Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів та моделей управління ІТ-проєктами з розробки комп'ютерних ігор
(тема)

Виконав:
здобувач 2 року навчання,
групи УПГІТм-23-2
Ярослав ДРОЗДОВ
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Управління проєктами в галузі інформаційних технологій
(повна назва освітньої програми)
Керівник проф. каф. ІУС. Костянтин ПЕТРОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Інформаційних управляючих систем

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Управління проєктами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

« 21 » квітня 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Дроздову Ярославу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів та моделей управління ІТ-проєктами з розробки комп'ютерних ігор

затверджена наказом університету від 28 березня 2025 р. № 235Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 03 06 2025 р.

3. Вихідні дані до роботи науково-технічна література, публікації та інтернет-ресурси, що стосуються теми кваліфікаційної роботи; матеріали звіту з науково-дослідницької практики

4. Перелік питань, що потрібно опрацювати в роботі аналіз предметної області; огляд існуючих методів та моделей управління ІТ-проєктами з розробки комп'ютерних ігор, виявлення їх переваг та недоліків; розробка власного комбінованого методу управління ІТ-проєктами з розробки комп'ютерних ігор; проведення експериментальної перевірки працездатності розробленого методу; аналіз отриманих результатів.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Пошук наукових публікацій, літературних джерел та інформаційних ресурсів у наукових виданнях	22.04.2025-01.05.2025	виконано
3	Огляд існуючих аналогів	02.05.2025-08.05.2025	виконано
4	Постановка задачі дослідження	08.05.2025	виконано
5	Розробка власного методу	09.05.2025-17.05.2025	виконано
6	Практична апробація розробленого методу	18.05.2025-20.05.2025	виконано
7	Аналіз отриманих практичних результатів використання розробленого методу	21.05.2025-22.05.2025	виконано
8	Оформлення пояснювальної записки	23.05.2025-28.05.2025	виконано
9	Підготовка презентації	29.05.2025	виконано
10	Надання роботи для перевірки на плагіат	30.05.2025	виконано
11	Надання роботи на рецензування	01.06.2025	виконано
12	Захист кваліфікаційної роботи в екзаменаційній комісії	05.06.2025	виконано

Дата видачі завдання _21_ __квітня_____2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. каф. ІУС Костянтин ПЕТРОВ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 61 с., 2 рис., 2 табл., 1 додаток,18 джерел.

ФРЕЙМВОРК ГНУЧКОЇ РОЗРОБКИ, КОМБІНОВАНИЙ МЕТОД, СПРИНТ, WATERFALL, SCRUM, KANBAN, LEAN, IHF.

Об’єкт дослідження – процес управління ІТ-проєктами з розробки комп’ютерних ігор.

Предмет дослідження – методи та моделі управління ІТ-проєктами з розробки комп’ютерних ігор.

Метою роботи є дослідження та проведення порівняльного аналізу методів і моделей управління ІТ-проєктами з розробки комп’ютерних ігор, а також розробка власного комбінованого методу управління, який би враховував якомога більше переваг і вирішував недоліки існуючих методів.

Дослідження, що проведені в роботі, ґрунтуються на системному аналізі методів управління ІТ-проєктами та на використанні методів комп’ютерного моделювання для оцінки ефективності запропонованого комбінованого методу управління ІТ-проєктами з розробки комп’ютерних ігор.

В процесі виконання роботи був створений комбінований метод управління ІТ-проєктами з розробки комп’ютерних ігор IHF, наведені результати експериментальної перевірки запропонованого методу, а також проведена його апробація.

Запропонований метод є корисним інструментом для керівників ІТ-компаній та менеджерів проєктів. Він спрощує процес управління ІТ-проєктами з розробки комп’ютерних ігор, забезпечує скорочення часу розробки та бюджету, а також мінімізацію втрат ресурсів за рахунок збалансованого поєднання ітеративності, гнучкості та контролю, що в свою чергу може підвищити успішність проєкту.

ABSTRACTION

Thesis contains: 61 pages, 2figures, 2 tables, 1 appendices, 18 sources.

AGILE DEVELOPMENT FRAMEWORK, COMBINED METHOD, SPRINT, WATERFALL, SCRUM, KANBAN, LEAN, IHF.

The object of research is the process of managing IT projects for the development of computer games.

The subject research is methods and models of managing IT projects for the development of computer games.

The purpose of the work is to study and conduct a comparative analysis of methods and models of managing IT projects for the development of computer games, as well as to develop our own combined method, which would take into account as many advantages as possible and solve the shortcomings of existing ones.

The research conducted in this paper is grounded on a systematic analysis of IT project management methods and on the use of computer modeling methods to assess the effectiveness of the proposed combined method of managing IT projects for the development of computer games.

As a result of the work, a combined method for managing IT projects for the development of computer games IHF was created, the results of experimental verification of the proposed combined method were presented, and its testing was also carried out.

The proposed method is a useful tool for IT company leaders and project managers. It simplifies the process of managing IT projects for the development of computer games, ensures a reduction in development time and budget, as well as minimizing resource losses due to a balanced combination of iterativity, flexibility and control, which in turn can increase the success of the project.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ІТ – інформаційні технології

ТЗ – технічне завдання

IHF – Iterative Hybrid Flow

MVP – Minimum Viable Product

WIP – Work In Progress

ВСТУП

Сучасна індустрія розробки комп'ютерних ігор є однією з технологічно складних сфер інформаційних технологій (IT), що швидко розвиваються. Останні десятиліття галузь розробки комп'ютерних ігор переживає бурхливе зростання. За даними аналітичної компанії Newzoo, у 2023 році світовий ринок відеоігор досяг обсягу в 184,4 млрд доларів, а кількість гравців перевищила 3 млрд. У зв'язку з цим зростає потреба в розробці ефективних методів управління проєктами в цій сфері. Однак, незважаючи на активне використання різних методологій управління IT-проєктами, розробка ігор залишається складним і багатогранним процесом, що включає не тільки програмування, але й роботу з художнім контентом, звуковим дизайном, анімацією і маркетингом.

Розробка комп'ютерних ігор значно відрізняється від інших IT-проєктів, тому цей процес потребує комплексного підходу до управління проєктами. На відміну від програмних продуктів, що призначені для вирішення конкретних завдань, ігри – це складні інтерактивні системи, що поєднують програмування, художнє оформлення, геймдизайн, звукорежисуру, сценарну майстерність та маркетинг. Таке поєднання вимагає не тільки грамотного управління розробкою, а й високого ступеню адаптивності до змін, що робить використання традиційних методів управління проєктами менш ефективним. Незважаючи на широке застосування, у геймдеві ці методи мають низку спільних проблем, таких як часті зриви термінів, брак гнучкості в плануванні, а також складність у координації міждисциплінарних команд.

Також серйозною проблемою є так звані «переробки» співробітників – явище, при якому розробники змушені працювати понаднормово на фінальних стадіях проєкту, що призводить до зниження якості продукту та вигоряння співробітників. Це пов'язано з недостатньою гнучкістю використовуваних методологій та нестачею ефективних інструментів планування.

Таким чином, незважаючи на наявність усталених методологій, таких як Scrum, Kanban та Waterfall, в ігровій індустрії залишається невирішеним низка проблем, що вимагають пошуку нових підходів до управління проєктами. Тож, можна дійти висновку, що існуючі методології, які широко застосовуються в ігровій індустрії, не завжди враховують її унікальні особливості. У зв'язку з цим виникає необхідність у розробці нового комплексного методу управління, який би дав змогу об'єднати найкращі сторони існуючих підходів та запропонувати рішення для специфічних проблем геймдева, та який був би здатен враховувати особливості розробки ігор та підвищіти ефективність цього процесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз процесу управління IT-проєктами з розробки комп'ютерних ігор

Управління IT-проєктами – це складний процес планування, організації, контролю та завершення розробки програмного забезпечення, що включає координацію роботи команди, управління ресурсами та дотримання термінів. У контексті ігрової індустрії цей процес набуває додаткових труднощів через багатокомпонентну структуру розробки ігор.

Процес створення відеоігор включає декілька ключових етапів:

- пре-продакшен (pre-production) – генерація ідей, написання концепту, вибір технологій, формування команди;
- продакшен (production) – програмування, створення графіки, написання сценарію, проєктування рівнів, розробка штучного інтелекту та ігрових механік;
- тестування – перевірка коду, балансування механік, виявлення та виправлення багів;
- запуск (release) – підготовка до релізу, рекламна кампанія, робота з спільнотою;
- підтримка (post-release) – виправлення помилок, випуск оновлень, доповнень, розвиток гри.

Особливістю управління ігровими проєктами є необхідність координації міждисциплінарної команди, що включає програмістів, художників, геймдизайнерів, сценаристів, звукорежисерів та тестувальників. На відміну від класичного програмного забезпечення, де основною метою є виконання певних функцій (наприклад, обробка даних, автоматизація процесів), відеоігри повинні працювати не тільки без помилок, але і бути цікавими, захоплюючими, викликати емоційний відгук у користувачів. Це призводить до унікальних викликів в управлінні проєктами, які існуючі методології не можуть ефективно вирішити [1].

Ігри являють собою поєднання різних напрямків:

- програмування (розробка движка, написання коду, оптимізація);
- геймдизайн (розробка ігрових механік, балансування);
- арт-дизайн (створення персонажів, оточення, анімацій, ефектів);
- звук (озвучення, музичний супровід, звукові ефекти);
- сценарій (сюжет, діалоги, світ гри).

Кожен із цих напрямів має свої темпи роботи, способи оцінки якості та критерії завершеності. Наприклад, програмісти можуть вимірювати прогрес за кількістю написаних функцій або виправлених багів, тоді як художники та сценаристи працюють ітеративно, покращуючи контент у процесі розробки.

Традиційні методології управління (Scrum, Waterfall, Kanban) не завжди враховують цей аспект. Agile, наприклад, добре працює для програмування, але погано адаптований до творчих завдань. У Scrum спринтах складно планувати скільки часу піде на розробку унікального візуального стилю або написання сценарію. В результаті часто виникають конфлікти між різними відділами і управління стає важко контрольованим [2].

Розробка сучасних ігор – це складний та непередбачуваний процес. На старті проєкту часто неможливо точно визначити, як виглядатиме фінальний продукт. На відміну від бізнес-додатків, де вимоги до функціонала фіксуються заздалегідь, у геймдеві багато рішень приймаються у процесі роботи.

Розглянемо основні фактори невизначеності.

1. Ігровий процес вимагає тестування та ітеративних покращень. Навіть якщо на папері деякі ігрові елементи виглядають добре, вони можуть виявитися непотрібними у грі, або може виявитися, що їх неможливо реалізувати.

2. Графіка та анімація можуть переглядатися. Якщо візуальний стиль не відповідає баченню розробника або вимогам гравців, його можуть переробити.

3. Історія та сценарій можуть змінюватися. Розробка сюжету йде паралельно зі створенням гри, і часто доводиться адаптувати оповідальні елементи під нові ігрові механіки.

Методології типу Waterfall вимагають чітких специфікацій на старті, але в розробці ігор це важко реалізувати. Agile дозволяє змінювати проєкт у процесі, але

може призвести до «нескінченної розробки», коли проєкт ніколи не досягає завершеного стану.

Через мультидисциплінарний характер розробки гри управління командою потребує складної координації. У IT-розробці є чіткі процеси передачі завдань між аналітиками, програмістами, тестувальниками. У геймдеві програмісти, художники, геймдизайнери і сценаристи повинні працювати синхронно, і збої в одному відділі можуть зупинити весь процес.

У традиційних IT-методах управління робота команди будується навколо програмного коду. У геймдеві такий підхід не враховує синхронність роботи креативних і технічних фахівців.

Проблема «переробок» – ще одна з головних проблем в ігровій індустрії. Багато студій опиняються в ситуаціях, коли перед релізом розробники працюють понаднормово тижнями та місяцями, щоб завершити проєкт у термін. Це з тим, що існуючі методології який завжди дозволяють адекватно планувати ресурси і тимчасові витрати.

Причини виникнення «переробок»:

- через невизначеність результату важко заздалегідь оцінити час розробки;
- зміна механік, графіки, рівнів може відбуватися навіть у пізніх стадіях;
- відділи залежать один від одного: якщо художники затримали моделі, програмісти не можуть завершити код;
- керівництво встановлює жорсткі дедлайни, які не враховують ітеративний процес розробки ігор.

У підходах Agile відсутнє довгострокове планування, що часто призводить до браку часу в кінці проєкту. У Waterfall ж, якщо на ранніх стадіях було прийнято помилкові рішення, на фінальних етапах вже неможливо внести зміни без величезних витрат.

Отже, жодна з існуючих методологій враховує специфіку розробки ігор повною мірою. Необхідно розробити новий підхід, який би:

- поєднував напрямки (геймдизайн, програмування, арт, звук, сценарій) в єдину керовану систему;

- буде збалансованим відносно гнучкості та жорсткими дедлайнами, дозволяючи адаптувати проєкт без ризику нескінченної розробки;
- враховував креативні процеси, дозволяючи художникам та сценаристам працювати ітеративно, а програмістам – дотримуватись чіткого плану;
- дозволяв мінімізувати кількість переробок, забезпечуючи рівномірне навантаження команди протягом усього проєкту.

Розробка такого методу управління дозволить підвищити ефективність роботи ігрових студій, знизити стрес у співробітників та покращити якість підсумкового продукту.

1.2 Опис критеріїв, які використовуються в процесі управління IT-проєктами з розробки комп'ютерних ігор

Процес управління проєктами в ігровій індустрії включає багато факторів, що впливають на успіх розробки. Для ефективного управління важливо враховувати ключові критерії, що розглядаються нижче.

1. Часові рамки та дедлайни:
- планування термінів – створення графіка розробки, що включає ключові етапи (pre-production, production, post-production);
- гнучкість термінів – можливість коригування дедлайн без втрати якості;
- контроль за переробками – запобігання ситуаціям, коли команда змушена працювати в екстремальному режимі перед релізом.

2. Якість продукту:
- відсутність критичних багів – робота QA-відділу, тестування та налагодження;
- оптимізація продуктивності – гра має стабільно працювати на цільових платформах;

- відповідність початкової концепції – механіки та дизайн мають бути реалізовані так, як задумано.

3. Баланс між технічною та творчою частиною:

- зв'язок між відділами – ефективна взаємодія програмістів, дизайнерів, художників та сценаристів;
- гнучкість у зміні концепції – можливість коригувати дизайн та геймплей без повного переписування коду;
- контроль за відповідністю арт-стилю та технічних можливостей – баланс між візуальною красою та оптимізацією.

4. Фінансове планування:

- оптимальний розподіл бюджету – фінансування ключових етапів без перевитрати;
- контроль за додатковими витратами – уникнення непотрібних витрат (наприклад, дорогих акторів озвучення, якщо гра цього не вимагає);
- вибір бізнес-моделі – визначення, чи гра продаватиметься як преміум-продукт, free-to-play або за підпискою.

5. Оптимізація роботи команди:

- чіткий розподіл ролей – кожен співробітник має розуміти свою зону відповідальності;
- мотивація та залучення – система бонусів та прозорих кар'єрних перспектив;
- мінімізація плинності кадрів – уникнення ситуацій, коли ключові фахівці йдуть у середині розробки.

6. Орієнтація на ринок та гравців:

- вивчення потреб аудиторії – аналіз трендів та очікувань гравців;
- дотримання жанрових стандартів – гравці очікують певного досвіду від ігрового процесу;
- зворотній зв'язок із спільнотою – альфа-тести, бета-тести, збір відгуків.

7. Гнучкість та адаптація у процесі розробки:

- здатність реагувати на зміни – можливість коригування проєкту без хаосу в управлінні;
- проведення проміжних тестів – регулярні перевірки працездатності ігрових елементів;
- пріоритетність виправлень – фокус на найкритичніших проблемах.

Ці критерії можна використовувати як ключові принципи ефективного управління IT-проєктами з розробки комп’ютерних ігор, які допомагають забезпечити успішну реалізацію проєкту та досягти поставлених цілей.

1.3 Аналіз існуючих моделей та методів управління IT-проєктами з розробки комп’ютерних ігор

Управління проєктами у сфері розробки комп'ютерних ігор – це складний процес, який потребує врахування як технічних, так і творчих аспектів. Для оптимізації процесу управління розробкою ком’ютерних ігор використовуються різні методології, кожна з яких має свої переваги та недоліки. Розглянемо найпоширеніші з них: Waterfall, Agile, Scrum і Kanban.

1. Модель Waterfall (каскадна модель).

Waterfall є лінійною послідовною моделлю, в якій кожен етап розробки завершується перед початком наступного. Типові етапи включають збирання вимог, дизайн, розробку, тестування, впровадження та підтримку [3].

Переваги:

- структурованість та передбачуваність – чітко певні етапи та результати полегшують планування та управління ресурсами;
- документованість – детальна документація на кожному етапі сприяє розумінню проєкту всіма учасниками.

Недоліки:

- негнучкість до змін – внесення змін на пізніх стадіях розробки утруднене та може бути дорогим;
- пізня інтеграція та тестування – робочий продукт стає доступним лише на завершальних етапах, що збільшує ризик виявлення критичних проблем наприкінці проєкту.

В ігровій індустрії Waterfall використовується рідко через необхідність гнучкості, але підходить для проєктів із чітко визначеними вимогами та мінімальним ризиком змін.

2. Методологія Agile (гнучка методологія).

Agile – це ітеративний підхід, орієнтований на гнучкість та адаптацію до змін. Проєкт розбивається на невеликі цикли (ітерації), кожен із яких завершується створенням працюючого продукту [4].

Переваги:

- гнучкість та адаптивність – легкість внесення змін до вимог та функціональності в ході розробки;
- раннє отримання працюючого продукту – можливість демонстрації та тестування на ранніх етапах;
- покращена комунікація – постійна взаємодія між членами команди та зацікавленими сторонами.

Недоліки:

- труднощі з довгостроковим плануванням – через фокус на короткострокових ітераціях складно оцінити тривалість та вартість проєкту;
- вимоги до високого залучення замовника – необхідність постійної участі та зворотного зв'язку від клієнта.

Agile широко використовується в ігровій індустрії завдяки гнучкості. Однак успішне застосування потребує дисципліни та розуміння методології всіма учасниками.

3. Методологія Scrum.

Scrum – це фреймворк Agile, який організує роботу команди у фіксовані часові проміжки (спринти), зазвичай 2-4 тижні. Кожен спринт завершується створенням потенційно готового випуску продукту [5].

Переваги:

- чітка структура та ролі – певні ролі (Scrum-майстер, власник продукту, команда розробки) та артефакти (беклог продукту, беклог спринту) сприяють організованості процесу;
- прозорість та контроль – щоденні стендапи та огляди спринтів забезпечують видимість прогресу та виявлення перешкод.

Недоліки:

- жорсткість у рамках спринту – зміни у завданнях спринту небажані, що може бути обмеженням у проєктах, що динамічно змінюються;
- неповна відповідність творчим процесам – строга структура може не враховувати непередбачуваність творчих завдань.

Scrum популярний серед ігрових студій, особливо у невеликих та середніх командах. Однак його застосування потребує адаптації для врахування творчих аспектів розробки ігор.

4. Методологія Kanban.

Kanban – візуальна методологія управління, що фокусується на безперервному потоці завдань. Робочий процес відображається на дошці Kanban, де задачі переміщаються між колонками, що відбивають стадії виконання [6].

Переваги:

- візуалізація процесу – прозорість поточного стану завдань та виявлення вузьких місць;
- гнучкість – відсутність фіксованих ітерацій дозволяє адаптувати пріоритети та обсяг роботи у будь-який час.

Недоліки:

- відсутність часових рамок – без чітко визначених термінів завдання може затягуватися;

- ризик перевантаження команди – без обмежень на обсяг роботи можливе накопичення завдань та навантаження учасників.

Kanban підходить для команд, які прагнуть гнучкості та прозорості. Однак для ефективного використання необхідні чіткі правила пріорітизації та контролю навантаження.

Як видно з проведенного аналізу, універсального рішення для управління IT-проєктами з розробки ігор не існує. Кожна студія та команда обирає найбільш підходящий для себе підхід, а іноді використовує гібридні методи для досягнення найкращого результату.

1.4 Постановка задачі дослідження

У ході проведення ознайомлення із сучасними методами та моделями управління IT-проєктів з розробки комп'ютерних ігор було визначено їх загальні недоліки:

- відсутність адаптації під специфіку ігрової промисловості;
- проблеми з креативними процесами;
- невизначеність результату на ранніх етапах;
- довгі цикли розробки;
- проблеми координації між відділами;
- складнощі з балансом між гнучкістю та дедлайнами;
- переробки;
- проблеми довгострокового планування.

Метою роботи є дослідження та проведення порівняльного аналізу методів і моделей управління IT-проєктами з розробки комп'ютерних ігор, а також розробка власного комбінованого методу управління, який би враховував якомога більше переваг і вирішував недоліки існуючих методів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати основні методи та моделі управління ІТ-проєктами та їх застосування у розробці ігор;
- виявити проблеми, що виникають під час використання існуючих методів в ігровій індустрії;
- розробити новий метод управління, адаптований для розробки комп'ютерних ігор;
- провести експериментальне тестування запропонованого методу та оцінити його ефективність;
- перевірити працездатність розробленого методу на практиці;
- провести аналіз і сформулювати рекомендації щодо впровадження та використання отриманих наукових результатів.

Об'єктом дослідження є процес управління ІТ-проєктами з розробки комп’ютерних ігор.

Предметом дослідження є методи та моделі управління ІТ-проєктами з розробки комп’ютерних ігор.

2 РОЗРОБКА КОМБІНОВАНОГО МЕТОДУ УПРАВЛІННЯ ІТ-ПРОЄКТІВ З РОЗРОБКИ КОМП'ЮТЕРНИХ ІГОР

Пропонується розглянути окремо кожний існуючий метод, який використовується для управління ІТ- проєктами з розробки комп'ютерних ігор, для того, щоб зрозуміти, як саме проходить розробка ігор, і з якими проблемами вона стикається з використанням певного методу.

2.1 Модель Waterfall

Waterfall, або каскадна модель – це лінійна і послідовна методологія управління проєктами, коли весь процес ділиться на певні фази, виконувані одна одною. Кожна наступна стадія починається лише після завершення попередньої. Зворотний перехід між етапами або неможливий, або допускається у виняткових випадках, що робить Waterfall дуже негнучким [7].

Модель Waterfall складається з таких основних фаз:

- збір та аналіз вимог;
- проєктування (дизайн);
- реалізація (кодинг);
- тестування;
- впровадження та випуск;
- супровід.

Waterfall застосовується переважно в інженерних та структурованих сферах розробки, де чіткі вимоги відомі заздалегідь. У цій моделі керування проєктом зосереджено на жорсткому плануванні. На етапі постановки завдань розробляється технічне завдання (ТЗ), у якому мають бути детально описані всі аспекти майбутньої гри: від механік та візуального стилю до вимог до платформи. Усі

рішення приймаються на початку, і зміни у процесі вважаються винятком, а не правилом.

Кожен етап завершується артефактом (документом, прототипом або системою), який має бути повністю завершений та затверджений, перш ніж розпочнеться наступний. Це робить Waterfall документально насиченим та жорстко структурованим методом.

Waterfall використовується в ігровій індустрії зрідка та у специфічних випадках. Це можуть бути:

- розробка ігор за ліцензією, де вихідні вимоги жорстко задані замовником (наприклад, ігри з кінофраншиз);
- розробка вузькоспеціалізованих ігрових рішень для конкретних платформ від внутрішніх студій;
- прототипні, освітні чи інді-проєкти, що виконуються за навчальними шаблонами.

Але деякі великі студії і сьогодні використовують адаптовані версії каскадної моделі, особливо щодо технічних та інфраструктурних частин проєкту (наприклад, движок, серверна архітектура тощо) [8].

Розглянемо більш детально, як Waterfall проходить етапи розробки.

1. Передвиробничий етап (Pre-production).

На цій стадії Waterfall працює найбільше ефективно, оскільки вся модель заснована на ретельному плануванні. Команда витрачає багато часу на підготовку: концептуальне опрацювання гри (жанр, аудиторія, механіки), створення повного GDD (Game Design Document), TDD (Technical Design Document), Art Bible та інших документів, узгоджуються всі питання із замовником або менеджерами. Проте творча невизначеність погано поєднується із жорсткими рамками ТЗ. Не все можна передбачити заздалегідь.

2. Прототипування (Prototype stage).

У чистому Waterfall прототипування може бути відсутнім, або бути формальним етапом всередині проєктування. Часто створюються паперові або текстові прототипи, які не дають повноцінного уявлення про геймплей. Через це

може бути важко виявити невдалі рішення на початок кодингу, або немає можливості швидко перевірити ігрову механіку в дії.

3. Виробничий етап (Production).

На цьому етапі починається реалізація всіх функцій, зазначених у документації. Розробники зрідка відходять від плану. Взаємодія між відділами (арт, код, звук) побудована навколо обміну затвердженими артефактами. Через це можуть виникнути деякі проблеми, наприклад якщо відділ відстає (наприклад, не готова графіка), зупиниться весь процес. Також якщо є якісь зміни у дизайні гри, то можливо доведеться перероблювати все, що вже написано. Команда не бачить «гру як ціле» до найпізніших стадій.

4. Тестування та налагодження.

На цій стадії QA-команда перевіряє, як працює готовий продукт. Тестування починається після повної реалізації всіх функцій. через що може можна пізно виявити баги, або невдалі дизайнерські рішення. Виправлення проблем вимагає повернення до коду, графіки, дизайну, що може порушити весь план. Ще одна проблема – не враховується ранній фідбек користувача, бо гру не можна випробувати на реальних гравцях у процесі.

5. Реліз.

Waterfall дозволяє жорстко контролювати терміни випуску. Якщо всі етапи пройшли успішно, реліз відбудеться у строк. Однак через відсутність ітеративності проблеми, виявлені на фінальній стадії, можуть залишитися у продукті. Інтерфейс та інші візуальні аспекти часто дороблюються в останній момент. Через те, що фокус стоїть на виконанні плану, а не на досвіді користувача, то у разі неуспіху продукт втрачає актуальність, а команда не може відреагувати швидко, що зрештою може призвести до провалу.

6. Пост-релізна підтримка.

Waterfall орієнтовано одноразовий випуск. Підтримка та оновлення не передбачені структурно, тому цей метод підходить лише для одиночних ігор без мультиплеєра. Також немає механізму для збирання фідбека, випуску оновлень, роботи зі спільнотою гравців.

Розглянемо загальні труднощі використання Waterfall у геймдеві.

Низька гнучкість: зміни в дизайні, механіці, інтерфейсі або візуалі спричиняють повну переробку.

Довга сліпа розробка: до тестування не можна оцінити, наскільки гра цікава.

Відсутність проміжних версій: не можна показати гру інвесторам чи гравцям до фінального релізу.

Погана сумісність із творчою природою розробки ігор.

Waterfall може бути корисним у чітко керованих, технічно насичених проєктах з мінімальною невизначеністю. Однак у контексті розробки комп'ютерних ігор – особливо креативних, інноваційних чи живих (ігри-сервіси) він вкрай неефективний. Геймдев вимагає ітеративності, зворотний зв'язок, творчої свободи та адаптивності – того, чого Waterfall практично не пропонує.

2.2 Методологія Scrum

Scrum – це фреймворк гнучкої розробки, орієнтований на ітераційне створення продукту з постійним зворотним зв'язком та участю всіх ключових сторін (розробників, замовників, гравців). Він пропонує короткі цикли (спринти) тривалістю від одного до чотирьох тижнів, у рамках яких створюється працюючий фрагмент гри – інкремент. Scrum добре підходить для проєктів, де неможливо відразу визначити вимоги, і важливо швидко перевіряти гіпотези.

Scrum особливо популярний в інді-розробці, LiveOps та мобільному геймдеві, де регулярні оновлення та підтримка гравців критичні.

Scrum – це не лише метод управління завданнями, а й спосіб організації командної роботи, в якій усі учасники залучені до процесу планування, пріоритезації та аналізу прогресу [9].

Розглянемо яким чином Scrum проходить етапи розробки.

1. Передвиробничий етап (Pre-production).

На старті формується початковий Product Backlog – впорядкований перелік функцій та вимог. Він представлений у вигляді «історій користувача»: коротких описів функціоналу з погляду гравця: що гравець має змогу бачити, що гравець може робити у грі тощо. Замість створення докладного технічного ТЗ, як у Waterfall, команда одразу розпочинає планування першого спринту. Product Owner формулює основне бачення гри, визначає ключові механіки, які необхідно перевірити, а команда визначає критерії завершеності завдань.

Переваги Scrum на цьому етапі:

- Scrum дозволяє відразу розпочати розробку, не гаючи часу на багатомісячне проєктування;
- зворотний зв'язок приходить після першого спринту – можна швидко переосмислити ідеї;
- формується загальна залученість всіх членів команди до процесу із самого початку.

Проблеми Scrum на цьому етапі:

- відсутність попереднього опрацювання архітектури може призвести до нестійкого фундаменту проєкту;
- Product Owner може недооцінити важливість технічних ризиків, якщо надто фокусується на геймдизайні;
- без чіткої концепції проєкт може розповзтися в різні боки – кожна ітерація рухатиметься в інший бік, без загального курсу.

2. Прототипування.

На цьому етапі Scrum особливо ефективний. Команда швидко створює прототипи ключових механік, збирає зворотний зв'язок та вносить коригування. Кожна ігрова механіка перетворюється на спринт-мету, і до кінця повинен бути робочий інкремент: можна протестувати стрілянину, стрибки, бій тощо.

Планування спринтів відбувається на основі пріоритетів Product Backlog, що оновлюється після кожного циклу. Scrum ідеально підходить для швидкої перевірки ігрових гіпотез: за кілька тижнів можна протестувати цілий ігровий цикл, та дозволяє командно сфокусуватися на одній задачі, уникаючи розсіювання уваги.

Також є можливість доопрацьовувати механіки та не боятися прийняти невдалі рішення. Якщо Product Owner не пріоритезує задачі правильно, то команді доведеться витрачати час на незначні завдання. Через це в свою чергу може бути недостатньо ресурсів на оптимізацію, і важливі аспекти, такі як UI/UX, продуктивность чи стабільності відкладаються на другий план.

3. Виробничий етап (Production).

Проєкт входить у фазу повномасштабної розробки. Команда продовжує працювати у спринтах, регулярно поставляючи нові інкременти: рівні, діалоги, анімації, штучний інтелект, систему завдань. Scrum допускає кілька команд, кожна з яких може мати свого Scrum Master'a та працювати над своїм блоком.

Scrum Master стежить за тим, щоб процес йшов за правилами фреймворку: щодня проводяться стендапи, спринти завершуються демонстрацією та ретроспективою. Завдяки цьому розробка має передбачуваний ритм: кожні 2-3 тижні команда демонструє прогрес, при чому Scrum дозволяє мати крос-функціональні команди, що поєднують дизайнерів, програмістів, художників та тестувальників. Короткі спринти можуть заважати виконанню складних завдань, наприклад, створення рівнів або великих систем. Якщо розробники не дотримуються критеріїв завершеності завдань, то буде накопичуватися технічний обов'язок. Вносити зміни до плану можна одразу: якщо якась механіка не працює, її можна переробити без зупинки всього проєкту.

4. Тестування та налагодження.

У Scrum тестування інтегроване у кожен спринт: завдання не вважаються завершеними, доки вони не протестовані. Це вимагає присутності QA-фахівців у команді із самого початку. Баги фіксуються у поточному чи наступному спринті. Через це немає «фінального періоду тестування», тому гра постійно підтримується у робочому стані. Баги не накопичуються, а виявляються та усуваються у контексті. Також завдяки такій моделі тестування є можливість швидко протестувати вплив нових механік на старий функціонал. Проте тестувальникам може не вистачати часу на глибоку перевірку – за щільного графіка спринтів виявляються лише очевидні баги.

5. Реліз.

Фінальний інкремент перетворюється на релізну збірку. Якщо план дотримувався, проєкт має бути до нього технічно готовий: немає критичних багів, продукт стабільний, контент допрацьовано. Реліз може збігатися із завершенням спринту або відбуватися у межах окремого релізного спринту. Після релізу частина команди залишається для підтримки, частина переводиться на інші проєкти. Фінальні доопрацювання (локалізація, компіляція під різні платформи) можуть не вкладатися у спринти.

На відміну від Waterfall, де реліз запланований від початку, Scrum не передбачає певної дати виходу. Натомість реліз планується, коли команда і Product Owner вважають, що досягнуто достатній рівень якості та функціональності. Іноді реліз намічається із зовнішніх причин (виставки, маркетинг), і тоді команда вирівнює темп під термін. Це гнучко, але може призвести до перенесення, невизначеності та проблем з плануванням маркетингу та продажів.

6. Пост-реліз (LiveOps).

Після релізу команда продовжує використовувати Scrum для випуску оновлень та нового контенту. Команда розробників переробляє Backlog на основі відгуків користувачів, аналітики, поведінки гравців. Scrum чудово підходить для оновлень кожні 2–4 тижні, постійний зворотний зв'язок дозволяє швидко реагувати на фідбек. Проте якщо спільнота гравців надто активно впливає на Product Backlog, команда може втратити фокус. Також Scrum-команда може перегрітися від постійних коротких дедлайнів.

Тож, можна дійти висновку, що Scrum – ефективний інструмент в умовах високої невизначеності, особливо на ранніх етапах або LiveOps. Але він не замінює повноцінної виробничої стратегії. Незважаючи на свою популярність та гнучкість, Scrum не позбавлений серйозних обмежень, особливо у специфіці геймдева. Для комплексної розробки ігор часто доводиться доповнювати іншими підходами – Kanban, технічними документами, архітектурним проєктуванням, а також продюсерським управлінням [10].

Розглянемо основні причини, чому він не завжди підходить у чистому вигляді:

1. Складнощі з довгостроковим баченням.

У Scrum пріоритети переглядаються кожні 1-2 тижні. Це ускладнює реалізацію цільної сюжетної кампанії, де треба продумувати все наперед, або монолітних систем, таких як бойова система або генерації світу, що вимагають багато часу до першого результату.

2. Недоліки під час створення художнього контенту.

Графіка, музика, кат-сцени – це трудомісткі процеси. Художники та композитори часто не можуть працювати за короткими спринтами.

3. Проблеми при масштабуванні.

У великих командах (понад 50 людей) Scrum може стикаєтися з організаційним навантаженням і проблемами узгодження спринтів, особливо під час розподіленої розробки.

4. Не охоплює бізнес-аспекти.

Scrum не включає управління маркетингом, релізною стратегією, бюджетом, аналітичну та монетизаційну роботу після релізу [11].

2.3 Методологія Kanban

Kanban – це методологія управління потоком завдань, що прийшла з виробничої системи Toyota. В IT-галузі вона адаптована для візуального контролю над роботою [12].

Розглянемо її основні принципи:

- візуалізація всього робочого процесу (зазвичай через дошку з колонками To Do, In Progress, Done);
- обмеження кількості завдань у роботі (Work In Progress, WIP);

- безперервна доставка та покращення без жорстко заданих ітерацій (на відміну від Scrum).

Замість фіксованих спринтів, як у Scrum, Kanban передбачає безперервний потік завдань: щойно одне завдання завершується, наступна надходить у роботу.

В розробці ігор Kanban часто застосовується:

- в невеликих командах, особливо інді-розробників, яким не потрібні формальні спринти;
- в LiveOps-проєктах та підтримці вже випущених ігор;
- в арт-відділах, де робота носить циклічний та індивідуальний характер;
- як допоміжний метод паралельно з іншими (наприклад зі Scrum).

Kanban проходить такі етапи розробки:

1. Передвиробничий етап і прототипування.

Kanban надає команді гнучкий спосіб відстежувати потік завдань, пов'язаних із розробкою концепції гри. Усі активності – від генерації ідей до раннього прототипування – заносяться у вигляді карток на дошку. Команда може вільно брати завдання працювати, виходячи зі своїх компетенцій і пріоритетів. Встановлення WIP-лімітів дозволяє не допускати навантаження. Цей метод відмінно підходить для дослідницької природи передвиробництва: коли цілі ще не чітко визначені, Kanban дозволяє вести паралельну роботу з кількома напрямами (наприклад, візуальний стиль, механіки, сюжет). Візуалізація завдань допомагає тримати фокус та бачити прогрес навіть за умов хаотичного початку проєкту, а відсутність жорстких ітерацій не заважає частим переосмисленням. Проте без структурних контрольних точок та аналізу результатів (ретроспектив) можливе застрягання у нескінченному ідеюванні. Якщо команда недосвідчена, відсутність термінів та рамок може призвести до відсутності фінальних рішень – жоден концепт не може бути завершений до кінця.

2. Виробничий етап (Production).

На етапі основного виробництва Kanban використовується для організації завдань з усіх напрямків: програмування, геймдизайн, графіка, звук. Кожне завдання представлене карткою, і робота над нею починається у міру звільнення

виконавця. Команда може гнучко реагувати на зміни – наприклад, внести правку в анімацію або переробити геймплейний елемент фідбеку. Kanban також дає можливість гнучкого перерозподілу зусиль без жорсткої прив'язки до часу і відмінно підходить для візуального відстеження прогресу великого обсягу завдань, особливо, коли над проєктом працює міждисциплінарна команда, що, в свою чергу, зручно для потокового виробництва контенту (арт, рівні, інтерфейс). Відсутність спринтів та контрольних точок може призвести до розфокусування, якщо команда працює над різноспрямованими завданнями. Проєкт може втратити відчуття загального прогресу, а тому стає складніше відстежити, коли продукт буде готовий до релізу.

3. Тестування та налагодження.

Завдання з тестування та виправлення багів заносяться на дошку як нові картки. Kanban дозволяє швидко реагувати на звіти тестувальників, відтворювати баги, а потім відстежувати їхнє виправлення. Як тільки знайдено баг, він відразу стає завданням, яке потрапляє в робочий потік. Але через те, що немає чіткого розуміння пріоритетів, якщо завдання не впорядковано, то можна витрачати час менш важливі баги. Відсутність чіткого циклу «тест – виправлення – регресія» може призвести до повторних помилок, якщо не налагодити дисципліну та документацію.

4. Реліз.

Фінальні завдання зі збирання проєкту, підготовки маркетингових матеріалів, інтеграції SDK та виходу на платформи вносяться на дошку та виконуються в міру можливості. Kanban дозволяє гнучко планувати та змінювати завдання, якщо раптово з'являються термінові вимоги від платформ чи видавців, та добре підходить для роботи над розрізненими та дрібними завданнями, що не потребують довгострокового планування. З іншого боку немає жорстких термінів та релізного тиску, як у Waterfall чи Scrum, тому реліз продукту може сильно затягнутися. Також ускладнюється синхронізація всіх компонентів гри до однієї дати.

5. Підтримка та оновлення (Post-release / LiveOps).

На цьому етапі Kanban показує себе найефективніше. Підтримка гравців, випуск оновлень та нового контенту реалізуються через безперервний потік завдань. Картки з технічної підтримки, відгуків гравців, балансування та нового контенту надходять на дошку та обробляються послідовно. Kanban відмінно працює для малих автономних команд, які займаються LiveOps, і він ідеальний для безперервної підтримки продукту: легко відстежувати та пріоритизувати звіти гравців. Проте якщо фідбек гравців занадто багатий, дошка може швидко переповнитися безліччю дрібних завдань. Через відсутність системи «спринтів» чи циклів немає аналізу ефективності рішень, що критично для довгострокового розвитку проєкту [13].

Отже, Kanban у розробці ігор показує найкращу ефективність на етапах передвиробництва, тестування та підтримки, де гнучкість та адаптивність важливіші за чітке планування. Однак на етапах виробництва та релізу метод страждає від відсутності структури, синхронізації та пріоритезації, що може уповільнити розробку проєкта або призвести до затягування термінів. У великих проєктах його часто поєднують з іншими методологіями, щоб компенсувати ці слабкості.

2.4 Метод Lean

Lean – це підхід, основна мета якого – усунення всіх видів втрат, забезпечення максимальної цінності для користувача за мінімальних витрат. Він фокусується на потоці створення цінності, гнучкості, ітеративності та постійному поліпшенні. В розробці ігор Lean часто поєднується з Agile або Kanban, але сам собою він представляє набір принципів, а не чітку процесну модель. Проте можна відстежити, як він застосовується на основних етапах розробки ком'ютерних ігор [14].

Розглянемо більш детально, як Lean проходить етапи розробки.

1. Передвиробничий етап.

Lean пропонує починати з мінімального – з ідеї, яка може бути реалізована швидко та з найменшими витратами, щоб перевірити, чи потрібна вона гравцям. Застосовується принцип «прибрати все зайве». Команда може почати з простого концепту (наприклад, «що, якщо змішати платформер і карткову механіку?») і одразу зосередитись на цінності для гравця, а не на масштабності.

Переваги:

- не витрачається час на велику документацію та тривалу розробку ТЗ;
- висока гнучкість на старті.

Проблеми:

- може бракувати формалізації: за відсутності чіткої документації ідеї губляться, команда може розуміти концепт по-різному;
- команда може занадто рано відмовитися від ідеї, не встигнувши її розкрити через надмірну орієнтацію на мінімальні витрати.

2. Прототипування.

Lean заохочує швидке створення MVP (Minimum Viable Product) - прототипу, що демонструє ключову цінність ідеї. Мета – перевірити гіпотезу, а не зробити гарний та закінчений продукт. Прототип швидко показується користувачам, збирається фідбек.

Переваги:

- мінімальні витрати часу та ресурсів;
- отримання зворотного зв'язку ще до початку «справжньої» розробки.

Проблеми:

- може виникнути бажання випустити сирий продукт як фінальний, особливо якщо фідбек виглядає позитивним.
- за слабкої організації командної роботи може виникнути безліч гіпотез та прототипів без системного підходу.

3. Виробничий етап.

На етапі основної розробки Lean вимагає мінімізації незавершеної роботи (WIP), оптимізації робочого процесу, постійного виявлення проблем у розробці та постійну синхронізацію з користувачами.

Переваги:

- висока ефективність за рахунок усунення непотрібних фіч;
- емпірична перевірка кожного компонента – те, що не додає цінності, видаляється.

Проблеми:

- важко визначити, що справді «не додає цінності» – особливо в іграх, де цінність часто суб'єктивна (наприклад, естетика, атмосфера);
- можлива переоптимізація – проєкт стає функціональним, але втрачає харизму та глибину.

4. Тестування.

Тестування в Lean інтегровано у процес розробки. Lean орієнтовано на безперервне поліпшення, тому помилки розглядаються як можливість зростання. Команда повинна постійно збирати дані, швидко усувати дефекти та постійно допрацьовувати гру на основі фідбеку.

Переваги:

- не накопичуються баги – проблеми усуваються зразу;
- орієнтація на цінність змушує зосередитись на тому, що дійсно заважає гравцю.

Проблеми:

- якщо немає хорошої автоматизації та дисципліни, можна застрягти у дрібних ітераціях та не досягти готового стану;
- перехід до наступної стадії може постійно відкладатися.

5. Реліз та підтримка.

Lean пропонує не робити великих релізів, а постачати цінність у міру готовності. Це призводить до ідеї безперервного релізу (continuous delivery). Особливо підходить для live-сервісу ігор.

Переваги:

- гравці швидко одержують новий контент;
- команда може оперативно реагувати на фідбек та ринкову ситуацію.

Проблеми:

- потрібен високий рівень DevOps та QA-автоматизації;
- гра може сприйматися як вічно недороблена, особливо якщо MVP був надто урізаним.

Таким чином, Lean чудово підходить як допоміжний філософський підхід для відстеження зайвого, швидкого прототипування та економії ресурсів. Але сам він не забезпечує повної структури і може нашкодити художній якості проєкту. Тому в розробці ігор Lean найчастіше комбінується з іншими підходами, наприклад, Kanban або Agile [15].

Розглянемо основні причини, чому Lean не є універсальнім рішенням для розробки комп'ютерних ігор.

1. Складнощі з визначенням «цінності».

Lean орієнтований на усунення всього, що приносить безпосередню цінність користувачеві. Однак у іграх цінність часто суб'єктивна: атмосфера, художній стиль, незвичайна механіка чи глибокий сюжет можуть давати миттєвої віддачі, але бути критично важливими сприйняття. Через це Lean може видалити важливі, але неочевидні елементи.

2. Ризик перезастосування MVP-підходу.

Lean вимагає якомога раніше створити мінімально життєздатний продукт (MVP). У розробці ігор це може призвести до того, що гра або випускається сирою та неповною, або отримує погані відгуки, які не відображають її реального потенціалу, або залишається на стадії «вічного MVP», ніколи не перетворюючись на завершений цілісний продукт.

3. Проблеми з художнім та креативним змістом.

Lean добре працює в проєктах, де фокус – на функції та ефективності. Але ігри – це не просто програмний продукт, а певною мірою форма мистецтва. Часто те, що неоптимально, робить гру унікальною, і за рахунок цього успішною. Lean може витісняти креативність заради продуктивності.

4. Розмита структура процесу.

Lean – це не методологія із жорсткими правилами, а набір принципів. В умовах обмеженого бюджету, недосвідченості чи тиску термінів це може призвести до того, що команда не знатиме, як діяти.

2.5 Розробка комбінованого методу

На основі розглянутих існуючих методів та їх застосування в ігровій індустрії було виявлено потребу в розробці комбінованого методу управління IT-проєктами з розробки комп'ютерних ігор, що поєднує переваги різних підходів та адаптованої спеціально під процес розробки відеоігор. Як рішення пропонується комбінований метод під назвою Iterative Hybrid Flow (IHF) – Ітеративно-Гібридний Потік.

IHF являє собою гібридну управлінську модель, структуровану поетапно, яка при цьому спирається на гнучкий ітеративний цикл, що дозволяє адаптуватися до вимог і особливостей ігрового дизайну, що змінюються.

Метод поєднує в собі такі ключові принципи.

1. Фазова структура розробки – як у Waterfall, але без суворої лінійності. Кожна фаза логічно завершує певний блок завдань, створюючи стійку основу наступного етапу.

2. Ітеративний розвиток – як у Scrum. Кожен етап допускає внутрішні цикли поліпшень без поступу, поки фаза не буде визнана завершеною.

3. Гнучкість та візуальне управління – як у Kanban. Використання візуальних дошок та обмежень WIP допомагає контролювати обсяг завдань та знижує ефект перевантаження команди.

4. Фокус на цінність та усунення втрат – як у Lean. На кожному етапі проводиться фільтрація ідей та функцій, які не несуть геймплейної або користувальницької цінності.

Отже, IHF не просто комбінує підходи, а формує адаптивний робочий процес, орієнтований на специфіку створення ігор.

Метод IHF складається з шести ключових фаз, що проходять послідовно, але допускають повернення на ранні етапи в рамках ітеративного циклу.

Розглянемо ці фази.

1. Vision & Core (Формування ядра) – постановка ідеї, жанрової концепції та основних геймплейних механік Розробка першого прототипу, що демонструє ключовий ігровий цикл.

2. Systems & Architecture (Технічний та дизайн-фундамент) – побудова архітектури проєкту, створення внутрішніх редакторів, інструментів, основних механік.

3. Iterative Expansion (Розширення та масштабування) – додавання контенту, рівнів, ворогів, завдань, діалогів тощо; паралельна робота кількох підгруп над різними частинами проєкту у межах загальної архітектури.

4. Testing (Тестування та шліфування) – багфікси, внутрішня та зовнішня валідація, бета-тести, збір зворотного зв'язку, робота над балансом та якістю користувальницького досвіду.

5. Release (Фіналізація та публікація) – підготовка фінальної збірки, сертифікація, маркетингова упаковка, публікація на платформах та запуск.

6. Post-Launch Iteration (Підтримка після релізу) – у разі тривалих проєктів – випуск оновлень, доповнень, реагування на відгуки, розвиток спільноти.

Кожна з фаз має свою чітку мету, власні критерії завершеності та внутрішні цикли ітерацій.

Відмінні риси методу:

- ітеративність із фазовими обмежувачами – на відміну від Scrum, де фічі можуть з'являтися на будь-якому етапі, IHF фаза обмежує тип завдань, які допустимі - наприклад, на етапі «Release» заборонені зміни геймплею;
- контроль архітектури – завдяки виділеній фазі Systems & Architecture, знижується ризик технічної нестабільності та переосмислень на пізніх етапах;
- поділ креативу та оптимізації – фази Iterative Expansion і Testing поділяють додавання нового контенту та його налагодження, дозволяючи команді сконцентруватися на одній меті за ітерацію;

- органічна підтримка мультидисциплінарних команд – метод добре масштабується: програмісти, дизайнери, художники та сценаристи можуть працювати паралельно, зберігаючи єдиний вектор.

Передбачувані переваги IHF:

- підвищення керованості великих ігрових проєктів без втрати гнучкості;
- поліпшення балансу між творчістю та інженерною дисципліною;
- усунення проблем «вічної бети» та релізів з низькою якістю;
- спрощення пріорітизації завдань та відсікання вторинних функцій.

Потенційні недоліки IHF:

- не підходить для дуже коротких проєктів та дуже малих команд;
- може бути складним в опануванні для зовсім недосвідченої команди – вимагає розуміння відразу кількох методологій;
- труднощі з масштабуванням у розподілених командах. Метод передбачає постійну комунікацію, коригування завдань, фокус на крос-функціональну взаємодію. Якщо команда розподілена по регіонах та часовим поясам, особливо без досвідченого продюсера, виникає загроза розриву в комунікації та провалів між фазами.

3 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ОЦІНКА ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ РОЗРОБЛЕНОГО МЕТОДУ

Для демонстрації працездатності розробленого методу IHF, концепцію якого описано в підрозділі 2.5, пропонується розлягнути гіпотетичний проєкт, для якого будуть використовуватися такі вхідні дані:

1. Обсяг проєкту (в умовних одиницях роботи) – це міра обсягу завдань, які необхідно виконати. Наприклад: 100 умовних завдань (фічі, механіки, екрани, системи тощо)

2. Продуктивність команди (швидкість роботи, од./міс) – середня кількість роботи, яку команда може виконати за місяць за ідеальних умов. Може відрізнятися залежно від методології (через управління, ітерації, зустрічі тощо)

3. Коефіцієнт втрат (рефакторинг, відкати, переробки) – наскільки більше роботи фактично виконується через повернення та виправлення. Наприклад, за 10% втрат проєкт зі 100 од. фактично вимагатиме 110 од. роботи.

4. Накладні витрати (мітинги, документація, планування тощо) – час, не витрачений безпосередньо на виробництво, але необхідний методологією. Виражається у частці від загального обсягу.

5. Місячний бюджет команди – скільки коштує робота команди за місяць. Наприклад: 2000 у.о. на місяць (включає ЗП, оренду тощо).

У рамках порівняльного аналізу методологій управління розробкою комп'ютерних ігор були використані умовні числові показники, що відображають ключові параметри ефективності: тривалість розробки (у місяцях), загальний бюджет проєкту (в умовних грошових одиницях) та рівень втрат ресурсів (у відсотках від бюджету). Оскільки реальні показники залежать від безлічі факторів і сильно варіюються залежно від специфіки конкретного проєкту та команди, для цілей моделювання та порівняння були прийняті умовні значення, що ґрунтуються на узагальнених даних з практики розробки та дослідницької літератури. Дані умовні значення є ілюстрацією порівняльних переваг і недоліків методологій,

дозволяючи наочно продемонструвати потенціал комбінованого методу. Реальні показники залежатимуть від багатьох факторів, включаючи специфіку проєкту, склад та досвід команди, а також зовнішні обставини.

Вхідні дані проєкту.

1. Назва проєкту: 3D гра в жанрі RPG.
2. Об’єм проєкту: 100 умовних одиниць (у. о.).
3. Місячний бюджет команди: 2000 у. о.

Умовні характеристики методів та методологій представлені в табл. 3.1.

Таблиця 3.1 – Характеристики методів та методологій

Метод/Методологія	Продуктивність(од/міс)	Втрати (%)	Накладні (%)
Waterfall	10	15	10
Scrum	12	10	5
Lean	13	5	2
Kanban	12,5	6	3
IHF	14	3	4

Розрахунки скоригованого об'єму робіт представлені нижче.

$$V_a = V * (1 + L + O), \quad (3.1)$$

де V – об'єм робіт;

L – втрати;

O – накладні.

1. Waterfall: $100 * (1 + 0{,}15 + 0{,}10) = 100 * 1{,}25 = 125$ од.
2. Scrum: $100 * (1 + 0{,}10 + 0{,}05) = 100 * 1{,}15 = 115$ од.
3. Lean: $100 * (1 + 0{,}05 + 0{,}02) = 100 * 1{,}07 = 107$ од.
4. Kanban: $100 * (1 + 0{,}06 + 0{,}03) = 100 * 1{,}09 = 109$ од.
5. IHF: $100 * (1 + 0{,}03 + 0{,}04) = 100 * 1{,}07 = 107$ од.

Розрахунки часу розробки представлені нижче.

$$T = V_a/P, \tag{3.2}$$

де V_a– скоригований об'єм робіт;

P – продуктивність.

1. Waterfall: $125 / 10 = 12{,}5$ міс.
2. Scrum: $115 / 12 = 9{,}58$ міс.
3. Lean: $107 / 13 = 8{,}23$ міс.
4. Kanban: $109 / 12{,}5 = 8{,}72$ міс.
5. IHF: $107 / 14 = 7{,}64$ міс.

Розрахунки бюджету представлені нижче.

$$B = T * B_m, \tag{3.3}$$

де T – час розробки;

B_m – місячний бюджет.

1. Waterfall: $12{,}5 * 2000 = 25000$ у.о.
2. Scrum: $9{,}58 * 2000 = 19166$ у.о.
3. Lean: $8{,}23 * 2000 = 16460$ у.о.
4. Kanban: $8{,}72 * 2000 = 17440$ у.о.
5. IHF: $7{,}64 * 2000 = 15280$ у.о.

Результати розрахунків параметрів проєкту з використанням різних методів та методологій представлені в табл. 3.2.

Таблиця 3.2 – Параметри проєкту (результати моделювання)

Метод/Методологія	Об’єм (од.)	Час (міс.)	Бюджет (у.о.)
Waterfall	125	12,5	25000
Scrum	115	9,58	19166
Lean	107	8,23	16460
Kanban	109	8,72	17440
IHF	107	7,64	15280

Для наочної демонстрації отриманих результатів було побудовано дві діаграми.

Стовчикова діаграма, що зображена на рис. 3.1, ілюструє тривалість розробки проєкту (у місяцях), залежно від обраної методології. Вона дозволяє візуально порівняти терміни реалізації проєкту під час використання різних підходів.

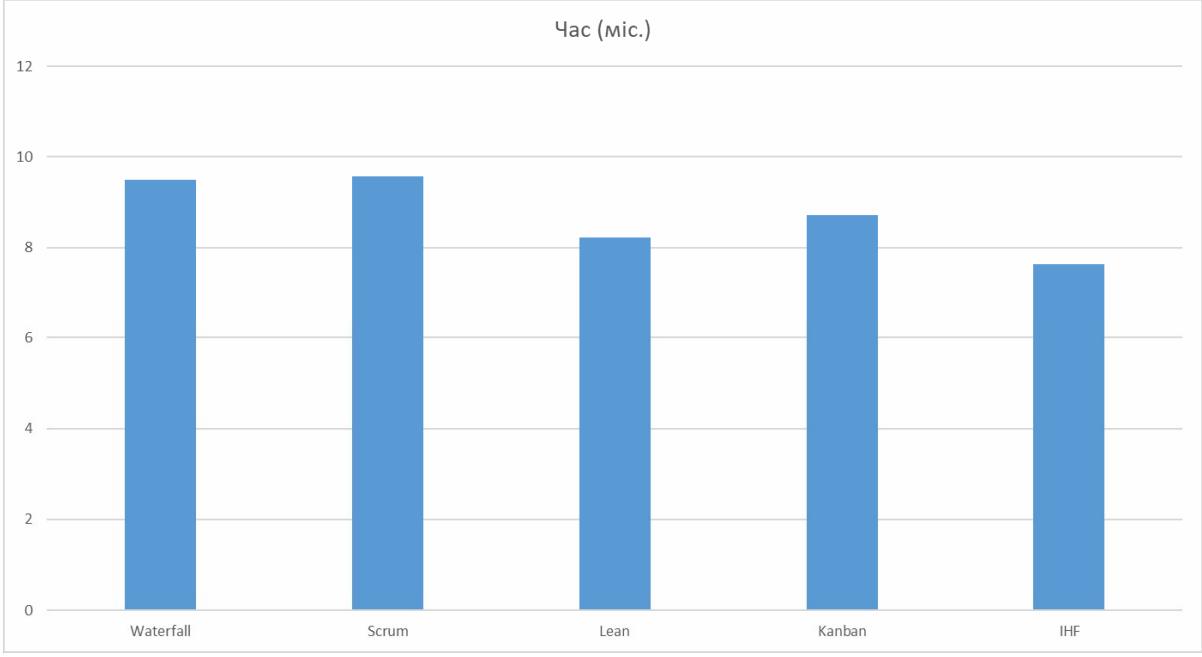

Рис.3.1 – Стовпчикова діаграма часу розробки

Стовпчикова діаграма, що зображена на рис. 3.2, демонструє бюджетні витрати (в умовних одиницях) на реалізацію проєкту в межах кожної методології. З

її допомогою можна легко визначити, які методи забезпечують найбільш економічну реалізацію.

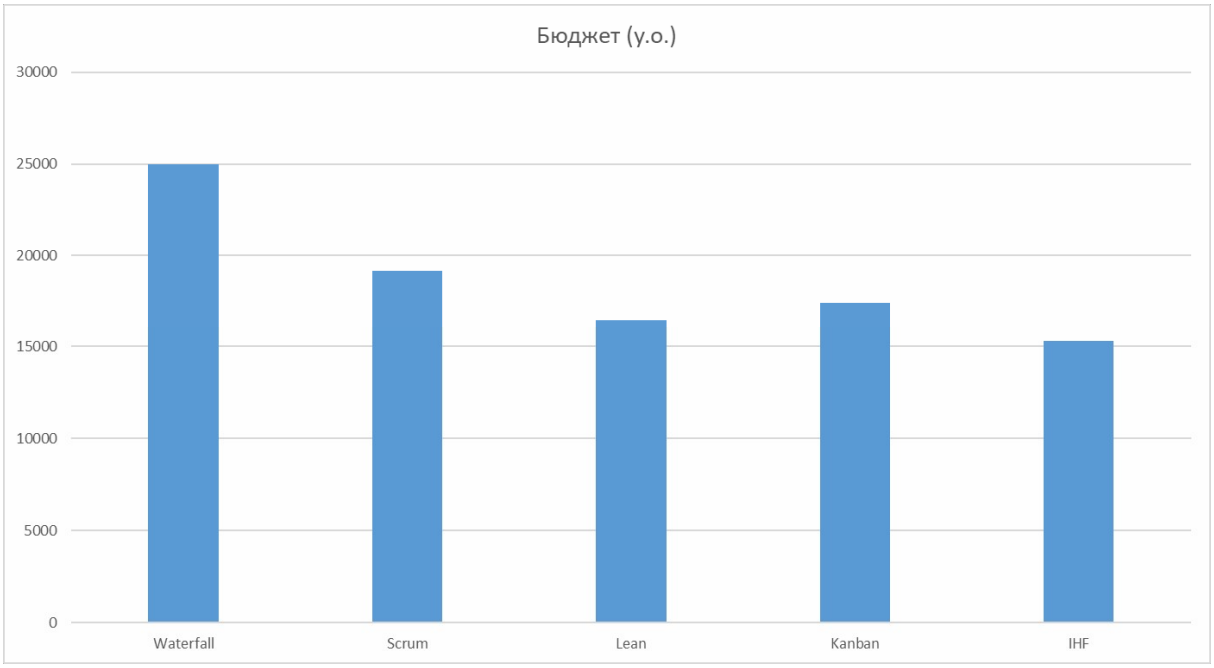

Рис.3.2 – Стовчикова діаграма бюджету проєкту

Побудовані діаграми дозволяють наочно продемонструвати різницю між методами та підтвердити ефективність розробленого комбінованого методу IHF у контексті управління ігровими проєктами.

Результати моделювання показали, що традиційна модель Waterfall характеризується найбільшою тривалістю циклу розробки – 12,5 місяців. Це пояснюється жорсткою послідовністю етапів та обмеженими можливостями для гнучкого реагування на зміни вимог у процесі роботи. Незважаючи на суворий контроль термінів на кожному етапі, загальна тривалість проєкту збільшується через відсутність ітеративності та можливість повернення до попередніх стадій.

Методологія Scrum завдяки ітеративному підходу та регулярним спринтам забезпечує скорочення часу до 9,58 місяців. Гнучкість процесу дозволяє швидше виявляти та виправляти помилки, проте накладні витрати на організацію зустрічей та планування спринтів збільшують час у окремі періоди.

Методологія Kanban продемонстрував показник часу 8,72 місяців, завдяки можливості безперервного потоку завдань та гнучкого управління пріоритетами. Тим не менш, відсутність жорстких часових рамок може призводити до затягування деяких етапів, що відбивається на загальній тривалості.

Метод Lean, орієнтований на максимальне скорочення втрат та оптимізацію процесів, показав трохи коротший час розробки – 8,23 місяців. Ефективне виявлення та усунення неефективних дій, а також фокус на цінності для користувача дозволяють прискорити виробництво.

Розроблений комбінований метод Iterative Hybrid Flow показав час розробки у 7,64 місяці – найкращий показник серед усіх розглянутих методологій. Це досягається рахунок об'єднання ітеративності Scrum, гнучкості Lean і суворого контролю ключових етапів. Можливість повертатися до попередніх фаз без значних втрат часу сприяє своєчасному коригуванню курсу та підвищенню ефективності.

У частині бюджетних витрат Waterfall також займає лідируючу позицію щодо найбільших витрат – 25000 умовних одиниць. Відсутність гнучкості призводить до збільшення витрат на виправлення помилок та переробки, особливо на пізніх стадіях.

Методології Scrum і Kanban показують схожі витрати в 19166 і 17440 у.о. зумовлені витратами на організацію процесу та ресурси, що витрачаються на ітеративні покращення та підтримку комунікацій.

Метод Lean завдяки оптимізації процесів та мінімізації втрат зміг знизити бюджет до 16,460 у.о., що свідчить про значну ефективність у користуванні ресурсами.

Комбінований метод IHF забезпечує бюджет у розмірі 15,280 у.о, що зумовлено оптимальним поєднанням ітеративного підходу з чітким контролем ресурсів. Це дозволяє уникнути надлишкових витрат, типових для методологій із менш структурованим управлінням.

Модель Waterfall через свою лінійність схильна до високих втрат на етапах тестування та виправлення помилок, які виявляються пізно, що веде до додаткових витрат часу та грошей.

Методології Scrum та Kanban знижують ці втрати завдяки безперервному зворотному зв'язку та гнучкому управлінню завданнями, проте накладні витрати на підтримку процесу залишаються суттєвими.

Метод Lean значно зменшує втрати за рахунок постійного аналізу та усунення неефективних дій, проте потребує високої кваліфікації команди та дисципліни.

Метод IHF, комбінуючи переваги всіх перелічених підходів, мінімізує ризики за рахунок адаптивного управління етапами і своєчасного реагування зміни. Проте цей метод також вимагає від команди високої організації та досвіду для успішної реалізації.

Порівняльний аналіз показує, що комбінований метод Iterative Hybrid Flow має високий потенціал для підвищення ефективності управління проєктами в розробці відеоігор. Вона забезпечує скорочення часу розробки та бюджету, а також мінімізацію втрат ресурсів за рахунок збалансованого поєднання ітеративності, гнучкості та контролю.

Проте слід зазначити, що ці результати є теоретичними і засновані на моделюванні з умовними параметрами. Реальна ефективність IHF буде залежати від конкретних умов проєкту, кваліфікації команди та культури розробки. Для підтвердження практичної доцільності цього методу необхідні пілотні проєкти та подальше емпіричне дослідження.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено комбінований метод Iterative Hybrid Flow (IHF) для управління IT-проєктами з розробки комп'ютерних ігор. Цей метод розроблено таким чином, щоб якнайкраще враховувати специфіку розробки відеоігор.

У першому розділі «Аналіз предметної області та постановка задачі дослідження» проведено огляд та загальний аналіз проблеми, яка розглядається в роботі, проаналізовано існуючи методи, які використовуються для розробки відеоігор, та здійснена постановка задачі.

У другому розділі «Розробка комбінованого методу управління IT-проєктів з розробки комп'ютерних ігор» на основі попередніх досліджень були описані принципи роботи існуючих методів та моделей в умовах розробки ігор, а також описано концепцію розробленого комбінованого методу IHF.

Третій розділ «Експериментальна перевірка працездатності та оцінка ефективності використання розробленого методу» присвячений демонстрації використання розробленого методу на прототипі гіпотетичного проєкту.

На основі отриманих результатів, можна зробити висновок, що запропонований метод IHF має високий потенціал для підвищення ефективності управління проєктами в розробці відеоігор. Він забезпечує скорочення часу розробки та бюджету, а також мінімізацію втрат ресурсів за рахунок збалансованого поєднання ітеративності, гнучкості та контролю.

Однак, слід врахувати певні обмеження цього дослідження. Для цілей моделювання та порівняння довелося прийняти умовні значення, що ґрунтуються на узагальнених даних. Реальні показники важко визначити, бо вони залежать від багатьох факторів: специфіка проєкту, склад та досвід команди, а також зовнішні обставини.

Роботу виконано відповідно до вимог методичних вказівок [16]. При оформленні даного звіту використано державні стандарти України [17, 18].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дроздов Я.Д. Дослідження методів та моделей управління IT-проєктами з розробки комп'ютерних ігор / Я. Д. Дроздов; наук. керівник д.т.н., проф. Петров К.Е. // Радіоелектроніка та молодь у XXI столітті : матеріали 29-го Міжнар. молодіж. форуму, 16–18 квітня 2025 р. – Харків : ХНУРЕ, 2025. – Т. 6, – С. 128–129.

2. Effective Project Management in Game Development. URL: https://www.argentics.io/effective-project-management-in-game-development (дата звернення 25.04.2025)

3. Waterfall Game Development Done Right. URL: https://www.gamedeveloper.com/business/waterfall-game-development-done-right (дата звернення 25.04.2025)

4. Agile Game Development With Scrum: Teams. URL: https://www.gamedeveloper.com/production/agile-game-development-with-scrum-teams (дата звернення 25.04.2025)

5. Game Development with Scrum methodology URL: https://www.researchgate.net/publication/343114637_Game_Development_with_Scrum_methodology (дата звернення 25.04.2025)

6. Kanban for Game Development URL https://kanbantool.com/blog/kanban-for-game-development (дата звернення 25.04.2025)

7. Waterfall Model Case Study. URL: https://ru.scribd.com/doc/117927198/waterfall-model-case-study (дата звернення 02.05.2025)

8. Case Study On Waterfall Model. URL: https://japosnefetua.medium.com/case-study-on-waterfall-model-bcd78073c896 (дата звернення 02.05.2025)

9. Using Scrum in «Real World» Game Production. URL: https://www.gamedeveloper.com/audio/using-scrum-in-real-world-game-production (дата звернення 02.05.2025)

10. The Use of Gamification for Learning SCRUM: Findings from a Case Study with Information Systems Students. URL: https://www.mdpi.com/2813-4346/3/2/14 (дата звернення 02.05.2025)

11. Applying Scrum in A Game Development Life Cycle For Small Scale Game Project. URL: https://ijitra.com/index.php/ijitra/article/view/73 (дата звернення 02.04.2025)

12. May Kanban Be With You: Practical Case Study. URL: https://dev.to/zerocodilla/may-kanban-be-with-you-practical-case-study-3bjd (дата звернення 03.04.2025)

13. Kanban Pizza Game. URL: https://www.agile42.com/en/agile-teams/kanban-pizza-game (дата звернення 03.05.2025)

14. Wasting Less with Lean Game Dev. URL: https://www.gamedeveloper.com/production/wasting-less-with-lean-game-dev (дата звернення 04.05.2025)

15. Applying the Lean Methodology to Game Development. URL: https://www.linkedin.com/pulse/applying-lean-methodology-game-development-zafer-elcik (дата звернення 04.05.2025)

16. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проєктами в галузі інформаційних технологій» / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2024. – 24 с.

17. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Чинний від 2017-07-01. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.

18. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016 07 01. – Вид. офіц. Київ: УкрНДНЦ, 2016. – 16 с.