МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ:

магістерський курс

Навчальний посібник

Рекомендовано
Луцьким національним технічним університетом

Луцьк-2024
ЛНТУ

УДК 004.4(075.8):378
І-62

Автори:
д.т.н., професор Андрущак І.Є. (розділ 4), к.т.н., доцент Ліщина Н.М. (розділ 3), к.т.н., доцент Суринович О.М. (розділ 1, 5), к.т.н., доцент Ящук А.А. (розділ 2).

Рецензенти:

Гаврилко Є. В. доктор технічних наук, професор, професор кафедри інженерії програмного забезпечення в енергетиці ІАТЕ Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Сверстюк А. С. доктор технічних наук, професор, професор кафедри медичної інформатики Тернопільського національного медичного університету імені І.Я. Горбачевського.

Конет І. М. доктор фізико-математичних наук, професор, професор кафедри теорії функцій та методики навчання математики ВНУ імені Лесі Українки

Рекомендовано Вченою радою ЛНТУ
(протокол № __ від _____ 2024 р.)

І-62 Інженерія програмного забезпечення : магістерський курс / за загальною редакцією І.Є. Андрущака. Луцьк: ЛНТУ, 2024. ____с.

У посібнику висвітлено теоретичні основи нормативних дисциплін освітньої програми «Інженерія програмного забезпечення». Наведено методичні вказівки до виконання ключових практичних робіт. Контрольні запитання та рекомендована література до кожного розділу сприятимуть самостійному засвоєнню матеріалу.

Для магістрів спеціальності 121 «Інженерія програмного забезпечення» та викладачів закладів вищої освіти.

УДК 004.4(075.8):378

© Колектив авторів, 2024

Навчальне видання

ЗМІСТ

ВСТУП

Посібник «Інженерія програмного забезпечення: магістерський курс» покликаний забезпечити магістрів інженерії програмного забезпечення теоретичними та практичними знаннями для глибшого розуміння та ефективного застосування сучасних технологій і методик в області програмування.

Матеріал курсу включає комплексний огляд ключових аспектів програмної інженерії, від теоретичних основ до конкретних практичних навичок. Від управління ризиками в IT-проектах до розробки мобільних додатків і захисту інформації в комп'ютерних системах, кожен розділ спрямований на те, щоб надати студентам і викладачам засоби для досягнення професійного успіху та академічного зростання.

Посібник включає методичні вказівки, контрольні запитання та рекомендовану літературу, що дозволяють читачам не лише вивчати, але й активно застосовувати отримані знання у реальних умовах, розвиваючи критичне мислення та навички вирішення проблем.

Таким чином, цей навчальний посібник стає незамінним ресурсом для кожного, хто бажає займатися програмною інженерією на професійному рівні в академічному або промисловому секторі.

РОЗДІЛ 1
ВІДКРИТІ НАУКОВІ ОСНОВИ ТА ПРАКТИКА УПРАВЛІННЯ ІНФОРМАЦІЄЮ

1.1 Визначення відкритої науки

Відповідно до Рекомендації ЮНЕСКО щодо наукової діяльності та науковців (дослідників) 2017 р. під «наукою» розуміється діяльність, у ході якої людство, діючи індивідуально, малими групами чи великими колективами, намагається в організованому порядку, на основі співпраці та конкуренції виявити та усвідомити ланцюжки причинно наслідкових зв'язків, відносин або взаємодій на основі об'єктивного вивчення явищ, що спостерігаються, і перевірки його результатів шляхом обміну висновками та даними та їх колегіального аналізу, пов'язує один з одним і зводить воєдино підсистеми знань шляхом систематичного осмислення і концептуалізації і таким чином отримує можливість у своїх інтересах розуміння процесів і явищ, що відбуваються в природі та суспільстві.

Спираючись на основні принципи академічної свободи, доброчесності дослідницької роботи та високого наукового рівня, відкрита наука встановлює нову парадигму, в якій у наукову діяльність включаються методи підвищення рівня відтворюваності, прозорості, обміну інформацією та співробітництва на основі розширення відкритого доступу до наукових матеріалів, інструментарію та процесів.

Термін «відкрита наука» відноситься до публікації наукових знань на різних мовах, їх загальнодоступності та їх придатності для спільного повторного використання в інтересах науки і суспільства (рис. 1.1).

Рисунок 1.1 – Відкрита наука [1]

Відкрита наука охоплює всі наукові дисципліни та аспекти наукової практики, включаючи галузі фундаментальної науки, прикладної науки, природознавства, соціальних наук та гуманітарних наук, і базується на таких важливих принципах: відкрите наукове знання, відкрита наукова інфраструктура, Наукова комунікація, відкрита участь соціальних суб'єктів та відкритий діалог. з іншими інформаційними системами.

Відкрита наукова інформація означає, що наукові публікації, дослідницькі дані, метадані, освітні ресурси, програмне забезпечення, вихідний код та обладнання доступні у відкритому доступі або захищені відповідною відкритою ліцензією. Цей доступ має бути швидким і, в можливих випадках, безкоштовним для всіх, незалежно від їхньої локації, національності, раси, віку, статі, доходу, соціально-економічного статусу, стадії професійної кар'єри, дисципліни, мови, релігії, інвалідності, етнічного чи імміграційного статусу чи інших факторів [2].

Цей підхід дозволяє кожному мати можливість отримати доступ до знань і ресурсів, необхідних для наукових досліджень, навчання та розвитку, сприяючи таким чином глобальному науковому прогресу і інклюзії.

Наукові знання також сприяють можливості відкриття доступу до методологій на наукові дослідження та процеси оцінки. Таким чином, користувачі безкоштовно отримують доступ до **матеріалів**, що розглянуті далі.

Наукові публікації. Включають рецензовані наукові журнали та книги, наукові звіти та матеріали конференцій, серед іншого, повинні бути розміщені у відкритому онлайн-сховищі з доступом до них відразу після публікації. Такі сховища підтримуються та підтримуються академічними установами, науковими асоціаціями, державними установами або авторитетними некомерційними організаціями та забезпечують відкритий доступ до матеріалів для загального блага, їх необмежений розподіл, Сумісність та довгострокове зберігання та архівування в цифровому форматі. Публікації, опубліковані під відкритою ліцензією або відкриті для громадськості (наприклад, публікації, опубліковані за відкритою ліцензією). Оригінальні результати досліджень, дані досліджень, програмне забезпечення, вихідний код, вихідний матеріал, дякуємо Вам за належну прив'язку до публікації). Доступ до публікацій за платною підпискою не відповідає цій рекомендації, якщо миттєвий доступ до наукових публікацій може бути наданий лише за окрему плату. Передача авторських прав третій стороні або її ліцензії не

повинна обмежувати право громадськості на негайний відкритий доступ до наукових публікацій.

Відкриті дані досліджень. До них відносяться зокрема необроблені та оброблені цифрові та аналогові дані, включаючи супутні метадані, числові дані, текстові записи, зображення, аудіозаписи, прототипи та інші дані. Відкриті дані дослідження легко доступні та зберігаються у зручному для користувача та машиночитаному форматі, який відповідає принципам належного управління та видалення даних, зокрема принципам справедливості (можливість пошуку, доступність, сумісність та повторне використання) та підлягають регулярному огляду та технічному обслуговуванню.

Дидактичні, освітні та дослідницькі матеріали на будь-яких носіях (цифрових чи інших). Вони публікуються під відкритою ліцензією.

Програмне забезпечення з відкритим кодом та вихідний код. Вони повинні бути негайно випущені за відкритою ліцензією. Вихідний код повинен бути включений у пакет програмного забезпечення та розміщений у загальнодоступних сховищах, а обрана ліцензія повинна передбачати можливість внесення змін, створення похідних продуктів та розповсюдження матеріалів на таких самих або подібних умовах відкритого доступу. У контексті відкритої науки у випадках, коли відкритий вихідний код є одним з компонентів науково-дослідного процесу, дозвіл на його повторне використання та відтворення, як правило, вимагає, щоб він супроводжувався відкритими даними та відкритими технічними параметрами середовища, в якому він буде складений та запущений.

Відкрите обладнання. Під відкритим обладнанням зазвичай розуміється специфікація проектування фізичного об’єкта, що має ліцензію на можливість перевіряти, модифікувати, створювати та розповсюджувати цей об’єкт, щоб якомога більше людей мали можливість проектувати та об’єднувати апаратні засоби, а також ділитися інформацією про дизайн та експлуатацію. У ситуаціях, коли і програмне, і апаратне забезпечення є відкритими, має застосовуватися контрольований громадськістю механізм внесення змін, їх атрибуції та управління ними з метою повторного використання, підвищення стійкості та усунення непотрібного дублювання зусиль. Програмний код, опис інструментів, зразки обладнання та саме обладнання можуть розповсюджуватися та адаптуватися безкоштовно за умови дотримання національного законодавства у сфері його безпечного використання (рис. 1.2).

Доступ до наукової інформації повинен бути максимально відкритим.

Обмеження доступу повинні бути пропорційно обґрунтовані. До них належать права людини, національна безпека, конфіденційність, право на конфіденційність, повага до людей учасників дослідження, дотримання юридичних процедур, охорона громадського порядку, права інтелектуальної власності, особиста інформація, священна та конфіденційна інформація корінних народів.

Деякі дані та програмні коди, які не знаходяться у відкритому доступі та не підлягають повторному використанню, можуть надаватися окремим користувачам відповідно до критеріїв доступу, встановлених відповідними органами управління місцевого, національного або регіонального рівня. У тих випадках, коли дані не можуть бути розміщені у відкритому доступі, важливим завданням є розробка інструментів та протоколів псевдонімізації та анонімізації даних, а також систем опосередкованого доступу, що забезпечують належну можливість обміну якомога більшим обсягом даних. Крім того, потреба в обґрунтованих обмеженнях може змінюватися з часом, тому ті чи інші дані можуть стати доступні на пізнішому етапі, або навпаки, доступ до них може бути обмежений.

Рисунок 1.2 – Відкриті наукові знання [1]

Відкрита наукова інфраструктура відноситься до віртуальної або фізичної громадської дослідницької інфраструктури (набір основних наукових інструментів та інструментів, заснованих на знаннях ресурсів, таких як колекції, журнали, публікації, сховища, архіви, платформи відкритого доступу до наукових даних, сучасні дослідження, інформаційні системи і т.д.).

Відкриті системи використовуються для оцінки та аналізу різних галузей науки, відкрита інфраструктура – для обчислень та обробки даних, включаючи відкриту інфраструктуру, що забезпечує агрегований та міждисциплінарний аналіз даних (рис. 1.3).

Рисунок 1.3 – Інфраструктура відкритої науки [1]

Відкриті наукові платформи, а також відкриті лабораторії використовуються в публікаціях, дослідницьких даних та вихідному коді, платформах масової розробки програмного забезпечення та віртуальних дослідницьких середовищах, аналізі та інтеграції, науковій літературі, тематичних наукових пріоритетах. Хоча різні сховища даних адаптуються до конкретних характеристик об’єктів, які вони розміщують (публікації, дані, код), місцевих умов, потреб користувачів та потреб дослідницької спільноти, необхідно розробити стандарти взаємодії та найкращі практики для забезпечення їх належної перевірки, прийому і повторного використання обладнання. Відкриті стенди для тестування інноваційних продуктів, такі як бізнес-інкубатори, громадські лабораторії, управління з використанням

відкритих ліцензій, наукові кіоски, музеї, парки та дослідницькі майстерні, є важливими компонентами відкритої наукової інфраструктури. Вони забезпечують загальний доступ до фізичних об'єктів, ресурсів та послуг.

Ці об'єкти інфраструктури відкритої науки часто створюються спільними зусиллями громади і мають за мету забезпечити довгострокову стійкість. Важливим аспектом є їхня некомерційна спрямованість і здатність забезпечити максимально широкий, постійний та необмежений доступ до ресурсів для всіх зацікавлених

Відкрита участь соціальних суб'єктів не тільки означає зміцнення співпраці між науковцями та соціальними суб'єктами за межами наукової спільноти, заснованої на використанні відкритих методів та інструментів дослідження, які є частиною дослідницького циклу, але також забезпечує більш всеосяжний декомунізаційний науковий процес та сприяє залученню більш широкого кола зацікавлених сторін, громадськості, сприяє новим формам співпраці та спільної діяльності, таким як співфінансування, участь громадськості та наукове волонтерство (рис. 1.4).

Рисунок 1.4 – Відкрита участь соціальних суб'єктів [1]

Прагнучи до створення колективного інтелекту, здатного до вирішення проблем, у тому числі з використанням міждисциплінарних методів дослідження, відкрита наука створює платформу для участі окремих громадян та їх колективів у процесі створення знань на користь розширення діалогу між науковцями, політиками, фахівцями практиками, підприємцями та

представниками громадськості, ставлячи всім його учасникам право голосу у питаннях проведення досліджень, що відповідають їх турботам, потребам та наполегливості.

Крім того, громадянська наука та наука, заснована на участі громадськості, розвивалися як модель наукових досліджень, що проводяться непрофесійними вченими, але відповідно до науково обґрунтованих методологій і нерідко у співпраці з офіційними науковими програмами або професійними дослідниками з використанням інтернет-платформ, соціальних мереж, а також заснованого на відкритому вихідному коді апаратного та програмного забезпечення (насамперед недорогого вимірювального обладнання та мобільних додатків) як важливі канали взаємодії. Для ефективного повторного використання продуктів цивільної та заснованої на участі громадськості науки іншими особами, включаючи вчених, ці продукти необхідно курувати, стандартизувати та зберігати так, щоб вони приносили максимальну користь усім сторонам.

Відкритий діалог з іншими інформаційними системами означає діалог між різними суб'єктами знань, декомунізованими існуванням різних інформаційних систем та гносеологічних моделей, та різноманітністю творців знань, таких як Всесвітня Декларація ЮНЕСКО про культурне різноманіття 2001 року (рис. 1.5).

Рисунок 1.5 – Відкритий діалог з іншими системами знань [1]

Цей діалог спрямований на популяризацію знань вчених (які традиційно маргіналізовані), зміцнення взаємозв'язків та забезпечення більшої взаємодоповнюваності між різними епістемологічними моделями, дотримання міжнародних норм і стандартів у сфері прав людини, повагу до суверенних прав на знання та їх використання, а також визнання прав носіїв знань на справедливу частку благ, які може принести їх застосування.

Державний сектор має відігравати провідну роль реалізації принципів відкритої науки. Разом з тим, цими принципами повинні також керуватися наукові дослідження, що фінансуються приватним сектором. Крім того, у науковій та інноваційній сфері задіяно багато учасників та зацікавлених сторін, і кожен із цих суб'єктів може відіграти певну роль у впровадженні відкритої науки. Незалежно від національності, етнічної приналежності, статі, мови, віку, дисципліни, соціально-економічного стану, джерел фінансування та етапу кар'єри, або інших причин, до суб'єктів відкритої науки серед інших входять дослідники, науковці, керівники науково-дослідних інститутів, педагоги, працівники вищої школи, члени професійних товариств, студенти та організації молодих дослідників, фахівці в галузі інформації, бібліотекарі, користувачі та широка громадськість, у тому числі об'єднана у спільноти, носії знань корінних народів, організації громадянського суспільства, фахівці з обчислювальної техніки, розробники програмного забезпечення, програмісти, творчі працівники, новатори, інженери, волонтери дослідники, правознавці, законодавці, судді, державні службовці, видавці, редактори, члени професійних товариств, технічні працівники, особи, які фінансують дослідження, та меценати, розробники політики, наукові товариства, фахівці практики, що працюють у професійних сферах, та представники приватного сектору, пов'язаного з наукою, технологіями та інноваціями.

1.2 Основні цінності і керівні принципи відкритої науки

Основні цінності відкритої науки випливають із її правозахисних, етичних, епістемологічних, економічних, правових, політичних, соціальних, інтегративних та технологічних наслідків для суспільства, а також із поширення принципів відкритості на всі етапи наукових досліджень (рис. 1.6) [3]. Коротко їх опишемо.

Рисунок 1.6 – Цінності та принципи відкритої науки [4]

Якість та сумлінність – відкрита наука має поважати академічні свободи та права людини, а також забезпечувати високу якість досліджень. Це досягається шляхом об'єднання різноманітних джерел знань і забезпечення широкого доступу до методів і результатів досліджень. Важливими аспектами є прозорі процедури, що дозволяють ретельний аналіз і перевірку дослідницьких висновків.

Відкритість наукового процесу означає, що результати досліджень і методи, використані для їх отримання, доступні для перегляду, критики та перевірки спільнотою. Це сприяє збільшенню довіри до наукових даних і результатів, покращує якість наукових досліджень і сприяє інноваціям. Відкрита наука є глобальним суспільним благом і має приносити користь всьому людству. Для цього наукові знання повинні бути доступними у відкритому доступі і стати частиною загального надбання. Відкритість наукової практики передбачає інклюзивність, стійкість і справедливість,

забезпечуючи рівні можливості для отримання наукової освіти і розвитку наукового потенціалу кожній особі.

Забезпечення відкритого доступу до наукових знань сприяє їхньому широкому використанню і впровадженню у різних сферах суспільного життя. Це сприяє зростанню інновацій, розвитку економіки, покращенню якості життя людей та вирішенню складних глобальних проблем.

Інклюзивна наукова практика означає, що всі люди, незалежно від їхнього походження, соціального статусу чи географічного положення, мають рівний доступ до можливостей для розвитку і використання наукових знань. Це сприяє створенню умов для розкриття потенціалу кожної людини і розширенню обсягів наукових досягнень.

Зрештою, відкрита наука має сприяти створенню більш справедливого і стійкого суспільства, де науковий прогрес є доступним і корисним для всіх його членів.

Рівноправність та справедливість – відкрита наука повинна відігравати важливу роль у забезпеченні рівноправності дослідників як з розвинених країн, так і з країн, що розвиваються. Вона повинна дозволяти їм на справедливій та взаємовигідній основі спільно використовувати наукові ресурси та результати, забезпечуючи рівний доступ до наукових знань як для їх творців, так і для споживачів. Цей доступ повинен бути незалежним від місцезнаходження, національності, расової приналежності, віку, статі, рівня доходу, соціально-економічного стану, етапу кар'єри, дисципліни, мови, релігії, інвалідності, етнічного походження, міграційного статусу або будь-яких інших факторів.

Відкрита наука повинна бути різноманітною та інклюзивною, охоплюючи широкий спектр знань, методів та процедур роботи, мов, результатів і тем досліджень. Це має відповідати потребам і відображати епістемологічний плюралізм наукової спільноти взагалі, різних дослідницьких колективів, науковців і ширшої громадськості, а також носіїв знань за межами традиційного наукового співтовариства.

Це включає в себе розгляд та уважне врахування потреб корінних народів і місцевих громад, а також інших соціальних суб'єктів з різних країн та регіонів. Важливо, щоб наукова спільнота враховувала різноманітність перспектив і досвіду у своїх дослідженнях та висновках, сприяючи таким чином створенню більш об'єктивного та комплексного знання.

Інклюзивність відкритої науки відображається у відкритому доступі до наукових даних, знань і ресурсів, які дозволяють різним групам дослідників і

спільнотам брати участь у наукових дебатах і інноваційних процесах. Це сприяє розширенню можливостей для всіх зацікавлених сторін у сприйнятті, розвитку і використанні наукових знань для вирішення важливих суспільних проблем.

Ось ключові принципи відкритої науки, які надають рамкову основу для створення умов і впровадження методів, спрямованих на підтримку зазначених вище цінностей та втілення ідеалів відкритості в наукових практиках.

Прозорість, контроль, критичний аналіз та можливість відтворення результатів – це принципи, які важливо підтримувати на всіх етапах наукової діяльності. Їх мета полягає в зміцненні надійності та точності наукових відкриттів, сприянні впливу науки на суспільство та збільшенні здатності суспільства вирішувати складні проблеми. Висока відкритість сприяє підвищенню прозорості та довіри до наукової інформації і зміцнює основи науки як форми знань, що базуються на перевірених фактах, логічному аналізі та експертному контролю.

Рівність можливостей відкритої науки означає, що всі вчені, інші суб'єкти та зацікавлені сторони незалежно від їх місцезнаходження, національності, раси, віку, статі, рівня доходу, соціально-економічного стану, етапу кар'єри, дисципліни, мови, релігії, інвалідності, етнічного походження та міграційного статусу чи інших причин мають однакові можливості щодо доступу до відкритої науки, участі в ній та користування її результатами.

Відповідальність, повага та підзвітність – більша відкритість веде до підвищення відповідальності всіх залучених у відкриту науку суб'єктів, яка, поряд із підзвітністю перед громадськістю, своєчасним виявленням конфліктів інтересів та можливих соціальних та екологічних наслідків науково-дослідницької діяльності, інтелектуальною сумлінністю та повагою етичних принципів та особливостей дослідницької діяльності, лягає основою ефективного управління відкритої науки.

Співпраця, участь та інклюзивність – це принципи, які передбачають співпрацю на всіх рівнях наукового процесу, незалежно від географічних, мовних та ресурсних обмежень. Важливо підтримувати міждисциплінарне співробітництво із залученням представників громадськості та використанням знань маргіналізованих спільнот для вирішення соціально значущих проблем.

Гнучкість відкритої науки виникає з різноманітності наукових систем, суб'єктів і ресурсів у всьому світі, а також через швидкий розвиток інформаційно-комунікаційних технологій. В умовах такого різноманіття не

існує універсальних підходів до застосування інструментарію відкритої науки. Важливо підтримувати різні траєкторії розвитку відкритої науки і форм її реалізації, зберігаючи при цьому основні цінності і дотримуючись принципів, що визначені у відповідних документах.

Це означає, що кожна наукова спільнота і кожна країна можуть вибирати і адаптувати підходи до відкритої науки відповідно до своїх потреб, інфраструктури та культурних особливостей. Збереження гнучкості дозволяє використовувати різноманітні методи і інструменти для забезпечення відкритого доступу до наукових даних і знань, що сприяє більш глибокому інтеграції наукових досягнень у глобальному масштабі.

Загальновизнані принципи, такі як прозорість, відкритість, інклюзивність та етичність, відіграють важливу роль у формуванні загальних стандартів і підходів до відкритої науки, забезпечуючи її розвиток на основі глобальних цінностей і норм.

Для того щоб відкрита наука була максимально ефективною і дієвою, вона повинна ґрунтуватися на стійких методах роботи, послугах, інфраструктурі і моделях фінансування, які забезпечують рівноправну участь науковців з менш благополучних установ і країн у довгостроковому плані. Інфраструктура відкритої науки має функціонувати та фінансуватися переважно на некомерційній основі, спираючись на довгострокову стратегію удосконалення методології відкритої науки, що забезпечує постійний та необмежений доступ для всіх зацікавлених сторін.

1.3 Напрямки діяльності

Для досягнення цілей державам-членам рекомендується вживати узгоджених дій у наступних семи областях відповідно до міжнародного права та з урахуванням політичних, адміністративних та правових систем, що склалися в їх країнах (рис. 1.7) [5].

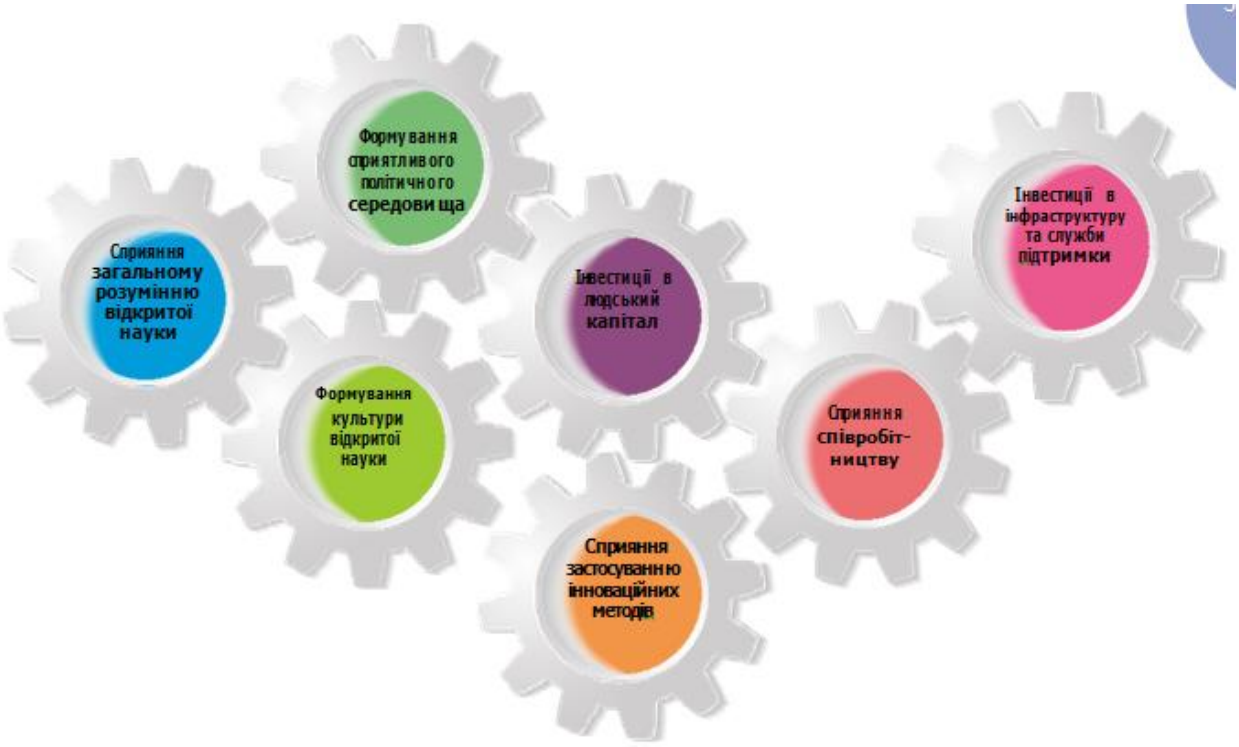

Рисунок 1.7 – Задачі відкритої науки [5]

Поширення загального усвідомлення про відкриту науку, її потенційні переваги та виклики, а також просування різних способів її впровадження.

Державам-членам рекомендується заохочувати та підтримувати єдине розуміння поняття «відкрита наука», як воно визначено у рекомендації, у науковій спільноті та серед різних суб'єктів відкритої науки, а також забезпечувати стратегічне планування та стимулювання діяльності з популяризації відкритої науки на інституційному, національному та регіональному рівнях, поважаючи при цьому різноманітність підходів та практик у галузі відкритої науки. Державам-членам пропонується розглянути можливість вжиття наступних заходів.

Забезпечення інтеграції у відкриту науку викладених у рекомендаціях цінностей та принципів, спрямованих на забезпечення спільного та взаємовигідного використання плодів відкритої науки, що виключають несправедливе та/або нерівноправне отримання даних та знань.

Забезпечення того, щоб фінансовані з державних джерел дослідження проводилися на основі принципів відкритої науки і відповідно до положень рекомендацій, включаючи наукові публікації, відкриті дані досліджень, відкрите програмне забезпечення, вихідні коди та відкрите апаратне забезпечення поширювалися на основі відкритих ліцензій або розміщувалися

у відкритому доступі.

Підтримка різноманіття в наукових публікаціях через впровадження різних форм і методів публікації, зокрема в гуманітарних і соціальних науках, та різних бізнес-моделей, сприяючи некомерційним, академічним і науковим видавничим ініціативам як суспільному благу.

Підтримка багатомовності у дослідницькій практиці, наукових публікаціях та комунікації:

– забезпечення того, щоб потреби та права громад, включаючи права корінних народів на їх традиційні знання, закріплені у Декларації Організації Об'єднаних Націй про права корінних народів 2007 р., не обмежувалися під час практичного застосування методів відкритої науки;

– зміцнення комунікації у сфері відкритої науки на підтримку поширення наукових знань серед науковців, зайнятих в інших галузях досліджень, керівних працівників та широкої громадськості;

– залучення приватного сектору до дискусії про шляхи розширення сфери застосування та спільної реалізації принципів та пріоритетів відкритої науки;

– створення умов для відкритих багатосторонніх дискусій з метою обговорення переваг відкритої науки та пов'язаних з нею реальних та передбачуваних проблем, що стосуються, наприклад, конкуренції, вилучення та експлуатації даних за допомогою більш передових технологій, взаємозв'язків з правами інтелектуальної власності, недоторканності приватного життя, безпеки та нерівного статусу досліджень, що фінансуються з державних та приватних джерел, з метою конструктивного вирішення цих проблем та застосування методів відкритої науки відповідно до викладених у цій рекомендації цінностей та принципів.

Формування сприятливого політичного середовища для відкритої науки.

Державам-членам слідує з обліком їх ситуації, структур управління та конституційних положень формувати або підтримувати на інституційному, національному, регіональному та міжнародному рівнях політичне середовище, сприятливе для впровадження методів відкритої науки та ефективного застосування її практичних інструментів, у тому числі приймати політику стимулювання використання таких інструментів дослідниками. У рамках транспарентного процесу, заснованого на участі різних зацікавлених сторін і, що включає діалог з науковою спільнотою, особливо з молодими вченими та іншими суб'єктами відкритої науки, державам-членам рекомендується

розглянути можливість вжиття таких заходів [6]:

– розробка ефективної інституційної та національної політики та правової бази в галузі відкритої науки відповідно до чинного міжнародного та регіонального законодавства та викладених у цієї рекомендації визначенням, цінностями, принципами та напрямами діяльності;

– узгодження політики, стратегій та дій у галузі відкритої науки на різних рівнях – від окремих установ до місцевого та міжнародного рівнів при повазі до різноманітності наукових підходів.

Всебічний облік аспектів гендерної рівності у політиці, стратегіях та практичних методах відкритої науки:

– заохочення науково-дослідних установ, насамперед тих, що отримують державні кошти, до застосування політики та стратегій на користь відкритої науки;

– заохочення науково-дослідних установ, університетів, наукових спілок та асоціацій, а також науковців до прийняття програмних заяв у руслі цієї рекомендації з метою підтримки впровадження методів відкритої науки у взаємодії з національними академіями наук, асоціаціями молодих учених, такими як академії молодих учених, та Міжнародною радою по науці (МСН);

– розширення інтеграції громадянської та заснованої на широкій участі науки в політику та практику відкритої науки на національному та інституційному рівнях, а також на рівні фінансуючи організацій;

– розробка моделей, що забезпечують спільне створення знань за участю різних суб’єктів, та вироблення принципів визнання партнерств, що виходять за межі наукової спільноти;

– підтримка відповідальної практики оцінки та аналізу досліджень та дослідників, що стимулює якісну науку та визнає різноманітність результатів досліджень, видів діяльності та завдань;

– сприяння створенню рівноправних державно-приватних партнерств в інтересах відкритої науки та залучення приватного сектору до діяльності в галузі відкритої науки за умови наявності відповідної сертифікації та інструментів регулювання, що перешкоджають виникненню залежності від конкретних постачальників, хижацькій поведінці та несправедливій та/або державою наукової діяльності. Враховуючи суспільний інтерес до відкритої науки та роль державного фінансування, держави-члени повинні забезпечити належне функціонування ринку пов’язаних з дослідницькою діяльністю та відкритою наукою послуг у глобальних та суспільних інтересах та запобігти

домінуванню будь-яких комерційних структур;

– розробка, здійснення та моніторинг політики та стратегій фінансування та інвестицій у галузі відкритої науки на основі її ключових цінностей та принципів. Супутні впровадження відкритої науки витрати, пов'язані з підтримкою методів відкритої науки у сфері досліджень, публікацій, роботи з даними та методів програмування, зі створенням та освоєнням об'єктів інфраструктури та служб підтримки відкритої науки, а також з нарощуванням потенціалу всіх учасників процесу та застосуванням новаторських, заснованих на широкому співробітництві та взаємодії методів організації, наукової діяльності.

Інвестиції в інфраструктуру та служби підтримки відкритої науки:

– відкрита наука вимагає і заслуговує на систематичні та довгострокові стратегічні інвестиції у наукові технології та інновації та, насамперед, у технічну та цифрову інфраструктуру та пов'язані з нею служби підтримки, у тому числі в їх довгострокову експлуатацію. Такі інвестиції мають здійснюватися як формі фінансових, і кадрових ресурсів. З урахуванням того, що відкрита наука є глобальним суспільним благом, служби підтримки відкритої науки слід розглядати як найважливішу частину дослідницької інфраструктури, що належить керуючому їй співтовариству та спільно фінансується урядами, спонсорами та установами, що відображають різні інтереси та потреби наукової спільноти та суспільства загалом. Державам членам рекомендується сприяти розвитку некомерційної інфраструктури відкритої науки та забезпечувати належні інвестиції в наступних областях;

– наука, технології та інновації, прагнучи спрямовувати на фінансування наукових досліджень та розробок, орієнтовно не менше 1% національного валового внутрішнього продукту (ВВП);

– надійний доступ до Інтернету з належною пропускною здатністю для вчених та споживачів наукової продукції у всьому світі;

– національні дослідні та освітні мережі (НДОМ) та їх функціональні можливості, заохочуючи регіональне та міжнародне співробітництво з метою забезпечення максимальної функціональної сумісності та узгодження служб НДОМ;

– об'єкти некомерційної інфраструктури, включаючи комп'ютерні засоби та суспільну цифрову інфраструктуру та послуги, що підтримують підходи відкритої науки. Вони мають сприяти забезпеченню довгострокового зберігання, супроводу та громадського контролю наукової продукції,

включаючи наукову інформацію, дані, вихідні коди та технічні параметри апаратного забезпечення, співпрацю між науковцями, а також поширення та повторного використання продуктів наукових досліджень. Усі об’єкти інфраструктури та служби підтримки, що обслуговують дослідницьку діяльність, повинні спиратися на потужну громадську базу і забезпечувати функціональну сумісність та інклюзивність. Цифрова інфраструктура відкритої науки повинна, по можливості, спиратися на комплекти програмного забезпечення з відкритим вихідним кодом. Таку відкриту інфраструктуру можна підтримувати шляхом прямого фінансування та за рахунок виділення певного відсотка коштів із кожного фінансованого гранту;

– взаємопов’язана та диверсифікована інформаційно-технологічна інфраструктура відкритої науки, включаючи, де це необхідно, високопродуктивні обчислювальні системи, хмарні обчислення та сховища даних, а також надійні, відкриті та керовані спільнотою об’єкти інфраструктури, протоколи та стандарти, потрібні для підтримки бібліорізноманіття та взаємодії із суспільством. Уникаючи фрагментації шляхом зміцнення взаємопов’язаності існуючих об’єктів інфраструктури та служб підтримки відкритої науки на національному, регіональному та міжнародному рівнях, слід приділяти увагу забезпеченню загального доступу до цієї інфраструктури, її міжнародної інтеграції, максимально можливої функціональної сумісності та відповідності певним базовим параметрам, таким, наприклад, як принципи управління даними FAIR (зручність пошуку, доступність, функціональна сумісність та можливість повторного використання) та CARE (колективна користь, контрольні повноваження, відповідальність та етика) [7]. Слід також розглянути технічні вимоги до кожного цифрового об’єкта, що має значення для науки, незалежно від того, чи йдеться про елемент даних, набір даних, метаданих, програмний код або публікацію. Потужності інфраструктури управління даними повинні обслуговувати потреби всіх наукових дисциплін на рівноправній основі, незалежно від обсягу та характеру даних, які вони використовують, та методів їх обробки. Інфраструктура та послуги відкритої науки повинні бути орієнтовані на потреби вчених та інших цільових аудиторій користувачів, передбачати розробку відповідних для їх методів роботи функцій, та використовувати зрозумілі оболонки для взаємодії з користувачем. Слід також приділяти увагу постійним ідентифікаторам цифрових об’єктів. Як приклади можна навести визначення та атрибуцію відповідних відкритих постійних

ідентифікаторів для кожного типу цифрових об’єктів, метаданих, необхідних для забезпечення їх ефективної оцінки, доступності, застосування та повторного використання, а також належне управління даними, що здійснюється заслуговують на довіру регіональними або глобальними мережами сховищ даних;

– громадські домовленості, досягнуті в рамках регіональних або глобальних дослідницьких співтовариств та визначають методи обміну даними, що використовуються спільнотою, формати даних, стандарти метаданих, онтології, термінологію, інструменти та інфраструктуру. Сприяння у виробленні таких домовленостей можуть надати міжнародні наукові спілки та асоціації, регіональні та національні об’єкти дослідницької інфраструктури та редакційні поради журналів. Крім того, конвергенція різних семантичних артефактів (зокрема, словників, таксономій, онтологій та схем метаданих) має важливе значення для забезпечення функціональної сумісності та повторного використання даних у рамках міждисциплінарних досліджень;

– співробітництво з метою оптимізації використання об’єктів інфраструктури та спільні стратегії створення багатонаціональних, регіональних та національних платформ відкритої науки загального користування, у тому числі шляхом сприяння спільним дослідженням, спільного використання інфраструктури відкритої науки, технічної допомоги, передачі та спільного виробництва, пов’язаних з відкритою наукою технологій та обміну передовим досвідом на взаємно узгоджених умовах. Такі ініціативи є механізмом координації підтримки відкритої науки, що охоплює доступ до служб та дослідницької інфраструктури відкритої науки (включаючи засоби зберігання, супроводу та спільного використання даних), узгодження політики, освітні програми та технічні стандарти. Враховуючи, що ці ініціативи здійснюються у різних регіонах, важливо, щоб вони були функціонально сумісні з погляду політики, методів роботи та технічних специфікацій. Крім того, важливим завданням є інвестиції в програми фінансування, що дозволяють вченим створювати та використовувати такі платформи, особливо у країнах з низьким та середнім рівнем доходу;

– нове покоління інструментів на основі відкритих інформаційних технологій, що автоматизують процес пошуку та аналізу взаємопов’язаних публікацій і даних і підвищують швидкість та ефективність процесу висування та перевірки гіпотез. Ці інструменти та сервіси дають максимальну віддачу при використанні в рамках системи відкритої науки, що виходить за рамки

інституційних, національних та міждисциплінарних кордонів та при цьому враховує потенційні ризики та етичні проблеми, які можуть виникнути у зв'язку з розробкою та застосуванням таких інструментів з використанням технологій на основі штучного інтелекту;

– новаторські методи роботи на різних етапах наукового процесу та міжнародне наукове співробітництво;

– фінансування для покриття необхідних витрат, пов'язаних із переходом на практичні методи відкритої науки та їх подальшим використанням, а також сприяння поширенню схем відкритого ліцензування;

– інфраструктура для нецифрових матеріалів (наприклад, реагентів);

– платформи для обміну інформацією та спільного створення знань вченими та суспільством, у тому числі за допомогою передбачуваного та сталого фінансування волонтерських організацій, які проводять дослідження методами громадянської науки за участю місцевого населення;

– громадські системи моніторингу та інформації, що доповнюють національні, регіональні та глобальні системи даних та інформації.

Фінансування людського капіталу, підготовки, освіти, цифрової грамотності і розвитку потенціалу для сприяння відкритій науці.

Відкрита наука потребує інвестицій у нарощування потенціалу та людський капітал. Трансформування наукової практики з метою її адаптації до змін, проблем, можливостей і ризиків цифрової епохи, що настала в XXI столітті, вимагає проведення цільових досліджень і розробки освітніх програм і програм підготовки кадрів, націлених на розвиток навичок роботи з новими технологіями і які охоплюють питання етики та практики відкритої науки. Державам-членам пропонується розглянути можливість наступних заходів:

– забезпечення систематичного та безперервного нарощування потенціалу в галузі концепцій та практик відкритої науки, у тому числі щодо широкого розуміння керівних принципів та основних цінностей відкритої науки, а також технічних навичок та компетенцій у сфері цифрової грамотності, цифрових методів спільної роботи, науки про дані та управління ними, курування, довгострокового зберігання та архівування матеріалів, інформаційної грамотності та грамотності у поводженні з даними, інтернетбезпеки, прав власності на контент та його спільного використання, а також розроблення програмного забезпечення та інформатики;

– узгодження набору компетенцій у сфері відкритої науки з окремими дисциплінами для дослідників, що знаходяться на різних етапах своєї кар'єри,

а також для суб’єктів державного та приватного сектору, яким необхідні конкретні навички для використання продуктів відкритої науки у своїй кар’єрі, а також розвиток затребуваних навичок та проведення програм підготовки, націлених на формування відповідних компетенцій. Набір базових навичок у галузі науки про дані та управління ними, компетенцій, пов’язаних із законодавством про інтелектуальну власність, а також навичок, необхідних для забезпечення відкритого доступу та відповідної взаємодії з суспільством, має розглядатися як частина ключових компетенцій усіх дослідників та включатись до навчальної програми вищих навчальних закладів щодо розвитку дослідницьких навичок;

– створення умов для розвитку відкритої науки також передбачає наявність регулюючих органів у галузі даних, які здатні у співпраці з науковою спільнотою виробляти стратегічні вказівки в галузі управління даними та забезпечення відкритого доступу до них на національному або місцевому рівнях та висококваліфікованих професійних керуючих даними, які здійснюють управління даними та їх курування, відповідно до узгоджених принципів, зокрема, принципів FAIR і CARE, які в рамках заслуговують на довіру установ або служб. Щоб скористатися наданими відкритою наукою можливостями, дослідницьким проектам, дослідницьким інститутам та ініціативам громадянського суспільства, необхідно залучати людей, які мають спеціалізовані навички в галузі науки про дані, у тому числі у сфері аналізу, статистики, машинного навчання, штучного інтелекту, візуалізації, програмування та застосування алгоритмів із дотриманням принципів наукової та етичної відповідальності;

– сприяння використанню відкритих освітніх ресурсів (ВОР) відповідно до визначення, що міститься в Рекомендації ЮНЕСКО про відкриті освітні ресурси (ВОР) 2019 р., як інструмент нарощування потенціалу в галузі відкритої науки. Таким чином, ВОР повинні використовуватися для розширення доступу до освітніх та дослідних ресурсів відкритої науки, покращення результатів навчання, забезпечення максимальної віддачі від державного фінансування та надання викладачам та здобувачам освіти можливостей для участі у створенні знань;

– підтримка наукової комунікації щодо методів відкритої науки з метою поширення наукових знань серед науковців, які представляють інші наукові дисципліни, осіб, відповідальних за прийняття рішень та широкої громадськості в цілому. Розповсюдження наукової інформації за допомогою

наукової журналістики та засобів інформації, популяризація науки, відкриті лекції та різні повідомлення в соціальних мережах зміцнюють суспільну довіру до науки, сприяючи залученню в цю сферу інших громадських діячів, які не належать до наукової спільноти. Найважливіше значення для наукової комунікації з питань відкритої науки мають якість оригінальних джерел інформації та їхнє коректне цитування, які дозволяють уникнути неправильного тлумачення та поширення невірної інформації.

Створення культури відкритої науки і синхронізація мотивацій для її впровадження.

Державам членам рекомендується, з урахуванням їхньої ситуації, структур управління та конституційних положень та, відповідно до міжнародних та національних правових норм, брати активну участь у усуненні бар'єрів, що перешкоджають розвитку відкритої науки, особливо у сфері оцінки наукових досліджень та професійної діяльності та винагороди за їх результатами. Системи оцінки наукового внеску та посадового зростання, що заохочують застосування передових методів відкритої науки, необхідні її впровадження. Увага слід також приділяти запобіганню та пом'якшенню небажаних негативних наслідків застосування методів відкритої науки, таких, як: корислива поведінка, міграція даних, експлуатація та приватизація даних наукових досліджень, збільшення витрат для вчених та високі збори за підготовку статей, які використовуються в деяких бізнес-моделях наукових публікацій: вони можуть призвести до нерівності серед наукових спільнот у всьому світі та, в деяких випадках, до втрати прав інтелектуальної власності та знань. Державам-членам рекомендується розглянути можливість прийняття наступних заходів:

– об'єднання зусиль широкого кола різних зацікавлених сторін, включаючи фінансові дослідження організації, університети, науково-дослідні установи, видавці та редактори, а також наукові товариства, що представляють різні дисципліни та країни, з метою зміни нинішньої культури наукових досліджень та визнання вкладу дослідників у поширення знань, співробітництво та взаємодія з іншими дослідниками тасуспільством, а також особливою підтримкою молодих вчених, які стануть провідною силою цих культурних змін;

– перегляд систем оцінки досліджень та результатів професійної діяльності з метою приведення їх у відповідність до принципів відкритої науки. З урахуванням того, що послідовне впровадження відкритої науки потребує

витрат часу, ресурсів та зусиль, які не можуть бути автоматично конвертовані у традиційні наукові продукти, такі як публікації, але які можуть надати значний вплив на науку та суспільство, системи оцінки повинні враховувати широке коло завдань у межах середовища, у якому створюються знання. Ці завдання пов'язані з різними формами створення та поширення знань, які не обмежуються публікаціями в міжнародних журналах, що рецензуються.

– сприяння розробці та впровадженню систем оцінки та аналізу, які:

1) спираються на існуючі інструменти вдосконалення методів оцінки наукових результатів, такі як СанФранциська декларація про оцінку досліджень 2012 р., наголошуючи на якості результатів досліджень, а не на їх кількості, та на цільовому використанні різноманітних показників та процесів, які не пов'язані з публікаціями в журналах, наприклад, з рейтингом цитування журнальних статей;

2) гідно оцінюють всі складові дослідницької діяльності та наукових продуктів, включаючи відповідні принципам FAIR, якісні дані та метадані, докладно описані та придатні для повторного використання програмні продукти, протоколи та робочі процедури, машиночитані зведення результатів, а також викладання, інформаційно-роз'яснювальну роботу та взаємодію з громадськими діями;

3) враховують показники впливу досліджень та обміну знаннями, такі як розширення участі у дослідницькому процесі, вплив на політику та практику та взаємодію у сфері відкритих інновацій з партнерами за межами наукової спільноти;

4) враховують той факт, що різноманітність дисциплін потребує різних підходів з погляду відкритої науки;

5) враховують той факт, що оцінка дослідників за критеріями відкритої науки повинна бути адаптована до різних етапів кар'єри, приділяючи особливу увагу дослідникам, які знаходяться на початку професійного шляху.

– забезпечення широкої популярності практичних навичок у галузі відкритої науки та їх обліку як критерій найму на роботу та просування по службі у науковому та академічному середовищі;

– заохочення фінансуючих організацій, дослідницьких установ, редакційних колегій журналів, наукових товариств та видавців до проведення політики, що передбачає та стимулює відкритий доступ до наукових знань, включаючи наукові публікації, дані відкритих досліджень, відкрите програмне забезпечення, вихідні коди та відкрите апаратне забезпечення, відповідно з

положеннями цієї рекомендації;

– забезпечення різноманіття форм наукової комунікації з урахуванням принципів відкритого, прозорого та справедливого доступу, а також підтримка некомерційних та партнерських видавничих моделей, що не включають оплату за підготовку наукових статей та книг;

– застосування ефективних заходів управління та належного законодавчого регулювання з метою вирішення проблеми нерівності та запобігання, пов'язаній з нею корисливої поведінки, а також з метою захисту інтелектуальної творчості, націленої на створення методів, продуктів та даних відкритої науки;

– сприяння розповсюдженню матеріалів, що знаходяться в громадському доступі, та розвитку існуючих схем відкритих ліцензій та винятків з авторського права та інших прав інтелектуальної власності у наукових та освітніх цілях, що дозволяють поширювати та повторно використовувати захищені авторським чи іншим правом інтелектуальної власності твори (у тому числі частково або в переробленому вигляді) за умови належної вказівки авторства та відповідно до міжнародного права;

– сприяння проведенню якісних та відповідальних досліджень відповідно до Рекомендації ЮНЕСКО щодо науки та науковців (дослідників) 2017 р. та вивчення потенціалу методів відкритої науки для скорочення недобросовісної поведінки у сфері науки, включаючи підробку та фальсифікацію результатів, порушення норм наукової етики та плагіат.

Сприяння застосуванню інноваційних методів відкритої науки на різних етапах наукового процесу.

Відкрита наука потребує відповідних змін у науковій культурі, методологіях, установах та інфраструктурі, а її принципи та методи охоплюють весь цикл досліджень, починаючи з формулювання гіпотез, розробки та тестування методологій, збору, аналізу, обробки та зберігання даних, колегіальної оцінки та перевірки результатів іншими методами та закінчуючи аналізом, критикою та тлумаченням, поширенням та взаємною перевіркою ідей та результатів, публікацією, поширенням, впровадженням, застосуванням та повторним використанням результатів. Відкрита наука постійно розвивається, і надалі з'являться нові методи. З метою сприяння використанню інноваційних методів забезпечення відкритості на різних етапах наукового процесу державам-членам рекомендується розглянути можливість наступних заходів:

– заохочення застосування методів відкритої науки, починаючи з найперших етапів дослідницького процесу, та розповсюдження принципів відкритості на всі його стадії з метою підвищення якості та відтворюваності результатів, у тому числі шляхом сприяння спільній роботі з ініціативи співтовариств та використання інших інноваційних моделей, наприклад випуску попередніх публікацій, які мають бути чітко відмежовані від минулих рецензування остаточних публікацій, а також формування поваги до різноманітності наукових методів з метою прискорення поширення наукових знань та заохочення їх швидкого накопичення;

– належне заохочення застосування відкритих процедур колегіального рецензування, які можуть передбачати розкриття особистості рецензентів, публікацію рецензій та можливість участі широкого загалу у процесі коментування та оцінки;

– заохочення і визнання цінності публікації та поширення негативних наукових результатів та результатів, що не відповідають очікуванням дослідників, які їх отримали, а також пов’язаних з ними даних, оскільки такі результати також сприяють розвитку науки;

– розробка нових, інклюзивних методів та механізмів перевірки з метою обліку та визнання цінності думок соціальних суб’єктів, що перебувають за межами традиційної наукової спільноти, у тому числі за допомогою інструментів громадянської науки, наукових проектів, що спираються на пряме колективне фінансування, залучення громадян до роботи громадських архівних установ та інших форм наукової діяльності, заснованої на широкій участі;

– розробка інклюзивних стратегій виявлення потреб маргіналізованих співтовариств та висвітлення соціально значущих питань, що підлягають включенню до програм досліджень у галузі науки, технологій та інновацій (НТІ);

– розробка стратегій, що полегшують розміщення даних в архівах з метою сприяння їхньому куруванню та збереженню, а також з метою створення можливостей для їх використання, у тому числі повторного, протягом відповідного періоду часу;

– сприяння розвитку спільних об’єктів інфраструктури для збору та зберігання програмного забезпечення з відкритим вихідним кодом та самих вихідних кодів, а також для забезпечення зручного для користувачів доступу до них;

– надання підтримки вченим та іншим членам суспільства у створенні та використанні ресурсів на базі відкритих даних у міждисциплінарному режимі з метою одержання максимальної наукової, соціальної, економічної та культурної користі та стимулювання формування гібридних міждисциплінарних просторів для спільної роботи, в яких представники, що представляють різні дисципліни, зможуть взаємодіяти розробниками програмного забезпечення, програмістами, творцями контенту, новаторами, інженерами, художниками та багатьма іншими;

– заохочення спільного використання, сприяння забезпеченню функціональної сумісності великомасштабних об'єктів дослідчої інфраструктури, наприклад, що обслуговують міжнародні програми з фізики, астрономії та космічної науки та партнерські проекти в інших галузях, таких як охорона здоров'я, екологічні та соціальні науки, а також розширення відкритого доступу до таких об'єктів;

– заохочення відкритої інноваційної практики, що поєднує методи відкритої науки з можливостями прискореного перекладу іншими мовами та розвитку новаторських розробок. Як і відкрита наука, відкрита інноваційна практика та інші партнерські зв'язки в галузі відкритої науки передбачають активне залучення до інноваційного процесу великої кількості учасників, а також визначення та розробку бізнес-моделі, що забезпечує ефективну комерціалізацію нових знань.

Підтримка міжнародного та багатостороннього співробітництва в рамках відкритої науки для зменшення розривів у цифровому та технологічному середовищі, а також у обміні знаннями.

Для сприяння розвитку відкритої науки в глобальному масштабі держави-члени повинні заохочувати та зміцнювати міжнародне співробітництво між усіма суб'єктами відкритої науки. З урахуванням значимості поточних зусиль та заходів у контексті відкритої науки на благо науки та суспільства в цілому, державам-членам пропонується розглянути можливість прийняття наступних заходів:

– заохочення міжнародного наукового співробітництва як одного з невід'ємних елементів відкритої науки, найважливішої рушійної сили інтенсивного обміну науковими знаннями та досвідом та ключовою передумовою відкритості науки;

– заохочення та стимулювання багатосторонньої транскордонної взаємодії в галузі відкритої науки, у тому числі з використанням існуючих

транснаціональних, регіональних та глобальних механізмів та організацій співробітництва. Ця діяльність також має бути спрямована на об'єднання зусиль з метою забезпечення загального доступу до результатів наукової діяльності незалежно від дисципліни, географічного місцезнаходження, ґендерної та етнічної приналежності, мови та соціально-економічних умов, або від будь-яких інших причин, розвиток та використання спільних об'єктів інфраструктури відкритої науки, а також на технічну допомогу та передачу технологій, нарощування потенціалу, створення сховищ даних та спільнот спеціалістів-практиків, а також зміцнення солідарності між усіма країнами незалежно від рівня розвитку в них відкритої науки;

– створення регіональних та міжнародних механізмів фінансування з метою підтримки та зміцнення позицій відкритої науки та виявлення механізмів, здатних підтримати дії, що вживаються на міжнародному, регіональному та національному рівнях, включаючи механізми партнерства;

– сприяння створенню та підтримці ефективних мереж співпраці для обміну передовими методами роботи в галузі відкритої науки та досвідом, накопиченим у ході розробки, вдосконалення та здійснення політики, ініціатив та методів роботи у сфері відкритої науки;

– сприяння співпраці між країнами у нарощуванні потенціалу відкритої науки, у тому числі в галузі розвитку інфраструктури, забезпечення стійкості програмного забезпечення, управління даними та їх супроводу, а також у сфері запобігання транскордонній експлуатації відкритих даних та їх неправомірного використання;

– сприяння міжнародному співробітництву у галузі розробки показників для відкритої науки;

– доручення ЮНЕСКО завдань щодо координації у взаємодії з державами-членами та відповідними зацікавленими сторонами розробки та затвердження комплексу цілей відкритої науки, які будуть спрямовувати та стимулювати міжнародне співробітництво в інтересах розвитку відкритої науки на благо людства та для забезпечення стійкості у масштабах планети.

1.4 Моніторинг

Державам-членам слід з урахуванням їхньої ситуації, структур управління та конституційних положень, проводити моніторинг політики та

механізмів у сфері відкритої науки з використанням того чи іншого поєднання кількісних та якісних методів за обставинами.

Державам-членам пропонується розглянути можливість вживання таких заходів:

– впровадження відповідних механізмів моніторингу та оцінки для вимірювання ефективності та дієвості політики та стимулів у галузі відкритої науки у порівнянні з її заявленими цілями, включаючи виявлення непередбачених та потенційно негативних наслідків, особливо для молодих учених;

– збір та розповсюдження інформації про виконану роботу, передовий досвід та інновації, а також звітів про дослідження, присвячені відкритій науці та наслідкам її впровадження, за підтримки ЮНЕСКО та із залученням різних зацікавлених сторін;

– аналіз можливості розробки механізму моніторингу, що включає якісні та кількісні показники, в рамках національних стратегічних планів, що надаються на міжнародному рівні, що містять завдання та заходи щодо виконання цієї рекомендації на короткострокову, середньострокову і довгострокову перспективу. Моніторинг відкритої науки слід відкрито проводити під громадським контролем, у тому числі з боку наукової спільноти, наскільки це можливо в рамках відкритої, не захищеної правами власності та прозорої інфраструктури. Цей аспект моніторингу може бути делегований приватному сектору;

– розробка стратегій моніторингу результативності та довгострокової ефективності відкритої науки за участю різних заінтересованих сторін. Такі стратегії можуть бути спрямовані на зміцнення взаємозв’язків між наукою, політикою та суспільством, а також на підвищення прозорості та підзвітності з метою проведення якісних та відповідальних принципам інклюзивності та рівноправності досліджень, що сприяють ефективному вирішенню глобальних проблем.

1.5 Управління даними досліджень та дані FAIR

Дослідницькі дані включають різноманітну інформацію, що зібрана, виявлена, згенерована або створена для підтвердження оригінальних результатів дослідження. Ці дані можуть приймати форму аудіо- та відеозаписів, електронних таблиць, документів, анкет, результатів тестів,

стенограм інтерв'ю, зображень, фотографій, лабораторних зошитів, полевих нотаток, коду, програмного забезпечення, зразків, екземплярів і артефактів [8].

Наукові статті традиційно базуються на даних експериментів, проте традиційний формат їхньої подачі часто обмежує можливості представлення даних. З розвитком технологій відкриваються нові шляхи для спільного публікування даних разом зі статтями або навіть окремо від них. Це підвищує достовірність досліджень і забезпечує їхню відтворюваність та багаторазовість використання результатів [9].

Відсутність даних може підірвати достовірність результатів дослідження і підкласти під сумнів їхню перевірку.

Управління даними протягом всього процесу наукового дослідження критично важливе. Цей процес можна описати за допомогою циклу даних досліджень (Research Data Cycle, RDC) (рис. 1.8).

Рисунок 1.8 – Процес управління даними [5]

Ось основні принципи FAIR для належного управління науковими даними:

1) Findable (відшуканість):

– присвоєння унікального ідентифікатора для кожного набору даних;

– публікація метаданих в Інтернеті для їхнього легкого пошуку;

2) Accessible (доступність):

– забезпечення доступу до даних з мінімальними обмеженнями, враховуючи конфіденційність та законодавство;

– забезпечення доступності даних через стандартизовані протоколи;

3) Interoperable (сумісність):

– використання стандартних мов інформаційного опису для метаданих;

– забезпечення використання семантичних та технічних стандартів для забезпечення інтерпретації даних;

4) Reusable (багаторазовість):

– публікація даних з документованим та структурованим форматом, що дозволяє їхнє повторне використання;

– забезпечення чіткої інструкції щодо умов використання даних, включаючи ліцензії та вимоги до цитування.

Ці принципи спрямовані на забезпечення максимальної відкритості та доступності даних для наукової спільноти, з урахуванням необхідності захисту конфіденційної інформації та виконання вимог законодавства про інтелектуальну власність.

Принципи FAIR стосовно відшуканості та доступності даних можуть бути сформульовані наступним чином:

1) відшукувані (Findable):

– F1. Кожному набору (мета)даних присвоюється унікальний і постійний ідентифікатор;

– F2. Дані супроводжуються значним обсягом метаданих, що дозволяє їхнє ефективне пошукове індексування;

– F3. (Мета)дані реєструються чи індексуються в достатньо легко знаходженому ресурсі;

– F4. Метадані містять посилання на ідентифікатори самого набору даних;

2) доступні (Accessible):

– дані мають бути доступними у вільному доступі;

– дані повинні зберігатися у спеціалізованому репозитарії для забезпечення стабільного доступу до них;

3) сумісні (Interoperable):

– I1. Метадані повинні використовувати формальну, загальну і широковживану мову для представлення знань;

– I2. Використання лексикону, який відповідає принципам FAIR, для опису даних;

– I3. Включення кваліфікованих посилань на різні дані і метадані, що дозволяє їхнє правильне інтегрування;

4) доступними для багаторазового використання (Reusable):

– R1. Дані мають бути детально описані з великою кількістю точних і відповідних атрибутів;

– R2 Дані мають бути видаються з чіткою і вільною ліцензією на використання;

– R3. Дані пов’язані з їхнім походженням і метадані;

– R4. Дані відповідають стандартам спільноти, що стосуються певної предметної області.

Ці принципи спрямовані на створення умов для максимальної використовуваності наукових даних. Їхня реалізація передбачає використання стандартизованих форматів, лексиконів та метаданих, що забезпечують їхню інтерпретацію та інтеграцію як людьми, так і машинами. Окрім того, важливою є чітка ліцензійна політика, що дозволяє використання даних з мінімальними обмеженнями, що сприяє їхньому повторному використанню та репродукції результатів досліджень.

Реалізація цих принципів вимагає від дослідників та організацій управління даними відповідно до найкращих практик та використання сучасних інструментів і технологій для забезпечення відкритості, доступності та інтегрованості наукових даних.

Наприклад, нещодавно в Чехії було відкрито новий центр PID, який обслуговує національні консорціуми для надання ідентифікаторів DataCite DOI та ORCID. Це сприятиме ширшому застосуванню ідентифікаторів у видачі IT-послуг, що, за довгостроковою перспективою, може зменшити витрати.

A – доступність

Дані та метадані мають бути доступними за простою схемою доступу, часто через стандартний протокол HTTP. Це включає налаштування доступу з використанням процедур автентифікації та авторизації. Використання сховищ даних часто є достатнім для виконання більшості вимог. Важливою є також доступність метаданих, навіть коли самі дані вже не доступні.

I – сумісність

Дані мають бути відкритими та сумісними з іншими джерелами і інструментами. Вибір формату даних повинен бути обґрунтованим, з урахуванням можливостей і потреб користувачів.

R – багаторазовість

Дані мають бути готовими до багаторазового використання з відповідною ліцензійною політикою. Це означає, що користувачі повинні мати чітке уявлення про умови їх використання та контекст утворення даних.

Це ініціатива, яка відповідає принципам FAIR і сприяє кращому управлінню даними досліджень.

Інституційні репозитарії створюються установами для власних потреб і забезпечують безпечне зберігання та обмін даними, що створюються дослідниками. Ці репозитарії відповідають специфічним вимогам установи і дозволяють підвищити контроль над даними.

Загальні репозитарії, навпаки, призначені для всіх типів дослідницьких даних без суворих обмежень. Вони забезпечують послуги відшукуваності та доступності, містять метадані та постійні ідентифікатори. Такі репозитарії, як Zenodo, підтримуються великими науковими організаціями, включаючи Європейську комісію, та надають широкий доступ до даних.

При виборі репозитарію дослідники часто вдаються до предметно-орієнтованих репозитаріїв, а потім до інституційних або загальних, залежно від їхніх конкретних потреб і вимог дослідження.

Питання для самоконтролю

1. Поясніть значення «відкрита наука» та «відкриті дані».
2. Опишіть відкриті принципи.
3. Охарактеризуйте відтворювані дослідження та аналіз даних.
4. Означте, як здійснюється управління та обмін даними досліджень.
5. Як оприлюднюється розробка ПЗ з відкритим кодом?
6. Опишіть переваги та недоліки розробки ПЗ з відкритим кодом.
7. Охарактеризуйте оцінку дослідників щодо відкритої науки.
8. Означте основні принципи загальнонаукових досліджень.
9. Проаналізуйте, що являють собою факти, наукові закони, поняття, принципи, концепції, парадигма.
10. Визначте із якими труднощами може стикнутися дослідник у ході проведення експерименту.

Тестові завдання

1. До первинних джерел наукової інформації не належать:
a) Монографії.
b) Енциклопедії.
c) Періодичні видання.
d) Дисертації.

2. Базовими завданнями, які виконують пошукові системи, є:
a) Індексування.
b) Визначення рейтингу.
c) Сканування.
d) Все вищеперераховане.

3. Кожна пошукова система використовує комп'ютерні програми для аналізу веб-сторінок. Такі програми називаються:
a) Веб-сканерами.
b) Веб-серверами.
c) Веб-додатками.
d) Веб-браузерами.

4. Першою академічною пошуковою системою була:
a) Microsoft Academic Search.
b) CiteSeer.
c) AOL.
d) Archie.

5. Google Scholar підтримує такі оператори розширеного пошуку:
a) .. + - «» AND intext:
b) … + - «» AND intitle:
c) . + - «» AND intitle:
d) .. + - «» OR intitle:

6. Електронні бази даних наукової інформації включають:
a) Бази цитувань (наукометричні бази).
b) Повнотекстові бази даних.

c) Бібліографічні (реферативні) бази даних.
d) Все вищеперераховане.

7. До баз даних цитувань належать:
a) Scopus, EBSCO, Google Scholar.
b) Scopus, Web of Science, Google Scholar.
c) Scopus, Web of Science, ProQuest.
d) Scopus, Web of Science, EBSCO.

8. Яким символом потрібно користуватися під час пошуку в базах даних, якщо закінчення пошукового слова може варіюватися (наприклад, computer, computers, computing)?
a) *
b) !
c) &
d) @

9. Фрази (тобто кілька слів) для пошуку у базах даних беруться у:
a) фігурні дужки, наприклад: {molecular genetics}
b) квадратні дужки, наприклад: [molecular genetics]
c) круглі дужки, наприклад: (molecular genetics)
d) лапки, наприклад: “molecular genetics”

10. Пошук з використанням логічних операторів знайде хоча б одне з ключових слів, або обидва, якщо в результатах пошуку є обидва, - з допомогою оператору:
a) AND NOT.
b) NOT.
c) OR.
d) AND.

***Рекомендована література*:**

1. An introduction to the UNESCO Recommendation on Open Science. Режим доступу: https://www.unesco.org/en/articles/introduction-unesco-recommendation-open-science (дата звернення: 14 січня 2024 р.).

2. Andrzejewski S., Julliard R. Citizen Science: Producing Data With People for Innovating Research. Zenodo. 2022, June 7. https://doi.org/10.5281/zenodo.6642538.

3. EU-citizen.science. Режим доступу: https://eu-citizen.science/ (дата звернення: 14 січня 2024 р.).

4. European Commission: Open Science. Режим доступу: https://researchand-innovation.ec.europa.eu/system/files/2019-12/ec_rtd_factsheet-open-science_2019.pdf (дата звернення: 14 січня 2024 р.).

5. How European Research Libraries Can Support Citizen-Enhanced Open Science/ Kaarsted T., Blake O., Nielsen K.H., Alving B., Rasmussen L.T., Overgaard A.K., Hansen S.M.-B.// Open Information Science, vol. 7, no. 1. 2023, pp. 202-214. https://doi.org/10.1515/opis-2022-0146.

6. Jon Tennant Elsevier are corrupting open science in Europe. Режим доступу:
https://www.theguardian.com/science/politicalscience/2018/jun/29/elsevier-are-corrupting-open-science-in-europe (дата звернення: 14 січня 2024 р.).

7. Merry Work: Libraries and Citizen Science/ Ignat T., Ayris P., Labastida i Juan I., Reilly S., Dorch B. F., Kaarsted T., & Overgaard A.K.// Insights, 31. 2018. https://doi.org/10.1629/uksg.431.

8. Ten Principles of Citizen Science/ Robinson L.D., Cawthray J.L., West S.E., Bonn A. & Ansine J.// In S. Hecker, M. Haklay, A. Bowser, Z. Makuch, J. Vogel, & A. Bonn. Citizen Science: Innovation in Open Science, Society and Policy. London, UCL Press. 1–23. 2018. https://doi.org/10.14324/111.9781787352339.

9. Відкриті наукові основи та практика управління інформацією [Текст]: Конспект лекцій для здобувачів спеціальності 121 Інженерія програмного забезпечення ОП «Інженерія програмного забезпечення» за другим (магістерським) рівнем вищої освіти денної та заочної форм навчання / уклад О. М. Суринович. Луцьк : ЛНТУ, 2022. 72 с.

10. Відкриті наукові основи та практика управління інформацією [Текст]: Методичні вказівки для практичних занять для здобувачів спеціальності 121 Інженерія програмного забезпечення ОП «Інженерія програмного забезпечення» за другим (магістерським) рівнем вищої освіти денної та заочної форм навчання / уклад О. М. Суринович. Луцьк : ЛНТУ, 2022. 64 с.

РОЗДІЛ 2
МЕТОДИ ТА ЗАСОБИ РОЗРОБКИ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

2.1 Вступ до JetPack Compose

Що таке JetPack Compose?

Jetpack Compose – це сучасний набір інструментів для створення рідного інтерфейсу Android. Jetpack Compose спрощує та прискорює розробку інтерфейсу користувача на Android за допомогою меншого коду, потужних інструментів та інтуїтивно зрозумілих API Kotlin.

Jetpack Compose – це повністю ***декларативне програмування***, що означає, що ви можете описати свій інтерфейс користувача, викликавши набір «composable». Протягом останніх десяти років ми використовували традиційний спосіб ***імперативного дизайну інтерфейсу користувача.***

Зразок коду та результат показані на рисунку 2.1.

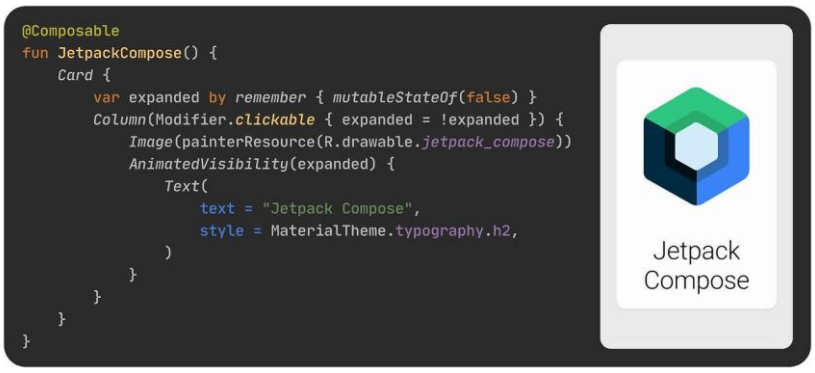

Рисунок 2.1 – Зразок коду «composable» та результат

Імперативний UI. Це найпоширеніша парадигма. Вона передбачає наявність окремого прототипу/моделі інтерфейсу користувача програми. Цей дизайн фокусується на тому, «як», а не на тому, «що». Хорошим прикладом є XML-макети в Android. Ми розробляємо віджети та компоненти, які потім відображаються для перегляду та взаємодії користувача.

Імперативний приклад коду інтерфейсу користувача

Xml:

```
<TextView
     android:id="@+id/tv_name"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
/>
```

Java:

```
public class MainActivity extends AppCompatActivity {

    TextView tvName;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_checkbox);
        tvName = findViewById(R.id.tv_name);
        tvName.setText("Hello World");
    }
}
```

Декларативний користувальницький інтерфейс. Цей шаблон є новою тенденцією, яка дозволяє розробникам проектувати інтерфейс користувача на основі отриманих даних. Він, з іншого боку, фокусується на тому, «що». Ця парадигма дизайну використовує одну мову програмування для створення цілої програми.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    setContent {
        Text(text = "Hello World")
    }
  }
}
```

У JetPack Compose немає потреби в жодному коді xml. Ми можемо створювати перегляди за допомогою «composable».

Переваги JetPack Compose:

- швидко працює та забезпечує плавну роботу;
- швидке навчання;
- можлива взаємодія з імперативним підходом;
- пропонує кращий спосіб реалізації принципів слабкого зв’язку;
- він на 100% створений у Kotlin, що робить його сучасним підходом до розробки Android.

Компонована функція. Це те саме, що й будь-яка інша функція в програмуванні. Але нам потрібно використовувати анотацію @Composable.

Синтаксис:

```
@Composable fun MethodName (parameter: String) {
    //ваш контент
}
```

Приклад:

```
@Composable fun Greeting (name: String) {
   Text ( text = "Hello $ name !" )
}
```

У цьому прикладі ми створили нову компоновану функцію – **Greeting().** Ми використовуємо Text() як вміст. Це вбудована компонована функція. Ми можемо додавати компоновані функції до іншої компонованої функції.

Після створення комбінованої функції ми можемо використовувати її як вміст.

Приклад програми Hello world за допомогою Jetpack Compose:

```
class MainActivity : ComponentActivity() {
     override fun onCreate (savedInstanceState: Bundle?) {
         super .onCreate(savedInstanceState)
        setContent { Greeting ( "World" )
        }
   }
}
@Composable fun Greeting (name: String) {
   Text ( text = "Hello $ name !" )
}
```

Також вам потрібно додати деякі залежності для Jetpack Compose.

```
//compose_version = '1.0.2'
implementation 'androidx.activity:activity-compose:1.3.0-alpha06'
implementation "androidx.compose.ui:ui:$compose_version"
implementation
"androidx.compose.material:material:$compose_version"
implementation "androidx.compose.ui:ui-tooling:$compose_version"
```

Зверніться до офіційного сайту розробників Android, щоб отримати деякі основні відомості про Jetpack Compose:
https://developer.android.com/jetpack/compose/tutorial

2.2 Попередній перегляд JetPack Compose

У Jetpack Compose ми можемо побачити попередній перегляд нашого коду в Android Studio. Це дозволяє нам ***бачити результат, не запускаючи нашу програму.***

Клацніть **split**, щоб переглянути код і попередній перегляд (рис. 2.2).

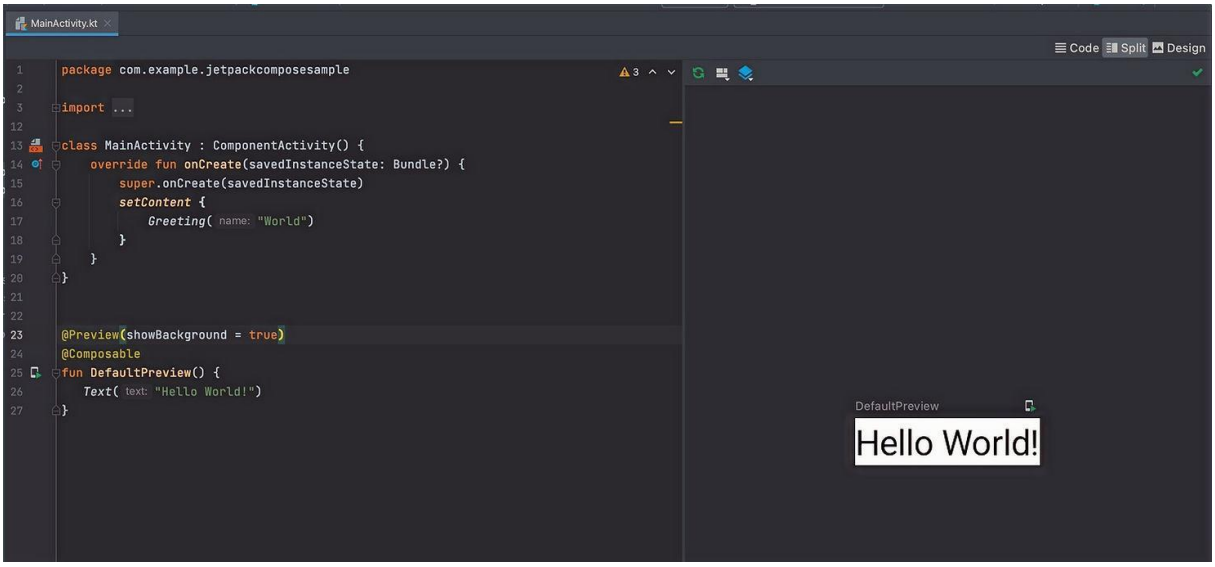

Рисунок 2.2 – Попередній перегляд

Як додати попередній перегляд?

Вам потрібно додати анотацію ***@Preview()*** перед функцією composable. Після додавання цієї анотації ми можемо побачити попередній перегляд нашого інтерфейсу користувача.

```
@Preview () @Composable fun DefaultPreview() {
  Text ( "Hello World!" )
}
```

Які налаштування доступні в @Preview? Коли ми аналізуємо клас @Preview, він має багато цікавих функцій. Клацніть «command (control) + клавіша миші» на анотацію @Preview, щоб побачити такі параметри:

```
annotation class Preview(
    val name: String = "",
    val group: String = "",
    @IntRange(from = 1) val apiLevel: Int = -1,
    // TODO(mount): Make this Dp when they are inline classes
    val widthDp: Int = -1,
    // TODO(mount): Make this Dp when they are inline classes
```

```
        val heightDp: Int = -1,
        val locale: String = "",
        @FloatRange(from = 0.01) val fontScale: Float = 1f,
        val showSystemUi: Boolean = false,
        val showBackground: Boolean = false,
        val backgroundColor: Long = 0,
        @UiMode val uiMode: Int = 0,
        @Device val device: String = Devices.DEFAULT
    )
```

Як налаштувати?

Ви можете налаштувати попередній перегляд двома способам: вручну або за допомогою інструмента редагування попереднього перегляду.

Кроки для ввімкнення інструмента попереднього перегляду

Інструмент попереднього перегляду недоступний за замовчуванням. Якщо ви хочете ввімкнути його, виконайте наступні кроки:

1. Натисніть **Android Studio** у верхньому меню.
2. Перейдіть до **налаштувань (Settings)** (користувачі Mac натискають -> **Preferences**).
3. Натисніть **Experimental** (рис. 2.3).
4. Увімкніть останній прапорець -> **Enable @Preview picker.**
5. Нарешті натисніть «**Apply**» та «**ОК**».

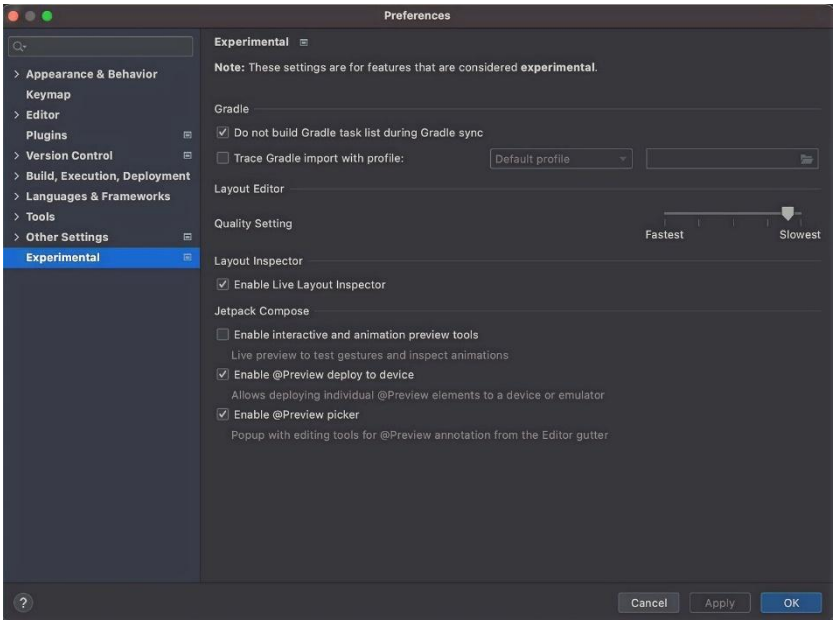

Рисунок 2.3 – Увімкнення інструмента попереднього перегляду

Увімкнувши цей інструмент, ми зможемо побачити піктограму налаштувань у @Preview (рис. 2.4).

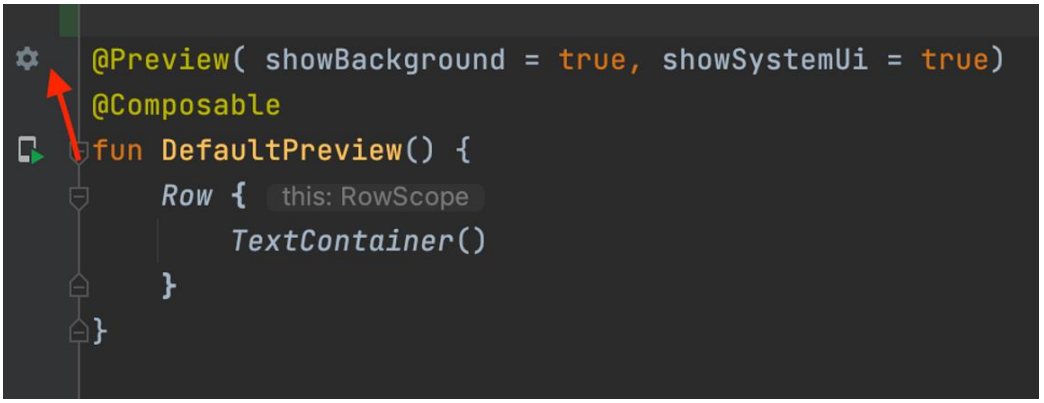

Рисунок 2.4 – Піктограма налаштувань у @Preview

Клацнувши цей значок налаштувань, ви можете легко налаштувати попередній перегляд (рис. 2.5).

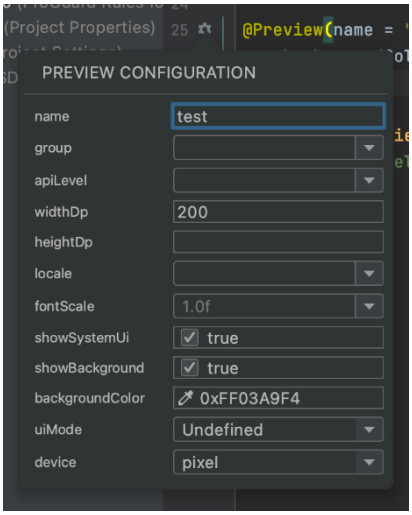

Рисунок 2.5 – Налаштування попереднього перегляду через піктограму

Ім’я

```
@Preview (name = "Preview1" )
```

Якщо ви вкажете аргумент імені, у вашій області попереднього перегляду відобразиться задане ім’я («Preview1»). Ми можемо додати декілька попередніх переглядів до одного класу, тож якщо ви вкажете ім’я, ви зможете легко ідентифікувати його.

Фон. Компонована функція не визначає жодного фону сама по собі, але ви можете додати її до попереднього перегляду, встановивши

showBackground = true в анотації Preview. Існує також аргумент **backgroundColor** для зміни кольору.

```
    @Preview (name = "Preview1" , showBackground = true,
backgroundColor = 0xFF03A9F4 )
```

Висота і ширина. Компонована функція встановлює ширину та висоту за замовчуванням у ***wrap_content.*** Якщо ви хочете змінити, передайте потрібні значення **widthDp** і **heightDp**.

```
    @Preview (name = "Preview1" , showBackground = true, widthDp =
200 , backgroundColor = 0xFF03A9F4 , heightDp = 50 )
     @Composable fun DefaultPreview() {
     Text ( text = "Hello world" )
    }
```

Попередній перегляд показано на рисунку 2.6.

Рисунок 2.6 – Попередній перегляд з налаштованими розмірами та фоном

SystemUI та пристрій. Ви можете попередньо переглянути компоновану функцію на фіктивному екрані (з рядком стану, панеллю інструментів і навігаційним меню). Просто встановіть **showSystemUi = true**, і все готово. Ви також можете змінити використовуваний фрейм пристрою – наприклад, **device = Devices.PIXEL.**

```
    @Preview (name = "Preview1", device = Devices. PIXEL, showSystemUi
= true )
     @Composable fun DefaultPreview() {
      Text ( text = "Hello world" )
    }
```

Попередній перегляд з фреймом пристрою показано на рисунку 2.7.

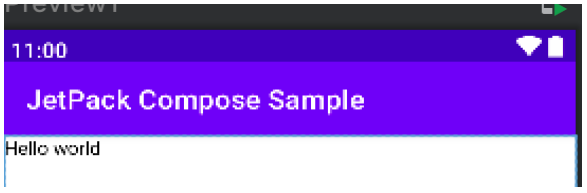

Рисунок 2.7 – Попередній перегляд з системним інтерфейсом та фреймом пристрою

Кілька попередніх переглядів. Ви можете додати кілька попередніх переглядів. Просто додайте анотацію @Preview у ваш composable.

```
      @Preview   (name   =   "Preview1"   ,   showBackground   =   true,
backgroundColor = 0xFF2196F3 )
       @Composable fun Preview1() {
        Text ( text = "Hello world 1" )
      }

      @Preview   (name   =   "Preview2"   ,   showBackground   =   true,
backgroundColor = 0xFF2196F3 )
       @Composable fun Preview2() {
           Text ( text = "Hello world 2" )
      }
```

Офіційний підручник:

https://developer.android.com/jetpack/compose/tooling?hl=it&skip_cache=false

2.3 Створення макету: рядок і стовпець

Що таке макет в Android?

Макет (layout) забезпечує невидимий контейнер для зберігання переглядів або макетів. Ми можемо розмістити групу переглядів/макетів усередині макетів. Рядок і стовпець – це макети для розміщення наших переглядів у лінійному порядку.

Що таке лінійний спосіб?

Лінійний спосіб означає один елемент на рядок. Таким чином розміщуються елементи один до одного в однаковому порядку горизонтально або вертикально (рис. 2.8).

Рядок (Row) – перегляди (**view**) розташовані горизонтально.

Стовпець (Column) – перегляди у ньому розташовані по вертикалі.

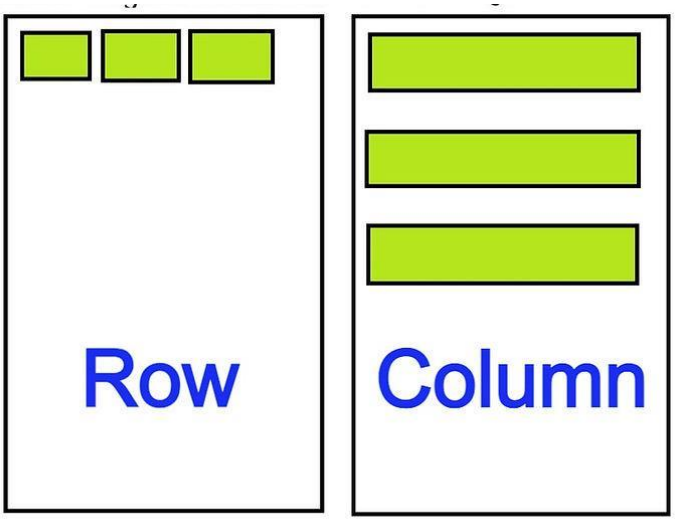

Рисунок 2.8 – Приклад розташування елементів у рядку та стовпці.

Рядок

Рядок покаже кожну дочірню частину поруч із попередньою дочірньою частиною. Це як **LinearLayout** з горизонтальною орієнтацією.

```
@Composable
fun SimpleRow(){
    Row {
        Text(text = "Row Text 1",Modifier.background(Color.Red))
        Text(text = "Row Text 2",Modifier.background(Color.White))
        Text(text = "Row Text 3",Modifier.background(Color.Green))
    }
}
```

Стовпець

Стовпець відображатиме кожну дочірню частину нижче попередніх дочірніх елементів. Це як **LinearLayout** з вертикальною орієнтацією.

```
@Composable
fun SimpleColumn(){
    Column {
        Text(text = "Column Text 1",
Modifier.background(Color.Red))
        Text(text = "Column Text 2",
Modifier.background(Color.White))
        Text(text = "Column Text 3",
Modifier.background(Color.Green))
    }
}
```

Результат для рядка та стовпця показано на рисунку 2.9.

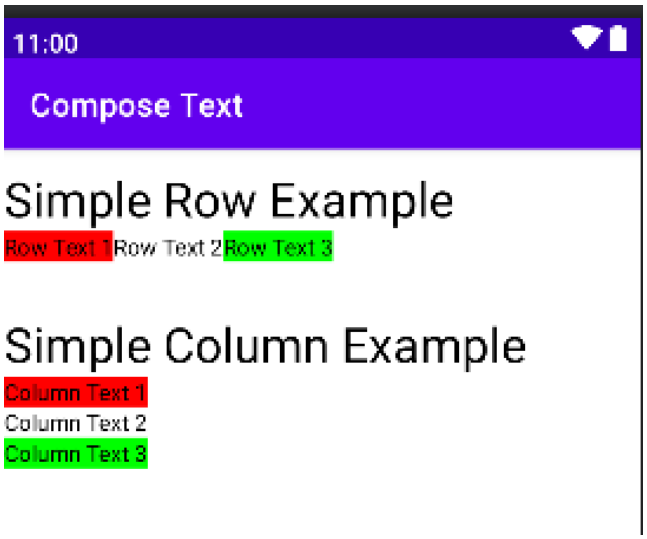

Рисунок 2.9 – Результат використання Row та Column в Jetpack Compose

Вирівнювання

Існує дев'ять варіантів вирівнювання, які можна застосувати до дочірніх елементів інтерфейсу користувача:

TopStart	**TopCenter**	**TopEnd**
CenterStart	**Center**	**CenterEnd**
BottomStart	**BottomCenter**	**BottomEnd**

Розташування

У нас також є три схеми, які можна застосовувати як вертикальні та горизонтальні:

- SpaceEvenly;
- SpaceBetween;
- SpaceAround.

Розташування **SpaceEvenly** розміщує дочірні елементи поперек головної осі, включаючи вільний простір перед першим і після останнього дочірнього елемента (рис. 2.10).

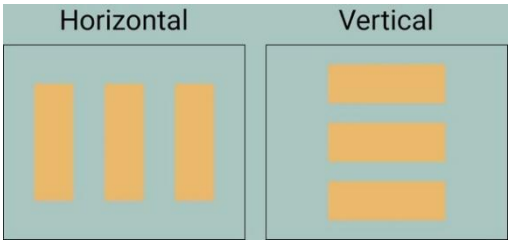

Рисунок 2.10 – Приклад розташування SpaceEvenly в Jetpack Compose

SpaceBetween розміщує дочірні елементи впоперек головної осі без вільного простору перед першим і після останнього дочірнього елемента (рис. 2.11).

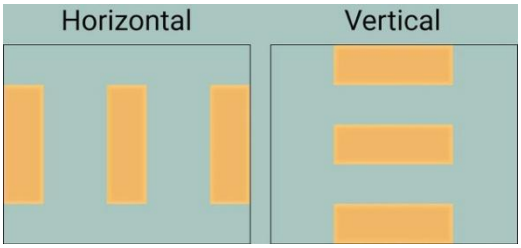

Рисунок 2.11 – Приклад розташування SpaceBetween в Jetpack Compose

Розташування **SpaceAround** розміщує дочірні елементи впоперек головної осі з половиною вільного простору перед першим і після останнього дочірнього елемента (рис. 2.12).

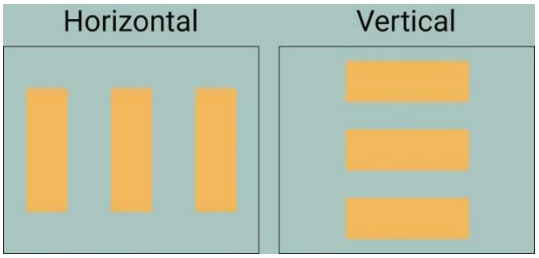

Рисунок 2.12 – Приклад розташування SpaceAround в Jetpack Compose

Розташування та вирівнювання рядків:

```
@Composable
fun RowArrangement(){
    Row(modifier = Modifier.fillMaxWidth(),
        verticalAlignment = Alignment.Top,
        horizontalArrangement  =  Arrangement.SpaceEvenly) {
        Text(text = " Text 1")
        Text(text = " Text 2")
        Text(text = " Text 3")
    }
}
```

Розташування та вирівнювання стовпців:

```
@Composable
fun ColumnArrangement(){
    Column(modifier = Modifier.fillMaxHeight().fillMaxWidth(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment  =  Alignment.End
    ) {
        Text(text = "Text 1",Modifier.background(Color.Red))
        Text(text = "Text 2",Modifier.background(Color.White))
        Text(text = "Text 3",Modifier.background(Color.Green))
    }
}
```

Результат показано на рисунку 2.13.

Рисунок 2.13 – Результат використання вирівнювання та розташування в Row і Column

2.4 Текст у JetPack Compose

Що таке текст?

Якщо ви розробник Android, то це ***TextView***.

Якщо ви новачок у програмуванні Android, це просто ***мітка*** або ***абзац***.

Розмір тексту. Змініть розмір тексту за допомогою параметра ***fontSize***

```
@Composable
fun TextWithSize(label : String, size : TextUnit) {
    Text(label, fontSize = size)
}

//TextWithSize("Big text",40.sp) -- call this method2. Колір тексту
```

Колір тексту. Змініть колір тексту за допомогою параметра ***color:***

```
@Composable
fun ColorText() {
  Text("Color text", color = Color.Blue)
}
```

Жирний текст. Використовуйте параметр ***fontWeight***, щоб зробити текст жирним:

```
@Composable
fun BoldText() {
    Text("Bold text", fontWeight = FontWeight.Bold)
}
```

Текст курсивом. Використовуйте параметр ***fontStyle***, щоб зробити текст курсивом:

```
@Composable
fun ItalicText() {
    Text("Italic Text", fontStyle = FontStyle.Italic)
}
```

Максимальна кількість рядків. Щоб обмежити кількість видимих рядків у компонованому тексті, установіть параметр ***maxLines***:

```
@Composable
fun MaxLines() {
    Text("hello ".repeat(50), maxLines = 2)
}
```

Переповнення тексту. Обмежуючи довгий текст, ви можете вказати переповнення тексту, яке відображається, лише якщо відображуваний текст обрізано. Для цього встановіть параметр ***textOverflow***:

```
@Composable
fun MaxLines() {
    Text("hello ".repeat(50), maxLines = 2)
}
```

Виділення тексту. За замовчуванням компоновані елементи не можна вибрати, що означає, що за замовчуванням користувачі не можуть вибирати та копіювати текст із вашої програми. Щоб увімкнути виділення тексту, вам потрібно обернути текстові елементи компонованим ***SelectionContainer***:

```
@Composable
fun SelectableText() {
    SelectionContainer {
        Text("This text is selectable")
    }
}
```

Результати показані на рисунку 2.14.

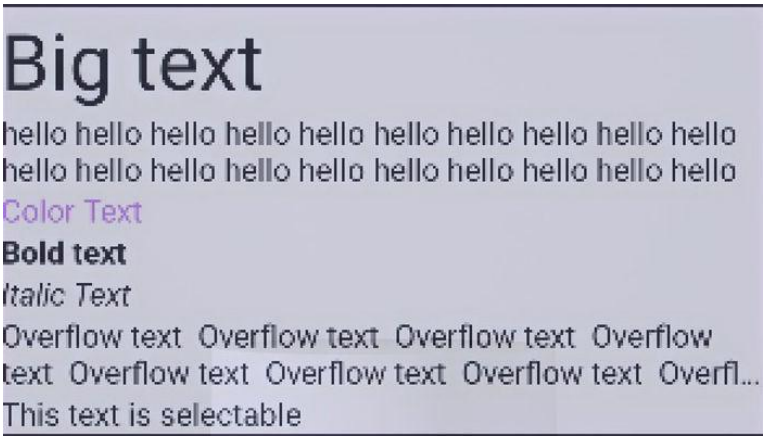

Рисунок 2.14 – Відображення тексту

Для отримання додаткової інформації зверніться до офіційної документації: https://developer.android.com/jetpack/compose/text

2.5 TextStyle у JetPack Compose

Текст відіграє важливу роль у мобільних/веб-додатках. Ми можемо представити деталі користувачеві за допомогою Text. Припустімо, якщо ми використовуємо той самий шрифт, той самий колір, той самий розмір для всього цього посібника, це виглядає погано і незрозуміло для користувача. Отже, нам потрібно стилізувати цей віджет за допомогою опції TextStyle, щоб покращити взаємодію з користувачем.

У Jetpack Compose ми можемо використовувати такі параметри для застосування ***TextStyle()***

```
Text(
    text = "Hello World",
    style = TextStyle(
        color = Color.Red,
        fontSize = 16.sp,
        fontFamily = FontFamily.Monospace,
        fontWeight = FontWeight.W800,
        fontStyle = FontStyle.Italic,
        letterSpacing = 0.5.em,
        background = Color.LightGray,
        textDecoration = TextDecoration.Underline
    )
)
```

Колір тексту (color):

```
Text(
    text = "Text with Color",
    style = TextStyle(color = Color.Red)
)
```

Колір фону (background):

```
Text(
    text = "Text with Background Color",
    style = TextStyle(background = Color.Yellow)
)
```

Тінь (shadow):

```
Text(
    text = "Text with Shadow",
    style = TextStyle(
        shadow = Shadow(
            color = Color.Black,
            offset = Offset(5f, 5f),
            blurRadius = 5f
        )
    )
)
```

Сімейство шрифтів (fontFamily)

Ви можете використовувати наступні системні шрифти або власний шрифт.

```
    val Default: SystemFontFamily = DefaultFontFamily()
    val SansSerif = GenericFontFamily("sans-serif")
    val Serif = GenericFontFamily("serif")
    val Monospace = GenericFontFamily("monospace")
    val Cursive = GenericFontFamily("cursive")
```

Приклад:

```
    Text(
        text = "Text with custom font",
        style = TextStyle(fontSize = 20.sp, fontFamily =
FontFamily.Cursive)
    )
```

Розмір шрифту (fontSize):

```
    Text(
        text = "Text with big font size",
        style = TextStyle(fontSize = 30.sp)
    )
```

Стиль шрифту (fontStyle)

Ви можете використовувати ***FontStyle.Normal*** або ***FontStyle.Italic*****:**

```
    Text(
        text = "Text with Italic text",
        style = TextStyle(fontSize = 20.sp, fontStyle =
FontStyle.Italic)
    )
```

Результат показаний на рисунку 2.15

Рисунок 2.15 – Стилізація тексту

Оформлення тексту: ви можете використовувати ***TextDecoration.Underline*** або ***TextDecoration.LineThrough*** (рис. 2.16). TextDecoration.Underline – малює горизонтальну лінію під текстом. TextDecoration.LineThrough – малює горизонтальну лінію над текстом.

```
@Composable
fun TextDecorationStyle() {
    Column {
        Text(
            text = "Text with Underline",
            style = TextStyle(
                color =  Color.Black, fontSize = 24.sp,
                textDecoration = TextDecoration.Underline
            )
        )
        Text(
            text = "Text with Strike",
            style = TextStyle(
                color =  Color.Blue, fontSize = 24.sp,
                textDecoration = TextDecoration.LineThrough
            )
        )
    }
}
```

Рисунок 2.16 – Оформлення тексту

Типографіка. З ***MaterialTheme*** ми можемо повторно використовувати типову типографіку. Налаштовується ***textstyle()*** із різними розмірами тексту.

У типографіці матеріальної теми доступні параметри, які показані на рисунку 2.17.

Рисунок 2.17 – Параметри тексту з теми MaterialTheme

Зразок коду:

```
@Composable
fun TextHeadingStyle() {
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color.Green)
    ) {
        Text(
            text = "Heading 3",
            style = MaterialTheme.typography.h3
        )
        Text(
            text = "Heading 4",
            style = MaterialTheme.typography.h4
        )
        Text(
            text = "Heading 5",
            style = MaterialTheme.typography.h5
        )
    }
}
```

Результат показаний на рисунку 2.18.

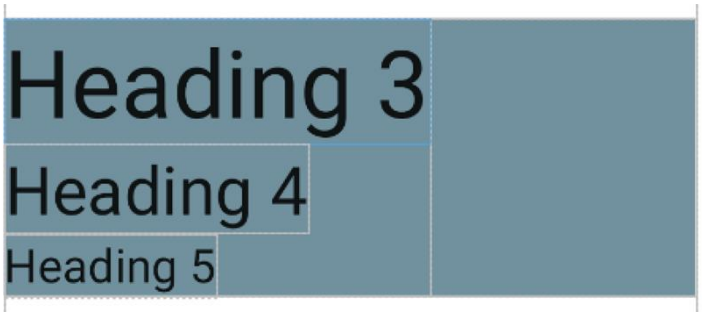

Рисунок 2.18 – Приклад тексту з використанням MaterialTheme

2.6 Модифікатори Jetpack Compose

Що таке модифікатори в Jetpack Compose?

Елементи-модифікатори прикрашають або додають поведінку елементам інтерфейсу користувача Compose. Наприклад, фони, відступи та прослуховувачі подій клацання прикрашають або додають поведінку до рядків, тексту чи кнопок.

1. Ми можемо задати розмір і інтервал за допомогою модифікаторів.
2. Розмістити віджети в межах макета.
3. Прикрасити віджети.

Якщо ви розробник Android, більшість атрибутів xml (id, padding, margin, color, alpha, ratio, elevation...) використовуються за допомогою модифікаторів.

Колір фону (Background color)

```
Text("Text with green background color",
            Modifier.background(color = Color.Green))
```

Підкладка (Padding)

Jetpack compose не має модифікатора поля. Ми повинні використовувати модифікатор ***padding*** як для ***padding***, так і для ***margin*** (рис. 2.19).

```
@Composable
fun TextWidthPadding() {
    Text(
        "Padding and margin!",
        Modifier.padding(32.dp) // Outer padding (margin)
            .background(color = Color.Green) //background color
            .padding(16.dp) // Inner padding
    )
}
```

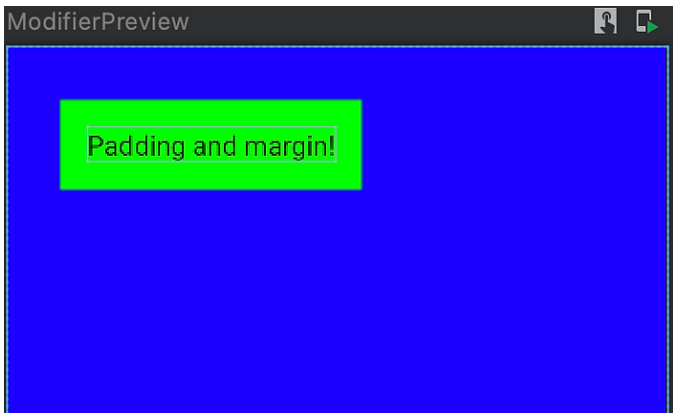

Рисунок 2.19 – Використання модифікатора padding

Ширина і висота (Width і Height)

Для ширини потрібно використовувати ***width(value : Dp)***.

Для висоти потрібно використовувати ***height(value: Dp)*** (рис. 2.20).

```
@Composable
fun WidthAndHeightModifier() {
    Text(
        text = "Width and Height",
        color = Color.White,
        modifier = Modifier
            .background(Color.Blue)
            .width(200.dp)
            .height(300.dp)
    )
}
```

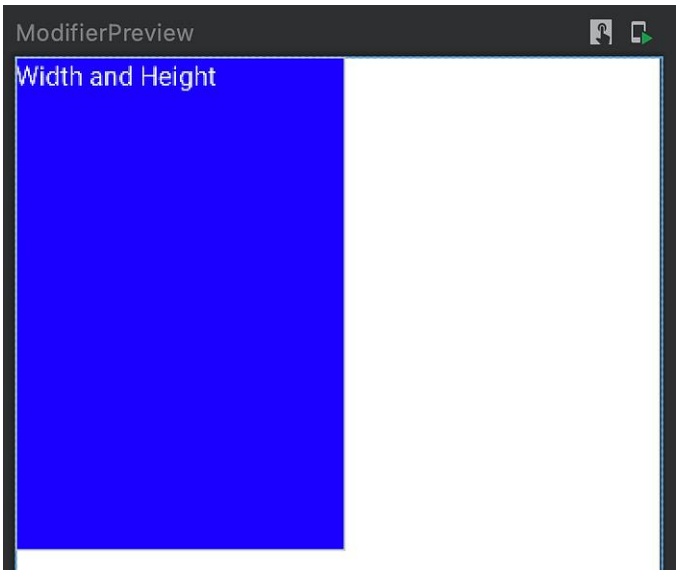

Рисунок 2.20 – Приклад застосування параметрів Width і Height

Розмір (Size)

Якщо вам потрібні і ширина, і висота в одному модифікаторі, використовуйте ***Modifier.size()*** .

Якщо і ширина, і висота однакові, використовуйте цей ***Modifier.size(size: Dp)***. приклад: ***Modifier.size(200.dp)***

Якщо вам потрібна інша ширина та висота, використовуйте ***Modifier.size(width:Dp,height:Dp)***,
наприклад: ***Modifier.size(width=200.dp,height=100.dp)*** (рис. 2.21).

```
@Composable
fun SizeModifier() {
    Text(
        text = "Text with Size",
        color = Color.White,
        modifier = Modifier
            .background(Color.Cyan)
            .size(width = 250.dp, height = 100.dp)
    )
}
```

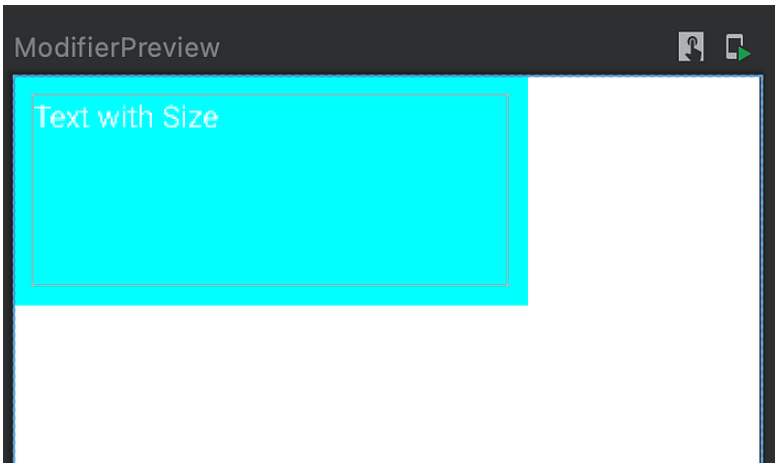

Рисунок 2.21 – Приклад застосування модифікатора Size

Заповнення максимальної ширини

Потрібно передати розмір дробу. Має бути від 0,0 до 1,0.

Якщо вам потрібна ширина як ***match_parent,*** ви можете використовувати 1.0.

Його значення за замовчуванням – 1,0. Якщо викликати метод без розміру дробу, він буде встановлений як 1,0

0,0 означає 0%	0,1 означає 10%	1,0 означає 100%

Якщо в горизонтальній області немає виду: якщо ви вкажете 1.0 -> він займатиме всю ширину (***match_parent)***.

Якщо ви вкажете будь-які інші значення дробу -> воно займатиме значення на основі значення дробу.

Наприклад, якщо ви надасте **0,75** дробу, воно займатиме **75%** ширини екрана.

Якщо деякі перегляди вже існують: якщо ви встановите 1.0 -> воно займе решту простору балансу (заповнить область балансу). Якщо ви вкажете будь-які інші дробові значення -> це заповнить відсоток дробу в просторі балансу (рис. 2.22).

```
Composable
fun FillWidthModifier() {
    Text(
        text = "Text Width Match Parent",
        color = Color.White,
        modifier = Modifier
            .background(Color.Gray)
            .padding(Dp(10f))
            .fillMaxWidth(1f))
}
```

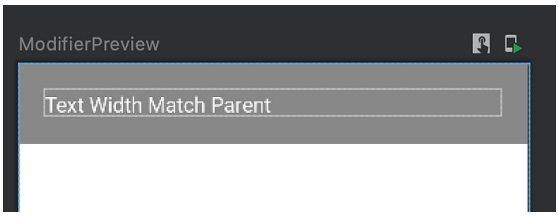

Рисунок 2.22 – Заповнення максимальної ширини

Заповнення максимальної висоти

Потрібно передати розмір дробу. Має бути від 0,0 до 1,0.

Якщо вам потрібна висота як ***match_parent,*** ви можете використати **fillMaxHeight(1.0)**.

Якщо у вертикальній області немає перегляду:якщо ви вкажете 1,0 -> він займе всю висоту ***(match_parent)***.

Якщо ви вкажете будь-які інші значення дробу -> воно займатиме значення на основі значення дробу. Наприклад, якщо ви вкажете **0,75** дробу, він займе **75%** висоти екрана (рис. 2.23). **Якщо деякі перегляди вже існують:** якщо ви встановите 1.0 -> воно займе решту простору балансу (заповнить

область балансу). Якщо ви вкажете будь-які інші значення тертя -> це заповнить відсоток дробу в просторі балансу.

```
@Composable
fun FillHeightModifier() {
    Text(
        text = "Текст з 75% висоти",
        color = Color.White,
        modifier = Modifier
            .background(Color.Green)
            .fillMaxHeight(0.75f) // 75% заповнення висоти
    )
}
```

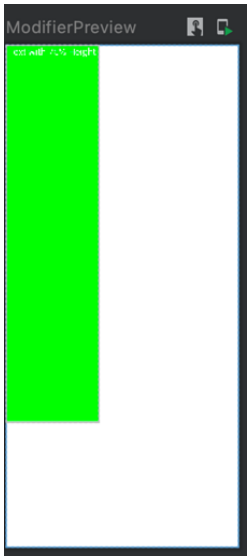

Рисунок 2.23 – Приклад заповнення 75% висоти

Ми також можемо використовувати такі модифікатори:

```
fun MinWidth(value: Dp): LayoutModifier//Minimum width
fun MaxWidth(value: Dp): LayoutModifier//Maximum width
fun MinHeight(value: Dp): LayoutModifier//Minimum height
fun MaxHeight(value: Dp): LayoutModifier//Maximum height
```

Альфа (Opacitiy)

Альфа використовується для встановлення непрозорості перегляду (рис. 2.24).

```
Modifier.alpha(alpha: Float)
```

Ви можете використовувати від 0,1 до 1,0.

0,0 означає 0%	0,1 означає 10%	1,0 означає 100%

```
@Composable
fun AlphaModifier() {
    Box(
        Modifier
            .size(250.dp)
            .alpha(0.5f)//50% прозорості
            .background(Color.Red)
    )
}
```

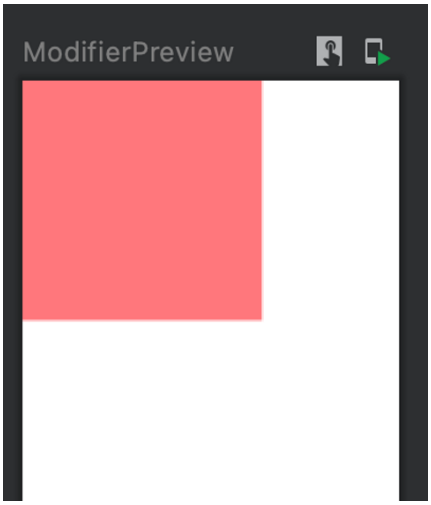

Рисунок 2.24 – Приклад червоного компонента з 50% прозорості

Поворот (Rotate)

Встановлює градус обертання подання навколо центру композиційного об’єкта. *Збільшення* ***значень*** призводить до обертання ***за годинниковою стрілкою***. ***Від’ємні градуси*** використовуються для обертання проти ***годинникової*** стрілки (рис. 2.25).

```
Modifier.rotate(degrees: Float)
@Composable
fun RotateModifier() {
    Box(
        Modifier
            .rotate(45f)
            .size(250.dp)
            .background(Color.Red)
    )
}
```

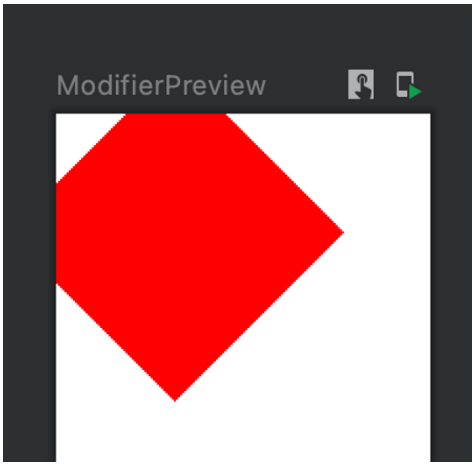

Рисунок 2.25 – Приклад обертання

Масштаб (Scale)

Масштабуйте вміст складеного об’єкта за такими коефіцієнтами масштабу по горизонтальній і вертикальній осі відповідно. Від’ємні масштабні коефіцієнти можна використовувати для відображення вмісту по відповідній горизонтальній або вертикальній осі.

```
@Composable
fun ScaleModifier() {
    Box(
        Modifier
            .scale(scaleX = 2f, scaleY = 3f)
            .size(200.dp, 200.dp)
    )
}
```

Вага (Weight)

За допомогою ваги ви можете вказати співвідношення розмірів між кількома видами.

Наприклад: якщо ви додаєте ***view1*** з вагою **1**, ***view2*** з вагою **1**, ***view3*** з вагою **2**, то він підсумує всі ваги **1** + **1** + **2** = **4** і виділить простір для перегляду на основі заданої ваги (рис. 2.26).

```
View1 отримує 25% місця --> 1 / 4 *100=25%
View1 отримує 25% місця --> 1 / 4 *100=25%
View3 отримує 50% місця --> 2 / 4 *100=50%
```

Перегляньте наступний код і вихід для розуміння.

```
@Composable
fun WeightModifier(){
    Row() {
        Column(
            Modifier.weight(1f).background(Color.Red)){
            Text(text = "Weight = 1", color = Color.White)
        }
        Column(
            Modifier.weight(1f).background(Color.Blue)){
            Text(text = "Weight = 1", color = Color.White)
        }
        Column(
            Modifier.weight(2f).background(Color.Green)
        ) {
            Text(text = "Weight = 2")
        }
    }
}
```

Рисунок 2.26 – Співвідношення розмірів за допомогою параметра Weight

Примітка: вага доступна з версії Compose 1.0.0. Якщо ви використовуєте бета-версію, вона недоступна.

Межа

Встановити межу (рис. 2.27) можна наступними способами:

1. Modifier.border(width: Dp , color: Color , shape: Shape = *RectangleShape*)
2. Modifier.border(width: Dp , brush: Brush , shape: Shape)
3. Modifier.border(border: BorderStroke , shape: Shape = *RectangleShape*)

Modifier.border(width: Dp, color: Color, shape: Shape = *RectangleShape*)

Тут параметр форми необов'язковий. Якщо ви не передасте цей параметр, за замовчуванням буде встановлено прямокутник. У цьому прикладі використовується значення за замовчуванням:

```
@Composable
fun BorderModifier() {
    Text(
        text = "Text with Red Border",
        modifier = Modifier
            .padding(10.dp)
            .background(Color.Yellow)
            .border(2.dp,Color.Red)
            .padding(10.dp)
    )
}
```

Modifier.border(width: Dp , brush: Brush , shape: Shape)

Тут ми встановлюємо межу форми із закругленими кутами.

width – ставимо 2dp

brush – ми використовуємо SolidColor()

RoundedCornerShape() – це вбудована форма в Jetpack Compose. Ми використовуємо цю форму, щоб досягти межі закруглених кутів.

```
@Composable
fun BorderWithShape() {
    Text(
        text = "Text with round border",
        modifier = Modifier
            .padding(10.dp)
            .border(2.dp,                SolidColor(Color.Green),
RoundedCornerShape(20.dp))
            .padding(10.dp)
    )
}
```

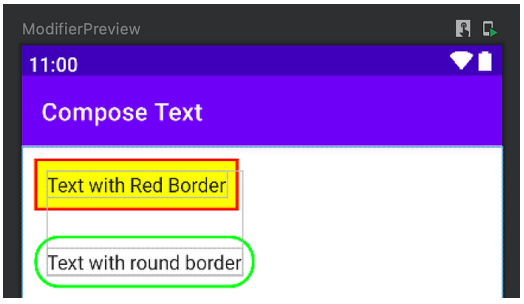

Рисунок 2.27 – Приклади компонентів з межами (border)

Обрізання (Clip)

Модифікатор обрізання дозволяє вирізати наявну форму. Ви можете використовувати форму за замовчуванням або власні форми (рис. 2.28).

Доступні форми в Jetpack Compose:

- RectangleShape;
- CircleShape;
- RoundedCornerShape;
- CutCornerShape.

У цьому прикладі ми використовуємо ***RoundedCornerShape.***

```
@Composable
fun ClipModifier() {
    Text(
        text = "Text with Clipped background",
        color = Color.White,
        modifier = Modifier
            .padding(Dp(10f))
            .clip(RoundedCornerShape(25.dp))
            .background(Color.Blue)
            .padding(Dp(15f))
    )
}
```

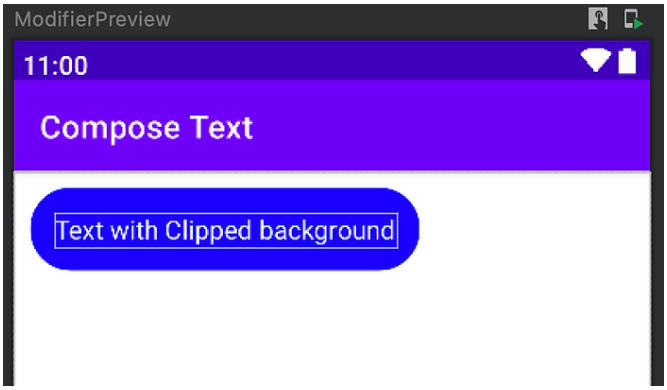

Рисунок 2.28 – Приклад застосування модифікатора обрізання (clip)

Для отримання додаткової інформації зверніться до офіційної документації:

https://developer.android.com/reference/kotlin/androidx/compose/ui/Modifier

2.7 Кнопки в Jetpack Compose

У кнопках Jetpack Compose (рис. 2.29) вам потрібно надати два аргументи для кнопок. Перший аргумент як зворотний виклик onClick, а інший – текстовий елемент кнопки. Ви можете додати Text-Composable або будь-який інший Composable як дочірні елементи кнопки.

Проста кнопка:

```
@Composable
fun SimpleButton() {
    Button(onClick = {
        //your onclick code here
    }) {
        Text(text = "Simple Button")
    }
}
```

Кнопка власного кольору:

```
@Composable
fun ButtonWithColor(){
    Button(onClick = {
        //your onclick code
        },
        colors  =  ButtonDefaults.buttonColors(backgroundColor  =
Color.DarkGray))

    {
     Text(text = "Button with gray background",color = Color.White)
    }
}
```

Кнопка з кількома текстами:

```
@Composable
fun ButtonWithTwoTextView() {
    Button(onClick = {
        //your onclick code here
    }) {
        Text(text = "Click ", color = Color.Magenta)
        Text(text = "Here", color = Color.Green)
    }
}
```

Кнопка зі значком:

```
@Composable
fun ButtonWithIcon() {
    Button(onClick = {}) {
        Image(
painterResource(id = R.drawable.ic_cart),
contentDescription ="Cart button icon",
modifier = Modifier.size(20.dp))
        Text(text = "Add to cart",Modifier.padding(start = 10.dp))
    }
}
```

Кнопка з фігурами

Форма прямокутника:

```
@Composable
fun ButtonWithRectangleShape() {
    Button(onClick = {}, shape = RectangleShape) {
        Text(text = "Rectangle shape")
    }
}
```

Форма круглого кута:

```
@Composable
fun ButtonWithRoundCornerShape() {
    Button(onClick = {}, shape = RoundedCornerShape(20.dp)) {
        Text(text = "Round corner shape")
    }
}
```

Форма зрізаного кута:

```
@Composable
fun ButtonWithCutCornerShape() {
    //CutCornerShape(percent: Int)- it will consider as percentage
    //CutCornerShape(size: Dp)- you can pass Dp also.
    //Here we use Int, so it will take percentage.
    Button(onClick = {}, shape = CutCornerShape(10)) {
        Text(text = "Cut corner shape")
    }
}
```

Кнопка з рамкою:

```
    @Composable
    fun ButtonWithBorder() {
        Button(
            onClick = {
                //your onclick code
            },
            border = BorderStroke(1.dp, Color.Red),
            colors  =  ButtonDefaults.outlinedButtonColors(contentColor
= Color.Red)
        ) {
            Text(text = "Button with border", color = Color.DarkGray)
        }
    }
```

Висота кнопки:

```
    @Composable
    fun ButtonWithElevation() {
        Button(onClick = {
            //your onclick code here
        },elevation =  ButtonDefaults.elevation(
            defaultElevation = 10.dp,
            pressedElevation = 15.dp,
            disabledElevation = 0.dp
        )) {
            Text(text = "Button with elevation")
        }
    }
```

Рисунок 2.29 – Кнопки з різними стилями

Питання для самоконтролю

1. Що таке Jetpack Compose і які його основні переваги?
2. Чим відрізняється імперативний підхід до розробки інтерфейсу користувача від декларативного?
3. Яким чином створюються компоновані функції в Jetpack Compose?
4. Яка роль анотації @Composable у Jetpack Compose?
5. Як створити простий додаток Hello World за допомогою Jetpack Compose?
6. Які залежності потрібно додати для використання Jetpack Compose у вашому проєкті?
7. Яка анотація використовується для створення попереднього перегляду в Jetpack Compose?
8. Як переглянути код і попередній перегляд у Android Studio?
9. Які налаштування доступні в анотації @Preview?
10. Як налаштувати висоту та ширину компонованої функції в попередньому перегляді?
11. Як увімкнути інструмент попереднього перегляду в Android Studio?
12. Як додати фон до попереднього перегляду?
13. Які можливості дає аргумент showSystemUi?
14. Як створити кілька попередніх переглядів для однієї компонованої функції?
15. Що таке текст у Jetpack Compose, якщо ви вже розробник Android?
16. Як можна змінити розмір тексту в Jetpack Compose?
17. Як змінити колір тексту в Jetpack Compose?
18. Як зробити текст жирним у Jetpack Compose?
19. Як зробити текст курсивом у Jetpack Compose?
20. Як обмежити кількість видимих рядків у тексті Jetpack Compose?
21. Як обмежити переповнення тексту в Jetpack Compose?
22. Як зробити текст виділяємим у Jetpack Compose?
23. Які параметри можна використовувати для налаштування стилю тексту у Jetpack Compose?
24. Як змінити колір тексту у Jetpack Compose?
25. Як задати колір фону для тексту у Jetpack Compose?
26. Як додати тінь до тексту у Jetpack Compose?
27. Які системні шрифти можна використовувати у Jetpack Compose?
28. Як задати стиль шрифту у Jetpack Compose?

29. Як використовувати TextDecoration у Jetpack Compose?
30. Що таке типографіка у MaterialTheme у Jetpack Compose?
31. Які параметри можна використовувати для налаштування стилю тексту у Jetpack Compose?
32. Як змінити колір тексту у Jetpack Compose?
33. Як задати колір фону для тексту у Jetpack Compose?
34. Як додати тінь до тексту у Jetpack Compose?
35. Які системні шрифти можна використовувати у Jetpack Compose?
36. Як задати стиль шрифту у Jetpack Compose?
37. Як використовувати TextDecoration у Jetpack Compose?
38. Що таке типографіка у MaterialTheme у Jetpack Compose?

Тестові завдання

1. Що таке Jetpack Compose?
a) Мова програмування для Android
b) Набір інструментів для створення рідного інтерфейсу Android
c) Фреймворк для бекенд-розробки
d) Бібліотека для роботи з базами даних

2. Яка основна парадигма програмування використовується в Jetpack Compose?
a) Імперативне програмування
b) Декларативне програмування
c) Функціональне програмування
d) Об’єктно-орієнтоване програмування

3. Яка анотація використовується для створення компонованих функцій у Jetpack Compose?
a) @Component
b) @Composable
c) @Android
d) @UI

4. Який синтаксис створення компонованої функції?
a)

```
@Composable fun MethodName(parameter: String) {
    // ваш контент
}
```

b)

```
@Component fun MethodName(parameter: String) {
    // ваш контент
}
```

c)

```
@Android fun MethodName(parameter: String) {
    // ваш контент
}
```

d)

```
@UI fun MethodName(parameter: String) {
    // ваш контент
}
```

5. Який приклад коду є правильним для створення функції Greeting в Jetpack Compose?

a)

```
@Composable fun Greeting(name: String) {
    Text(text = "Hello $ name!")
}
```

b)

```
@Composable fun Greeting(name: String) {
    TextView(text = "Hello $ name!")
}
```

c)

```
@Component fun Greeting(name: String) {
    Text(text = "Hello $ name!")
}
```

d)

```
@UI fun Greeting(name: String) {
    Text(text = "Hello $ name!")
}
```

6. Яка анотація використовується для створення попереднього перегляду в Jetpack Compose?

a) @ComposablePreview

b) @ComponentPreview

c) @Preview

d) @UIPreview

7. Яке значення аргументу showBackground використовується для додавання фону в попередньому перегляді?
a) showBackground = false
b) showBackground = true
c) showBackground = “blue”
d) showBackground = 0xFFFFFF

8. Яке налаштування використовується для зміни кольору фону в попередньому перегляді?
a) backgroundColor
b) background
c) bgColor
d) color

9. Який аргумент у @Preview використовується для налаштування висоти попереднього перегляду?
a) heightPx
b) height
c) heightDp
d) previewHeight

10. Який аргумент у @Preview дозволяє показати SystemUI?
a) systemUi
b) showUi
c) showSystemUi
d) systemUiMode

11. Як можна створити кілька попередніх переглядів для однієї компонованої функції?
a) Додати кілька анотацій @Preview до однієї компонованої функції
b) Використовувати різні назви функцій у кожному попередньому перегляді
c) Використовувати різні файли для кожного попереднього перегляду
d) Використовувати цикл для створення кількох попередніх переглядів

12. Яке налаштування @Preview використовується для зміни локалі?
a) locale
b) language
c) region
d) lang

13. Як можна додати ім’я до попереднього перегляду?
a) Використовувати аргумент title
b) Використовувати аргумент name
c) Використовувати аргумент previewName
d) Використовувати аргумент label

14. Що таке текст у Jetpack Compose для новачків у програмуванні Android?
a) Компонент для відображення зображень
b) Абзац або мітка
c) Контейнер для розміщення інших компонентів
d) Компонент для відтворення звуків

15. Як змінити розмір тексту в Jetpack Compose?
a) Використовуючи параметр textSize
b) Використовуючи параметр fontSize
c) Використовуючи параметр size
d) Використовуючи параметр textScale

16. Який параметр використовується для створення жирного тексту в Jetpack Compose?
a) textWeight
b) fontBold
c) textStyle
d) fontWeight

17. Який параметр використовується для встановлення переповнення тексту в Jetpack Compose?
a) textLimit
b) textOverflow
c) overflowLimit
d) overflowText

18. Який код використовується для налаштування кольору тексту в Jetpack Compose?

a)

```
Text(text = "Hello World", color = Color.Red)
```

b)

```
Text(text = "Hello World", style = TextStyle(textColor = Color.Red))
```

c)

```
Text(text = "Hello World", style = TextStyle(color = Color.Red))
```

d)

```
Text(text = "Hello World", textColor = Color.Red)
```

19. Як задати колір фону для тексту в Jetpack Compose?

a)

```
Text(text = "Hello World", backgroundColor = Color.Yellow)
```

b)

```
Text(text = "Hello World", style = TextStyle(backgroundColor = Color.Yellow))
```

c)

```
Text(text = "Hello World", style = TextStyle(background = Color.Yellow))
```

d)

```
Text(text = "Hello World", background = Color.Yellow)
```

20. Які з наступних шрифтів є системними шрифтами у Jetpack Compose?

a) DefaultFontFamily
b) Roboto
c) Monospace
d) Helvetica

21. Як змінити розмір шрифту у Jetpack Compose?

a)

```
Text(text = "Hello World", textSize = 30.sp)
```

b)

```
Text(text = "Hello World", style = TextStyle(fontSize = 30.sp))
```

c)

```
Text(text = "Hello World", style = TextStyle(textSize = 30.sp))
```

d)

```
    Text(text = "Hello World", fontSize = 30.sp)
```

22. Який із наведених нижче прикладів є правильним для створення простої кнопки в Jetpack Compose?

a)

```
    Button(onClick = {}, content = "Simple Button")
```

b)

```
    Button(text = "Simple Button", onClick = {})
```

c)

```
    Button(onClick = {}) {
        Text(text = "Simple Button")
    }
```

d)

```
    Button(onClick = "Simple Button") {
        Text(text = {})
    }
```

23. Як змінити колір фону кнопки на сірий?

a)

```
    Button(onClick = {}, backgroundColor = Color.Gray) {
        Text(text = "Button with gray background")
    }
```

b)

```
    Button(onClick = {}, colors = 
ButtonDefaults.buttonColors(backgroundColor = Color.Gray)) {
        Text(text = "Button with gray background", color = Color.White)
    }
```

c)

```
    Button(onClick = {}, backgroundColor = Color.Gray, textColor = 
Color.White) {
        Text(text = "Button with gray background")
    }
```

d)

```
    Button(onClick = {}, colors = 
ButtonDefaults.buttonColors(backgroundColor = Color.Gray), textColor = 
Color.White) {
        Text(text = "Button with gray background")
    }
```

24. Як додати два текстових елементи до кнопки в Jetpack Compose?

a)

```
    Button(onClick = {}, content = {
        Text(text = "Click")
        Text(text = "Here")
    })
```

b)

```
    Button(onClick = {}, content = {
        Text(text = "Click Here")
    })
```

c)

```
    Button(onClick = {}, content = {
        Text(text = "Click Here", color = Color.Green)
    })
```

d)

```
    Button(onClick = {}, content = {
        Text(text = "Click ", color = Color.Magenta)
        Text(text = "Here", color = Color.Green)
    })
```

25. Як додати значок до кнопки в Jetpack Compose?

a)

```
    Button(onClick = {}) {
        Icon(painter = painterResource(id = R.drawable.ic_cart),
contentDescription = "Cart button icon")
        Text(text = "Add to cart")
    }
```

b)

```
    Button(onClick = {}) {
        Image(painter = painterResource(id = R.drawable.ic_cart),
contentDescription = "Cart button icon")
        Text(text = "Add to cart")
    }
```

c)

```
    Button(onClick = {}) {
        Icon(painter = painterResource(id = R.drawable.ic_cart),
contentDescription = "Cart button icon", Modifier.size(20.dp))
        Text(text = "Add to cart", Modifier.padding(start = 10.dp))
    }
```

d)

```
    Button(onClick = {}) {
        Image(painter = painterResource(id = R.drawable.ic_cart),
contentDescription = "Cart button icon", Modifier.size(20.dp))
        Text(text = "Add to cart", Modifier.padding(start = 10.dp))
    }
```

Рекомендована література

1. Android Compose Tutorial. Android Developers. URL: https://developer.android.com/develop/ui/compose/tutorial.

2. JetPack Compose Tutorial. Android JetPack Compose Tutorial. URL: https://www.jetpackcompose.net/.

3. Kotlin Programming Language. Kotlin. URL: https://kotlinlang.org/.

4. Android Developers. Android Mobile App Developer Tools – Android Developers. URL: https://developer.android.com/.

РОЗДІЛ 3
УПРАВЛІННЯ РИЗИКАМИ В ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Процес управління ризиками програмного забезпечення

Управління ризиками є важливою складовою планування проекту. Управління ризиками в програмній інженерії передбачає виявлення та оцінку ймовірності ризиків у порядку їх впливу на проект.

Кожен проект розробки програмного забезпечення містить елементи невизначеності. Рівень ризику, пов'язаного з кожною проектною діяльністю, визначає успіх проекту розробки програмного забезпечення. Недостатньо просто усвідомлювати небезпеку. Щоб досягти успіху, керівництво проектом має визначити, оцінити, визначити пріоритети та керувати всіма основними ризиками. Більшість проектів розробки програмного забезпечення та розробки програмного забезпечення прагнуть бути унікальними, незалежно від того, створюють вони нові функції чи підвищують ефективність.

Ризик впливає на плановану діяльність, мету та результати роботи через невизначеність. Організоване керівництво ризиками часто називається **ризик-менеджментом** [1].

Ризики можуть бути як негативними (погіршують результат), так і позитивними – такі ризики відомі як можливості (сприяють удосконаленню діяльності або покращенню результатів).

Об'єктом ризику вважається економічна система, ефективність і умови її функціонування, які не відомі наперед.

Суб'єкт ризику – особа, зацікавлена в керуванні об'єктом ризику та має відповідну компетенцію.

Джерело ризику – це чинники або явища, які породжують невизначеність результатів або конфліктність.

Управління ризиками, відоме як практика ризик-менеджменту, це процес ухвалення оптимальних рішень і вжиття заходів для забезпечення прийнятного рівня ризику.

Актуальність управління ризиками в розробці програмного забезпечення (ПЗ) визначається декількома факторами:

– складність проектів ПЗ. Сучасні програмні проекти можуть бути дуже складними, з великою кількістю взаємозалежних частин та компонентів. Це

створює багато можливостей для ризиків, таких як затримки, вартість, якість та інші;

– збільшення вимог. Клієнти постійно змінюють свої вимоги та очікування щодо програмного забезпечення, що може призвести до збільшення ризиків зміни обсягу робіт, термінів виконання та інших факторів;

– технологічні виклики. Швидкий розвиток технологій у програмній індустрії може створювати технічні ризики, такі як несумісність, безпека даних, архітектурні складнощі та інші;

– командні динаміки. Управління ризиками також охоплює внутрішні аспекти командної роботи, такі як комунікація, конфлікти, недосвідченість учасників команди та інші фактори, які можуть вплинути на успішність проекту;

– строки та бюджет. Нерідко програмні проекти мають обмежені строки та бюджети, що створює тиск на команду та може призвести до виходу за межі проекту, якщо не здійснювати управління ризиками ефективно [1].

Управління ризиками в розробці програмного забезпечення допомагає компаніям забезпечити успішне завершення проектів, зменшити витрати та час на розробку, покращити якість продукту і задоволеність клієнтів. Тому ця тема залишається важливою і актуальною для будь-якої IT-компанії чи розробника програмного забезпечення.

Управління ризиками – це процес, спрямований на виявлення, оцінку та подальше управління ризиками, що можуть виникнути при реалізації проекту. Враховуючи той факт, що абсолютно безризикових проектів не існує, оскільки завжди існує можливість виникнення подій, які можуть негативно вплинути на нього, управління ризиками стає ключовою складовою управління проектом.

Важливо розуміти, що управління ризиками не передбачає усунення ризику напряму, а скоріше полягає в ідентифікації та оцінці можливих ризиків, а також в розробці та застосуванні стратегій для їх управління. Цей процес спрямований на мінімізацію впливу ризиків на успішність проекту. Вирішальним є здатність вчасно виявляти потенційні ризики, ретельно їх оцінювати та розробляти ефективні заходи для стримування та пом’якшення їхніх наслідків.

По-перше, потрібно визначити та спланувати. Тоді, коли виникне ризик, спираючись на досвід і знання всієї команди потрібно діяти, щоб мінімізувати вплив на проект.

Управління ризиками включає в себе наступні етапи:

– **ідентифікація ризиків** – визначення критеріїв ризику, розробка методів для виявлення ризиків та створення системи класифікації ризиків;

– **аналіз і оцінка ризиків** – складання переліку можливих ризикових ситуацій, оцінка рівня ризику для кожної ситуації та визначення їх пріоритетності;

– **розробка стратегій управління ризиками** – визначення методів управління ризиками, організація процесу управління ризиками та розробка запобіжних заходів;

– **моніторинг та оперативне управління ризиками** – впровадження методів управління ризиками, оцінка ефективності цих заходів та створення системи управління знаннями про ризики [8].

На першому етапі процесу йде ідентифікація ризиків.

Мета цього етапу включає:

– складання повного списку ризиків (якщо ризик не включено в список, його не буде розглянуто пізніше);

– застосування інструментів і технік ідентифікації;

– залучення максимальної кількості персоналу до роботи з ідентифікації (робочі групи);

– включення до списку ризиків навіть у випадках, коли причина їх виникнення не зрозуміла [1].

Врахування контексту найважливіше саме на етапі ідентифікації ризиків, хоча його можна враховувати і на наступних етапах.

Зовнішній контекст – це зовнішнє середовище, де організація планує досягти цілей (зовнішні впливові фактори), які, як правило, залишаються сталими.

Внутрішній контекст – це внутрішнє середовище, де організація планує досягнути цілей (організаційна структура, відповідальності, інформаційні системи, підпорядкованість, взаємовідношення персоналу).

Ідентифікація ризиків – це процес виявлення, усвідомлення та реєстрації ризиків. Її призначення – визначити можливі ситуації, які можуть впливати на досягнення цілей системи або організації [1].

Після ідентифікації ризику організація повинна визначити будь-які наявні засоби контролю, зокрема стосовно конструктивних особливостей, персоналу, процесів і систем.

Процес ідентифікації ризику включає в себе визначення причин і джерел ризику (небезпеки в контексті фізичної шкоди), подій, ситуацій або обставин, які можуть мати матеріальний вплив на досягнення цілей, а також визначення характеру цього впливу.

Методи ідентифікації ризику можуть включати: доказові методи, системні методи групової роботи, методи індуктивного мислення. Для покращення точності і повноти ідентифікації ризику можна використовувати різні додаткові методи, такі як «мозковий штурм» і метод Дельфі. Незалежно від використаних методів під час ідентифікації ризику важливо звертати увагу на людські та організаційні чинники. Тому під час процесу ідентифікації ризику слід враховувати відхилення людських та організаційних чинників від очікуваних станів, а також події, пов'язані з технічними та програмними засобами.

Метод «мозкової атаки» (або «брейнштормінг») – це креативний процес збирання ідей від учасників групи для розв'язання конкретної проблеми або створення нових концепцій, продуктів або послуг. Основна мета – стимулювати продуктивний обмін думками і збільшити кількість можливих рішень.

Процес «мозкової атаки» зазвичай виглядає так:

– формулювання завдання або проблеми. Учасники групи чітко визначають завдання або проблему, що потребує розв'язання або уваги;

– генерація ідей. Учасники вільно висловлюють будь-які ідеї, що приходять їм на думку, без обговорень або критики. Ці ідеї можуть бути записані на дошці або папері;

– стимулювання та комбінування ідей. Учасники можуть взаємодіяти між собою, додавати додаткові ідеї або комбінувати та розширювати існуючі. Ключовою метою є створення атмосфери відкритості та творчості, щоб кожен міг долучитися до процесу;

– аналіз та оцінка ідей. Після генерації достатньої кількості ідей, група може приступити до їх аналізу та оцінки, визначаючи їхню відповідність поставленому завданню або проблемі;

– вибір та реалізація найкращих ідей. Після оцінки ідей група може вибрати ті, які найбільше відповідають їхнім потребам чи критеріям ефективності, та розробити план їх реалізації.

Метод «мозкової атаки» дозволяє залучити різноманітні погляди та досвід учасників групи, стимулюючи творчий процес і підвищуючи шанси на

знаходження інноваційних ідей та рішень. Він є ефективним інструментом для розв'язання складних проблем та стимулювання інноваційного мислення.

Метод Дельфі – це стратегічний процес, який використовується для отримання консенсусу серед групи експертів стосовно певної проблеми або завдання. Цей метод виник у середині 20-го століття в США в рамках проекту з дослідження технологічних тенденцій. Однак він широко застосовується й у інших галузях, включаючи менеджмент, економіку, медицину та інші.

Основна ідея методу полягає в тому, щоб дослідити думки експертів через послідовні раунди опитування з метою досягнення консенсусу або зближення точок зору. Процес може виглядати наступним чином:

– **визначення проблеми або завдання**. Група експертів чітко визначає ту проблему або завдання, для якого потрібно отримати думки і рекомендації;

– **проведення першого раунду опитування**. Кожен експерт заповнює анонімний опитувальник, де вони висловлюють свої думки, оцінки або прогнози щодо проблеми або завдання;

– **аналіз результатів першого раунду**. Модератор (чи команда модераторів) аналізує відповіді, іноді агрегує їх, ідентифікує пункти згоди та розбіжності;

– **проведення наступних раунді**в. Після аналізу результатів першого раунду, модератор може повторити процес, запрошуючи експертів переглянути свої відповіді, виправити їх або надати додаткові коментарі на основі зібраних даних та думок інших учасників;

– **досягнення консенсусу або зближення точок зору.** Цей процес триває доти, доки не буде досягнуто консенсусу або стане очевидним, що певного рівня узгодження досягнуто, і більшість думок у групі співпадають.

Метод Дельфі дозволяє залучити думки експертів незалежно від їх місця розташування та створити консенсус або зближення точок зору, що допомагає приймати більш обґрунтовані рішення в умовах невизначеності або складності проблеми.

Потім приймається рішення щодо необхідності подальшої обробки, наприклад, якщо ступінь тяжкості наслідків висока, а ймовірність виникнення також висока, або якщо існують будь-які ризики, пов'язані з невизначеністю. Також вирішується, чи потрібно додатково обробляти саме цей ризик, чи можливості щодо його поліпшення, враховуючи наше розуміння ситуації та наявні ресурси.

Наступним кроком є **пріоритезація ризиків**.

Мета цього процесу полягає у встановленні пріоритетів для подальшого аналізу шляхом оцінки та комбінування ймовірностей виникнення. Розміщення ризиків у порядку пріоритетів відповідає потенційному ступеню важливості їх наслідків для досягнення цілей проекту.

Визначення пріоритетів ризиків у рамках проекту є критичним аспектом управління ризиками. Один із широко використовуваних методів для цього – показник важливості ризику, відомий як «рівень витрат від ризику» (risk exposure). Цей показник спрощено визначає сукупний вплив ризику на проект, враховуючи як його ймовірність, так і можливі втрати від реалізації цього ризику.

Проте, необхідно розглянути деякі обмеження цього методу. Наприклад, він надає однакові результати незалежно від того, чи маємо ми високий рівень втрат при низькій ймовірності або низький рівень втрат при високій ймовірності. Це може призвести до недооцінки або переоцінки ризиків, що негативно вплине на ефективність стратегій управління ризиками.

Тому, для більш об’єктивної оцінки ризиків та визначення їх пріоритетності, рекомендується використовувати матрицю ризиків. Ця матриця базується на оцінці ймовірності та впливу кожного ризику, що дозволяє отримати більш детальну картину щодо їх важливості. Шляхом систематичного аналізу цих факторів можна визначити, які ризики потребують найбільшої уваги та розробки стратегій управління ними. Такий підхід дозволяє більш точно визначити пріоритетність ризиків та зосередити зусилля на найбільш критичних аспектах управління ризиками.

На основі цього та аналогічного розподілу для загроз проекту від наслідків дії ризику будується матриця ризиків (табл. 3.1).

Таблиця 3.1 – Матриця ризиків

Ймовірність/наслідок	Низький=1	Середній =2	Високий=3
Висока=3	3	6	9
Середня=2	2	4	6
Низька=1	1	2	3

Важливість ризику з таблиці має такі якісні характеристики:

1–2 – низький рівень;

3–5 – середній;

6–9 – високий.

Оцінка ризиків згідно зі стандартами ISO – оцінка ризиків являє собою результат двох процедур – аналізу та вимірювання ризиків.

Аналіз ризиків це систематичне використання інформації про ризик, його вимірювання, оцінка, порівняння з прийнятним ризиком, обґрунтування раціональних заходів захисту. Аналіз ризиків може мати якісний або кількісний підхід (рис. 3.1, табл. 3.2.).

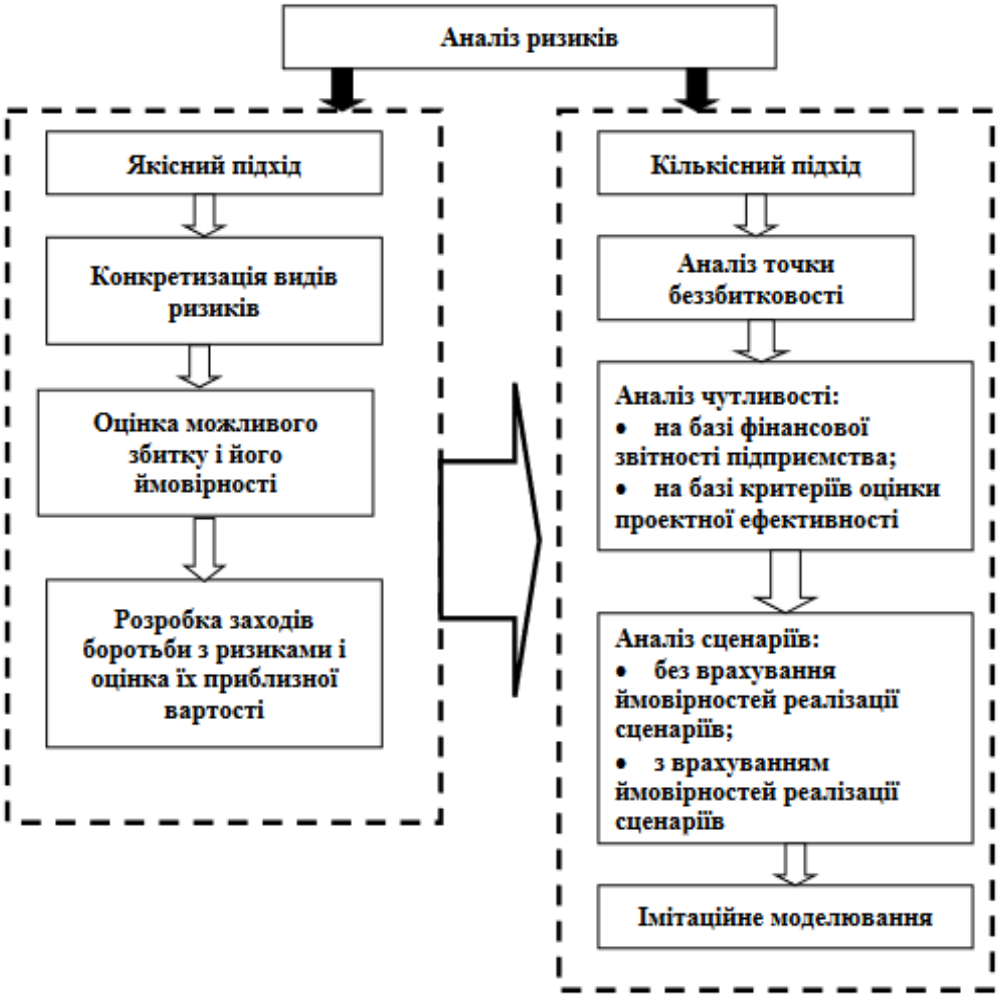

Рисунок 3.1 – Процедура аналізу ризиків [2]

Таблиця 3.2 – Основні якісні та кількісні методи аналізу [2]

Метод	Суть методу	Особливості методу
Якісні методи аналізу ризиків		
Конкретизація видів ризиків	Це означає зібрання та ретельне вивчення інформації щодо майбутнього проекту та характерних йому ризиків.	Потребує інвестицій часу та фінансів для отримання необхідної інформації.
Метод аналогій	Включає в себе порівняння характеристик майбутнього проекту з параметрами раніше реалізованих проектів.	Потребує наявності всіє необхідної інформації для визначення, наскільки цей метод може бути застосований у конкретній ситуації.
Причинно-наслідковий аналіз	Це означає використання евристичного методу для ідентифікації можливих ризикових подій, подальший формальний логічний аналіз їх потенційних причин та розробку заходів для протидії кризовим ситуаціям.	Використовується на етапі перед вкладеннями для пошуку способів підвищення загальної надійності проекту.
Метод «Події-наслідки»	Здійснюється розбиття проекту на компоненти та виявлення ризиків для кожного з них.	Використовується для виявлення специфічних ризиків.
Методи, що поєднують кількісний та якісний аналіз		
Метод експертних оцінок	Головною особою в цьому методі є експерт, який проводить оцінку за допомогою логічних та математично-статистичних методів.	Перевага цього методу полягає в тому, що для оцінки впливу різноманітних факторів може використовуватися досвід експерта, і немає потреби у точних вихідних даних або дороговартісних програмних засобах.
Створення профілю ризиків або карти ризиків	Цей метод включає в себе аналіз ризиків проекту за різними параметрами та їх представлення.	Цей метод служить для візуалізації структури ризиків та оцінки відповідності проекту політиці ризиків у організації.

Апріорний стратегічний аналіз в управлінні ризиками в програмному забезпеченні – це етап, на якому передбачається виникнення ризикових ситуацій та розробляються сценарії їх розвитку.

Цей аналіз також включає оцінку ймовірності виникнення ризиків та серйозності їх наслідків. Він проводиться у відсутність реальної інформації про розвиток ризикової ситуації, тому основними джерелами для прийняття рішень є прогнозна або експертна інформація, а також база знань організації. Однак такі джерела інформації часто можуть надати недостатньо точну або неоднозначну інформацію, що створює додаткові виклики для аналізу та управління ризиками [3].

Оперативно-тактичний аналіз управління ризиками є процесом, що відбувається під час виконання конкретних заходів з управління ризиками. Основна мета цього аналізу – перевірка достатності та ефективності прийнятих заходів, спрямованих на зменшення чи уникнення ризиків. Він включає наступні етапи:

− оцінка відповідності між прогнозними сценаріями розвитку ризикових ситуацій та реальними подіями, що відбуваються. Це дозволяє визначити, наскільки точними є прогнози і наскільки вони відповідають реальній ситуації;

− виявлення нових факторів ризику, які можуть виникнути під час реалізації заходів з управління ризиками. Це допомагає підтримувати актуальність стратегій та приймати додаткові заходи для мінімізації впливу цих нових ризиків;

− оцінка ефективності вжитих заходів, що полягає в аналізі результатів заходів управління ризиками та їх впливу на загальну ситуацію. Це дозволяє визначити, наскільки успішними були прийняті заходи та чи були досягнуті поставлені цілі;

− оперативна оцінка ефективності заходів щодо управління ризиками відбувається у реальному часі відповідно до актуальної ситуації. Це дозволяє оперативно реагувати на зміни та вчасно коригувати стратегії управління ризиками для досягнення кращих результатів [3].

Після виконання **постеріорного системного аналізу** формується основа для створення системи та бази знань з управління ризиками. Цей аналіз ґрунтується на об’єктивній та докладній інформації щодо ризикових ситуацій, які виникають під час діяльності. Під час проведення цього аналізу формується комплексна база даних про ризики, що включає в себе перелік потенційних ризикових сценаріїв, аналіз структури впливових факторів з внутрішнього та

зовнішнього середовища, уявлення про можливі сценарії розвитку ризикових ситуацій, а також статистичні дані щодо імовірності та наслідків таких ситуацій. Отримані дані дозволяють класифікувати ризики за їх імовірністю виникнення та ступенем тяжкості наслідків, від низьких до високих [3].

Ступінь ризику є важливим показником, що використовується для оцінки можливості та статистичної частоти виникнення небажаної події або ситуації, що може негативно вплинути на проект, діяльність або організацію. Цей показник може бути як кількісним, так і якісним, залежно від методології оцінки ризиків і конкретних умов використання (табл. 3.3).

Фізичний зміст показника полягає в оцінці ймовірності виникнення ризику, тобто ймовірності того, що небажана подія або ситуація відбудеться. Ця оцінка може базуватися на різних факторах, таких як історичні дані, експертні оцінки, аналіз схожих ситуацій тощо. Кількісний показник ступеня ризику може виражатися у відсотках, числових значеннях або шкалах від 1 до 5 або від 1 до 10, де більш високі значення вказують на вищу ймовірність виникнення ризику.

Оцінка ступеня ризику допомагає розподілити увагу та ресурси на ті аспекти проекту чи діяльності, де ймовірність виникнення ризику найвища. Це дає можливість приймати обґрунтовані рішення щодо стратегій управління ризиками та розробки планів мінімізації негативних наслідків. Важливою складовою є постійне оновлення оцінки ступеня ризику в процесі реалізації проекту або діяльності, оскільки нові умови та інформація можуть вплинути на його зміну.

Таблиця 3.3 – Шкала для визначення ступеня ризику

Оцінка ймовірності	Величина ймовірності	Індикатори
ВИСОКА (ймовірно)	Ситуації можливо виникнуть протягом року або ймовірність їх настання перевищує 75%.	Існує конкретна та визначена можливість, що ситуація виникне у найближчий час за умови використання існуючих бізнес-процесів.
СЕРЕДНЯ (можливо)	Ймовірність того, що ситуація виникне протягом наступного року, становить від 25% до 75%.	Уникнення ситуацій можливе за умови систематичного й відкритого управлінням ризиками.
НИЗЬКА (мало-ймовірно)	Сприятлива обставина може виникнути з великою затримкою або ймовірність її настання протягом року менше 25%.	При використанні існуючих бізнес-процесів ситуація вважається малоймовірною, або ж може вимагати проведення додаткових досліджень.

Міра ризику, як кількісний показник, оцінює рівень небезпеки, яка пов’язана з можливою реалізацією ризикової ситуації, відображаючи можливі негативні наслідки цього сценарію.

Фізичний зміст цього показника може бути представлений через прямі або непрямі втрати, а також втрачену користь у зв’язку з настанням ризику.

Ціна ризику, також у кількісному виразі, відображає співвідношення між максимально можливим результатом і рівнем ризику.

Фізичне значення цього показника виявляється у відображенні позитивного результату, який може бути отриманий при настанні негативного сценарію ризику.

Принципи оцінки ризику включають наступне:

− величина збитків від різних типів ризику є незалежною одна від одної: якщо один тип ризику реалізується, це не впливає на збитки від інших типів;

− реалізація одного типу ризику не обов’язково збільшує або зменшує ймовірність виникнення іншого, за винятком форс-мажорних обставин;

− максимальні можливі збитки від певного ризику не мають перевищувати фінансові можливості організації чи суб’єкта господарювання. Це ґрунтується на концепції прийнятного ризику, що передбачає здатність суб’єкта господарювання приймати ризики в межах власних фінансових можливостей.

Методи якісного аналізу формально-логічні спрямовані на розкриття сутності та чинників ризиків. Серед них можна виділити морфологічний аналіз, семантичний аналіз, декомпозицію й композицію, методи, що базуються на нечіткій логіці, причинно-наслідковий аналіз, методи сценаріїв, зіставлення, контекстний аналіз, «зворотний інжиніринг» та інші.

Експертні методи, у свою чергу, спрямовані на отримання різноманітних якісних та кількісних експертних оцінок в різних аспектах аналізу ризиків.

Експертні оцінки є найбільш поширеними методами, що застосовуються у процесі якісного аналізу ризиків. Суть їх полягає у зборі відповідної інформації про можливі ризики, які можуть вплинути на діяльність підприємства або конкретний проект, шляхом обробки думок експертів – співробітників, досвідчених підприємців чи інших фахівців.

Особливості експертних оцінок передбачають наявність у експертів відповідної компетентності; вони базуються на більш складних та трудомістких процедурах обробки даних у порівнянні з кількісними оцінками; потребують розробки адекватних рейтингових шкал для систематизації оцінок.

Види методів експертних оцінок різняться в залежності від складу та підготовки експертів. Масові опитування передбачають широке залучення непрофесійних респондентів у проведення аналізу.

Найбільш поширеною формою масових опитувань є анкетування, інтерв'ювання, телефонні опитування, а також явні та неявні статистичні спостереження. Результатом масового опитування може стати як кількісна оцінка ризику, так і якісні показники. У вибіркових дослідженнях ретельно досліджуються індивідуальні думки та мотиви непрофесійних респондентів в послідовному порядку.

До цього спектру методів відносяться такі методи як фокус-групи, протоколи, імітаційні ігри, тренінги та інші; професійні оцінки - це використання професійних експертів для формування конкретних рекомендацій та оцінок.

Серед них найбільш відомі:

- мозковий штурм;
- метод Дельфі;
- порівняння парами;
- метод аналогій.

Визначення переваг – це процес, в якому як професіонали, так і непрофесіонали встановлюють порядок якісних параметрів. До цієї категорії належать ранжування, порівняння парами, «м'яке голосування» та інші.

Кількісний аналіз ризику використовує числові оцінки і показники, які можна обробляти за допомогою основних математичних операцій. Його особливості включають:

- ефективне застосування лише у випадку наявності достовірних даних для обчислень;
- основу для розрахунків становлять репрезентативні дані, такі як хронологічні послідовності, статистичні вибірки тощо;
- вимагає від аналітика розуміння формальних математичних методів та інструментів аналізу [4].

Використання кількісних оцінок зменшує рівень невизначеності, приводячи до більшої чіткості у прийнятті рішень.

Аналіз ризиків може проводитися за допомогою різних кількісних методів, серед яких:

– **економіко-статистичні методи** передбачають структуровану обробку статистичних даних про ризикові ситуації. Ці методи включають стохастичні аналізи, кореляційно-регресійні моделі та інші;

– **розрахунково-аналітичні методи** використовуються для обробки кількісних та фінансових оцінок ризику. Сюди входять фундаментальний аналіз витрат, методи прямого та зворотного розрахунків, аналіз стійкості й чутливості, факторний аналіз, розрахунки граничних значень та інші;

– **нормативні методи** передбачають визначення діапазонів ризиків та гранично допустимих значень ризикових параметрів. Сюди входять методи фінансових коефіцієнтів, критичних значень і лімітів, рейтингів та інші;

– **методи моделювання** використовуються для імітації реальних умов діяльності з метою виявлення вузьких місць, первинних і вторинних центрів ризику, перевірки гіпотез і т.д.

Моделювання може бути втілене у формі макетування, лабораторних випробувань, економіко-математичних та віртуальних моделей.

Опишемо окремі кількісні методи аналізу, зазначивши суть методу та їх особливості (табл. 3.4).

Таблиця 3.4 – Основні кількісні методи аналізу [2]

Метод	Суть методу	Особливості методу
Коригування норми дисконтування	Припускається підвищення процентної ставки дисконтування згідно з усіма ризиками, які впливають на «проект».	Не враховує зміни рівня ризику протягом виконання «проекту».
Метод достовірних еквівалентів	Це передбачає, що грошові потоки будуть експертно коригуватися в залежності від суб'єктивної оцінки рівня ризику, що пов'язаний з цими потоками.	Недолік цього методу полягає у тому, що відсутні обґрунтовані методи розрахунку без ризикових еквівалентів, а також у суб'єктивності експертної оцінки.
Аналіз показників ефективності та динаміки грошового потоку	Цей підхід передбачає оцінку резерву міцності проекту за допомогою відносних показників.	Забезпечує тільки загальну оцінку всіх ризиків проекту.
Аналіз чутливості	Шляхом послідовного внесення окремих змін до техніко-економічних параметрів «проекту» виявляються ризики, які мають найбільший вплив на проект.	Цей метод також дає можливість оцінити міру відхилення параметра, коли «проект» перетворюється на невигідний.

Продовження таблиці 3.4

Метод сценаріїв	Альтернативні сценарії розвитку «проекту» формуються шляхом одночасного внесення змін до різних техніко-економічних параметрів.	Використання цього методу, на відміну від аналізу чутливості, не обмежується кількістю факторів.
Імітаційне моделювання	Цей метод включає в себе створення фінансової моделі та повторний розрахунок різних сценаріїв «проекту», враховуючи взаємозв'язки між його параметрами.	Використання цього методу вимагає спеціалізованого програмного забезпечення і проведення додаткових досліджень, що робить його складним у використанні.

Наступним етапом є розробка заходів щодо управління ризиками.

Під час розробки стратегій управління ризиками активно ведеться керівництво проектом. Для кожного з критичних ризиків необхідно розробити відповідну стратегію. Використовуються три стратегії:

− **перенесення (Transfer).** Відповідальність за наслідки ризику перекладається на третю сторону, таку як замовник, партнер чи страхова компанія. Цей підхід доцільно використовувати, коли ми не маємо можливості вплинути на ризик і є сторони, які можуть взяти на себе цю відповідальність;

− **прийняття (Accept).** Ми беремо на себе відповідальність за наслідки ризику, але нічого не робимо для його запобігання. Цей підхід виправданий лише тоді, коли ми не можемо нічого зробити щодо ризику, і витрати на перенесення відповідальності на треті сторони виявляються нецільово високими;

− **зменшення (Mitigate).** Ми активно боремося з ризиком, беручи відповідальність на себе. Ефективно мати кілька планів дій: основний, щоб запобігти ризику, і запасний, в разі виникнення ризику і його негативного впливу на проект.

Методи реагування на ризик (табл. 3.5): ухилення від ризиків; прийняття ризиків; розподіл та аутсорсинг ризику; попередження ризику; зниження ступеня ризику.

Таблиця 3.5 – Основні напрями боротьби з ризиками реалізації програмних проектів

Напрями боротьби з ризиками	Варіанти заходів	Рівень загрози ризику
Пом’якшення	Зниження ймовірності виникнення потенційних ризиків та (або) масштабів можливих збитків від небажаних подій сприяє зменшенню впливу ризиків на хід виконання етапів програмного проекту, при цьому джерело ризику залишається незмінним.	Виправданий ризик
Прийняття	Підтвердження можливості виникнення небажаних подій, свідоме прийняття рішення про узяття на себе їхніх наслідків та компенсацію збитків із власних коштів. У цьому контексті робляться спроби передчасного виявлення потенційних ризиків і їхнього усунення.	Прийнятний ризик
Ухилення	Анулювання потенційних загроз або джерел ризику шляхом усунення можливості виникнення небажаних подій.	Неприпустимий ризик
Передача	Перехід відповідальності за можливі негативні події на інших учасників проекту, таких як страхова компанія, не втручаючись у джерело ризику.	Виправданий ризик

Ухилення від ризику означає уникнення виконання певних дій або прийняття рішень, які мають високий ризик, включаючи:

− відмову від інвестицій у ризиковані та інноваційні проекти;

− уникальних послуг від ненадійних (сумнівних) партнерів та контрагентів, а також від некомпетентних, ненадійних співробітників;

− уникнення ризикованих рішень;

− непроведення операцій, рівень ризику яких є надто високим.

Прийняття ризику передбачає взяття на себе ризику або його збереження, не приймаючи жодних заходів для захисту від нього. Це використовується, коли ризик знаходиться на прийнятному рівні, а вплив на нього неможливий або економічно неефективний, а відчутні наслідки ризику зводяться до мінімуму.

Незаплановане прийняття ризику виникає у таких ситуаціях:

− джерело потенційної небезпеки не було виявлено;

− третя сторона не виконала свої зобов’язання з компенсації збитків, що виникли внаслідок ризику;

− виявлені збитки перевищують обсяги, передбачені у контракті щодо передачі ризику третій стороні;

− можливість зменшення ризику була проігнорована через низьку ймовірність його реалізації.

Заплановане прийняття ризику передбачає два підходи до покриття можливих втрат: покриття втрат по мірі їх виникнення; резервування певної суми доходу в кожен період.

Розподіл ризику передбачає розподіл відповідальності за ризик між підприємством та третіми особами за умови збереження загального рівня ризику. Способи розподілу ризику включають укладення договорів та використання диверсифікації.

Аутсорсинг ризику передбачає передачу відповідальності за зниження можливості виникнення небажаних подій на сторонню організацію (інший суб’єкт), яка зазвичай здійснюється за допомогою договору.

Попередження ризику передбачає проведення превентивних заходів, спрямованих на зниження ймовірності виникнення негативних подій.

Зниження ризику (локалізація) передбачає проведення превентивних заходів для зменшення розміру потенційних збитків.

Загалом, основними принципами управління ризиками є максимізація (розгляд всіх можливих чинників випадковості та невизначеності), мінімізація (зменшення переліку можливих ризиків і їх впливу), адекватна реакція (швидке реагування на зміни вимог), а також прийнятність ризику (знаходження компромісу між безпекою та можливостями).

Планування зменшення ризиків включає ряд дій, які дозволяють визначити, оцінити та вибрати можливості збереження ризиків на прийнятному рівні, враховуючи обмеження та цілі проекту.

У процесі планування зменшення ризиків необхідно розглянути типи та конкретні характеристики зменшення для кожного ризику, включити виявлені можливості зменшення ризиків у загальний план або створити окремий план для кожного конкретного ризику. Такий план, як правило, включає інформацію про назву ризику, дату в плані, відповідальну особу, короткий опис ризику, причину виникнення, можливі шляхи зменшення ризику, конкретні події та дії для мінімізації ризику, критерії успіху для кожної події, статус ризику,

альтернативні шляхи виконання проекту в разі виникнення проблем, рекомендації з управління та необхідні ресурси.

Зниження ризиків потрібно здійснювати на відповідному рівні структури проекту, щоб уникнути їх поширення на більш високий рівень.

Під час етапу впровадження заходів зменшення ризиків проводяться дії, які можуть вплинути на ймовірність виникнення ризику та його наслідки. На цьому етапі визначаються зміни в плані, бюджеті, вимогах та контрактах, встановлюється взаємозв'язок між керівництвом проекту, замовником та виконавцями, визначаються конкретні кроки для зниження ризику, встановлюється вплив зменшення ризику на графік виконання робіт та ретельно документуються всі зміни.

Важливо, щоб дії з зменшення ризиків були зрозумілими та прийнятними для всіх сторін: замовника, керівництва проекту і виконавців. Застосування методів зниження ризиків дозволяє ефективно оцінювати та збільшувати рівень прибутку, але це потребує комплексної оцінки та достовірних прогнозів.

Важливо зауважити, що всі наведені методи застосовуються лише у випадку, якщо ризик вже виник або має велику ймовірність виникнення. Такий підхід не завжди відповідає принципу проактивності – необхідно передбачати витрати на недопущення виникнення ризику, а не на його усунення. При такому підході запобіжні та адекватні заходи дозволяють зменшити витрати та вибрати оптимальний рівень ризику з мінімальними витратами (табл.3.6).

Таблиця 3.6 – Основні недоліки зниження ризиків

Методи зниження ризиків	Недоліки
Розподіл ризику	Можливе зниження рівня доходу
Резервування коштів	Низьке покриття збитків
Страхування ризику	Витрачання коштів, обмеженість застосування
Використання методу приватних ризиків	Обмеження застосування даного методу
Хеджування	Тільки цінові ризики
Лімітування	Зниження доходу
Диверсифікація	Зниження доходу
Уникнення	Втрата доходу

Таблиця 3.6 пропонує огляд основних методів зниження ризиків та їх недоліків. Отже:

− розподіл ризику. Цей метод передбачає розподіл ризику між різними сторонами. Недоліком є можливе зниження рівня доходу, оскільки частина прибутку може бути розподілена серед учасників;

− резервування коштів. За цим методом створюється фінансовий резерв для покриття можливих збитків. Недолік полягає в тому, що резерв може бути недостатнім для повного покриття збитків у випадку серйозних ризиків;

− страхування ризику. Цей метод передбачає використання страхових полісів для покриття можливих збитків. Однак, недоліком є витрачання коштів на страхові внески та обмеженість обсягу покриття страховки;

− використання методу приватних ризиків. Цей метод передбачає використання фінансових інструментів для управління ризиками. Недоліком є обмеження застосування даного методу до певних видів ризиків;

− хеджування. Цей метод спрямований на захист від цінових ризиків. Недолік полягає у тому, що хеджування не може захистити від інших видів ризиків, окрім цінових;

− лімітування. Цей метод передбачає обмеження ризику за допомогою різних стратегій. Недоліком є можливе зниження потенційного доходу у зусиллях зменшити ризик;

− диверсифікація. Цей метод полягає в розподілі інвестицій між різними активами для зменшення загального ризику. Недолік полягає в тому, що диверсифікація може привести до зниження потенційного доходу;

− уникнення. Цей метод передбачає уникнення ризикованих ситуацій. Недолік полягає в можливій втраті доходу, оскільки уникнення ризику може призвести до відмови від потенційно прибуткових можливостей [2].

Моніторинг ризиків у процесі розробки програмного забезпечення є складним та многогранним завданням, спрямованим на постійне оновлення переліку ідентифікованих ризиків, що виникають у процесі розробки ПЗ, а також на збереження актуального плану виконання програмного проекту, який враховує потенційні ризикові ситуації.

Моніторинг – це складний та многозначний процес, який може бути використаний для різних цілей і в різних масштабах, проте він має спільні характеристики. В сучасних наукових джерелах найпоширеніше таке визначення цього терміну: «Моніторинг – це постійне спостереження за будь-яким процесом з метою виявлення його відповідності бажаному результату або первісному припущенню – спостереження, оцінювання та прогнозування стану довкілля в контексті людської діяльності».

Таким чином, моніторинг ризиків при розробці програмного забезпечення представляє собою інформаційну систему, яка впливає на два аспекти:

− з точки зору процесу реалізації, цей моніторинг створює умови для прийняття ефективних управлінських рішень щодо впровадження заходів для зменшення та усунення різних ризикових подій;

− з точки зору результативності, він використовується для оцінювання фактичного стану етапів реалізації програмного проекту, регулювання їхнього стану в залежності від наслідків настання різних ризикових подій та прогнозування подальших змін.

Мета моніторингу ризиків розроблення програмного забезпечення полягає у встановленні прогнозованих значень ймовірності виникнення ризикових подій за допомогою інформаційно-аналітичного забезпечення. Це дозволяє учасникам процесу управління ризиками приймати адекватні керівні рішення щодо впровадження заходів для зменшення або усунення ризиків в процесі розробки програмного забезпечення.

Моніторинг ризиків у розробленні програмного забезпечення – це багатогранний процес, що включає в себе не лише виявлення потенційних небезпек, але й уважне вивчення усіх аспектів діяльності, пов’язаних з проектом програмного забезпечення. Ось кілька ключових аспектів, які включаються в об’єкт моніторингу ризиків у розробленні програмного забезпечення:

− слідкування за зовнішніми та внутрішніми факторами. Моніторинг ризиків охоплює постійне спостереження за змінами в зовнішньому середовищі, такими як технологічні та ринкові тенденції, законодавчі зміни, а також внутрішніми процесами, такими як зміни в управлінні, командній роботі та ресурсах;

− вивчення змін у моделі діяльності проектного менеджменту. Це включає аналіз та оцінку ефективності використання методів та інструментів управління проектами на різних етапах розроблення програмного забезпечення. Це допомагає виявити можливі слабкі місця та вдосконалити стратегії управління ризиками;

− встановлення фактичної позиції ІТ-компанії на ринку ПЗ. Це включає аналіз конкурентної ситуації, оцінку ринкової долі та рівня конкурентоспроможності компанії. Ця інформація допомагає приймати

обґрунтовані рішення щодо стратегій управління ризиками та розробки програм для попередження або усунення ризиків у розробленні ПЗ.

Цей процес моніторингу вимагає систематичного аналізу, оновлення та адаптації під час усього проекту розроблення ПЗ для забезпечення ефективного управління ризиками та досягнення успішних результатів.

Дієвість цих заходів можна визначити в абсолютних величинах, зазвичай у грошовому еквіваленті, або відносно за відношенням до використаних методів і засобів, а також професіоналізму команди розробників ПЗ.

Фактична позиція ІТ-компанії на ринку програмного забезпечення може бути класифікована як одна з трьох типових:

− стратегічна позиція виникає в результаті перетворення короткострокових окремих заходів на довгострокову систему заходів щодо запобігання та усунення ризиків розробки програмного забезпечення;

− тактична позиція виникає після залучення та витрачання коштів на систему заходів щодо запобігання та усунення ризиків розробки програмного забезпечення у формі постійних відрахувань від кожного програмного проекту;

− бізнес-позиція виникає внаслідок впровадження конкурентних стратегій або ефективних операційних дій на ринку програмного забезпечення за допомогою сучасних високоефективних інструментів або технологій розробки програмного забезпечення.

Ефективність цих заходів можна виміряти в абсолютних величинах, часто в грошовому еквіваленті, або порівнювати з використаними методами та ресурсами, а також враховувати професійність команди розробників програмного забезпечення (табл. 3.7).

Таблиця 3.7 – Загальна система ризик-менеджменту в ІТ-компанії

Суб’єкт управління	Функції в системі моніторингу ризиків розроблення ПЗ
Спостережна рада	Встановлення ключових стратегічних напрямів та цілей управління ризиками в процесі розробки програмного забезпечення, які потребують систематичного моніторингу.
Керівництво ІТ-компанії	Забезпечення стратегічного моніторингу ризиків розроблення ПЗ в ІТ-компанії.
Комітет управління системо заходів із запобігання та знешкодження ризиків	Спостереження за впровадженням стратегії ІТ-компанії, аналіз динаміки основних та показників, розгляд і оцінка відповідних стандартів та обмежень, оцінка позиції керівництва стосовно прийнятних ризиків у розробці програмного забезпечення, вибір заходів щодо зниження та усунення ризиків, а також підготовка висновків для керівництва та видання директив та наказів підрозділам – це всі етапи моніторингу, що здійснюються в ІТ-компанії.

Продовження таблиці 3.7

Експертна рада IT -компанії	Контроль над ризиками у розробці програмного забезпечення, забезпечення відповідності лімітам на ресурси, включаючи людські, встановлення базових рівнів для ризикових подій у децентралізованих структурах, укладання короткотермінових угод з підрядниками та взаємодія з постачальниками, оцінка ринку існуючого програмного забезпечення, а також фінансове проектування для клієнтів преміум-класу.
Підрозділ ризик-менеджменту	Аналіз та оцінка різноманітних ризикових ситуацій, розробка ефективних методів запобігання та нейтралізації ризиків у розробці програмного забезпечення, визначення та оцінка можливих порушень рівня виникнення ризикових подій, надання рекомендацій керівництву IT-компанії щодо стратегій роботи з ризиками та наслідками їх реалізації.
Фінансово-аналітична служба	Збір, реєстрація, обробка та систематизація початкової інформації щодо наслідків виникнення ризикових ситуацій, аналіз наявних збитків та прогнозування можливих втрат, створення відповідних звітів для керівництва IT-компанії та її інвесторів.
Інформаційний відділ	Надання інформаційної підтримки для управління ризиками у процесі розроблення програмного забезпечення, а також створення та супровід системи заходів щодо запобігання та ліквідації ризиків у цьому процесі.
Служба внутрішнього контролю	Оцінка якості та результативності полягає у перевірці: ефективності впроваджених заходів зі зниження та усунення ризиків; ефективності методів та засобів управління ризиками у процесі розробки програмного забезпечення; підтримки та керування системою заходів щодо запобігання та ліквідації ризиків у розробці програмного забезпечення.

Механізм реалізації моніторингу ризиків розроблення ПЗ наведено на рис. 3.2.

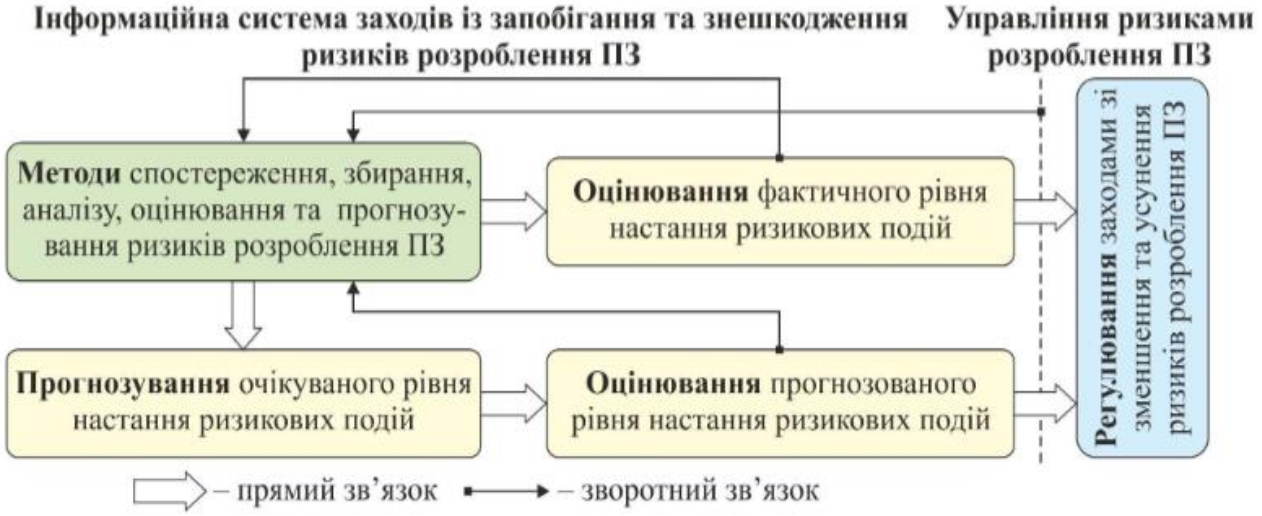

Рисунок 3.2 – Механізм реалізації моніторингу ризиків розробки ПЗ в ІТ-компанії

Важливою складовою механізму моніторингу ризиків розробки ПЗ в ІТ-компанії є підсистема забезпечення, яка включає наступні блоки:

– технічне забезпечення:

1) спеціалізоване ПЗ;
2) засоби передачі даних;
3) засоби захисту інформації;
4) системи управління базами даних;

– інформаційне забезпечення:

1) інформація з зовнішніх джерел, яка включає показники загальноекономічного розвитку країни, розвиток ІТ-компанії та кон'юнктуру ринку ПЗ;

2) внутрішня інформація, що складається з обов'язкової для звітування перед регулювальними органами чи інвесторами, а також управлінської інформації, яка формується компанією для власних потреб;

– законодавче та нормативне забезпечення:

1) зовнішнє – це система законодавчих та нормативних актів, які визначають вимоги до функціонування системи управління ризиками в ІТ-компанії.

2) внутрішнє – це система внутрішньої політики компанії, відповідні положення та процедури, що регламентують процес управління ризиками розробки програмного забезпечення.

Моніторинг і управління ризиками, а також інші процеси ризик-менеджменту, тривають протягом всього життєвого циклу проекту. Додаткові

цілі процесу моніторингу та управління ризиками можуть бути визначені, якщо:

- припущення проекту залишаються дійсними;
- аналіз трендів показав зміни в стані ризику з моменту первісної оцінки;
- правила і процедури управління ризиками виконуються належним чином;
- резерви вартості і розпису оновлюються разом зі змінами ризиків проекту.

Моніторинг і контроль ризиків можуть включати вибір альтернативних стратегій, виконання планів на випадок непередбачених обставин, коригувальні дії та оновлення плану управління проектом. Відповідальний за реагування на ризики повинен регулярно звітувати менеджеру проекту про ефективність виконання плану, про всі непередбачені ефекти та необхідні коригування для належного управління ризиками.

Моніторинг і управління ризиками також передбачають оновлення активів організаційних процесів, включаючи бази даних накопичених знань проекту та шаблони управління ризиками, які знадобляться для майбутніх проектів.

Хоча повністю усунути ризики неможливо, багато з них можна попередити шляхом:

- зменшення ризику;
- мінімізації ризику;
- оптимізації ризиків.

Для цього проектній команді слід виконати такі дії і дати відповідь на такі запитання (табл. 3.8).

Таблиця 3.8 – Дії команди проекту по попередженню ризиків

Дії	Перелік питань до вирішення
(Research) дослідження	Чи маємо ми достатню інформацію про цей конкретний ризик? Чи можливо, нам варто здійснити більший аналіз, щоб отримати додаткові дані і зрозуміти характеристики цього ризику перед прийняттям будь-яких дій?
(Accept) ухвалення	Чи ми здатні витримати наслідки ризиків у випадку їх реалізації? Чи можемо ми прийняти ризики, не приймаючи жодних подальших заходів у зв’язку з цим?
(Avoid) уникнення	Чи можемо ми запобігти ризикам, змінивши наш спосіб дії?

Продовження таблиці 3.8

(Transfer) перенесення	Чи можемо ми передати ризик іншому проекту, групі, організації або приватним особам?
(Mitigation) запобігання	Чи існують можливості вжити попередні заходи для зменшення ймовірності ризику або мінімізації його загрози?
(Contingency) пом'якшення наслідків	Чи можливо зменшити ризикову загрозу шляхом розробки конкретного плану реагування на неї?

Дослідження ризиків дозволяє чітко визначити ризик, його наслідки та ймовірність, а також розробити стратегію його запобігання. На цьому етапі слід створити план управління ризиками на основі оцінки ймовірностей. Необхідно передбачити як обов'язкові заходи, так і заходи на випадок, якщо певний ризик почне негативно впливати (резервний план). Важливо також передбачити часовий і ресурсний резерв з урахуванням впливу ризиків.

Прийняття ризику передбачає взяття на себе відповідальності за можливі негативні наслідки і готовність компенсувати всі можливі збитки власними коштами.

Кр = У/С,

де: Кр – коефіцієнт ризику;

У – максимально можлива сума збитку;

С – обсяг власних ресурсів з урахуванням точно відомих надходжень коштів.

Вибір методу зниження ризику полягає в порівнянні необхідних засобів для мінімізації ризиків із користю від запобігання можливим збиткам. Це співвідношення оцінюється за допомогою коефіцієнта ризику, і оптимальне значення цього коефіцієнта становить 0,3.

Одним зі способів зниження ризику, який стає все більш популярним, є передача управління ризиком третій стороні, яка не прямо залучена до проекту. Це може включати такі заходи, як страхування, найм зовнішніх консультантів з великим досвідом, покупка готових компонентів результату проекту або залучення зовнішніх підрядників.

При виборі конкретного методу зниження ризику власник проекту повинен керуватися такими принципами:

− не перевищувати ризик, який може дозволити власний капітал, враховуючи майбутні вигоди від проекту;

− розглядати наслідки ризику перед прийняттям рішення;

– не ризикувати великими ризиками заради малих вигод.

Моніторинг і контроль ризиків викликають зміни в плані проекту. План проекту повинен бути піддається змінам у процесі виявлення та усунення ризиків.

Для цього на практиці використовуються різні статистичні методи прогнозування ризиків.

Метод Buffer Time 30% полягає у додаванні 30% до загальної тривалості запланованих завдань. Цей додатковий час резервується для вирішення можливих ризиків.

Метод Load Factor (або «фактор завантаження») визначається на основі статистичних даних і залежить від складності та унікальності проекту. Цей коефіцієнт використовується для оцінки, наскільки необхідно збільшити час, відведений на виконання завдання, щоб врахувати можливі ризики.

Схема PERT розрахунку реального терміну:

$$Реальний\ термін = \frac{Оптимістичний\ термін + 4 \cdot Очікуваний\ термін + Песимістичний\ термін}{6}$$

У світовій практиці найчастіше використовується метод Монте-Карло для моделювання ризиків. Системи, що базуються на методі Монте-Карло, відзначаються вищою точністю і дають можливість встановлювати рівень ризику в проекті.

Інформаційні технологічні компанії зазвичай застраховують свої ризики та відповідальність через укладання окремих договорів з клієнтами або за допомогою фінансових гарантій. Наприклад, вони можуть використовувати «Угоду про рівень послуг» (Service Level Agreement, SLA), що широко застосовується в проектах з підтримки та супроводу програмного забезпечення, а також в роботах, пов’язаних з аутсорсингом бізнес-процесів. Підхід SLA для інформаційно-технологічних послуг передбачає керування нестандартними ситуаціями, проблемами, змінами, релізами та рівнем надання сервісу.

3.2 Класифікація ризиків у проектах із розробки програмного забезпечення

Класифікація ризиків створюється для подальшого розроблення методології їх аналізу та розробки методів управління ризиками. З одного боку,

будь-яка класифікація ризиків підпорядковується загальним формальним правилам, з іншого боку, вона повинна чітко відповідати поставленим цілям. Тому не існує загальноприйнятих класифікацій.

Ризики ідентифікуються, класифікуються та управляються ще до фактичного початку розробки системи. Ці ризики групуються за різними категоріями.

По-перше, це ризики, пов'язані з графіком виконання робіт. Графік проекту може порушитися, якщо завдання та ризики, пов'язані з розкладом випуску, не будуть належним чином враховані. Графічні ризики переважно впливають на хід проекту і, в кінцевому підсумку, на фінансове становище компанії, можуть призвести до невдачі у виконанні проекту.

Розклади часто порушуються з наступних причин:

- неправильна оцінка часу, неадекватне урахування часових рамок для завершення завдань;
- недостатній контроль за ресурсами. Всі ресурси, такі як персонал, системи та навички персоналу, можуть бути неправильно розподілені або не відстежені належним чином;
- необ'єктивна оцінка складних функцій та нечітке визначення часу, потрібного для їх розробки;
- неочікувані збільшення масштабу проекту, що можуть виникнути під час його виконання [4].

Бюджетний ризик:

- неправильна оцінка бюджету;
- перевитрати;
- розширення обсягу проекту.

Ризики збитків, спричинені неправильним впровадженням процесу, несправністю системи або ризиками деяких зовнішніх подій.

Причини операційних ризиків:

- нездатність вирішити конфлікти пріоритетів;
- невиконання обов'язків;
- недостатні ресурси;
- відсутність належної предметної підготовки;
- відсутність планування ресурсів;
- відсутність спілкування в команді [5].

Технічні ризики зазвичай призводять до збою функціональності та продуктивність.

Причинами технічних ризиків є:

- вимоги, що постійно змінюються;
- відсутня передова технологія або існуюча технологія знаходиться на початкових стадіях;
- продукт складний у реалізації;
- складна інтеграція модулів проекту.

Програмні ризики – це зовнішні ризики поза операційними межами. Усі ці невизначені ризики знаходяться поза контролем програми.

Ці зовнішні події можуть бути:

- закінчуються кошти;
- розвиток ринку;
- зміна стратегії та пріоритету продукту клієнта;
- зміни урядових правил [5].

Хоча ризики розробки ПЗ поділяються на кілька основних категорій (Kuzminykh, Khaustov & Korostelov, 2010), у кожному конкретному випадку можуть виникати додаткові типи ризиків, які не розглянуто в цьому дослідженні. До основних категорій ризиків належать:

- ризики, пов’язані з неповнотою вимог до ПЗ. Включають врахування лише очевидних і реалізацію другорядних вимог, а також ігнорування критичних або відкладання концептуальних вимог;
- технологічні ризики. Пов’язані з незнанням технологій, які персонал планує використовувати для розробки ПЗ, або з недостатньою апробацією та опрацюванням цих технологій у команді виконавців проекту;
- ризики, пов’язані з низькою кваліфікацією персоналу: Керівник проекту повинен бути обізнаний про можливості своїх працівників і, за потреби, організувати їх навчання до початку реалізації проекту, щоб уникнути помилок у вже розробленому ПЗ;
- політичні ризики. Саботаж, який зазвичай не демонструється відкрито, але може знищити проект, якщо учасники мають власні цілі, що не завжди збігаються з цілями керівника проекту [4].

Згідно з класифікацією ризиків розробки програмного забезпечення залежно від наслідків, їх можна поділити на дві категорії: чисті та спекулятивні. Особливість чистих ризиків полягає в тому, що вони, як правило, призводять лише до втрат від підприємницької діяльності, тоді як спекулятивні (комерційні) ризики можуть призвести до фінансових та матеріальних збитків або надати можливість ІТ-компанії отримати додатковий прибуток.

Чисті ризики розробки ПЗ можна класифікувати як ризики проектного управління, проектні, кадрові та делікатні, з подальшим поділом на такі категорії:

- ризики поганої взаємодії між замовником і виконавцем ПЗ. Виникають через відсутність комунікації між керівниками або їх представниками. Недостатнє обговорення вимог до ПЗ або його архітектури може негативно вплинути на функціонал і якість ПЗ;
- ризики неефективного управління проектом. Пов'язані з відсутністю навичок проектного менеджменту у керівника проекту, а також з браком мотивації або інтересу до успішної реалізації проекту;
- ризики недостатньої обізнаності керівника про поточний стан проекту. Виникають через відсутність прямого та зворотного зв'язку під час виконання завдань проекту, що ускладнює контроль на всіх етапах реалізації;
- ризики неефективного планування етапів проекту. Викликані відсутністю навичок планування у керівника або виконавців, які не можуть точно визначити терміни виконання завдань проекту;
- ризики недостатньої системи контролю за виконанням завдань. Зумовлені складністю передбачити всі можливі ситуації в проектному менеджменті ПЗ;
- ризики появи нових користувацьких вимог. Виникають під час розробки ПЗ, коли замовник додає нові вимоги, що відсуває терміни виконання і ускладнює оцінку прогресу;
- ризики суперечностей у вимогах до ПЗ. Виявляються на етапі декомпозиції користувацьких вимог у системні або під час кодування і інтеграції продуктів проекту;
- ризики неправильно сформульованих системних вимог. Виникають через некоректно визначені характеристики цільової системи на початку проекту, включаючи програмне оточення і вимоги до апаратної частини;
- ризики використання нових технологій. Пов'язані з впровадженням нестабільних, неапробованих технологій у виробничому процесі або інших проектах;
- ризики нездатності персоналу впоратися зі складністю розробки ПЗ. Виникають, коли команда розробників не може впоратися з надто складним ПЗ;

− ризики нездатності керівника впоратися з труднощами реалізації проекту. Виникають, коли проект настільки складний, що команда виконавців не може з ним упоратися;

− ризики низької продуктивності команди розробників. Зумовлені значною тривалістю виконання етапів проекту, що призводить до втрати часу на початку і потребує інтенсивної роботи на пізніших етапах;

− ризики зміни працівників проекту. Виникають, коли ключові виконавці покидають проект, втрачаючи знання про його специфіку;

− ризики розкрадання продуктів проекту. Виникають, коли розробники забирають з собою розроблені компоненти і використовують їх у інших проектах;

− ризики порушення закону про авторське право. Виникають, коли виконавці використовують захищені продукти без відома керівника, що порушує авторське право [3].

Ризики розробки програмного забезпечення, пов’язані зі спекулятивними чинниками, можна класифікувати на наступні категорії: фінансові обмеження, зміни кон’юнктури ринку та валютні ризики. Зокрема:

− ризики фінансових обмежень можуть виникнути через недостатність фінансових ресурсів, що може бути спричинено або керівником проекту, що запланував його бюджет і терміни виконання, або іншими причинами, такими як низька фінансова стабільність замовника програмного забезпечення;

− ризики зміни кон’юнктури ринку пов’язані зі змінами економічної ситуації на ринку, що виникають після планування термінів і бюджету проекту. Ці зміни можуть бути не передбачені в початкових умовах проекту, але можуть суттєво вплинути на його реалізацію;

− валютні ризики пов’язані з можливими втратами або додатковими прибутками внаслідок невигідних або вигідних змін валютних курсів, особливо якщо компанія взаємодіє з іноземними замовниками або виконавцями проекту [4].

3.3 Інструменти та техніки управління ризиками

Існує програмне забезпечення, яке може бути використане для управління ризиками в розробці програмного забезпечення (ПЗ). Ось деякі з найпопулярніших інструментів:

− JIRA. JIRA – це платформа для управління проектами, яка включає в себе інструменти для відстеження задач, управління ризиками, спринтів, витрат та багато іншого. Вона дозволяє створювати, відстежувати та аналізувати ризики в рамках розробки ПЗ;

− Microsoft Azure DevOps. Це інтегрована платформа для управління розробкою ПЗ, яка містить у собі інструменти для планування проектів, управління версіями, спринтами, а також для виявлення та керування ризиками;

− IBM Rational DOORS Next Generation. Цей інструмент призначений для керування вимогами та ризиками в проектах розробки ПЗ. Він дозволяє створювати та відстежувати вимоги, а також аналізувати ризики, пов’язані з цими вимогами;

− RiskWatch. Це програмне забезпечення для управління ризиками, яке спеціалізується на оцінці ризиків та управлінні ними в інформаційних технологіях та програмному забезпеченні;

− RiskSense. Це інтегроване програмне забезпечення для управління кібербезпекою, яке допомагає виявляти, аналізувати та керувати ризиками кібербезпеки в розробці ПЗ та інших технологічних проектах.

Ці інструменти можуть надавати різні функції та можливості для управління ризиками в розробці програмного забезпечення. Вибір конкретного програмного забезпечення буде залежати від потреб вашого проекту, розміру команди, бюджету та інших факторів.

Розглянемо детальніше кожен з цих програмних засобів для управління ризиками в розробці програмного забезпечення.

JIRA – це потужна платформа для управління проектами, розроблена компанією Atlassian. Вона пропонує інструменти для планування проектів, відстеження задач, управління версіями, спринтами, а також для виявлення та керування ризиками.

Функції:

− створення задач та епіків;

− організація задач за спринтами та версіями;

− відстеження часу та ресурсів;

− створення звітів та аналіз даних;

− інтеграція з іншими інструментами розробки ПЗ, такими як Bitbucket, Confluence тощо.

Azure DevOps – це інтегрована платформа для управління розробкою ПЗ, яка містить інструменти для спільної розробки коду, тестування, відстеження робочого процесу, управління версіями, а також виявлення та керування ризиками.

Функції:

- спільна робота над кодом та іншими аспектами проекту;
- відстеження задач та вимог;
- управління спринтами, релізами та версіями;
- автоматизація тестування та розгортання;
- інтеграція з Azure, GitHub, Jenkins тощо.

DOORS Next Generation (раніше відомий як IBM Rational DOORS) – це інструмент для керування вимогами та ризиками в проектах розробки ПЗ. Він дозволяє створювати та відстежувати вимоги, а також аналізувати ризики, пов’язані з цими вимогами.

Функції:

- збір, організація та відстеження вимог;
- аналіз вимог та ідентифікація конфліктів;
- виявлення та керування ризиками;
- забезпечення відповідності стандартам та вимогам.

RiskWatch – це програмне забезпечення для управління ризиками, яке спеціалізується на оцінці та управлінні ризиками в інформаційних технологіях та програмному забезпеченні.

Функції:

- оцінка та аналіз ризиків в інформаційних технологіях;
- створення та відстеження планів з управління ризиками;
- автоматизований процес збору та оцінки ризиків;
- генерація звітів та документації.

RiskSense – це інтегроване програмне забезпечення для управління кібербезпекою, яке допомагає виявляти, аналізувати та керувати ризиками кібербезпеки в розробці ПЗ та інших технологічних проектах.

Функції:

- сканування та оцінка вразливостей програмного забезпечення;
- аналіз загроз кібербезпеки та виявлення вразливих місць;
- виявлення критичних ризиків та пріоритизація їхнього вирішення;
- моніторинг та аналіз кіберзагроз на основі великих обсягів даних.

Ці програмні засоби можуть бути використані для ефективного управління ризиками в розробці програмного забезпечення, забезпечуючи аналіз, відстеження.

SEI (Software Engineering Institute) рекомендує три методології для управління ризиками проектів.

Методологія оцінювання ризику SRE (Software Risk Evaluation) включає формальний підхід до ідентифікації, аналізу, контролю та усунення ризиків проектів. Ця методологія застосовується на ранніх етапах розробки програмного забезпечення (ще до укладення договору з розробником) і регулярно протягом усього життєвого циклу проекту.

Методологія безперервного управління ризиками (CRM, Continuous Risk Management) заснована на постійних принципах управління ризиками протягом усього життєвого циклу проекту, незалежно від конкретних методів та інструментів оцінки та усунення ризиків.

Методологія колективного управління ризиками (TRM, Team Risk Management) передбачає додаткові заходи для управління ризиками, які включають спільну участь замовника проекту та виконавця в процесі управління ризиками.

Методологія оцінювання ризику у сфері надійності сайтів (Site Reliability Engineering, SRE) є важливим аспектом управління сучасними IT-системами. Вона допомагає командам SRE ідентифікувати потенційні загрози, оцінити їх вплив і ймовірність, а також розробити стратегії для їх мінімізації або усунення. Ось основні етапи і принципи цієї методології.

Перший крок – визначення основних цілей системи та її ключових показників ефективності (Key Performance Indicators, KPIs). Важливо розуміти, що саме необхідно захистити і які метрики визначають успіх роботи системи.

Ідентифікація загроз. Цей етап включає визначення всіх можливих загроз, які можуть вплинути на надійність системи. Загрози можуть бути внутрішніми (помилки програмного забезпечення, людські помилки) і зовнішніми (кібератаки, природні катастрофи).

Оцінка ризиків. На цьому етапі кожна ідентифікована загроза оцінюється за двома основними параметрами: ймовірністю її виникнення і потенційним впливом на систему. Для оцінки можуть використовуватися як якісні, так і кількісні методи.

Після оцінки ризиків важливо визначити, які з них потребують першочергової уваги. Це дозволяє ефективно розподіляти ресурси для управління ризиками.

Розробка стратегії управління ризиками. На цьому етапі розробляються плани щодо мінімізації або усунення ідентифікованих ризиків. Це можуть бути технічні рішення (резервування даних, підвищення стійкості системи), процесні зміни (впровадження нових політик і процедур) або навчання персоналу.

Плани з управління ризиками впроваджуються в практику, і встановлюються механізми моніторингу для відстеження їх ефективності. Це може включати регулярні перевірки системи, тестування на стійкість до різних видів загроз, а також аналіз логів і метрик.

Оцінка ефективності заходів з управління ризиками здійснюється на регулярній основі. За необхідності плани коригуються для забезпечення постійного покращення надійності системи.

Для підтримки процесу оцінювання ризиків у SRE використовуються різні інструменти, такі як:

- Automated monitoring tools. Наприклад, Prometheus, Grafana;
- Incident management systems. Наприклад, PagerDuty, Opsgenie;
- Risk assessment frameworks. Зразок NIST, ISO 31000.

Методологія оцінювання ризику у SRE є комплексним процесом, що включає ідентифікацію, оцінку, пріоритизацію і управління ризиками з метою забезпечення максимальної надійності і стійкості IT-систем. Вона допомагає командам SRE ефективно планувати і діяти у разі виникнення проблем, зменшуючи негативний вплив інцидентів на бізнес.

Методологія безперервного керування ризиком (Continuous Risk Management, CRM) – це підхід, який використовується для ідентифікації, оцінки, моніторингу і управління ризиками на постійній основі протягом усього життєвого циклу проєкту або системи. Цей підхід дозволяє організаціям проактивно управляти ризиками, забезпечуючи гнучкість і стійкість до змінних умов. Основні етапи та принципи методології CRM включають наступне.

Ідентифікація ризиків. Перший крок у CRM – це постійне виявлення можливих ризиків, які можуть вплинути на проєкт або систему. Це може бути досягнуто шляхом регулярних оглядів, аналізу минулих інцидентів, консультацій з експертами та використання спеціалізованих інструментів.

Оцінка ризиків. Після ідентифікації ризиків необхідно оцінити їх ймовірність і потенційний вплив. Це включає як якісні, так і кількісні методи оцінки.

Розробка стратегії управління ризиками. На цьому етапі розробляються заходи з управління ризиками, включаючи їх мінімізацію, уникнення, передачу або прийняття. Ці заходи повинні бути інтегровані в загальну стратегію управління проєктом або системою.

Впровадження заходів. Заплановані заходи щодо управління ризиками реалізуються на практиці. Важливо забезпечити, щоб всі учасники проєкту були обізнані про свої ролі і відповідальність у процесі управління ризиками.

Моніторинг і контроль. Ризики постійно відстежуються і контролюються на всіх етапах життєвого циклу проєкту або системи. Це включає регулярні перевірки, аналіз даних і проведення аудитів для оцінки ефективності заходів з управління ризиками.

Оцінка і вдосконалення Методологія CRM передбачає регулярне оцінювання ефективності впроваджених заходів і коригування планів за необхідності. Це дозволяє забезпечити постійне вдосконалення процесу управління ризиками.

Для підтримки процесу CRM використовуються різні інструменти, такі як:

− Risk management software. Наприклад, RiskWatch, RiskWatch International;

− Project management tools. Наприклад, Microsoft Project, Jira;

− Data analysis tools. Наприклад, R, Python для статистичного аналізу ризиків.

Переваги методології CRM:

− прозорість. Регулярне відстеження ризиків забезпечує прозорість процесів і прийняття рішень;

− проактивність. CRM дозволяє проактивно виявляти і усувати ризики до того, як вони стануть серйозною проблемою;

− гнучкість. Постійне управління ризиками допомагає швидко адаптуватися до змінних умов і нових викликів;

− поліпшення якості. CRM сприяє підвищенню якості управління проєктами за рахунок постійного вдосконалення процесів і процедур.

Методологія безперервного керування ризиком (CRM) є ефективним підходом до управління ризиками, що забезпечує безперервний моніторинг і

контроль на всіх етапах життєвого циклу проєкту або системи. Вона дозволяє організаціям проактивно управляти ризиками, що сприяє підвищенню гнучкості, стійкості і загальної ефективності управління проєктами.

Методологія колективного управління ризиком (Team Risk Management, TRM) – це підхід, що акцентує увагу на спільному зусиллі команди для ідентифікації, оцінки, моніторингу і управління ризиками. Мета TRM полягає в об'єднанні знань, досвіду і навичок усіх учасників команди для ефективнішого виявлення та вирішення ризиків. Це забезпечує кращу координацію, підвищує якість прийняття рішень і сприяє створенню спільного бачення проблем і можливостей. Основні етапи та принципи TRM включають наступне [6].

Формування команди. Процес TRM починається з формування міждисциплінарної команди, яка включає представників з різних функціональних областей. Важливо, щоб у команді були експерти, які мають різний досвід і знання для забезпечення всебічного аналізу ризиків.

Ідентифікація ризиків. На цьому етапі команда спільно ідентифікує всі можливі ризики, які можуть вплинути на проєкт. Це може включати мозкові штурми, аналіз минулих інцидентів, SWOT-аналіз і використання спеціалізованих інструментів для виявлення ризиків.

Оцінка ризиків. Після ідентифікації ризиків команда оцінює їх ймовірність і вплив. Цей процес зазвичай включає наступні етапи.

Пріоритизація ризиків. Команда спільно визначає пріоритети для кожного ризику, що дозволяє сфокусувати зусилля на найважливіших і найнебезпечніших ризиках. Це може бути досягнуто шляхом голосування, матричного аналізу ризиків або інших методів.

Розробка стратегії управління ризиками. На цьому етапі команда розробляє стратегії і плани щодо управління ризиками. Це включає розробку заходів для мінімізації, уникнення, передачі або прийняття ризиків, а також визначення відповідальних осіб за виконання цих заходів.

Впровадження заходів і моніторинг. Розроблені стратегії впроваджуються на практиці, а команда постійно моніторить їх ефективність. Це включає регулярні зустрічі для обговорення прогресу, аналіз даних і коригування планів за необхідності.

Оцінка і вдосконалення. Команда проводить регулярні оцінки ефективності впроваджених заходів і вносить корективи для постійного вдосконалення процесу управління ризиками.

Переваги TRM:

− колективна мудрість. Об’єднання знань і досвіду всіх учасників команди для більш повного і точного виявлення ризиків;

− координація зусиль. Поліпшення комунікації і координації між учасниками команди, що підвищує ефективність управління ризиками;

− спільне прийняття рішень. Залучення всіх членів команди до процесу прийняття рішень, що підвищує їх прихильність і відповідальність;

− гнучкість і адаптивність. Можливість швидко реагувати на зміни в проєкті або зовнішньому середовищі завдяки колективному підходу.

Інструменти для TRM:

− Collaborative software. Інструменти для спільної роботи, такі як Microsoft Teams, Slack, Confluence;

− Risk management tools. Спеціалізовані програми для управління ризиками, як-от RiskWatch, Active Risk Manager;

− Project management software. Програми для управління проєктами, такі як Jira, Trello, Asana.

Методологія колективного управління ризиком (TRM) є ефективним підходом для управління ризиками, який базується на спільній роботі команди. Вона дозволяє об’єднати зусилля, знання і досвід різних учасників для ефективнішого виявлення, оцінки і управління ризиками. TRM сприяє поліпшенню комунікації, координації і прийняття рішень, що в кінцевому рахунку підвищує успішність проєктів і стійкість організації до ризиків [7].

Питання для самоконтролю

1. Які основні етапи включає процес управління ризиками в розробці програмного забезпечення, і в чому полягає мета кожного етапу?
2. Чому управління ризиками є критично важливим для успішного завершення проектів розробки програмного забезпечення, і які фактори визначають актуальність цієї теми?
3. Як класифікуються ризики у проектах розробки програмного забезпечення, і які приклади ризиків можуть виникати в кожній категорії?
4. Які методи та інструменти можуть бути використані для ідентифікації та оцінки ризиків у проектах розробки програмного забезпечення?

5. Які стратегії можуть бути розроблені для управління різними типами ризиків, і як визначити ефективність цих стратегій у процесі реалізації проекту?

6. Які є основні принципи оцінки ризику і як вони впливають на процес управління ризиками?

7. У чому полягає відмінність між мірою ризику та ціною ризику, і як їх фізичний зміст відображається у проекті?

8. Які методи якісного аналізу ризиків ви знаєте, і як вони допомагають у розкритті сутності та чинників ризиків?

9. Як експертні методи використовуються для якісного аналізу ризиків і які переваги вони мають порівняно з іншими методами?

10. Які кількісні методи аналізу ризиків ви знаєте, і які їхні особливості у порівнянні з якісними методами?

11. Які основні відмінності між чистими та спекулятивними ризиками в розробці програмного забезпечення?

12. Назвіть кілька прикладів чистих ризиків розробки програмного забезпечення та поясніть, чому вони відносяться до цієї категорії.

13. Які стратегії можуть бути використані для управління ризиками розробки програмного забезпечення, зокрема для мінімізації чистих ризиків?

Тестові завдання

1. Що включає в себе процес управління ризиками в програмній інженерії?
a) тільки ідентифікація ризиків;
b) ідентифікація, аналіз, розробка стратегій та моніторинг ризиків;
c) тільки аналіз ризиків.

2. Які фактори визначають актуальність управління ризиками в розробці програмного забезпечення?
a) тільки складність проектів ПЗ;
b) збільшення вимог та технологічні виклики;
c) тільки командні динаміки.

3. Які етапи включає в себе управління ризиками в розробці програмного забезпечення?
a) тільки аналіз ризиків;

b) ідентифікація, аналіз, розробка стратегій та моніторинг ризиків;
c) тільки розробка стратегій управління ризиками.

4. Що означає ухилення від ризику в управлінні проектами?
a) заплановане прийняття ризику;
b) зниження ризику;
c) уникають виконання певних дій або рішень з високим ризиком.

5. Який підхід використовується, коли ризик знаходиться на прийнятному рівні, а вплив на нього неможливий або економічно неефективний?
a) заплановане прийняття ризику;
b) попередження ризику;
c) розподіл ризику.

6. Який метод управління ризиками передбачає проведення превентивних заходів для зниження ймовірності виникнення негативних подій?
a) аутсорсинг ризику;
b) заплановане прийняття ризику;
c) попередження ризику.

7. Які з наведених нижче не відносяться до основних категорій ризиків розробки програмного забезпечення?
a) ризики, пов’язані з неповнотою вимог до ПЗ;
b) ризики використання нових технологій;
c) ризики фінансових обмежень.

8. Які з наведених характеристик не відносяться до чистих ризиків розробки ПЗ?
a) ризики недостатньої обізнаності керівника про поточний стан проекту;
b) ризики порушення закону про авторське право;
c) ризики валютних коливань.

9. Які з наведених факторів не відносяться до спекулятивних чинників ризиків розробки програмного забезпечення?
a) ризики зміни кон’юнктури ринку;
b) ризики недостатньої системи контролю за виконанням завдань;
c) ризики політичних обставин, що можуть вплинути на проект.

Рекомендована література

1. Управління ризиками: Навчальний наочний посібник: навч. посіб. для студ. спеціальності 073 «Менеджмент» / М. О. Кравченко, К. О. Бояринова, К. О. Копішинська; КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2021. 432с. URL: https://ela.kpi.ua/items/bf47e3e2-a76e-4096-9647-85aa0d36f486.
2. Проскура В. Ф. , Білак Р. Г. Методологічні підходи до управління ризиками. URL: https://economyandsociety.in.ua/journals/9_ukr/102.pdf.
3. Управління ризиками в інженерії програмного забезпечення: конспект лекцій для здобувачів другого (магістерського) рівня вищої освіти освітньої програми «Інженерія програмного забезпечення» галузі знань 12 Інформаційні технології спеціальності 121 Інженерія програмного забезпечення денної та заочної форм навчання / уклад. Н. Ліщина, В. Ліщина. Луцьк : ЛНТУ, 2022. 122 с.
4. Трофименко О. Г., Логінова Н. І., Тесленко П. О., Савєльєва С.В., Поляков В. М. Класифікація ризиків у проєктах із розробки програмного забезпечення. Вісник Херсонського національного технічного університету. Інформаційні технології, 2023. № 3(86). С. 119-128. URL: https://journals.kntu.kherson.ua/index.php/visnyk_kntu/article/view/456.
5. Er. Jaspreet Kaur. Models, Techniques and Metrics for Managing Risk in Software Engineering. International Research Journal of Engineering and Technology (IRJET). Volume: 07 Issue: 10 | Oct 2020. URL: https://www.irjet.net/archives/V7/i10/IRJET-V7I10351.pdf.
6. Грицюк Ю. І., Далявський В. С. Формалізація процесу управління ризиками розроблення програмного забезпечення. URL: https://nv.nltu.edu.ua/index.php/journal/article/view/1797.
7. Данченко О. Б., Занора В. О. Проектний менеджмент: управління ризиками та змінами в процесах прийняття управлінських рішень: монографія /. Черкаси: ПП Чабаненко Ю.А., 2019. 278 с.

РОЗДІЛ 4
МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ В КОМП'ЮТЕРНИХ СИСТЕМАХ
4.1 Основи систем захисту інформації у комп'ютерних системах

Інформаційні ресурси держави, суспільства, організацій та фізичних осіб мають значну цінність, матеріальне вираження та потребують захисту від різноманітних впливів, що можуть знизити їхню цінність. Впливи, що спричиняють зниження цінності інформаційних ресурсів, називаються несприятливими, а потенційні несприятливі впливи – загрозами.

Захист інформації в комп'ютерних системах полягає в створенні та підтримці ефективної системи технічних (інженерних, програмно-апаратних) і нетехнічних (правових, організаційних) заходів, що дозволяють запобігти або ускладнити реалізацію загроз та знизити потенційні збитки. Це спрямовано на забезпечення безпеки оброблюваної інформації та комп'ютерних систем загалом, забезпечуючи збереження їхніх властивостей. Система заходів для захисту інформації в комп'ютерних системах називається комплексною системою захисту інформації.

Більшість проблем захисту інформації в комп'ютерних системах може бути вирішена організаційними заходами, однак з розвитком інформаційних технологій зростає потреба у застосуванні технічних засобів захисту.

Правовою основою забезпечення технічного захисту інформації в Україні є Конституція України, Закони України «Про інформацію», «Про захист інформації в комп'ютерних системах», «Про державну таємницю», «Про науково-технічну інформацію», Концепція національної безпеки України, Концепція технічного захисту інформації в Україні, інші нормативно-правові акти та міжнародні договори України у сфері інформаційних відносин [1].

Комп'ютерні системи підприємств, відповідно до класифікації НД ТЗІ 2.5-005, включають локалізовані багатомашинні багатокористувацькі комплекси. До складу таких систем входять обчислювальна система, фізичне середовище, користувачі та оброблювана інформація, включаючи технологію її обробки. Під час захисту інформації необхідно враховувати всі характеристики цих складових, які впливають на реалізацію політики безпеки.

Інформація в комп'ютерних системах підприємства існує у вигляді даних, тобто подається у формалізованому вигляді, придатному для обробки. Обробка включає введення, виведення, зберігання, передачу тощо (ДСТУ

2226–93). Надалі терміни «інформація» і «дані» використовуються як синоніми.

Інформація завжди потребує наявності носія. Носієм інформації може бути поле, речовина або навіть людина. Втрата інформацією своєї цінності (порушення безпеки інформації) може статися внаслідок переміщення інформації або зміни фізичних властивостей носія.

При аналізі проблеми захисту від несанкціонованого доступу (НСД) до інформації, що циркулює в комп'ютерних системах, зазвичай розглядаються лише інформаційні об'єкти, які слугують приймачами або джерелами інформації, та інформаційні потоки (порції інформації, що пересилаються між об'єктами), безвідносно до фізичних характеристик їх носіїв.

Загрози інформації в комп'ютерних системах залежать від характеристик операційної системи, фізичного середовища, персоналу та оброблюваної інформації. Загрози можуть бути об'єктивними, такими як зміна умов фізичного середовища (пожежі, повені тощо) або відмова елементів операційної системи, або суб'єктивними, такими як помилки персоналу чи дії зловмисника. Суб'єктивні загрози можуть бути випадковими або навмисними. Спроба реалізації загрози називається атакою [2].

Найпридатнішою для аналізу є класифікація загроз за результатом їх впливу на інформацію, тобто порушення конфіденційності, цілісності та доступності інформації:

– **конфіденційність** інформації зберігається, якщо дотримуються встановлені правила ознайомлення з нею;

– **цілісність** інформації зберігається, якщо дотримуються встановлені правила її модифікації (видалення);

– **доступність** інформації зберігається, якщо дотримуються правила ознайомлення з нею або її модифікації впродовж будь-якого певного проміжку часу.

Загрози, реалізація яких призводить до втрати інформацією однієї з цих властивостей, відповідно є загрозами конфіденційності, цілісності або доступності інформації.

Загрози можуть впливати на інформацію опосередковано. Наприклад, втрата керованості комп'ютерною системою може призвести до нездатності забезпечувати захист інформації, що, в свою чергу, призведе до втрати певних властивостей оброблюваної інформації.

Під несанкціонованим доступом (НСД) слід розуміти доступ до інформації з використанням засобів, внесених до складу комп'ютерної системи (КС), що порушує встановлені правила розмежування доступу (ПРД). НСД може здійснюватись як з використанням штатних засобів, тобто сукупності програмно-апаратного забезпечення, внесеного до складу КС розробником під час розробки або системним адміністратором в процесі експлуатації, що входить у затверджену конфігурацію КС, так і з використанням програмно-апаратних засобів, внесених до складу КС зловмисником.

До основних способів НСД належать:

– безпосереднє звернення до об'єктів з метою одержання певного виду доступу;

– створення програмно-апаратних засобів, що виконують звернення до об'єктів в обхід засобів захисту;

– модифікація засобів захисту, що дозволяє здійснити НСД;

– впровадження в КС програмних або апаратних механізмів, що порушують структуру та функції КС і дозволяють здійснити НСД.

Під захистом від НСД слід розуміти діяльність, спрямовану на забезпечення додержання правил розмежування доступу шляхом створення і підтримки в дієздатному стані системи заходів із захисту інформації.

Комп'ютерна система (КС) являє собою організаційно-технічну систему, що об'єднує обчислювальну систему, фізичне середовище, персонал і оброблювану інформацію. Прийнято розрізняти два основних напрями технічного захисту інформації (ТЗІ) в КС:

– захист КС і оброблюваної інформації від несанкціонованого доступу;

– захист інформації від витоку технічними каналами (оптичними, акустичними, захист від витоку каналами побічних електромагнітних випромінювань і наведень).

Кінцевою метою всіх заходів захисту інформації є забезпечення її безпеки під час обробки в автоматизованих системах (АС). Захист інформації повинен забезпечуватися на всіх стадіях життєвого циклу КС, на всіх технологічних етапах обробки інформації і в усіх режимах функціонування. Життєвий цикл КС охоплює розробку, впровадження, експлуатацію та виведення з експлуатації.

Якщо в КС планується обробка інформації, порядок обробки і захисту якої регламентується законами України або іншими нормативно-правовими актами (наприклад, інформація, що становить державну таємницю), то для

обробки такої інформації необхідно мати дозвіл відповідного уповноваженого державного органу. Підставою для видачі такого дозволу є висновок експертизи КС, тобто перевірки відповідності реалізованої комплексної системи захисту інформації (КСЗІ) встановленим нормам.

В процесі експертизи оцінюється КСЗІ КС в цілому, включаючи засоби захисту, реалізовані в операційній системі (ОС) КС. Засоби захисту від НСД, реалізовані в обчислювальній системі, слід розглядати як підсистему захисту від НСД у складі КСЗІ. Характеристики фізичного середовища, персоналу, оброблюваної інформації, організаційної підсистеми впливають на вимоги до функцій захисту, що реалізуються ОС.

Обчислювальна система комп'ютерної системи складається з апаратних засобів, програмних засобів (в тому числі програм ПЗП), призначених для обробки інформації. Кожен з компонентів ОС може розроблятись і надходити на ринок як незалежний продукт. Кожен з цих компонентів може реалізовувати певні функції захисту інформації, оцінювання яких може виконуватись незалежно від процесу експертизи КС і має характер сертифікації. За підсумками сертифікації видається сертифікат відповідності реалізованих засобів захисту певним вимогам (критеріям). Наявність сертифіката на обчислювальну систему КС або її окремі компоненти може полегшити процес експертизи КС.

Оцінювання реалізованих функцій захисту інформації виконується відповідно до встановлених критеріїв, визначених НД ТЗІ 2.5-004-99 "Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу" (далі – Критерії).

Як КС можуть виступати:

– ЕОМ загального призначення або персональна ЕОМ;

– операційна система;

– прикладна або інструментальна програма (пакет програм);

– комплекс засобів захисту (КЗЗ), що окремо поставляється, або підсистема захисту від НСД, наприклад, мережа, яка являє собою надбудову над ОС;

– локальна обчислювальна мережа як сукупність апаратних засобів, ПЗ, що реалізує протоколи взаємодії мережевої операційної системи;

– ОС комп'ютерної системи, яка реально функціонує;

– в найбільш загальному випадку – сама КС або її частина.

Коли для побудови КС використовуються компоненти, кожний або деякі з яких мають сертифікат, це підтверджує, що дані компоненти реалізують певні функції захисту інформації. Однак це не означає, що КС, яка складається з таких компонентів, буде реалізовувати всі ці функції [3].

Для гарантії реалізації всіх функцій захисту необхідно виконати проектування КС з метою інтеграції засобів захисту, що надаються кожним компонентом, в єдиний комплекс засобів захисту. Таким чином, наявність сертифіката слід розглядати як потенційну можливість КС реалізувати певні функції захисту оброблюваної інформації від певних загроз.

Під політикою безпеки інформації слід розуміти набір законів, правил, обмежень та рекомендацій, які регламентують порядок обробки інформації та спрямовані на її захист від певних загроз. Термін "політика безпеки" може застосовуватися до організації, комп'ютерної системи (КС), операційної системи (ОС), послуги або функцій, реалізованих системою.

Політика безпеки інформації в КС є частиною загальної політики безпеки організації та може успадковувати положення державної політики у галузі захисту інформації. Для кожної КС політика безпеки може бути індивідуальною, залежно від реалізованої технології обробки інформації, особливостей ОС, фізичного середовища та інших чинників.

КС може реалізовувати декілька технологій обробки інформації, що може призвести до складеної політики безпеки, де її частини, що відповідають різним технологіям, можуть істотно різнитися. Політика безпеки повинна визначати ресурси КС, які потребують захисту, встановлювати категорії оброблюваної інформації, формулювати основні загрози для ОС, персоналу та інформації різних категорій, а також вимоги до захисту від цих загроз.

Як складові частини загальної політики безпеки інформації в КС мають існувати політики забезпечення конфіденційності, цілісності та доступності оброблюваної інформації. Відповідальність персоналу за виконання положень політики безпеки повинна бути персоніфікована.

Типові адміністративні та організаційні вимоги до обчислювальної системи КС, умов її функціонування та забезпечення захисту інформації визначаються таким чином:

– для КС та її компонентів повинен бути сформований перелік необхідних функціональних послуг захисту і визначено рівень гарантій їх реалізації [4];

– сервери, робочі станції, периферійні пристрої та інші технічні засоби обробки інформації повинні бути категоризовані згідно з вимогами нормативних документів із технічного захисту інформації;

– засоби захисту інформації, інші технічні засоби та програмне забезпечення КС, що задіяні в КСЗІ, повинні мати підтвердження їх відповідності нормативним документам із захисту інформації (атестат, сертифікат відповідності, експертний висновок) і використовуватися згідно з цими документами;

– технічна та експлуатаційна документація на засоби захисту та обробки інформації, системне та функціональне програмне забезпечення повинна бути належним чином класифікована, а для кожної категорії користувачів визначено перелік документації, до якої вони можуть отримати доступ. Доступ до документації фіксується у відповідних реєстрах.сервери і робочі станції, що здійснюють зберігання та обробку інформації, повинні розташовуватися в приміщеннях з обмеженим доступом, визначеним СЗІ та затвердженим керівником установи;

– повинен здійснюватися контроль за доступом користувачів та обслуговувального персоналу до робочих станцій, серверів КС і компонентів підсистеми обміну даними на всіх етапах життєвого циклу КС;

– КС повинна мати можливість оперативного, безперервного обслуговування та модернізації обчислювальної системи без припинення її функціонування. Порядок введення в експлуатацію нових компонентів, якщо це впливає на захист інформації, визначається СЗІ;

– програмно-апаратні засоби захисту повинні забезпечувати СЗІ інформацією про користувачів, які працюють в системі, з локалізацією їх входу та переліком технічних засобів і процесів, до яких вони отримали доступ;

– має бути визначено порядок організації та проведення СЗІ процедур періодичного та/або динамічного тестування комплексу засобів захисту інформації під час функціонування КС.

Характеристика користувачів КС. За рівнем повноважень щодо доступу до інформації, характером та складом робіт, які виконуються в процесі функціонування комп’ютерних систем, особи, що мають доступ до КС, поділяються на такі категорії:

– користувачі, які мають повноваження розробляти й супроводжувати КСЗІ (адміністратор безпеки, співробітники ПЗІ);

– користувачі, які мають повноваження забезпечувати управління КС (адміністратори ОС, СКБД, мережевого обладнання, сервісів тощо);

– користувачі, які мають право доступу до інформації одного або декількох класифікаційних рівнів;

– користувачі, які мають право доступу тільки до відкритої інформації;

– технічний обслуговувальний персонал, що забезпечує належні умови функціонування КС;

– розробники та проектувальники апаратних засобів КС, що забезпечують її модернізацію та розвиток;

– розробники програмного забезпечення, які здійснюють розробку та впровадження нових функціональних процесів, а також супроводження вже діючих;

– постачальники обладнання і технічних засобів КС та фахівці, що здійснюють його монтаж, поточне гарантійне й післягарантійне обслуговування;

– технічний персонал, що здійснює повсякденне підтримання життєдіяльності фізичного середовища КС (електрики, технічний персонал з обслуговування будівель, ліній зв'язку тощо).

Усі користувачі та персонал КС повинні пройти підготовку щодо умов та правил використання технічних і програмних засобів, які застосовуються ними під час виконання службових та функціональних обов'язків. Доступ осіб всіх категорій до інформації та її носіїв здійснюється на підставі дозволу, що надається наказом (розпорядженням) керівника організації. Дозвіл надається лише для виконання службових та функціональних обов'язків і на термін не більший, ніж той, що цими обов'язками передбачений.

Якщо в КС встановлено декілька класифікаційних рівнів інформації, кожній особі з допущених до роботи в КС мають бути визначені повноваження щодо доступу до інформації певного класифікаційного рівня. Дозвіл на доступ до інформації, що обробляється в КС, може надаватися лише користувачам. Як виняток, у випадках аварій або інших непередбачених ситуацій, дозвіл може надаватися іншим особам на час ліквідації негативних наслідків і відновлення працездатності КС.

Персонал КС, розробники програмного забезпечення, розробники та проектувальники апаратних засобів, постачальники обладнання та фахівці, що здійснюють монтаж і обслуговування технічних засобів КС і не мають дозволу на доступ до інформації, можуть мати доступ до програмних і апаратних

засобів КС лише під час робіт із тестування та інсталяції програмного забезпечення, встановлення та регламентного обслуговування обладнання тощо, за умови обмеження їх доступу до даних конфіденційного характеру.

Зазначені категорії осіб повинні мати дозвіл на доступ лише до конфіденційних відомостей, що містяться в програмній і технічній документації на КС або на окремі її компоненти та необхідні для виконання їх функціональних обов'язків [5].

Порядок і механізми доступу до інформації та компонентів КС для різних категорій осіб розробляються ПЗІ та затверджуються керівником організації. Для організації управління доступом до інформації та компонентів

Характеристика оброблюваної в КС інформації. В комп'ютерних системах (КС) обробляється інформація з обмеженим доступом (ІзОД), володіти, користуватися чи розпоряджатися якою можуть окремі фізичні та/або юридичні особи, що мають доступ до неї відповідно до правил, встановлених власником цієї інформації.

В КС може зберігатися та циркулювати відкрита інформація, яка не потребує захисту або захист якої забезпечувати недоцільно, а також відкрита інформація, яка, відповідно до рішень її власника, може потребувати захисту. Конфіденційна й відкрита інформація можуть циркулювати та оброблятися в КС як різними процесами для кожної з категорій інформації, так і в межах одного процесу.

Інформація в КС характеризується за рівнем інтеграції як:

– сукупність сильно пов'язаних об'єктів, що вимагають забезпечення своєї цілісності як сукупності;

– окремі слабо пов'язані об'єкти, що мають широкий спектр способів подання, зберігання та передачі й вимагають забезпечення своєї цілісності кожний окремо.

Об'єкти можуть бути структурованими або неструктурованими незалежно від способу подання. Інформація в КС за часом існування та функціонування може бути:

– швидкозмінюваною з відносно коротким терміном її актуальності;

– мати відносно тривалий час існування при високому ступені інтеграції та гарантуванні стану її незруйнованості за умови належності різним користувачам у рамках сильно- або слабо пов'язаних об'єктів.

Комплексна система захисту інформації (КСЗІ) повинна забезпечити доступність зазначених видів інформації відповідно до особливостей процесів, що реалізують інформаційну модель конкретного фізичного об'єкта.

КС повинна підтримувати окремі класи сукупностей сильно пов'язаних об'єктів стандартними для галузі системами керування базами даних, іншими функціональними чи системними процесами, які забезпечують паралельну обробку запитів та гарантують конфіденційність і цілісність інформації на рівні таблиць, стовпців таблиці та записів таблиці.

КС також повинна підтримувати окремі класи сукупностей слабо пов'язаних об'єктів стандартними для галузі операційними системами, які мають засоби для гарантування конфіденційності й цілісності інформації на рівні сукупності файлів та окремих файлів.

КСЗІ повинна гарантувати цілісність, конфіденційність і доступність інформації, що міститься в сильно- або слабо пов'язаних об'єктах та має ступінь обмеження доступу, згідно з визначеними вимогами до відповідного функціонального профілю захищеності.

Принципи, на яких базується інформаційна безпека:

1) **доступність:**

– забезпечення доступу до загальнодоступних даних усім користувачам, захист цих даних від спотворення та блокування зловмисниками;

– забезпечення доступу до даних на основі розподілу прав доступу;

– захист даних від зловмисного або випадкового видалення чи спотворення;

2) конфіденційність:

– забезпечення доступу до загальнодоступних даних усім користувачам, захист цих даних від спотворення та блокування зловмисниками;

– забезпечення доступу до даних на основі розподілу прав доступу.

– захист даних від зловмисного або випадкового видалення чи спотворення.

Класифікація загроз за метою:

– зловмисні;

– випадкові;

– зовнішні;

– внутрішні;

– природні;

– техногенні.

Кримінальним кодексом України НЕ передбачено кримінальну відповідальність за:

– порушення таємниці листування, телефонних розмов, телеграфної чи іншої кореспонденції, що передаються засобами зв'язку або через комп'ютер;

– незаконне відтворення, розповсюдження творів науки, літератури і мистецтва, комп'ютерних програм і баз даних, їх незаконне тиражування та розповсюдження на аудіо- та відеокасетах, дискетах, інших носіях інформації;

– незаконні дії з документами на переказ, платіжними картками та іншими засобами доступу до банківських рахунків, електронними грошима, обладнанням для їх виготовлення;

– несанкціоноване втручання в роботу електронно-обчислювальних машин (комп'ютерів), автоматизованих систем, комп'ютерних мереж чи мереж електрозв'язку;

– створення з метою використання, розповсюдження або збуту шкідливих програмних чи технічних засобів, а також їх розповсюдження або збут;

– несанкціонований збут або розповсюдження інформації з обмеженим доступом, яка зберігається в електронно-обчислювальних машинах (комп'ютерах), автоматизованих системах, комп'ютерних мережах або на носіях такої інформації;

– недотримання морально-етичних та/або правових норм під час використання електронно-обчислювальної техніки (комп'ютера), комп'ютерних мереж тощо.

Рекомендована література

1. Бобало Ю. Я. Інформаційна безпека: навч. Посібник. Львів: Видавництво Львівської політехніки, 2019. 580 с.
2. Гніліцький В.В., Орєхов Є.Г. Захист інформації. Навчальний посібник для студентів технічних спеціальностей. Житомир, 2021. 164 с.
3. Гребенюк А. М. Основи управління інформаційною безпекою. Навч. посібник. Дніпро: Дніпроп. держ. університет внутрішніх справ, 2020. 144 с.
4. Клінцев Л.М. Безпека програм і даних. Чернігів: ВСП Чернігівський інститут інформації, бізнесу і права, 2019. 482 с.
5. Кузнецов О. О. Захист інформації в інформаційних системах. Навчальний посібник. Х.: ХНЕУ, 2020. 510 с.

4.2 Принципи створення комплексної системи захисту інформації

Загроза інформаційної безпеки

Загроза інформаційної безпеки – це сукупність умов і факторів, що створюють небезпеку порушення інформаційної безпеки. Загрози інформаційної безпеки можуть бути класифіковані за різними ознаками.

За аспектом інформаційної безпеки, на який спрямовані загрози:

1) загрози конфіденційності (неправомірний доступ до інформації):

– загроза порушення конфіденційності полягає в тому, що інформація стає відомою особі, яка не має повноважень доступу до неї. Вона має місце, коли отримано доступ до інформації обмеженого доступу, що зберігається в комп'ютерній системі або передається від однієї системи до іншої. Використовується термін «витік». Подібні загрози можуть виникати через людський фактор (наприклад, випадкове делегування привілеїв іншому користувачу), збої роботи програмних та апаратних засобів. До інформації обмеженого доступу належать державна таємниця, комерційна таємниця, персональні дані та інші види таємниць (лікарська, адвокатська, банківська, службова, нотаріальна таємниця страхування, слідства й судочинства, листування, телефонних переговорів, поштових відправлень, телеграфних або інших повідомлень, відомості про сутність винаходу, корисної моделі або промислового зразка до офіційної публікації) [1];

2) загрози цілісності (неправомірна зміна даних):

– загрози порушення цілісності пов'язані з ймовірністю модифікації інформації, що зберігається в інформаційній системі. Порушення цілісності може бути викликано різними чинниками – від умисних дій персоналу до виходу з ладу обладнання;

3) загрози доступності (здійснення дій, які унеможливлюють або ускладнюють доступ до ресурсів інформаційної системи):

– порушення доступності створює умови, за яких доступ до послуги або інформації блокується або можливий лише за час, що не забезпечить виконання бізнес-цілей;

За розташуванням джерела загроз:

– внутрішні. Джерела загроз розташовуються всередині системи;

– зовнішні. Джерела загроз знаходяться поза системою.

За розмірами нанесеного збитку:

– загальні. Нанесення збитку об'єкту безпеки в цілому, значна шкода;

– локальні. Заподіяння шкоди окремими частинами об’єкта безпеки;

– приватні. Заподіяння шкоди окремим властивостям елементів об’єкта безпеки.

За ступенем впливу на інформаційну систему:

– пасивні. Структура і зміст системи не змінюються;

– активні. Структура і зміст системи піддаються змінам.

За природою виникнення:

– природні (об’єктивні. Викликані впливом об’єктивних фізичних процесів або стихійних природних явищ, що не залежать від волі людини;

– штучні (суб’єктивні). Викликані впливом людини на інформаційну сферу. Серед штучних загроз виділяють:

– ненавмисні (випадкові). Помилки програмного забезпечення, персоналу, збої в роботі систем, відмови обчислювальної та комунікаційної техніки;

– навмисні (умисні). Неправомірний доступ до інформації, розробка спеціального програмного забезпечення для здійснення неправомірного доступу, розробка та поширення вірусних програм.

Згідно з дослідженнями фахівців з інформаційної безпеки, понад 65% шкоди, що наноситься інформаційним ресурсам, є наслідком ненавмисних помилок, що підкреслює важливість ефективного впровадження комп’ютерних систем для забезпечення безпеки [2].

Класифікація джерел загроз інформаційної безпеки. Носіями загроз безпеки інформації є джерела загроз, які можуть бути суб’єктами (особистість) або об’єктивними проявами (наприклад, конкуренти, злочинці, корупціонери, адміністративно-управлінські органи). Джерела загроз переслідують такі цілі: ознайомлення з охоронюваними відомостями, їх модифікація в корисливих цілях і знищення для нанесення прямого матеріального збитку.

Всі джерела загроз інформаційної безпеки можна поділити на три основні групи:

– обумовлені діями суб’єкта (антропогенні джерела): Суб’єкти, дії яких можуть призвести до порушення безпеки інформації. Ці дії можуть бути кваліфіковані як навмисні або випадкові злочини. Джерела можуть бути як зовнішніми, так і внутрішніми. Ці джерела можна спрогнозувати і прийняти адекватні заходи;

– обумовлені технічними засобами (техногенні джерела): Ці джерела загроз менш прогнозовані, залежать від властивостей техніки і потребують особливої уваги. Можуть бути як внутрішніми, так і зовнішніми;

– стихійні джерела: Обставини, що становлять непереборну силу (стихійні лиха або інші обставини, які неможливо передбачити або запобігти). Стихійні джерела зазвичай зовнішні і розуміються як природні катаклізми.

Несанкціонований доступ до інформації – це доступ до інформації з порушенням посадових повноважень співробітника, доступ до закритої для публічного доступу інформації з боку осіб, які не мають дозволу на доступ до цієї інформації. Це також може бути доступ до інформації особою, що має право на доступ до цієї інформації в обсязі, що перевищує необхідний для виконання службових обов'язків.

Причини несанкціонованого доступу до інформації:

– помилки конфігурації (прав доступу, брандмауерів, обмежень на масовість запитів до баз даних);

– слабка захищеність засобів авторизації (розкрадання паролів, смарт-карт; фізичний доступ до устаткування, що погано охороняється; доступ до незаблокованих робочих місць співробітників під час їх відсутності);

– помилки в програмному забезпеченні;

– зловживання службовими повноваженнями (викрадення резервних копій, копіювання інформації на зовнішні носії при праві доступу до інформації);

– прослуховування каналів зв'язку при використанні незахищених з'єднань всередині локальних обчислювальних мереж (ЛОМ);

– використання клавіатурних шпигунів, вірусів і троянів на комп'ютерах співробітників.

Типові кроки для отримання несанкціонованого доступу:

1) **збір інформації:**

– збір даних про IT профіль жертви, так званого цифрового сліду (англ. footprint), включаючи відкриті TCP та UDP порти, а також існуючі способи отримання доступу до комп'ютерної мережі жертви;

2) аналіз інформації:

– аналіз зібраної інформації, щоб визначити вразливі точки в системі;

3) отримання логінів і паролів:

– використання соціальної інженерії: хакер видає себе за співробітника IT відділу компанії, щоб отримати облікові дані користувача;

– доступ до паперового сміття жертви, щоб знайти записи з логінами та паролями;

– використання спеціальних програм для злому паролів, які використовують списки слів, фраз або інших комбінацій букв, цифр і символів.

4) отримання доступу до мережі:

– після доступу до комп'ютерної мережі жертви хакер намагається набути адміністраторських привілеїв, використовуючи трояни та пошук на файлових системах;

5) детальний аналіз системи:

– після отримання адміністраторських привілеїв хакер проводить детальний аналіз комп'ютерної системи жертви, отримуючи доступ до баз даних, вебсерверів та інших ресурсів;

6) організація бекдору:

– хакер створює нестандартні методи отримання доступу до мережі жертви (бекдор) для запобігання виявленню та блокуванню з боку ІТ спеціалістів компанії;

Модель порушника. Порушник – це особа, яка може одержати доступ до роботи з включеними до складу КС засобами. Порушники класифікуються за рівнем можливостей, які надаються їм штатними засобами КС. Класифікація є ієрархічною, тобто кожен наступний рівень включає в себе функціональні можливості попереднього:

1) перший рівень:

– найнижчий рівень можливостей проведення діалогу з КС. Можливість запуску фіксованого набору завдань (програм), що реалізують заздалегідь передбачені функції обробки інформації;

2) другий рівень:

– можливість створення і запуску власних програм з новими функціями обробки інформації;

3) третій рівень:

- можливість управління функціонуванням КС, тобто вплив на базове програмне забезпечення системи та склад і конфігурацію її устаткування;

4) четвертий рівень:

– повний обсяг можливостей осіб, що здійснюють проектування, реалізацію та ремонт апаратних компонентів КС, включаючи можливість включення до складу КС власних засобів з новими функціями обробки інформації.

Припускається, що порушник найвищого рівня – це фахівець вищої кваліфікації, який має повну інформацію про КС і комплекс захисту (КЗЗ). Така класифікація порушників корисна для оцінки ризиків, аналізу вразливості системи та ефективності існуючих і планованих заходів захисту.

У кожному конкретному випадку, виходячи з технології обробки інформації, необхідно розробити модель порушника, яка повинна бути адекватною реальному порушнику для даної автоматизованої системи (АС). Модель порушника – це абстрактний формалізований або неформалізований опис дій порушника, що відображає його практичні та теоретичні можливості, апріорні знання, час та місце дії тощо. Порушники можуть бути внутрішніми (співробітники, користувачі системи) або зовнішніми (сторонні особи або будь-які особи за межами контрольованої зони) [3].

Модель порушника повинна визначати:

– можливу мету порушника та її градацію за ступенями небезпечності для АС;

– категорії осіб, з числа яких може бути порушник;

– припущення про кваліфікацію порушника;

– припущення про характер його дій.

Метою порушника можуть бути:

– отримання необхідної інформації у потрібному обсязі та асортименті;

– можливість вносити зміни в інформаційні потоки відповідно до своїх намірів (інтересів, планів);

– нанесення збитків шляхом знищення матеріальних та інформаційних цінностей.

Рекомендується класифікувати порушників за рівнем можливостей, що надаються їм засобами АС. Класифікація є ієрархічною, тобто кожен наступний рівень включає в себе функціональні можливості попереднього:

1) перший рівень:

– найнижчий рівень можливостей ведення діалогу з АС. Можливість запуску фіксованого набору завдань (програм), що реалізують заздалегідь передбачені функції обробки інформації;

2) другий рівень:

– можливість створення і запуску власних програм з новими функціями обробки інформації;

3) третій рівень:

– можливість управління функціонуванням АС, вплив на базове програмне забезпечення системи та склад і конфігурацію її устаткування;

4) четвертий рівень:

– повний обсяг можливостей осіб, що здійснюють проектування, реалізацію, впровадження, супроводження програмно-апаратного забезпечення АС, включаючи можливість включення до складу АС власних засобів з новими функціями обробки інформації.

Порушників можна класифікувати як таких, що:

– володіють інформацією про функціональні особливості АС, основні закономірності формування в ній масивів даних та потоків запитів, вміють користуватися штатними засобами;

– володіють високим рівнем знань та досвідом роботи з технічними засобами системи та їх обслуговування;

– володіють високим рівнем знань у галузі обчислювальної техніки та програмування, проектування та експлуатації АС;

– володіють інформацією про функції та механізм дії засобів захисту.

Порушники можуть використовувати:

– виключно агентурні методи одержання відомостей;

– пасивні технічні засоби перехоплення інформаційних сигналів;

– виключно штатні засоби АС або недоліки проектування комплексної системи захисту інформації (КСЗІ) для реалізації спроб несанкціонованого доступу (НСД);

– способи і засоби активного впливу на АС, що змінюють конфігурацію системи (підключення додаткових або модифікація штатних технічних засобів, підключення до каналів передачі даних, впровадження і використання спеціального програмного забезпечення).

Порушники можуть класифікуватися за місцем здійснення дій як такі, що:

– не одержують доступу на контрольовану територію організації (АС);

– одержують доступ на контрольовану територію, але без доступу до технічних засобів АС;

– одержують доступ до робочих місць кінцевих (у тому числі віддалених) користувачів АС;

– одержують доступ до місць накопичення і зберігання даних (баз даних, архівів, АРМ відповідних адміністраторів тощо);

– одержують доступ до засобів адміністрування АС і засобів керування КСЗІ.

Сучасна інформаційна безпека перебуває під впливом декількох чинників:

– зростання кількості фінансово мотивованих зловмисників;

– діяльність хакерів, підтримуваних і спонсорованих урядами різних країн;

– зростання державного регуляторного тиску з метою захисту інформації та інформаційної інфраструктури;

– потреби бізнесу та держави в ІТ-рішеннях;

– правове регулювання, що відповідає сучасним викликам.

Таким чином, сучасна інформаційна безпека базується на розумінні основних явищ, термінів та концепцій інформаційної безпеки [5].

Рекомендована література

1. Andrushchak I.Ye, Martseniuk V.P., Androshchuk I.V., Chudovets V.V. Cloud computing and analysis features of cloud information security. Науковий журнал “Компютерно-інтегровані технології: Освіта, наука, виробництво”. Випуск №37, Луцьк. 2019. С. 5-10.

2. Майданюк В. П. Основи теорії інформації та кодування: електронний навчальний посібник комбінованого (локального та мережного) використання Вінниця: ВНТУ, 2022. 133 с.

3. Остроухов В. В. Інформаційна безпека. Підручник. К.: Видавництво Ліра-К, 2021. 412 с.

4. Євсеєв С. П. Технології захисту інформації. Мультимедійне інтерактивне електронне видання комбінованого використання. Х.: ХНЕУ ім. С. Кузнеця, 2022. 1013. Мб. ISBN 978-966-676-624-6

5. Martsenyuk V., Sverstyuk A., Andrushchak I., Chudovets V., Koshelyuk V. Aspects of protection of accounting data in the conditions of use of innovation and information technologies. Науковий журнал “Комп’ютерно-інтегровані технології: Освіта, наука, виробництво”. Випуск №42, Луцьк. 2021. С. 172-176.

4.3 Загальна характеристика шкідливого програмного забезпечення

Класифікація шкідливого програмного забезпечення

Шкідливе програмне забезпечення (ШПЗ) – це програмне забезпечення, яке за умови запуску може завдати шкоди пристрою різними способами. Воно може призвести до:

– блокування пристрою та його непридатності для використання;

– крадіжки, видалення або шифрування даних;

– використання пристрою для атак на інші пристрої;

– отримання кіберзловмисниками інформації щодо облікових даних, які дозволяють дістати доступ до систем або служб, якими ви користуєтесь.

– використання для незаконного майнінгу криптовалюти на вашому пристрої.

– використання платних послуг на основі ваших даних (наприклад, телефонні дзвінки на платні номери).

Серед найвідоміших видів ШПЗ:

– віруси – порушують нормальну роботу пристроїв, записуючи, пошкоджуючи або видаляючи дані. Поширюються на інші пристрої після того, як користувачі відкривають зловмисні файли.

– трояни – програми, які користувач ненавмисно завантажує під виглядом надійних файлів або програм. Після завантаження можуть завантажувати й інсталювати додаткове шкідливе програмне забезпечення, використовувати пристрій для шахрайства або записувати натискання клавіш.

– кейлогери – програми, що записують натискання клавіш, щоб викрасти паролі та іншу конфіденційну інформацію.

– програми-вимагачі – програми, які блокують доступ до пристрою або даних і вимагають викуп за їх відновлення.

– хробаки – самопоширювані програми, що використовують вразливості мереж і систем для розповсюдження.

– шпигунські програми – програми, які інсталюються на пристрій без згоди користувачів і збирають інформацію про їх діяльність.

– рекламне програмне забезпечення – програми, що показують агресивну рекламу, зазвичай у формі спливаючих оголошень.

– експлойти та набори експлойтів – використовують вразливості програмного забезпечення для розгортання шкідливого програмного забезпечення.

– безфайлове шкідливе програмне забезпечення – поширюється у вигляді зловмисних мережевих пакетів, які інсталюють шкідливе програмне забезпечення виключно в пам’ять ядра, використовуючи вразливості системи.

– руткіти – програми, які перехоплюють контроль над стандартними процесами операційної системи та можуть залишатися непомітними протягом тривалого часу.

Задачі захисту програмного забезпечення. Методи і засоби захисту програмного забезпечення

Основними задачами захисту програмного забезпечення є:

– забезпечення цілісності програмного забезпечення;

– захист від несанкціонованого доступу та модифікації;

– захист від несанкціонованого копіювання та використання.

Методи і засоби захисту програмного забезпечення:

– антивірусні програми – програмні рішення, що виявляють і видаляють шкідливе програмне забезпечення;

– брандмауери – захищають комп’ютери та мережі від несанкціонованого доступу;

– шифрування – захищає дані від несанкціонованого доступу, перетворюючи їх у формат, який може бути прочитаний лише за наявності спеціального ключа;

– багатофакторна автентифікація – використання кількох способів підтвердження особи для доступу до систем;

– регулярні оновлення – завантаження та інсталяція оновлень і доповнень до програмного забезпечення для усунення вразливостей;

– резервне копіювання – створення копій даних для їх відновлення у разі втрати або пошкодження.

Методи захисту програм від вивчення. Типова архітектура системи захисту програмного забезпечення

Методи захисту програм від вивчення включають:

– обфускація коду – ускладнення коду програми, щоб зробити його важкочитним і важкорозумілим для аналізу;

– використання ліцензійних ключів – захист програмного забезпечення від несанкціонованого використання;

– захист від дебаггінгу – використання технік, що ускладнюють аналіз програми з використанням налагоджувачів;

– використання апаратних ключів – використання спеціальних пристроїв, які повинні бути підключені до комп’ютера для запуску програми.

– Типова архітектура системи захисту програмного забезпечення включає:

– модуль автентифікації – перевіряє право доступу користувача до програми;

– модуль шифрування – захищає дані, що передаються та зберігаються;

– модуль обфускації – ускладнює аналіз коду програми;

– модуль захисту від копіювання – перевіряє ліцензійні ключі та обмежує несанкціоноване копіювання;

– модуль моніторингу – відстежує активність у програмі та виявляє спроби несанкціонованого доступу або модифікації.

Рекомендована література

1. Горбенко В. І. Безпека програм та даних: навчальний посібник для здобувачів ступеня вищої освіти бакалавра спеціальності 121 «Інженерія програмного забезпечення» освітньо-професійної програми «Програмна інженерія». Запоріжжя: ЗНУ, 2022. 372 с.
2. Демяненко В. А. Безпека програм та даних: навч. посібник. Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2021. 295 с.
3. Згуровський М. Проблеми інформаційної безпеки в Україні, шляхи їх вирішення. Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. Київ. 2019. 244 с.
4. Клінцев Л.М. Безпека програм і даних. Чернігов: ВСП Чернігівський інститут інформації, бізнесу і права, 2019. 482 с.
5. Мнушка О.В. Безпека програм і даних: конспект лекцій для студентів за спеціальністю 121 «Інженерія програмного забезпечення». Харків, ХНАДУ, 2020. 248 с.

4.4 Захист електронної пошти

Захист електронної пошти. З розвитком Інтернету електронна пошта стала невід’ємною частиною комунікацій на підприємствах та в установах. Вона дозволяє швидко пересилати повідомлення, документи, графічні, аудіо- та відеоматеріали. Надійність захисту даних у системі електронної пошти

впливає на загальний рівень інформаційної безпеки організації і, як наслідок, на ефективність її діяльності.

Електронна пошта є важливою частиною сучасного життя більшості людей. Проте безпека електронних листів залежить не лише від складності та унікальності пароля до облікового запису. Для захисту поштової скриньки необхідно забезпечити:

– захист вмісту повідомлення під час передачі;

– перевірку вмісту листів на наявність посилань на шкідливі сайти та спам;

– аутентифікацію та авторизацію для безпечного доступу до облікових записів;

– цілісність та функціональність поштової програми.

Основні загрози для електронної пошти. Електронна пошта – один з найпопулярніших методів онлайн-листування, особливо для підприємств. Велика частина шкідливого ПЗ потрапляє у мережі через вкладення електронної пошти. Без належних заходів безпеки електронна пошта може легко стати точкою входу для зловмисників.

Облікові записи корпоративної електронної пошти є основними цілями для кіберзлочинців, оскільки вони містять цінні бізнес-дані та дані про клієнтів. Впровадження передових методів захисту електронної пошти допомагає знизити ризик кібератак та запобігти використанню електронної пошти для доступу до ваших даних і систем.

Основні методи захисту електронної пошти:

1) **шифрування вмісту листів:**

– шифрування на транспортному рівні. Захищає повідомлення під час його передачі через Інтернет. Це схоже на вкладення листа в конверт: можна побачити, звідки та куди надсилається повідомлення, проте його зміст залишається прихованим;

– повне шифрування даних. Повідомлення шифрується у відправника та розшифровується у одержувача, забезпечуючи захист вмісту на всіх етапах передачі;

2) перевірка вмісту листів:

– перевірка на наявність шкідливих посилань та спаму;

– використання антивірусних програм для сканування вкладень;

3) аутентифікація та авторизація:

– використання багатофакторної автентифікації для доступу до облікових записів;

– перевірка прав доступу до інформації;

4) цілісність та функціональність поштової програми:

– регулярне оновлення програмного забезпечення;

– використання надійних поштових клієнтів з розширеними функціями безпеки.

Вибір провайдера електронної пошти. Першим кроком у налаштуванні облікового запису електронної пошти є вибір надійного провайдера. Популярні поштові провайдери, такі як Google та Microsoft, мають розширені функції безпеки, які захищають обліковий запис від кібератак. Важливо обрати провайдера, який відповідає вашим уподобанням щодо конфіденційності та безпеки.

Переваги захисту електронної пошти. Компанії всіх розмірів починають усвідомлювати важливість захисту електронної пошти. Рішення для захисту електронної пошти, яке забезпечує безпеку листування працівників і зменшує кількість кібератак, є надзвичайно важливим, оскільки воно пропонує такі переваги:

– хахист бренду, репутації та чистого прибутку компанії. Атаки на електронну пошту можуть призвести до значних матеріальних збитків, збоїв у роботі та інших тяжких наслідків;

– підвищення продуктивності. Надійне рішення для захисту електронної пошти мінімізує потенційні збої в роботі та зменшує час простоїв через кібератаки. Ефективне рішення допомагає командам безпеки швидко реагувати на дедалі складніші загрози та запобігати їм;

– забезпечення відповідності вимогам законів щодо захисту даних. Виконання вимог таких нормативних актів, як Генеральний регламент із захисту персональних даних (GDPR), допомагає уникнути зупинок у роботі компаній, судових витрат та нормативних штрафів.

Практичні поради щодо захисту електронної пошти. Для підтримання безпеки комунікацій і захисту від кібератак організації можуть скористатися такими порадами:

– регулярно проводьте тренінги. Навчайте працівників, як уникати помилок, що можуть призвести до кібератак. Працівники повинні розуміти важливість захисту електронної пошти, адже вони часто є першою лінією оборони;

– інвестуйте в тренінги з підвищення обізнаності. Допоможіть користувачам розпізнавати ознаки фішингових атак та інші індикатори зловмисних намірів;

– впровадьте багатофакторну автентифікацію (БФА). Використовуйте кілька методів входу в облікові записи, щоб запобігти порушенням безпеки;

– перевіряйте захист від атак на корпоративну електронну пошту. Використовуйте методи імітації та підробки для перевірки надійності захисту;

– перенесіть процеси й транзакції, пов'язані з високими ризиками, в автентифіковані системи.

Підвищення обізнаності про кібербезпеку. Обізнаність про кібербезпеку є важливим аспектом успішної стратегії кібербезпеки. Кожен співробітник, незалежно від стажу роботи, повинен пройти комплексне навчання з питань кібербезпеки. Незалежно від розміру компанії, вона може стати метою кібератак.

Поради щодо підвищення обізнаності:

– підкресліть важливість поділу ділової та особистої електронної пошти;

– відговорюйте співробітників перевіряти ділову електронну пошту зі своїх мобільних телефонів;

– заохочуйте співробітників регулярно оновлювати паролі електронної пошти;

– заохочуйте співробітників використовувати складні унікальні паролі.

Організації зазнають низки складних атак, серед яких:

– ексфільтрація даних. Несанкціоноване передавання даних за межі організації;

– шкідливе програмне забезпечення. Програми, створені зловмисниками, що шкодять комп'ютерам і системам;

– спам. Небажані повідомлення, що містять шкідливе програмне забезпечення;

– підробка. Маскування кіберзлочинців під надійну особу або організацію для виманювання грошей або даних;

– фішинг. Кіберзлочинці видають себе за надійну особу або організацію для обману жертв і розкриття делікатної інформації.

Служби захисту електронної пошти допомагають компаніям убезпечувати облікові записи й повідомлення від кіберзагроз. Нижче наведено можливості найпоширеніших служб захисту електронної пошти:

– сканування на наявність зловмисних посилань і вкладень;

– шифрування даних для захисту повідомлень від кіберзлочинців;

– керування вкладеними або вбудованими зображеннями та вмістом;

– спам-фільтри для відфільтровування небажаної пошти;

– системи автентифікації для перевірки справжності відправників.

Паролі є основною лінією захисту від кіберзлочинів. Впровадження політики паролів, що включає:

– регулярне скидання паролів;

– унікальні паролі, які не використовуються повторно;

– виключення загальних фраз або особистої інформації;

– паролі з мінімум восьми символів, що включають літери, цифри та символи;

– безпечне зберігання паролів за допомогою рішень для управління паролями.

Розробка плану кібербезпеки. Комплексний план кібербезпеки включає політики, керівні принципи та вимоги щодо використання технологій. Важливо враховувати загрози, що передаються через електронну пошту, та забезпечити відповідний захист.

Використання спам-фільтрів. Більшість провайдерів електронної пошти мають вбудовані спам-фільтри. Додатково захищайте поштову скриньку, позначаючи спам-листи вручну або створюючи власні фільтри.

Регулярне оновлення програмного забезпечення. Оновлення часто включають патчі безпеки, які усувають вразливості. Регулярно оновлюйте поштовий клієнт та операційну систему.

Використання антивірусного рішення. Антивірусне програмне забезпечення може сканувати файли та вебсайти, проактивно виявляючи загрози.

Шифрування електронної пошти. Шифрування електронної пошти захищає конфіденційні повідомлення. Використовуйте шифрування, надсилаючи важливу інформацію.

Впровадження рішень для захисту електронної пошти. Впровадження інструментів захисту електронної пошти допомагає виявляти цільові атаки та захищати облікові записи.

Комплексний підхід до захисту електронної пошти. Для надійного захисту використовуйте комплексний підхід, що зменшує потенційні уразливості та загрози.

Рекомендована література

1. Майданюк В. П. Основи теорії інформації та кодування: електронний навчальний посібник комбінованого (локального та мережного) використання Вінниця: ВНТУ, 2022. 133 с.

2. Остроухов В. В. Інформаційна безпека. Підручник К.: Видавництво Ліра К. 2021. 412 с.

3. Полторак В.П. Інформаційна безпека та захист даних в комп'ютерних технологіях і мережах. Навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2020. 278 с.

4. Stewart J.M., Kinsey D. Network security, firewalls, and VPNs. Burlington: Jones & Bartlett Learning, 2021. 482 p.

5. Wenliang Du. Computer Security: A Hands-on Approach, 2022. – 543 p.

4.5 Симетрична та асиметрична криптографія

Криптографія, або криптологія, виникла з необхідності захисту інформації від несанкціонованого доступу. Її історія бере свій початок ще в давнину. Наприклад, стародавні єгиптяни використовували нестандартні ієрогліфи для захисту важливих текстів ще у 1900 році до н.е. У Київській Русі теж використовували різні методи шифрування в 12-13 століттях. Проте справжній розквіт криптографії в Західній Європі припав на пізнє середньовіччя, коли з'явилися перші роботи та настанови з кодування та шифрування документів.

Симетрична криптографія, або шифрування з секретним ключем, є одним з найстаріших методів шифрування. У цьому методі використовується один і той самий ключ як для шифрування, так і для дешифрування даних. Один з найвідоміших симетричних алгоритмів - Data Encryption Standard (DES), який широко застосовувався з 1970-х років у банківських та комерційних сферах. На початку 21 століття його замінив Advanced Encryption Standard (AES).

Крім симетричного шифрування, існують комбіновані методи, які використовують як симетричні, так і асиметричні алгоритми для забезпечення більшої безпеки даних. Посимвольне кодування інформації передбачає шифрування кожного символу окремо, тоді як кодування за змістом базується на шифруванні блоків даних або навіть цілих повідомлень.

Асиметрична криптографія, або шифрування з відкритим ключем, використовує пару ключів: відкритий ключ для шифрування і закритий ключ для дешифрування. Принцип асиметричної криптографії полягає в тому, що навіть якщо відкритий ключ відомий всім, без закритого ключа розшифрувати повідомлення неможливо. Це забезпечує високу безпеку при передачі даних через незахищені канали.

Відкритий ключ використовується для шифрування даних і є загальнодоступним. Закритий ключ, який використовується для дешифрування, зберігається в таємниці. Інфраструктура відкритого ключа (PKI) включає в себе набір ролей, політик та процедур, необхідних для створення, управління, розповсюдження, використання, зберігання та відкликання цифрових сертифікатів і управління криптографічними ключами.

Цифровий підпис є одним із важливих застосувань асиметричної криптографії. Він забезпечує аутентифікацію відправника та цілісність повідомлення. Підпис створюється за допомогою закритого ключа відправника і перевіряється за допомогою його відкритого ключа. Це дозволяє отримувачу переконатися, що повідомлення дійсно надіслане заявленим відправником і не було змінене під час передачі.

Використання криптографії у сучасному світі. Прогрес у комп’ютерних технологіях зробив дані більш доступними, але водночас підвищив ризики крадіжок і пошкоджень. Криптографія забезпечує захист інформації від несанкціонованого доступу, використовуючи математичні методи для шифрування та розшифрування даних. Вона є ключовим елементом безпеки в таких сферах, як банківська справа, електронна комерція, захист персональних даних, а також у криптовалютних мережах, де використовується для забезпечення безпеки транзакцій у децентралізованих системах.

Сучасна криптографія базується на використанні складних математичних алгоритмів і ключів для перетворення відкритого тексту в зашифрований, який може бути переданий через незахищені канали. Для відтворення відкритого тексту отримувач використовує свій закритий ключ. Це забезпечує конфіденційність, цілісність і аутентифікацію інформації.

Розуміння криптографії є важливим для розуміння безпеки даних у сучасних цифрових системах, де вона використовується для захисту від різних загроз і забезпечення надійного обміну інформацією.

Перед тим як розглянути класифікацію криптографічних систем, варто зазначити, що їхній розвиток можна поділити на три основні етапи.

Етап донаукової криптології (до 1949 року). У цей період криптографія розвивалася як мистецтво, а не наука. Шифрування здійснювалося за допомогою простих методів, таких як заміна та перестановка символів у відкритому тексті. Серед відомих прикладів таких шифрів — шифр Цезаря, шифр «скітала», шифруючі таблиці, квадрат Полібія та шифр Віжінера.

Етап наукової криптології із секретними ключами (з 1949 до 1970-х років). У цей період криптографія почала розвиватися як наука. Основним методом шифрування був симетричний шифр, де для шифрування та дешифрування використовувався один і той самий секретний ключ. Основні принципи шифрування на цьому етапі включали розсіювання (diffusion) та перемішування (confusion). З'явилися блокові та потокові шифри.

Етап наукової криптології з використанням ЕОМ (з 1970-х років до теперішнього часу). Прогрес у галузі обчислювальної техніки значно збільшив можливості як криптографів, так і криптоаналітиків. З'явилися більш складні алгоритми шифрування, які стали можливими завдяки використанню електронно-обчислювальних машин (ЕОМ). Перші дослідження в цій галузі почалися під час Другої світової війни, зокрема, з робіт Алана Тюрінга над створенням машини «Колос», яка могла зламувати шифри німецької машини «Енігма».

Симетричні криптографічні системи

Блокові шифри. У блокових шифрах відкритий текст розбивається на блоки фіксованої довжини, і кожен блок шифрується окремо за допомогою одного і того ж алгоритму. Приклади:

– DES (Data Encryption Standard). Використовує 56-бітовий ключ і шифрує дані в блоках по 64 біти;

– AES (Advanced Encryption Standard). Відомий також як Rijndael, підтримує ключі розміром 128, 192 і 256 біт, шифрує дані в блоках по 128 біт.

Потокові шифри. У потокових шифрах дані шифруються побітно або побайтово. Потокові шифри генерують псевдовипадковий потік (keystream), який комбінується з відкритим текстом за допомогою операції XOR. Приклади:

– RC4. Один з найвідоміших потокових шифрів, який використовувався в багатьох протоколах, таких як SSL та WEP.

Асиметричні криптографічні системи

Алгоритми з відкритим ключем. Використовують пару ключів: відкритий ключ для шифрування та закритий ключ для дешифрування.

Асиметричні алгоритми вирішують проблему передачі ключів у симетричних системах. Приклади:

– RSA (Rivest-Shamir-Adleman). Один з перших і найвідоміших асиметричних алгоритмів, базується на складності факторизації великих чисел.

– DSA (Digital Signature Algorithm). Стандарт для цифрового підпису, базується на математичних проблемах, таких як дискретне логарифмування.

Еліптична криптографія. Базується на властивостях еліптичних кривих і використовується для створення швидших і більш ефективних алгоритмів шифрування та цифрового підпису. Приклади:

– ECDSA (Elliptic Curve Digital Signature Algorithm). Використовується для цифрових підписів, зокрема у стандартах як SSL/TLS;

– ECC (Elliptic Curve Cryptography). Загальний підхід, який включає алгоритми для шифрування, генерації ключів та цифрового підпису.

Комбіновані методи шифрування. Комбіновані методи шифрування використовують як симетричні, так і асиметричні алгоритми для досягнення високої безпеки даних. Наприклад, у багатьох системах для передачі ключа використовується асиметричний алгоритм, а для шифрування основного обсягу даних - симетричний. Це дозволяє поєднувати швидкість симетричних шифрів з безпекою асиметричних.

Відкриті та закриті ключі:

– відкритий ключ використовується для шифрування даних або перевірки цифрового підпису. Є загальнодоступним;

– закритий ключ використовується для дешифрування даних або створення цифрового підпису. Зберігається в таємниці.

Інфраструктура відкритого ключа (PKI). PKI включає в себе набір ролей, політик, процедур і технологій, необхідних для створення, управління, розповсюдження, використання, зберігання та відкликання цифрових сертифікатів і управління криптографічними ключами. Це забезпечує надійний механізм для аутентифікації особистостей та захисту даних у цифрових комунікаціях.

Поняття цифрового підпису. Цифровий підпис забезпечує аутентифікацію відправника та цілісність повідомлення. Створюється за допомогою закритого ключа відправника і перевіряється за допомогою його відкритого ключа. Це дозволяє отримувачу переконатися, що повідомлення дійсно надіслане заявленим відправником і не було змінене під час передачі.

Криптографія, таким чином, є важливою наукою для забезпечення безпеки в цифровому світі, дозволяючи захищати дані від несанкціонованого доступу та зловмисних дій.

Як працює криптографія?

Сучасна криптографія складається з різних областей дослідження, включаючи симетричне та асиметричне шифрування, хеш-функції і цифрові підписи. Протокол Bitcoin, наприклад, використовує криптографічні докази для захисту мережі й забезпечення дійсності кожної транзакції, зокрема, за допомогою цифрових підписів та функції Hashcash, що є основою механізму консенсусу Proof of Work.

Криптографія є важливою частиною блокчейн-технології і, отже, має вирішальне значення для будь-якої криптовалюти. Криптографічні докази, застосовані до розподілених мереж, дозволили створити економічні системи без потреби в довірі, що дало початок Bitcoin та іншим децентралізованим цифровим валютам.

Електронний цифровий підпис (ЕЦП). ЕЦП – це цифровий засіб підтвердження автентичності та цілісності електронних документів. Підпис створюється за допомогою спеціального алгоритму, який використовує електронний ключ (публічний та приватний).

Види підпису:

1) **електронний цифровий підпис фізичної особи**. Використовується фізичними особами для підтвердження їхньої автентичності в онлайн-середовищі;

2) **ключ електронного цифрового підпису**. Публічний ключ використовується для перевірки підпису, приватний ключ - для його створення.

Застосування ЕЦП:

- бізнесові транзакції;
- електронна пошта;
- онлайн-банкінг.

Переваги використання ЕЦП:

- забезпечує автентичність;
- забезпечує цілісність;
- зручність і ефективність;
- зменшення витрат;
- віддалений доступ.

Недоліки використання ЕЦП:

- потребує інфраструктури;
- вартість;
- проблеми з безпекою;
- спрощеність підробки;
- юридичні питання.

Таким чином, використання ЕЦП має багато переваг, особливо в сучасному цифровому світі, але важливо бути обережним і впевнитися, що правильно забезпечено безпеку інфраструктури та ключів ЕЦП.

Рекомендована література

1. Гніліцький В.В., Орєхов Є.Г. Захист інформації. Навчальний посібник для студентів технічних спеціальностей. Житомир: 2021. 164 с.
2. Гребенюк А. М. Основи управління інформаційною безпекою. Навч. посібник. Дніпро: Дніпроп. держ. університет внутрішніх справ, 2020. 144 с.
3. Клінцев Л.М. Безпека програм і даних. Чернігів: ВСП Чернігівський інститут інформації, бізнесу і права, 2019. 482 с.
4. Кузнецов О. О. Захист інформації в інформаційних системах. Навчальний посібник. Х.: ХНЕУ, 2020. 510 с.
5. Остроухов В. В. Інформаційна безпека. Підручник. К.: Видавництво Ліра-К, 2021. 412 с.

Питання для самоконтролю

1. У чому полягає проблема захисту інформації в комп’ютерних системах?
2. Охарактеризуйте нормативне забезпечення системи технічного захисту інформації в Україні.
3. Порівняйте основні технології захисту інформації в комп’ютерних системах.
4. Назвіть і порівняйте основні тенденції проблеми захисту інформації в комп’ютерних системах.
5. Що таке: законодавчий, організаційний та технічний аспекти проблеми захисту інформації в комп’ютерних системах?

6. Яка наука, займається проблемою захисту інформації шляхом її перетворення?

7. Що таке електронний цифровий підпис?

8. В чому полягає криптографічний алгоритм?

9. Скільки використовується ключів в асиметричних криптосхемах для шифрування і дешифрування?

10. Як називається система умовних знаків для таємного письма, яка читається за допомогою ключа?

11. В чому полягають принципи створення комплексної системи захисту інформації?

12. Охарактеризуйте загрози конфіденційності, цілісності, доступності

13. Що таке комплекс засобів захисту і об'єкти комп'ютерної системи?

14. Дайте визначення поняття несанкціонованиого доступу.

15. На конкретному прикладі опишіть «Модель порушника».

16. Що належить до шкідливого програмного забезпечення?

17. Як розповсюджується шкідливе програмне забезпечення?

18. Як розпізнати шкідливе програмне забезпечення?

19. Як запобігти зараженню шкідливим програмним забезпеченням?

20. Якого типу буває шкідливе програмне забезпечення?

21. В чому полягає технологія захисту електронної пошти?

22. Назвіть та охарактеризуйте переваги та недоліки захисту електронної пошти.

23. Охарактеризуйте покрокову розробку плану кібербезпеки.

24. Порівняйте основні протоколи безпеки, які використовуються для доставки електронної пошти: SSL і TLS. В Чому їх відмінність?

25. Що таке багаторівневий захист електронної пошти?

Тестові завдання

1. Загроза, яка полягає в тому, що інформація стає відомою тому, хто не володіє повноваженнями доступу до неї:

a) загроза доступності;
b) загроза цілісності;
c) загроза конфіденційності;

d) загроза автентичності.

2. Загрози, пов’язані з імовірністю модифікації тієї чи іншої інформації, що зберігається в інформаційній системі:
a) загрози доступності;
b) загрози цілісності;
c) загрози конфіденційності;
d) загрози автентичності.

3. До яких загроз належить розповсюдження конфіденційної інформації клієнтів працівником організації?
a) внутрішніх;
b) зовнішніх;
c) системних;
d) змішаних.

4. Сукупність умов і факторів, що створюють небезпеку порушення інформаційної безпеки.
a) загроза доступності;
b) загроза інформаційної безпеки;
c) загроза безпеки особистості;
d) загроза автентичності.

5. Які загрози завдають більшість шкоди, що наноситься інформаційним ресурсам?
a) навмисні;
b) ненавмисні.

6. Шифрування з симетричними ключами – це:
a) схема шифрування, у якій ключ шифрування відсутній;
b) схема шифрування з відкритим ключем;
c) схема шифрування, у якій ключ визначити неможливо;
d) схема шифрування, у якій ключ шифрування та ключ дешифрування збігаються.

7. Криптостійкість – це:

a) характеристика шрифту, що визначає його стійкість до дешифрування без знання ключа;
b) властивість гами;
c) всі відповіді вірні;
d) характеристика шрифту, яки визначає його спосіб кодування.

8. Що потрібно для відновлення зашифрованого тексту:
a) вектор;
b) ключ;
c) матриця.

9. Вкажіть основні методи шифрування:
a) асиметричне;
b) знакове;
c) симетричне;
d) логічне.

10. До шкідливого ПЗ належать:
a) троянські програми;
b) класичні файлові віруси;
c) макровіруси;
d) хакерські утиліти;
e) мережеві хробаки;
f) завантажувальні.

11. Загрозами інформаційній безпеці можуть бути:
a) спам;
b) фішинг;
c) комп’ютерні віруси;
d) дії неавторизованих користувачів;
e) «природні» загрози;
f) дії хакерів;
g) дії авторизованих користувачів;
h) природні віруси.

12. Спам – це:

a) небажані електронні листи;
b) електронні листи, в яких не дотримуються правил електронного етикету;
c) телефонні дзвінки, текстові повідомлення, що надходять без згоди користувача.

13. Фішинг – це:
a) вид атаки, кінцевим результатом якої є викрадення коштів користувача;
b) безкоштовна допомога користувачам користуватися послугами інформаційної системи за допомогою ІКТ;
c) інтернет-шахрайство.

14. Комп'ютерні віруси – це:
a) програми або фрагменти програм, здатні розмножувати, розповсюджуватися в комп'ютерній системі псуючи, спотворюючи файли, папки та ін;
b) програми або фрагменти програм, які потрапляють в комп'ютер, але ніякої шкоди не приносять;
c) драйвери.

15. Яких правил потрібно дотримуватися для захисту своєї електроної поштової скриньки?
a) надійний пароль;
b) двоетапна авторизація;
c) для різних акаунтів використовувати завжди один і той самий пароль;
d) створювати пароль, який легко запам'ятається;
e) тримати пароль в таємниці.

16. Пароль поштової скриньки використовується для:
a) захисту інформації;
b) відмінності від інших поштових скриньок;
c) щоб чужі люди не змогли надіслати вам повідомлення;
d) щоб не забути дату народження чи інші свої дані.

17. Що з перерахованого нижче гарантує високу безпеку використання електронної пошти?
a) завжди використовувати HTTP;

b) загальний обліковий запис електронної пошти і збереження повідомлень тільки в чернетках;
c) використання анонімного акаунта електронної пошти;
d) використання HTTPS.

18. Як захистити таємницю листування в Інтернеті?
a) використовувати програму шифрування;
b) налагодити зв'язки з інтернет-провайдером;
c) відмовитися від використання електронної пошти;
d) використовувати анонімний обліковий запис.

19. Як називається термін, коли хтось імітує вашу адресу електронної пошти або один з ваших контактів?
a) виявлення (кров'янисті виділення);
b) обман (лежачи);
c) підміна (спуфінг);
d) злам (злом).

РОЗДІЛ 5
УПРАВЛІННЯ ІТ-ПРОЄКТАМИ?

5.1 Що таке управління ІТ-проєктами?

Перш ніж приступити до основ, нам потрібно визначити, що саме таке «проєкт».

Інститут управління проєктами (PMI) визначає «проєкт» як «тимчасове зусилля, спрямоване на створення унікального продукту, послуги або результату» [1].

У цьому визначенні слід звернути увагу на кілька ключових речей.

Слово «тимчасовий» означає, що проєкти повинні мати визначений початок і кінець. Це означає, що кожен проєкт має включати графік, масштаб і ресурси. Той факт, що він є тимчасовим із початком і кінцем, також означає, що він не є частиною поточних операцій. Це підводить нас до другого пункту.

Метою проєкту має бути «створення унікального продукту, послуги або результату». Це означає, що проєкт буде розпочато для досягнення конкретної мети, яка зазвичай виходить за межі звичайної повсякденної бізнес-операції. Це означає, що команда проєкту може включати людей, які зазвичай не працюють разом і потребують ресурсів, які зазвичай виходять за рамки повсякденних операцій.

Кожен проєкт повинен мати наступні компоненти:

– мета. Чого ви прагнете досягти?
– хронологія. Коли ви намагаєтеся цього досягти?
– бюджет. Скільки буде коштувати досягнення?
– зацікавлені сторони. Хто основні гравці зацікавлені в цьому проєкті?
– керівник проєкту. Хто подбає про те, щоб усе, що потрібно було виконати, було виконано?

Проєкт – це не щось рутинне. Повсякденні операції та технічне обслуговування не вважаються проєктами, оскільки вони не мають остаточного початку та кінця.

Управління ІТ-проєктами – це процес планування, організації, виконання та управління проєктами в галузі інформаційних технологій. Часто це вимагає поєднання людей, ресурсів, навичок і досвіду, щоб об’єднати зусилля для реалізації успішного проєкту [2].

По суті, управління ІТ-проєктами не відрізняється від будь-якого іншого типу управління проєктами. ІТ-проєкти все ще вимагають планування,

контролю, командної роботи та лідерства – і для цього організації використовують системи управління IT-проєктами , щоб досягти максимального успіху.

Ось кілька прикладів поширених IT-проєктів:

- перенесення IT-систем на нові сервери;
- створення веб-сайтів або програмних систем;
- створення мобільних, планшетних або веб-додатків;
- розгортання нової IT-інфраструктури, такої як мережі та обладнання;
- оновлення або міграція систем баз даних;
- впровадження нових IT-процесів, таких як аварійне відновлення.

Це далеко не вичерпний список. Як правило, якщо вам потрібно змінити будь-яку частину вашої мережі чи обладнання, швидше за все, вам потрібно запустити IT-проєкт.

5.2 Що робить менеджер IT-проєктів? Чим він унікальний?

Керівникам IT-проєктів потрібні такі ж навички та досвід, як і іншим спеціалізованим PM-менеджерам, наприклад, керівникам будівельних або технічних проєктів .

Однак, оскільки IT-проєкти часто потребують глибшого розуміння технологій, багато IT-менеджерів мають досвід розробки програмного забезпечення або IT-підтримки. Це допомагає їм зрозуміти складність IT, краще взаємодіяти зі своїми зацікавленими сторонами та вирішувати технічні проблеми.

Щоб зрозуміти унікальність світу IT-менеджера проєктів, розглянемо порівняльну характеристику **традиційного менеджера проєкту та PM IT-проєктів (**табл. 5.1).

Таблиця 5.1 – Порівняльна **характеристика** традиційного менеджера проєкту та менеджера IT-проєктів [3]

Традиційний менеджер проєкту	**Менеджер IT-проєктів**
Очолює та керує групою експертів у відповідній галузі	**Керує** такими експертами, як розробники програмного забезпечення, бізнес-аналітики, власники продуктів або менеджери інфраструктури
Визначає обсяг і результати проєкту та керує ними	**Визначає** елементи невиконаних завдань або історії користувачів, які розбиваються на спринти

Продовження таблиці 5.1

Відстежує роботу, виконану командою, і допомагає вирішити будь-які проблеми, що виникають	Відстежує роботу, виконану командою, і допомагає вирішити будь-які проблеми, що виникають, працюючи з розробниками, тестувальниками або власниками продуктів під час церемоній, таких як щоденні стенд-апи, демонстрації продуктів або планування спринту
Працює на спонсора проєкту, який визначає бачення та переваги проєкту	Працює на спонсора проєкту, який визначає бачення та переваги проєкту: але спонсор, ймовірно, буде старшим IT-менеджером або навіть технічним директором/IT-директором
Керує визначеним бюджетом проєкту	**Керує** певними видами витрат, включаючи години розробки, ліцензії на програмне забезпечення та елементи обладнання
Визначає та зменшує ризики для проєкту	Стикаються з загрозою помилок у розробці, невдалого розгортання та простою системи
Працює з членами команди для оцінки та підписання стандартів якості	**Працює** над деякими з найбільш жорстко контрольованих проєктів, допомагаючи команді в перевірці коду, інтеграційному тестуванні та перевірці прийнятності користувачами

5.3 Життєвий цикл IT-проєктів

Ключем до якісного управління проєктами є структура, яка не відрізняється від управління IT-проєктами. Найкращий спосіб створити таку структуру – дотримуватися життєвого циклу проєкту, який розбивається на чіткі фази (рис. 5.1).

Наведемо типові етапи, управління IT-проєктами (табл. 5.2).

Таблиця 5.2 – Фази життєвого циклу проєкту [4]

Фаза життєвого циклу проєкту	Цілі	Ресурси та результати
Ініціація	Визначаються цілі та результати проєкту Розкриваються потенційні ризики проєкту, залежності та обмеження Встановлюється приблизний обсяг роботи Отримується підписка від зацікавлених сторін	Односторінковий шаблон документа вимог до продукту (PRD)

Продовження таблиці 5.2

Планування	Створюється графік проєкту та встановлюються пріоритети Розбиваються віхи на практичні завдання Визначаються показники успіху та короткострокові цілі Вирішуються потенційні проблеми та можливості зменшення ризиків	Шаблон проєктної пропозиції Шаблон обсягу робіт Посібник і шаблон діаграми Ганта План управління ризиками
Виконання та моніторинг	Починається із стартової зустрічі проєкту Відстежується та контролюється прогрес за допомогою регулярних оглядів або церемоній Agile Коригується завдання, цілі та часові рамки на основі відгуків Здійснюється спілкування із зацікавленими сторонами та своєю командою	Порядок денний стартових зустрічей проєкту та посібник Шаблон плану спілкування Шаблони та посібник із планування спринту
Закриття	Перевіряються кінцеві результати відповідно до визначення готовності та передачі Проводиться ретроспективна зустріч проєкту (і надається відгук клієнта) Документуються отримані уроки Звільняється проєктну команду та ресурси	Визначення виконаного контрольного списку Ретроспективний довідник спринту та шаблони Шаблон отриманих уроків

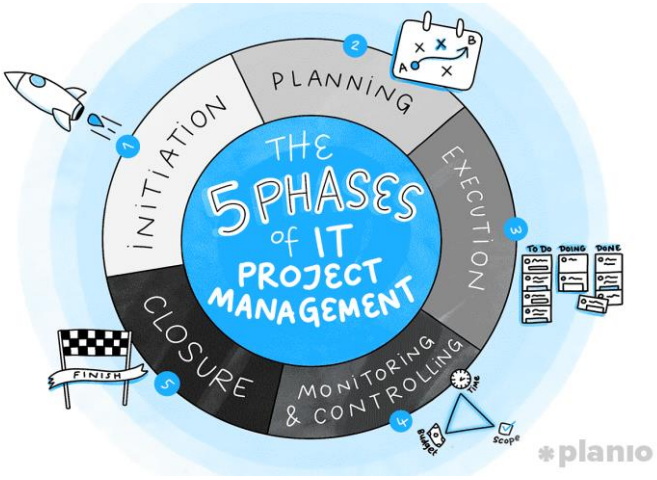

Рисунок 5.1 – 5 фаз ЖЦ ІТ-проєкту [3]

Ініціація. Кожен проєкт починається з розуміння потреб бізнесу, того, як новий продукт або функція вирішить їх, а також ризиків, проблем і пріоритетів, з якими можна зіткнутися на цьому шляху.

У життєвому циклі проєкту ця фаза називається «Initiation».

Замість того, щоб відразу переходити до детального плану, потрібно витратити час на дослідження, відкриття та обговорення. Це може означати розмову із зацікавленими сторонами, вивчення сподівання та мотивації клієнта.

Незалежно від ситуації, етап ініціації включає кілька ключових кроків і результатів:

1) визначити цілі та результати проєкту;
2) розкрити ризики, залежності та обмеження проєкту;
3) почати встановлювати приблизний обсяг роботи;
4) отримати згоду від зацікавлених сторін.

Розберемося в кожному детальніше.

Визначення цілей та результатів проєкту. Найважливішим питання, на яке можна відповісти перед тим, як почати будь-яку роботу, це: навіщо цей проєкт має відбуватися?

Якщо у виникають проблеми з відповіддю на це запитання, потрібно скористатися однією з цих підказок:

Якого результату ви хочете досягти?

Яку поведінку користувачів ви хочете змінити?

Який ваш «показник Полярної зірки» скаже вам, чи буде цей проєкт успішним чи ні?

Чудовим місцем для початку визначення цілей проєкту є ваша продуктова стратегія (рис. 5.2).

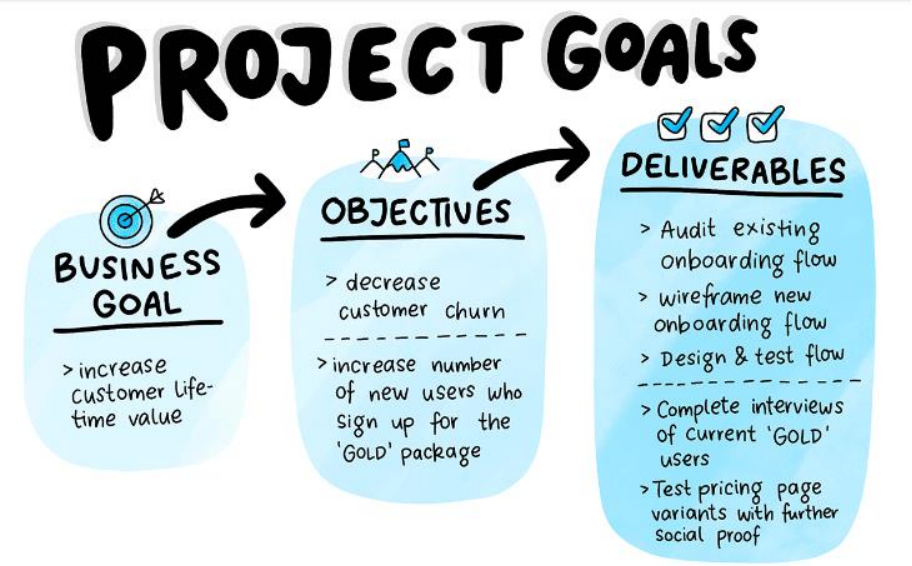

Рисунок 5.2 – Основні цілі продуктової стратегії [3]

Продуктова стратегія виходить за рамки конкретних функцій і пояснює загальне бачення, ідеального клієнта та ринкові можливості.

З цього моменту можна почати відкривати конкретні можливості зростання.

Ваші ідеальні користувачі не знаходять вас? Або вони стають тому, що ваші конкуренти мають кращі функції? Чи зупинився ваш ріст протягом минулого року?

Усе це є реальними цілями для початку планування проєкту.

Якщо є агентство, яке працює із зовнішнім клієнтом, то отримує ту саму інформацію, дослідивши їхні потреби, поточні проблеми та історію.

Що вони робили в минулому, що спрацювало чи ні? Де вони борються чи бачать простір для зростання?

Коли визначите кілька цілей, тоді зможете перетворити їх у конкретні результати (тобто те , що збираєтеся створити, щоб отримати результати, яких прагнете).

Задокументуйте ці результати та зберігайте записи про те, чому вони важливі. Коли працюєте над життєвим циклом проєкту, дуже важливо повернутися до своєї початкової мети.

Визначивши ці високорівневі цілі та результати, настав час подумати про ресурси, які є у розпорядженні.

Кожен проєкт має певні обмеження та ризики, які слід враховувати перед тим, як взятися за будь-яку роботу. У більшості випадків це означає подумати про:

1) час, бюджет і команду: те, що можна створити, буде обмежено доступним часом, бюджетом і наявною у командою. Більшість людей називають це класичним «потрійним обмеженням» управління проєктами;

2) ризики проєкту: кожному проєкту властиві ризики, які можуть негативно вплинути на цілі або графік. Близько 30 % усіх проєктів зазнають невдачі через непередбачені або невизначені ризики;

3) залежності: чи залежать ці результати від попереднього завершення інших проєктів? Які рухомі частини можуть стати на заваді досягненню цілей?

Інші пріоритети. Пріоритезація функцій – це балансування. Чи є інші роботи чи виправлення помилок, які зараз мають бути пріоритетними?

Наразі потрібно тримати цю частину на високому рівні та використовувати її, щоб допомогти визначити обсяг роботи. Пізніше буде створено повний план управління ризиками .

Тепер, коли відомо, що потрібно завершити, а також обмеження та ризики, які можуть стати на заваді, можна почати визначати обсяг роботи.

Виходячи з наявних у ресурсів, почніть встановлювати межі обсягу проєкту.

– Що можна завершити за цей термін ?
– Які ресурси знадобляться для його завершення?
– Яке визначення зробленого?
– Що не буде включено в цей проєкт?

Потрібно використовувати цю інформацію, щоб почати складати PRD – документ вимог до продукції. На відміну від більш поглибленого плану проєкту, PRD – це детальна розбивка мети, особливостей, функцій і поведінки продукту, який потрібно створити.

Нарешті настав час отримати відгук щодо створеної пропозиції. Можна використовувати свій PRD як путівник, щоб показати зацікавленим сторонам проєкту, що пропонується, чому це важливо та як це буде створено.

Варто отримати підпис на план високого рівня, перш, ніж витрачати час або енергію на детальний план.

Планування. Після налаштування проєкту, настає час вникнути в деталі, спланувавши, як дістатися від А до Б. На етапі планування IT-менеджери проєктів працюють із зацікавленими сторонами, щоб охопити вимоги до проєкту/продукту в інструменті керування проєктами, наприклад, Planio, намітити графік роботи та підтвердити кошторис (рис. 5.3). Наприкінці етапу планування отримуємо детальний план проєкту та управління ризиками, а

також буде створено структуру управління між IT-менеджером, командою та спонсором.

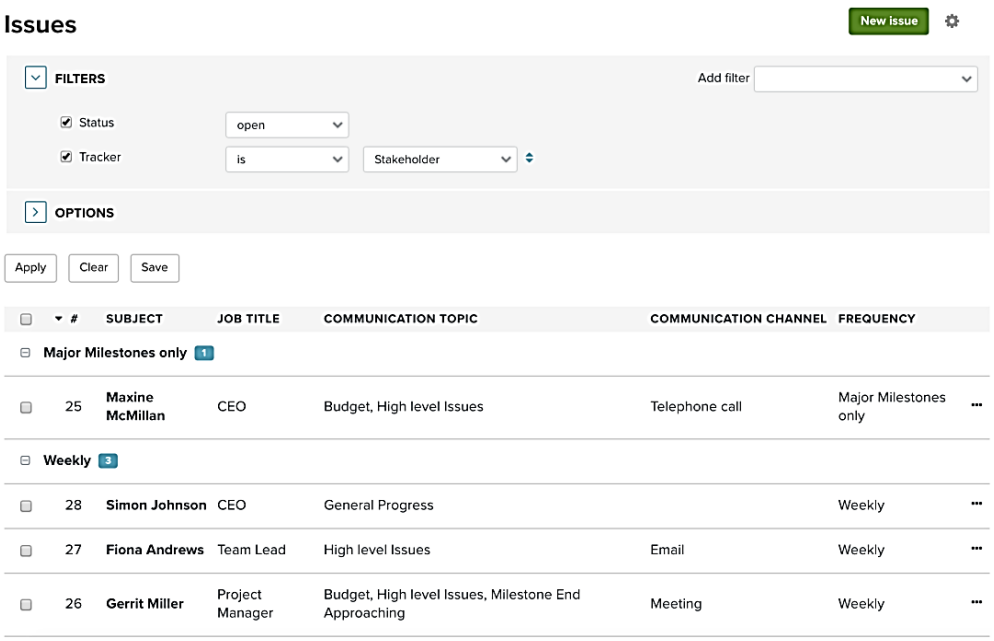

Рисунок 5.3 – Інструмент керування проєктами Planio [3]

На етапі планування життєвого циклу проєкту створюється детальний план, який стане основою протягом всього проєкту. Як правило, для цього потрібно:

– створити графік проєкту та встановити пріоритети;
– розбити віхи на практичні завдання;
– визначити показники успіху та короткострокові цілі;
– вирішіть потенційні проблеми та зменшити будь-які ризики.

Виконання такого обсягу роботи наперед може лякати. Однак важливість плану проєкту не можна недооцінювати. Коротко опишемо, як створити план проєкту, який буде тримати команду на правильному шляху.

Керівник проєкту встановлює реалістичний графік і дорожню карту для команди.

Це, мабуть, найважливіша частина життєвого циклу проєкту, оскільки створюються часові обмеження для кожного результату, які потім можна використовувати для відстеження прогресу та прийняття рішень щодо будь-яких змін обсягу.

Почати потрібно із приблизних дат: запропонованої дати початку та завершення проєкту, а також коли потрібно завершити конкретні етапи чи завдання.

Якщо команда використовує традиційний стиль управління проєктом (він же «водоспад»), можна візуалізувати цю часову шкалу на діаграмі Ганта .

Діаграми Ганта показують часову шкалу кожного завдання, накладеного одне на одне, щоб ви могли побачити залежності або визначити, де члени команди можуть бути перевантажені роботою.

Інструмент керування проєктами, наприклад, Planio, може автоматично створювати діаграми Ганта на основі дат початку та завершення, які встановлено для кожного завдання.

Однак, якщо робота є в команді Agile, потрібно планувати всю роботу над проєктом одразу.

Хронологія будується навколо більших віх – набору завдань або проблем, які рухатимуть проєкт вперед. Однак у плані проєкту потрібно детальніше розповісти, що включає кожна віха.

Процес збору цих деталей і розбивки більших етапів називається керуванням завданнями . Наведемо кілька способів, якими можна почати організовувати та розбивати свої етапи на завдання:

1) почати з кінця і працювати назад: потрібно уявити свій закінчений проєкт і повернуться до початку. Варто написати кожен крок, який можна придумати, на дошці або розумовій карті, а потім почати будувати шлях до початку;

2) варто залучити команду, щоб побачити, що, можливо, пропустили. Потрібно показати команді великий список і попросити їх заповнити будь-які прогалини або додати контекст;

3) варто розбити багатоетапні завдання на окремі кроки. Потрібно шукати будь-які великі завдання (так звані «міні-віхи») і розбити їх на окремі кроки. Наприклад, не потрібно писати «побудувати потік адаптації». Замість цього варто вказати кожен крок, який потрібно зробити;

4) далі запитати експерта. Якщо робота є над абсолютно новим проєктом, то команда може пропустити важливі завдання чи кроки. Варто запросити когось більш досвідченого, щоб переглянути список.

Нарешті, коли ви заповнюєте свій список завдань, ви повинні переконатися, що кожне питання містить будь-яку та всю відповідну інформацію та контекст.

Будь-який член команди повинен мати можливість дивитися на завдання і знати, що це таке, наскільки воно важливе, хто відповідальний і, хто над ним працює.

Наприклад, у Planio кожне завдання містить місце для детальної інформації, включаючи приблизний час, пріоритет завдань і навіть файли та посилання на сховища коду (рис. 5.4).

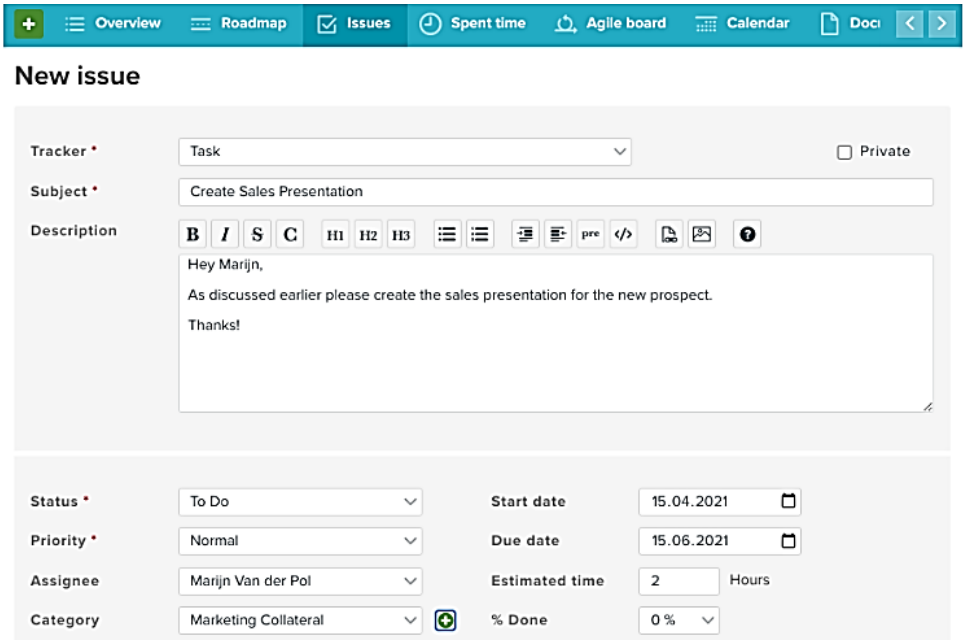

Рисунок 5.4 – Створення нового завдання у Planio [3]

Можна навіть створювати власні поля або шаблонні контрольні списки для конкретного проєкту (рис. 5.5).

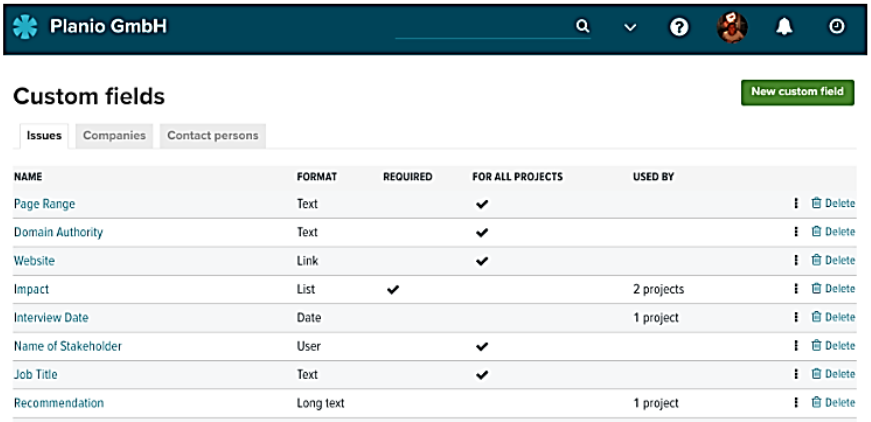

Рисунок 5.5 – Створення власного поля у Planio

Як дізнатися, що кожне питання або завдання «виконано» і можна рухатися далі?

План проєкту має містити визначення «виконаного» – критерії, за якими буде оцінюватися завдання. Визначення «готового» не тільки гарантує досягнення очікуваного рівня якості, але й створює спільне розуміння у команді.

Варто подумати про критерії виконання на різних етапах проєкту:

1) Визначення готового проєкту: чого потрібно досягти, щоб зробити цей проєкт успішним?

2) Визначення етапу завершення: як дізнатися, коли етап завершено, і можна переходити до наступного?

3) Критерії виконання завдання: Що конкретно потрібно зробити, щоб завдання було виконане?

Чудовий спосіб упорядкувати критерії виконання – це створити контрольний список у Planio за шаблоном (рис. 5.6).

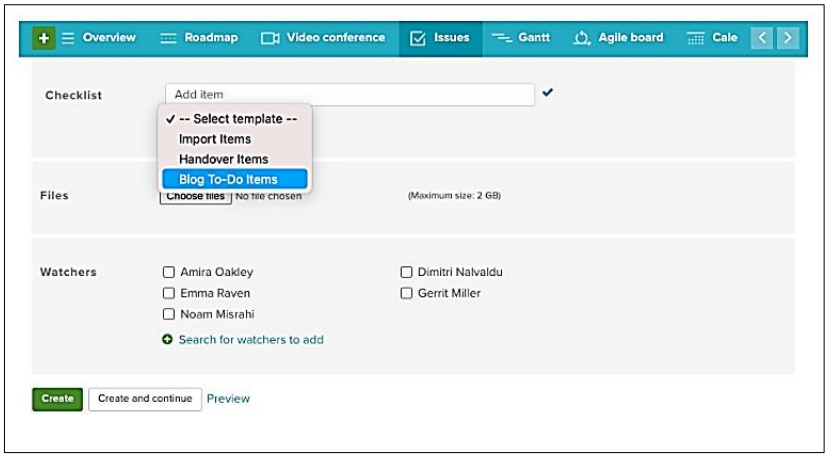

Рисунок 5.5 – Створення контрольних списків у Planio [3]

Таким чином, кожне нове завдання або етап автоматично має місце для визначення «виконаного».

Останньою частиною етапу планування є визначення та усунення будь-яких ризиків проєкту.

Раніше уже було складено список проблем і обмежень високого рівня. Тепер, маючи завершений план проєкту, настав час розглянути ці ризики та побачити, які з них все ще можливі. Потім варто спланувати, як їх виконуватимете.

Виконання. Часто починаючи зі стартової зустрічі проєкту, команда IT-проєкту починає розробку нового програмного забезпечення, переміщення даних або оновлення процесів, відповідно до обсягу проєкту. У світі agile починаються щоденні та перші спринти.

Наведемо короткий опис того, як виглядають завдання та обов'язки на цьому етапі проєкту:

– варто почати із стартової зустрічі проєкту;
– потрібно контролювати прогрес за допомогою регулярних оглядів;
– варто скоригувати завдання, цілі та часові рамки на основі відгуків;
– церемонії Run Agile (планування спринту та ретроспективи);
– потрібно поспілкуватися із зацікавленими сторонами та своєю командою.

Є дві окремі частини цієї фази. По-перше, це є можливість допомогти своїй команді працювати якнайкраще, тримаючи їх організованими, мотивованими та продуктивними. Водночас буде відстежуватися прогрес, перевірятиметься робота за планом і відповідно змінюватиметься графік або план.

Розглянемо, як ефективно впоратися з обома аспектами цієї фази.

Незважаючи на те, що було витрачено весь цей час на ініціювання та планування проєкту, решта команди не має того самого рівня контексту та розуміння.

Стартова зустріч проєкту – це чудовий шанс створити контекст, мотивувати команду та пояснити мету роботи. Наведемо шаблон для того, як провести чудову стартову зустріч:

– почніть зі знайомства;
– дайте резюме проєкту;
– поясніть контекст клієнта або проєкту;
– ознайомтеся з обсягом робіт і основними результатами;
– обговоріть свій підхід/робочий процес;
– розподіліть ролі та обов'язки;
– обговоріть співпрацю та план спілкування;
– виділіть часові рамки, графіки та етапи;
– завершіть запитаннями та відповідями, та наступними кроками.

Моніторинг і контроль. Окрім етапу виконання, керівник IT-проєкту, моніторить та контролює проєкт, зосереджуючись на потрійному

обмеженні часу, вартості та обсягу. Якщо щось починає йти не так, його завдання виправити це та підтримувати команду на правильному шляху.

Протягом усього проєкту потрібно виділити час, щоб переглянути прогрес і перевірити роботу за планом. Існує майже 100 % ймовірність того, що робота піде не зовсім за планом. І це нормально. Потрібно це контролювати це на ранній стадії та внести корективи.

Якщо учасник в команді Agile, то можна зробити це під час ретроспективи спринту та нарад з планування спринту. В іншому випадку потрібно виділяти час щотижня або принаймні для кожного етапу під час реєстрації.

Ніхто не любить, коли під час проєкту змінюється обсяг роботи. Однак замість того, щоб ігнорувати нову інформацію та відгуки, які могли б покращити остаточний проєкт, слід чітко визначити, як і коли коригувати обсяг.

Це називається процесом контролю змін. У своїй найпростішій формі цей процес складається з п’яти кроків:

Запропонуйте зміни. Встановіть вказівки щодо того, які пропонуються зміни. Яка інформація потрібна для їх розгляду?

Узагальніть їх вплив. Як ця зміна вплине на ваш графік, бюджет або іншу залежну роботу?

Вирішіть, що з цим робити. Знайте, хто приймає рішення щодо коригування масштабу.

Впровадити зміни. Якщо зміну буде схвалено, як ви відкоригуєте свій список завдань?

Закрийте зміну. Після включення, зміну потрібно «закрити», щоб рухатися далі без подальшого обговорення чи переривання.

Протягом усього етапу виконання життєвого циклу проєкту потрібно повідомляти про прогрес зацікавленим сторонам та іншим командам. План комунікацій визначає, кому потрібні звіти про хід роботи, частоту спілкування та рівень зворотного зв’язку, який очікується.

Комунікаційні плани є важливими для агентств, які працюють з клієнтами. Однак вони так само важливі для внутрішніх команд із кількома зацікавленими сторонами або залежними командами.

Закриття. Коли завершується IT-проєкт, потрібно зосередитися на двох речах: виконати те, що було обіцяно, і передати роботу команді BAU (Business-as-Usual).

Якщо проєкт пройшов успішно, досягнуто результатів і отримано підтвердження від зацікавлених сторін, настав час офіційно закрити проєкт. Цей етап називається закриттям .

Залежно від подальших кроків, етап закриття може виглядати дещо інакше. Можна передати роботу іншій команді для обслуговування, використання результатів проєкту, для планування нової роботи або навіть просто намагатися зрозуміти, що пішло не так.

Однак кроки фази закриття відбуватимуться за подібною схемою:

– потрібно перевірити кінцеві результати відповідно до визначення «готового» та передати;

– варто провести ретроспективну зустріч проєкту (і попросити свого клієнта надати відгук);

– потрібно задокументувати отримані уроки;

– вкінці варто звільнити проєктну команду та ресурси.

Коротко опишемо кожен із них.

Настав час переглянути кінцеві результати і перевірити, чи відповідає виконана робота критеріям випуску.

Якщо це так, тоді можна випустити функцію або передати роботу іншій команді. Якщо ні, потрібно буде зрозуміти, чому і що має статися далі.

Коли є задоволення результатами, настав час відступити та формально оцінити роботу. Цей процес передбачає зустріч із командою, щоб переглянути роботу та відповісти на кілька простих запитань:

– Що задумали?

– Що насправді вдалося зробити?

– Що спрацювало?

– Що також не спрацювало?

– Чого можемо навчитися з цього проєкту?

Кожен результат проєкту – успіх чи невдача – це шанс поглибити знання команди.

Переконайтеся, що приділено трохи часу, щоб задокументувати всі отримані уроки , щоб допомогти майбутнім командам працювати над подібними проєктами. Потрібно переконатися, що ці уроки зберігаються десь у легкодоступному місці, наприклад, у вкладці Planio (рис. 5.6).

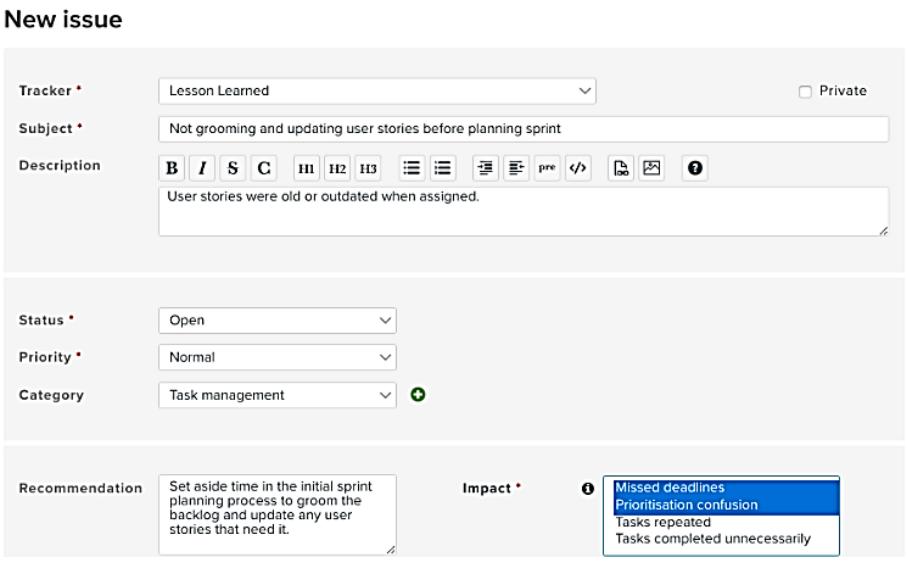

Рисунок 5.6 – Отримані уроки у Planio [3]

Останнім кроком будь-якого проєкту є звільнення команди та ресурсів. Звільнення команди є, по суті, символічним актом, але це допомагає уточнити, що проєкт завершено і настав час рухатися далі.

Керівники IT-проєктів несуть відповідальність за перевірку того, чи результат відповідає вимогам, а потім за те, щоб кінцеві користувачі/замовники/оператори знали, як ефективно використовувати нову систему.

Методологія, яку обирає IT-менеджер, залежить від проєкту, який запускається, але, як правило:

– **якщо проєкт легко охопити або має високий рівень ризику**, найкраще вибрати водоспад, щоб забезпечити найбільшу структуру. Зміни IT-інфраструктури, оновлення серверів і встановлення апаратного забезпечення часто належать до цієї категорії;

– **якщо проєкт більш інноваційний**, вимагає мислення «тестування та навчання» або покладається на постійні відгуки клієнтів, тоді краще вибрати Agile. IT-проєкти, такі як створення веб-сайтів/додатків, оновлення процесів і міграція даних, найкраще обслуговуються за допомогою гнучкого підходу.

Чотири фази життєвого циклу проєкту застосовуються майже до будь-якого проєкту, який можливо уявити. Однак сувора структура іноді може здатися недоступною (або неприродною) для гнучких команд.

Якщо постійно надавати функції, збирати й застосовувати відгуки, дивно думати про замкнутий цикл створення програмного забезпечення. Однак

гнучкі проєкти можуть отримати вигоду від дотримання фаз життєвого циклу проєкту з кількома незначними налаштуваннями.

Але життєвий цикл продукту полягає в тому, що він ніколи не повинен закінчуватися. Найкращі команди беруть будь-який результат – успіх чи невдачу – і використовують його для просування вперед із наступним проєктом.

5.4 Як запустити успішний IT-проєкт: 8 кроків до успіху

Тепер, коли отримано базові знання з управління IT-проєктами, настав час детально розібратися в тому, як керувати власним проєктом:

Крок 1. Потрібно знайти ділову потребу або проблему, яка потребує IT-рішення.

Усім проєктам потрібна причина для існування – і IT-проєкти нічим не відрізняються. Зазвичай проєкт розпочинається з однієї з двох причин: або у компанії є можливість або потреба клієнта, яку вони хотіли б використати, або є проблема, яку потрібно вирішити за допомогою IT-рішення.

Приклад реального проєкту.

Анна працює менеджером IT-проєктів у компанії MedCo Ltd, яка є дистриб'ютором фармацевтичних препаратів у Європі.

Анна розмовляла з колегами з операційного відділу, які скаржилися, що система керування замовленнями MedCo постійно дає збої. Це означало, що вони не могли обробити замовлення, і клієнти були незадоволені. Анна знала, що це проблема, яку необхідно вирішити.

Крок 2. Потрібно знайти когось, хто буде спонсорувати проєкт.

Кожному проєкту потрібна підтримка високопоставленої зацікавленої сторони. Це не тільки допомагає стимулювати ініціативу, але й забезпечує необхідне фінансування та ресурси.

Хоча це часто не робота менеджера IT-проєктів, якщо він працює в допоміжній ролі, наприклад менеджері з портфоліо/бізнес-менеджері, вам потрібно буде переконатися, що для всіх проєктів є відповідний бізнес-спонсор (рис. 5.7).

Бізнес-спонсор також повинен взяти на себе відповідальність за переваги проєкту, обґрунтувавши причину інвестування в ініціативу. Зрештою, проєкт, який не приносить жодних переваг, взагалі не дуже хороший проєкт.

Рисунок 5.7 – Обов'язки спонсора IT-проєктів [3]

Приклад реального проєкту

Анна розмовляє зі своїм портфоліо-менеджером Сарою про проблему із системою керування замовленнями, і вони погоджуються, що це був би чудовий проєкт.

Сара та Анна звертаються до директора з операцій Джима, який погоджується спонсорувати проєкт. Джим знає, що нова система замовлень спростить бізнес-операції, дозволить MedCo обробляти більше замовлень і підвищить задоволеність клієнтів.

Крок 3. Потрібно визначити обсяг, графік і бюджет проєкту.

Тепер, коли проєкт схвалено та визначено переваги, настав час подумати, як перейти від А до Б. Це розбивається на чотири підкроки:

1) **потрібно почати з оцінки обсягу проєкту.** Визначити роботу, щоб реалізувати своє бачення, а також зазначити, що потрібно робити (що виходить за рамки). Наприклад, можна вказати нові функції продукту, які потрібно створити, або, які нові апаратні елементи придбати;

2) потрібно створити розклад проєкту **високого рівня.** Це можна зробити кількома способами, зокрема, за допомогою методів детального планування, наприклад, методу критичного шляху;

3) **потрібно визначити ресурси, які знадобляться для виконання проєкту.** Знайти час, щоб опрацювати сферу діяльності та визначити фізичні

предмети, які знадобляться (наприклад, сировина), і людей, які будуть підтримувати (наприклад, розробники програмного забезпечення);

4) **потрібно зібрати усе це разом і спланувати бюджет усього проєкту.** Хоча вартість сировини легко розрахувати, потрібно зосередитися на витратах ресурсів, таких як розрахунок щоденних/тижневих/місячних ставок підрядника.

Приклад реального проєкту

Анна опрацьовує обсяг, вирішивши, що MedCo розробить нову систему управління замовленнями з нуля.

За її оцінками, це займе сім місяців: два місяці на проєктування, три місяці на розробку, один місяць на тестування та один на запуск.

Для цього проєкту їй знадобиться підтримка одного бізнес-аналітика, трьох розробників програмного забезпечення, Scrum-майстра та двох тестувальників програмного забезпечення. Також можна придбати кілька ліцензій на хмарний хостинг і розробку. Вона додає все це, щоб створити бюджет проєкту.

Крок 4. Потрібно вибрати правильну методологію управління IT-проєктами.

Визначивши детальну інформацію високого рівня, потрібно подумати про те, як це зробити.

Потрібно здійснити порівняння «Вотерспаду» та «Agile», але якщо не можна вирішити, яку методику обрати, тоді можна вибрати гібридний підхід. Насправді дослідження показали, що 21% менеджерів проєктів обирають цей шлях, щоб допомогти їм отримати найкраще з обох варіантів у своїх поставках.

Приклад реального проєкту

Враховуючи, що IT-проєкт Анни – це нова збірка додатків, яка буде покладатися на регулярні відгуки зацікавлених сторін, вона вирішує застосувати підхід до проєкту Agile.

Крок 5. Потрібно визначити свої ключові ризики та створити план управління ризиками.

Перш ніж розпочати проєкт, потрібно подумати про ризики, які можуть виникнути. Це означає аналіз середовища, планування реагування на ризики, пом’якшення впливу ризиків і просування до безпечної реалізації проєкту.

Приклад реального проєкту

Анна розглядає свої ризики, обговорюючи свої плани з колегою-менеджером проєкту, який нещодавно виконав подібний проєкт. Оскільки

існуюча система керування замовленнями з кожним днем стає все більш нестабільною, основний ризик для Анни полягає в її довгому графіку, тому вона вносить корективи в обсяги, щоб швидше виконувати роботу.

Крок 6. Потрібно спланувати ключові завдання та етапи та приступити до виконання.

Щоб налагодити роботу команди, потрібно розділити сферу діяльності на критичні завдання та етапи та призначити їх найкращим членам команди для цієї роботи. Коли це зробити, тоді можна побачити, як розклад «оживає» (а також визначити будь-які потенційні вузькі місця або блокувальники, перш ніж вони виникнуть).

Щоб полегшити це, можна використовувати програмне забезпечення для керування IT-проєктами, наприклад, Planio, яке допоможе спланувати та візуалізувати весь проєкт в одному місці. За допомогою Planio можна переглядати всі завдання проєкту у вигляді діаграми Ганта, календаря або дошки Kanban (рис. 5.8).

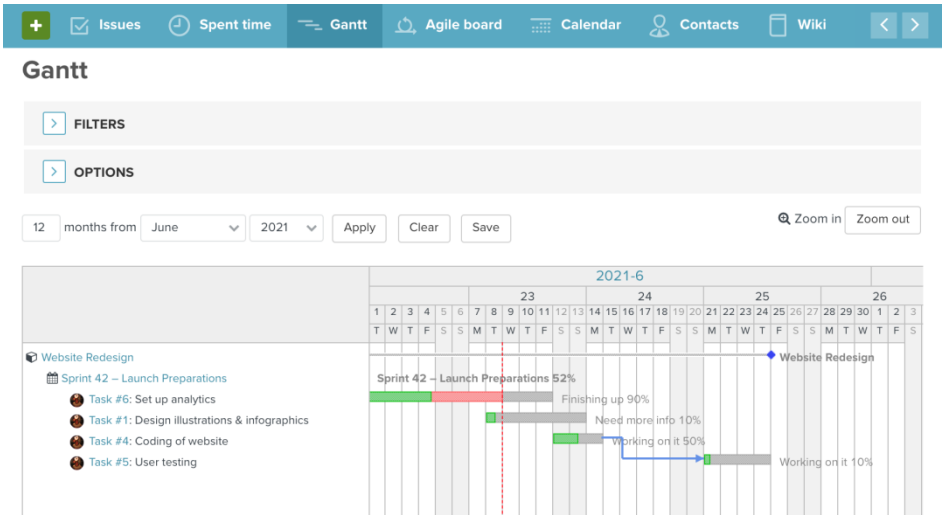

Рисунок 5.8 – Діаграма Ганта у Planio

Приклад реального проєкту

Анна розділяє роботу над проєктом зі своєю командою. Вона встановлює віхи для підетапів проєктування та створення, рівномірно розподіляючи роботу між бізнес-аналітиком і розробниками програмного забезпечення. З наближенням етапу тестування вона залучає тестувальників, щоб переконатися, що вони можуть почати якомога швидше.

Крок 7. Потрібно відстежити прогрес, отримати відгуки та надіслати оновлення.

Особливо з гнучкою методологією, ключем до успіху проєкту є постійний моніторинг прогресу, отримання зворотного зв'язку від зацікавлених сторін і оновлення способів роботи по ходу роботи.

Як керівнику ІТ-проєктів важливо тримати руку на пульсі, активно уникати розповзання масштабів, перевищення бюджету, несподіваних ризиків і відставання від графіка.

Приклад реального проєкту

Анна запроваджує належне підпорядкування, щоб допомогти їй контролювати проєкт. Вона проводить щоденні зустрічі зі своєю командою, щоб продовжувати роботу та керувати обсягом, а також призначає зустріч з питань фінансів раз на два тижні для оцінки фінансів.

Крок 8. Потрібно здійснити підготовку до запуску та закриття.

Коли наблизиться завершення, потрібно підготуватися до запуску нового проєкту чи продукту. Опишемо кілька кроків, які слід врахувати на етапах закриття та запуску:

- повністю **протестувати** власну нову ІТ-систему, переконавшись, що вона стабільна та відповідає потребам користувачів;
- **задокументувати** всі технічні примітки, щоб команди могли використовувати їх у майбутньому;
- розгорнути **навчання** для користувачів, переконавшись, що вони мають необхідні навички та знання;
- залишити деяких членів команди в режимі очікування для **відстеження та виправлення** будь-яких помилок;
- **зберегти** отримані уроки, щоб покращити свою роботу наступного разу.

Приклад реального проєкту

Коли проєкт наближається до завершення, Анна та команда проводять низку тренінгів з користувачами щодо того, як працювати з системою, використовуючи ці сесії для виявлення та виправлення будь-яких непоширених помилок.

Далі вони документують всю свою роботу та передають її командам ІТ-підтримки, відзначаючи завершення добре виконаної роботи.

Бути менеджером ІТ-проєктів – це різностороння, складна та корисна кар'єра. Розглянемо основні поради, як стати менеджером ІТ-проєктів;

1) потрібно опанувати основи управління проєктами. Опанування основ є ключовим для того, щоб стати чудовим менеджером ІТ-проєктів. Також

потрібно зосередитися на зміцненні своїх основних навичок управління проєктами, включаючи управління зацікавленими сторонами, комунікації, планування, бюджетування та управління ризиками;

2) потрібно **оцінити і розвивати свої технічні навички.** Окрім основ PM, це добре послужить, щоб бути в курсі технологій. Також потрібно бути в курсі останніх технологічних тенденцій, отримати базові знання про кодування та навчитися користуватися та розуміти складні набори даних;

3) потрібно **спілкуватися з іншими IT-менеджерами або менеджерами технічних проєктів.** Якщо є бажання проникнути у світ управління IT-проєктами, то варто почати будувати свою мережу, щоб отримати підказки, поради та вказівки щодо навігації у світі технологій.

І, звичайно, потрібно не забувати вибирати інструмент управління проєктами, який полегшить життя PM і його команді.

5.5 Вибір найкращого програмного забезпечення для управління IT-проєктами

Як і всі менеджери проєктів, менеджери IT-проєктів покладаються на чудове програмне забезпечення для виконання своєї роботи. Але, на жаль, дослідження 2020 року показало, що лише 35 % користувачів були помірно або дуже задоволені своїми інструментами PM.

Програмне забезпечення для керування проєктами – це потужний інструмент, який пропонує низку функцій, які дозволяють керівникам проєктів контролювати витрати на проєкт, досягати основних етапів і дотримуватись термінів. Основна мета програмного забезпечення для управління проєктами – допомогти компаніям виконати зобов’язання, які вони взяли перед зацікавленими сторонами проєкту, гарантуючи, що виконання проєкту приведе до бажаних результатів.

Для організацій, що надають професійні послуги, які отримують значні доходи від проєктів, орієнтованих на клієнтів, основна увага приділяється виконанню проєктів у строк, у межах бюджету та відповідно до обсягу, узгодженого із замовником у технічному завданні (SOW). Багато програмних рішень для управління проєктами, орієнтованих на галузь професійних послуг, навіть допоможуть менеджерам відстежувати платежі клієнтів і керувати витратами проєкту [5].

Опишемо загальні функції програмного забезпечення для керування проєктами.

Інформаційні панелі проєкту, які дають менеджерам можливість швидко вносити зміни та миттєво розуміти продуктивність проєкту замість пошуку в експортованих даних.

Програмне забезпечення для відстеження робочого часу, куди члени команди вносять час, відпрацьований для виконання конкретних завдань, для підтримки точного виставлення рахунків клієнтам і відстеження швидкості використання проєкту.

Інструменти керування завданнями, які дають менеджерам можливість призначати детальну роботу, пов'язану з проєктом, ресурсам, із сучасними системами, які дозволяють виконувати все це на одній інформаційній панелі, яку легко контролювати.

Діаграми Ганта, які визначають і контролюють потік взаємопов'язаних завдань, які складають проєкт для покращення використання та дотримання термінів. Діаграми Ганта часто використовуються в стилі управління SCRUM .

Шаблони проєктів, які дозволяють швидко та легко повторювати проєкти під час створення роботи, схожої на попередні проєкти.

Протягом усього проєкту програмне забезпечення для управління проєктом надаватиме як інформацію щодо його продуктивності, так і засоби контролю, необхідні для зміни виконуваної роботи. Усе, від концептуалізації та підписання контракту до постачання продукту, можна завершити та відстежити за допомогою потужного сучасного програмного забезпечення для управління проєктами. Тому організації можуть краще контролювати всі аспекти роботи, а також краще розуміти, де можна вдосконалити майбутні проєкти.

Опишемо 6 важливих переваг програмного забезпечення для управління проєктами.

Покращене спілкування та співпраця. Програмне забезпечення для керування проєктами централізує всю комунікацію, пов'язану з проєктом, полегшуючи членам команди ефективну співпрацю. Завдяки таким функціям, як коментарі до завдань, спільний доступ до файлів і оновлення в реальному часі, члени команди можуть залишатися в курсі та працювати разом без проблем, зменшуючи ризик неправильного спілкування та втрати інформації.

Покращене управління ресурсами. Інструменти управління проєктами допомагають визначати та ефективно розподіляти ресурси між проєктами.

Завдяки таким функціям, як відображення навичок, робочі календарі та відстеження часу, менеджери проєктів можуть оптимізувати використання ресурсів, запобігти вузьким місцям і забезпечити, щоб члени команди працювали над завданнями, які відповідають їхнім сильним сторонам, що сприяє підвищенню продуктивності та задоволенню працівників.

Централізовані та організовані дані проєкту. Зберігаючи всю пов’язану з проєктом інформацію в одному доступному місці, програмне забезпечення для керування проєктами позбавляє членів команди від необхідності витрачати час на пошук важливих даних. Цей централізований підхід не тільки покращує доступність даних, але й сприяє прозорості та полегшує прийняття кращих рішень протягом життєвого циклу проєкту.

Спрощене звітування та аналітика. Інструменти управління проєктами пропонують налаштовані інформаційні панелі та потужні можливості звітування, що дає змогу керівникам проєктів відстежувати прогрес, визначати тенденції та отримувати корисну інформацію. Завдяки спрощеній звітності та аналітиці команди можуть приймати рішення на основі даних, оптимізувати процеси та чітко повідомляти зацікавленим сторонам про статус проєкту.

Розширені можливості віддаленої роботи. Хмарне програмне забезпечення для керування проєктами дозволяє командам ефективно працювати будь-де та в будь-який час. Завдяки таким функціям, як комунікаційні інструменти, відстеження часу та віртуальна співпраця, віддалені команди можуть залишатися на зв’язку, продуктивними та злагодженими, незалежно від їх фізичного розташування.

Покращене управління бюджетом. Програмне забезпечення для управління проєктами допомагає командам залишатися в межах бюджету, надаючи інструменти для прогнозування бюджету, відстеження витрат і сповіщень про можливі перевитрати. Завдяки кращим можливостям управління бюджетом керівники проєктів можуть приймати обґрунтовані фінансові рішення, повідомляти зацікавленим сторонам про статус бюджету та гарантувати, що проєкти завершені в межах виділених ресурсів.

Підсумовуючи, програмне забезпечення для управління проєктами пропонує широкий спектр переваг, які можуть значно підвищити ефективність, продуктивність і успіх проєктів. Використовуючи ці інструменти, команди можуть оптимізувати свої робочі процеси, оптимізувати розподіл ресурсів і постійно забезпечувати високоякісні результати, зрештою сприяючи загальному успіху своєї організації.

Не лише менеджери проєктів використовують такі програми – у цих організаціях майже всі, хто займається наданням послуг клієнтам протягом складного життєвого циклу реалізації проєкту, торкаються програмного забезпечення для керування проєктами. Сюди входять групи продажів, які встановлюють SOW, групи управління ресурсами, які призначають ресурси для завдань, консультанти, які відстежують час і витрати та надають оновлення щодо призначених завдань, і члени фінансової команди, які відстежують витрати, платежі та доходи проєкту. Ці члени команди потребують інструментів, які дозволять їм підтримувати високу якість роботи, підживлювати креативність, максимально використовувати членів своєї команди та підтримувати зацікавленість і задоволення клієнтів.

Відповідно до останнього звіту дослідження S&P Global Market Intelligence «Макротехнологічний зсув, що впливає на індустрію професійних послуг», існує зростаючий попит на потужніші програмні рішення, які можуть задовольнити зростаючі потреби сучасних фірм, що надають професійні послуги. Згідно з дослідженням, 41 % респондентів хочуть подолати перешкоди для співпраці, 40 % хочуть краще керувати ресурсами протягом життєвих циклів проєкту, а 39 % хочуть підвищити свою гнучкість і реагувати на зміни та сценарії на основі даних.

У професійних послугах кожен проєкт – це можливість отримати дохід. Але це також шанс побудувати довготривалі відносини з клієнтами, які сприяють повторюванню бізнесу. Це означає більш передбачувані прибутки та більшу фінансову стабільність як бізнесу.

У цьому може допомогти програмне забезпечення, яке відстежує деталі кожного проєкту. Але перш ніж вибрати програмне забезпечення для управління проєктами, потрібно розглянути плюси та мінуси кожного варіанту.

Програмне забезпечення для управління проєктами надає низку переваг компаніям, які прагнуть оптимізувати свою діяльність, підвищити ефективність і підтримувати послідовний безперебійний робочий процес. У динамічному бізнес-середовищі важливо підтримувати високий рівень продуктивності, і саме в цьому програмне забезпечення для управління проєктами є кращим.

Покращена організація та визначення пріоритетів є одними з головних переваг програмного забезпечення для управління проєктами. Ці рішення дозволяють менеджерам чітко визначати завдання, встановлювати терміни

виконання, розподіляти відповідальність і відслідковувати всі компоненти проєкту в централізованому місці. Це також допомагає в плануванні та забезпеченні ресурсами, запобігаючи вузьким місцям або недостатньому використанню ресурсів.

Програмне забезпечення для керування проєктами також покращує комунікацію та співпрацю, що особливо важливо в умовах розподіленої або віддаленої роботи. Члени команди можуть обмінюватися ідеями, оновленнями та відгуками в режимі реального часу, що сприяє покращенню прийняття рішень, підвищенню задоволеності клієнтів і кращому утриманню клієнтів. Крім того, програмне забезпечення для управління проєктами забезпечує чіткий огляд робочого навантаження та прогресу кожного члена команди, сприяючи справедливому розподілу завдань.

Ще однією важливою перевагою програмного забезпечення для управління проєктами є зниження ризиків. Ці інструменти дають менеджерам уявлення про потенційні підводні камені проєкту та про те, як їх обійти, таким чином гарантуючи, що проєкт залишається на правильному шляху. Завдяки ранньому виявленню будь-яких відхилень від встановленого плану можна швидко вжити профілактичні заходи, заощадивши час і ресурси.

Деякі програми зменшують надмірність, особливо в компаніях, які залежать від електронних таблиць для керування проєктами та ресурсами проєкту – програмне забезпечення для керування проєктами консолідує системи, щоб інформація існувала лише в одному місці. Це допомагає членам команди уникнути надмірності файлів, що може призвести до того, що деякі називають «кошмаром керування версіями». Окрім зменшення кількості різних файлів, програмне забезпечення для керування проєктами також гарантує, що команди використовують один метод для відстеження часу, завдань, витрат, інших статей бюджету, таких як рахунки-фактури тощо. Наявність однієї системи заохочує команди творчо працювати в межах заданої структури та запобігає непорозумінням і помилкам, які часто виникають через використання кількох рішень для одного проєкту.

Нарешті, сучасне програмне забезпечення для управління проєктами може автоматизувати багато різних повторюваних процесів і забезпечити центр для управління ресурсами, завданнями, графіками, проєктами та спілкуванням. Уся ця інформація, що зберігається в одній системі, дає величезну можливість запускати звіти про хід роботи – щоденні, щотижневі, щомісячні чи щоквартальні – все за автоматизованим розкладом. Сучасні

передові системи можуть пропонувати звіти про хід роботи в режимі реального часу, тож ви можете легко візуалізувати та обмінюватися інформацією, пов’язаною з працездатністю проєкту. Це не лише дає вам уявлення про те, де ви можете налаштувати свої ресурси, щоб приймати розумніші рішення щодо виконання проєкту; це також дозволяє давати детальні, прозорі відповіді на запитання зацікавлених сторін.

Незважаючи на численні переваги, використання програмного забезпечення для управління проєктами має потенційні недоліки, які необхідно враховувати перед його впровадженням.

Одним з головних недоліків є початкові витрати. Придбання ліцензій на програмне забезпечення, оновлення обладнання та навчання можуть стати значним фінансовим тягарем для малих і середніх підприємств.

Крім того, використання програмного забезпечення для керування проєктами може збільшити складність, а не оптимізувати його. З великою кількістю функцій, які пропонує таке програмне забезпечення, члени команди можуть бути приголомшені. Якщо не буде проведено відповідне навчання, опір користувача та небажання використовувати програмне забезпечення можуть негативно вплинути на продуктивність.

Програмне забезпечення для управління проєктами також ризикує знеособити управління проєктами. Надмірна залежність від програмного забезпечення може зробити членів команди менш зв’язаними, зменшуючи міжособистісне та спонтанне спілкування, яке є ключовим у динамічному робочому середовищі. Це може створити хибне відчуття безпеки, змушуючи користувачів нехтувати або пропускати важливі завдання, вважаючи, що програмне забезпечення вловить будь-який недогляд.

Це також створює потенційні проблеми з безпекою та конфіденційністю. Обмін конфіденційною інформацією на платформі, доступній для кількох користувачів, може бути проблематичним, якщо не реалізовано відповідні протоколи та елементи керування.

Нарешті, програмне забезпечення для управління проєктами часто вимагає початкових інвестицій для користувачів, щоб ознайомитися з його можливостями та функціями. Це може призвести до тимчасового зниження продуктивності, коли користувачі «переміщаються» програмним забезпеченням, перериваючи звичайний робочий процес, доки не буде встановлено знайомство та плавність роботи з системою.

Наведемо приклади найбільш популярних продуктів для управління проєктами.

Monday.com – це універсальне програмне забезпечення для управління проєктами, яке обслуговує команди будь-якого розміру та бюджету. Його безкоштовний план підтримує команди з двох осіб, тоді як платні плани від 9 до 19 доларів США на члена команди на місяць пропонують розширені функції, такі як автоматизація та інтеграція (рис. 5.9).

Перегляд таблиці на monday.com особливо корисний для керування окремими проєктами з кольоровим кодуванням статусів, налаштованими стовпцями та знімками облич чи ініціалами для легкого відстеження завдань. Розглянемо переваги та недоліки ПЗ «monday.com» (табл. 5.3).

Таблиця 5.3 – Переваги та недоліки «monday.com»

Плюси	Мінуси
Універсальний і підходить для команд будь-якого розміру	Вид Канбан може бути захараценим
Інтуїтивно зрозумілий вигляд таблиці для керування одним проєктом..	Розширені параметри звітування обмежені планами з вищою ціною
Прості у використанні функції автоматизації	Мінімум 3 місця для покупки
Для невеликих команд доступний безкоштовний план	Мобільний додаток може мати проблеми із синхронізацією

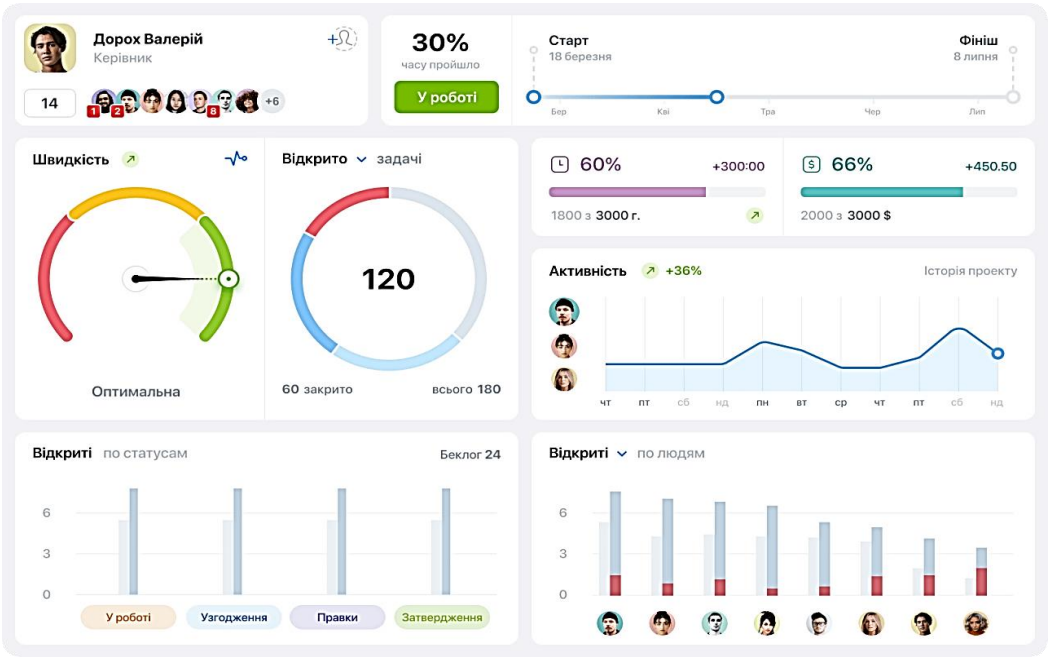

Рисунок 5.9 – Інтерфейс ПЗ «Monday.com» [6]

HubSpot Project Management, частина платформи HubSpot CRM, призначена для оптимізації управління завданнями та проєктами в рамках єдиної системи. Це дозволяє командам створювати, призначати та відстежувати завдання, встановлювати пріоритети та терміни, а також полегшувати співпрацю через зручний інтерфейс. Незважаючи на простоту використання та інтеграцію з іншими інструментами HubSpot, йому не вистачає деяких розширених функцій, доступних у спеціальному програмному забезпеченні для керування проєктами.

Однак користувачі можуть розширити функціональність за допомогою інтеграцій, доступних у App Marketplace HubSpot, таких як діаграми Ганта та інструменти керування ресурсами. Це робить HubSpot Projects універсальним варіантом для компаній, які хочуть ефективно керувати своїми проєктами в екосистемі HubSpot. Загалом, це потужний інструмент для покращення видимості, співпраці та ефективності управління проєктами (рис. 5.10).

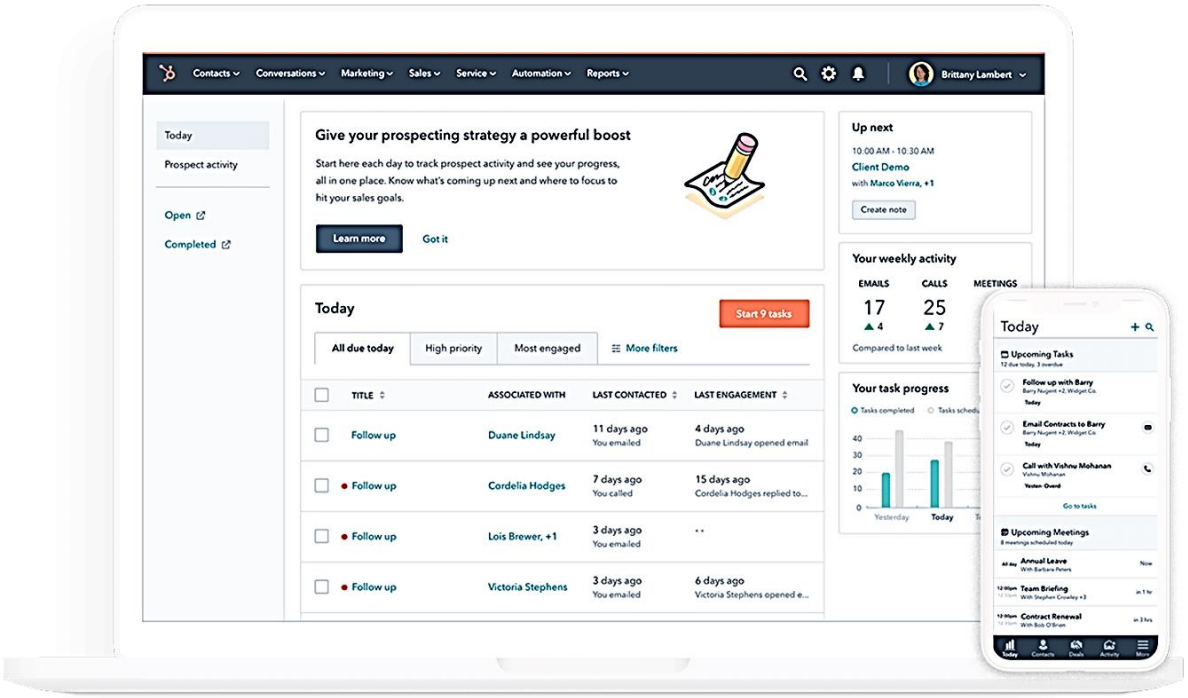

Рисунок 5.10 – Інтерфейс ПЗ «HubSpot Projects» [7]

Розглянемо переваги та недоліки ПЗ «HubSpot Projects» (табл. 5.4).

Таблиця 5.4 – Переваги та недоліки «HubSpot Projects»

Плюси	Мінуси
Універсальна платформа для продажів і маркетингу	Новіші в корпоративному просторі
Зручний для користувача з функціями перетягування	Менш налаштована, ніж інші CRM
Доступна безкоштовна версія CRM	Може швидко подорожчати
Чудова підтримка клієнтів	Річні контракти, без дострокового розірвання

Також коротко опишемо ще одне програмне забезпечення в управлінні проєктами – ClickUp.

ClickUp об’єднує команди ближче за допомогою підключених робочих процесів, документів, інформаційних панелей у реальному часі тощо, допомагаючи кожному рухатися швидше, працювати розумніше та економити час (рис. 5.11).

Продукт дозволяє прискорити планування та виконання проєкту за допомогою ClickUp AI; створювати підзавдання автоматично на основі описів завдань, узагальнювати ланцюжки коментарів, автономно писати оновлення тощо за допомогою свого менеджера проєкту AI.

Основними функціями ClickUp є:

– універсальне управління знаннями та роботою;
– спеціальні перегляди для міжфункціональних проєктів;
– підвищення ефективності за допомогою автоматизації та звітності.

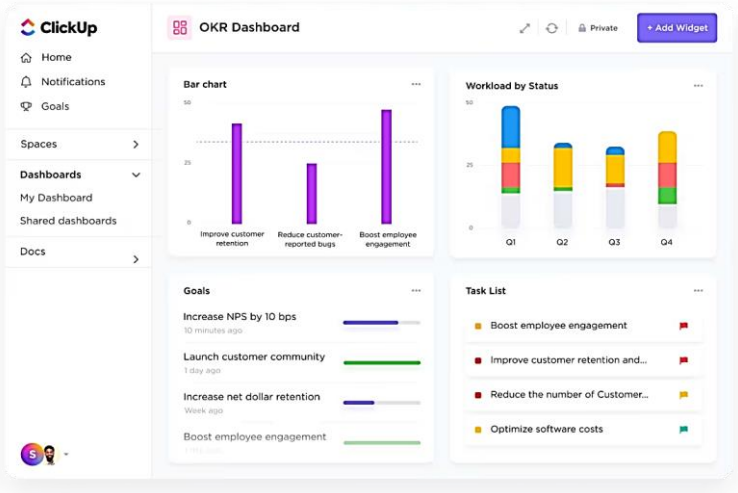

Рисунок 5.11 – Інтерфейс ПЗ «ClickUp» [8]

Питання для самоконтролю

1. Що таке проєкт?
2. Дайте визначення поняттю «управління ІТ-проєктами».
3. Які функції виконує проєктний менеджер?
4. Які ПЗ найефективніші в проєктному менеджменті?
5. Які існують переваги та недоліки програмних продуктів для управління проєктами?
6. Опишіть 8 кроків для успішного запуску ІТ-проєкту.
7. Опишіть фази ЖЦ проєкту?
8. Які кроки виконуються на фазі «завершення проєкту»?
9. Які гнучкі методики управління ІТ-проєктами існують?
10. Опишіть правильність рішень Анни, що були прийняті в реальному проєкті відповідно до представленого прикладу.

Тестові завдання

1. Що НЕ має зазнавати змін в проєкті?

a) терміни;
b) замовники;
c) бюджет;
d) цілі;
e) ресурси.

2. Проєкт – це ...

a) організаційно-правова, технічна, інженерна документація з реалізації запланованого дійства;
b) це комплекс завдань, які реалізують із метою отримання конкретних унікальних результатів у межах відведеного часу і у межах затвердженого бюджету;
c) група елементів (люди та технічні елементи), які організовані таким чином, що вони в змозі діяти як єдине ціле з метою досягнення поставлених перед ними цілей;
d) комплекс робіт, послуг і продуктів, виробництво яких має бути забезпечено з метою досягнення поставленої мети;
e) сукупність робіт та часу, витраченого на ці роботи.

3. Учасники проєкту – це ...

a) організації або особи, що беруть активну участь в проєкті, або на чиї інтереси впливають результати виконання чи завершення проєкту;

b) кінцеві користувачі результатів проєкту;

c) замовник, інвестор, менеджер проєкту і команда проєкту;

d) команда, що управляє проєктом;

e) інвестори проєкту.

4. Команда проєкту – це ...

a) група співробітників, які завжди зривають терміни проєкту;

b) сукупність осіб, об’єднаних в роботі над проєктом;

c) постачальники і підрядники в проєкті;

d) автори, редактори та учасники проєктної діяльності;

e) усі учасники, що задіяні в проєкт.

5. Проєкт вважається успішним коли:

a) виготовлений продукт проєкту;

b) спонсор проєкту заявив про його завершення;

c) продукт проєкту переданий до серійного виробництва;

d) проєкт задовольнив вимоги зацікавлених осіб або перевершив їх очікування;

e) усі кошти проєкту використано повністю.

6. Яку роль виконує Scrum-команда?

a) постановка для ітерації реально досяжних і пріоритетних для проєкту в цілому завдань;

b) забезпечує максимальну працездатність і продуктивну роботу команди;

c) представляє в проєкті інтереси замовника;

d) контролює фінансову сторону проєкту;

e) контролює забезпечення ресурсами проєкту.

7. Який процес розробки ПЗ є універсальним для розробки ПЗ будь-якого виду?

a) універсального процесу не існує;

b) Scrum;

c) CMMI;
d) UML;
e) вірної відповіді нема.

8. Що є одним з найбільш вагомих навиків керівника проєкту?
a) навики ведення переговорів;
b) навики впливу;
c) комунікативні навики;
d) навики вирішення проблем;
e) навики залучення коштів.

9. Застосування знань, навичок, інструментів і методів для виконання вимог, що висуваються до проєкту – це ...
a) управління проєктами;
b) структурне планування;
c) календарне планування;
d) всі з перерахованого;
e) нічого з перерахованого.

10. Який вид діяльності процесу розробки ПЗ акцентує увагу на принципах реалізації ПЗ?
a) тестування;
b) проєктування;
c) складання функціональних вимог до ПЗ;
d) аналіз;
e) верифікація.

Рекомендована література

1. Словник термінів з управління проєктами PMI. URL: https://pmiukraine.org/lexicon (дата звернення: 5.05.2024 р.).
2. Етапи управління IT-проєктами: ініціація проєкту для менеджера/ URL: https://iampm.club/ua/blog/etapi-upravlinnya-it-projektami-inicziacziya-projektu-dlya-menedzhera/ (дата звернення: 6.05.2024 р.).

3. IT project management explained: How to run great IT projects. URL: https://plan.io/blog/it-project-management/#what-is-it-project-management-with-examples-of-it-projects (дата звернення: 15.05.2024 р.).

4. Back to Basics: A Master Class on the 4 Phases of the Project Life Cycle. URL: https://plan.io/blog/project-life-cycle-phases/ (дата звернення: 17.05.2024 р.).

5. What is Project Management Software? URL: https://www.kantata.com/resource-article/what-is-project-management-software (дата звернення: 25.05.2024 р.).

6. 15 best project management software for 2024. URL: https://monday.com/blog/project-management/the-complete-project-management-software-list/ (дата звернення: 25.05.2024 р.).

7. Get Started With HubSpot. URL: http://surl.li/unrmn (дата звернення: 25.05.2024 р.).

8. The all-in-one platform for projects. URL: http://surl.li/unrlr (дата звернення: 25.05.2024 р.).

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ:

магістерський курс

Навчальний посібник

Рекомендовано
Луцьким національним технічним університетом

Комп’ютерне верстання та обкладинка І.Є. Андрущак

Формат 60×84/16. Умовн. друк. арк. 16,0
Тираж 50 пр.

ЛНТУ
43018, Луцьк, вул. Львівська ,75
Свідоцтво Держкомтелерадіо України № 4123 від 28.07.2011 р.