МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Савченко А.С.

"____"_____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТРА"

ЗА СПЕЦІАЛІЗАЦІЄЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)"

Тема: "Управління ризиками при розробці програмних продуктів"

Виконавець: Алиєв Сеймур Ханпута огли

Керівник: к.т.н., доцент Харченко Олександр Григорович

Нормоконтролер: _____ Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, компютерної,та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань 12 «Інформаційні технології»

Спеціальність 122 «Комп'ютерні науки»

Спеціалізація «Інформаційні управляючі системи та технології (за галузями)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

(А.С. Савченко)

"_____" _____2020

З А В Д А Н Н Я

на виконання дипломної роботи студента

Алиєв Сеймур

(П.І.Б. випускника)

1. Тема роботи: «Управління ризиками при розробці програмних продуктів затверджена наказом ректора від "02.10.2020" за №1891.

2. Термін виконання роботи: з 05.10.2020 по 31.12.2020

3. Вихідні данні до роботи: проблеми розробки програмних проектів .Основні причини провалів при розробці проектів . Результати емпіричних досліджень по визначенню основних причин зривів проектів , та вплив не врахованих ризиків на ці результати . Математична формалізація процесів.

4. Зміст пояснювальної записки: Основні поняття управління програмними проектами. Аналіз планування і моніторинг ризиків при управлінні розробкою програмних проектів. Математична формалізація процесів при моніторингу, та управлінні ризиками. Основні правила визначення ризиків для конкретних програмних продуктів. Включення задачі управління ризиками в процес управління проектом на основі моделей якості.

5. Перелік обов'язкового ілюстративного матеріалу: Категорії ризиків за моделлю SEI, атрибути ризиків, формалізація моделі SEI , модифікація моделі якості ISO 9126 для задачі управління ризиками, життєвий цикл розробки програм на основі моделі якості,
UML-моделі засобу підтримки керування ризиками, таблиці, рисунки, діаграми, графіки, а також слайди презентації доповіді у PowerPoint.

6. Календарний план

№ з/п	Завдання	Термін виконання	Підпис керівника
1	Аналіз літератури з управління програмними проектами,та управління ризиками.	05.10.2020 – 10.10.2020	
2	Розробка та затвердження плану дипломної роботи.	11.10.2020 – 15.10.2020	
3	Огляд методів , та технологій класифікації ризиків , при розробці програмних продуктів .	16.10.2020 – 18.10.2020	
4	Математична формалізація задачі управління ризиками при забезпеченні вимог якості.	19.10.2020 – 29.10.2020	
5	Розробка алгоритму та UM моделей засобу підтримки управління ризиками.	30.10.2020 – 10.11.2020	
6	Опрацювання матеріалів, структуризація та формування розділів роботи.	11.11.2020 – 30.11.2019	
7	Технічне оформлення пояснювальної записки.	01.12.2020 – 10.11.2019	
8	Підготовка до захисту дипломної роботи.	10.12.2020 – 16.12.2020	

7. Консультація з окремого(мих) розділу(ів) роботи:

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв

8. Дата видачі завдання_____

Керівник дипломної роботи _____ Харченко О. Г.

Завдання прийняв до виконання _____

(підпис випускника) (ПІБ)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Управління ризиками при розробці програмних продуктів”: складається із вступу, трьох розділів, загальних висновків, списку використаних джерел і має 80 сторінок основного тексту, 1 лістинг коду; 14 рисунків та 9 таблиць. Список використаних джерел містить 13 найменувань і займає 2 сторінки. Загальний обсяг роботи 84 сторінки.

Мета роботи: Метою дипломної роботи магістра є дослідження методів і засобів управління ризиками при розробці автоматизованих систем керування для підвищення якості кінцевого продукту Для досягнення зазначеної мети у магістерській роботі поставлено та розв’язано наступні задачі: аналіз наукових публікацій щодо джерел виникнення, аналізу та управління ризиками при розробці автоматизованих систем керування; визначення потенційних ризиків при розробці автоматизованих систем керування; обґрунтування моделі SEI для опису ризиків при створенні автоматизованих систем керування; розробка методу управління ризиками на етапах життєвого циклу автоматизованих систем керування; інтеграція розробленого методу керування ризиками у загальний процес управління проектами на основі моделей якості ISO/IEC 9126; розробка програмного засобу управління ризиками на основі розробленого методу.

Об’єкт дослідження: Об’єктом дослідження є процес управління ризиками при розробці автоматизованих систем керування,. Предметом дослідження є методи і засоби управління ризиками при розробці автоматизованих систем керування. Щоб вирішити поставлені задачі було використано наступні методи: синтез під час визначення ризиків при розробці автоматизованих систем керування; моделювання і системний аналіз під час побудови моделі ризиків автоматизованих систем керування; проектування та програмування під час реалізації програмного засобу

підтримки процесу управління ризиками при створенні автоматизованих систем керування. Розроблений в роботі програмний комплекс показує можливості раннього оцінювання архітектурних рішень на етапі проектування системи.

Можливі напрямки розвитку роботи пов’язані з розширенням можливості варіативних архітектур, додавання інших методів оцінювання та розширенням загальної функціональності.

Ключові слова: УПРАВЛІННЯ РИЗИКАМИ, РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЖИТТЄВИЙ ЦИКЛ ПРОДУКТУ, МАТЕМАТИЧНА ФОРМАЛІЗАЦІЯ ЗАДАЧ УПРАВЛІННЯ РИЗИКАМИ, МОДЕЛІ ЯКОСТІ, РОЗРОБКА ВИМОГ, АНАЛІЗ ЖИТТЄВОГО ЦИКЛУ, ОЦІНЮВАННЯ РИЗИКІВ ПРОГРАМИХ СИСТЕМ, ІНФОРМАЦІЙНА СИСТЕМА.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

АСК - автоматизована система керування

ЖЦ - життєвий цикл

ПЗ – програмне забезпечення

ІС – інформаційна система

ЗМІСТ

ВСТУП

Сучасні тенденції розвитку технологій проектування автоматизованих систем керування спрямовані на максимальне задоволення кінцевого споживача в максимально короткі терміни та з максимальним прибутком. Однак рівень якості виконання проектів все ж залишається не дуже високим, що пов’язано з недосконалістю існуючих методів і засобів керування ризиками. Дослідженню методів і засобів керування ризиками при проектуванні автоматизованих систем присвячено ряд наукових публікацій як українських, так і закордонних вчених. Серед українських вчених потрібно відмітити праці Глушкова, Ющенко, Зіля та ін. Серед закордонних – праці Демарко, Фатрел, Фентон та ін. Однак, комплексного підходу до управління ризиками з врахуванням особливостей сучасних гнучких методологій розробки програмного забезпечення у них не наведено. Тому актуальними задачами при розробці автоматизованих систем керування є розробка методів і засобів управління ризиками для підвищення якості як процесів проектування, так і готової системи, а також зниження імовірності настання ризикових подій та власне ризиків.

Наукова новизна одержаних результатів: Наукова новизна одержаних результатів полягає в наступному: уперше розроблено метод управління ризиками на основі формалізованої моделі SEI, що дало змогу реалізувати процеси ідентифікації, аналізу та моніторингу ризиків при розробці автоматизованих систем керування. уперше визначено множину ризиків та класифіковано їх за категоріями моделі SEI, що дало змогу структурувати та уніфікувати їх та спростити процес управління ними. уперше модифіковано моделі якості стандарту ISO/IEC 9126 шляхом додавання ризиків при описі атрибутів, що дало змогу оцінювати та керувати ризиками, пов’язаними з конкретним атрибутом якості автоматизованої системи керування.

Управління ризиками

В даному випадку ризик – це негативні події, що мають негативний характер та негативно впливають на результат розробки, аналізують збитки та недоліки процесів та продуктів, що є наслідком дефектів під час проектування вимог до продукту, неякісним обгрунтуванням проекту(-ів) технічного завдання, а також при подальших етапах розробки, реалізації та протягом всього ЖЦ програмного забезпечення. Треба розуміти, що ризики – це ймовірнісні події, що можуть виникнути, а можуть і ні. Тому заради успішності реалізації програмного проекту важливе управління ризиками, що охоплює весь ЖЦ програмного забезпечення. Управління ризиками – зводиться до процесу прийняття та виконання рішень з управління, які спрямовані на зменшення шансу того, що виникнуть події несприятливого характеру та на те, щоб змінімізувати потенційні втрати, які викликані неякісною реалізацією; це систематичні процеси з ідентифікації, аналізу та прийняття рішень, що мінімізують негативні наслідкі появи ризикових ситуацій і, в той самий час, максимізують ймовірність та наслідкі появи позитивних подій. Управління ризиками дає повну картину внутрішніх та зовнішніх причин, що є впливовими для проекту та можуть спричинити невдачу. Аналіз ризиків робиться на основі сформованого плану. Головна мета управління ризиками – це знаходження та контроль факторів, які нечасто виникають та спричиняють варіації розроблюємого проекту. Є різні моделі управління ризиками, найбільш поширений з них це модель (SEI), яка включає в себе вимоги стандартів, а також популярні «кращі практики» з управління ризиками. Дана модель розроблена у вигляді рекомендацій текстового формату, формалізованого методу управління ризиками немає, що призводить до вільного використання та переосмислення даної моделі, тому основною задачею авторів є деталізація та формалізація методу управління ризиками при розробленні програмного забезпечення.

РОЗДІЛ 1. УПРАВЛІННЯ ПРОЕКТАМИ

1.1 Сучасний стан та проблеми розробки програмних проектів

Аналітичні дослідження, які за останні деілька років, які проводились зарубіжними аналітиками, дали не дуже позитивні результати у галузі розроблення (ПЗ). Дивлячись на стрімкий розвиток технологій програмування, все ще залишається велика кількість програмних проектів та продуктів, успіх яких не можна вважати доволі успішним. Успішно реалізованим програмним проектом можна розуміти вчасне виконання програмного проекту у зв'язці з виділеним на нього бюджетом та реалізованість запланованих функцій та можливостей програмного продукту (рис.1.1.)

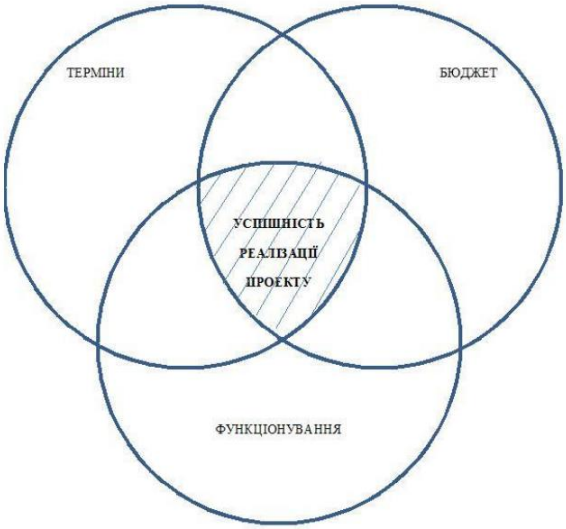
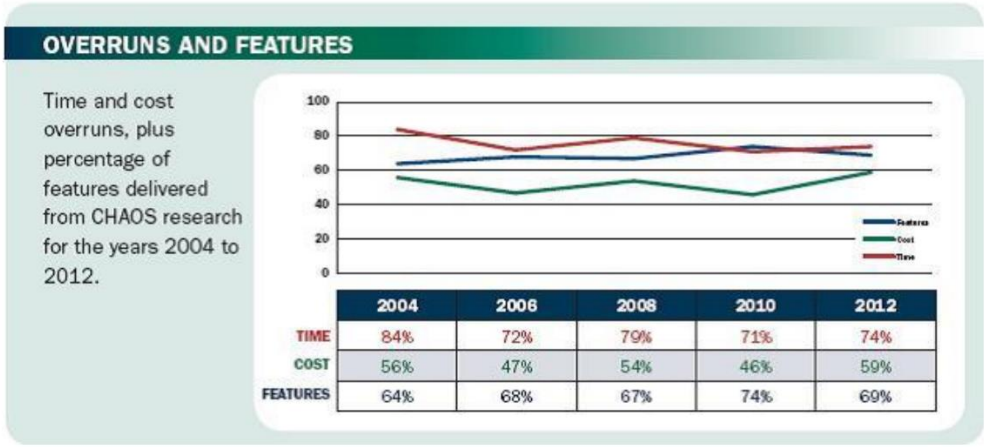


Рис. 1.1. Візуалізація поняття «успішність реалізації програмного проекту»

Статистика успішних реалізацій ПП за 1994-2012 р., представлена у The Standish Group International (CHAOS report), дає змогу відслідкувати приріст кількості успішних проектів та зменшення кількості невдалих проектів у 2010-2012 р., але у цей час частина проблемних продуктів та проектів (які не

задовольняють з одного або декількох критеріїв оцінки успішності проекту) є відносно сталою величиною у 2006-2012 роках і її частина – 42-46% від усіх проектів. Аналітика критеріїв успішності проектів за витратами, термінами та функціоналом для проблемних програмних проектів у 2004-2012 роках, за даними, представлена на рис.1.2. Статистика зображує також, що тільки 16% проектів завершуються “успішно” середніми компаніями у визначені термін та бюджет. У великих компаній статистика значно гірша – лише 9% проектів вкладаються у визначені терміни та бюджет. Програмні проекти, створені великими американськими компаніями, забезпечені 42% функціональності, від запланованої на початку створення. Малы компанії більш успішні в цьому плані: 78.4% проектів реалізовують 74.2% своєї запланованої функціональності.



	2004	2006	2008	2010	2012
TIME	84%	72%	79%	71%	74%
COST	56%	47%	54%	46%	59%
FEATURES	64%	68%	67%	74%	69%

Рис. 1.2. Статистика по перевитратах, перевищенню термінів та відсутності необхідної функціональності для проблемних програмних проектів

Дослідження відомої міжнародної консалтінгової компанії McKinsey & Company спільно з Оксфордським Університетом, проведені у 2010 році, також показали, що половина масштабних програмних проектів з загальним бюджетом більше ніж 15000000$ в витрачають перевищують заплановані витрати, зокрема: середнє перевищення витрат на проект складає 66%, середнє перевищення часу реалізації проекту - 33%, а середнє число втрат прибутку - 17% (рис.1.3., 1.4.).

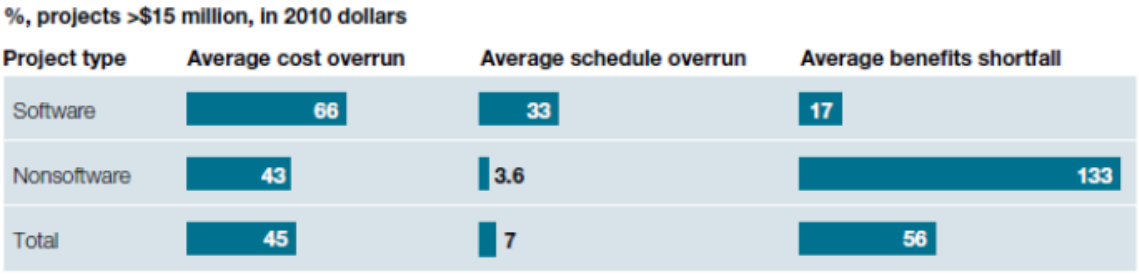


Рис. 1.3. Продуктивність програмних проектів різних типів

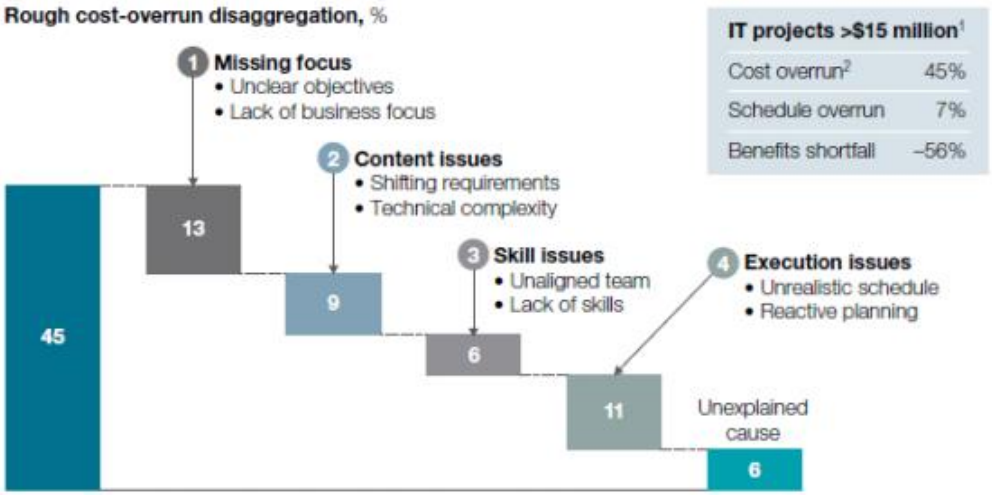


Рис. 1.4. Чотири групи питань, які викликають більшість невдач програмних проектів

Є багато прикладів великих фінансових втрат через некомпетентне управління ризиками на стадіях роботи над програмним продуктом), як відмова на етапі розроблення від програмного проекту Virtual Case File (VCF), який розроблювався ФБР США у 2000-2005 р., загальна вартість витрат на який перейшла відмітку 170 мільйонів доларів США; 2) відмова від ПЗ Міністерства оборони США з назвою Enterprise Resource Planning після розробки близько семи років роботи над проектом у 2005 р., коли бюджет проекту вже перейшов 1 мільярд доларів США; 3) невдача програмного проекту уряду Великобританії у 2011 році, метою якого було створення електронних медичних записів громадян, з загальними витратами у 18,7 мільярдів доларів США.

Отже, розробка програмного забезпечення може завершитися і невдало, про успішність проекту говорять такі критерії як бюджет, функціонал і термин виконання. Актуальною задачею є управління ризиками

при розробці програмного забезпечення. Управління ризиками дає змогу проаналізувати прогавини проекту та його недоліки, підрахувати збитки, прорахувати терміни та функціонал ПЗ. Задачею розробників є скорочення та усунення ризиків. Зниження ризиків проекту підвищує його якість, ефективність та результативність.

Проблеми управління програмними проектами вперше з’явились в 60-х - початку 70-х років, коли провалились багато великих проектів по розробці програмних продуктів. Були зафіксовані затримки в створенні ПЗ, вони були ненадійними, витрати на розробку в кілька разів перевищували початкові оцінки, створені програмні системи часто мали низькі показники продуктивності. Провали цих проектів обумовлювалися не тільки некомпетентністю керівників і програмістів. Навпаки, в цих великих пошукових проектах брали участь люди, рівень кваліфікації яких був явно вище середнього. Причини провалів були в тих підходах, які використовувалися в управлінні проектами. Застосовувана методика була заснована на досвіді управління технічними проектами і засвідчила свою неефективність при розробці програмного забезпечення. Тут важливо зрозуміти різницю між професійною розробкою ПЗ і аматорським програмуванням. Необхідність управління програмними проектами випливає з того 'сумного' факту, що процес створення професійного ПЗ завжди є суб'єктом бюджетної політики організації, де воно розробляється, і має тимчасові обмеження. Робота керівника програмного проекту за великим рахунком полягає в тому, щоб гарантувати виконання цих бюджетних і тимчасових обмежень з урахуванням бізнес-цілей організації щодо розроблюваного ПЗ. Менеджери проектів покликані спланувати всі етапи розробки програмного продукту. Вони також повинні контролювати хід виконання робіт і дотримання всіх необхідних стандартів. Постійний контроль за ходом виконання робіт необхідний для того, щоб процес

розробки не виходив за тимчасовий і бюджетні обмеження. Добре управління не гарантує успішного завершення проекту, але погане управління обов'язково призведе до його провалу. Це може виразитися в затримці термінів здачі готового ПО, в перевищенні зазначеної вартості проекту і в невідповідності готового ПО специфікації вимог. Керівники програмних проектів виконують таку ж роботу, що і керівники технічних проектів. Разом з тим процес розробки ПО істотно відрізняється від процесів реалізації технічних проектів, що породжує певні складності в управлінні програмними проектами. Наведемо невеликий список цих відмінностей.

1. Програмний продукт нематеріальний. Менеджер суднобудівного проекту або проекту спорудження будівлі бачить результати виконання свого проекту. Якщо реалізація проекту відстає від графіка, це також видно на власні очі, так як частина конструкції не завершена. На противагу цьому програмне забезпечення нематеріальне. Його не можна побачити чи поторкати. Менеджер програмного проекту не бачить процес 'зростання' розробки ПО. Він може покладатися лише на документацію, яка фіксує процес розробки програмного продукту.

2. Не існує стандартних процесів розробки ПЗ. На сьогоднішній день не існує чіткої залежності між процесом створення ПЗ і типом створюваного програмного продукту. Інші технічні дисципліни мають тривалу історію, процеси розробки технічних виробів багаторазово випробувані і перевірені. Процеси створення більшості технічних систем добре вивчені. Вивченням же процесів створення ПЗ фахівці займаються тільки кілька останніх років. Тому поки не можна точно передбачити, на якому етапі процесу розробки ПО можуть виникнути проблеми, що загрожують всьому програмному проекту.

3. Великі програмні проекти – це часто 'одноразові' проекти. Великі програмні проекти, як правило, значно відрізняються від проектів, реалізованих раніше. Тому, щоб зменшити невизначеність у плануванні проекту, керівники проектів повинні мати дуже великий практичний досвід

Але постійні технологічні зміни в комп'ютерній техніці і комунікаційному устаткуванні знецінюють попередній досвід. Знання та навички, накопичені досвідом, можуть бути не затребувані в новому проекті.

Перераховане вище може призвести до того, що реалізація проекту вийде з тимчасового графіка або перевищить бюджетні асигнування. Програмні системи часто виявляються новинками як в 'ідеологічному', так і в технічному плані. Технічні проекти, які є інноваційними (наприклад, нова транспортна система), також часто порушують терміни їх виконання робіт. Тому, передбачаючи можливі проблеми в реалізації програмного проекту, слід завжди пам'ятати, що багатьом з них властиво виходити за рамки тимчасових і бюджетних обмежень.

Управління програмними проектами - тема вельми велика, яку неможливо висвітлити в одній главі. Тому в цьому розділі дається тільки введення в цю тему і розглядається три основних процеси, які виконуються в рамках управління проектами, а саме: планування проекту, складання графіка робіт і управління ризиками. У частині VI представлені інші аспекти управління проектами розробки ПЗ, в тому числі управління персоналом, оцінювання вартості проекту і управління якістю ПЗ.

1.2 Процеси управління

Неможливо описати і стандартизувати всі роботи, що виконуються менеджером проекту зі створення ПЗ. Ці роботи має велике значення та залежать від організації, де виконується розробка ПЗ, і від типу створюваного програмного продукту. Але в будь-якому випадку більшість менеджерів відповідальні за виконання всіх або деяких з наведених нижче процесів управління.

- Написання пропозицій по створенню ПЗ.
- Планування та складання графіка робіт зі створення ПЗ.
- Оцінювання вартості проекту.
- Контроль за ходом виконання робіт.
- Підбір персоналу.
- Написання звітів.

Перша стадія програмного проекту може складатися з написання пропозицій щодо реалізації цього проекту. Пропозиції повинні містити опис цілей проектів і способів їх досягнення. Вони також зазвичай включають в себе оцінки фінансових і тимчасових витрат на виконання проекту. При необхідності тут можуть наводитися обґрунтування з приводу передачі проекту на виконання сторонньої організації або команді розробників.

Написання пропозицій - дуже відповідальна робота, тому що для багатьох організацій питання про те, чи буде проект виконуватися самою організацією або розроблятися за контрактом сторонньою компанією, є критичним. Неможливо дати будь-яких рекомендацій з написання пропозицій, багато що тут залежить від досвіду менеджера. Ейрон (Агоп) вважає цю роботу менеджера однієї з найважливіших серед інших виконуваних ним робіт.

На етапі планування проекту визначаються процеси, етапи, та отримані на кожному з них результати, які повинні привести до виконання проекту. Реалізація цього плану призведе до досягнення цілей проекту. Визначення вартості проекту прямо пов'язане з його плануванням, оскільки тут оцінюються ресурси, потрібні для виконання плану.

Контроль за ходом виконання робіт (моніторинг проекту) – це безперервний процес, що триває протягом усього терміну реалізації проекту. Менеджер повинен Постійно відслідковувати хід реалізації проекту та порівнювати фактичні і планові показники виконання робіт з їх вартістю. Хоча багато організацій мають механізми формального моніторингу робіт, досвідчений менеджер може скласти ясну картину про стадії розвитку проекту просто шляхом неформального спілкування з розробниками.

Неформальний моніторинг часто допомагає виявити потенційні проблеми, які в явному вигляді можуть виявитися пізніше. Наприклад, щоденне обговорення ходу виконання робіт може виявити окремі недоробки в створюваному програмному продукті. Замість очікування звітів, в яких буде відображений факт 'пробуксовки' графіка робіт, менеджер може обговорити з фахівцями намічені програмістські проблеми і не допустити зриву графіка робіт.

Протягом реалізації проекту зазвичай відбувається кілька формальних контрольних перевірок ходу виконання робіт зі створення ПЗ. Такі перевірки повинні дати загальну картину ходу реалізації проекту в цілому і показати, наскільки вже розроблена частина ПО відповідає цілям проекту.

Час виконання великих програмних проектів може займати кілька років. Протягом цього часу цілі і наміри організації, яка замовила програмний проект, можуть істотно змінитися. Може виявитися, що розробляється програмний продукт став вже непотрібним або вихідні вимоги до створюваного ПО просто застаріли і їх необхідно кардинально міняти. У такій ситуації керівництво організації-розробника може прийняти рішення

про припинення розробки ПО або про зміну проекту в цілому з тим, щоб врахувати змінилися цілі і наміри організації-замовника.

Керівники - менеджери проектів зазвичай зобов'язані самі підбирати виконавців для своїх проектів. В ідеальному випадку професійний рівень виконавців повинен відповідати тій роботі, яку вони будуть виконувати в ході реалізації проекту. Однак у багатьох випадках менеджери повинні покладатися на команду розробників, яка далека від ідеальної. Така ситуація може бути викликана наступними причинами.

1. Бюджет проекту не дозволяє залучити висококваліфікований персонал. У такому випадку за меншу плату залучаються менш кваліфіковані фахівці.

2. Бувають ситуації, коли неможливо знайти фахівців необхідної кваліфікації як в самій організації-розробника, так і поза нею. Наприклад, в організації 'кращі люди' можуть бути вже зайняті в інших проектах.

3. Організація хоче підвищити професійний рівень своїх працівників. В цьому випадку вона може залучити до участі в проекті недосвідчених або недостатньо кваліфікованих працівників, щоб вони придбали необхідний досвід і повчилися у більш досвідчених фахівців.

Таким чином, майже завжди підбір фахівців для виконання проекту має певні обмеження і не є вільним. Разом з тим необхідно, щоб хоча б кілька членів групи розробників мали кваліфікацію і досвід, достатні для роботи над даним проектом. В іншому випадку неможливо уникнути помилок в розробці ПЗ.

Менеджер проекту зазвичай зобов'язаний посилати звіти про хід його виконання як замовнику, так і підрядним організаціям. Це повинні бути короткі документи, засновані на інформації, що витягується з докладних звітів про проект. У цих звітах має бути та інформація, яка дозволяє чітко оцінити ступінь готовності створюваного програмного продукту.

1.3 Планування проекту

Ефективне управління програмним проектом безпосередньо залежить від правильного планування робіт, необхідних для його виконання. План допомагає менеджеру передбачати проблеми, які можуть виникнути на будь-яких етапах створення ПЗ, і розробити превентивні заходи для їх попередження або вирішення. План, розроблений на початковому етапі проекту, розглядається всіма його учасниками як керівний документ, виконання якого повинно привести до успішного завершення проекту. Цей первісний план повинен максимально докладно описувати всі етапи реалізації проекту.

Структура плану створення ПЗ розглядається в розділі 1.2.1. Тут лише зазначимо, що, крім розробки плану проекту, на менеджера лягає обов'язок розробки інших видів планів. Ці види планів коротко описані в табл. 1.1 і детально обговорюються у відповідних розділах книги.

План	**Опис**
План якості	Описує стандарти та заходи з підтримки якості розроблюємого ПЗ (розділ 24)
План атестації	Описує способи, ресурси та перелік рабіт, необходних для атестації програмної системи (глава 19)
План управління конфігурацією	Описує структуру та процесу управління конфигурацією (розділ 29)
План супроводу ПЗ	Пропонує план заходів, потрібний для супроводу ПЗ у процесі його эксплуатації, а також розрахунок вартості супроводу та необхідні для цього ресурси (глава 27)
План з управління персоналом	Описує заходи, направлені на зростання кваліфікації членів команди розробників (розділ 22)

Таблиця 1.1. Види планів

У лістингу 1.1. показаний процес планування створення ПО у вигляді псевдокоду. Тут зроблено акцент на тому, що планування - це ітераційний процес. Оскільки в процесі виконання проекту постійно надходить нова інформація, план повинен регулярно переглядатися. Важливими чинниками, які повинні враховуватися при розробці плану, є фінансові та ділові зобов'язання організації. Якщо вони змінюються, ці зміни також повинні знайти відображення в плані робіт.

```
Визначення проектних обмежень
Первісна оцінка параметрів проекту
Визначення етапів виконання проекту і контрольних відміток
    While поки проект не завершиться, чи не буде зупинений loop
    Складання графіка робіт
    Початок виконання робіт
    Очікування закінчення чергового етапу робіт
    Відстеження ходу виконання робіт
    Перегляд оцінок параметрів проекту
    Зміна графіка робіт
    Перегляд проектних обмежень
        If (виникла проблема) then
    Перегляд технічних або організаційних параметрів проекту
    end if
    end loop
```

Лістинг 1.1. Процес планування проекта

Процес планування починається з визначення проектних обмежень (тимчасові обмеження, можливості готівкового персоналу, бюджетні обмеження і т.д.). Ці обмеження повинні визначатися паралельно з оцінюванням проектних параметрів, таких як структура і розмір проекту, а

також розподілом функцій серед виконавців. Потім визначаються етапи розробки і те, які результати (документація, прототипи, підсистеми або версії програмного продукту) повинні бути отримані після закінчення цих етапів. Далі починається циклічна частина планування. Спочатку розробляється графік робіт по виконанню проекту або дається дозвіл на продовження використання раніше створеного графіка. Після цього (зазвичай через 2-3 тижні) проводиться контроль виконання робіт і відзначаються розбіжності між реальним і плановим ходом робіт.

Далі, у міру надходження нової інформації про хід виконання проекту, можливий перегляд первинних оцінок параметрів проекту. Це, в свою чергу, може привести до зміни графіка робіт. Якщо в результаті цих змін порушуються терміни завершення проекту, повинні бути переглянуті (і погоджені із замовником ПО) проектні обмеження.

Звичайно, більшість менеджерів проектів не думають, що реалізація їх проектів пройде гладко, без будь-яких проблем. Бажано описати можливі проблеми ще до того, як вони проявлять себе в ході виконання проекту. Тому краще складати 'песимістичні' графіки робіт, ніж 'оптимістичні'. Але, звичайно, неможливо побудувати план, що враховує всі, в тому числі випадкові, проблеми і затримки виконання проекту, тому і виникає необхідність періодичного перегляду проектних обмежень і етапів створення програмного продукту.

1.3.1. План проекта

План проекту повинен чітко показати ресурси, необхідні для реалізації проекту, поділ робіт на етапи і часовий графік виконання цих етапів. У деяких організаціях план проекту складається як єдиний документ, що містить всі види планів, описаних вище. В інших випадках план проекту описує тільки технологічний процес створення ПЗ. У такому плані

обов'язково присутні посилання на плани інших видів, але вони розробляються окремо від плану проекту.

План, структуру якого представлено нижче, належить саме до останнього типу планів. Деталізація планів проектів дуже різниться в залежності від типу розроблюваного програмного продукту і організації-розробника. Але в будь-якому випадку більшість планів містять такі розділи.

1. Введення. Короткий опис цілей проекту і проектних обмежень (бюджетних, тимчасових і т.д.), які важливі для управління проектом.

2. Організація виконання проекту. Опис способу підбору команди розробників і розподіл обов'язків між членами команди.

3. Аналіз ризиків. Опис можливих проектних ризиків, ймовірності їх прояви і стратегій, спрямованих на їх зменшення. Тема управління ризиками розглянута в розділі 1.4.

4. Апаратні і програмні ресурси, необхідні для реалізації проекту. Перелік апаратних засобів і програмного забезпечення, необхідного для розробки програмного продукту. Якщо апаратні засоби потрібно закуповувати, наводиться їх вартість спільно з графіком закупівлі і поставки.

5. Розбиття робіт на етапи. Процес реалізації проекту розбивається на окремі процеси, визначаються етапи виконання проекту, наводиться опис результатів ('виходів') кожного етапу і контрольні позначки.

6. Графік робіт. У цьому графіку відображаються залежності між окремими процесами (етапами) розробки ПО, оцінки часу їх виконання і розподіл членів команди розробників по окремих етапах.

7. Механізми моніторингу та контролю за ходом виконання проекту. Описуються надаються менеджером звіти про хід виконання робіт, терміни їх надання, а також механізми моніторингу всього проекту.

План повинен регулярно переглядатися в процесі реалізації проекту. Одні частини плану, наприклад графік робіт, змінюються часто, інші більш стабільні. Для внесення змін до плану потрібна спеціальна організація документопотока, що дозволяє відстежувати ці зміни.

1.3.2. Контрольні відмітки етапів роботи

Менеджеру для організації процесу створення програмного забезпечення та управління їм необхідна інформація. Оскільки саме програмне забезпечення невловимі, ця управлінська інформація може бути отримана тільки у вигляді документів, що відображають виконання чергового етапу розробки програмного продукту. Без цієї інформації не можна судити про ступінь готовності створюваного продукту, неможливо оцінити зроблені витрати або змінити графік робіт.

При плануванні процесу визначаються контрольні позначки - віхи, що відзначають закінчення певного етапу робіт. Для кожної контрольної позначки створюється звіт, який надається керівництву проекту. Ці звіти не повинні бути великими об'ємними документами; Вони повинні підводити короткі підсумки закінчення окремого логічно завершеного етапу проекту. Етапом не може бути, наприклад, 'Написання 80% коду програм', оскільки неможливо перевірити завершення такого 'етапу': крім того, подібна інформація практично марна для управління, оскільки тут не відображається зв'язок цього 'етапу' з іншими етапами створення ПО.

Зазвичай після закінчення основних великих етапів, таких як розробка специфікації, проектування і т.п., замовнику ПЗ надаються результати їх виконання, так звані контрольні проектні елементи. Це може бути документація, прототип програмного продукту, закінчені підсистеми ПЗ і т.д. Контрольні проектні елементи, що надаються замовникові ПЗ,

можуть збігатися з контрольними відмітками (точніше, з результатами виконання будь-якого етапу). Але зворотне твердження не так. Контрольні позначки - це внутрішні проектні результати, які використовуються для контролю за ходом виконання проекту, і вони, як правило, не надаються замовникові ПЗ.

Для визначення контрольних відміток весь процес створення ПЗ повинен бути розбитий на окремі етапи з зазначеним 'виходом' (результатом) кожного етапу. Наприклад, на рис. 1.2.2.1 показані етапи розробки специфікації вимог у разі, коли для її перевірки використовується прототип системи, а також представлені вихідні результати (контрольні позначки) кожного етапу. Тут контрольними проектними елементами є вимоги і специфікація вимог.

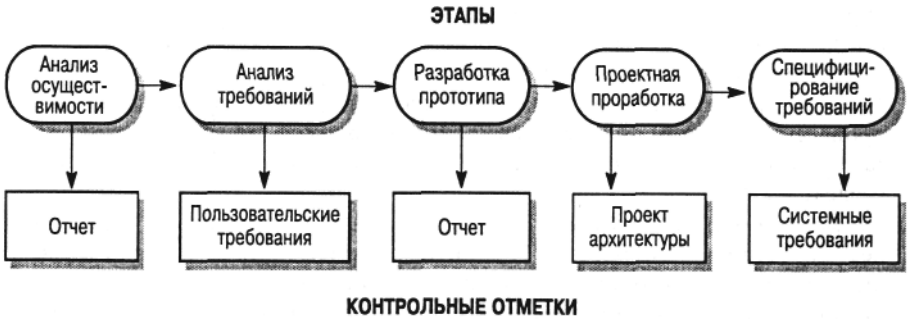


Рис. 1.5.. Етапи процесу розробки специфікації

1.3.3. Графік робіт

Складання графіка - одна з найвідповідальніших робіт, виконуваних менеджером проекту. Тут менеджер оцінює тривалість проекту, визначає ресурси, необхідні для реалізації окремих етапів робіт, і представляє їх (етапи) у вигляді узгодженої послідовності. Якщо даний проект подібний до раніше реалізованому, то графік робіт останнього проекту можна взяти за основу для даного проекту. Але потім слід врахувати, що на окремих етапах

нового проекту можуть використовуватися методи і підходи, відмінні від використаних раніше.

Якщо проект є інноваційним, початкові оцінки тривалості і необхідних ресурсів напевно будуть занадто оптимістичними, навіть якщо менеджер спробує передбачити всі можливі несподіванки. З цієї точки зору проекти створення ПЗ не відрізняються від великих інноваційних технічних проектів. Нові аеропорти, мости і навіть нові автомобілі, як правило, з'являються пізніше спочатку оголошених термінів їх здачі або надходження на ринок, що спричинено є несподівано виникли проблеми і труднощі. Саме тому графіки робіт необхідно постійно оновлювати у міру надходження нової інформації про хід виконання проекту.

У процесі складання графіка (рис. 1.6.) весь масив робіт, необхідних для реалізації проекту, розбивається на окремі етапи і оцінюється час, потрібний для виконання кожного етапу. Зазвичай багато етапів виконуються паралельно. Графік робіт повинен передбачати це і розподіляти виробничі ресурси між ними найбільш оптимальним чином. Брак ресурсів для виконання будь-якого критичного етапу – часта причина затримки виконання всього проекту.

Тривалість етапів зазвичай повинна бути не менше тижня. Якщо вона буде менше, то виявиться нижче точності тимчасових оцінок етапів, що може привести до частого перегляду графіка робіт. Також доцільно (в аспекті управління проектом) встановити максимальну тривалість етапів, що не перевищує 8 або 10 тижнів. Якщо є етапи, які мають велику тривалість, їх слід розбити на етапи меншою тривалості.

При розрахунку тривалості етапів менеджер повинен враховувати, що виконання будь-якого етапу не обійдеться без великих або маленьких проблем і затримок. Розробники можуть допускати помилки або затримувати свою роботу, техніка може вийти з ладу або апаратні або програмні засоби підтримки процесу розробки можуть надійти із запізненням. Якщо проект

інноваційний і технічно складний, це стає додатковим фактором появи непередбачених проблем і збільшення тривалості реалізації проекту в порівнянні з початковими оцінками.

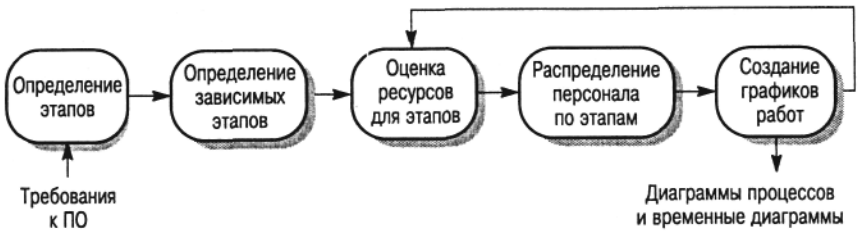


Рис. 1.6. Процес складання графіка робіт

Крім тимчасових витрат, менеджер повинен розрахувати інші ресурси, необхідні для успішного виконання кожного етапу. Особливий вид ресурсів - це команда розробників, залучена до виконання проекту. Іншими видами ресурсів можуть бути необхідне вільний дисковий простір на сервері, час використання будь-якого спеціального обладнання і бюджетні кошти на відрядження персоналу, що працює над проектом. Більш детально види і методи оцінювання необхідних ресурсів розглядаються в главі 23.

Існує хороше емпіричне правило: оцінювати часові витрати так, як ніби нічого непередбаченого і 'поганого' не може трапитися, потім збільшити ці оцінки для обліку можливих проблем. Можливі, але важко прогнозовані проблеми істотно залежать від типу і параметрів проекту, а також від кваліфікації і досвіду членів команди розробників. До вихідних розрахункових оцінок можна додавати 30% на можливі проблеми і потім ще 20%, щоб бути готовим до того, що неможливо передбачити.

Графік робіт за проектом зазвичай представляється у вигляді набору діаграм і графіків, що показують розбиття проектних робіт на етапи, залежності між етапами і розподіл розробників по етапах. Ці діаграми розглядаються в наступному розділі. Зазначу, що в даний час існує багато різних програмних засобів підтримки управління проектами, наприклад Microsoft Project.

1.4. Часові и мережеві діаграми

Тимчасові і мережеві діаграми корисні для подання графіка робіт. Тимчасова діаграма показує час початку і закінчення кожного етапу і його тривалість. Мережева діаграма відображає залежності між різними етапами проекту. Ці діаграми можна створити автоматично за допомогою програмних засобів підтримки управління на основі інформації, закладеної в базі даних проекту.

Розглянемо етапи якогось проекту, представлені в табл. 1.2., з якої, зокрема, видно, що етап Т3 залежить від етапу Т1. Це означає, що етап Т1 повинен завершитися раніше, ніж почнеться етап Т3. Наприклад, на етапі Т1 проводиться компонентний аналіз створюваного програмного продукту, а на етапі Т3 - проектування системи.

Я	**Я**	**Залежність**
Т1	8	
Т2	15	
Т3	15	Т1(М1)
Т4	10	
Т5	10	Т2, Т4 (М2)
Т6	5	Т1,Т2(М3)
Т7	20	Т1(М1)
Т8	25	Т4 (М5)
Т9	15	Т3, Т6 (М4)
Т10	15	Т5, Т7 (М7)
Т11	7	Т9 (Мб)
Т12	10	Т11(М8)

Таблиця 1.2. Етапи проекта

На основі наведених значень тривалості етапів і залежності між ними будується мережевий графік послідовності етапів (рис. 1.7.). На цьому графіку видно, які роботи можуть виконуватися паралельно, а які повинні виконуватися послідовно один за одним. Етапи позначені прямокутниками. Контрольні позначки і контрольні проектні елементи показані у вигляді овалів і позначені (як і в табл. 1.2.) буквою М з відповідним номером. Дати на даній діаграмі відповідають початку виконання етапів. Мережеву діаграму слід читати зліва направо і зверху вниз.

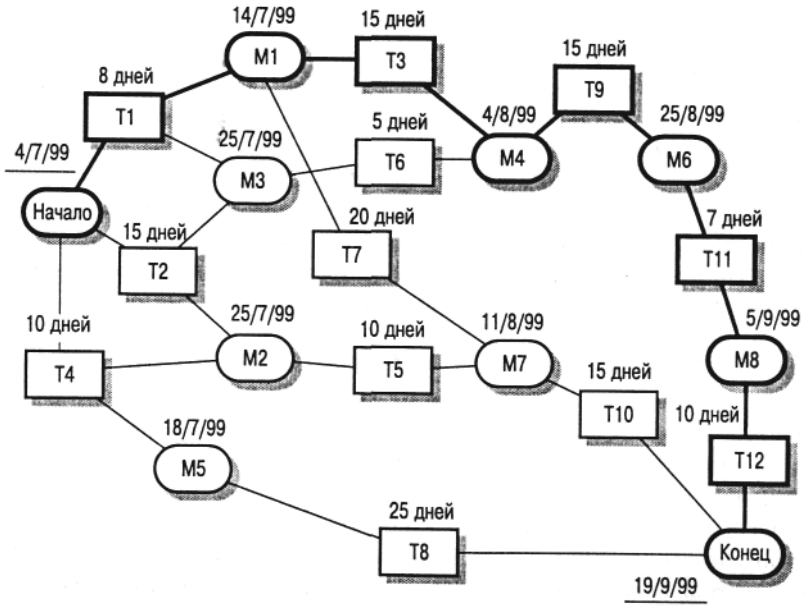


Рис. 1.7. Мережева діаграма этапів

Якщо для створення мережевої діаграми використовуються програмні засоби підтримки управління проектом, кожен етап повинен закінчуватися контрольної відміткою. Черговий етап може початися тільки тоді, коли буде отримана контрольна відмітка (яка може залежати від кількох попередніх етапів). Тому в третьому стовпці табл. 1.2. наведені контрольні позначки; Вони будуть досягнуті тільки тоді, коли буде завершено етап, в рядку якого поміщена відповідна контрольна відмітка.

Будь етап не може початися, поки не виконані всі етапи на всіх шляхах, що ведуть від початку проекту до даного етапу. Наприклад, етап Т9 не може початися, поки не будуть завершені етапи Т3 і Т6. Відзначимо, що в даному випадку досягнення контрольної позначки М4 говорить про те, що ці етапи завершені.

Мінімальний час виконання всього проекту можна розрахувати, підсумувавши в мережевий діаграмі тривалості етапів на самому довгому шляху * від початку проекту до його закінчення (це так званий критичний шлях). У нашому випадку тривалість проекту становить 11 тижнів або 55 робочих днів. На рис. 1.8. критичний шлях показаний більш товстими лініями, ніж інші шляхи. Таким чином, загальна тривалість реалізації проекту залежить від етапів робіт, що знаходяться на критичному шляху. Будь-яка затримка в завершенні будь-якого етапу на критичному шляху призведе до затримки всього проекту.

(* Довжина шляху тут вимірюється не кількістю етапів на шляху, а сумарною тривалістю цих етапів)

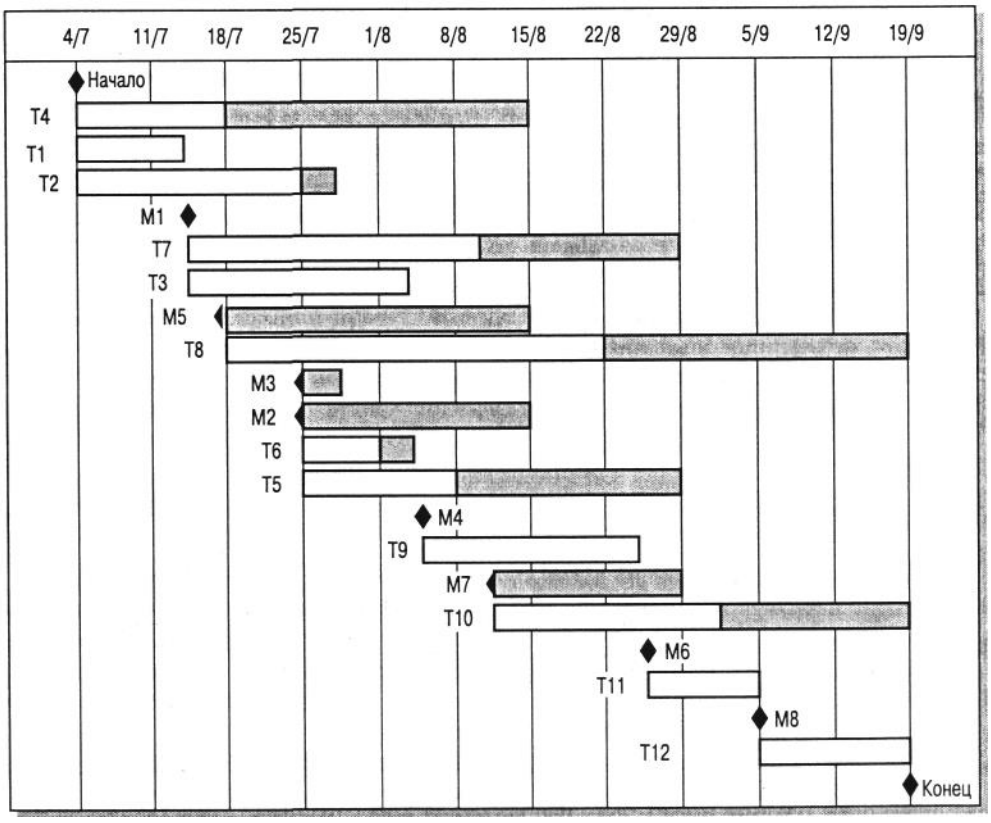


Рис. 1.8. Тимчасова діаграма тривалості етапів

Затримка в завершенні етапів, що не входять в критичний шлях, не впливає на тривалість усього проекту до тих пір, поки сумарна тривалість цих етапів (з урахуванням затримок) на якомусь шляху не перевищить тривалості робіт на критичному шляху. Наприклад, затримка етапу Т8 на термін, менший 20 днів, ніяк не впливає на загальну тривалість проекту. На рис. 1.9. представлена тимчасова діаграма, на якій показані можливі затримки на кожному етапі.

Мережева діаграма дозволяє побачити в залежності етапів значимість того чи іншого етапу для реалізації всього проекту. Увага до етапів критичного шляху часто дозволяє знайти способи їх зміни з тим, щоб скоротити тривалість всього проекту. Менеджери використовують мережеву діаграму для розподілу робіт.

На рис. 1.9. показано інше уявлення графіка робіт. Це тимчасова діаграма (іноді звана по імені її винахідника діаграмою Гантта) може бути побудована програмними засобами підтримки процесу управління. Вона показує тривалість виконання кожного етапу і можливі їх затримки (показані затіненими прямокутниками), а також дати початку і закінчення кожного етапу. Етапи критичного шляху не мають затінених прямокутників; Це означає, що затримка із завершенням даних етапів призведе до збільшення тривалості всього проекту.

Подібно розподілу часу виконання етапів, менеджер повинен розрахувати розподіл ресурсів по етапах, зокрема призначити виконавців на кожен етап. У табл. 1.3. наведено розподіл розробників на кожен етап, представлений на рис. 1.9.

Этап	**Виконавець**
Т1	Джейн

T2	Анна
T3	Джейн
T4	Фред
T5	Мери
T6	Анна
T7	Джим
T8	Фред
T9	Джейн
T10	Анна
T11	Фред
T12	Фред

Таблица 1.3. Розподіл працівників по етапах

Наведена таблиця може бути використана програмними засобами підтримки процесу управління для побудови тимчасової діаграми зайнятості співробітників на певних етапах робіт (рис. 1.9.). Персонал не зайнятий в роботі над проектом весь час його реалізації. Протягом періоду незайнятості співробітники можуть бути у відпустці, працювати над іншими проектами, проходити навчання і т.д.

У великих організаціях звичайно працює багато фахівців, які задіюються у проекті в міру необхідності. Звичайно, такий підхід може створити певні проблеми для менеджерів проектів. Наприклад, якщо фахівець зайнятий в проекті, який затримується, це може створити прямі складності для інших проектів, де він також повинен брати участь.

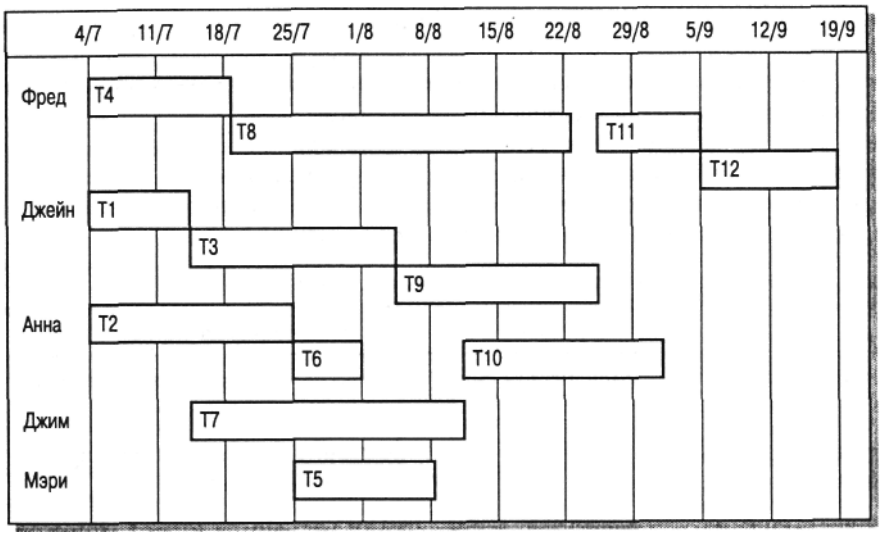


Рис. 1.9.. Тимчасова діаграма розподілу працівників по етапах

Початковий графік робіт неминуче містить якісь помилки або недоробки. У міру реалізації проекту розраховані оцінки тривалості виконання етапів робіт повинні порівнюватися з реальними термінами виконання цих етапів. Результати порівняння повинні використовуватися в якості основи для перегляду графіка робіт ще не реалізованих етапів проекту, зокрема для того, щоб спробувати зменшити тривалість етапів критичного шляху.

1.5 Управління ризиками

Важливою частиною роботи менеджера проекту є оцінка ризиків, які можуть вплинути на графік робіт або на якість створюваного програмного продукту, і розробка заходів щодо запобігання ризиків. Результати аналізу ризиків повинні бути відображені в плані проекту. Визначення ризиків та розробка заходів щодо зменшення їх впливу на хід виконання проекту називається управлінням ризиками.

Спрощено ризик можна розуміти як імовірність прояву будь-яких несприятливих обставин, що негативно впливають на реалізацію проекту.

Ризики можуть загрожувати проекту в цілому, створюваного програмного продукту або організації-розробнику. Можна виділити три типи ризиків.

1. Ризики для проекту, які впливають на графік робіт або ресурси, необхідні для виконання проекту.
2. Ризики для продукту, що розробляється., Що впливають на якість або продуктивність розроблюваного програмного продукту.
3. Бізнес-ризики, які стосуються організації-розробнику або постачальникам.

Звичайно, ці типи ризиків можуть перетинатися. Наприклад, якщо досвідчений програміст залишає проект, це буде ризиком для проекту (оскільки затримується термін здачі готового продукту), ризиком для продукту (так як новий програміст, який замінив пішов, може виявитися не надто досвідченим і зробити помилки в програмі) і бізнес-ризиком (Оскільки затримка даного проекту може негативно вплинути на майбутні ділові контакти між замовником і організацією-розробником).

Конкретні типи ризиків, які можуть вплинути на даний проект, залежать від виду створюваного програмного продукту і від організаційного оточення, де реалізується програмний проект. Разом з тим багато типів ризиків здатні вплинути на будь-які програмні проекти, ці ризики наведені в табл. 1.4.

Ризик	Тип ризику	Опис ризику
Плинність розробників	Ризик для проекта	Досвідчені розробники залишають проект до його завершення
Зміна в управлінні організацією	Ризик для проекта	Організація змінює свої пріоритети в управлінні проектом
Неготовність апаратних засобів	Ризик для проекта	Апаратні засоби, які необхідні для проекта, не поступили у відведений час, або не готові для використання
Зміна вимог	Ризик для проекта та для розробляємого продукту	Поява великої кількості непередбачених змін у вимогах, пропонованих до розробляємого ПЗ
Затримка в розробці специфікації	Ризик для проекта та для розробляємого продукта	Специфікація основних інтерфейсів підсистем не дійшли до розробників у відповіді з графіком робіт
Недооцінка розміру розробляємої системи	Ризик для проекту та для розробляємого продукту	Розмір системи значно перевищує первісну оцінку
Недостатня эффективність CASE-засобів	Ризик для розробляємого продукту	CASE-засоби, призначені для підтримки проекта, виявились менш ефективними, ніж очікувалось
Зміни в технології розробки ПЗ	Бізнес-ризик	Основні технології побудови програмной системи замінюються новими
Поява конкуруючего		На ринці програмних продуктів до закінчення

програмного продукту	Бізнес-ризик	проекта зявилась конкуруюча програмна система

Таблиця 1.4. Можливі ризики програмних проектів

Схема процесу управління ризиками показана на рис. 1.10. Цей процес складається з чотирьох стадій.

1. Визначення ризиків. Визначаються можливі ризики для проекту, для продукту, що розробляється і бізнес-ризики.
2. Аналіз ризиків. Оцінюється ймовірність і послідовність появи ризикових ситуацій.
3. Планування ризиків. Плануються заходи щодо запобігання ризиків або мінімізації їх впливу на проект.
4. Моніторинг ризиків. Постійне оцінювання ймовірностей ризиків і виконання заходів щодо пом'якшення наслідків прояви ризикових ситуацій.

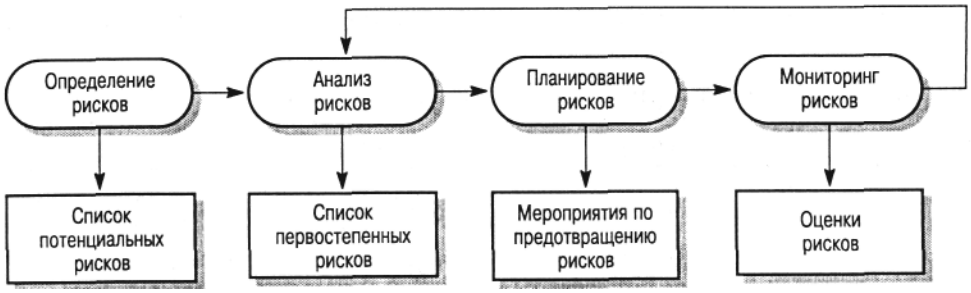


Рис. 1.10. Процес управління ризиками

Процес управління ризиками, як і інші процеси планування, є ітераційним, виконуваних протягом усього терміну реалізації проекту. Спочатку розробляються плани управління ризиками, потім постійно відстежується ситуація навколо реалізації проекту. При надходженні нової

інформації про можливі ризики заново проводиться аналіз ризиків і першорядну увагу приділяється новим ризикам. У міру надходження нової інформації також змінюються плани заходів щодо запобігання і пом'якшення ризиків.

Результати процесу управління ризиками документуються у вигляді планів управління ризиками. Вони повинні включати опис можливих проектних ризиків, їх аналіз та перелік заходів, необхідних для управління ризиками.

1.6 Ідентифікація ризиків

Процес управління ризиками, як і інші процеси планування, є ітераційним, виконуваних протягом усього терміну реалізації проекту. Спочатку розробляються плани управління ризиками, потім постійно відстежується ситуація навколо реалізації проекту. При надходженні нової інформації про можливі ризики заново проводиться аналіз ризиків і першорядну увагу приділяється новим ризикам. У міру надходження нової інформації також змінюються плани заходів щодо запобігання і пом'якшення ризиків.

Результати процесу управління ризиками документуються у вигляді планів управління ризиками. Вони повинні включати опис можливих проектних ризиків, їх аналіз та перелік заходів, необхідних для управління ріскамі.Определеніе ризиків - перша стадія процесу управління ризиками. На цій стадії описуються ризики, які можуть проявитися при реалізації проекту. В принципі на цій стадії не повинна оцінюватися ймовірність і значимість ризиків, але на практиці малоймовірні ризики з незначними наслідками зазвичай відкидаються відразу.

Визначення ризиків може виконуватися в режимі командної роботи з використанням підходу 'мозковий штурм' або ґрунтуватися на досвіді

менеджера. При визначенні ризиків може допомогти наведений нижче список можливих категорій ризиків.

1. Технологічні ризики. Є наслідком програмних і апаратних технологій, на основі яких розробляється система.

2. Ризики, пов'язані з персоналом. Пов'язані з членами команди розробників.

3. Організаційні ризики. Є наслідком організаційного оточення, в якому виконується проект.

4. Інструментальні ризики. Пов'язані з використовуваними CASE-засобами та іншими засобами підтримки процесу створення ПО.

5. Ризики, пов'язані з системними вимогами. Виявляються при зміні вимог, що пред'являються до розроблюваної системі.

6. Ризики оцінювання. Пов'язані з оцінюванням характеристик програмної системи і ресурсів, необхідних для реалізації проекту.

У табл. 1.5. представлені деякі приклади, які підпадають під кожну з описаних категорій ризиків. Результатом етапу визначення ризиків буде довгий список можливих ризиків, які можуть вплинути на розроблювальний програмний продукт, проект або організацію-розробника.

Категория рисков	**Примеры рисков**
Технологічні ризики	База даних, яка використовується в програмній системі, не забезпечує обробку очікуваного об’єма транзакцій.
	Програмні компоненти, які використовуються повторно, мають дефекти, що обмежують їх функціональні можливості
Ризики, звязані з персоналом	

	Неможливо підібрати працівників з необхідним професійним рівнем.
Організаційні ризики	Провідний розробник захворів в найкритичніший час.
	Неможливо організувати необхідне навчання персоналу
Інструментальні ризики	В організації, яка виконує розробку ПО, відбулася реорганізація, в результаті чого змінилися пріоритети в управлінні проектом.
	Фінансові труднощі в організації привели до зменшення бюджету проекту
	Програмний код, що генерується CASE-засобами, не ефективний.
Ризики, пов’язані з системними вимогами	CASE-засобу неможливо інтегрувати з іншими засобами підтримки проекту
	Зміни вимог призводять до значних повторним робіт з проектування системи.
	Первісна нечітке формулювання призначених для користувача вимог призвела до значних змін системних вимог, що проявилися на пізніх стадіях розробки проекту
Ризики оцінки	Недооцінки часу виконання проекту.
	Швидкість виявлення дефектів в системі нижче раніше запланованої.
	Розмір системи значно перевищує спочатку розрахований

Таблиця 1.5. Категорії ризиків

1.7 Аналіз ризиків

При аналізі для кожного певного ризику підраховується ймовірність його прояви і збиток, який він може нанести. Не існує простих методів виконання аналізу ризиків - в значній мірі він заснований на думці і досвіді менеджера. Не претендуючи на виняткову точність, можна навести таку шкалу ймовірностей ризиків і їх наслідків.

1. Імовірність ризику вважається дуже низькою, якщо вона має значення менше 10%; Низькою, якщо її значення від 10 до 25%; Середньої при значеннях від 25 до 50%; Високою, якщо значення коливається від 50 до 75%; Дуже високою при значеннях більше 75%.

2. Можливі збитки від ризикових ситуацій можна поділити на катастрофічний, серйозний, терпимий і незначний.

Результати аналізу ризиків повинні бути представлені у вигляді таблиці ризиків, упорядкованих за ступенем можливого збитку. У табл. 1.6. наведено упорядкований список ризиків, описаних в табл 1.6.. Там же вказані ймовірності цих ризиків. Тут ймовірності ризиків і ступінь шкоди від них вказані довільно. На практиці для їх визначення необхідна детальна інформація про проект, технології створення ПЗ, команді розробників і про саму організацію.

Ризик	**Ймовірність**	**Степень ущерба**
Фінансові труднощі в організації привели до зменшення бюджету проекту	Низька	Катастрофічна
Неможливо підібрати працівників з потрібними професійним рівнем	Висока	Катастрофічна

Провідний розробник захворів в найкритичніший час	Середня	Серйозна
Програмні компоненти, що використовуються повторно, мають дефекти, що обмежують їх функціональні можливості	Середня	Серйозна
Зміни вимог призводять до значних повторним робіт з проектування системи	Середня	Серйозна
В організації, яка виконує розробку ПО, відбулася реорганізація, в результаті чого змінилися пріоритети в управлінні проектом	Висока	Серйозна
База даних, яка використовується в програмній системі, не забезпечує обробку очікуваного обсягу транзакцій	Середня	Серйозна
Недооцінки часу виконання проекту	Висока	Серйозна
CASE-засобу неможливо інтегрувати з іншими засобами підтримки проекту	Висока	Терпима
Первісна нечітке формулювання призначених для користувача вимог призвела до значних змін системних вимог, що проявилися на пізніх стадіях розробки проекту	Середня	Терпима
Неможливо організувати необхідне навчання персоналу	Середня	Терпима
Швидкість виявлення дефектів в системі нижче раніше запланованої	Середня	Терпима
Розмір системи значно перевищує спочатку розрахований	Висока	Терпима

Програмний код, що генерується CASE-засобами, неефективний	Середня	Незначна	

Таблиця 1.6. Перелік ризиків після проведення їх аналізу

Звичайно, як ймовірні ризиків, так і можливі збитки від них повинні переглядатися при надходженні додаткової інформації про ці ризики і в міру реалізації заходів з управління ними. Тому подібні таблиці ризиків повинні перероблятися на кожній ітерації процесу управління ризиками.

Після проведення аналізу ризиків визначаються найбільш значущі ризики, які потім відслідковуються протягом всього терміну виконання проекту. Визначення цих значущих ризиків залежить від їх ймовірностей і можливого збитку. У загальному випадку завжди відслідковуються ризики з катастрофічними наслідками, а також ризики з серйозною шкодою, значення ймовірності яких вище середнього.

Кількість ризиків, які необхідно відстежувати, залежить від конкретного проекту. Це може бути п'ять ризиків, а може - п'ятнадцять. Але, звичайно, кількість ризиків, за якими проводиться моніторинг, має бути доступним для огляду. Велика кількість відслідковуються ризиків зажадає величезної кількості інформації, що збирається. Зі списку ризиків, представлених в табл. 1.6., для моніторингу слід відібрати всі вісім ризиків, які можуть призвести до катастрофічних і серйозних наслідків.

1.8 Планування ризиків

Планування полягає у визначенні стратегії управління кожним значущим ризиком, відібраним для моніторингу після аналізу ризиків. Тут також не існує загальноприйнятих підходів для розробки таких стратегій - багато грунтується на 'чуття' і досвіді менеджера проекту. У табл. 1.7. показані можливі стратегії управління основними ризиками, наведеними в табл. 1.6.

Ризик	**Стратегія**
Фінансові проблеми організації	Підготувати короткий документ для керівництва організації, що показує важли даного проекту для досягнення фінансових цілей організації
Проблеми некваліфікованого персоналу	Попередити замовника про потенційних труднощі і можливу затримку проекту розглянути питання про покупку компонентів системи
Хвороби персоналу	Реорганізувати роботу команди розробників таким чином, щоб обов'язки і робо членів команди перекривали один одного, внаслідок цього розробники будуть і розуміти завдання, що виконуються іншими співробітниками
Дефектні системні компоненти	Замінити потенційно дефектні системні компоненти покупними компонентами гарантують якість роботи
Зміни вимог	Спробувати визначити вимоги, найбільш ймовірно схильні змінам; В структурі системи не відображати детальну інформацію
Реорганізація компанії-розробника	Підготувати короткий документ для керівництва компанії, що показує важливіс даного проекту для досягнення фінансових цілей компанії
Недостатня продуктивність бази даних	Розглянути можливість покупки більш продуктивної бази даних
Недооцінки часу виконання проекту	Розглянути питання про покупку системних компонентів, дослідити можливіст використання генератора програмного коду

Таблиця 1.7. Стратегії управління ризиками

Існує три категорії стратегій управління ризиками.

1. Стратегії запобігання ризикам. Згідно з цими стратегіям слід проводити заходи, що знижують імовірність прояву ризиків. Прикладом може служити стратегія виключення потенційно дефектних компонентів, описана в табл. 1.7.

2. мінімізаційні стратегії. Спрямовані на зменшення можливих збитків від ризиків. Прикладом служить стратегія зменшення шкоди від хвороби членів команди розробників (див. Табл. 1.7.).

3. Планування 'аварійних' ситуацій. Згідно з цими стратегіям необхідно мати план заходів, які слід виконати в разі прояву ризикової ситуації. У табл. 1.7 це стратегія поведінки при виникненні фінансових проблем у організації-розробника.

1.9 Моніторинг ризиків

Моніторинг ризиків полягає в регулярному перерахунку ймовірностей ризиків і збитків, який вони можуть завдати. Для цього необхідно постійно відслідковувати фактори, які впливають на ймовірність ризиків і можливі збитки. Ці фактори залежать від типів ризику. У табл. 1.8. наведені ознаки, які допомагають визначити тип ризику.

Тип ризику	**Ознаки**
Технологічні ризики	Затримки в постачанні обладнання або програмних засобів підтримки процесу створення ПО, численні документовані технологічні проблеми
Ризики, пов'язані з персоналом	Низьке моральний стан персоналу, натягнуті відносини між членами команди розробників, низька якість виконаної роботи
Організаційні ризики	Розмови серед персоналу про пасивність і недостатню компетентність

	вищого керівництва організації
Інструментальні ризики	Небажання розробників використовувати програмні засоби підтримки, несхвальні відгуки про CASE-засобах, запити на більш потужні інструментальні засоби
Ризики, пов'язані з системними вимогами	Необхідність перегляду багатьох системних вимог, невдоволення замовника ПО
Ризики оцінювання	Зміни графіка робіт, численні звіти про порушення графіка робіт

Таблиця 1.8. Ознаки ризиків

Моніторинг ризиків повинен бути безперервним процесом, який відслідковує хід виконання заходів з управління ризиками, при цьому кожен основний ризик повинен розглядатися.

Висновки до розділу 1

В данному розділі було розглянуто сучасний стан та проблеми розробки програмних проектів. Описані процеси, етапи та плани проектування. Дана теоретична та практична інформація по ідентифікації, аналізу та моніторингу ризиків, що дозволить нам надалі математично формалізувати задачі управління ризиками.

РОЗДІЛ 2 МАТЕМАТИЧНА ФОРМАЛІЗАЦІЯ ЗАДАЧІ УПРАВЛІННЯ РИЗИКАМИ.

Моніторинг ризиків передбачає у постійному перерахунку ймовірностей ризиків і збитків, що можуть з'явитись. Тому важливо постійно враховувати фактори, які можуть вплинути на ймовірність ризиків і можливі збитки. Ці фактори залежать від типів ризику. У табл. 1.8.1 приведены чинники, які допоможуть визначити до якої групи належить ризик.

2.1 Ідентифікація ризиків

Відповідно до моделі SEI, існує 18 найбільш розповсюджених причин ризиків, тому приведемо деякі джерела ризиків у виді слідуючої множини:

$$RS = \{rs_1,...,rs_{18}\}, \quad (2.1)$$

де rs_i – ймовірні джерела ризиків ($i = 1..18$).

Джерелами ризиків - це технічні ризики представлені множиною:

$$TRS = \{rs_1,...,rs_7\}, \quad (2.2)$$

$TRS \subset RS$ – підмножина джерел технічних ризиків, де:

rs_1 – характеристики функціональності;

rs_2 – характеристики якості;

rs_3 – характеристики надійності;

rs_4 – застосовність;

rs_5 – часова продуктивність;

rs_6 – супроводжуваність.

rs_7 – повторне використання компонентів.

Додатково, можна виділити підмножину ризиків, пов’язаних з бюджетом, виділеним на реалізацію проекта:

$$CRS = \{rs_8,...,rs_{10}\}, \quad (2.3)$$

$CRS \subset RS$ - підмножина джерел вартісних ризиків, де:

rs_8 – обмеження сумарного бюджету;

rs_9 – недоступна вартість проекту;

rs_{10} – низький ступінь реалізму при оцінюванні витрат на проект.

На успішність та якість проекту впливають також ризики, пов’язані з процесом планування проекту. Формально їх можна представити у вигляді підмножини:

$$PRS = \{rs_{11},...,rs_{13}\} \quad (2.4)$$

$PRS \subset RS$ – підмножина джерел планових ризиків, де:

rs_{11} – властивості та можливості гнучкості зміни планів;

rs_{12} – можливості порушення встановлених термінів етапів життєвого циклу;

rs_{13} – низький ступінь реалізму планів та етапів життєвого циклу.

Наступну підмножину ризиків при виконанні проектів передбачаючих розробку АСК становлять ризики, пов’язані із процедурами, які є допоміжними та організаційними, а також з процесами ЖЦ. Така підмножина може бути подана у наступному виді:

$$MRS = \{rs_{14},...,rs_{18}\} \quad (2.5)$$

$MRS \subset RS$ – підмножина джерел ризиків процесів та процедур управління проектом, де:

rs_{14} – стратегія проекту;

rs_{15} – планування проекту;

rs_{16} – оцінювання проекту;

rs_{17} – документування проекту;

rs_{18} – прогнозування проекту.

Аналіз специфікації вимог до ПЗ (SRS) дає змогу встановити ймовірні джерела ризиків. Введемо слідуючий регламент встановлення джерел ризиків:

1 Якщо у SRS відсутні або наявні нереалістичні чи неоціненні функціональні характеристики, то $rs_1 = 1$, інакше $rs_1 = 0$;

2 Якщо відсутні або існують нереалістичні чи неоціненні характеристики якості, то $rs_2 = 1$, інакше $rs_2 = 0$;

3 Якщо відсутні або наявні нереалістичні чи неоціненні характеристики надійності, то $rs_3 = 1$, інакше $rs_3 = 0$;

4 Якщо у SRS відсутні рекомендації щодо майбутньої застосовності ПЗ, то $rs_4 = 1$, інакше $rs_4 = 0$;

5 Якщо відсутні або існують нереалістичні чи неоціненні характеристики часової продуктивності, то $rs_5 = 1$, інакше $rs_5 = 0$;

6 Якщо у SRS відсутні рекомендації щодо майбутньої супроводжуваності ПЗ, то $rs_6 = 1$, інакше $rs_6 = 0$;

7 Якщо відсутні або існують нереалістичні чи неоціненні пропозиції щодо повторного використання компонентів, то $rs_7 = 1$, інакше $rs_7 = 0$;

8 Якщо у SRS присутні обмеження сумарного бюджету, то $rs_8 = 1$, інакше $rs_8 = 0$;

9 Якщо у SRS вказано недоступну вартість проекту, то $rs_9 = 1$, інакше $rs_9 = 0$;

10 Якщо у SRS присутній низький ступінь реалізму при оцінюванні витрат на проект, то $rs_{10} = 1$, інакше $rs_{10} = 0$;

11 Якщо відсутні або наявні нереалістичні чи неоціненні властивості та можливості гнучкості зміни планів, то $rs_{11} = 1$, інакше $rs_{11} = 0$;

12 Якщо відсутні або наявні нереалістичні чи неоціненні можливості порушення встановлених термінів етапів життєвого циклу, то $rs_{12} = 1$, інакше $rs_{12} = 0$;

13 Якщо у SRS присутній низький ступінь реалізму планів та етапів життєвого циклу етапів життєвого циклу, то $rs_{13} = 1$, інакше $rs_{13} = 0$;

14 Якщо відсутня або наявна нереалістична чи неоціненна стратегія проекту, то $rs_{14} = 1$, інакше $rs_{14} = 0$;

15 Якщо відсутнє або наявне нереалістичне чи неоціненне планування проекту, то $rs_{15} = 1$, інакше $rs_{15} = 0$;

16 Якщо відсутнє або наявне нереалістичне чи неоціненне оцінювання проекту, то $rs_{16} = 1$, інакше $rs_{16} = 0$;

17 Якщо відсутнє документування проекту, то $rs_{17} = 1$, інакше $rs_{17} = 0$;

18 Якщо відсутнє або наявне нереалістичне чи неоціненне прогнозування проекту, то $rs_{18} = 1$, інакше $rs_{18} = 0$;

19 Якщо $(rs_1 =1) \cup (rs_2 =1) \cup (rs_3 =1) \cup (rs_4 =1) \cup (rs_5 =1) \cup (rs_6 =1) \cup (rs_7 =1)$, то мають місце технічні ризики;

20 Якщо $(rs_8 =1) \cup (rs_9 =1) \cup (rs_{10} =1)$, то мають місце вартісні ризики;

21 Якщо $(rs_{11} =1) \cup (rs_{12} =1) \cup (rs_{13}=1)$, то мають місце ризики планування;

22 Якщо $(rs_{14}=1)\cup(rs_{15}=1)\cup(rs_{16}=1)\cup(rs_{17}=1)\cup(rs_{18}=1)$, то мають місце ризики процесів та процедур управління проектом.

2.2 Ідентифікація потенційних ризикових подій

Ідентифікація ризику – це фіксація усіх чинників збудження та стурбованості, пов'язаною із проектом, і далі в сталому слідкуванні за іншими можливими побоюваннями. Справжня проблема це ризики, які не вдається ідентифікувати. Користуючись основними публікаціями у провідних галузях [4-7] сформулюємо множину можливих ризиків за вищезазначеними критеріями:

$i=\overline{1,11}$

$$TR=\{tr_1,...,tr_{11}\}, \tag{2.6}$$

де TR – множина технічних ризиків,

tr_1 – затримки у постачанні обладнання, необхідного для процесу створення автоматизованої системи;

tr_2 – затримки у постачанні програмних засобів, необхідних для підтримки процесу створення автоматизованої системи;

tr_3 – небажання розробників використовувати програмні засоби підтримки життєвого циклу;

tr_4 – відмова від CASE-засобів;

tr_5 – запити на більш потужні інструментальні засоби розроблення;

tr_6 – недостатня продуктивність баз(и) даних;

tr_7 – програмні компоненти, які використовуються повторно, мають дефекти та обмежені функціональні можливості;

tr_8 – неефективність програмного коду, генерованого CASE засобами;

tr_9 – неможливість інтеграції CASE-засобів з іншими засобами підтримки проекту;

tr_{10} – швидкість виявлення дефектів в системі нижче раніше запланованої;

tr_{11} – дефектні системні компоненти.

Множину ризиків, пов’язаних із бюджетом проекту можна представити у вигляді:

$$CR = \{cr_1,...,cr_6\}, \quad (2.7)$$

де cr_1 – недооцінка витрат на виконання проекту (надмірно низька вартість);

cr_2 – переоцінки витрат на виконання проекту (надмірно висока вартість);

cr_3 – фінансові ускладнення у компанії-розробника;

cr_4 – зменшення бюджету проекту під час його виконання;

cr_5 – висока вартість повторних робіт, необхідних через зміни вимог;

cr_6 – реорганізація компанії-розробника.

Множина потенційних ризикових подій з планування проекту представляється у вигляді множини:

$$PR = \{pr_1,..., pr_{10}\} \quad (2.8)$$

де pr_1 – зміни графіку робіт;

pr_2 – порушення графіку робіт;

pr_3 – необхідність зміни багатьох вимог;

pr_4 – необхідність великої кількості повторних робіт;

pr_5 – недооцінки часу виконання проекту;

pr_6 – переоцінки часу виконання проекту;

pr_7 – розмір АСК перевищує планований розмір;

pr_8 – розмір АСК значно менший за планований розмір;

pr_9 – поява на ринку аналогічної АСК до виходу розроблюваного АСК;

pr_{10} – поява на ринку більш конкурентоздатної АСК.

Ризикові події, пов’язані із процесами і процедурами управління проектом можна представити у вигляді:

$$MR = \{mr_1,...,mr_{16}\}, \tag{2.9}$$

де: $mr1$ – низький моральний стан персоналу;

mr_2 – низька взаємодія між членами колективу розробників;

mr_3 – пасивність керівника (менеджера) проекту;

mr_4 – недостатня компетентність керівника (менеджера) проекту;

mr_5 – незадоволеність замовника;

mr_6 – недостатня кількість фахівців з необхідним професійним рівнем;

mr_7 – хвороба провідного розробника в найкритичніший час;

mr_8 – одночасна хвороба декількох розробників;

mr_9 – неможливість організації необхідного навчання персоналу;

mr_{10} – зміна пріоритетів в управлінні проектом;

mr_{11} – недооцінка необхідної кількості розробників;

mr_{12} – переоцінка необхідної кількості розробників;

mr_{13} – надмірне документування проекту;

mr_{14} – недостатнє документування проекту;

mr_{15} – нереалістичне прогнозування результатів проекту;

mr_{16} – недостатній професійний рівень розробників.

Тоді множина (список) потенційних ризиків має вигляд: $R = \{r_1,...,r_{43}\}$, де $(r_1,...,r_{11}) \in TR$, $(r_{12},...,r_{17}) \in CR$, $(r_{18},...,r_{27}) \in PR$, $(r_{28},...,r_{43}) \in MR$, а, в свою чергу, $TR \subset R$, $CR \subset R$, $PR \subset R$, $MR \subset R$. Множина (список) потенційних ризиків конкретного програмного проекту має вигляд:

$$R_p = \{r_{1p},...,r_{43p}\} \qquad (2.10)$$

Введемо наступні правила визначення ризиків для конкретного програмного проекту:

1. Якщо можливі затримки у постачанні обладнання, необхідного для процесу створення ПЗ, то r_{1p}=1, інакше r_{1p}=0;

2. Якщо можливі затримки у постачанні програмних засобів, необхідних для підтримки процесу створення ПЗ, то $r_{2p} = 1$, інакше $r_{2p} = 0$;

Якщо у проектний колектив входять розробники з недостатнім професійним рівнем, то r_{43p}=1, інакше r_{43p}=0.

Сформуємо множину ризиків конкретного програмного проекту $ERp = \{er_{ip}\}$, де $i = 1..n$, n – кількість ризиків проекту, за наступними правилами:

1. Якщо r_{1p}=1 , то $er_{i\,p} = 1$, $i = i + 1$;

2. Якщо r_{2p}=1, то $er_{i\,p} = 1$, $i = i + 1$;

3. Якщо r_{43p}=1, то $er_{i\,p} = 1$, $i = i + 1$.

2.3 Аналіз ризиків

Встановлення ймовірностей ризиків (ймовірність того, що виникне ризикова подія). Для кожного з них з описаної множини ER_p розробники повинні визначити ймовірність настання рихику у певному діапазоні [0;1].

Множина ймовірностей ризиків має вигляд:

$$PRER_p = \{preri_p\}, \qquad (2.11)$$

де $i = 1..n$, n - кількість ризиків проекту.

Згідно з провідними галузевими публікаціями [4-7], можна сформулюти правила категоризації ризиків за їх шансом виникнення:

1. $\forall(prer_{ip} \in PRER_p)$, $\forall(er_{ip} \in ER_p)$: якщо $preri_{\,p} < 0{,}1$, то ймовірність ризику eri_p є дуже низькою;

2. $\forall(\ prer_{ip} \in PRER_p)$, $\forall(er_{ip} \in ER_p)$: якщо $(\ preri_p \geq 0{,}1) \wedge (\ preri_p < 0{,}25)$, то ймовірність ризику er_{ip} є низькою;

3. $\forall(prer_{ip} \in PRER_p)$, $\forall(er_{ip} \in ER_p)$: якщо $(\ preri_p \geq 0{,}25) \wedge (\ preri_p < 0{,}5)$, то ймовірність ризику eri_p є середньою;

4. $\forall(prer_{ip} \in PRER_p)$, $\forall(er_{ip} \in ER_p)$: якщо $(preri_p \geq 0{,}5) \wedge (preri_p < 0{,}75)$, то ймовірність ризику *eri p* є високою;

5. $\forall(prer_{ip} \in PRER_p)$, $\forall(er_{ip} \in ER_p)$: якщо $(preri_p \geq 0{,}75)$, то ймовірність ризику er_{ip} є дуже високою.

Визначення потенційних збитків від ризиків (ступінь загрози для проекту, якщо ризик виникне). Для кожного з них у множині ER_p, розробники повинні визначити розміри потенційних збитків від спричиненої події – в діапазоні [0;1] .

Множина збитків ризиків має наступний вигляд: $LRER_p = \{lrer_{ip}\}$, де $i = 1..n$, n – n-кількість ризиків проекту.

Встановлення розміру ризику (мат. сподівання збитків). Для кожного з ризиків множини ER_p потрібно визначити величину. Множина величин ризиків має наступний вигляд: $VRER_p = \{vrer_{ip}\}$, де $i = 1..n$, n – кількість ризиків проекту, $vrer_{ip} = prer_{ip} \cdot lrer_{ip}$.

Встановлювання рівнів пріоритету та ранжування ризиків за цими пріоритетами. Щоб встановити рівень пріоритету та ранжувати ризики потрібно знайти максимальний та мінімальний елемент множини $VRER_p$.

Максимальний елемент – це елемент $\max \in VRER_p$, для якого: $\forall vrer_{ip} \in VRER_p$: $(\max < vrer_{ip} \Rightarrow \max = vrer_{ip})$, де $i = 1..n$, n – кількість ризиків проекту.

Мінімальний елемент – це елемент $\min \in VRER_p$, для якого: $\forall vrer_{ip} \in VRER_p$: $(\min > vreri\ p \Rightarrow \min = vrer_{ip})$, де $i = 1..n$, n – кількість ризиків проекту.

Якщо розбити далі отриманий інтервал [min; max] на три інтервали:

- $[\min; \min + \frac{\max - \min}{3}]$;
- $[\min + \frac{\max - \min}{3}; \min + 2 \cdot \frac{\max - \min}{3}]$;
- $[\min + 2 \cdot \frac{\max - \min}{3}; \max]$.

То набір правил для визначення рівнів пріоритетів ризиків має наступний вигляд:

1 $\forall(vrer_{ip} \in VRER_p)$, $\forall(er_{ip} \in ER_p)$: якщо $(vrer_{ip} \geq \min) \wedge (vrer_{ip} < \min + \frac{\max - \min}{3})$, то: рівень пріоритету ризику er_{ip} – низький, $ler_p = er_p$ ($ler_{jp} \in LER_p$, $j = 1..m$, $LER_p = \{ler_{1p},...,ler_{mp}\}$ – множина ризиків з низьким пріоритетом, m – кількість ризиків з низьким пріоритетом), $j = j + 1$;

2 $\forall(vrer_{i\,p} \in VRER_p)$, $\forall(er_i\ p \in ER_p)$: якщо $(vrer_{ip} > \min + \frac{\max - \min}{3}) \wedge (vrer_{ip} \leq \min + 2 \cdot \frac{\max - \min}{3})$, то: рівень пріоритету ризику eri_p – середній, $mer_{k\,p} = er_{ip}$ ($mer_{kp} \in MER_p$, $k = 1..v$, $MER_p = \{mer_{1p},...,mer_{v\,p}\}$ – множина ризиків з середнім пріоритетом, v – кількість ризиків з середнім пріоритетом), $k = k + 1$;

3 $\forall(vrer_{i\,p} \in VRER_p)$, $\forall(er_{i\ p} \in ER_p)$: якщо $(vrer_{i\,p} > \min + 2 \cdot \frac{\max - \min}{3}) \wedge (vreri\ p \leq \max)$, то: рівень пріоритету ризику $er_{i\ p}$ – високий, $her_{g\ p} = eri_{\ p}$ ($her_{g\ p} \in HER_p$, $g = 1..O$, $HER_p = \{\ her_{1\ p}... \ her_{o\ p}\ \}$ – множина ризиків з високим пріоритетом, O – кількість ризиків з високим пріоритетом), $g = g + 1$.

У результаті застосування вищенаведених правил для встановлення рівня пріоритету ризиків до всіх ризиків проекту маємо множину HER_p першочергових ризиків, множину MER_p другочергових ризиків та множину LER_p ризиків останньої, третьої, черги конкретного програмного проекту, які пропонуються членам проектного колективу в якості допомоги при обранні заходів із зменшення або усунення ризиків.

2.4 Планування ризиків

Заходи направлені на зменшення чи ліквідацію ризиків. За допомогою публікацій у провідних галузях [4-7] опишемо множину можливих заходів із зменшення або ліквідації ризиків: $EV = \{ev_1,...,ev_{19}\}$, де ev_1 – попереднє навчання членів проектного колективу; ev_2 – узгодження детального переліку вимог із замовником; ev_3 – включення узгодженого списку вимог замовника в

договір; ev_4 – точне слідування вимогам замовника з узгодженого списку вимог; ev_5 – попередні дослідження ринку; ev_6 – експертна оцінка проекту досвідченим стороннім консультантом; ev_7 – консультації досвідченого консультанта зі сторони; ev_8 – тренінг з вивчення необхідних інструментів розроблення; ev_9 – укладення договору страхувки; ev_{10} – використання «шаблонних» рішень з вдалих попередніх проектів при управлінні проектом; ev_{11} – підготовка документів, які показують важливість даного проекту для досягнення фінансових цілей компанії-розробника; ev_{12} – реорганізація роботи проектного колективу таким чином, щоб обов'язки та робота членів колективу перекривали один одного; ev_{13} – придбання (замовлення) частини компонентів розроблюваної автоматизованої системи; ev_{14} – заміна потенційно дефектних компонентів розроблюваної автоматизованої системи придбаними компонентами, які гарантують якість роботи; ev_{15} - придбання більш продуктивної бази даних; ev_{16} – використання генератора програмного коду; ev_{17} – переорганізація роботи колективу проекту в залежності від рівня складності задач та професійних рівнів розробників; ev_{18} – повторне використання підходящих компонентів ПЗ, які були розроблені для інших проектів; ev_{19} – аналіз доцільності створення даного ПЗ.

Введено слідуючі правила встановлення заходів із зменшення або усунення ризиків окремого програмного продукту (передбачено один, найкращий, захід для кожного з ризиків):

1. ∀(erip ∈ ERp) : якщо ризик eri p може бути зменшений або усунутий за допомогою заходу ev1 ∈ EV , то ever i p =ev1;
2. ∀(erip ∈ ERp): якщо ризик eri p може бути зменшений або усунутий за допомогою заходу ev2 ∈ EV , то ever i p =ev2;
3. ∀($er_{ip} \in ER_p$) : якщо ризик $er_{i\,p}$ може бути зменшений або усунутий за допомогою заходу $ev_{19} \in EV$, то $ever_{\,i\,p} = ev_{19}$.
4. Тоді у ході застосування зазначених вище правил визначення заходів із зменшення або усунення ризиків маємо множину заходів

для конкретного програмного проекту: $EVER_p = \{ever_{i\,p}\}$, де $i = 1..n$, n – кількість ризиків проекту.

2.5 Моніторинг ризиків

Оцінка ризиків. Важливо встановити, що всі величини, що пов'язані із ризиком, не є сталими в процесі створення проекту. Ймовірність ризиків та подій, що з ними пов’язані, а також можливі збитки можуть збільшитись та зменшитись в ході прийняття заходів із зменшення ризиків або їх ліквідації. Саме для цього потрібна оцінка ймовірності, збитків та величини ризиків після впровадження таких заходів.

Для ризиків з множини ER_p розробники повинні встановлювати його ймовірність (в діапазоні [0;1]) та його ймовірність впливу після застосування конкретного заходу із зменшення або усунення ризиків.

Множина ймовірностей ризиків після застосування заходів має вигляд: $EPRER_p = \{eprer_{i\ p}\}$, де $i = 1..n$, n – кількість ризиків проекту. Для ризиків з множини ER_p розробники повинні встановлювати розміри можливих збитків (в діапазоні [0;1]) від його ймовірності після застосування конкретного заходу, пов’язаного із зменшенням або усуненням ризиків. Множина збитків ризиків після застосування заходів має вигляд: $ELRER_p = \{elrer_{i\,p}\}$, де $i = 1..n$, n – кількість ризиків проекту.

Для кожного з ризиків у множині ER_p треба визначити його величину після впровадження конкретного заходу із зменшення або усунення ризиків. Множина величин ризиків після заходів має вигляд: $EVRER_p = \{evrer_{i\,p}\}$, де $i = 1..n$, n – кількість ризиків проекту, $evrer_{ip} = eprer_{ip} \cdot elrer_{ip}$. Можливе повторне встановлення рівня пріоритету ризиків після застосування заходів із зменшення або усунення ризиків.

Висновки до розділу 2

Успішність проекту залежить від ризиків, які спричинені недостатньою функціональностю автоматизованої системи(АС), недотримання термінів, зазначених у регламенті на виконання проекту, перевищення проектного бюджету. Запропонований метод управління ризиками при розробленні програмного забезпечення, як частини АСК, дає нам можливість ідентифікувати джерела ризиків та потенційні ризики для будь-якого програмного проекту, а також оцінити ризики, визначити їх пріоритет та заходи із зменшення або усунення ризиків. Окрім цього, даний метод дає змогу зробити оцінку ризиків після впровадження окремих заходів із зменшення або ліквідації ризиків, і це дає можливість підбирати найкращі заходи для максимального зменшення величини кожного ризику. Описаний метод на основі моделі SEI має математичне підгрунтя, через що зменшується складність та зростає ефективність управління ризиками.

Заради зменшення ризиків, що призводять до некоректної розробки при проектуванні АСК запропоноване інтегрування методів управління ризиками в процес управління проектом, що базується на моделях якості.

РОЗДІЛ 3 МЕТОДИ УПРАВЛІННЯ РИЗИКАМИ.

У сучасних сферах розробки ПЗ можна побачити тенденції спрямовані на вдосконалення методологій розробки ПЗ визначення цілей, меж, недоліків та переваг.

Найперспективними в данних напрямках є гнучкі методології розробки ПЗ

В яких можна визначити наступні стадії та процеси:

- аналіз, формування вимог, складання технічних характеристик та процесів проектування;

- процес реалізації;

- тестування, введення в експлуатацію та підтримка вже розробленого програмного забезпечення.

Розглядаючи загальну структуру гнучких методологій, ми можемо побачити актуальність оцінки ризику.

Це особливо важливо зараз із збільшенням кількості кібератак та пошуком вразливостей у програмному забезпеченні.

На жаль, важко взяти до уваги все, це обумовлено низкою суб'єктивних та об'єктивних причин. Частково це відбувається через відсутність структурованих знань в цій галузі.

Метою даного розділу є аналіз сучасних методів оцінки ризиків при розробці програмного забезпечення, яке дозволить мінімізувати кількість вразливостей програмного забезпечення, враховуючи багато соціальних та

економічних факторів в процесі оцінки ризиків у життєвому циклі програмного забезпечення.

Існує кілька досліджень, які описують якісні та кількісні методи оцінки ризику. Ці методи включають:

- «Інтерв'ю»;

- «Метод процентної ставки»;

- «Аналіз чутливості»;

- «Аналіз рішень»;

- «Метод сценарію»;

- «Метод дерева рішень»;

- «Моделювання»;

- «Оцінка тенденцій ризику»;

- «Оцінка потенційних ризиків та їх впливу».

Але через нехтування врахуванням факторів інформаційної безпеки представлені методи не можуть повністю оцінювати ризики розробки програмного забезпечення. Ми переглянемо можливі рішення та методи управління ризиками.

Розробка програмного забезпечення, беручи до уваги також фактор кібер-вразливості. Роблячи все створено класифікацію методів оцінки та управління ризиками.

3.1 Методи дослідження для оцінки ризику

Кількісні методи. Кількісна оцінка ризику (Рис. 3.1) визначає ймовірність ризику та його вплив на проект.

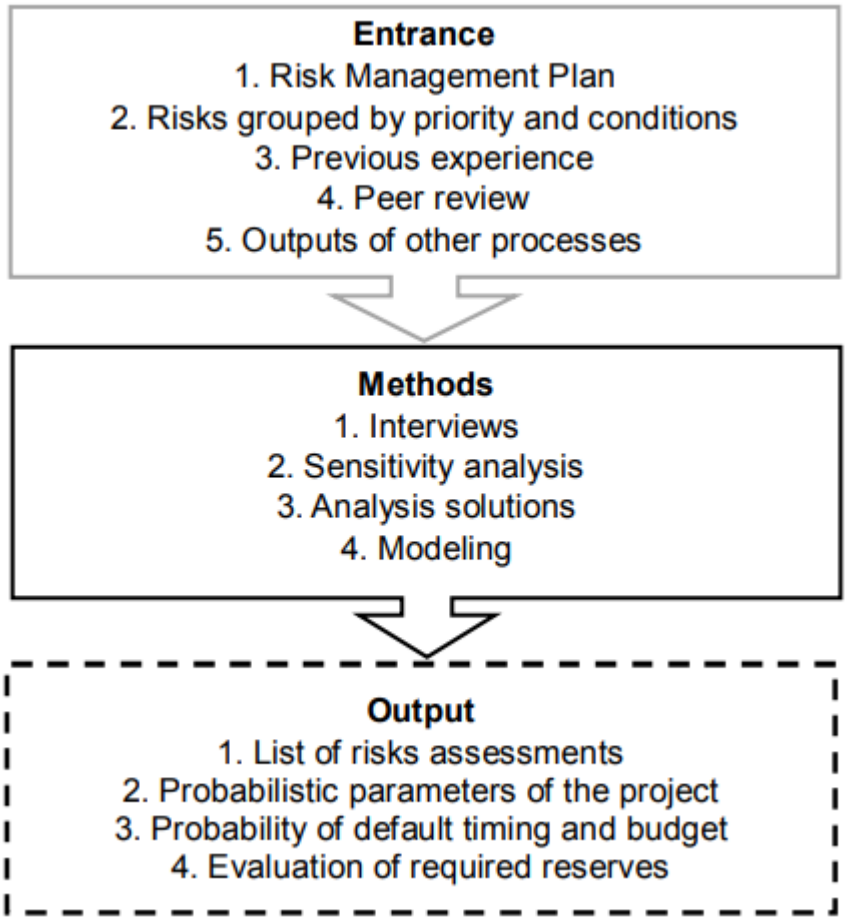


Рис. 3.1 Методи кількісної оцінки ризику

Це допомагає групі управління проектами у створенні правильних рішення та уникати невизначеностей. Кількісна оцінка ризику, як правило, супроводжується якісною оцінкою, а також вимагає ідентифікації ризику. Кількісну та якісну оцінку ризику можна використовувати окремо або разом, залежно від кількості часу, бюджету та їх необхідності.

Якісні методи – це процес аналізу ідентифікації ризиків (Рис. 3.2.) та пошуку необхідних ризиків для їх негайного вирішення. Такий метод оцінки ризику визначає важливість кожної небезпеки та вибирає спосіб реагування. Доступність допоміжної інформації робить це легшим для визначення пріоритетів для різних категорій ризиків.

Ця оцінка оцінює причини небезпеки та визначає їх вплив на проект за допомогою стандартних методів та засобів. Використання цих інструментів допомагає частково уникати невизначеності, яка часто зустрічається в проектах.

Протягом усього терміну реалізації проекту повинна здійснюватися постійна переоцінка ризиків.

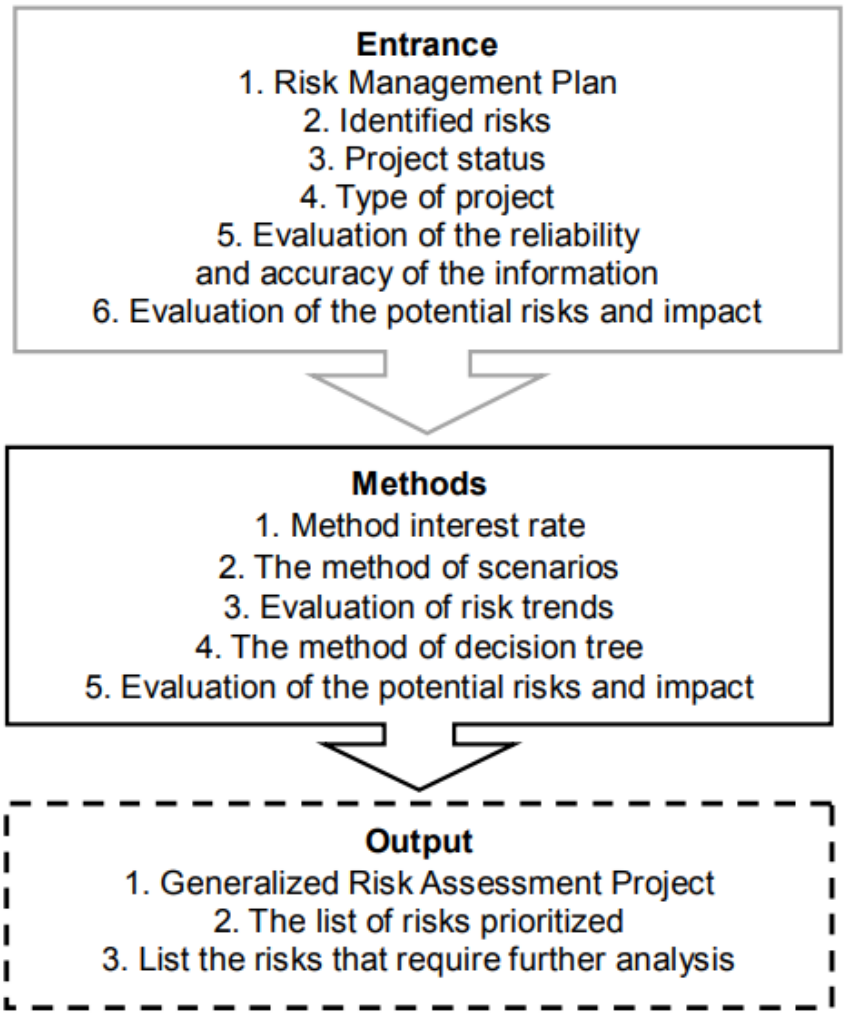


Рис. 3.2 Методи якісної оцінки ризику

Використовуючи метод дерева рішень, ми можемо вирішити завдання класифікації та зробити прогнозування у трьох напрямках можливих варіантів: оптимістичний, песимістичний і нормальний.

Дерево рішень представляє мережеві діаграми, які показують час, коли сталася подія, і ймовірність отримання необхідного результату. Кожна гілка

дерева представляє різні способи того, як все буде відбуватися. Більше варіацій у прогнозованих критеріях означає більше ризиків для проекту.

Метод сценарію дозволяє перейти від детального опису стратегічного та операційного ризику, однакового для всіх видів корпоративної діяльності, до обробки двох ймовірних сценаріїв: песимістичний (найгірший випадок) та оптимістичний (в кращому випадку). На завершальному етапі стратегічного планування такої оцінки ризику необхідно здійснити виконання планових завдань: напружена – оптимістичний сценарій, найбільш реальний і занижений.

Крім цього, під час розробки програмного забезпечення, ризик стратегічних ризиків підприємства в цілому та певна діяльність оперативності також враховуються такі ризики, як програмне забезпечення для управління, виробництво та маркетинг і це все береться до уваги.

Серед цих методів, представлених на (мал. 3.2)

Були вказані «метод дерева рішень» та «сценарний метод ».

Ці способи забезпечують кращий досвід розробки програмного забезпечення через їх високу точність та швидкий процес навчання, а також наскрізну оцінку можливих сценаріїв, що вони дозволяють прийняти єдине правильне рішення. Представляючи сценарії та аналізуючи їх ми можемо побачити раціональну стратегію впливу на ситуацію для прийняття правильного рішення.

3.2 Класифікація методів оцінки управління ризиками розробки програмного забезпечення

Зараз багато користувачів стикаються з недоробками у процесі використання програмного забезпечення. Для того, щоб зменшити кількість вразливостей, існує безліч методів оцінки ризику.

Серед методів, що використовуються в управлінні розробкою програмного забезпечення, є:

- Метод 'команда безпеки';

- Метод «стерильне приміщення»;

- Метод правильності конструкції;

- Модель вдосконалення процесу CMMI.

Метод 'Команда безпеки' полягає у виділенні структури організації або відділу, що називається службою безпеки, яка відповідає за розвиток та вдосконалення інформаційної безпеки, також виступає експертом з інформаційної безпеки для всієї організації в цілому і для кожного проекту окремо.

Відділ безпеки призначає співробітника з питань відповідності або групу працівників на роль інженера з безпеки для конкретної галузі.

Інженер безпеки допомагає команді розробників, аналізуючи всі їх дії та документи, створені під час процесу розробки, такі як перелік вимог безпеки та проектні документації та дає рекомендації на основі проведеного аналізу.

Таким чином Інженер з безпеки відповідає за методологію розробки та за безпеку продукту в цілому.

З приводу цього методу слід зазначити, що використанням такого підходу не гарантує безпеки розроблення програмного забезпечення. Однак використання такого підходу може зменшити загальну кількість вразливостей у розробленому програмному забезпеченні.

Основні соціальні та економічні фактори, зазначені у цьому способі, є:

- культурні проблеми та проблеми довкілля;

- не виявлення вразливостей в програмній безпеці;

- непередбачена технічна проблема.

Розробка програмного забезпечення методом 'стерильної кімнати' теоретично обґрунтована та орієнтована на розвиток процесу, перевірки та сертифікації правильності програмного забезпечення систем зі статистичним контролем якості.

Стерильна кімната' включає весь життєвий цикл розробки, включаючи управління проектами, визначення функцій та специфікацій архітектури, перевірку функціональності, а також статистичне тестування для сертифікації програми.

Основна ідея цього методу – запобігання помилок та дефектів програмного забезпечення, а не їх усунення.

Основними принципами методу є:

- поступовий розвиток на основі статистичних даних контролю якості;

- використання принципу структурованих дизайнерських концепцій;

- тестування на основі статистичних методів;

- ітеративний розвиток;

- перерозподіл часу між етапами життєвого циклу;

- розробка програмного забезпечення повинна базуватися на офіційних методах.

До переваг методу 'стерильної кімнати' відносять - широкі можливості для перевірки програмного забезпечення систем, завдяки використанню формальних специфікацій,

Метод передбачає детальний формальний опис усіх можливих сценаріїв виконання розроблюваємих програм, що значно зменшує ймовірність некоректної роботи програми та зменшує ймовірність помилок у специфікаціях та вимогах до програмного забезпечення.

Метод дозволяє виявити та усунути помилки тп вразливості на ранніх стадіях розробки, починаючи зі стадії формування специфікацій і, таким чином запобігає таким критичним помилкам на етапі проектування.

Серед недоліків можна зазначити, що метод орієнтований на правильний розвиток окремих компонентів системи, але у той же час не забезпечує достатньо ефективної інструментів для аналізу та перевірки системи в цілому.

Метод 'стерильного приміщення' не передбачає аналізу поведінки системи в динаміці, та вимагає складних допоміжніх засобів для автоматизованої перевірки різних систем та їх відповідність технічним вимогам.

Основними соціальними та економічними факторами є:

- неадекватний вибір стратегії розробки програмного забезпечення;

- нестабільність постачальників;

- не виявлення вразливості програмного забезпечення;

- неефективна взаємодія між зацікавленими сторонами.

Метод структурної коректності включає визначення формальних позначень для специфікацій системи та компонентів архітектури на основі їх узгодженості та правильністості. Це дозволяє, використовуючи офіційні методи, перевірити програмне забезпечення на наявність дефектів і негайно їх усунути протягом його життєвого циклу.

Цей метод включає визначення формальних позначень для специфікацій системи та компонентів архітектури на основі їх послідовності та правильності.

Для безпечних систем виділяють категорії системних станів та операції, які визначаються на основі їх впливу на загальну безпеку.

Кінцевою метою є створення архітектури, яка мінімізує кількість функцій, критичних для захисту та їх ізоляції. Це допомагає додатково зменшити вартість і обсяг роботи, пов'язаної з перевіркою правильності всіх елементів.

Основними соціальними та економічними факторами при цьому методі є:

- неадекватний вибір стратегії розробки програмного забезпечення;

- неефективне управління проектами;

- невідповідність організаційної структури.

Модель вдосконалення процесу CMMI, що використовується для оцінки загальної ефективності організації, відповідно до критеріїв, зазначених у моделі, та знаходження шляхів вдосконалення його. Модель CMMI включає основні напрямки процесів розвитку в організації, набори практики та методи розвитку, адаптація та оптимізація яких характерна для конкретної організації є досить ефективним інструментом для вдосконалення в цілому процесу розвитку. Основними соціально-економічні факторами, які враховуються при цьому методі, є:

- зміни в законодавстві;

- неадекватне управління фінансами;

- невідповідність організаційної структури.

Як видно з результатів вищезазначеного аналізу, основні недоліки цих методів оцінки та управління ризиками – це відсутність виявлення вразливості у програмному забезпеченні. І це може призвести до значних проблем під час роботи програмного забезпечення. Рішенням цього протиріччя є аналіз програмних уразливостей та проведення PEN-тестувань.

Висновки

Загалом можна зазначити, що правильне застосування методів оцінки та управління ризиками під час розробки програмного забезпечення може суттєво підвищити якість та безпеку виробу, розробленого при відносно помірних витратах. Таким чином, враховуючи ефективність різних методів та відмінності в їх застосуванні

Здається, що найефективніший спосіб їх використання – змішаний, для безпечного розвитку на різних етапах.

3.2 Метод управління ризиками в невеликих програмних проектах

Для малих проектів управління ризиками буде відрізнятись в порівнянні з більшими проектами. Невелекими можна вважати проекти, яким менше 1 року, кількість людей, що беруть участь у створенні якого в середньому менше 12, бюджет таких проектів менше $2 млн., менеджери у таких проектах можуть виконувати як керівницькі функції, так розробницькі, в таких проектах зазвичай не більше одного підрядника.

Розробка невеликих проектів відрізняється від великих ще в тому плані, що він розробляється трохи інакше зі сторони ризик-менеджменту. Вплив характеристик невеликого проекту на процес ризик-менеджменту зображено в таблиці 3.1.

Характеристики проекту	Вплив на процес ризик-менеджменту
Короткий графік	Невеликий час для пом'якшувальних дій Короткострокова перспектива залишає менше можливостей ризикувати
Лімітовані ресурси	Обмежена кількість розробників, які можуть керувати ризиками Незначні резерви бюджету, які можна використати для найняття додаткового персоналу
Маленька команда розробників	Ризик, пов'язаний з персоналом – критичний для кожного проекту
Менше зовнішніх зв'язків	Менше ризиків, через те що менше некерованих залежностей.

Таблиця. 3.1. Особливості малого проекту

В малих проектах визначають слідуючі джерела ризиків: незрозумілі або некомпетентні вимоги, необ'єктивні часові вимоги до проекту, некомпетентність виконавців, проблеми з бюджетом проекта.

Ранжування ризиків в малих проектах відбувається за допомогою матриці ідентифікації ризиків, данні в якій визначаються за допомогою матриці імовірності

ризиків та матриці наслідків матеріалізації ризиків. Після ідентифікації ризиків, складають план пом'якшення ризиків та запроваджують його в життя.

При дослідженні процесів управління ризиками в невеликих проектах обрано проект з розробки Internet-порталу. Це актуальний проект, так як велика кількість ПЗ, що розробляються в українських аутсорсингових компаніях – це проекти по розробці різних сайтів, інтернет-порталів та ін.

Дані проекти обмежені у часі, не потребують великої роботи та великої кількості розробників, бюджет таких проектів в середньому складає близько $1000. Зачасту, компанія паралельно реалізацує декілька подібних проектів, тому ризики, ідентифіковані та зменшені на одному проекті можуть спростити розробку інших.

3.3 Адаптація методу управління ризиками в загальний процес управління проектом на основі моделей якості

Для представлення структорованої інформації по проекту та автоматизованих систем управління необхідне використання моделей якості та представлення способів їх вимірювання. За означенням якості, як властивості, що повинна задовольнити потреби замовників та (або) користувачів, модель якості можна пояснити як сукупність властивостей та відношень між ними, які становлять інформаційну базу для опису вимог та їх оцінки при розробці. В наслідок цього, такі вимоги, як адекватність та повнота відображення вимог якосіт, представлення способів оцінки та здатність проекту до змін на різних етапах розробки ПЗ. Так як, вхідні дані –

це потреби замовника та властивості системи, пред'явлені на основі аналізу предметної області, то для опису їх на основі моделей якості потрібно встановити сукупність характеристик якості і визначити відповідні метрики цих характеристик.

Так як якість за визначенням – це «сукупність властивостей ПС, які виражають ступінь задоволення та відповідності потребам». Але розрізняють три категорії якості:

– якість у використанні – сукупність властивостей, що визначають міру досягнення користувачами поставлених цілей у визначеному середовищі та відповідному контексті експлуатації.

– зовнішня якість – сукупність властивостей, які виражають ступінь відповідності вимогам при її використанні у середовищі розробника з набором тестових даних.

– внутрішня якість – сукупність властивостей, які відображають ступінь задоволення внутрішніх вимог якості і може бути визначена шляхом проведення інспекцій коду, тестування та рев'ю.

Для представлення якості на практиці запропоновано чотири характеристики якості, а для зовнішньої та внутрішньої – шість. Щоб деталізувати характеристи зовнішньої та внутрішньої якості АСК визначені деякі набори підхарактеристик. Також компонентами моделей, які кількісно інтерпретують три категорії якості, є визначені набори метрик.

Для побудови моделі якості у використанні необхідно визначити, зафіксувати та формалізувати множину потреб замовника і користувачів R_c бізнес-системи.

Формалізацію потреб у бізнес-системі виконано з використанням теоретико-множинної нотації, тобто у вигляді множини, компонентами якої є потреби замовників і користувачів автоматизованої системи керування, а також відповідних обмежень на ці потреби

$$R_c = \{P_i, C_{iк}\},\ i = \overline{1,N},\quad K = \overline{1,M_i}, \tag{3.1.}$$

де P_i – потреби користувача;

$C_{iк}$ - обмеження на потреби;

N – кількість потреб замовника;

K – кількість обмежень на потреби.

Виходячи з бізнес-вимог та вимог предметної області, для кожної потреби P_i задається множина атрибутів, $\{A_{iK}\}$, $K = \overline{1,S_i}$, які відображають ступінь задоволення і-ої потреби. В результаті отримуємо сукупність

$$\{P_i, A_{iK}, C_{iK}\},\ i = \overline{1,N},\ K = \overline{1,S_i}. \tag{3.2.}$$

Сукупність $\{P_i, A_{iK}, C_{iK}\},\ i = \overline{1,N},\ K = \overline{1,S_i}$ представляє вимоги до автоматизованої системи керування користувача бізнес-системи. З кожним атрибутом пов’язаний деякий ризик з моделі SEI. Тоді, для представлення вимоги до автоматизованої системи і сукупності ризиків, множина (2.2) трансформується у множину

$$\{P_i, A_{iK}, C_{iK}, Risk_{iL}\},\ i = \overline{1,N},\ K = \overline{1,S_i}, K = \overline{1,R_i} \tag{3.3.}$$

$Risk_{iL}$ – множина ризиків, пов’язана з атрибутом якості, $i = \overline{1,N},\ L = \overline{1,R_i}$

$Risk_{iL}$ є елементом множини, що формується об'єднанням множин ризиків моделі SEI, тобто

$$Risk_{iL} \in TRS \cup CRS \cup PRS \cup MRS \qquad (3.4.)$$

де TRS – множина технічних ризиків, пов'язаних з реалізацією автоматизованих систем керування;

CRS – множина ризиків щодо бюджетних витрат при розробці автоматизованих систем керування;

PRS – множина ризиків при плануванні проектів автоматизованих систем керування;

MRS – множина ризиків при управління проектами із створення автоматизованих систем керування

Щоб записати дані вимоги в стандартизованому вигляді представимо (3.3.5) у вигляді елементів структури моделі якості під час використання. Як наслідок можна отримати модель вимог якості R_{use} користувача бізнес-системи, сформульовані в стандартизованих термінах.

$$R_{use} = \left\{H_i^u, A_{ik}^u, C_{ik}^u, M_{ik}^u\right\},\ i \in N_u^k, K = \overline{1, S_i}. \qquad (3.5.)$$

Процедуру відображення на структуру реалізовано за два кроки:

– представлення (2.1) у вигляді шаблону $\left\{s_1, s_2, s_3\right\}$ де s_1 – поле «назва компоненту до якого сформульована вимога» s_2 – поле «атрибут або характеристика якості» виділені з тексту (2.1) та s_3 – поле «метрика вимірювання»;

– класифікація атрибутів s_2 за стандартизованими наборами характеристик і метрик з використанням бази знань, сформованої експертним шляхом. У базі знань містяться асоціації між атрибутом шаблона та стандартною характеристикою і відповідним їй атрибутом якості, визначеним з аналізу предметної області та специфіки класу до якого належить ПС. Класифікація проводиться шляхом пошуку в базі знань такої пари $\{s_{1n}, s_{2n}, s_{3n}\}$ та $\{H_i^u, A_{ij}^u, M_{ij}^u\}$ для якої виконується нерівність $\{Supp_l\} >= \{\overline{Supp_l}\}$, де $\{Supp_l\}, l = \overline{1, L}$, – підтримка асоціації, $\{\overline{Supp_l}\}$ – визначений граничний рівень асоціації.

Представлення вимог якості у використанні у вигляді моделі (3.3.6) дає змогу для формалізації та стандартизації уніфікованих термінів. Що, в свою чергу, забезпечує адекватне і повне відображення потреб користувачів бізнес-системи, уникнення нечітких формулювань та «підміни понять», і суттєвоспрощує розробку засобів автоматизації, які орієнтовані на підтримку технологічних процесів на ранніх стадіях життєвого циклу. Взявши за основу модель (3.3.6) зручно представляти користувацькі вимоги до АСК та в наступному управляти вимогами якості ефективно, забезпечувати їх поліморфність та здійснювати моніторинг структурованих раніше ризиків.

Заради забезпечення зовнішньої якості АСК на етапах життєвого циклу пропонується використання моделі зовнішньої якості, що складається з характеристик, підхарактеристик, атрибутів та метрик. Атрибути, якими вимірюється рівень задоволення потреби по певній підхарактеристиці, вибираються, виходячи з аналізу предметного середовища використання АСК.

Вимоги зовнішньої якості формулюються в означеннях моделі зовнішньої якості, через відображення вимог якості у використанні на компоненти моделі (3.3.6) та додаванням тих атрибутів, що не були враховані на попередній стадії. При цьому вимоги зовнішньої якості повинні бути встановлені у специфікації на основі зовнішніх атрибутів і в подальшому

можуть бути використані як критерії оцінювання готового програмного продукту.

Для формалізації процедури побудови і представлення моделі зовнішньої якості запишемо її у вигляді множини:

$$\left\{H_i^x, \Pi_{ij}^x, M_{ij}^x\right\},\ i=\overline{1,6},\ j=\overline{1,K_i}, \tag{3.6.}$$

де H_i^x – характеристики зовнішньої якості;

Π_{ij}^x – підхарактеристики зовнішньої якості;

M_{ij}^x – відповідні метрики;

K_i – кількість підхарактеристик і-ої характеристики.

Так як, модель зовнішньої якості повинна відображати потреби користувача на рівні проектування програмного комплексу в цілому та його підсистем, то необхідно відобразити (2.16) на елементи (2.17) і додати необхідні елементи множини, які не використовувались на попередньому етапі. В результаті отримаємо специфікації вимог зовнішньої якості R_{ext} в стандартизованих термінах.

$$R_{ext}=\left\{H_i^x, \Pi_{iK}^x, A_{iK}^x, C_{iK}^x, M_{iK}^x\right\},\ i\in N_x,\ K=\overline{1,F_i^x}. \tag{3.7.}$$

Формалізоване представлення вимог зовнішньої якості дає змогу однозначно та повно відобразити вимоги якості у використанні, які реалізують потреби користувача та замовника автоматизованої системи керування. Крім того, формалізація вимог на етапі їх розробки забезпечує точність та простоту подальшої розробки проекту, зокрема це стосується

вибору та побудови майбутньої архітектури. Тому впровадження запропонованої формалізації моделі зовнішньої якості для автоматизованих систем керування є досить актуальним, оскільки дозволяє спростити та автоматизувати цей процес, а також науково обґрунтувати доцільність і практичність застосування стандартів.

Вимоги внутрішньої якості використовують для визначення властивостей проміжних станів продукту. При цьому можна використати статичні та динамічні моделі, технічну документацію та код. Вимоги внутрішньої якості можуть бути використані для визначення стратегії подальшої розробки та можуть виступати в якості критеріїв оцінювання і верифікації процесу розробки. Вимоги внутрішньої якості, як і всіх інших категорій запропоновано оцінювати кількісно з використанням відповідних атрибутів і метрик.

Переходячи до формалізації моделі внутрішньої якості, зазначимо ієрархічну структуру залежностей між характеристиками внутрішньої якості, які є такими ж як у моделі зовнішньої якості. Фактично модель внутрішньої якості, відображає та доповнює модель зовнішньої якості з погляду структури та функціональності модулів системи.

Оскільки, структура стандартизованої внутрішньої моделі якості має вигляд аналогічний до структури зовнішньої моделі якості, то формально її можна представити у вигляді (3.3.8). Атрибути внутрішньої моделі якості запропоновано формувати на основі відображення атрибутів зовнішньої моделі якості на функціональні модулі продукту та елементів, які не увійшли до жодної з розглянутих вище моделей. Формалізоване представлення вимог внутрішньої якості R_{in} при відображенні на модель (2.18) матиме наступний вигляд:

$$R_{in} = \left\{ H_i^x, \Pi_{iK}^x, A_{iK}^y, C_{iK}^y, M_{iK}^y \right\}, i \in N_x, K = \overline{1, F_i^y} \qquad (3.8.)$$

Формування специфікацій вимог R_{in} до модулів автоматизованої системи керування виконується відображення вимог (2.18) на елементи структури моделі внутрішньої якості, в результаті отримується модель (2.19). Це відображення виконується для кожного модуля з врахуванням його функціональності або шляхом відображення на архітектурні шари Метрики для вимірювання атрибутів на основі асоціативних правил або застосування методу Text Mining можна визначити з відповідного розділу стандарту ISO 9126.

Якість виконання етапу розробки вимог залежить від процедур формалізації сформульованих замовником потреб. Такі процедури є досить трудомісткими та слабоформалізованими, що у свою чергу знижує якість виконання проекту. Проте запропоновано технологію формулювання вимог до автоматизованих систем керування у термінах стандартизованих моделей якості з використанням процедур формалізації відповідних процесів.

Застосування цих моделей дозволяє мінімізувати неоднозначні трактування вимог до автоматизованих систем керування, а також стандартизувати та адекватно представити їх. Тому за основу моделі ЖЦ, яка ґрунтується на «парадигмі якості» нами використано саме цю технологію. При цьому додатковими вхідними даними для етапу розробки вимог, окрім потреб замовника, запропоновано використати критерії якості вимог. На рис. 3.1. наведено графічне зображення процесу розробки вимог з інтеграцією критеріїв якості та можливих ризиків.

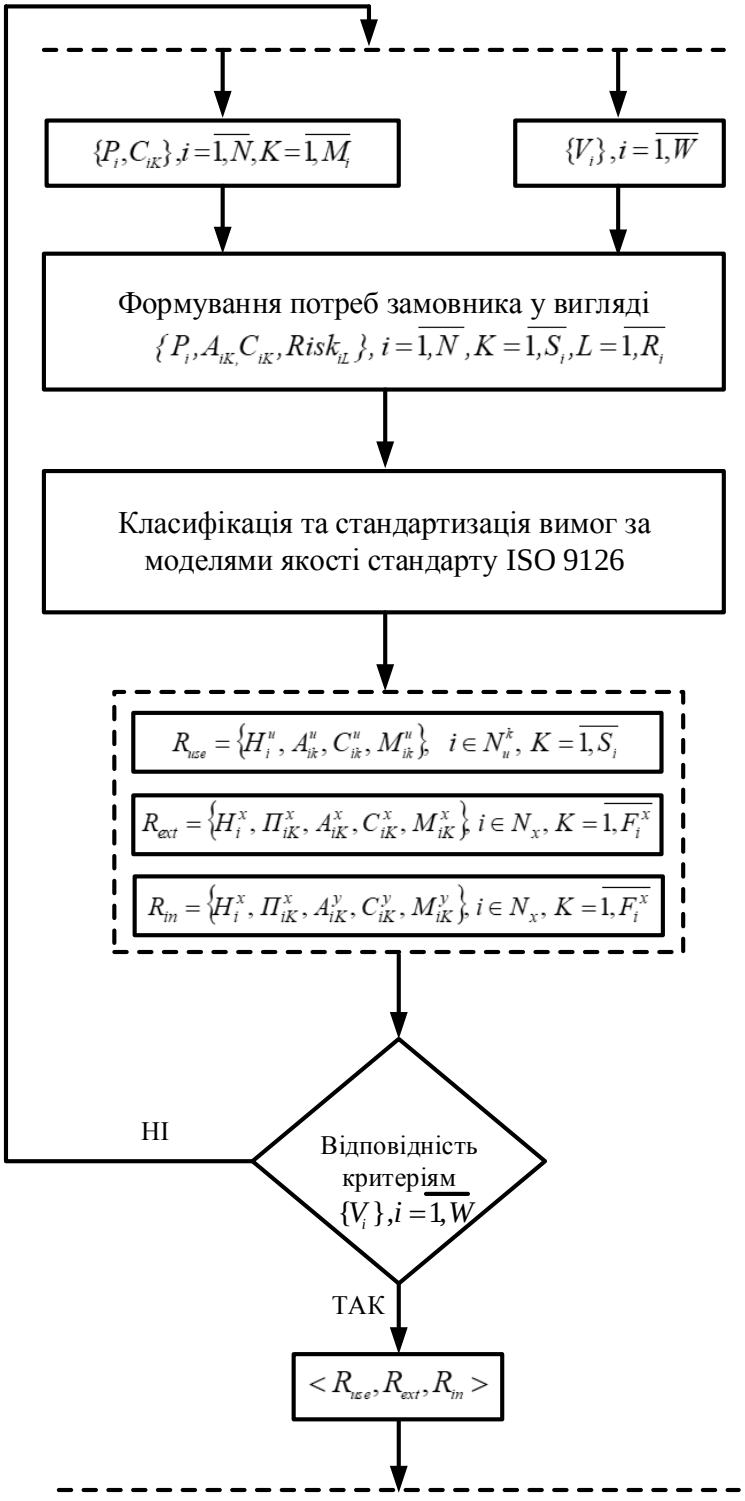


Рис. 3.1. Алгоритм розробки вимог до автоматизованих систем керування з врахуванням потенційних ризиків для кожного атрибута якості

Висновки до розділу 3

В данному розділі було проведено аналіз сучасних методів оцінки ризиків при розробці програмного забезпечення, які дозволять мінімізувати кількість вразливостей програмного забезпечення, враховуючи багато соціальних та економічних факторів в процесі оцінки ризиків у життєвому циклі програмного забезпечення. Також були розглянуті методи дослідження для оцінки ризику та види оцінки управління ризиками при розробці програмного забезпечення, і як наслідок адаптовано метод управління ризиками в загальний процес управління проектом на основі моделей якості.

Формалізацію потреб у бізнес-системі виконано з використанням теоретико-множинної нотації, тобто у вигляді множини, компонентами якої є потреби замовників і користувачів автоматизованої системи керування, а також відповідних обмежень на ці потреби.

ВИСНОВКИ

В першому розділі було розглянуто сучасний стан та проблеми розробки програмних проектів. Були описані процеси, етапи та плани проектування. Дана теоретична та практична інформація по ідентифікації, аналізу та моніторингу ризиків, що дозволила нам надалі математично формалізувати задачі управління ризиками.

У другому розділі запропонований метод управління ризиками при розробленні програмного забезпечення, як частини АСК, який дає нам можливість ідентифікувати джерела ризиків та потенційні ризики для будь-якого програмного проекту, а також оцінити ризики, визначити їх пріоритет та заходи із зменшення або усунення ризиків. Окрім цього, даний метод дає змогу зробити оцінку ризиків після впровадження окремих заходів із зменшення або ліквідації ризиків, і це дає можливість підбирати найкращі заходи для максимального зменшення величини кожного ризику. Описаний метод на основі моделі SEI має математичне підгрунтя, через що зменшується складність та зростає ефективність управління ризиками.

Заради зменшення ризиків, що призводять до некоректної розробки при проектуванні АСК запропоноване інтегрування методів управління ризиками в процес управління проектом, що базується на моделях якості.

Також були розглянуті методи дослідження для оцінки ризику та види оцінки управління ризиками при розробці програмного забезпечення, і як наслідок адаптовано метод управління ризиками в загальний процес управління проектом на основі моделей якості.

У третьому розділі було здійснено формалізацію потреб у бізнес-системі, яку виконано з використанням теоретико-множинної нотації, тобто у вигляді множини, компонентами якої є потреби замовників і користувачів автоматизованої системи керування, а також відповідних обмежень на ці потреби.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ould M. *Managing Software Quality and Business Risk* – Chichester: John Wiley and Sons, 1999.

2. Hall E. *Managing Risk: Methods for Software Systems Development.* – Reading, MA: Addison-Wesley Longman, 1998.

3. Лавріщева к.м. Програмна інженерія.–к.– 2008.–319

4.М.В.Островий, М.В.Красовський, В.І.Грибинчук. Метод управління ризиками при розробленні програмного забезпечення. Всеукраїнська НПК «Інтелектуальні технології в системному програмуванні» Хмельницький, 2015р

5. Эрик Дж. Брауде. Технология разработки программного обеспечения. – СПб: Питер, 2004. – 655

6. D.O. Lysytsia, S.S. Bulba. Classification of methods assessment and management risk development software . Наука і техніка Повітряних Сил Збройних Сил України, 2016, № 1(22)

7. Fenton N. Decision Support Software for Probabilistic Risk Assessment Using Bayesian Networks / N. Fenton, M. Neil // IEEE Software. – 2014. – № 31 (2). – P. 21-26.

8. Fenton N. Risk Assessment and Decision Analysis with Bayesian Networks / N. Fenton, M. Neil // CRC Press. – 2012. – P. 33-38.

9. Soumya Krishnan M. Software Development Risk Aspects and Success Frequency on Spiral and Agile Model / M. Soumya Krishnan // Int. Journal of Innovative Research in Computer and Comm. Eng. – 2015. - Vol. 3. – P. 122-129.

10. Hijazi H. A Review of Risk Management in Different Software Development Methodologies / H. Hijazi // Int. Journal of Comp. Appl. – 2012. – Vol. 45, № 7. – P. 1311-131

11. DoD. USA. 2006. Risk management guide for risk acquisition.

12. Ray C. Williams, George J. Pandelios, Sandra G. Behrens. 1999. Software Risk Evaluation (SRE) Method Description

13. Donna L. Johnson. Nashville, Tennessee. 2006. Risk Management and the Small Software