ПРОГРАМНИЙ ПРОТОТИП АДАПТЕРНОГО ШАРУ ДЛЯ ПОРТИНГУ МЕРЕЖЕВОГО МОДУЛЯ C#-ІГРОВОГО ЗАСТОСУНКУ

Шпильовий М. М.
Київський столичний університет імені Бориса Грінченка, м. Київ

ВСТУП

Сучасний ринок відеоігор характеризується значною кількістю проєктів, реалізованих на рушіях із застарілою архітектурою, які вже не відповідають актуальним технічним вимогам. Зокрема, кооперативна мультиплеєрна гра Barotrauma - підводний симулятор виживання з відкритою екосистемою модифікацій - побудована на модифікованій версії рушія MonoGame. Для реалізації мережевої взаємодії між гравцями використовується бібліотека Lidgren Network, остання версія якої вийшла у 2020 році та з того часу не підтримується її розробниками.

Актуальність і постановка проблеми. Наслідком використання застарілої мережевої бібліотеки є стійкі технічні проблеми, які неможливо усунути без кардинальних архітектурних змін: нестабільний пінг, часті розриви з'єднання при навантаженні понад чотири-шість одночасних гравців, а також збої у роботі системи модифікацій при активній мережевій взаємодії. Першопричиною цих проблем є те, що підтвердження доставки пакетів у поточній реалізації покладено на ігрову логіку, а не на транспортний рівень, що призводить до нелінійного зростання затримок зі збільшенням кількості учасників. Зазначені проблеми систематично задокументовано спільнотою у публічному репозиторії проєкту у вигляді понад двохсот відкритих звернень [4].

Перспективним технічним рішенням є перехід до платформи Godot Engine 4.x, яка оснащена вбудованим мережевим стеком на основі протоколу ENet - надійної бібліотеки для передачі даних через UDP з каналізованим контролем доставки пакетів. Godot Engine 4.x також забезпечує нативну підтримку мови C# та має активну спільноту розробників. Попри очевидну технічну доцільність такого переходу, жодного систематизованого програмного рішення, яке б дозволяло здійснити портинг мережевого модуля з MonoGame на Godot Engine, досі не існує, що й зумовлює актуальність пропонованого дослідження.

Мета дослідження - розробити та реалізувати програмний прототип адаптерного шару, що забезпечує портинг мережевого модуля C#-ігрового застосунку Barotrauma із MonoGame на Godot Engine 4.x шляхом прозорої підміни транспортного рівня без переписування ігрової логіки.

Аналіз останніх досліджень і публікацій. Теоретичне підґрунтя дослідження становлять праці, присвячені міграції програмного забезпечення та архітектурним патернам. Зокрема, у роботі М. Фаулера [1] ґрунтовно обґрунтовано доцільність поетапного підходу до рефакторингу

legacy-систем та детально розглянуто патерн «Адаптер» як засіб заміни інфраструктурних компонент без порушення цілісності бізнес-логіки. Питання переносу ігрових застосунків досліджувались на прикладі переходу з бібліотеки XNA на її відкриту реімплементацію FNA [2], проте специфіка міграції саме з MonoGame на Godot Engine залишається поза увагою наукової спільноти. Архітектуру вбудованого мережевого стеку Godot Engine та переваги протоколу ENet над альтернативними рішеннями висвітлено в офіційній документації платформи [3]. Теоретичні засади проєктування адаптерних інтерфейсів описано в класичній праці про шаблони об'єктно-орієнтованого проєктування [5-7], принципи побудови масштабованих мережевих архітектур ігрових застосунків — у [8]. Таким чином, незважаючи на розвиненість суміжних досліджень, програмних рішень для портингу мережевого модуля з Lidgren та MonoGame безпосередньо на Godot Engine у відкритому доступі виявлено не було, що підтверджує наукову новизну пропонованого дослідження.

Методи дослідження. У процесі виконання роботи застосовувався комплекс взаємодоповнювальних методів. Статичний аналіз вихідного коду використовувався для інвентаризації всіх точок залежності від бібліотеки Lidgren Network. Порівняльний аналіз архітектур рушіїв дозволив обґрунтувати вибір цільової платформи. Методи структурного проєктування - зокрема, шаблони «Адаптер» та «Фасад» - застосовувались при розробці прототипу. Профілювання мережевого трафіку за допомогою аналізатора пакетів забезпечувало верифікацію отриманих результатів. Функціональне та стрес-тестування мультиплеєрних сесій дозволило оцінити коректність і стабільність реалізованого рішення.

Засоби реалізації. Практична частина дослідження виконувалась із використанням таких інструментів: Godot Engine версії 4.4 з підтримкою мови C# через бібліотеку GodotSharp, платформа .NET 8 SDK, аналізатор мережевого трафіку Wireshark 4.x, бібліотека Lidgren Network 3.x, відкритий вихідний код Barotrauma версії 1.x (репозиторій GitHub), а також середовище розробки JetBrains Rider.

Розроблення прототипу здійснюється відповідно до п'ятиетапної моделі, наведеної на рис. 1.

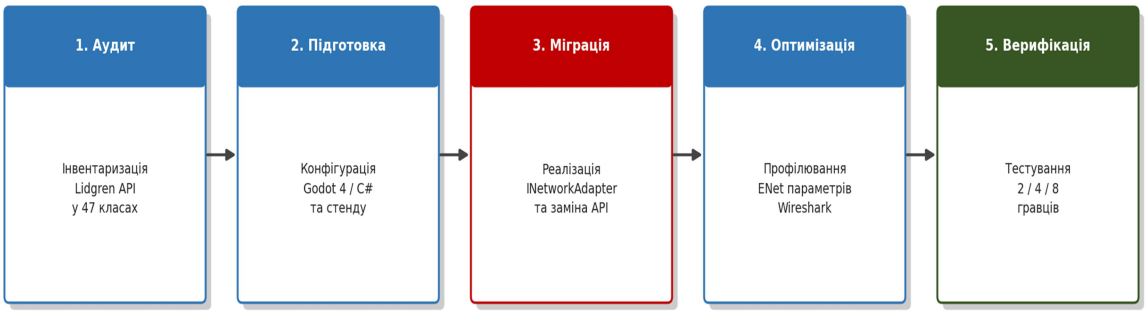

Рис. 1. П'ятиетапна модель портингу мережевого модуля

Теоретичний етап включає порівняльний аналіз архітектур MonoGame та Godot Engine 4.x (Таблиця 1), а також детальний архітектурний аналіз вихідного коду Barotrauma з акцентом на структурі мережевого модуля та точках його залежності від зовнішніх бібліотек.

Практичний етап передбачає послідовне виконання таких кроків: автоматизований аудит залежностей засобами авторського скрипту статичного аналізу; підготовку програмного середовища Godot Engine 4 з підтримкою C#; безпосередню реалізацію програмного прототипу адаптерного шару; налаштування та оптимізацію параметрів мережевого стеку; верифікаційне тестування мультиплеєрних сесій за участю двох, чотирьох та восьми гравців.

Таблиця 1

Порівняльний аналіз MonoGame та Godot Engine 4.x за ключовими характеристиками

Характеристика	**MonoGame (модиф.)**	**Godot Engine 4.x**
Мережевий стек	Lidgren (UDP, ручна реалізація)	ENet (вбудований, UDP)
Контроль доставки пакетів	Ручний (у коді гри)	Вбудований (ENet channels)
Мережева архітектура	Власний протокол поверх Lidgren	ENet + High-Level Multiplayer API
Система модифікацій	Обмежена (XML / CS-файли)	GDExtension + C# плагіни
Активна підтримка	Обмежена (community-driven)	Активна (core team + спільнота)
Масштабованість з'єднань	Деградує при понад 4 гравців	До 4095 одночасних з'єднань

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Архітектурний аналіз вихідного коду Barotrauma, проведений у квітні 2026 року на основі актуального стану репозиторію GitHub, дозволив отримати детальну картину залежностей мережевого модуля від бібліотеки Lidgren Network. Загальна кількість класів, що мають пряму залежність від зазначеної бібліотеки, становить 47. Сукупна кількість унікальних програмних викликів до функцій бібліотеки у вихідному коді дорівнює 134. За функціональним призначенням ці виклики було розподілено на чотири категорії, співвідношення яких наведено на рис. 2.

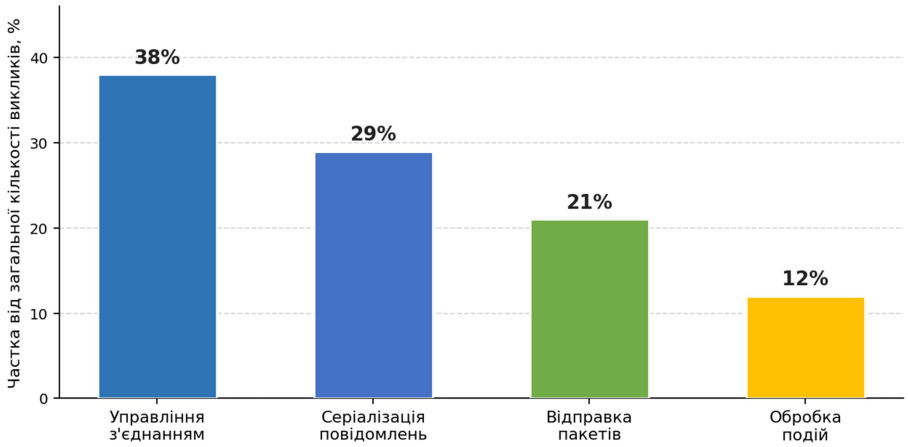

Рис. 2. Розподіл викликів бібліотеки Lidgren Network за функціональними категоріями

Встановлено, що більшість виявлених залежностей стосується управління з'єднаннями та серіалізації мережевих повідомлень. При цьому лише у дванадцяти класах з сорока семи транспортний рівень повністю відокремлено від ігрової логіки. В решті класів ці рівні переплетені, що є безпосередньою причиною ускладненого супроводу та недостатньої гнучкості мережевого модуля. Саме ця архітектурна вада унеможливлює просту заміну транспортної бібліотеки без масштабного переписування коду [9].

Отримані результати аналізу підтвердили доцільність застосування шаблону проєктування «Адаптер» як центрального архітектурного рішення: він дозволяє інкапсулювати деталі реалізації транспортного рівня та забезпечити прозору підміну бібліотеки Lidgren Network без внесення змін до 47 залежних класів ігрової логіки.

На підставі виконаного аналізу розроблено такі компоненти програмного прототипу:

Перша складова прототипу - інтерфейс INetworkAdapter з двома конкретними реалізаціями. Перша з них, LidgrenAdapter, інкапсулює оригінальну бібліотеку Lidgren і застосовується як контрольний варіант при порівняльному тестуванні продуктивності. Друга, GodotENetAdapter, реалізує той самий інтерфейс засобами вбудованого мережевого стеку Godot Engine. Такий підхід забезпечує єдину точку підміни транспортного рівня, при цьому усі сорок сім залежних класів залишаються повністю незмінними.

Друга складова - таблиця відповідностей між програмними інтерфейсами двох бібліотек для мови C#. Вона охоплює сорок два методи, одинадцять подій та вісім типів даних і слугує практичним довідником для розробників, що виконують аналогічну міграцію. Таблиця є самостійним науковим результатом дослідження.

Третьою складовою є скрипт статичного аналізу залежностей, реалізований мовою Python 3.12. Інструмент призначений для автоматичної інвентаризації всіх точок використання бібліотеки Lidgren Network у довільній кодовій базі, написаній мовою C#. За результатами аналізу формується структурований звіт у форматі JSON та інтерактивна карта залежностей у форматі HTML. Повний аналіз кодової бази Barotrauma, яка містить близько ста вісімдесяти тисяч рядків коду, виконується впродовж 4,3 секунди.

Четверта складова - чеклист портингу, що є структурованим переліком із тридцяти восьми контрольних точок, організованих відповідно до п'яти етапів моделі. Кожна точка містить конкретний опис дії та критерій її успішного виконання. Чеклист забезпечує відтворюваність методології при застосуванні прототипу до інших ігрових проєктів на базі Lidgren Network.

Поточний стан реалізації прототипу. На момент підготовки тез завершено три з п'яти етапів розробленої моделі. Виконано аудит залежностей, налаштовано програмне середовище Godot Engine 4 з підтримкою C# та реалізовано інтерфейс INetworkAdapter. Підміну транспортного рівня здійснено у тридцяти одному класі зі сорока семи. Верифікаційне тестування та вимірювання кількісних показників мережевої продуктивності, зокрема затримки відповіді, рівня пакетних втрат і стабільності з'єднання, заплановано на травень 2026 року. Відповідно до технічних характеристик протоколу ENet та задокументованих переваг Godot Engine 4.x, очікується суттєве зниження затримок завдяки вбудованому каналізованому контролю доставки пакетів, а також підвищення стабільності з'єднань при навантаженні понад чотири гравці.

ВИСНОВКИ

У ході виконання дослідження проведено детальний архітектурний аналіз мережевого модуля C#-ігрового застосунку Barotrauma. Встановлено, що сорок сім класів мають пряму залежність від бібліотеки Lidgren Network, яка охоплює сто тридцять чотири унікальних виклики та розподіляється за чотирма функціональними категоріями. Виявлена архітектурна вада - змішана відповідальність транспортного та логічного рівнів - науково обґрунтувала вибір шаблону «Адаптер» як основного підходу до портингу.

Розроблено п'ятиетапну модель портингу та реалізовано програмний прототип адаптерного шару INetworkAdapter, що забезпечує повну ізоляцію транспортного рівня від ігрової логіки. Це принципове архітектурне покращення: прототип дозволяє замінювати транспортну бібліотеку без внесення змін до класів вищих рівнів, що значно спрощує подальший супровід і розширення застосунку.

Розроблені допоміжні компоненти - таблиця відповідностей програмних інтерфейсів, скрипт статичного аналізу залежностей та чеклист портингу - є універсальними артефактами, придатними для застосування при міграції будь-якого C#-ігрового застосунку, що використовує Lidgren Network, на платформу Godot Engine. Перспективним напрямом подальших досліджень є розширення прототипу на інші підсистеми гри, зокрема рендеринг, фізичний рушій та систему модифікацій.

ДЖЕРЕЛА

1. Fowler M. Refactoring: Improving the Design of Existing Code. 2nd ed. Boston : Addison-Wesley, 2018. 448 p.

2. Ethan Lee. FNA: Accuracy-Focused XNA4 Reimplementation. GitHub. 2024. URL: https://github.com/FNA-XNA/FNA (дата звернення: 12.03.2026).

3. Godot Engine Documentation. High-level multiplayer. 2025. URL: https://docs.godotengine.org/en/stable/tutorials/networking/high_level_multiplayer.html (дата звернення: 01.05.2026).

4. Barotrauma Source Repository. GitHub. 2024. URL: https://github.com/Regalis11/Barotrauma (дата звернення: 02.03.2026).

5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

6. Самойленко, А. П., Бондарчук, А. П. (2023). Порівняння комбінованого алгоритму фазингу на основі мутаційного аналізу з алгоритмом фазингу на основі покриття коду. Зв’язок, (2), 54-57.

7. Бондарчук, А. П., Глушак, О. М. (2025). Предиктивне управління оновленнями програмного забезпечення в інтернеті речей.Зв'язок, (5), 13-17.

8. Nystrom R. Game Programming Patterns. Genever Benning, 2014. 345 p. URL: https://gameprogrammingpatterns.com/ (дата звернення: 07.03.2026).

9. Shantyr, A., Zinchenko, O., Storchak, K., Bondarchuk, A., & Pepa, Y. (2025). Prediction of quality software quality indicators with applied modifications of integrated gradiates methods. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 15(2), 139–146. https://doi.org/10.35784/iapgos.6892

10. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km