МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ВЕРБИЦЬКИЙ Назар Ігорович

Моделювання систем управління проєктами підприємства / Modeling of Project Management Systems for Enterprise

спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Управління проєктами

Кваліфікаційна робота

Виконав студент групи
КНУПм-21
Н. І. Вербицький
_____

Науковий керівник:
к.е.н., доцент Г. М. Гладій
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ М. П. Комар

ТЕРНОПІЛЬ – 2021

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Сучасні умови світового розвитку характеризуються постійним зростанням обсягу цифрових технологій, що ґрунтуються на використанні штучного інтелекту (ШІ), інноваційними процесами в суспільстві загалом, а також у сферах виробництва, науки та бізнесу. Зміни в суспільстві та в ІТ-галузі породжують нові вимоги до якості управління проєктами, що втілюють в життя «розумні» програми: розпізнавання осіб, постановка діагнозів, безпілотне управління транспортом тощо.

Робота зі створення програмних продуктів і з використанням штучного інтелекту за своїм змістом є науково-дослідною, тобто має творчий характер, а тому структура виробничих відносин у процесі створення програмного продукту є менш жорсткою та ієрархічною, причому ієрархія протягом часу створення програми може порушуватися. Це обумовлює необхідність вибору гнучкіших і стійкіших до змін методологій управління проєктами в компаніях, що спеціалізуються на розробці ШІ-продуктів. На сьогоднішній день найпоширенішими і популярними методологіями є Agile-сімейство, що охоплює Scrum, Kanban, XP та інші методи. Але вони не забезпечують повною мірою запити управління ШІ-проєктами, котрі є наукомісткими і оперують великими масивами даних. Тому проєктувальники вважають кращим метод Crisp-DM, який довів свою ефективність при роботі з великими обсягами даних.

Велика розмаїтість клієнтських запитів на ринку ІТ-послуг, постійно розширюваний спектр застосування ШІ обумовлюють існування різних видів компаній, котрі розрізняються як за кількістю співробітників, так і за створюваним продуктом. Тому постають актуальними питання адаптації існуючих моделей управління проєктами для конкретної компанії.

З точки зору менеджменту, дослідницька робота є одним з найскладніших об’єктів управління. Це обумовлено особливостями творчого процесу мислення, невизначеністю і важкою передбачуваністю кінцевих результатів, неможливістю повної алгоритмізації процесу пошуку нових рішень.

Отже, виникає протиріччя: з одного боку, об’єктивно зростає частка дослідницької роботи в сфері ІТ-бізнесу, посилюється необхідність постійного вдосконалення системи управління інноваційними та дослідницькими проєктами в галузі ШІ; з іншого боку, методи системи управління є недостатньо розробленими і часто неадекватними об’єкту управління. Така невідповідність призводить до того, що проєкт не окупається і робота виявляється нерентабельною.

Пошук шляхів вирішення зазначеного протиріччя зумовив вибір теми дослідження. Таким чином, такі чинники як: постійні й досить швидкі зміни в інформаційних технологіях; збільшення кількості точок прикладання штучного інтелекту в усіх сферах соціуму; виникнення великої кількості конкурентних компаній на ринку програмних продуктів; ускладнення розробок з використанням ШІ; унікальність кожного проєкту, який використовує ШІ; збільшення частки дослідницької, новаторської, творчої праці в роботі ІТ-фахівця; створення R&D-відділів у компаніях; посилення вимог інвесторів; збільшення ступеня невизначеності та ризику – підвищують відповідальність менеджменту всіх рівнів компанії, зумовлюють необхідність постійного вдосконалення методів управління проєктами та актуалізують дане дослідження.

Мета роботи: теоретичне обґрунтування і побудова моделі комп’ютеризованої системи управління проєктами ІТ-компанії.

Об’єкт дослідження: система управління проєктами, базованих на використанні штучного інтелекту і машинного навчання.

Предмет дослідження: методи і засоби моделювання комп’ютеризованої системи управління проєктами в компанії, орієнтованої на випуск продуктів на основі ШІ.

Реалізація мети дослідження конкретизується через вирішення таких завдань:

– вивчити сучасні методи та інструменти управління проєктами, з’ясувати їхні переваги та недоліки, особливості застосування в різних ІТ-компаніях;

– проаналізувати існуючі інформаційні системи з управління проєктами;

– проаналізувати зміст діяльності проєктної команди ІТ-компанії у процесі створення програмного продукту, що використовує штучний інтелект;

– з’ясувати особливості управління проєктами в ІТ-компанії, застосовувані методології та інструменти управління проєктами;

– проаналізувати організаційну структуру управління проєктною командою в ІТ-компанії, оцінити можливості удосконалення процесу управління проєктами;

– виконати проєктування і розробити модель КСУ проєктами на ТОВ «Неткрекер»;

– виконати якісне оцінювання розробленої моделі, застосовуючи такі методики: метод експертних оцінок; SWOT-аналіз; порівняльний аналіз рівня комп’ютеризації бізнес-процесів підприємства при використанні моделей управління «As Is» і «To Be».

В процесі виконання дослідження були використані такі **підходи і методи**:

– загальнонаукові прийоми і способи логічного пізнання: аналіз і синтез, абстрагування;

– системно-структурний, функціональний та формально-логічний підходи;

– аналіз літератури з питань теорії управління проектами, моделювання КСУ проектами, існуючих стандартів в управлінні проєктами;

– емпіричні методи (індивідуальні та групові інтерв’ю зі співробітниками компанії, спостереження за їх діяльністю протягом процесу створення програмного продукту, мозковий штурм, бенчмаркінг);

– методи структурного і об’єктно-орієнтованого аналізу моделювання інформаційної системи (процесне моделювання, ER-моделювання, опис прецедентів використання, контекстна діаграма);

– експертне оцінювання, SWOT-аналіз, оцінювання рівня комп’ютеризації бізнес-процесів.

Наукова новизна дослідження полягає в тому, що:

1) створена модель КСУ проєктами в компанії, яка спеціалізується на виробництві продуктів, що містять ШІ;

2) запропоновано спосіб представлення життєвого циклу проєкту і фаз проєктної діяльності у вигляді дерева рішень;

3) розроблено алгоритм процесу проєктної діяльності на етапі R&D.

Теоретичну значущість дослідження для подальшого розвитку представляють: модель КСУ ШІ-проєктами; спосіб візуалізації життєвого циклу ШІ-проєкту у вигляді дерева рішень; алгоритм процесу проєктної діяльності на етапі R&D.

Практичну значимість виконане дослідження представляє для невеликих стартап-компаній, що спеціалізуються на створенні нейронних мереж, де процеси комп'ютеризованого управління поки не налагоджені; для усталених компаній подібної спеціалізації для підвищення продуктивності роботи R&D-відділу.

Апробація і публікації результатів. Результати дослідження доповідалися автором на XXII Міжнародній науково-практичній конференції «Сучасні виклики і актуальні проблеми науки, освіти та виробництва: міжгалузеві диспути» (Київ, 19 листопада 2021 р.) і міжнародній науковій інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення: Випуск 63», (м. Тернопіль, 11 листопада 2021 р.), та опубліковані в матеріалах вказаних конференцій (див. додаток А).

1 АНАЛІЗ СИСТЕМ УПРАВЛІННЯ ПРОЄКТАМИ

1.1 Особливості проєктів у сфері інформаційних технологій

Як галузь людської діяльності інформаційні технології займаються створенням, впровадженням, експлуатацією і розвитком інформаційних систем, під якими розуміють системи обробки інформації та відповідні організаційні ресурси (людські, технічні, фінансові тощо), які забезпечують і поширюють інформацію.

Можна виділити наступні напрямки проєктів у сфері інформаційних технологій:

– проєкти розробки і розвитку програмного забезпечення (ПЗ);

– проєкти впровадження інформаційних систем (ІС);

– інфраструктурні та організаційні проєкти.

Особливості проєктів розробки і розвитку програмного забезпечення полягають в тому, що така розробка здійснюється в межах методологій, методів і підходів програмної інженерії. Програмна інженерія (Software Engineering) – це інженерна дисципліна, пов’язана з усіма аспектами виробництва програмних засобів від початкових стадій створення специфікації до підтримки системи після здачі в експлуатацію.

Під моделлю програмного процесу розуміють спрощений опис програмного процесу, представлене з певної точки зору. Моделі завжди є спрощеннями. Метод програмної інженерії – це структурний підхід до створення ПЗ, спрямований на створення ефективного продукту найприбутковішим (cost-effective) шляхом. Практично всі методи побудовані на ідеї створення графічних моделей системи з подальшим використанням цих моделей в якості специфікації або архітектури системи.

Основні типи моделей програмного процесу:

– модель технологічного процесу (workflow model) – показує послідовність дій, поряд зі входами, виходами і залежностями;

– модель потоків даних (data flow or activity model) – відображає процес у вигляді набору дій, кожен з яких виконує певне перетворення даних. У цій моделі дії можуть бути нижчого рівня, ніж в попередній моделі;

– модель роль/дія (role/action model) – показує ролі людей, що беруть участь у програмному процесі, а також дії, за які вони відповідають.

Основною відмінністю ІТ-проєктів від проєктів, що реалізуються в інших сферах людської діяльності, наприклад, у будівництві чи виробництві є те, що проєктне управління в ІТ має справу з невловимими результатами в інформаційному просторі. Крім того, ІТ-проєкти мають низку властивих лише їм чинників, що впливають на успішність виконання проєкту.

У процесі управління ІТ-проєктом, менеджмент проєкту крім питань управління, властивих звичайним проєктам: дедлайни, обмеження бюджету та брак людей, які можуть бути задіяні в проєкті, стикається з необхідністю вирішення унікальних технологічних питань, пов’язаними з технічними засобами, операційною системою, програмним забезпеченням , проблемами з базами даних і т.д.

У зв’язку з тим, що результат ІТ-проєкту (наприклад, впровадження ІС) невловимий – його неможливо виміряти в загальноприйнятих одиницях виміру, уявити в просторі, відчувати фізично – вимоги до результатів проєкту та планування робіт повинні бути максимально докладні. На відміну від проєктів інших галузей у ІТ-проєктів відсутні нормативи витрат для типових операцій, та й самі типові операції є лише дуже укрупнені та в порівнянні зі схожими проєктами.

З огляду на те, що сучасні інформаційні технології в принципі є елементом отримання конкурентної переваги, багато організацій намагаються впровадити інформаційні системи в якомога стислі строки, не проводячи детального планування і постановки завдання з інформатизації, що вкрай негативно позначається на результатах проєкту.

Можна стверджувати, що особливостями ІТ-проєктів є їхня підвищена складність і вищий ступінь ризику. Складність ІТ-проєкту залежить від наступних чинників:

1. Організаційний обсяг проєкту.

Організаційний обсяг проєкту стосовно ІТ-проєктів із впровадження ІС визначає кількість відокремлених організаційних одиниць, в яких буде здійснено впровадження/тиражування ІС. Під відокремленою організаційною одиницею тут розуміється і окрема, як юридична особа організація, і окремий підрозділ в межах конкретної організації – юридичної особи. Показник практично лінійно впливає на вартість робіт. Розмір і структура організаційного обсягу впливає як на вибір методів управління організаційним обсягом, способів координації робіт, так і на інші аспекти проєкту.

2. Функціональний обсяг проєкту.

Цей показник характеризує набір функціональних можливостей ІС, що входять до складу впроваджуваного рішення. Також можна стверджувати, що функціональний обсяг – це сукупність бізнес-процесів, функцій і операцій, інформатизація яких передбачається завдяки впровадженню ІС. Функціональний обсяг проєкту впровадження ІС – найважливіший показник для оцінювання складності проєкту, оскільки він становить основну частину змісту робіт проєкту, описує об’єкт комп’ютеризації – бізнес-процеси організації. Його зміна спричиняє пропорційну зміну інших показників проєкту, таких як методологічний обсяг, інтеграційний обсяг, вартість проєкту тощо.

3. Методологічний обсяг проєкту.

Методологічний обсяг проєкту описує набір нормативно-регламентуючої документації, яку потрібно розробити або доопрацювати в ході реалізації проєкту. До такої документації належать регламенти виконання комп’ютеризованих процесів і операцій, інструкції користувачів, методики виконання конкретних операцій і процесів та ін.

4. Інтеграційний обсяг проєкту.

Впроваджувана інформаційна система може здійснювати обмін даними з іншими ІС. Особливістю інтеграційного обсягу проєкту є необхідність координації робіт з власниками та розробниками суміжних ІС. Інтеграційний обсяг робіт проєкту передбачає виявлення і формалізацію таких інформаційних взаємодій, розробку механізмів здійснення інформаційного обміну та їх впровадження.

У випадку, якщо потреби в інтеграції впроваджуваної системи з іншими ІС були виявлені своєчасно до початку проєкту і роботи з інтеграції були включені в проєкт, то вартість цих робіт буде закладено в бюджет проєкту. Однак, якщо необхідність розробки інтеграційних рішень була виявлена вже в ході реалізації проєкту і такі роботи спочатку не були заплановані, може виникнути ризик виходу за бюджет і зриву строків впровадження системи в цілому. На жаль, практика показує, що найчастіше керівники проєктів нехтують такими очевидними речами, як проведення детального обстеження поточного стану об'єкта інформатизації та детального опису вимог до цільового стану об'єкта саме в частині інтеграційного обсягу.

5. Стандартизація і перенесення даних.

Ще однією особливістю ІТ-проєктів є необхідність у переносі даних з успадкованих систем (у випадку, коли це не первинна комп'ютеризація) і визначення правил роботи з даними у впроваджуваній системі. Оскільки призначення ІС – робота з даними, то від якості та повноти передачі залежить і якість роботи системи в цілому. Для проєктів впровадження ІС очищення, стандартизація та перенесення даних – ключовий етап робіт, котрий значно визначає успіх проєкту.

6. Забезпечення інформаційної безпеки.

Впроваджувана система може містити дані, що представляють комерційну таємницю організації чи відповідно до законодавства мають бути захищеними (наприклад, фінансові чи персональні дані). Роботи щодо забезпечення інформаційної безпеки, якщо вони включені в проєкт, збільшать його вартість і складність, оскільки вони самі по собі досить складні та повинні враховувати забезпечення інформаційної безпеки на рівні робочих місць користувачів, серверів і місць зберігання даних, мережі.

В межах проведення робіт із забезпечення інформаційної безпеки потрібно розробити нормативно-регламентуючі документи, а ІС у випадку обробки даних, захист яких має бути забезпечений згідно із законодавством, повинна бути сертифікована державними органами для обробки таких даних. Зазвичай, забезпечення інформаційної безпеки виокремлюють в окремий напрямок робіт або

в підпроєкт, що має свій організаційний, функціональний, методологічний, інтеграційний та інші обсяги робіт, специфічні чинники складності та інструменти управління такими роботами.

Перераховані основні чинники впливають на складність ІТ-проєктів і присутні у всіх проєктах такого виду, однак остаточний набір чинників, їхня вага і критичність залежать від специфіки кожного конкретного проєкту.

Менеджмент ІТ-проєкту повинен враховувати чинники, що визначають складність проєкту, надавати керуючий вплив на них, застосовуючи специфічні для цього виду проєктів інструменти управління.

Хоча ІТ-проєкт має всі головні характеристики проєкту: мета, строки, обмежені ресурси, команда, однак, володіє і власною специфікою:

– між замовником (бізнесом) і виконавцем (ІТ) проєкту часто виникають проблеми комунікації;

– проблема з формулюванням кінцевого результату, потрібна робота з трансформування вимог бізнес-сторони в технічне завдання;

– у зв’язку з попереднім пунктом, відповідальність за успіх проєкту лягає як на виконавця, так і на замовника, від якого вимагається співпраця і чітке розуміння завдань, які має вирішити ІТ-проєкт;

– ІТ-проєкт нерідко пов’язаний зі структурною зміною в компанії, через що можуть виникати конфліктні ситуації;

– результати ІТ-проєкту впливають на декілька підрозділів компанії;

– висока вартість проєкту;

– у більшості випадків (90%) проєкти є типовими.

Для реалізації ІТ-проєктів характерне наступне:

– множинність ІТ-проєктів у компанії;

– важливість окремого ІТ-проєкту може змінюватися залежно від інших показників діяльності компанії;

– вимоги до результату і зміст проєкту змінюється в ході його реалізації, а, тому виникають складності планування, які збільшують загальну ризикованість ІТ-проєктів;

– потрібні кваліфіковані трудові ресурси різних профілів (які можуть бути задіяні в декількох проєктах);

– очевидність прогресу реалізації;

– роботи часто виконуються за принципом «снігової кулі», а не по кроках.

Для результатів такого роду проєкту характерні:

– складність визначення ефективності в грошовому еквіваленті;

– взаємозв’язок з іншими проєктами компанії.

За призначенням ІТ-проєкти можуть бути [1]:

– стратегічними (спрямовані на майбутнє, мають високу важливість для сьогодення);

– підтримуючими поточні операції (мають високу важливість у сьогоденні);

– обов’язковими (вимоги стандартів, критичні для роботи компанії);

– інноваційними (важливі для майбутнього благополуччя компанії).

Загальні технології управління проєктами застосовні й до ІТ-проєктів (розробка цілей і результатів проєкту, строків, бюджету, угоди про якість). ІТ-проєкти можливо планувати, їхніми ризиками можливо управляти [1, 2].

У зв’язку з великою вартістю і складністю ІТ-проєктів, доцільно розглянути процедури управління ризиками. У «Зводі знань з управління проєктами» [22] виділяють шість кроків, включених у процес управління ризиками:

– планування;

– ідентифікація ризиків (зазвичай виконується);

– якісна оцінка ризиків (зазвичай виконується);

– кількісна оцінка (в грошовому вираженні зазвичай не рахується);

– планування реагування на ризики (частково виконується);

– моніторинг та контроль ризиків.

Типовими ризиками ІТ-проєктів є зрив строків (і подальше перевищення вартості чи підвищення навантаження на співробітників, що спричиняє погіршення якості), невідповідність бажаного і фактичного результатів і т.д.

Звичайними причинами для цього є некваліфікований менеджмент і неготовність компанії до змін разом із недостатньо чітко сформульованими вимогами.

Щоб уникнути реалізації основних ризиків ІТ-проєктів, рекомендується документування на всіх етапах, достатній бюджет для залучення кваліфікованих співробітників (включаючи аутсорсинг), різне заохочення комунікації та регулярне обговорення ходу виконання проєкту між замовниками та виконавцями.

У зв’язку зі згаданими особливостями ІТ-проєктів, їхній моніторинг повинен бути постійним (періодичним, з невеликим інтервалом між збором і аналізом даних), а у випадку негативних трендів варто приймати впевнені управлінські рішення.

Підводячи підсумок, можна стверджувати, що ІТ-проєкти є складними, високо ризикованими і різноманітними [3].

1.2 Моделі життєвих циклів проєктів і методи управління проєктами

Управління проєктами як вид професійної діяльності та як об’єкт наукових досліджень отримав значного розвитку в 1980-х роках, коли світова економіка виходить з кризи, зростає насичення ринку і виникає необхідність вирішення нових, великих завдань. Саме тоді виходить перша значна робота PMBoK (A Guide to the Project Management Body of Knowledge) [22], виконана в Project Management Institute (PMI). Ця організація на сьогоднішній день є найавторитетнішою професійною асоціацією, що розробляє стандарти в галузі управління проєктами. Серед інших інститутів, що займаються стандартизацією проєктного менеджменту, варто назвати IPMA (International Project Management Association), OGC (The Office of Government Commerce, стандарти PRINCE2), ISO (International Standartization Organization), APM (Association for Project Management) й інші. Варто також зазначити, що крім міжнародних стандартів існують й національні системи стандартизації, галузеві та корпоративні.

Розвиток ІТ-сфери, інтенсивна робота в галузі програмного забезпечення призвели до накопичення (і цей процес не зупиняється) великого практичного

досвіду («best practice»). Комплекс таких «кращих практик», що реалізуються на різних стадіях життєвого циклу проєкту і базуються на загальній ідеології, стандарт SWEBOK [4] називає «методологія розробки програмного забезпечення». Методології, або методи розробки програмних продуктів на сьогоднішній день є областю інформаційних технологій, що найшвидше розвивається, так як спираються на реальні практичні знання. Методи управління проєктами тісно пов'язані з життєвим циклом проєкту, тобто вони взаємно обумовлюють один одного. Фази життєвого циклу проєкту (ініціація, планування, виконання, завершення) є досить узагальненими і однаково притаманними проєктам, реалізованим, наприклад, в будівельній сфері, де всі етапи чітко визначені і зрозуміла послідовність різних видів робіт, або, наприклад, в авіаційній чи космічній галузі, де можливе розпаралелювання операцій, або повторення (якщо є необхідність) одних і тих же видів робіт [5-8]. Тому при розробці конкретного програмного продукту необхідна докладна деталізація його життєвого циклу і адекватна (змістом проєкту) модель.

«У загальному випадку, життєвий цикл визначається моделлю і описується у формі методології (методу). Модель або парадигма життєвого циклу визначає концептуальний погляд на організацію життєвого циклу і, часто, основні фази життєвого циклу і принципи переходу між ними. Методологія (метод) задає комплекс робіт, їх детальний зміст і рольову відповідальність фахівців на всіх етапах обраної моделі життєвого циклу, зазвичай визначає і саму модель, а також рекомендує практики (best practices), що дають змогу максимально ефективно скористатися відповідною методологією і її моделлю» [9].

Традиційними і хронологічно першими були розроблені каскадна (водоспадна, waterfall) модель, вперше описана в 1970 р. у роботі В. Ройса [10] і спіральна (Spiral), запропонована Б. Боемом в 1986 р. [11]. Пізніше, з ускладненням інформаційних технологій і наростаючим функціоналом програмних продуктів виникли так звані гнучкі (Agile) моделі [12]. Саме вони набули найбільшого поширення останнім часом.

Розглянемо сучасні методи і алгоритми управління проєктами і проведемо їх порівняльний аналіз за найістотнішими параметрами.

І. Каскадна модель (Waterfall).

Найстарішою і найвідомішою моделлю побудови багаторівневого процесу розробки є каскадна (або водоспадна) модель: в ній кожен етап розробки, відповідний стадії життєвого циклу ПЗ, продовжує попередній. Тобто новий етап починається лише після повного завершення поточного.

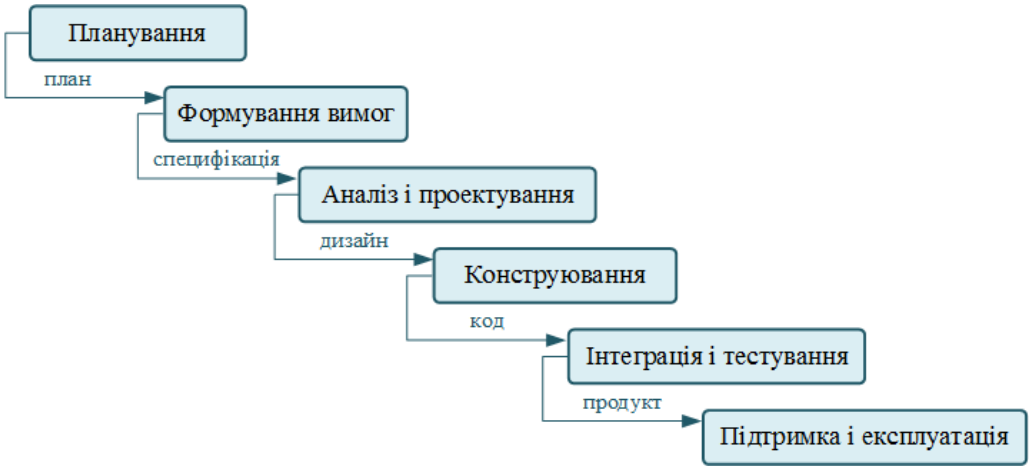

Рисунок 1.1 – Каскадна модель життєвого циклу

Каскадна модель проста і зрозуміла, але не так практична, як раніше. В умовах динамічно змінюваних вимог, строго структурований процес може з переваги перетворитися в перешкоду на шляху успішного завершення розробки системи. Тому сьогодні водоспадна модель застосовується переважно великими компаніями для великих і складних проєктів, які передбачають всеосяжний контроль ризиків [13].

Переваги каскадної моделі:

– зрозуміла і чітка схема робочого процесу;

– можливість прорахунку точної кількості витрачених на проєкт ресурсів;

– не вимагає витрат з налагодження комунікацій між усіма членами команди.

Недоліки каскадної моделі:

– пріоритет формального підходу до послідовності процесу роботи;

– неможливість внесення змін замовником до закінчення розробки продукту;

– у випадку нестачі ресурсів страждає якість проєкту через скорочення етапу тестування.

Незважаючи на те, що каскадна модель все ще використовується, вона вже втратила колишні позиції. Сьогодні їй на зміну приходять більш просунуті моделі і методології розробки програмного забезпечення.

II. Спіральна модель

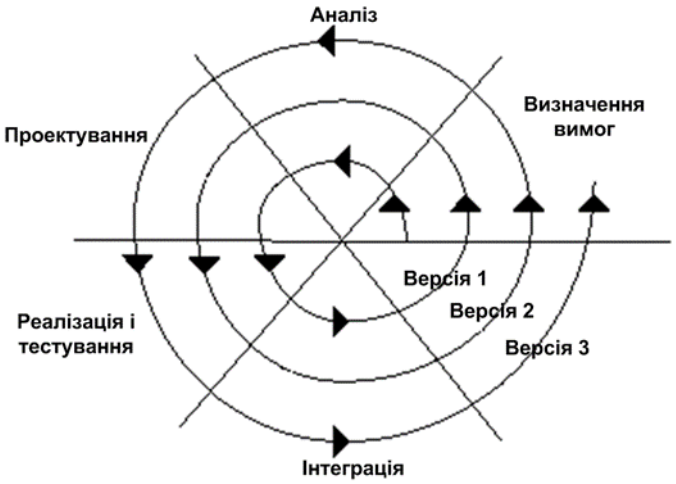

Рисунок 1.2 – Спіральна модель життєвого циклу

Спіральна модель використовує розбиття проєкту на ітерації. Підвищена увага приділяється початкових етапах розробки – аналізу і проектування. Основна функція початкового етапу – обґрунтування можливості реалізації програмного рішення; для цього створюється прототип майбутнього продукту [14].

Свою назву спіральна модель отримала внаслідок того, що життєвий цикл проєкту в цій моделі завдяки ітераційному підходу створює образ спіралі – з подоланням кожної ітерації, або окремого циклу, майбутній програмний продукт набуває необхідної форми і змісту.

Такий підхід дає змогу на кожній ітерації приділяти час уточненню цілей і параметрів проєкту, з’ясуванню виникаючих труднощів, плануванню робіт на наступному витку.

Завдяки такому підходу до розробки час виконання всіх поставлених завдань зменшується.

Переваги спірального підходу:

– подолання жорсткого, регламентованого процесу розробки в каскадної моделі; створення додаткових можливостей;

– полегшення процесу внесення коректив до проєкту, якщо вимоги замовника змінилися;

– постійний зв’язок користувача майбутнього продукту з розробником, що дає змогу оперативно відслідковувати виникаючі помилки або слабкі місця проєкту;

– ефективніше управління проєктом завдяки наявності ітерацій;

– зниження рівня ризиків.

Труднощі використання спіральної моделі:

– відсутність обмежень часу на кожному з етапів життєвого циклу;

– перевага, що є можливість постійно вносити зміни до вимог до проекту, обертається тим, що кількість циклів може рости, отже, подовжуються строки розробки;

– досить складна структура моделі, що створює труднощі для розробників і менеджерів при її застосуванні.

III. Гнучкі (ітеративні) моделі (Agile).

Крім послідовних моделей, існують ітеративні (інкрементні). Це ціле сімейство методологій, об’єднаних загальним підходом в управлінні проєктами:

– люди і взаємодія важливіші процесів та інструментів;

– працюючий продукт важливіший вичерпної документації;

– співпраця з замовником важливіша узгодження умов контракту;

– готовність до змін важливіша проходження попереднім планом [15, 16].

У таких моделях життєвий цикл проєкту розбитий на кілька міні-циклів, кожен з яких складається з базових стадій моделі життєвого циклу. Ці міні-цикли називаються ітераціями. Розробка окремого компонента систем відбувається всередині ітерації, потім цей компонент додається до раніше розробленого функціоналу. Ітеративна модель не потребує повного обсягу вимог для початку робіт над продуктом. Розробка програми може починатися з вимог до частини

функціоналу, причому згодом вимоги можуть зазнавати зміни і доповнення. Шляхом повторення процесу відбувається оновлення версії продукту на кожному циклі.

Рисунок 1.3 – Ітеративна модель життєвого циклу

Рішення про використання результатів попередньої ітерації в якості вхідних для наступної приймається після закінчення попередньої (т.зв. інкрементне прототипування). В кінцевому підсумку, досягається точка, в якій всі вимоги були втілені в продукті – відбувається реліз. Використовуючи математичну термінологію, ітеративна модель реалізує послідовну апроксимацію, тобто наближення до заданих параметрів, тобто готового продукту. Успіх використання цієї моделі визначається строгою валідацією вимог і верифікацією функціональності на кожній ітерації. Основні стадії процесу розробки в ітеративній моделі фактично повторюють модель водоспаду. У кожній ітерації створюється програмне забезпечення, яке потребує тестування на всіх рівнях.

Переваги ітеративної моделі:

– високий рівень взаємодії між членами команди проєкту;
– швидкий результат (робочий код) в результаті «спринтів»;
– стимулювання зміни і поліпшень продукту під час його розробки;
– безпосереднє залучення замовника в робочий процес;
– висока здатність до адаптивності під різні процеси і умови;
– швидкий відгук на зміни зовнішніх і внутрішніх умов проєкту;

– пристосований для розробки інноваційних продуктів з високим ступенем невизначеності і недостатньою інформативністю;

Недоліки:

– ризик нескінченних змін продукту;

– велика залежність від рівня кваліфікації та досвіду команди;

– практично неможливо точно підрахувати підсумкову вартість проєкту;

– ефективність застосування Agile-методик істотно залежить від кваліфікації менеджменту (для полегшення застосування підходу прийнято використовувати методи Scrum, Kanban та інші).

Найбільш затребуваними представниками сімейства Agile-методології є Scrum, Lean, Canban, Six Sigma, PRINCE2. Розглянемо докладніше Scrum і Canban.

III.1. Scrum – найструктурованіший у групі Agile-методів.

Спрощена схема роботи за Scrum полягає в наступному. Згідно з технологією Scrum, проєкт розбивається на частини – завдання, які будуть створюватися протягом проєкту. Після цього отриманим частинам присвоюється свій пріоритет.

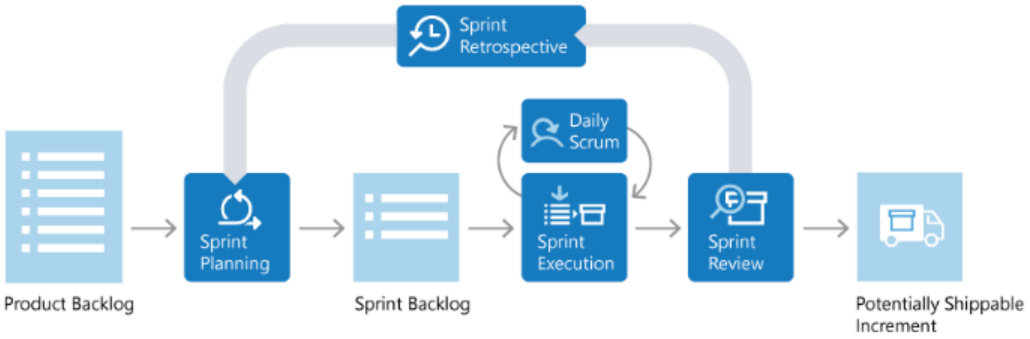

Рисунок 1.4 – Scrum модель життєвого циклу

Хронологічно проєкт розбивається на ітерації – певні проміжки часу, що встановлюються командою і узгоджені з замовником для поточного контролю і управління проєктом. Найважливіші завдання першими відбираються для виконання в ітерації. За підсумками ітерації замовник отримує робочий проміжний варіант продукту. Щоб переконатися у відповідності проєкту вимогам замовника, перед початком будь-якого спринту потрібно виконувати переоцінку ще не

виконаного змісту проєкту і вносити в нього зміни. Участь в цьому процесі приймає керівник проєкту, команда та ініціатор. Відповідальність розподіляється на всіх учасників.

Процес роботи в середовищі Scrum спирається на п’ять виробничих нарад:

– нарада-упорядкування; в перший день нової ітерації відбувається обговорення того, що вже зроблено за проєктом, що ще належить виконати;

– нарада-планування; команда вибирає зі списку пріоритетних завдань ті, які вона буде виконувати протягом поточної ітерації;

– наради-летючки; відбуваються щодня, оперативно (до 15 хвилин) для обміну відомостями про поточний стан виконання індивідуальних завдань кожного члена команди;

– підсумкові наради; команда демонструє свій результат всім зацікавленим особам; з’ясовується, наскільки продукт відповідає цілям і очікуванням;

– нарада-ретроспектива. Проводиться аналіз того, наскільки злагоджено працювала команда, обговорюються виникли проблеми методів і комунікації, робляться певні висновки.

Переваги Scrum:

– зручний для проєктів, в яких можливий швидкий, оформлений і придатний для початкового використання, результат;

– тісна і постійна комунікація співробітників допускає присутність в команді малодосвідченого розробника;

– швидке виправлення помилок завдяки миттєвому зворотного зв’язку між усіма учасниками проєкту.

Недоліки Scrum:

– підходить не кожному темпераменту; наявність постійної і дуже тісної комунікації всіх учасників проєкту висуває вимоги до психологічних властивостей і соціальних навичок учасників проєкту;

– не підходить для проєктів із строго регламентованих і нормованих областей (наприклад, юриспруденція).

III.2. Kanban-метод з групи Agile.

На відміну від Scrum, Kanban-метод не передбачає наявність чітких часових ітерацій. Процес створення продукту не поділяється на ітерації, а передбачає обмеження по кількості одночасно виконуваних завдань. Особливості методу полягають у тому, що немає обмежених за часом ітерацій, регламентованих виробничих нарад, кожен співробітник команди може вирішувати кілька завдань одночасно. Відсутність часових рамок кожного етапу дає змогу призупинити виконання поточного завдання, якщо змінився її пріоритет або з'явилися інші, важливіші завдання.

Переваги Kanban:

– відсутність дедлайнів;

– економічно ефективний внаслідок розумного розподілу кількості завдань на кожного члена команди.

Недоліки Kanban:

– найефективніший в командах, члени яких взаємозамінні;

– непридатний для проєктів, строки виконання яких строго обумовлені.

IV. Crisp-DM. З розвитком методів створення штучного інтелекту постало питання перегляду життєвої моделі циклу проєктів. Програмні продукти, що розробляються на основі штучного інтелекту, істотно відрізняються від стандартних програмних продуктів (тобто таких, які не містять ШІ) в організації структури життєвого циклу. Крім того, такі продукти ґрунтуються на використанні великих масивів даних, а це вимагає інших підходів до управління проєктами.

Платформа CRISP-DM спочатку була створена для Data Mining – глибинного аналізу даних, та згодом виявилося, що вона добре моделює життєвий цикл ШІ-вмістимих програмних продуктів. Зміст моделі CRISP-DM життєвого циклу дослідження даних полягає в інтерпретації та обробці даних. Усередині процесу побудови моделі Data Science не відбувається накопичення результатів як при ітеративних, спіральних і інших моделях. Кожен новий цикл оновлює (або навіть обнулює) результати, отримані в попередньому циклі. Розглянемо докладніше модель CRISP DM. У загальному вигляді вона представлена на рис.1.5.

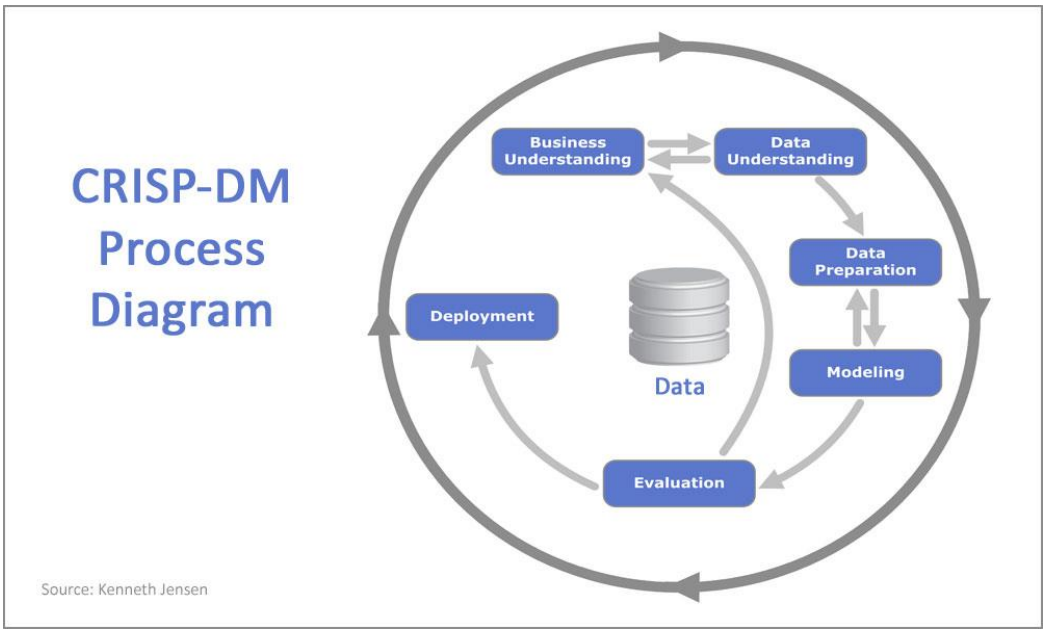

Рисунок 1.5 – Модель життєвого циклу дослідження даних CRISP-DM

Модель життєвого циклу CRISP-DM містить шість фаз:

1) розуміння бізнес-цілей (Business Understanding). Змістом даної фази є наступне: дослідження бізнес-процесів компанії і висунення пропозицій щодо аналізу даних виходячи з кінцевих цілей аналізу. В обговоренні бере участь максимальна кількість зацікавлених осіб, у кінцевому підсумку виробляють план аналітичного проєкту. Вирішується також питання про доцільність проєкту;

2) початкове вивчення даних (Data Understanding): докладне вивчення наявних даних з метою усунення проблем на стадії підготовки даних. В результаті організовується доступ до даних, їх дослідження з використанням таблиць і графіків, оцінювання якості даних, розробка необхідної документації;

3) підготовка даних (Data Preparation): найбільш трудомісткий і відповідальний етап, який поглинає 50-70% часу і ресурсів. Підготовка даних – це їх консолідація; формування вибірок; агрегування; збагачення; очищення; поділ даних на навчальні та тестові;

4) моделювання (Modeling): побудова та впровадження аналітичних моделей, як правило, протягом декількох ітерацій;

5) оцінювання (Evaluation) відповідності результатів проєкту критеріям. Рішення про відповідність приймається найвідповідальнішими особами компанії, які формулюють бізнес-цілі;

6) впровадження (Deployment) як процес використання нових ідей і знань для підвищення ефективності діяльності підприємства.

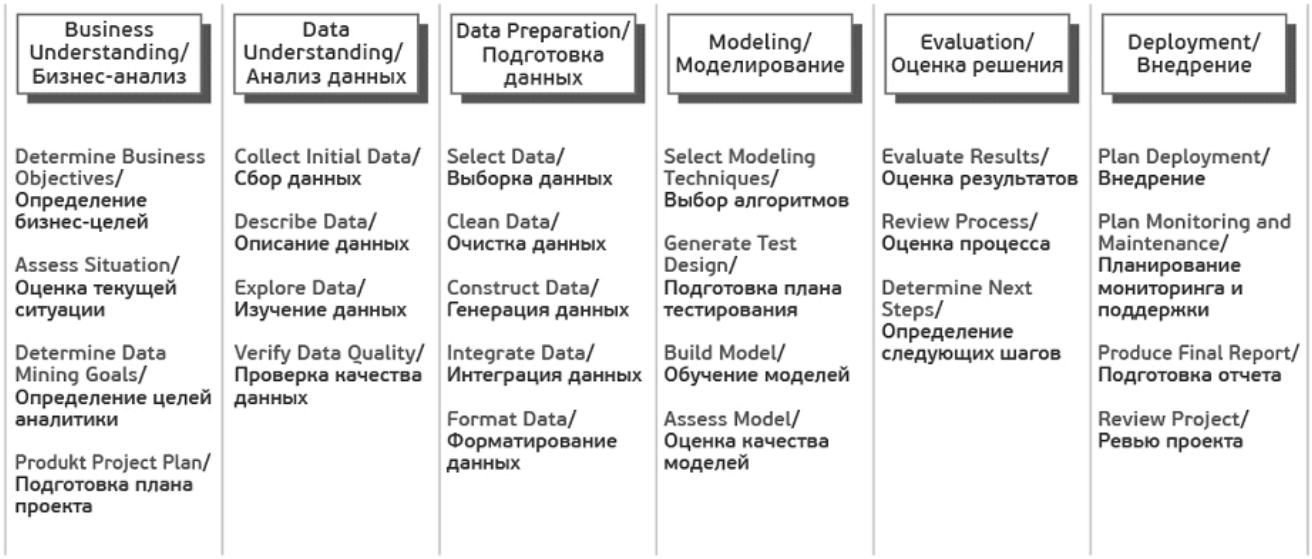

Рисунок 1.6 – Завдання фаз Crisp-DM

На рис.1.8 показані завдання кожної фази життєвого циклу проєкту, що розробляється на основі нейронної мережі.

До переваг даної моделі можна віднести:

– нейтральність щодо предметних областей;

– зручність використання в проєктах, що використовують ШІ;

– одна з небагатьох моделей, що містить інтелектуальний аналіз даних;

– є одним з найважливіших понять для технологій великих даних (Big Data).

Недоліки моделі наступні:

– відсутність інструментів управління проєктами;

– відсутній критерій часу для кожного з циклів; цикли можуть бути різної тривалості.

1.3 Аналіз інструментів управління проєктами

Інструменти, використовувані менеджером проєкту, можуть застосовуватися як протягом всього проєкту, так і на певному етапі (фазі) життєвого циклу. Під

інструментами управління ми будемо розуміти закінчені формалізовані методики, процедури, а також шаблони необхідних проєктних документів.

Програмне забезпечення Trello для управління Agile-проєктами. Trello – хмарна програма для управління проєктами невеликих груп, розроблена Fog Creek Software. Це безкоштовна багатоплатформенна система управління проєктами (рис. 1.7, 1.8).

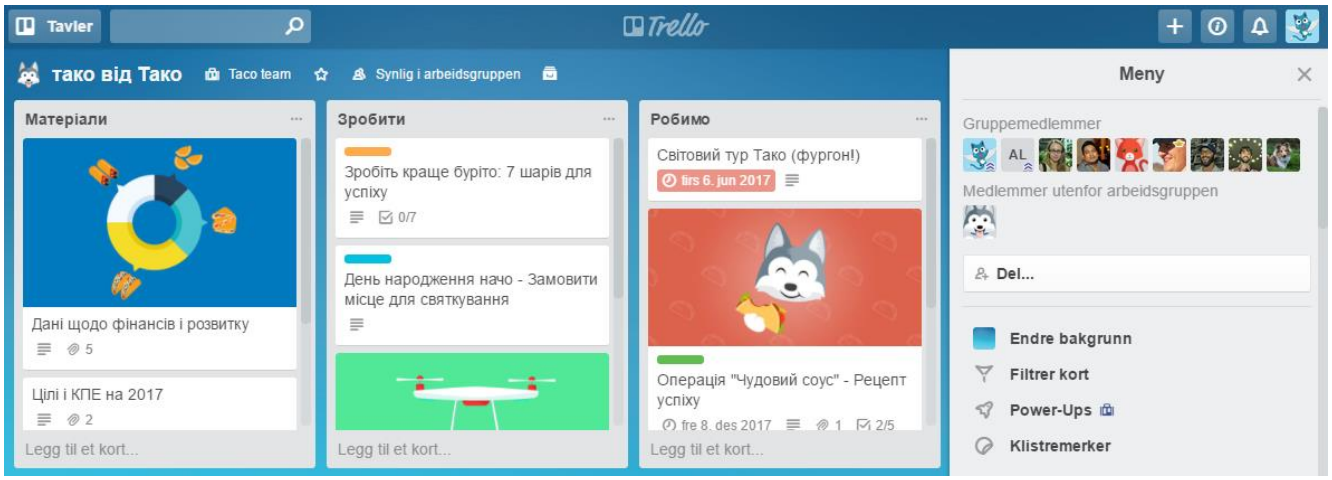

Рисунок 1.7 – Trello Board

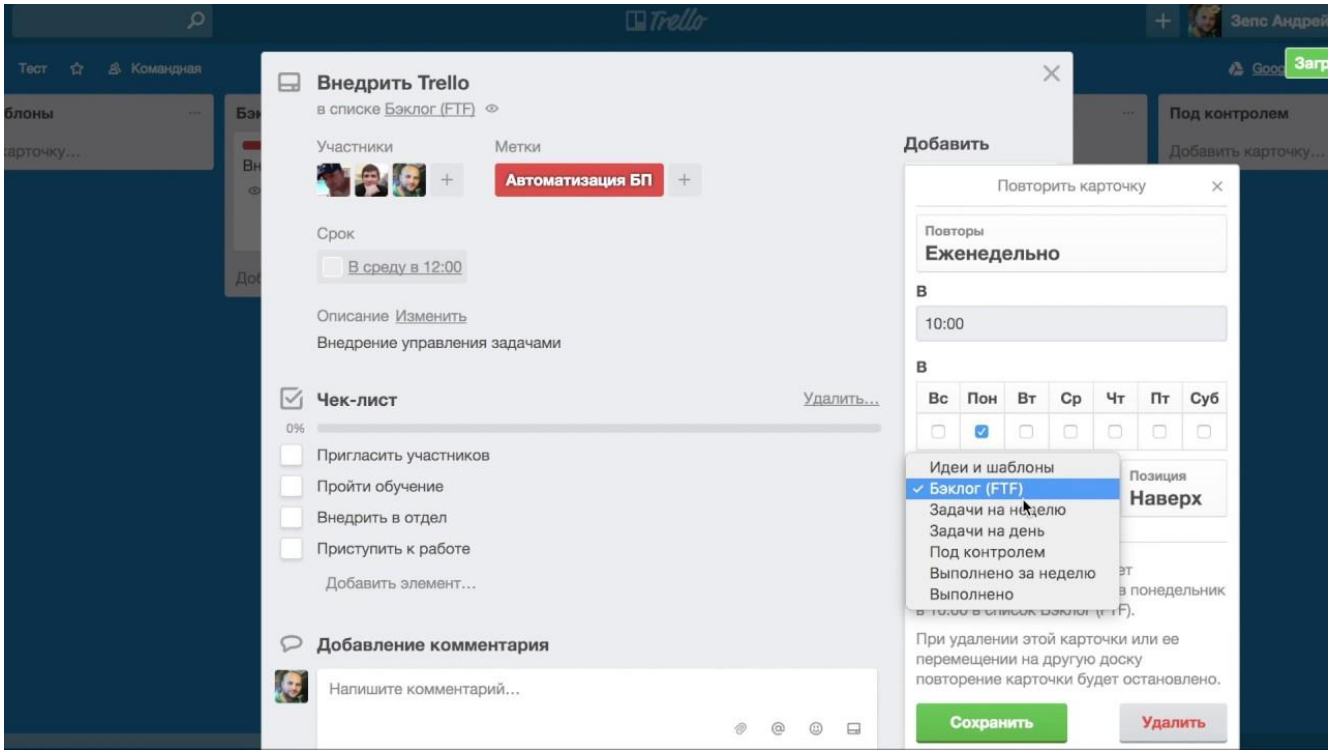

Рисунок 1.8 – Картки завдання в Trello

Вона використовує парадигму управління проектами, відому як Канбан. Проєкти зображуються дошками, що містять списки. Списки містять карти, якими зображуються завдання. Картки повинні переходити з попереднього списку до наступного (за допомогою перетягування), таким чином зображуючи рух якоїсь функції від ідеї до тестування.

Картці може бути присвоєно ім’я відповідальних за неї користувачів. Користувачі і дошки можуть об’єднуватися в команди.

Trello має обмежену підтримку тегів у вигляді шести кольорових міток. Картки можуть містити коментарі, вкладення, дату завершення та переліки (списки підзадач).

Форматуються картки розміткою Markdown [17].

Переваги роботи з Trello полягають в наступному:

– наявність функціональних можливостей, що спрощують процес управління проєктом, а саме: обмін файлами (включаючи фотографії та відео) зі своїми членами команди; коментарі до карти; стеження за to-do list; встановлення кольорових міток відповідно до пріоритетом; необмежене створення карт під проєкт;

– вартість. Порівняно з іншими інструментами управління проєктами, Trello має простішу побудову ціноутворення. Простіший варіант версії дає змогу запросити необмежену кількість учасників, створити дошки, карти і списки. Версія бізнес-класу, вартість якої 25 доларів США в місяць, надає низку функцій, таких, як інтеграція в Служби Google, легкий об’ємний експорт, а також можливість доступу і керування всіма дошками [18];

– зрозумілий моніторинг часу. Ви ніколи не пропустите дату завершення проєкту або важливу зустріч з Trello Борд. При створенні Cards можна додати відповідні дати (deadline). Картка стане жовтою за 1-2 дня до закінчення проміжку часу, відведеного на виконання, картка стане повністю червоною в день здачі проекту;

– mobile friendly. Trello працює на кожній платформі. Незалежно від того, чи перебуває він на комп'ютері, планшеті чи телефоні, інструмент може адаптуватися під розмір екрану;

– instant notifications. Ви завжди будете в курсі новин, так як Trello оснащена функцією миттєвого оповіщення при оновленні, коментуванні або видаленні завдання. Також можна отримувати сповіщення по електронній пошті.

Таблиця 1.1 – Огляд Trello

Призначення продукту	Відстеження завдань проєкту та їх статусів. Управління невеликими проєктами
Users	Software developers Project managers SCRUM masters
Use Cases	Управління проєктами Менеджмент продукту Управління завданнями Розробка програмного забезпечення Agile розробка програмного забезпечення
Інтеграція	Confluence JIRA Slack GitHub
Hosting options	Cloud

Недоліки роботи з Trello наступні:

– відсутній offline-доступ. Trello не працюватиме, якщо немає wi-fi. Якщо ви подорожуєте або ваш електронний пристрій переключено на режим «у літаку», ви не отримаєте доступу до Trello. Доступ до сайту Trello вимагає підключення до Interwebs. Ці особливості, властиві Trello, створюють певний дискомфорт для користувача, що знаходиться оф-лайн;

– ця платформа не призначена для роботи над великими проєктами. Оптимальне її застосування – стартапи, домашні проєкти або деяка ідея, яка

потребує перевірки. Trello – незручний інструмент управління масштабними проєктами з декількома командами з усього світу;

– обмежений обсяг файлосховища. Кількість вкладених файлів на карті Trello може бути довільним, але для кожного вкладеного файлу існує ліміт завантаження файлу до 10 МБ. Члени бізнес-класу і Trello Gold можуть користуватися обмеженням завантаження файлів розміром до 250 Мб. Ліміту зберігання даних облікового запису не існує;

– незручність коментування. Після опублікування і збереження коментаря на мапі редагувати його у випадку потреби неможливо, доведеться писати новий коментар;

– недостатній рівень візуалізації процесу роботи всіх членів команди одночасно;

– неможливо створити довгострокові плани. Дошка проста і зручна для розробки програмного забезпечення невеликих проєктів, але для проєктів тривалістю навіть на два тижні створити дорожню карту виявляється неможливим;

– немає можливості переглядати ітерації. Ефективність роботи підвищується за умови регулярного спостереження за виконанням поточних завдань, продуманого плану контролю майбутніх завдань, а також можливості аналізу і коригування вже виконаної роботи. Trello не має вбудованих інструментів для перегляду та аналізу реалізованих завдань для успішного проведення ретроспектив;

– немає можливості вносити дані про витрачений на завдання час. Відсутність можливості для менеджерів проєкту і його учасників відстежувати кількість часу, витраченого на виконання кожного завдання. Це ускладнює складання щомісячної звітності, а також ускладнює контроль за виконанням завдання.

Програмне забезпечення ASANA для управління Agile-проектами.

Asana (Асана) – один з провідних мобільних і веб-застосунків для управління проєктами в невеликих командах.

Це універсальне рішення для управління проєктами та завданнями, яке дасть змогу автоматизувати деякі з найскладніших завдань, таких, як комунікація і

зручне співробітництво. Багато компаній хвалять Asana за точність і ефективне використання часу, особливо зазначаючи зручність відстеження завдань і обговорення їх в реальному часі.

Основний акцент творці сервісу роблять на тому, що тепер управляти проєктами можливо і без використання електронної пошти. Розробники Asana надали користувачам можливість обговорювати робочі процеси на тій же платформі, де вони були створені, і вдосконалили програму додатковим функціоналом: notes, groups, combined tasks, followers [19].

Asana миттєво повідомляє про кожну зміну, який міг би впливати на проєкт. Програмне забезпечення допомагає зробити якісне оцінювання проєкту і управляти ризиками до моменту підписання контракту з клієнтом.

Сервіс пропонує синхронізацію в режимі реального часу для різних пристроїв; перевірку вашої поштової скриньки; створення плану на прийдешній день; перегляд, редагування та створення завдань і проєктів з пріоритетами; структурування проєктів за допомогою розділів; лайки до завдань; обговорення по проєкту і завданням; настроювані дешборди; календарі; пошук завдань, проєктів, людей і тегів; прикріплення файлів з Dropbox, Box або Google-Диску; канбан-дошки.

Основні характеристики Asana представлені в табл.1.2.

Таблиця 1.2 – Огляд Asana

Призначення продукту	Планування, документування, відстеження і випуск програмного забезпечення
Users	Software developers Project managers SCRUM masters
Use Cases	Управління проєктами Менеджмент продукту Управління завданнями Розробка програмного забезпечення Agile розробка програмного забезпечення
Інтеграція	JIRA Slack GitHub
Hosting options	Cloud

Розглянемо переваги Asana:

– розширений список функцій управління;

– існує безкоштовна пробна версія. Сервіс також безкоштовний для команди до 15 чоловік;

– позначення кольором найважливіших завдань. Зручний функціонал для making priority for task (рис.1.9);

– необмежений функціонал додавання більше одного тега до кожного елементу (рис.1.10);

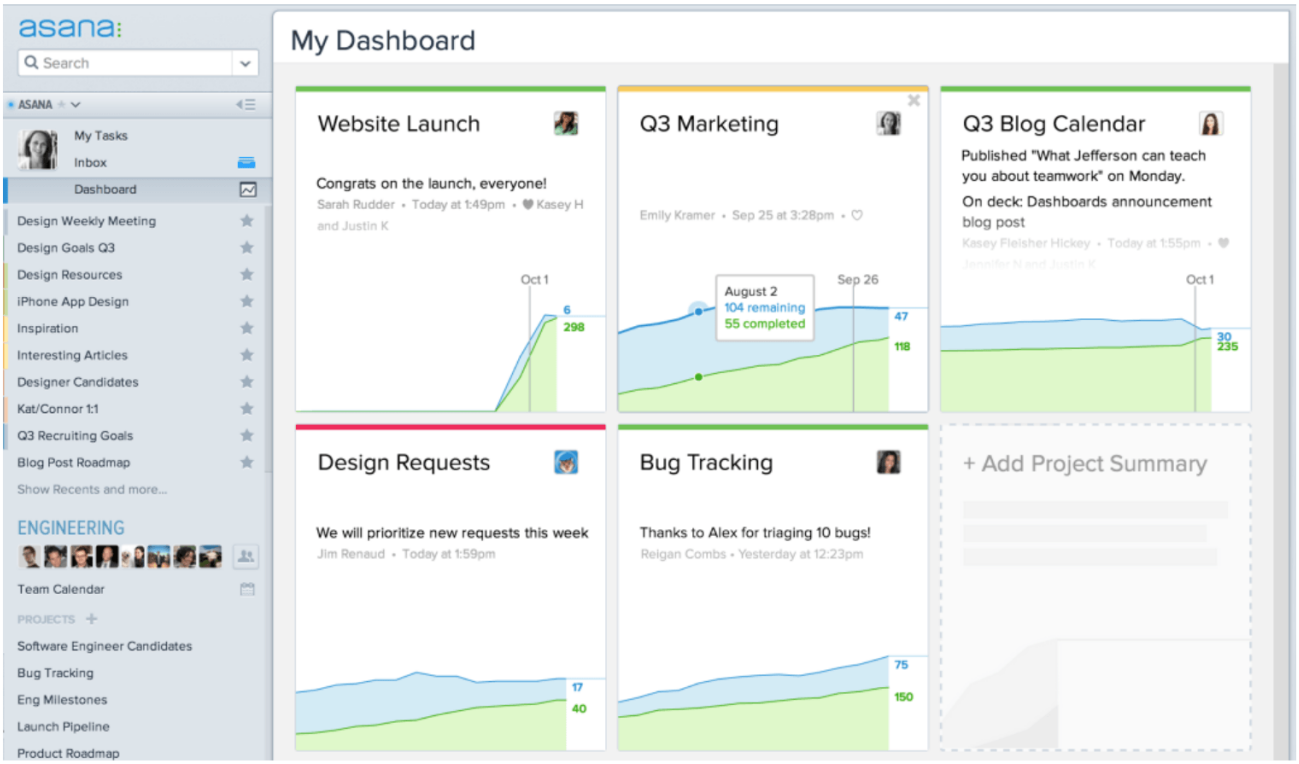

Рисунок 1.9 – Asana Dashboard

– підтримка мобільної версії для різних платформ;

– одна історія / елемент / завдання може мати посилання на більш ніж один проєкт і тег;

– підтримка гарячих клавіш;

– додавання нових завдання без необхідності заповнення багатьох полів;

– інтеграція електронної пошти;

– чітке дерево завдань, які потрібно виконати. Можливість побачити всі завдання в одному місці.

– платформа «Преміум» коштує 10 доларів США за кожного члена / в місяць (в разі, якщо ви хочете мати додатковий і розширений функціонал) [20].

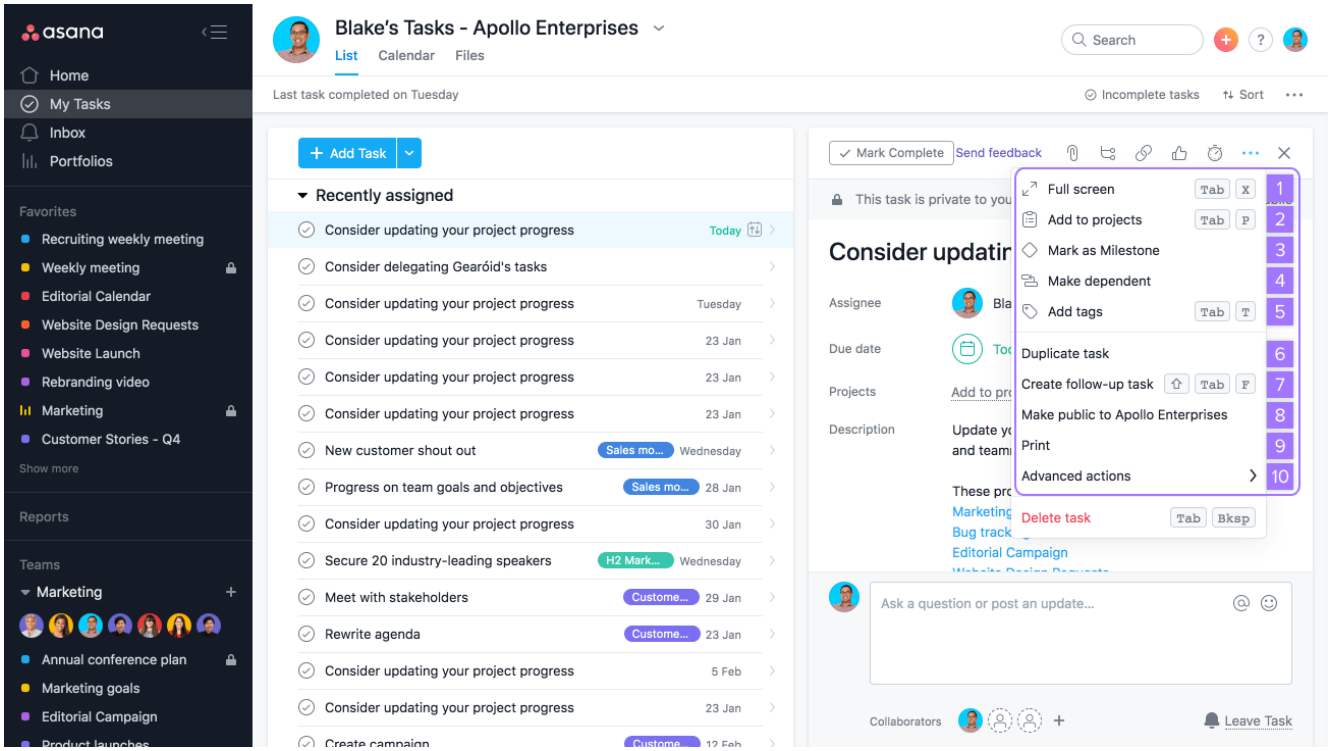

Рисунок 1.10 – Завдання в Asana

Недоліки Asana наступні:

– можна призначити історію / завдання тільки одному з членів вашої команди;

– графічні деталі перевантажують інтерфейс, роблячи його повільним при переміщенні;

– важко працювати з завданнями в завданні (sub-tasks). Є можливість втратити зв'язок при переміщенні елементів;

– надмірна кількість функцій, які відображаються на екрані, ускладнюють і уповільнюють процес навчання.

– повідомлення через програму зашифровані, але, на жаль, не існує двоетапної перевірки. Відсутність two-step verification робить програмне забезпечення небезпечним для ваших особистих даних [20].

Програмне забезпечення JIRA для управління Agile-проєктами.

JIRA – система стеження за вадами. Даний продукт розроблений для організації взаємодії з користувачами, також її можна використовувати і для управління проєктами. Розробник зазначеної системи – компанія Atlassian.

Jira є одним з двох її основних продуктів (поряд з вікі-системою Confluence). Однією з переваг Jira є наявність веб-інтерфейсу [23].

JIRA – це комплекс систем, що дає змогу координувати роботу всіх команд, зайнятих розробкою нового продукту. JIRA пропонує кілька продуктів і варіантів розгортання, призначених спеціально для програмного забезпечення, IT, бізнесу, команд операторів тощо.

Продукти і програми, побудовані на платформі JIRA, допомагають командам планувати, призначати, відстежувати, звітувати і управляти роботою. Платформа JIRA об'єднує команди у всьому: від гнучкої розробки програмного забезпечення і підтримки клієнтів до управління списками покупок і сімейними справами (рис.1.11).

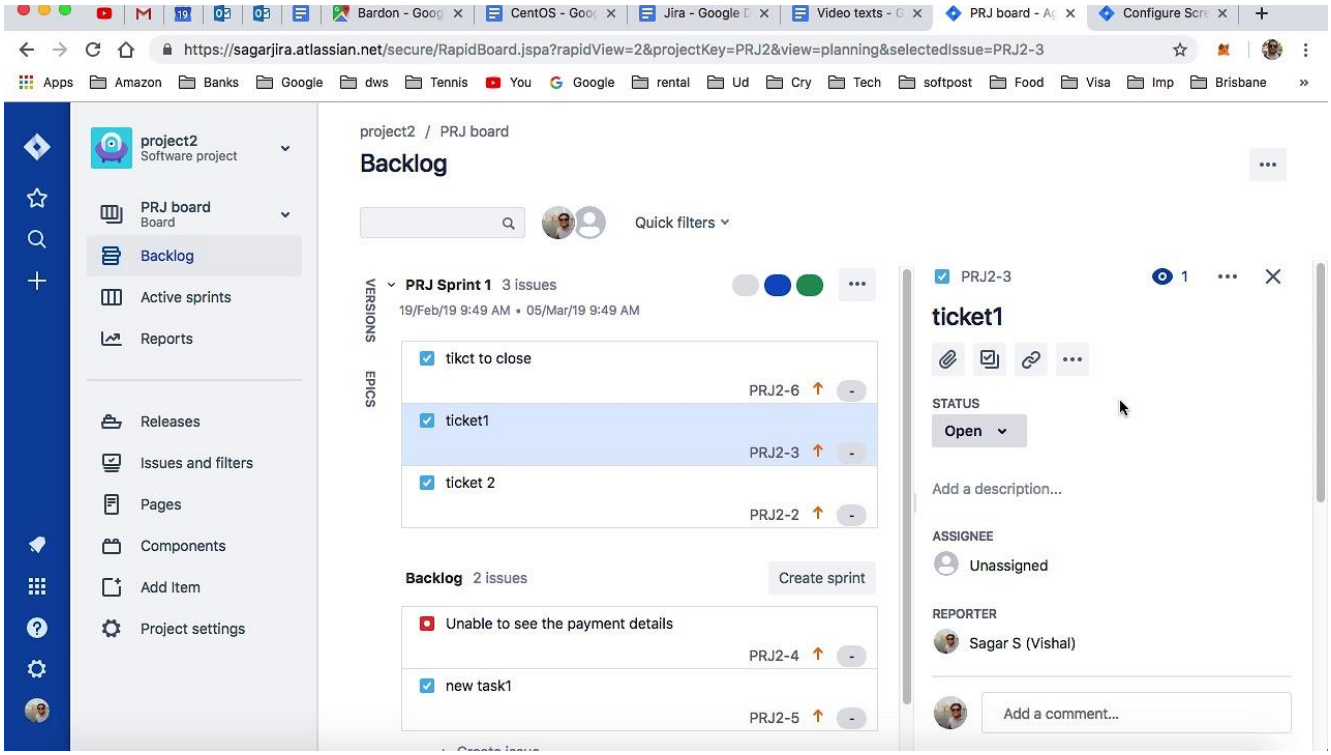

Рисунок 1.11 – Відкрите завдання в JIRA

Чотири продукти побудовані на платформі JIRA: JIRA Software, JIRA Service Desk, JIRA Ops і JIRA Core. Кожен продукт поставляється з вбудованими

шаблонами для різних випадків використання і легко інтегрується, тому команди з різних організацій можуть працювати краще разом.

Розглянемо основні характеристики JIRA Software.

Робочі процеси – це послідовний шлях від зародження ідеї до її реалізації.

Схема базового робочого процесу показана на рис.1.12.

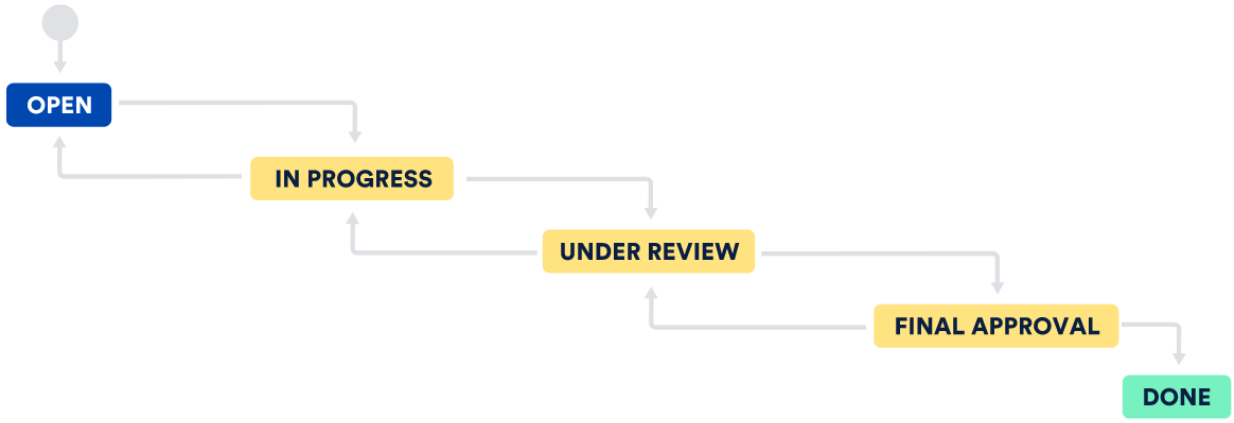

Рисунок 1.12 – Jira Workflow

В цьому випадку Open, Done і мітки між ними відображають статус, в якому може виникнути проблема, а стрілки – потенційні переходи від одного стану до іншого.

Робочі процеси можуть бути простими або складними, з умовами, тригерами, валідаторами і функціями повідомлення.

Рекомендується для адміністраторів, які починають використовувати JIRA Software, зберігати максимально простий варіант своїх робочих процесів до тих пір, поки при реалізації проєкту не виникне необхідність в ускладненні конфігурації робочого процесу [21].

Табл.1.3 містить відомості про важливі особливості продукту Jira Software, а саме:

– призначення;
– потенційних користувачів;
– сферу застосування;
– можливості інтегрування з іншими системами;

– доступних опціях для хостингу.

Таблиця 1.3 – Огляд JIRA Software

Призначення продукту	Планування, відстеження і випуск програмного забезпечення світового рівня
Users	Software developers Project managers SCRUM masters
Use Cases	Відстеження помилок Управління проєктами Менеджмент продукту Управління процесами Управління завданнями Розробка програмного забезпечення Agile розробка програмного забезпечення
Інтеграція	Confluence Bitbucket Slack GitHub
Hosting options	Cloud, Server, Data Center

Розглянемо позитивні сторони використання JIRA Software:

– хороша видимість (visibility). Одним з факторів, що уповільнюють здійснення будь-якого проєкту (не тільки в області розробки програмного забезпечення, а й в інших професійних сферах, та й просто в житті), є відсутність чіткої постановки завдань і вибудовування ієрархії їх виконання. JIRA усуває цю проблему, оскільки вона об'єднує команди таким чином, що всі члени команди отримують можливість бачити поступ у виконанні завдань іншими співробітниками в режимі реального часу, так як завдання мають теги «started» і «completed». Це допомагає всім членам команди знати, на якій стадії знаходиться проєкт, що прискорює роботу над програмним продуктом;

– зручне визначення пріоритетів. Ще одна перевага використання JIRA полягає в тому, що вона дає змогу краще визначити порядок пріоритетів завдань, що стоять перед усіма членами команди, внаслідок чого можна побачити, які завдання необхідно виконати негайно і які можуть бути вирішені пізніше. Це дуже

корисно з точки зору дотримання строків, особливо при роботі над різними проєктами з різними датами завершення;

– підвищення продуктивності. Використовуючи JIRA, члени команди отримують можливість в будь-який момент часу бачити послідовність завдань в списку (backlog), внаслідок чого зменшується час простою, витрачений на обговорення поточних проблем, і підвищується продуктивність роботи. Хоча час простоїв може здатися незначним, проте загальна їх кількість може призвести до перевищення часу, відведеного на виконання завдання. Таким чином, JIRA сприяє усуненню цієї проблеми і підвищенню продуктивності праці;

– забезпечення неперервності взаємодії між членами команди, де б вони не знаходилися. Ще однією перевагою програмного забезпечення JIRA є те, що JIRA поставляється з доступними мобільними додатками. Це означає, що всі члени команди можуть залишатися на зв'язку не тільки в офісі чи вдома через ноутбук, а й використовуючи мобільні телефони та планшети;

– понад 1000 плагінів. JIRA поставляється з понад 1000 додатками, які допоможуть зробити програмне забезпечення ще кориснішим для команд гнучкої розробки, два найпопулярніших з них – GreenHopper і Bonfire [24].

Розглянемо недоліки використання JIRA:

– велика насиченість функціональними можливостями, що створює додаткові труднощі не тільки для користувачів-початківців, а й для просунутих, так як процес кастомізації продукту виявляється досить трудомістким в силу великої кількості параметрів системи. Таким чином, потрібен якийсь час, щоб зрозуміти архітектуру програмного забезпечення;

– складний UX / UI. Велика кількість візуальної інформації, а також наявність не завжди інтуїтивно зрозумілих елементів інтерфейсу створює користувачеві труднощі взаємодії з JIRA;

– вимога наявності навичок DevOps, знання Scrum, вміти будувати спринт для створення структури роботи;

– відсутність вбудованої функціональності управління календарем;

– висока вартість, яка становить для команди до 10 чоловік 10 доларів США в місяць, до 100 чоловік – 7 доларів США за кожного члена в місяць.

Для кращої організації спільної роботи разом з Jira в IT-компаніях зазвичай використовують Confluence, так як це продукти однієї компанії Atlassian і вони добре інтегровані один з одним. Розглянемо переваги і недоліки роботи з Confluence.

Confluence – платформа, що дає змогу публікувати веб-сторінки і документи в стилі wiki, а також обмінюватися контентом між учасниками команди і організувати обговорення.

Призначення Confluence:

– централізація інформаційних потоків;

– повне обслуговування файлової системи і документації (створення, зберігання, перегляд, обмін, редагування);

– функція пошуку документів або іншої робочої інформації, наприклад, листів;

– вбудований корпоративний календар;

– постановка завдань і контроль процесу виконання;

– корпоративні чати.

Серед недоліків даного програмного продукту користувачі вказують наступні:

– незручна система коментарів;

– погано організована система одночасного редагування текстів;

– не налагоджене відстеження розвитку проєкту;

– незручний тайм-менеджмент;

– відсутнє управління ресурсами;

– система ціноутворення; ціна на продукт зростає з ростом кількості користувачів [25].

Як видно, можливості Confluence чималі. Розробники постаралися врахувати всі потреби роботи в офісі.

Висновки до розділу 1

На підставі розглянутих моделей життєвих циклів проєкту, моделей і засобів управління проєктами можна зробити такі висновки.

1. Модель життєвого циклу проекту CRISP-DM найадекватніше відображає процеси розробки нейронної мережі. Для управління процесами розробки програмного продукту с використанням ШІ найоптимальнішим буде використання комбінації гнучкої моделі і моделі CRISP-DM.

2. Гнучка модель є найпридатнішою для IT-компаній, стартапів, проєктів в інноваційних сферах, модель CRISP-DM дає змогу управляти діяльністю відділу R&D; каскадна модель зручна в проєктах, де ключовим обмежувачем є строк реалізації проєкту, а не фінанси.

3. Паралельне використання даних методологій дасть змогу оптимізувати процеси і синхронізувати принципово різну діяльність команд, що працюють над створенням одного продукту.

2 МЕТОДИ І АЛГОРИТМИ УПРАВЛІННЯ ПРОЄКТАМИ В ІТ-КОМПАНІЇ

2.1 Алгоритми процесу розроблення програмних продуктів і методи управління проєктами в ТОВ «Неткрекер»

Компанія ТОВ «Неткрекер» зареєстрована 24.02.2006 р. за юридичною адресою: Україна, 04071, м. Київ, вул. Ярославська, 58. Керівником організації є Терентьєв Віталій Васильович. Основним предметом діяльності ТОВ «Неткрекер» є: 62.01 – Комп'ютерне програмування. Дохід компанії за 2020 р. становить 963 655 000 грн.

Засновником ТОВ «Неткрекер» є компанія НЕТКРЕКЕР ТЕКНОЛОДЖІ КОРПОРЕЙШН (NETCRACKER TECHNOLOGY CORPORATION) – постачальник продуктових рішень BSS/OSS для провайдерів послуг зв'язку та кабельних операторів у світі. Компанія спеціалізується на створенні, впровадженні та супроводі систем підтримки бізнесу BSS [26] та систем операційної підтримки OSS [27]. Крім цього, компанія веде інноваційні проєкти у Cloud, B2B2X, Analytics/AI, LEO, підтримує перехід на 5G та технологію Open vRAN операторів зв'язку по всьому світу.

Штаб-квартира компанії розташована в місті Уолтем, штат Массачусетс, США. Офіси Netcracker розташовані в Північній та Латинській Америці, Європі, Південній Африці, на Близькому Сході та в Азіатсько-Тихоокеанському регіоні. В Україні підрозділи розташовані в Києві, Сумах, Одесі.

З 2008 р. Netcracker є дочірньою компанією NEC Corporation. Корпорація об'єднала всі свої продукти у галузі телекомунікаційного програмного забезпечення та послуг під брендом Netcracker.

«Netcracker пропонує повний набір консультаційних, керованих і професійних послуг. Комплексне портфоліо програмного забезпечення побудоване на єдиній, готовій до використання в хмарному середовищі, платформі, яка використовує штучний інтелект і розширену аналітику». [28] Ця платформа охоплює такі компоненти:

1. Взаємодія із клієнтами (Customer Engagement). Продукт включає Channel Management, Customer Journey Management, Marketing Management.

2. Цифрова система підтримки бізнесу (Digital BSS), що складається з Customer Management, Revenue Management, Sales Management, Partner Management, Product Management.

3. Цифрова система підтримки операцій (Digital OSS) пропонує такі інструменти, як Service Management & Orchestration, SDN/NFV Management & Orchestration, Hybrid Resource Management, Infrastructure Management.

4. Розширена аналітика (Advanced Analytics) включає Analytics Domains і Analytics Platform.

5. Integration & API Management допомагає інтегрувати сервіси Netcracker із сервісами та системами замовників в умовах різних типів даних, інтерфейсів та вимог.

Процес виготовлення прикладного програмного забезпечення, що використовує ШІ, має складнішу структуру порівняно з процесом виготовлення типового програмного продукту.

Нижче представлена укрупнена блок-схема алгоритму створення будь-якого програмного продукту, як з штучним інтелектом, так і стандартного (рис. 2.1).

Цей процес охоплює п’ять стадій: від аналізу умов поставленого завдання до супроводу готового виробу в процесі його експлуатації замовником.

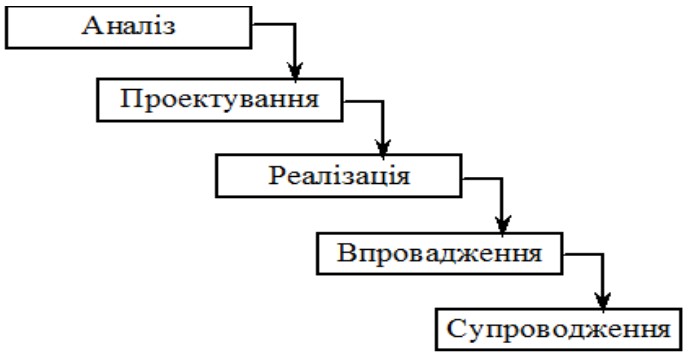

Рисунок 2.1 – Блок-схема робочого процесу розробки програмних продуктів

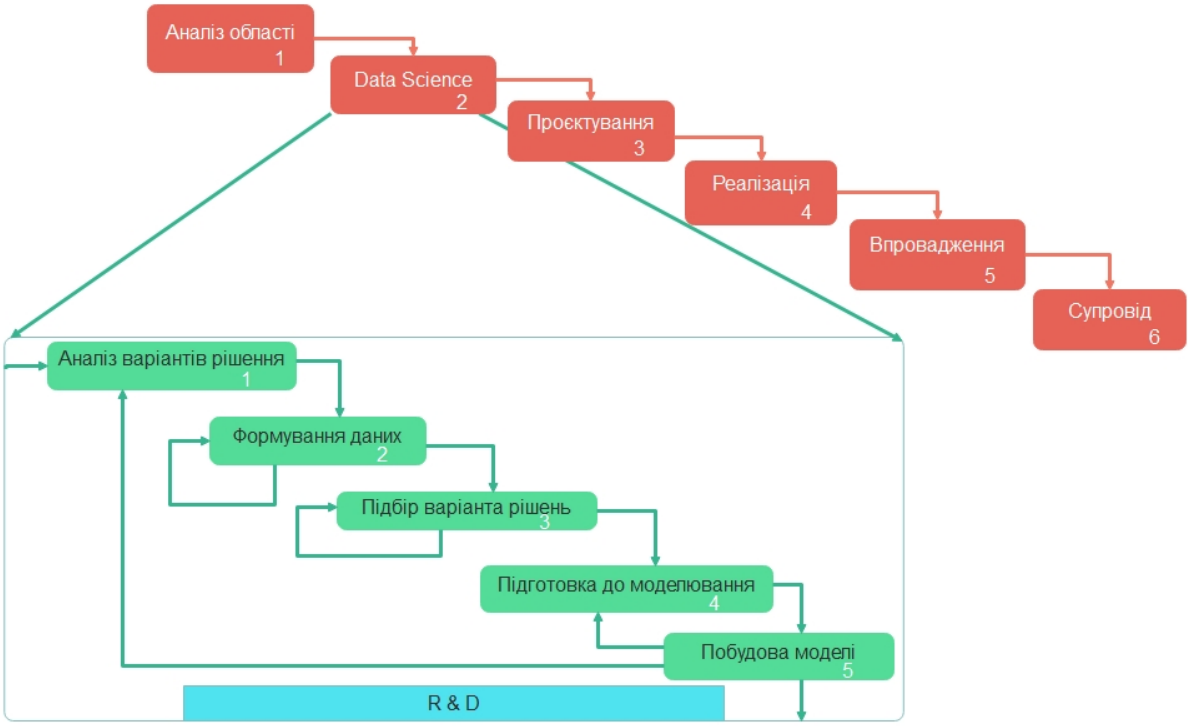

Рисунок 2.2 – Блок-схема робочого процесу розробки програмних продуктів із штучним інтелектом

Однак детальніша блок-схема (рис.2.2) у випадку роботи над створенням сервісів, програм, платформ з використанням ШІ (наприклад, нейромереж) дає змогу виділити в робочому процесі цілий етап, який передує безпосередньо проєктуванню програмного рішення – дослідницьку діяльність (Research and Development – R&D).

Такий тип діяльності притаманний проєктам Data Science і спрямований на дослідження можливих варіантів вирішення поставленого завдання, а також розробку і навчання моделі ШІ для кожного з варіантів. Особливістю такої діяльності є те, що після етапу аналізу предметної області неможливо однозначно переходити до етапу проєктування кінцевого програмного продукту (як це відбувається при вирішенні стандартних завдань), оскільки вихідні дані, отримані після етапу аналізу, можуть бути багаторазово переглянуті та уточнені.

Уточнений алгоритм етапу R&D представлений на рис.2.3.

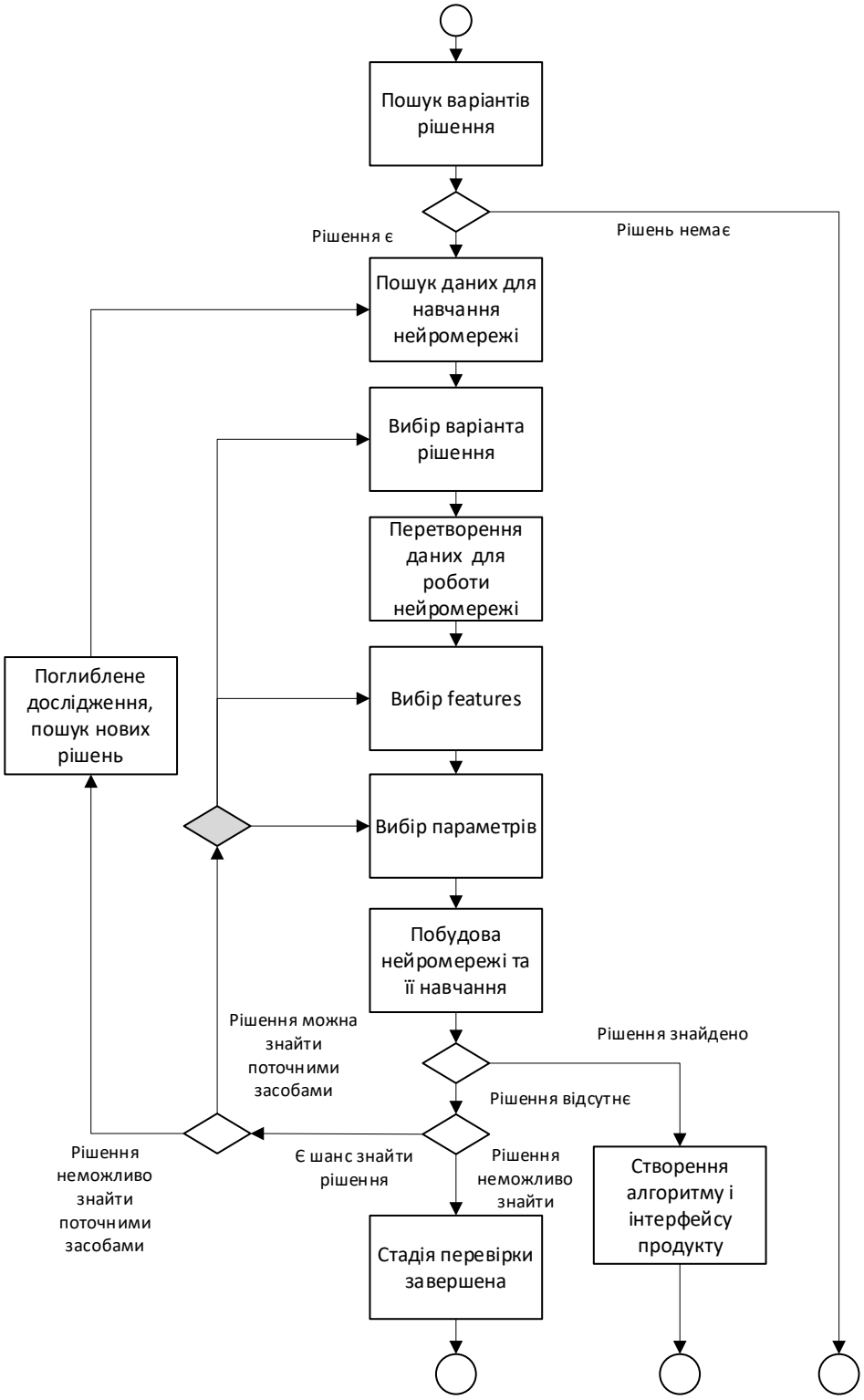

Рисунок 2.3 – Блок-схема алгоритму R&D

Як показано на рис.2.3, на відміну від процесу реалізації традиційних рішень, життєвий цикл ШІ-проєктів ускладнюється необхідністю проведення низки підготовчих робіт, які мають циклічний характер. Кількість цих циклів заздалегідь невідома і залежить як від ступеня складності поставленого завдання, так і від кваліфікації співробітників на проєкті та рівня розвитку їх творчого мислення.

Більш того, можливий результат, при якому детальне вивчення поставленого завдання показує, що воно не має рішення.

Вузол прийняття рішення, позначений сірим кольором, відображає той момент дослідження, коли команда, ґрунтуючись на своєму досвіді та інтуїції, вибирає напрямок, пов'язаний або зі зміною параметрів моделі, або зі зміною «features», або з пошуком нового варіанту вирішення поставленого завдання.

Циклічний характер процесу створення ШІ-продукту істотно відрізняється від Agile-моделей (Scrum, Kanban і т.д.), основу яких становлять ітерації, на кожній з яких отримують деяку закінчену частину цільового програмного продукту.

У випадку з ШІ-проєктами завершення кожного циклу не завжди означає створення частини або функції кінцевого продукту; навпаки, часто буває, що виконавці проєкту змушені повернутися до точки нульового відліку і почати заново роботу з вихідною інформацією. Крім того, діяльність R&D ґрунтується на роботі з великими масивами даних, причому протягом усього життєвого циклу проєкту.

Ця ключова обставина змусила фахівців Data Science шукати спеціалізовані методи роботи з Big Data.

На даний момент в компаніях, що спеціалізуються на створенні продуктів Data Science, найпоширенішою є CRISP-DM- методологія дослідження даних.

Таким чином, діяльність команди в процесі створення ШІ-продукту, складається з дослідницьких дій, які завжди носять унікальний, оригінальний характер, і стандартних дій з проєктування, реалізації та впровадження.

Протягом усього процесу менеджмент компанії ТОВ «Неткрекер» для управління проєктом використовує Jira, а ведення документації виконується за допомогою Confluence. Ці інструменти містять весь необхідний функціонал для управління завданнями, персоналом і документацією на стандартних проєктах, й ідеально підходять для управління етапами проєктування та реалізації, проте недостатні для управління дослідницькою діяльністю на етапі R&D. Як показало проведене автором вивчення особливостей управління проєктами в компанії, на етапі R&D можливості Jira використовуються частково, фрагментарно та неефективно.

2.2 Організація процесу дослідницької діяльності в ТОВ «Неткрекер»

Процес R&D передбачає велику залученість проєктної команди у завдання, які за Agile-методологією виконує проєктний менеджер.

Співробітники компанії є висококваліфікованими фахівцями і володіють перехресними вміннями, що є необхідною умовою роботи в сфері виробництва ШІ-продуктів.

У процесі роботи над програмним продуктом усі співробітники, включаючи керівництво, тісно пов’язані між собою і підтримують постійний зв’язок із замовником. Щоранку проводяться стендап-мітинги, на яких обговорюються результати минулого дня і плануються завдання на поточний день.

При цьому також приймаються рішення про відхилення проміжних завдань або зміну постановки завдання, якщо не вдається знайти її рішення. Щотижня проводяться брейн-шторми, де виносяться завдання, виконання яких здається неможливим. В обговоренні пошуку можливих варіантів вирішення бере участь весь склад компанії, в особливо скрутних випадках можуть залучатися сторонні фахівці. Варто зазначити, що результати проведених мітингів практично не документуються. Виникаючі в процесі обговорення думки, ідеї, висновки, пропозиції записуються на листочках, за допомогою стікерів на дошці, а часом взагалі не фіксуються. Поки компанія невелика, подібна практика не є катастрофічною, але тим не менше на сьогоднішній день існує велике упущення в процесах управління проєктами.

Схема організаційної структури підприємства в загальному вигляді представлена на рис.2.4.

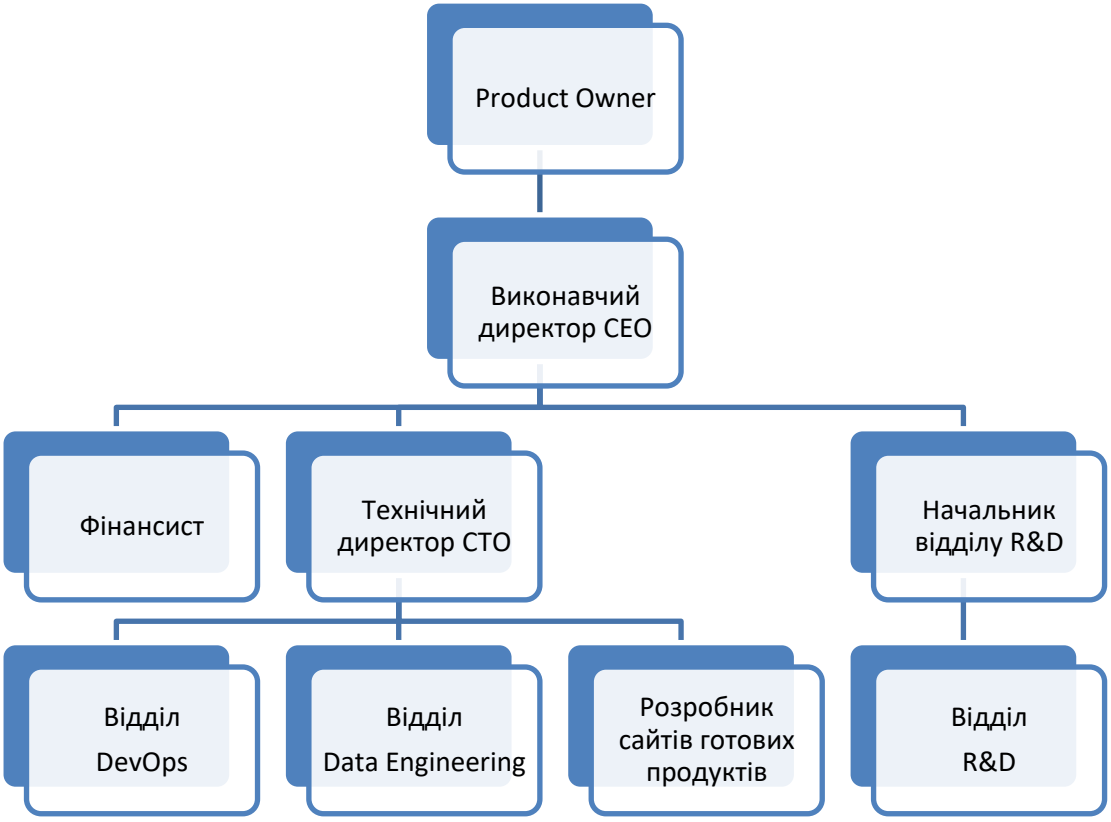

Рисунок 2.4 – Загальна організаційна структура підприємства

Розглянемо, як відбувається створення програмного продукту в компанії з точки зору поділу праці.

Аналіз організації процесу розробки проектів в компанії ТОВ «Неткрекер» на етапі R&D показав, що вона має складну нелінійну структуру (рис. 2.5).

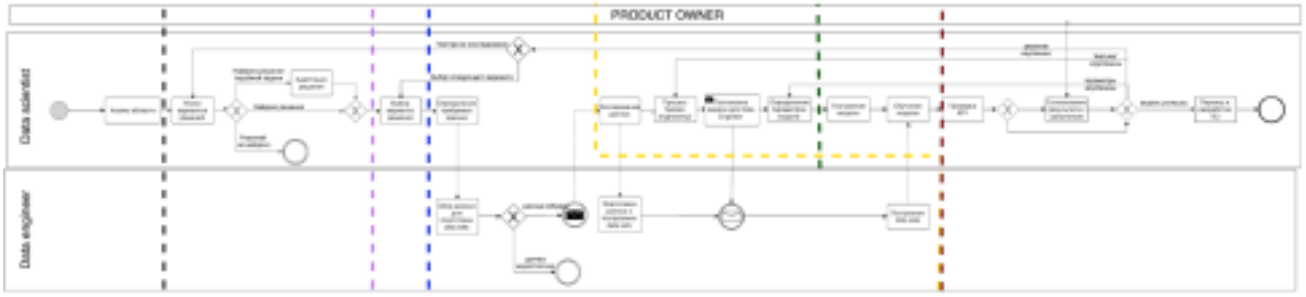

Рисунок 2.5 – Модель бізнес-процесів етапу R&D на ТОВ «Неткрекер»

Представлена модель бізнес-процесів відображає розгортання процесу розробки продукту в часі та розподіл робіт між виконавцями, а також зв’язок між ними.

Рис.2.5 представляє собою наступний рівень декомпозиції блок-схем алгоритму R&D, виконаних на рис.2.2 і 2.3. Оскільки докладніша деталізація ускладнює схему дослідницької діяльності, виділення чітких етапів (як це було зроблено на рис.2.2), стає вельми складним, так як реальний, фізичний процес створення програмного продукту є нелінійним, ситуативним, а його підпроцеси можуть бути паралельними чи перетинатися.

Процес створення і навчання нейромережі розподілений між трьома сторонами: Product Owner (особа, яка приймає рішення з боку замовника), фахівці з Data Science та інженери роботи з даними (Data Engineer). Як уже зазначалося, бувають випадки, коли виконавці міняються ролями і тимчасово беруть на себе обов'язки іншого члена команди.

Простежимо організаційні зв'язки між відділами і співробітника на ТОВ «Неткрекер» на кожному етапі алгоритму R&D.

Як раніше показано на рис.2.3, R&D-діяльності передує аналіз предметної області. Результатами цього аналізу повинні бути:

– визначення мети продукту;

– визначення бізнес-вимог;

– визначення KPI для кожної бізнес-вимоги;

– визначення можливих способів задоволення бізнес-вимог, а саме: визначення проєктною командою стратегії вирішення кожної бізнес-вимоги – за допомогою побудови ШІ чи алгоритмічних методів.

Попередня орієнтовна діяльність в проєктах з іноземними інвестиціями має значення більше, ніж в інших видах проєктів. На цій стадії відбувається формулювання критеріїв, яким в подальшому буде відповідати готовий продукт.

Реалізація будь-якого проєкту завжди починається з формулювання потреби замовника/ринку в ньому. Product Owner повинен мати уявлення про те, як цей продукт або його нова функція повинні працювати в кінцевому підсумку. Наприклад, впровадження деякого сервісу запобіжить відтоку відвідувачів або стимулюватиме продаж інших продуктів.

Формулювання такої потреби не претендує на повний опис проєкту, вона тільки позначає проблему: «наші клієнти повинні зрозуміти, як вони

розпоряджаються своїм бюджетом», «якщо ми додамо функцію передбачення години пік в аеропорту, то клієнти охочіше купуватимуть додаток».

Тут необхідно викласти цілі продукту, на підставі яких формуються бізнес-вимоги, тобто набір тієї функціональності, яка визначає призначення ПЗ.

На цьому етапі всі учасники проєкту пропонують ідеї, які можна використовувати при вирішенні сформульованої проблеми, відбувається створення деяких початкових начерків можливого рішення. Цим етапом зазвичай керує фахівець Data Science, але важливо, щоб брали участь абсолютно всі учасники команди проєкту, включаючи менеджмент, так як можливі рішення і продуктивні ідеї можуть лежать в абсолютно різних областях знань. В результаті цього етапу зазвичай стає ясно, наскільки глибоким буде процес дослідження даних.

Потім варто провести визначення кількісних показників, за якими встановлюватимуть, чи досягнута бізнес-вимога. Цей етап полягає в спільному визначенні обсягу і ключових показників ефективності проекту (KPI). Далі ці KPI повинні бути уточнені та переведені у вимірні.

Можливо, вдасться отримати дуже чіткі показники. Однак у деяких випадках необхідно використовувати якісні показники, наприклад, «час, необхідний для вивчення теми з використанням згенерованих розширених запитів, буде скорочено і / або покращиться якість результатів порівняно з вихідними запитами». Це особливо вірно, коли модель призначена для надання допомоги людині в деякій складній для неї діяльності.

Ключові показники можуть бути уточнені в зв’язку з наявними часовими рамками і ресурсами. Для їх уточнення потрібен подальший збір даних.

Зазвичай ці метрики (KPI) визначаються проєктною командою спільно з Product Owner у точних кількісних показниках. Але залежно від ресурсів і часових обмежень вони можуть бути піддані перегляду після консультацій із замовником. Оскільки будь-який зворотний зв’язок означає додаткові витрати часу, то проєктна команда може спробувати знайти додаткові жорсткі метрики, на які можна спертися для продовження роботи над продуктом, і уникнути додаткових консультацій із замовником.

Визначення обсягу робіт на цьому етапі особливо важливе, оскільки дослідницькі проєкти мають тенденцію збільшуватися в розмірах і масштабах з появою нових можливостей при проведенні досліджень.

Процес визначення значень KPI є, таким чином, ітераційним. Він завершується тоді, коли знайдені значення задоволять клієнта. Зазначимо наступний факт, що відображає певний конфлікт бізнес-інтересів замовника і тих можливостей програмного продукту, як їх оцінює проєктна команда. Бізнес завжди прагне мати максимальні показники KPI. Розробники стверджують, що програмний продукт об’єктивно не в змозі відобразити реальну ситуацію (в предметній області) на 100%. Наприклад, якщо система передбачає поведінку покупців з точністю менше 95%, то така система, на думку більшості замовників, є марною.

Однак у багатьох випадках ретельний аналіз і обговорення припущень про продукт можуть привести до появи дуже цінних рішень, які можуть виявитися не такими технічно складними, як здавалося на першій ітерації KPI.

Визначити обсяг робіт на даному етапі також є важливим з психологічної точки зору. Дослідник, який починає вивчати отримане завдання, прагне вивчити якомога більшу кількість статей і книг. Тому відсутність чітко окреслених меж часу, ресурсів і цілей проекту може привести до довгих тижнями або місяцями, витраченим на розробку прекрасних моделей, які в кінцевому підсумку не відповідають реальній потребі, або призводять до неврахування деяких KPI, які могли бути визначені раніше.

Аналіз рішення.

На цьому етапі перед проєктною командою стоять завдання вивчення літератури та існуючих готових рішень. Дослідник аналізує як академічну літературу, так і існуючі програмні рішення та інструменти.

Глибина занурення в літературу залежить від особливостей проєкту: чи є він фундаментальним, тобто визначає подальшу стратегію компанії, чи майбутній програмний продукт призначений для вирішення разової проблеми. Також необхідно відповісти на питання: чи планується публікація результатів

дослідження з цього питання в академічних виданнях; чи планує дослідник стати експертом команди по цій темі і т. п.

Наприклад, проєкт призначений для того, щоб допомогти відділу продажів краще прогнозувати відтік клієнтів і збитки від відтоку. Для вирішення поставленого завдання потрібно добрі знання теорії випадкових процесів, на якій будуються багато спільних рішень подібних проблем.

Необхідна глибина розуміння залежить від технічних аспектів проблеми. Деякі з аспектів можуть бути передбачені заздалегідь, а деякі виявлені лише пізніше. Наприклад, якщо створюваний продукт повинен інтегруватися в готове програмне середовище, написане на мовах Java і Scala, то доводиться використовувати ті технології, які застосовуються в готовому продукті. І якщо розробники не володіють цими мовами, то їх доведеться вивчити.

При вивченні літератури фахівець Data Science повинен підготувати для обговорення в команді не лише обрані варіанти вирішення проблеми, а й зробити короткий огляд усієї області та всіх розглянутих рішень, пояснюючи переваги і недоліки кожного рішення і обґрунтувати свій вибір.

Після того, як підібрано відповідний математичний апарат, знайдені напрямки рішень необхідно оцінити з точки зору способу реалізації та її складності. Цілі продукту і бізнес-вимоги до нього, а також структура і характеристики пропонованих варіантів рішень служать обґрунтуванням для вибору способу зберігання і обробки даних, здатність до масштабування по горизонталі та вертикалі (scalability) і приблизну оцінку вартості проєкту.

Це важлива перевірка, яку необхідно виконати на цьому етапі, оскільки обробка даних і розробка програмного забезпечення можуть починатися паралельно з розробкою моделі. Крім того, пропоноване рішення може виявитися неадекватним або занадто дорогим з технічної точки зору, і в цьому випадку це повинно бути виявлено та усунуто якнайскоріше. Коли технічні питання розглядаються до початку розробки моделі, знання, отримані на етапі дослідження, можуть потім використовуватися для пропозиції альтернативного рішення, яке могло б краще відповідати технічним обмеженням. Це ще одна причина, за якою

етап дослідження повинен закінчитися виробленням спектра рішень, а не обмежуватися одним.

Підготовка даних і вибір рішення.

На цьому етапі проєктна команда визначає набір даних, необхідних у подальшому для навчання нейронних мереж; збір первинних даних, обробка їх (очищення від сміттєвої інформації), перетворення в необхідний формат. Зберігаються підготовлені дані разом із запитами в базах даних компанії.

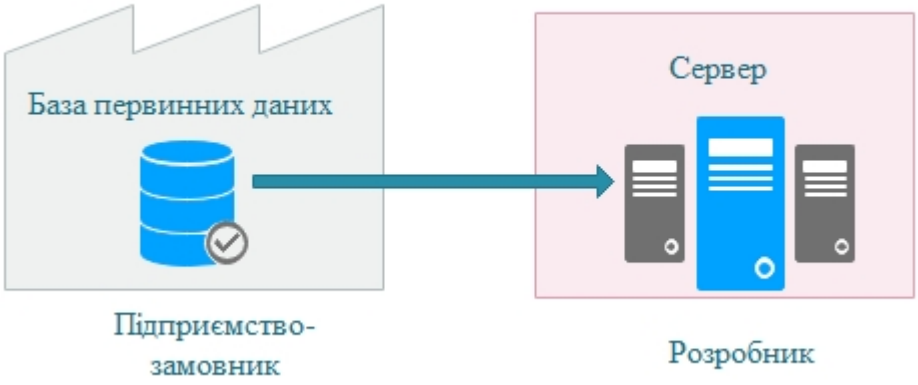

Рисунок 2.6 – Схема взаємодії баз даних

Процес підготовки даних часто супроводжується вивантаженням великих масивів даних (Big Data) з баз даних замовника на сервер компанії (якщо дозволяють умови конфіденційності). Надалі відбувається обробка Big Data в формат, зручний для швидких запитів і складних обчислень. Ці дії необхідні для початку процесу дослідження даних й іноді займають більше часу, ніж очікувалося.

Далі починається процес дослідження даних. Тепер фахівець Data Science може оперувати фактичними жорсткими KPI і показниками моделі. Однак і тут треба витримувати динамічну рівновагу між дослідженням і розробкою; навіть маючи на увазі чіткі KPI, корисно не втрачати з поля зору деякі, здавалося б, невідповідні напрямки вирішення.

Оброблені дані повинні мати формат, зручний для роботи відділу Data Engineering. Однак можуть бути виявлені деякі недоліки сформованого робочого масиву даних (наприклад, даних, наданих замовником, може виявитися

недостатньо). Data Engineer даних повинен бути готовий до такої ситуації, і шукати додаткове джерело даних.

Варто зазначити, що хоч процес підготовки даних і виглядає відокремленим від процесу теоретичного дослідження та аналізу варіантів рішень, вони зазвичай або виконуються паралельно, або чергуються між собою.

Підготовка до моделювання.

Підготовка до початку розробки моделі значно залежить від наявного технічного забезпечення і обсягу технічної підтримки, доступної Data Scientist. Можливо, що Data Scientist створить новий репозиторій коду і запустить локальний сервер Jupyter Notebook або запросить потужнішу хмарну машину для виконання обчислень.

Також може бути, що співробітники відділу Data Science створять код для управління версіями даних і моделей або для відстеження та управління експериментом. Можливо, що така функціональність вже існує в використовуваних сервісах, то тоді буде потрібно створення деяких налаштувань для розподілу ресурсів або настройки призначених для користувача пакетів тощо.

Цей етап пов’язаний не тільки з технічною підготовкою, а й підготовкою «features» і параметрів для майбутньої нейромережі.

Розробка і навчання моделі.

Завданнями цього етапу є розгортання моделі, навчання і неперервний моніторинг результатів навчання.

У підготовленому на попередньому етапі середовищі з вибраними параметрами створюється модель нейромережі. Фахівці відділу Data Engineering передають до відділу Data Science набори даних і створені до них запити, підготовлені відповідно до набору features. Відділ Data Science завантажує ці дані в навчальне середовище для подальшого навчання моделі. Фахівці відділу Data science здійснюють моніторинг результатів навчання, використовуючи функціональність сервісів. Після того, як модель вважається навченою, можна переходити до модельного тесту.

При розробці та навчанні моделі різні її версії (і обслуговуючий масив даних) повинні постійно перевірятися на відповідність заздалегідь встановленим KPI.

Це дає можливість оцінити ступінь просування в навчанні, а також дає змогу відділу Data Science вирішити, коли модель працює досить добре, щоб гарантувати повну відповідність виробленим на початковому етапі KPI.

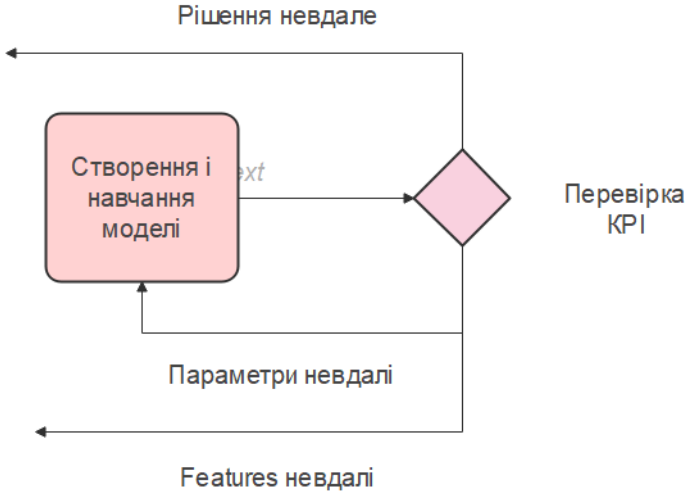

Рисунок 2.7 – Алгоритм прийняття рішень про успішність моделі

Якщо результати навчання моделі не відповідають очікуваним показникам, то існує кілька причин такого результату.

Спеціаліст відділу Data Science аналізує результати навчання нейронної мережі та приймає рішення про подальші дії: змінити параметри, змінити «features», або даний варіант моделі не придатний для вирішення завдання і треба переходити до вибору наступного варіанту моделі.

Іншим можливим результатом невдалого навчання є перегляд мети продукту. У рідкісних випадках зміна мети тягне за собою незначні зміни в технічній реалізації проєкту. Але в загальному, це означає повернення на фазу аналізу предметної області.

Не виключений найекстремальніший варіант – скасування проєкту; якщо фахівець з даними впевнений, що всі напрямки досліджень були вивчені, а проєктний менеджер впевнений, що продукт вичерпав свій потенціал, можливо, прийшов час перейти до іншого проєкту.

Якщо навчання моделі пройшло успішно і вона реалізує задані KPI, то етап R&D можна вважати успішно закінченим і переходити до розробки алгоритмічної частини кінцевого продукту, а також створення інтерфейсу користувача.

У ситуаціях, коли розбіжність між отриманими і заданими значеннями KPI не є критичним, потрібно ретельніший їх перегляд та узгодження з Product Owner. Необхідно переконатися, що поточне значення показника знаходиться в інтервалі прийнятного розкиду. При цьому необхідно враховувати вимоги до продукту та інтереси клієнта.

Коли Product Owner переконаний, що модель відповідає заявленим цілям проєкту (в задовільному ступені), команда може приступити до виробництва програмного продукту.

На підставі виконаного аналізу організації процесу дослідницької діяльності в ТОВ «Неткрекер» можна зробити висновок про те, що структура цього процесу не є лінійною.

Розгортка Agile-моделі життєвого циклу проєкту має лінійну архітектуру, як представлено на рис.2.8.

Рисунок 2.8 – Розгортка Agile-моделі життєвого циклу проєкту

Моделі Crisp-DM, на відміну від Agile-моделей, можна представити у вигляді дерева рішень, де кожна гілка відображає новий цикл розробки.

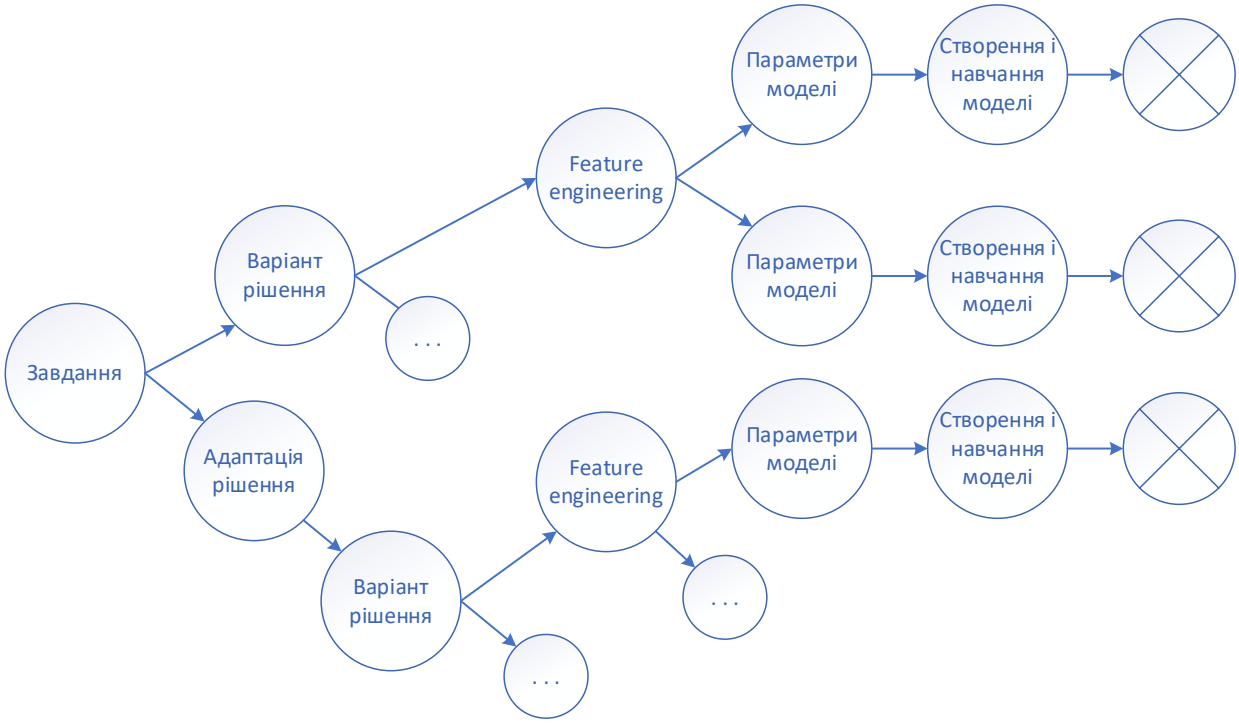

Рисунок 2.9 – Дерево рішень моделі Crisp-DM

Кореневим вузлом дерева є певні межі проєкту у вигляді цілей, бізнес-вимог і набору KPI. Вузлами другого рівня можна представити знайдені або адаптовані можливі вирішення поставленого завдання. Далі кожен вузол має розгалуження на розроблені features. З даних вузлів розходяться гілки за обраними параметрами. Кожен вузол належить одній з розглянутих раніше фаз етапу R&D.

2.3 Модель системи управління проєктами «As Is»

Виконаний аналіз процесу створення програмного продукту з ШІ, а також організаційної структури процесу в ТОВ «Неткрекер» дає змогу побудувати модель «As Is» («Як є») процесів управління в компанії.

Модель зображена на рис.2.10. Вона відображає управлінські дії проєктного менеджера в інструментах Jira та Confluence, а також дії інших співробітників компанії, які беруть на себе функції Project Manager (PM) відповідно з поточними потребами проєкту. Побудована модель відображає процеси управління, які

супроводжують всі фази R&D-діяльності. Процеси розподілені між Project Manager, відділом Data Science і відділом Data Engineering.

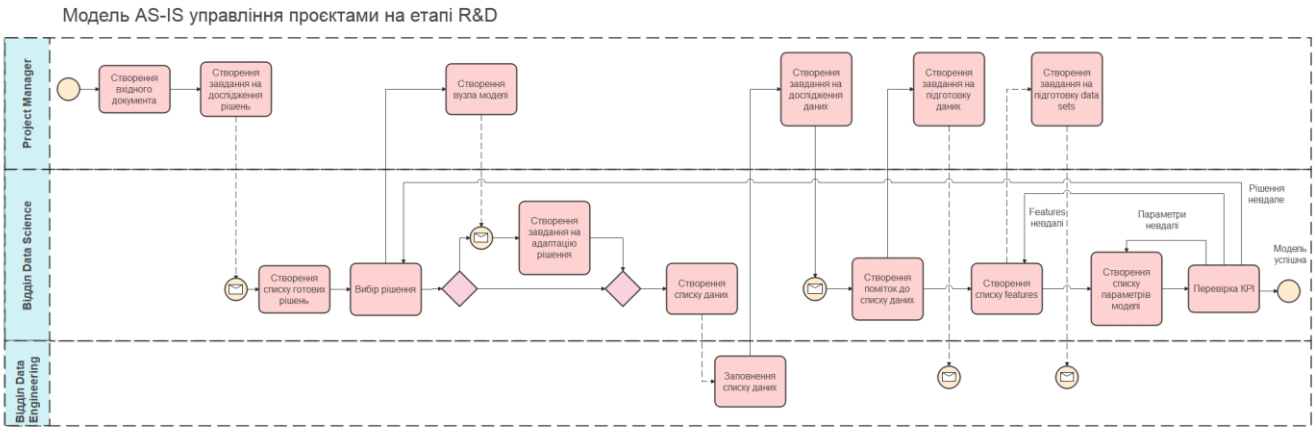

Рисунок 2.10 – Модель управління процесами «As Is»

З точки зору управління даними етап R&D можна представити таким чином.

Розглянемо докладніше представлену модель процесу управління проєктом, що функціонує в ТОВ «Неткрекер».

1) створення вхідного документа. Цей процес належить фазі аналізу предметної області. Тут команда збирається на нараду і записує на листочку чи дошці всі прийняті на нараді рішення, а саме: список цілей, список бізнес-вимог, список KPI, а також позначки до кожної бізнес-вимоги, за допомогою чого вона вирішується (ШІ або алгоритм). Після наради PM робить фотографію дошки або листочка і переносить всі записи в Confluence;

2) створення завдання на дослідження рішень. По закінченню наради PM створює завдання в Jira і закріплює завдання за співробітником відділу Data Science. Співробітник, який отримав завдання, приступає до її виконання;

3) створення списку готових рішень. Після дослідження співробітник створює список знайдених рішень у вигляді презентації і робить доповідь для проєктної команди;

4) вибір рішення. Після прослуховування доповіді команда обговорює його і вирішує, які з представлених рішень перспективні, які треба відкинути, а які залишити «про запас»;

5) створення завдання на адаптацію рішення. Якщо жодне з обраних рішень не можна застосувати в готовому вигляді, то PM створює завдання в Jira на адаптацію рішення і закріплює за співробітником відділу Data Science;

6) створення списку адаптацій для вирішення. Співробітник відділу Data Science, який отримав і виконав завдання, створює список адаптацій і заводить його в Confluence;

7) створення списку даних. Після вибору варіанту рішення проєктна команда збирається на нараду і створюється список даних, необхідних для роботи над проєктом. Цей пункт також може виконати співробітник відділу Data Science самостійно. Даний список створюється окремим документом в Confluence;

8) оновлення списку даних. Співробітник відділу Data Engineering роздруковує документ, створений в п.7 і займається збором даних. На роздрукованому документі створюються позначки про те, які дані вдалося зібрати і в якому обсязі;

9) створення завдання на дослідження даних. Після успішного збору даних PM створює завдання в Jira для начальника відділу Data Science;

10) створення позначок до списку даних. Начальник відділу Data Science досліджує дані і готує документ з позначками до зібраного списку даних. Даний документ створюється в Confluence;

11) створення завдання на підготовку даних. PM створює завдання в Jira для співробітника відділу Data Engineering на підготовку даних. Співробітник відділу Data Engineering отримує завдання разом з документом з п.10;

12) створення списку «Features». Після створення позначок до списку даних співробітники відділу Data Science проводять нараду з метою визначення списку «Features». Цей список заводиться в Confluence одним із співробітників відділу Data Science;

13) створення завдання на підготовку наборів даних. Після виконання п.12 PM створює завдання в Jira для співробітників відділу Data Engineering с прикріпленим документом з п.12;

14) створення списку параметрів моделі. Після виконання завдання, поставленого в п.12, співробітники відділу Data Science створюють список параметрів для поточної моделі;

15) перевірка KPI. Після створення і навчання моделі з обраними параметрами і обраним списком «Features» співробітники відділу Data Science перевіряють результати навчання моделі засобами Jupiter Notebook. Після цього показники поточного циклу не зберігаються у вигляді документа.

Як можна побачити, проєктний менеджер створює завдання за допомогою Jira. Найбільша кількість інформації, що підлягає документуванню, виробляється у відділі Data science.

Оскільки проєктний менеджмент не входить в прямі обов'язки дослідників і розробників, то документування виконується ними непрофесійно. Вони переважно просто нехтують питаннями збереження потенційно корисної інформації.

В результаті у проєктного менеджера немає візуального відображення стану справ в будь-який момент часу. Він може дізнатися про статус виконання завдань і дані, що стосуються поточної гілки лише в усній формі.

Виконане дослідження організації управління в ТОВ «Неткрекер» дало змогу зробити наступні спостереження. Більшість документації не має чіткої структури і створюється на розсуд виконавця.

Не вся документація ведеться за допомогою корпоративної системи Confluence. Деякі з документів заводяться в особистих облікових записах співробітників усередині інших систем або створюються в паперовому вигляді.

Документи, створені поза Confluence, не завжди переносяться в Confluence.

Багато документів зовсім не створюються, обмін робочою інформацією відбувається або в усній формі, або в паперовому вигляді.

Висновки до розділу 2

Вивчення методів управління проєктами, прийнятих в ТОВ «Неткрекер», а також алгоритму R&D-діяльності дали змогу зробити наступні спостереження та висновки.

1. Компанія стикається з такими проблемами:

– неповна документація. Ця проблема виникає в зв’язку з тим, що створення документації традиційно входить до обов’язків менеджерського складу та бізнес-аналітиків. В умовах дослідницької діяльності документація є технічною і проєктний менеджер не в змозі її коректно заповнити без допомоги (диктування) технічних фахівців. Тому розробники створюють її самостійно. Але оскільки подібного роду обов’язки не є традиційними для них, необхідна дозвільна документація формується неякісно, а іноді взагалі не створюється, так як виконавці не хочуть витрачати на неї час і покладаються на свою пам’ять. Таким чином, цінні думки, ідеї, творчі рішення можуть бути загублені. Внаслідок незбереженої документації в деяких випадках проєктній команді доводиться витрачати додатковий час на повторне обговорення завдань, способів вирішення, а також на повторні дослідження, мозкові штурми і т. д.;

– неуніфікований формат. З причини, описаної вище, документація створюється в різних форматах. Це ускладнює процес її спільного використання і т. д.;

– слабка візуалізація. Дерево рішень не візуалізується і ніде не зберігається. Це може впливати на загальне уявлення про стадії проєкту, а також ускладнює проєктному менеджеру завдання звітності замовникам про виконану роботу;

– неповний облік часу. Завдання повторюються від гілки до гілки, деякі завдання є незначними (з точки зору часу), проєктна команда компанії невелика і кожен знає свої обов’язки, з цих причин деякі завдання можуть не створюватися в інструменті управління проєктами. Це ускладнює облік часу, так як в Jira буде зберігатися неповна інформація про завдання. Також це може породити проблему «втрати» деяких завдань.

2. Модель CRISP-DM є на даний момент єдиною прийнятною для Data Science проектів, але вона володіє певними недоліками, які впливають на процес управління етапом R&D. До них належать: відсутність коштів підтримки і відсутність критерію часу.

Дана модель стала застосовуватися недавно до подібного роду проєктів, та й самі проєкти Data Science з’явилися недавно. Тому існуючі інструменти управління не встигли ще адаптуватися під потреби проєктів Data Science, а нові не з’явилися на ринку.

3 ПРОЄКТУВАННЯ КОМП'ЮТЕРИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ ІТ-КОМПАНІЇ

3.1 Модель комп'ютеризованої системи управління проєктами в ТОВ «Неткрекер»

Вивчення способу організації управління проєктами в ТОВ «Неткрекер» показало, що на підприємстві існують істотні можливості для підвищення ефективності процесу управління проєктами.

Багато в чому ручна модель управління, яка функціонує в компанії, не цілком адекватна процесам створення проєктів, виконуваних командою.

Для організації роботи проєктної команди необхідний єдиний інформаційний центр, за допомогою якого можна вирішувати такі завдання управління:

– зберігати проєктну документацію;

– вести протоколи мітингів, стендапів, мозкових штурмів, виробничих нарад;

– зберігати презентації доповідей співробітників;

– повідомляти співробітників про події і плани;

– створювати реєстри завдань, бізнес-процесів, власних розробок;

– створювати власну бібліотеку готових рішень за профілем компанії;

– створювати «запасники» цікавих ідей, методів, процедур, які можуть стати в нагоді в майбутньому;

– призначати завдання і роздавати доручення;

– здійснювати зворотний зв'язок за поставленими завданнями і дорученнями.

Тому варто розробити модель подібного інформаційного центру у вигляді комп'ютеризованої системи управління проєктами. Ця КСУ проєктами повинна бути інтегрована в існуючу систему управління проєктами і за рахунок синергії підвищити ефективність управління проєктами, і, отже, роботи компанії в цілому.

Аналіз методів і алгоритмів процесу управління проєктами в ТОВ «Неткрекер» показав, що управління на етапі R&D істотно відрізняється від процесу управління на етапі виконання програмного продукту.

Модельована система повинна базуватися на програмних продуктах, які на даний момент використовуються в ТОВ «Неткрекер»: Jira та Confluence. Цей продукт не буде замінювати поточні системи, а буде вбудований в неї з метою комп'ютеризації рутинних процесів. Продукт не є самостійною і незалежною системою, а є плагіном для системи Jira, тому повинен бути адаптований до роботи з нею.

Існують наступні методи адаптації програмного забезпечення:

1) параметрична адаптація, пов'язана з налаштуванням параметрів програми. Це найпростіший спосіб адаптації; він передбачає зміну значень характеристик, що регулюють роботу програми. За допомогою параметричної адаптації можливо проводити необхідну настройку функцій і компонентів програми, і відбирати стратегії поведінки з запропонованого набору стратегій;

2) функціональна адаптація передбачає зміну функцій програмного забезпечення в установлених межах. Функціональна адаптація може виконуватися спільно з параметричною. Структура і організація ПЗ при цьому залишаються незмінними;

3) організаційна адаптація – переналаштування потоків і процесів у системі. Вона означає перерозподіл потоків і процесів системи як її внутрішніх ресурсів, без зміни її структури. Може поєднуватися з функціональної адаптацією;

4) структурна адаптація – зміна структури системи. Використовуючи цей спосіб адаптації, здійснюється модифікація або заміна одних структурних елементів системи чи алгоритмічних модулів на інші. Адаптована програма стає адекватнішою виконуваним завданням. При цьому можливе використання всіх попередніх видів адаптації системи;

5) розмноження – породження собі подібних нащадків. Адаптація розмноженням є ефективним методом адаптації. Система плодить собі подібну, але володіє наявністю вільних ресурсів і здатністю до змін;

6) розвиток – спрямований процес еволюції систем. Адаптація програмного забезпечення через розвиток подібна процесам еволюції живих систем. Адаптоване ПЗ проходить процес еволюції через накопичення інформації про себе, зовнішнє середовище, розв’язувані завдання. Проходячи через етапи зародження, становлення певних якостей, стійкого функціонування, деградацію і загибель, ПЗ як система стає пристосованішим для вирішення завдань.

Модельована система управління проєктами повинна являти собою окремий функціональний блок, вбудований в систему Jira та Confluence, а також взаємодіяти з ними за допомогою їхніх функціональних можливостей. Таким чином, існуюча система управління проєктами в ТОВ «Неткрекер», що поєднує ручне і автоматизоване управління, дещо змінює свою структуру, організацію та функції.

Розглянемо, як змінюються функції. У моделі «As Is» функції створення цілого ряду документів і завдань лежали на проєктній команді і проєктному менеджерові. За допомогою R&D-плагіну ця робота буде виконуватися автоматично, оскільки в створювану систему буде закладено алгоритм R&D-діяльності, завдяки чому шаблони документів на кожному етапі процесу управління будуть формуватися автоматично; завдання і доручення також будуть генеруватися за допомогою нової системи завдяки функції відстеження дій проєктної команди.

Таким чином, не порушуючи встановлених меж зміни функцій існуючої системи управління, система «To Be» розширює спектр функцій, які використовуються в команді.

Організаційна адаптація системи управління проєктами полягає в тому, що колишні потоки управлінської інформації змінюють свою траєкторію. Взаємодія між проєктним менеджером і відділами компанії буде здійснюватися в новій моделі через єдиний інформаційний центр управління. При цьому точки входу і виходу організаційних потоків залишаються без змін.

Також дещо змінюється структура існуючої системи управління проєктами. Методи ручного управління перестають бути обов’язковими і в системі Jira з’являється новий елемент – R&D-плагін.

На основі структурно-функціонально-організаційного методу адаптації була розроблена модель КСУ проєктами на ТОВ «Неткрекер». Модель представлена наступними діаграмами:

– верхнєрівнева діаграма середовища «R&D-плагін»;
– діаграма бізнес-процесів управління «То Ве»;
– діаграма прецедентів;
– ER-діаграма.

Верхнєрівнева діаграма середовища «R&D-плагін» показана на рис.3.1.

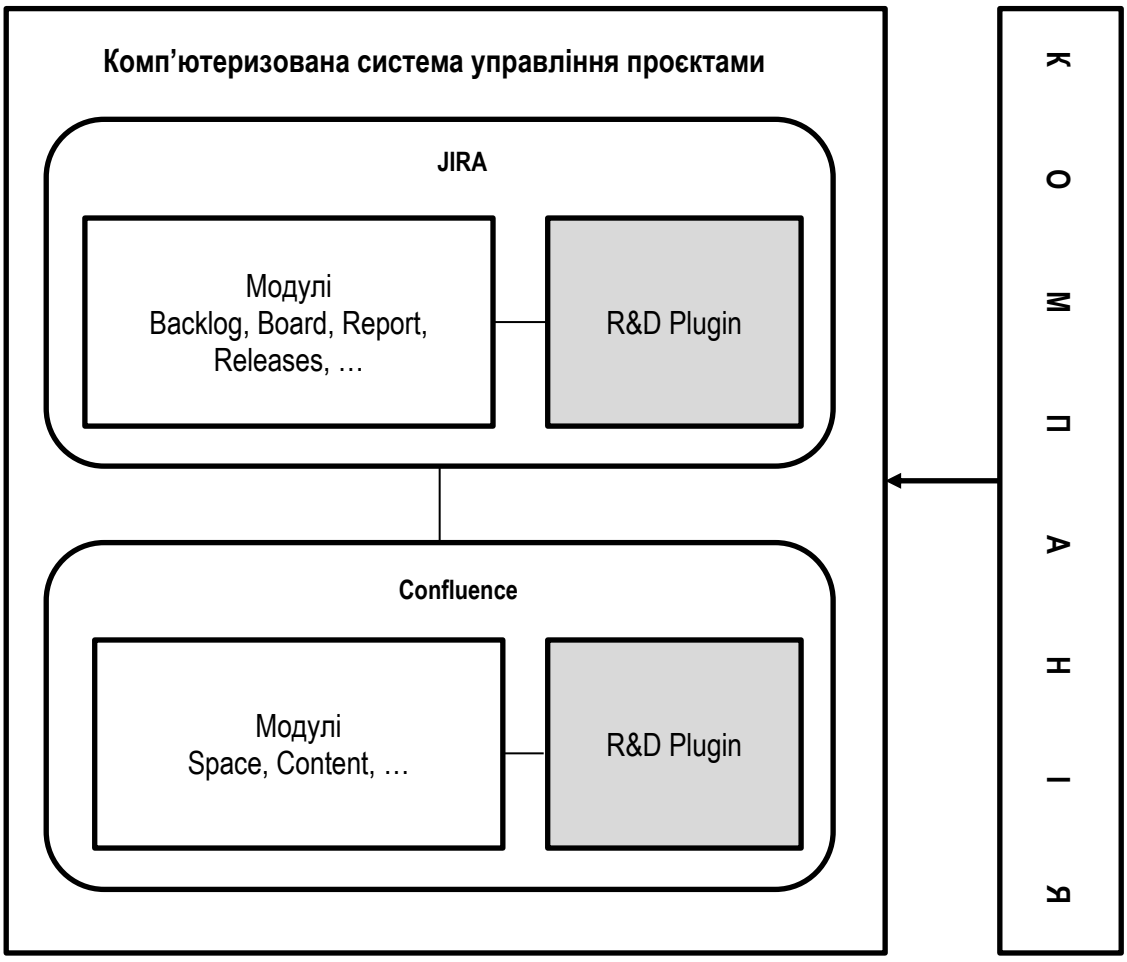

Рисунок 3.1 – Діаграма середовища вбудованого плагіну для управління проєктами на етапі R&D

Діаграма середовища показує, з якими вже існуючими підсистемами належить взаємодіяти новому плагіну, і відображає зв'язки між ними.

Рис.3.2 зображує модель взаємодії системи управління проєктами «То Ве» («Як буде») з учасниками етапу R&D.

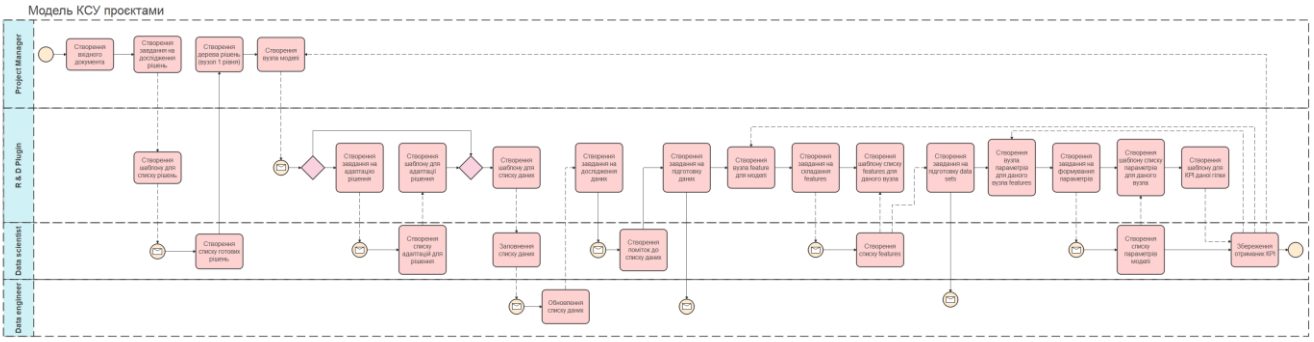

Рисунок 3.2 – Діаграма бізнес-процесів моделі «To Be»
управління проєктами на етапі R&D

В даній моделі з’являється ще один елемент – R&D Plugin. Процеси, які були присутні в моделі «As Is», перейшли з пулів учасників проєкту в пул плагіна, що означає, що виконання цих процесів виконується плагіном.

Забираючи на себе частину обов’язків інших учасників процесу, новий плагін може значно скоротити час, необхідний проєктній команді на створення завдань, документації та на відстеження процесу розробки.

Розглянемо докладніше запропоновану діаграму бізнес-процесів управління проєктами на етапі R&D «To Be»:

1) створення вхідного документа. Для ініціації створення нового проєкту проєктним менеджером (PM) плагін пропонує йому заповнити вхідний документ за встановленим шаблоном: заповнити список цілей проєкту, заповнити список бізнес-вимог, до кожної бізнес-вимоги додати коментар – створюється проєкт за допомогою ШІ чи за допомогою алгоритмізованого підходу, а також створити набір KPI майбутнього продукту;

2) створення завдання на дослідження рішень. Після закінчення наради PM закінчує роботу над створенням документа, перевіряє заповнені дані та підтверджує створення нового проєкту в системі Jira. Плагін створює новий проєкт в Jira, створює завдання на дослідження рішень, а також зберігає протокол, заповнений PM у системі Confluence. Далі PM закріплює завдання за співробітником відділу Data Science. Співробітник, який отримав завдання, приступає до її виконання;

3) створення шаблону списку можливих варіантів вирішення. Плагін автоматично створює шаблон для знайдених варіантів рішення в системі Confluence. Цей шаблон як посилання буде прикріплений до завдання, створеного в п.2;

4) створення списку готових рішень. Після п.3 співробітник Data Science заповнює список знайдених рішень у документ. Після наради, на якому проєктна команда буде обговорювати варіанти вирішення, творець списку робить позначки, в яких розставить пріоритети для кожного знайденого рішення. Також позначаються рішення, які вимагають адаптації;

5) створення дерева рішень (вузол 1-го рівня). На сторінці плагіна проєктним менеджером створюється кореневий вузол майбутнього дерева рішення;

6) створення вузла моделі. Проєктний менеджер ініціює створення вузла рішення. КСУ створює вузол рішення відповідно обраним у п.4 пріоритетам і переходить до наступного етапу;

7) створення завдання на адаптацію рішення. Якщо рішення, відповідне створеному в п.6 вузлові, не можна застосувати в готовому вигляді, то плагін створює завдання в Jira на адаптацію рішення. PM може закріпити її за конкретним співробітником відділу Data Science;

8) створення шаблону для адаптації рішення. Якщо був виконаний, то плагін створює шаблон в системі Confluence для документа на адаптацію рішення і прикріплює його посиланням до завдання, створеного в п.7;

9) створення списку адаптацій для вирішення. Співробітник відділу Data Science, який отримав і виконав завдання, заповнює список адаптацій і за створеним в Confluence шаблоном;

10) створення шаблону для списку даних. Після закінчення п.6, якщо рішення не вимагало адаптацій, або після п.9, якщо вимагало, плагін створює шаблон для списку даних, необхідних для поточного вузла;

11) створення списку даних. Після вибору варіанту рішення проєктна команда збирається на нараду і створюється список даних, необхідних для роботи над проєктом. Цей пункт також може виконати співробітник відділу Data Science самостійно. Цей список заповнюється за шаблоном, створеним у Confluence;

12) оновлення списку даних. Співробітник відділу Data Engineering роздруковує документ, створений в п.7 або відкриває його в мобільному додатку Confluence і займається збором даних. У документі створюються коментарі про те, які дані вдалося зібрати і в якому обсязі. Після закінчення цього етапу СТО компанії відзначає на інтерфейсі плагіна, що збір даних закінчено;

13) створення завдання на дослідження даних. Після виконання п.12 плагін автоматично створює завдання в Jira з прикріпленим посиланням на сторінку Confluence зі списком даних і призначає її виконання начальнику відділу Data Science;

14) створення позначок до списку даних. Начальник відділу Data Science досліджує дані і править документ зі списком даних, роблячи позначки. Після закінчення виконання цього завдання начальник відділу Data Science зазначає на інтерфейсі плагіна, що завдання завершене;

15) створення завдання на підготовку даних. КСУ створює завдання в Jira на підготовку даних. PM може призначити виконавця даної задачі. Співробітник відділу Data Engineering отримує завдання разом з документом з п.10;

16) створення вузла «features» для даної моделі. Після того, як виконаний п.15, плагін створює вузол «features»;

17) створення завдання на складання «features». Після закінчення виконання п.16, плагін створює в Jira завдання на складання списку «features». PM може призначити виконавця цього завдання;

18) створення шаблону списку «features». Як тільки створено завдання, описане в п.17, КСУ створює шаблон списку «features» для даного вузла. Цей шаблон як посилання на сторінку в системі Confluence прикріплюється до завдання з п.17;

19) створення списку «features». Після створення позначок до списку даних співробітники відділу Data Science проводять нараду з метою визначення списку «features». Даний список заповнюється в Confluence одним із співробітників відділу Data Science в заздалегідь створеному шаблоні. Після закінчення цього етапу начальник відділу Data Science позначає на інтерфейсі плагіна, що список «features» складено;

20) створення завдання на підготовку Data Sets. Після виконання п.19 плагін створює завдання в системі Jira на підготовку Data Sets. До завдання прикріплюється документ, створений в п.19. PM може призначити виконавця цього завдання;

21) створення вузла параметрів для даного вузла «features». Після створення завдання, описаного в п.20, плагін створює новий вузол в системі – вузол параметрів;

22) створення завдання на складання параметрів. Після створення вузла параметрів КСУ створює завдання в Jira на створення списку параметрів поточного варіанту моделі. PM може призначити виконавця даної задачі;

23) створення шаблону списку параметрів для даного вузла. Плагін створює шаблон списку параметрів в Confluence і прикріплює посилання на дану сторінку Confluence до завдання, описаного в п.22;

24) створення списку параметрів моделі. Після виконання завдання, поставленого в п.22, співробітники відділу Data Science заповнюють список параметрів для поточної моделі в шаблоні, створеного плагіном. Проєктна команда займається створенням моделі по заданих параметрах і навчання її на підготовлених Data Sets. Після того, як модель буде вважатися навченою, команда повинна проаналізувати її показники з точки зору досягнення KPI продукту;

25) створення шаблону для KPI даної гілки. Після виконання п.24, плагін створює шаблон в Confluence для списку показників KPI для даної гілки;

26) збереження одержаних KPI. Після закінчення навчання моделі співробітник відділу Data Science заповнює створений в п.25 шаблон.

По закінченню п.26 проєктна команда приймає рішення про те, чи створене рішення є відповідним і можна вважати етап R&D завершеним, або KPI вважаються досягнутими. У випадку, якщо KPI вважаються досягнутими, перед проєктною командою стоїть вибір – в який з вузлів рішення повертатися:

- створити інші параметри моделі з цим же набором «features» і повернутися в п.21;
- створити новий набір «features» і повернутися в п.16;
- повернутися до вибору нового варіанту рішення і повернутися в п.6.

Побудуємо діаграму прецедентів (Use Case діаграма), що відображає варіанти взаємодії учасників етапу R&D з модельованим плагіном.

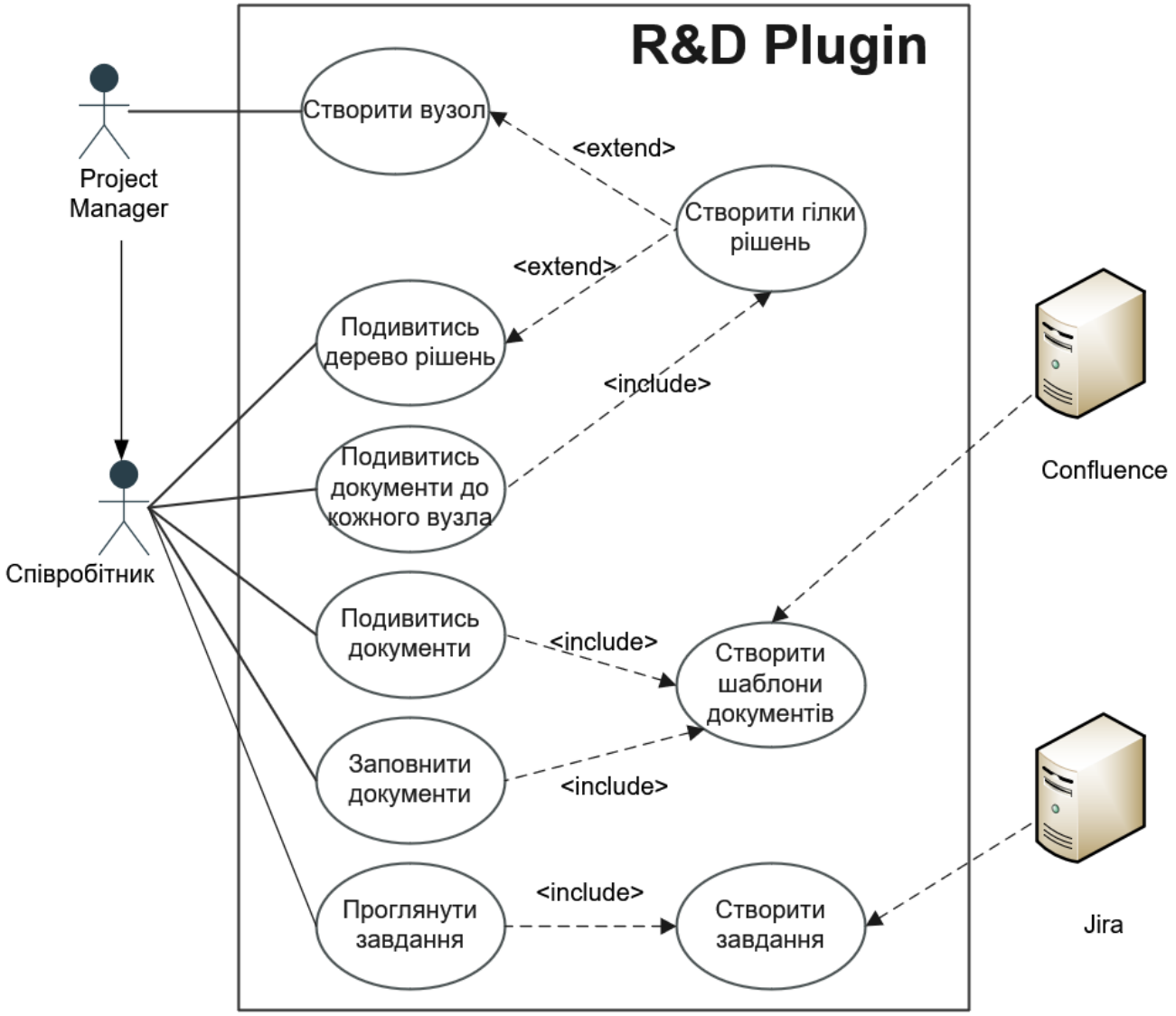

Рисунок 3.3 – Діаграма прецедентів для моделі «To Be» управління проєктами на етапі R&D

Діаграма варіантів використання дає змогу описати функціональне призначення системи, а також показує межі системи. R&D Plugin являє собою систему, що автоматизує процеси, які вимагають ручного управління. КСУ проєктами може покрити виконання таких функцій як створення документів і їх структури, створення завдань. Ці функції є автоматичними і виконуються в потрібний момент часу на підставі системи тригерів, яка корелюється з закладеним в систему алгоритмом етапу R&D. Завдяки функціям відстеження виконання завдань і своєчасного створення шаблонів потрібної документації система спрощує

процес створення документації до проєкту, а також полегшує створення уніфікованого її формату. Також плагін дає змогу зручним способом візуалізувати фази етапу R&D, а також бачити статуси кожної фази дослідження.

Для проведення структурного аналізу модельованої системи був використаний метод моделювання «сутність-зв'язок» (або ER-діаграма).

Цей метод дає змогу відобразити об'єкти, які будуть використовуватися в системі, а також способи їх взаємодії. Діаграма ER КСУ ШІ-проєктами на етапі R&D приведена на рис.3.4.

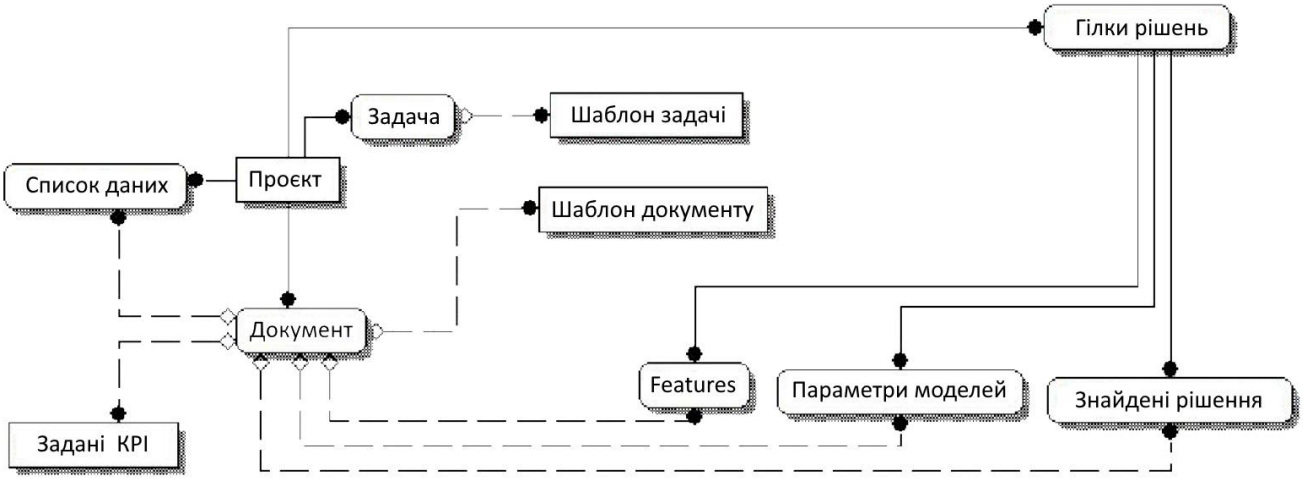

Рисунок 3.4 – ER-діаграма моделі «То Ве» управління проєктами на етапі R&D

ER-діаграма відображає взаємодію між ключовими сутностями даної системи (проєкт, документи, завдання і гілки рішень) в статиці.

Таким чином, сукупність описаних діаграм (діаграма середовища, діаграма бізнес-процесів, діаграма прецедентів і ER-діаграма) дає повний опис моделі «То Ве» інформаційної системи управління ШІ-проєктами на етапі R&D на ТОВ «Неткрекер».

3.2 Якісне оцінювання моделі комп'ютеризованої системи управління проєктами в ТОВ «Неткрекер»

З метою якісного оцінювання запропонованої моделі комп'ютеризованої системи управління (КСУ) проєктами були застосовані наступні методи:

– оцінювання ступеня комп’ютеризації бізнес-процесів для порівняння моделей «As Is» і «To Be»;

– експертних оцінок.

Для виявлення слабких і сильних сторін розробленої моделі КСУ проєктами, а також потенціалу її подальшого розвитку та визначення можливих ризиків був застосований SWOT-аналіз.

Оцінювання ступеня комп’ютеризації проводилася за методикою, описаною в [29]. Для проведення оцінювання ступеня комп’ютеризації необхідно було підрахувати кількість рутинних або ручних дій, які виконуються в даний час в компанії (модель «As Is»), і кількість тих же дій після можливого впровадження плагіна (модель «To Be») і комп’ютеризації процесу управління.

Найбільша кількість рутинних дій припадає на два процеси:

– створення завдання в системі Jira;

– створення документа в системі Confluence.

Створення завдань є рутинним дією, так як на етапі R&D завдання в основному є короткостроковими, але їх досить багато, тому проєктному менеджеру іноді є недоцільним витрачати час на створення і управління завданням в системі Jira.

Створення документа є ще і трудомістким саме по собі, оскільки система Confluence не найзручніший інструмент.

Також незручністю є те, що документи через специфіку етапу R&D доводиться створювати технічним фахівцям.

Подібні обов’язки рідко покладаються не на проєктних менеджерів, тому технічні фахівці часто замість повноцінного створення документа прикріплюють фотографію того ж документа, написаного від руки, а то і зовсім нехтують виконанням цього завдання.

Оцінювання ступеня комп’ютеризації проводилася на підставі порядкової шкали 10-ти рівнів комп’ютеризації (LOA) Т. Шерідана і В. Вепланка [30] (табл.3.1).

У табл.3.2 перераховані елементарні дії користувача в процесі створення завдання і створення документа As Is і дана оцінка ступеня їх комп’ютеризації.

Таблиця 3.1 – Шкала рівнів комп’ютеризації Шерідана і Вепланка

Рівень комп’ютеризації (LOA)	Опис
1	Комп’ютер не пропонує допомогу: людина повинна приймати всі рішення і виконувати всі дії сама
2	Комп’ютер пропонує людині повний набір рішень / дій, альтернативи (приклад – робота з електронним довідником)
3	Комп’ютер пропонує повний набір рішень / дій, альтернативи і звужує вибір до кількох варіантів
4	Комп’ютер пропонує одну альтернативу
5	Комп’ютер пропонує одну альтернативу і автоматично виконує цю пропозицію, якщо людина погоджується
6	Комп’ютер пропонує одну альтернативу і виконує цю пропозицію, якщо людина протягом обмеженого часу не накладає вето на автоматичне виконання операції
7	Комп’ютер виконує операції автоматично, обов’язково інформуючи людину
8	Комп’ютер виконує операції автоматично; інформує людину, лише по запиту
9	Комп’ютер виконує операції автоматично; інформує людину, тільки якщо він (комп’ютер) вирішить
10	Комп’ютер вирішує все і діє автономно, не звертаючи увагу на людину

Розглянуті процеси (створення документа і створення завдання) неодноразово повторюються протягом усього етапу R&D.

Знайдемо загальну кількість операцій, які виконуються учасниками проєктної команди на одній повній гілці дерева рішень. Також розділимо ці операції на 2 категорії – ручні (LOA<6) і автоматичні (LOA>=6) – і знайдемо відсоток вмісту кожної з категорій операцій на етапі R&D.

Таблиця 3.2 – Ступінь комп’ютеризації процесів створення завдання і створення документів As Is

Назва процесу	Опис операцій в даному процесі	LOA
Створення документа	Відкрити Confluence	1
	Відкрити потрібну папку	1
	Створити документ	1
	Створити структуру документа	1
	Заповнити документ	1
	Опублікувати документ	1
Створення завдання	Відкрити Jira	1
	Відкрити Backlog	1
	Створити завдання	1
	Описати завдання	1
	Назначити виконавця	1
	Перенести в поточний спринт	1

Таблиця 3.3 – Ступінь комп’ютеризації для процесу As Is на одній повній гілці дерева рішень

Вид операції	Кількість операцій у завданні	Кількість завдань на одній гілці	Загальна кількість операцій на етапі R&D для одної гілки	Відсоток операцій на етапі R&D
Процес створення завдання				
Ручні операції	6	8	48	100%
Автоматичні операції	-	-	-	-
Процес створення документа				
Ручні операції	6	6	46	100%
Автоматичні операції	-	-	-	-

Як видно з табл.3.3, процеси створення завдання і створення документа є ручними на 100%.

Побудуємо таблиці, аналогічні табл.3.2 і 3.3, для моделі To Be.

Таблиця 3.4 – Ступінь комп’ютеризації процесів створення завдання і створення документів To Be

Назва процесу	Опис операцій в даному процесі	LOA
Створення документа	Відкрити Confluence	1
	Відкрити потрібну папку	10
	Створити документ	10
	Створити структуру документа	10
	Заповнити документ	1
	Опублікувати документ	1
Створення завдання	Відкрити Jira	10
	Відкрити Backlog	10
	Створити завдання	10
	Описати завдання	10
	Назначити виконавця	1
	Перенести в поточний спринт	10

Таблиця 3.5 – Ступінь комп’ютеризації для процесу To Be на одній повній гілці дерева рішень

Вид операції	Кількість операцій у завданні	Кількість завдань на одній гілці	Загальна кількість операцій на етапі R&D для одної гілки	Відсоток операцій на етапі R&D
Процес створення завдання				
Ручні операції	1	8	8	14,3%
Автоматичні операції	6	8	48	85,7%
Процес створення документа				
Ручні операції	3	5	15	50%
Автоматичні операції	3	5	15	50%

Аналіз табл.3.4-3.5 показує, що для однієї гілки дослідження ступінь комп’ютеризації процесів створення завдань збільшилася на 85,7%, а ступінь комп’ютеризації процесу створення документа на 50%.

Збільшення рівня комп’ютеризації процесів призводить також до зростання деяких якісних показників управління, серед яких можна вказати поліпшення

візуального представлення циклів рішення, потенційне збереження більшої кількості даних про створювані програмні рішеннях і багато інших.

Для оцінювання ефективності розробленої моделі КСУ проєктами застосовано SWOT-аналіз, котрий дав змогу виявити переваги та недоліки системи, на які варто звернути увагу при подальшій розробці програмного продукту. Також були виявлені потенційні напрямки розвитку і способи монетизації програмного продукту. Останнім завданням SWOT-аналізу стало виявлення можливих ризиків, пов’язаних з розробкою модельованого програмного продукту.

Рисунок 3.5 – SWOT-аналіз КСУ проєктами в компанії

SWOT-аналіз показав, що запропонована система має значну кількість сильних сторін і переваг перед старою моделлю управління.

Модель To Be пропонує такі поліпшення як: упорядкування документообігу, збільшення ступеня візуального та функціонального комфорту роботи всіх учасників проєкту, поліпшення комунікації між ними і т.д.

Слабкі сторони в основному пов’язані з «прив’язкою» до системи Jira та їх залежності відповідно від цінової політики компанії Atlassian і стабільності функціоналу продуктів Atlassian. Також до уразливостей запропонованого плагіна можна віднести звуження цільової аудиторії до користувачів продуктів Atlassian, оскільки не всі працюючі з Data Science компанії використовують Jira та Confluence.

Висновки до розділу 3

1. Дослідження методів, засобів та існуючих інформаційних систем управління проєктами; докладний аналіз діяльності учасників проєктної команди на етапі R&D; розроблений алгоритм процесу проєктної діяльності на етапі R&D; вивчена модель «As Is» існуючих процесів управління в компанії ТОВ «Неткрекер» дали змогу розробити модель «To Be» комп’ютеризованої системи управління проєктами, яка враховує недоліки управління, наявні в компанії, наприклад, неналагодженість документообігу і наявність рутинних дій.

2. Модель «To Be» пропонує адаптувати існуючу в компанії систему управління проєктами Jira та Confluence, вмонтувавши в неї плагін управління етапом R&D. При цьому (в установлених межах) дещо змінюється структура, функції та організація інформаційних потоків існуючої системи.

3. Модель «To Be» повністю описується чотирма діаграмами:

– діаграма екосистеми вбудованого плагіна для управління проєктами на етапі R&D,

– діаграма бізнес-процесів моделі «To Be» управління проєктами на етапі R&D,

– діаграма прецедентів для моделі «To Be» управління проєктами на етапі R&D,

– ER-діаграма моделі «To Be» управління проєктами на етапі R&D.

4. Запропонованого опису моделі «To Be» за допомогою зазначених діаграм досить для того, щоб приступити до написання SRS і розробки плагіна.

ВИСНОВКИ

1. Для досягнення поставленої мети були вивчені сучасні методи, інструменти та інформаційні системи управління проєктами, визначені їхні переваги та недоліки; шляхом спостереження за діяльністю проєктної команди IT-компанії зі створення ШІ-продукту було проаналізовано її зміст і виявлено велику кількість рутинних дій, некерованих, дублюючих і тому непотрібних робіт, а також недоліки документообігу.

2. Оскільки управління проєктами в компанії проводиться в системі Jira, було прийнято рішення про розробку моделі КСУ проєктами з метою подальшого створення на її основі плагіна, вбудованого в Jira.

3. Теоретичне обґрунтування та аналіз процесів управління в компанії дали змогу розробити модель комп'ютеризованої системи управління проєктами, повний опис якої сформовано з чотирьох діаграм для етапу R&D: діаграми екосистеми вбудованого плагіна для управління проєктами, діаграми бізнес-процесів моделі «To Be», діаграма прецедентів для моделі «To Be, ER-діаграма моделі «To Be».

4. Було проведено якісне оцінювання розробленої моделі із застосуванням методів експертних оцінок і порівняльного аналізу рівня комп'ютеризації бізнес-процесів підприємства із використанням моделей управління «As Is» і «To Be». Виявлено, що модель «To Be» дає змогу збільшити ступінь комп'ютеризації створення завдань на 85,7%, а створення документа – на 50%.

5. На основі SWOT-аналізу моделі «To Be» виявлено сильні та слабкі сторони нової моделі, а також можливі ризики і потенціал подальшого розвитку.

6. Економічність і результативність запропонованої моделі комп'ютеризованої системи управління проєктами забезпечується наступними її властивостями:

– комп'ютеризація рутинної роботи на підприємстві;

– комп'ютеризація ручних методів управління;

– збереження всіх документів, що супроводжують процес створення проекту;

– напрямки організаційних потоків через єдиний інформаційний центр;

– скорочення часу, необхідного для кожного комунікативного акту;

– незалежність від взаємної територіальної віддаленості співробітників на проект;

– підвищення комфорту і швидкості взаємодії між співробітниками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ноздріна Л. В., Ящук В. І., Полотай О. І. Управління проектами: підручник. Київ: Центр учбової літератури, 2010. 432 с.

2. Довгань Л. Є., Мохонько Г. А., Малик І. П. Управління проектами: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського, 2017. 420 с.

3. Катренко А. В. Управління ІТ-проектами: підручник. Кн. 1. Стандарти, моделі та методи управління проектами / за наук. ред. В. В. Пасічника. Львів: Новий Світ-2000, 2011. 550 с.

4. Software Engineering Body of Knowledge (SWEBOK). URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering/v3 (Last accessed: 17.11.2021)

5. Семко І. Б. Особливості управління проектами в енергетичній галузі. *Східно-Європейський журнал передових технологій*. 2011. № 1 (49). С.46–47.

6. Мазур И. И. Управление инвестиционно-строительными проектами: международный подход. Москва: Омега-Л, 2011. 736 с.

7. Глушенкова А. А. Особливості управління інноваційними проєктами в сфері телекомунікацій та інформатизації. *Економіка. Менеджмент. Бізнес*. 2015. №4 (14). С.72–77.

8. Чемерис А. Розроблення та управління проектами у публічній сфері: європейський вимір для України. Практичний посібник. Київ: Софія-А, 2012. 80 с.

9. Орлик С. Основы программной инженерии (по SWEBOK). URL: https://ua1lib.org/book/3109458/707825 (дата звернення: 17.11.2021)

10. Royce W. Managing the Development of Large Software Systems: Concepts and Techniques. 1970. URL: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf (Last accessed: 17.11.2021)

11. Boehm B. W. A Spiral Model of Software Development and Enhancement. *Computer*. Vol.21, Issue 5. 1988. P. 61–72.

12. Амблер С. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки. СПб: Питер, 2005. 412 с.

13. У чому переваги каскадної методології управління проектами. URL:

https://ua.waykun.com/articles/u-chomu-perevagi-kaskadnoi-metodologii-upravlinnja.php (дата звернення: 17.11.2021)

14. Павлиш В. А., Гліненко Л. К., Шаховська Н. Б. Основи інформаційних технологій і систем. Львів: Львівська політехніка, 2018. 620 с.

15. Чуланова О. Л. Технология управления проектами и проектными командами на основе методологии гибкого управления проектами Agile. *Вестник евразийской науки*. 2018. №1. С.31–35.

16. Agile-маніфест розробки програмного забезпечення. URL: https://agilemanifesto.org/iso/uk/manifesto.html (дата звернення: 17.11.2021)

17. Тесля Ю. М. Інформаційні технології управління проектами. Київ: КНУБА, 2013. 120 с.

18. Advantages and Disadvantages of Trello. URL: https://www.software-developer-india.com/advantages-and-disadvantages-of-trello. (Last accessed: 17.11.2021)

19. What is Asana? URL: http://comparecamp.com/asana-reviews-pricing-benefits-and-features-analysis/ (Last accessed: 17.11.2021)

20. Asana и Trello. URL: https://asana.com/ru/compare/asana-vs-trello (Last accessed: 17.11.2021)

21. Product Guides & Tutorials. URL: https://www.atlassian.com/software/jira/guides/getting-started/overview (Last accessed: 17.11.2021)

22. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 6th ed. Newton Square, PA: Project Management Institute, 2017. – 756 p.

23. Макашов П. Л., Романенко Н. А. Сервис-ориентированный подход к управлению ИТ проектами на примере использования программного продукта "Jira". *Современные информационные технологии и ИТ-образование*. 2015. №11. С.12-16.

24. The benefits of using Jira in different industries. URL: https://community.atlassian.com/t5/Marketplace-Apps-Integrations/The-benefits-of-using-Jira-in-different-industries/ba-p/1770984 (Last accessed: 17.11.2021)

25. Confluence. URL: https://startpack.ru/application/confluence (Last accessed: 17.11.2021).

26. Digital BSS. URL: https://www.netcracker.com/portfolio/products/digital-bss/ (Last accessed: 17.11.2021).

27. Digital OSS. URL: https://www.netcracker.com/portfolio/products/digital-oss/ (Last accessed: 17.11.2021).

28. Netcracker Technology. URL: https://uk.wikipedia.org/wiki/Netcracker_Technology (Last accessed: 17.11.2021).

29. Кораблев И. Г. Оценка уровня автоматизации бизнес-процессов предприятия. *Вестник Череповецкого государственного университета*. 2016. №1 (70). URL: https://cyberleninka.ru/article/n/otsenka-urovnya-avtomatizatsii-biznes-protsessov-predpriyatiya (дата звернення: 17.11.2021)

30. Frohm J., Lindstrom V., Stahre J., Winroth M. P. Levels of Automation in Manufacturing. *International Journal of Ergonomics and Human Factors*. 2008. Vol.30, Issue 3. P.181-207.

31. Колянко О. В., Озимок Г. В. Використання жорсткої "Waterfall" та гнучкої "Agile" моделей управління проєктами. *Вісник Львівського торговельно-економічного університету. Економічні науки*. 2017. Вип.52. С. 177–182.

32. Бабаєв В. М. Управління проектами: навч. пос. Харків: ХНАМГ, 2006. 244 с.

33. Жизненный цикл проектной задачи. URL: http://projectimo.ru/upravlenie-proektami/zhiznennyj-cikl-proekta.html (дата звернення: 17.11.2021)

34. Шарова Е. С. Управление ИТ проектами. URL: http://www.cfin.ru/management/practice/supremum2002/03.shtml (дата звернення: 17.11.2021)

35. Товб А. С., Ципес Г. Л. Управление проектами: стандарты, методы, опыт Москва: Олимп-Бизнес, 2003. 240 с.

36. Guide to the software engineering body of knowledge (SWEBOK): Version 3.0. / ed. Bourque P., Fairley R. E. IEEE Computer Society, 2014. 335 p.

37. Моделі життєвого циклу, принципи і методології розробки програмного забезпечення. URL: https://evergreens.com.ua/ua/articles/software-development-metodologies.html (дата звернення: 17.11.2021)

38. Александрова Т. В. Повышение эффективности проектного управления в

организации на основе гибкой методологии Agile. *Экономика и бизнес: теория и практика*. 2019. №9. С.11-15.

39. Вербицький Н. І. Життєвий цикл ІТ-проєктів із штучним інтелектом. Міжнародна наукова інтернет-конференція *Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 63)*. Тернопіль. 2021. С.17-18.

40. Вербицький Н. І. SWOT-аналіз моделі комп'ютеризованої системи управління проєктами. *Сучасні виклики і актуальні проблеми науки, освіти іта виробництва: Міжгалузеві диспути*. Матеріали XXII Міжнародної науково-практичної інтернет-конференції (м. Київ, 19 листопада 2021 року). Київ. 2021. С.309-310.