроміси, якщо вони необхідні.

Результатом цієї проблеми може бути те, що команда даремно витрачає час на повторювані питання стосовно базової інформації про проєкт. Відсутність хороших контрольних показників для використання членами команди як під час

проєкту, так і після нього. Недостатні знання членів команди, які приєдналися до проєкту на півдорозі.

Мінімізація цього ризику:

1. Навіть мінімальна проєктна документація може зіграти велику роль в запобіганні наслідків. Згідно кращим практикам Agile: "працює ПО важливіше детальної документації". Тим не менш, не треба вважати документацію малозначним елементом.

2. Необхідно відвести деякий час на написання документації з самого початку. В такому випадку, у вас не буде ніяких відмовок. Використовуйте такі інструменти, як JIRA, Confluence або QA Touch - вони дійсно полегшують роботу. Також існує багато більш спеціалізованих інструментів, які допоможуть вам написати документацію для ІПП та інших звітних матеріалів по проєкту. Необхідно визначити яка інформація повинна завжди бути доступною. Гарне місце для її зберігання - Confluence. Це система, що дозволяє знайти всю базову проєктну документацію, членів команди, їх ролі та іншу важливу інформацію за властивостями проєкту, його середовища, опису користувачів і переліку функцій.

3. Потрібно постійно використовувати спеціальні умовні позначення для позначення і опису завдань. Документація не повинна бути об'ємною. Її завдання - всебічне опис проєкту простою і зрозумілою мовою.

Більш детально розглянемо систему Confluence, яка значно спрощує процес внесення змін до проєкту та їх документування. На малюнку 3.6. зображений інтерфейс користувача системи confluence.

Confluence - це простір для команд, в якому накопичені знання об'єднані з можливостями для спільної роботи. Динамічні сторінки являють собою майданчик для творчості, збору інформації та спільної роботи учасників команди над будь-якими проєктами та ідеями. Завдяки розділах можна структурувати і організовувати роботу в команді, а також надавати загальний доступ до бази знань організації і до інформації, необхідної учасникам для ефективної роботи.

Confluence можуть використовувати команди будь-якого розміру і типу - як ті, які займаються великими, критично важливими проєктами і повинні строго дотримуватися рекомендацій, так і ті, які шукають простір для формування командного культури і більш відкритого і природного способу взаємодії один з одним.

Озброївшись Confluence, команда зможе швидко приймати рішення, домогтися узгодженості і досягти великих результатів при спільній роботі.

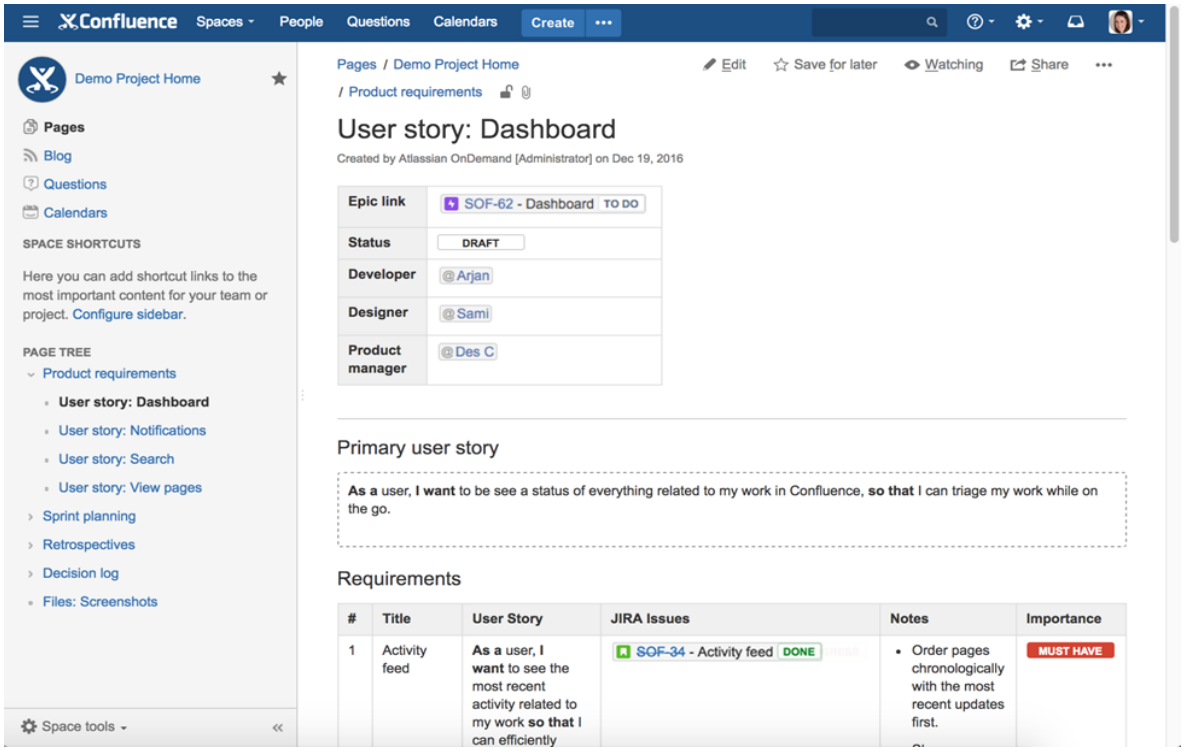

Рис. 3.6. Документування функціоналу

Озброївшись Confluence, команда зможе швидко приймати рішення, домогтися узгодженості і досягти великих результатів при спільній роботі.

Недостатня кількість кваліфікованих співробітників. Залученість всіх членів команди є обов'язковою умовою для успіху кожного проєкту. Тому дуже важливо, щоб кожен учасник процесу був відданий спільної мети, розумів свою роль і підтримував інших членів команди. Кожне незаплановане відсутність члена команди служить підставою для занепокоєння. Нестача знань про проєкт, якщо даний фахівець був головним членом команди (знову ж, стає очевидною важливість гарної документації!).

Мінімізація цього ризику:

1. Важливо, щоб всі члени команди мали однаковими основними знаннями про проєкт. Залежно від того, як довго відсутній співробітник і на якому етапі знаходиться проєкт, проджект-менеджер повинен прийняти рішення про необхідність заміни;

2. Новому співробітнику буде простіше приступити до роботи якщо з ним діляться інформацією про проєкт і нададуть документацію;

3. Також необхідно розглянути варіант розширення штату співробітників

на проєкті та навчання нового персоналу, але потрібно мати на увазі що це потребує багато часу.

3.4. Експериментальна перевірка ефективності запропонованих заходів мінімізації ризиків

Після того як на проєкті були виявлені ризики, оцінені ймовірні витрати на їх компенсацію збитків, отримано результати мінімізації ризикових збитків і впроваджено план реагування, було проаналізовано та виявлено, що частота виникнення непередбачених ризиків значно знизилася, як результат і менше витрат йде на покриття компенсації цих ризиків. Також продуктивність команди дуже зросла і тепер роботи робить у рази більше, незважаючи на те, що часу витрачається стільки ж, а в деяких випадках навіть менше.

Значно покращилася здатність команди об'єктивно оцінювати обсяг передбачуваної роботи, чітко визначати вимоги та визначати яку кількість часу потрібно, щоб виконати поставлене завдання. Колективні зустрічі та обговорення допомогли покращити взаємини у команді. Також наперед виділений час на ознайомлення з вимогами відіграв важливу роль у прийнятті рішення щодо оцінки обсягу робіт. Як результат кількість "промахів" по годинах скоротилася, недовантажені спринти або ж навпаки перевантажені спринти перестали з'являтися. Більше того, команда стала чіткіше повідомляти інформацію замовнику про свої оцінки по роботі, що відіграло позитивну роль на відносини із замовником.

Важливими змінами є покращення якості програмного забезпечення та коду загалом. Він тепер відповідає всім стандартам якості, як результат менше багів та непрацюючого функціоналу доходить до кінцевих користувачів. Також кількість позитивних відгуків збільшилася через те, що всі релізи проходять якісніше та частіше. Перестало витрачатися зайвий час виправлення всіх недоліків. Важливу роль цьому поліпшенні також зіграло тестування після кожної ітерації розробки. За виділений час команда тестувальників повною мірою покриває код тестами та перевіряє весь функціонал. Тепер час, який команда витратила на повторне виправлення дефектів та рефакторинг витрачається на тестування.

Проблему з недостатньою комунікацією із замовником практично усунули. Команда запровадила додаткові мітинги із замовником вже на етапі розробки програмного забезпечення, а не лише на етапі планування роботи. Дані мітинги

відіграють велику роль для того, щоб переконатися, що за підсумком розроблюваний продукт відповідає очікуванням замовника. Команда внесла обмеження за часом на такі зустрічі щоб не було перевитрати часу, тому що як вже було з'ясовано надлишок комунікації так само позначається на проєкті як її нестача. Також слід зауважити, що замовника став брати участь у проціджені приймального тестування після фази розробки. Це допомагає команді показувати результат, а заказнику дає можливість визначити чи відповідає це його вимогам.

Поліпшилося становище із вимогами на проєкті, адже тепер вони добре задокументовані вже до початку розробки функціоналу. Також були проведені розмови із замовником щодо зміни вимог у процесі розробки підзавдання та які збитки це може зазнавати. Було безперечно на якому ще моменті розробки команда зможе підлаштуватися під нові вимоги, а де вже ні. Була впроваджена система, за допомогою якої бізнес аналітик та проджект менеджер чітко прописують вимоги до функціоналу, який раніше узгоджують із замовником. У замовника також є доступ до даної системи і за потреби він і сам може її перевіряти.

На жаль, питання з недостатньою кількістю висококваліфікованих розробників все ще відкрите, оскільки цей ризик дуже складно передбачити завчасно. Склад співробітників у будь-який момент може змінитися, співробітники можуть захворіти, перейти на інший проєкт, піти у відпустку чи звільнитися. Вирішили виділити частину часу навчання нових стажерів розробників, які мають достатньо досвіду на проєкті. Досвідчені розробники почали вести менторство над джуніорами розробниками. Поліпшення роботи у цьому напрямі складно оцінити, оскільки для того, щоб повною мірою побачити результат, необхідна велика кількість часу на навчання нового персоналу. Проте незважаючи на це, команда впровадила дії, які допомогли звести цей ризик до мінімуму.

Таблиця 3.1

Результати опитування експертів щодо вірогідності виникнення ризиків після впровадження плану реагування

Ризик	Вірогідність виникнення ризику до впровадження плану реагування на ризики	Вірогідність виникнення ризику після впровадження плану реагування

Невірна оцінка часових меж проєкту	0,3	0,15

Закінчення таблиці 3.1

Недостатня взаємодія із замовником	0,4	0,15
Недоліки тестування	0,5	0,1
Зміна вимог до підзадач проєкту у процесі його реалізації	0,6	0,2
Недостатня кількість кваліфікованих співробітників	0,3	0,05

Отримані результати після впровадження плану реагування на ризики були обговорені повторно з експертами. Було прийнято рішення, що показники ймовірності виникнення ризиків на проєкті значно знизилися. У табл. 3.1 наведено порівняльну характеристику результатів оцінювання експертів щодо вірогідності виникнення ризиків.

Виходячи з цього, можна дійти висновку що запропонований план управління ризиками є ефективним для конкретно розглянутого проєкту, тому як показники вірогідності виникнення ризику вже стали значно нижче.

Були використані нові данні оцінок по вірогідності виникнення ризиків на проєкті, які були отримані в результаті проведення ще однієї експертної оцінки (табл. 3.1).

Була побудована цільова функція з новими коефіцієнтами:

$$F(x) = \sum_{i=5}^{n} R_i x_i = 3x_1 + 3{,}75x_2 + 3x_3 + 3x_4 + 2x_5 \rightarrow min$$

Повторно була розв’язана задача мінімізації вартості ризикових збитків на рис. 3.7.

На цей раз ресурси були розподілені іншим чином. Можна побачити, що тепер на оцінювання робіт може бути витрачено менше часу, порівняно з першим результатом, а на написання коду – більше. Також мінімальне значення цільової функції знизилось на приблизно 80% і дорівнює 1131 умовних одиниць.

Задача мінімізації ризикових збитків для удосконалення процесів проектування застосунку на підприємстві

Числові значення для системи обмежень	Формули для системи обмежень
500	500
700	120

Кількість одиниць ресурсів					Цільова функція							
						0,15	0,15	0,1	0,2	0,05	*Pi*	вірогідність виникнення ризику
x1	x2	x3	x4	x5		20	25	30	15	40	*Ci*	віртість ризику, якщо він трапиться, на одиницю ресурсу x_i
15	*15*	*60*	*30*	*380*	1131,25	3	3,75	3	3	2	*Ri*	вартість ризику з урахуванням вірогідності виникнення

Рис. 3.7. Результати повторного розв’язання задачі мінімізації вартості ризикових збитків

$x_1 = 15$ год.
$x_2 = 15$ год.
$x_3 = 60$ год.
$x_4 = 30$ год.
$x_5 = 380$ год.

На рис. 3.8 наведено порівняльний графік розподілення ресурсів на кті, одержаних у результаті розв’язання задачі мінімізації ризикових збитків, до та після впровадження заходів для зниження ризиків.

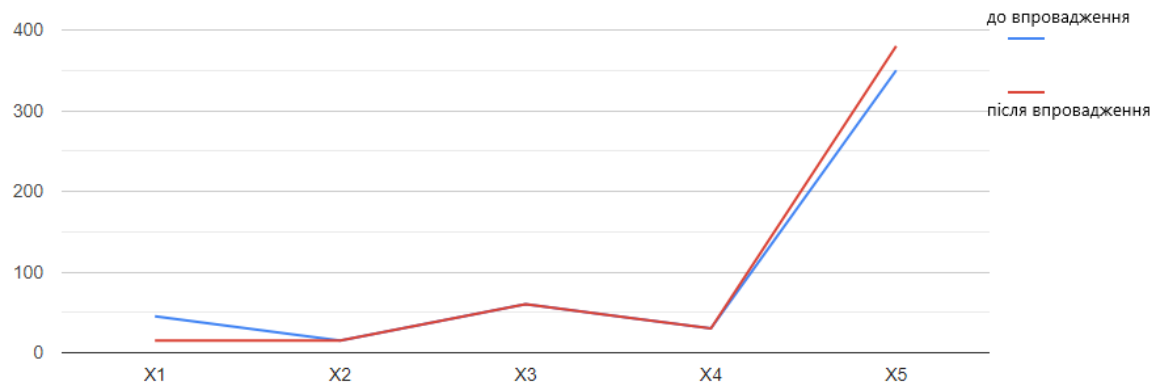

Рис. 3.8. Порівняльний графік розподілення ресурсів

Висновки до розділу 3

У третьому розділі було використано оптимізаційний метод для управління ризиками при проєктуванні програмного забезпечення. Було побудовано математичну модель задачі мінімізації ризикових збитків для удосконалення процесів проєктування спираючись на результати отримані в ході експертної оцінки. Дану модель мінімізації ризикових збитків було впроваджено для удосконалення процесів проєктування програмного забезпечення на підприємстві. Для проєкту були запропоновані заходи для зниження вірогідності ризиків та ризикових збитків та експериментально перевірено їх ефективність. Стан на підприємстві після впровадження було оцінено повторно, використовуючи експертну оцінку.

Результати показали, що завдяки запропонованим заходам для зниження вірогідності ризиків та ризикових збитків можна досягти зниження вартості ризикових збитків.

ВИСНОВКИ

У ході роботи було виконано усі поставлені завдання, що дозволяє зробити такі висновки.

1. Проаналізовано сучасні підходи до проєктування програмного забезпечення. Були виявлені переваги та недоліки кожного з цих методів. Було виявлено що усі методології мають загальні проблеми при проєктуванні програмного забезпечення, а саме підвищений ризик якщо знадобиться внести зміни, необхідність у жорсткому управлінні та контролі, проблеми щодо проєктування архітектури системи заздалегідь, підвищення вартості кту за рахунок додаткових часів на планування, прототипування, документування та визначення цілей кту. Було розглянуто поняття ризик, та його класифікацію, а саме зовнішні та внутрішні, технічні та нетехнічні, передбачувані та непередбачувані, фінансові, маркетингові та ризики персоналу.

2. Розглянуто сучасні методи ідентифікації ризику, а саме метод Delphi, метод карт Кроуфорд, Мозковий штурм, SWOT аналіз та метод експертної оцінки, який й був використаний для ідентифікації ризиків на кті. Було розглянуто ризик менеджмент як невід'ємну складову проєктування програмного забезпечення та сучасні методи управління ризиками, а саме визначення параметрів граничного рівня, теорія статистичних рішень, метод ставки дисконту, метод достовірних еквівалентів та оптимізаційний метод, який у подальшому буде використано для мінімізації ризикових збитків. Також у другому розділі була побудована цільова функція у загальному виді.

3. Було використано оптимізаційний метод для управління ризиками при проєктуванні програмного забезпечення та побудовано математичну модель задачі мінімізації ризикових збитків для удосконалення процесів проєктування спираючись на результати отримані в ході експертної оцінки. Дану модель мінімізації ризикових збитків було впроваджено для удосконалення процесів проєктування програмного забезпечення на підприємстві. Було запропоновано заходи для зниження вірогідності ризиків та ризикових збитків та експериментально перевірено їх ефективність. Стан на підприємстві після впровадження було оцінено повторно, використовуючи експертну оцінку.

Результати показали, що завдяки запропонованим заходам для зниження вірогідності ризиків та ризикових збитків можна досягти зниження вартості ризикових збитків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Агальцов В.П. Математичні методи у програмування. М., 2019.
2. Арбон Д. Як тестують Гугл / Д. Арбон, Д. Каролло, Д. Уіттакер., 2019. – 303 с.
3. Батенко Л.П. Управління проєктами «Принципи, запропоновані авторами». Л.П. Батенко, О.А. Загородніх, В.В. Ліщинська. – К., 2020. – 231с
4. Биков В. Ю. Хмарні технології, ІКТ-аутсорсинг і нові функції ІКТ підрозділів освітніх і наукових установ / В. Ю. Биков // Інформаційні технології в освіті. – 2018. – № 10. – С. 8–23.
5. Блек Р. Ключові процеси тестування / Рекс Блек., 2020. – 112 с.
6. Бондаренко Е. HR-аналітика в українських компаніях // HR ЛІГА : співтовариство кадровиків та фахівців з управління персоналом. [Електронний ресурс] – Режим доступу :https://hrliga.com/index.php?module=profession&op=view&id=1844
7. Великі дані. [Електронний ресурс] – Режим доступу :https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D1%96_%D0%B4%D0%B0%D0%BD%D1%96
8. Винокурова В. К. Дослідження ризиків при проєктуванні ПЗ за різними методологіями / В. К. Винокурова, Л. Е. Гризун // Інформаційні технології та системи: зб. тез Міжнар. наук-практ. конф. (Х – Х грудня 2021 р.; ХНЕУ ім. С. Кузнеця). – Х. : ФОП Бровін О. В., 2021. – С. 45.
9. Винокурова В. К. Управління ризиками при проєктуванні програмного забезпечення за різними методологіями / В. К. Винокурова, Л. Е. Гризун // Інформаційні технології та системи: зб. тез Міжнар. наук-практ. конф.(14 – 15 квітня 2022 р.; ХНЕУ ім. С. Кузнеця). – Х. : ФОП Бровін О. В., 2022. – С. 5.
10. Говер І.І., Шапіро В.Д. Управління ризиками на проєкті. Довідковий посібник. М: Вища школа, 2021.
11. Гокін В.П. Математичні методи у програмування. М., 2019.
12. Грей К, Ларсен Е. Управління проєктами. Пров. з англ. - М.: «Справа та Сервіс».2020.
13. Данко П.Є. Вища математика у заняттях і завданнях: У 2 т. навч. посіб. М: Вища. шк., 2018.
14. Дастін Е. Автоматизація тестування програмного забезпечення / Е. Дастін, Д. Решка, Д. Пол., 2019. – 356 с.

15. Дитхелм Герд Управління проєктами, Бізнес-преса, 2018, Том 1 "Основи", Том 2 "Особливості",.

16. Електронний ресурс. – Режим доступу: http://upma.kiev.ua/content/view/40/78/lang,ru/

17. Електронний ресурс. – Режим доступу: http://www.aacei.org/mbr/whoWeAre.shtml

18. Електронний ресурс. – Режим доступу: http://www.aacei.org/mbr/whoWeAre.shtml

19. Електронний ресурс. – Режим доступу: http://www.ipma.ch/Pages/default.aspx

20. Електронний ресурс. – Режим доступу: http://www.ppe.гф/mbr/ecommerce.shtml

21. Електронний ресурс. – Режим доступу: https://alib.com/b/2598/read

22. Електронний ресурс. – Режим доступу: https://coollib.com/b/222084/read

23. Електронний ресурс. – Режим доступу: https://studfile.net/preview/7153489/

24. Електронний ресурс. – Режим доступу: https://ukrbukva.net/107494-Strategiya-razvitiya-predpriyatiya.html

25. Єрмаков В.І. Загальний курс математики для економістів. М: Інфра-М, 2019.

26. За загальною редакцією Шапіро В.Д. Управління проєктами. Підручник.: «Два Три», 2020.

27. Канер С. Тестування програмного забезпечення / С. Канер, Д. Фолк, Е. Нгуен., 2020. – 421 с.

28. Керівництво до Зводу знань з управління проєктами. Третє видання (Посібник PMBOK)/. Американський національний стандарт ANSI/PMI 99-001-2019).

29. Класифікація методів управління ризиками [Електронний ресурс]. URL: https://www.intuit.ru/studies/courses/3506/748/lecture/26284?page=2 (дата звернення: 30.03.2020)

30. Кобилянський Л.С. Управління ризиками [навча.посібник] / Л.С. Кобилянський. – К.: Мир управления проєктами: основы, методы, организация, применение / Под. ред. Х. Решке, Х.Шелле.

31. Комплексний тренінг: методичні рекомендації для студентів спеціальності 126 “Інформаційні системи та технології” другого (магістерського) рівня [Електронне видання] / укл. О. В. Дорохов, І. О. Ушакова. – Харків : Вид. ХНЕУ ім. С. Кузнеця, 2021. – 22 с. (Укр. мов.)

32. Левичев Ю., Ворогушин Е. HR-аналітика : основні тенденції, визови, визови та практика, 2019 // Міжнародна аналітична компанія «Price Waterhouse Coopers» (PwC). [Електронний ресурс] – Режим доступу : https://www.pwc.ru/ru/publications/HR-analytics.pdf

33. Мазур І.І., Шапіро В.Д. та ін Управління проєктами (довідник для професіоналів). М.: «Вища школа», 2018.

34. Майерс Г. Мистецтво тестування програм / Г. Майерс, К. Сандлер, Т. Баджетт., 2019. – 272 с.

35. Марков А., Цирлов У. Управління ризиками – нормативний вакуум інформаційної безпеки [Електронний ресурс]. - Режим доступу: http://www.osp.ru/os/2007/08/4492873/, вільний. Інформаційна технологія. Методи та засоби забезпечення безпеки. Менеджмент ризику інформаційної безпеки.

36. Методичні рекомендації до виконання магістерської дипломної роботи для студентів ОПП "Комп'ютерні науки" спеціальності 122 "Комп'ютерні науки" другого (магістерського) рівня: [Електронне видання] / уклад. С.В.Мінухін, І.О.Ушакова, Д.Ю. Голубничий, О.В.Щербаков. – Х. : ХНЕУ ім. С. Кузнеця, 2021. – 59 с. (Укр. мов.)

37. Методичні рекомендації до оформлення звітів, курсових ктів та дипломних робіт (ктів) для студентів спеціальності 121 "Інженерія програмного забезпечення", 122 "Комп'ютерні науки", 126 "Інформаційні системи і технології": [Електронне видання] / уклад. І.О.Ушакова, Г.О. Плеханова, О.М. Беседовський. – Х. : ХНЕУ ім. С. Кузнеця, 2021. – 48 с.

38. Організація. Посібник. Управління. (Методологія та філософія оргуправницької діяльності. Курс лекцій/з архіву Г.П. Щедровицького. Т.5). М., 2020 – 288 с.

39. Основи професійних знань. Національні вимоги до компетенції спеціалістів. - М.: Вид-во "Консалтингове Агентство "КУБС Груп - Кооперація, Бізнес-Сервіс", 2021.

40. Палеха Ю.І., Горбань Ю.І. Інформаційний бізнес : підручник / Ю.І. Палеха, Ю.І. Горбань – К.: Вид-во Ліра-К. 2015. – 492 с.

41. Первушин Т.Л. Стратегія розвитку підприємства з урахуванням ризиків. Економіка та управління народним господарством: збірка статей.

42. Первушина Т.Л. Ризики управління: монографія. Красноярськ: Сиб-ГТУ, 2020. 3. Первушина Т.Л. Аналіз ризиків підприємств хіміко-лісового комплексу. Економіка та управління в сучасних умовах

43. Покровський М.А. Основи управління проєктами Навчальний посібник. За ред. Фалько С.Г. М.: Вид-во МДТУ ім. Баумана, 2018, 104 с.

44. Посібник до Зводу знань з управління проєктами. Третє видання (Посібник PMBOK)/. Американський національний стандарт ANSI/PMI 99-001-2019.

45. Савін Р. Тестування крапка ком / Роман Савін., 2018. – 312 с.

46. Середа Г. В. Гейміфікація в менеджменті персоналу: зарубіжний та український досвід. Економіка і організація управління. 2017. Вип. 4. С. 216–223. [Електронний ресурс] – Режим доступу :http://nbuv.gov.ua/UJRN/eiou_2018_4_22.

47. Ступаненко П.Є. Вища математика у заняттях і завданнях: У 2 т. навч. посіб. М: Вища. шк., 2018.

48. Управління проєктами. Основи професійних знань. Національні вимоги до компетенції спеціалістів. – М.: Вид-во «Консалтингове Агентство «КУБС Груп – Кооперація, Бізнес-Сервіс», 2019.

49. Управління проєктами. У 2 т.: пров. з ним. - СПб.: Видавничий дім "Бізнес - преса", 2019.-258 с.

50. Щедровицький Г.П. Організація. Посібник. Управління. (Орг. мислення: ідеологія, методологія, технологія. Курс лекцій/з архіву Г.П. Щедровицького. Т.4). М.: «Шлях», 2020 – 384 с.

51. «PersonPro 2.0» htp та «PersonPro 2.0 та SQL» Renaisc [Електронний ресурс] – Режим доступу: Oracle http://personpro.ami.ua

52. 20 базових HR-метрик, котрі допоможуть виміряти ефективність [Електронний ресурс] – Режим доступу: https://hurma.work/ru/blog/20-bazovyh-hr-metrik-kotorye-pomogut-izmerit-effektivnost-raboty-kompanii/

53. Mondore, S., Douthitt, S., Carson, M. Maximizing the impact and effectiveness of HR Analytics to drive business outcomes. People & Strategy, Vol. 34, 2011. 20–27 pp

54. Oracle Mangemt Human підприємства Resources та Analyzer Вісник [Електронний ресурс] – Режим доступу: http://www.oracle.com

55. SAP HRuman Вид Resources управління Management систем System Хмельницького [Електронний ресурс] – Режим доступу: http://www.sap.com