DOI: https://doi.org/10.36910/6775-2524-0560-2026-64-13
УДК 004.5:004.8
Дмитрук Олександр Ярославович, студент PhD
https://orcid.org/0009-0001-4586-0333
Луцький національний технічний університет, м. Луцьк, Україна

ПРОБЛЕМАТИКА ЗАСТОСУВАННЯ ІНТЕЛЕКТУАЛЬНИХ ПРОГРАМНИХ СИСТЕМ ПРИ СТВОРЕННІ КОНТЕКСТНО-АДАПТИВНИХ ІНТЕРФЕЙСІВ

Дмитрук О. Я. Проблематика застосування інтелектуальних програмних систем при створенні контекстно-адаптивних інтерфейсів. В даній статті проаналізовано основні принципи створення контекстно-адаптивних інтерфейсів, який передбачає використання адаптивних сіток. Також розглянуто переваги із застосування сучасних напрямків для розвитку адаптивних інтерфейсів на основі штучного інтелекту. Окреслено окремі напрямки, які послугували основою для встановлення проблематики застосування інтелектуальних програмних систем при створенні інтерфейсів. Наведено схему життєвого циклу створення контекстно-адаптивного інтерфейсу. Зазначені зміни, яких зазнає життєвий цикл створення інтерфейсів після застосування інтелектуальних програмних систем на усіх етапах. Наведена схема нового циклу. Основний фокус роботи наведено на виділення проблем, з якими зустрінуться користувачі та розробники після інтегрування інтелектуальних систем у створення контекстно-адаптивних інтерфейсів, з використанням певного контексту, як приклад. Створено таблицю, до нової схеми життєвого циклу, у якій наведено ряд недоліків, з якими потрібно буде мати справу після застосування інтелектуальних програмних систем. Висунуто прогнози та можливі варіанти використання інтелектуальних систем при створенні інтерфейсів, для кращого сприйняття користувачами змін у використанні програм з контекстно-адаптивними можливостями інтерфейсів. Визначено оптимальних підхід для створення зручного продукту, який дозволить користувачам та розробникам мати більший вплив на вдосконалення інтерфейсу, а також дозволить знизити ризик та значимість помилок при застосуванні інтелектуальних програмних систем під час створення контекстно-адаптивних інтерфейсів.
Ключові слова: Інтелектуальні програмні системи, контекстно-адаптивний інтерфейс, користувач, контекст, життєвий цикл

Dmytruk O. Issues in the application of intelligent software systems in the creation of context-adaptive interfaces. This article analyzes the basic principles of creating context-adaptive interfaces, which involve the use of adaptive grids. It also examines the advantages of applying modern approaches to the development of adaptive interfaces based on artificial intelligence. Specific areas that served as the basis for identifying the challenges of using intelligent software systems in interface design are outlined. A diagram of the life cycle for creating a context-adaptive interface is presented. The changes that the interface development life cycle undergoes after the application of intelligent software systems at all stages are noted. A diagram of the new cycle is presented. The main focus of this work is on identifying the problems that users and developers will encounter after integrating intelligent systems into the creation of context-adaptive interfaces, using a specific context as an example. A table has been created to accompany the new life cycle diagram, listing a number of shortcomings that will need to be addressed following the implementation of intelligent software systems. Predictions and possible options for using intelligent systems in interface design have been proposed to help users better adapt to changes in the use of applications with context-adaptive interface capabilities. An optimal approach has been identified for creating a user-friendly product that will allow users and developers to have a greater influence on interface improvement, as well as reduce the risk and severity of errors when applying intelligent software systems during the creation of context-adaptive interfaces.
Keywords: Intelligent software systems, context-adaptive interface, user, context, life cycle

Постановка наукової проблеми. Сьогодні розвиток інтелектуальних програмних систем це рушійна сила, яка набирає обертів і охоплює майже усі напрямки технологічного розвитку. Постійно озвучуються переваги таких нововведень та надто мало приділяється уваги проблемам, які йдуть пліч-о-пліч із позитивними змінами. Інтерфейс це перша лінія у будь якій системі, це складова, яка висвітлює усі переваги, проте, якщо придивитись, там можна помітити недоліки. Контекстно-адаптивний інтерфейс, зосереджений на максимальній зручності, але де ж проходить ця тонка межа між зручним та нав'язливим програмним середовищем. Як забезпечити створення та дотримання правильних вимог, адаптованих, які приведуть до створення якісного продукту. Чи можливе усунення усіх проблем, яких витрат ресурсів вистачить, щоб привести продукт до позначки «якісний». Чим ретельніше ми оглядаємо застосування інтелектуальних програмних систем під час створення контекстно-адаптивних інтерфейсів, тим більше потрібно аналізувати даних, щоб обійти усі проблеми.

Аналіз досліджень. Розробники та науковці в своїх роботах висвітлюють різні методики вирішення проблем під час створення контекстно-адаптивних інтерфейсів із використанням інтелектуальних програмних систем. Створення гнучкої сітки є основою адаптивного дизайну[1], це твердження зустрічається в багатьох дослідженнях і справді має чітко аргументоване підґрунтя та використовується як основа для контекстно-адаптивних інтерфейсів. Воно передбачає сітку, як основу всього інтерфейсу, яка дозволяє усі зібрані дані проектувати відповідно до заданих типів

параметрів, кожен з яких відноситься до певної групи. Така методика виокремлює декілька типових інтерфейсів, які поєднують у собі різні параметри.

Перший етап роботи інтелектуальних програмних систем – це збір різного типу контексту: місце, час, пристрій, роль та досвід користувача, фізичне оточення і т. д. Кожен з них має у собі різних набір підтипів, а самих типів можна виділити настільки багато, що деякі попросту ігноруються, щоб не навантажувати систему. Після збору інформації необхідне використання інтелектуальних програмних систем для аналізу даних, які поступили, після чого ми бачимо зміни та отримуємо контекстно-адаптивний інтерфейс.

Такий підхід виглядає логічним і зрозумілим, але під час його роботи постає безліч питань. Чи достатньо інформації зібрано, чи здатні закладені сітки висвітлити усю необхідну структуру. Сьогодні ми проєктуємо систему під одну кількість приладів, а вже завтра виходять в продаж пристрої з іншими характеристиками. Чи вірно проаналізована інформація, якщо її надто багато наскільки сильно така аналітика навантажуватиме пристрій, а якщо будуть перебої у роботі інтернету, як запевнитись, що усі дані вірні.

При аналізі досліджень[1,2], можна зробити висновок, що при створенні контектно-адаптивних інтерфейсів все зводиться до розробки системи під окремі типи користувачів чи пристроїв. Практично неможливо розробити інтерфейс, який міг би адаптуватись під усіх. Така система вимагала б великої кількості вхідних даних та значного часу на обробку, а також постійного оновлення заготовлених схем роботи.

Підхід із сіткою якісний, проте він має межу, з плином часу такі системи будуть ставати просто неактуальними. Все виглядає так наче, єдиний вихід – це постійне створення нових продуктів, але джерело [4], висвітлює нові можливості. Проаналізувавши зібрані там оновлені напрямки для адаптивних інтерфейсів на основі штучного інтелекту, ми бачимо, що з'являються альтернативи. Основні положення, які варто відмітити, які здатні принести зміни для попередніх систем:

- контекстно-залежне налаштування інтерфейсу користувача, тобто адаптація інтерфейсу відбувається не постійно і не в зазначені часові рамки, а саме тоді коли відбулись такі зміни, які потребують реакції інтерфейсу;
- відстежування поведінки користувача у реальному часі. Перед нами постає підхід, який збирає постійно інформацію про користувача, а не відносить його до заданої групи. Ми бачимо, які дії виконує користувач і залишається задоволеним, а які ігнорує, помічаємо, які операції забирають більше часу, а які менше. Система вчиться під індивідуальні потреби і змінюються під ваші потреби та вподобання;
- адаптивне зменшення складності, дозволяє використання простих чи розвинутих функцій, залежно від ваших потреб;
- пропозиції щодо передбачуваних функцій. В цьому положенні ідеться не проте, що програма буде відкриватись у звичний час чи щось подібне, а саме про те, що відповідно до дій користувача програма буде інтуїтивно підкреслювати необхідні функції;
- динамічне форматування контенту. Коли програма не працює лише із зазначеними параметрами фону, шрифту чи кольорів, а має здатність змінювати увесь зміст відповідно до Ваших потреб та непотрібно надіятись, що розробник вибере та реалізує, щось підходяще під Вас;
- адаптивні методи введення. Коли користувач, може отримувати потрібні функції незалежно від того чи він користується клавіатурою, сенсором чи навіть голосом;
- моделювання стану користувача. Програма, яка розуміє з якою силою чи швидкістю виконуються операції, чи користувач поспішає, чи тривожиться чи ще не встиг прокинутись. Усе, щоб покращити його взаємодію після адаптації приладу.

Використання таких положень теж орієнтоване більшою мірою на користувача, проте дозволяє виокремлювати його, з його потребами та вподобаннями, а не відносити до групи схожих користувачів.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. Провідний метод створення контекстно-адаптивних інтерфейсів сьогодні стандартно передбачає постійний цикл, у якому постійно повторюються етапи, поки не буде досягнуто результату, який задовільнить конкретну групу користувачів. Життєвий цикл створення контектно-адаптивного інтерфейсу наведений на рисунку 1.

Така схема передбачає спочатку збір інформації, про місце, час, пристрій, фізичні та інші особливості. Після цього потрібно структурувати усі вхідні дані, які з них важливіші, корисніші та несуть більший вплив. Наступний етап – проєктування, потрібно поєднати усі зібрані дані і створити майбутні сітки, за якими буду працювати система. Далі іде реалізація, ми закладаємо усі необхідні дані до спроєктованих сіток. Тестування спочатку на помилки тоді на те чи усіх вимог дотримано. Після усіх перевірок, ми або доопрацьовуємо якісь елементи, або в разі якихось суттєвих відхилень переходимо знову до початку роботи. Цикл закінчиться лише в тому варіанті коли програму просто перестануть використовувати, а до тієї пори, нам постійно доведеться щось змінювати чи удосконалювати.

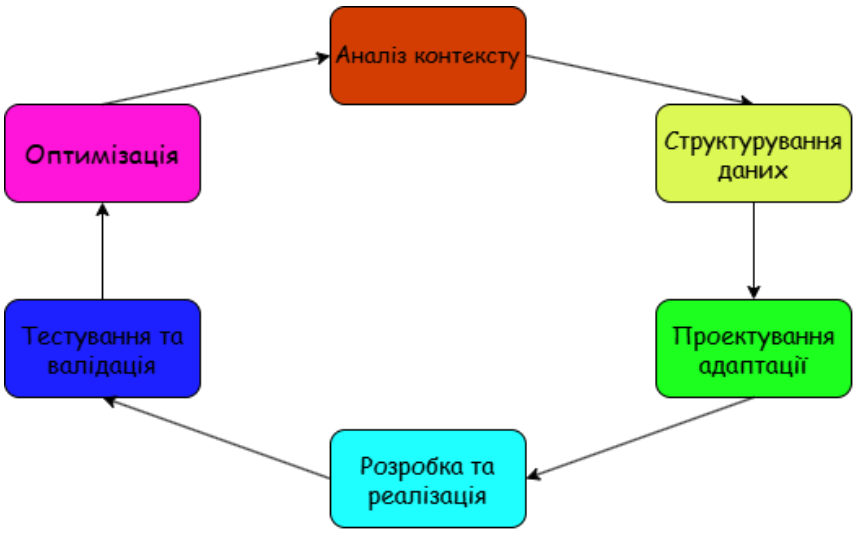

Рис. 1. Життєвий цикл створення контектно-адаптивного інтерфейсу

Сучасні тенденції підштовхують нас до використання інтелектуальних програмних систем (ІПС) в кожному етапі життєвого циклу, для покращення та оптимізації роботи системи, як результат ми бачимо нову схему, наведену на рисунку 2.

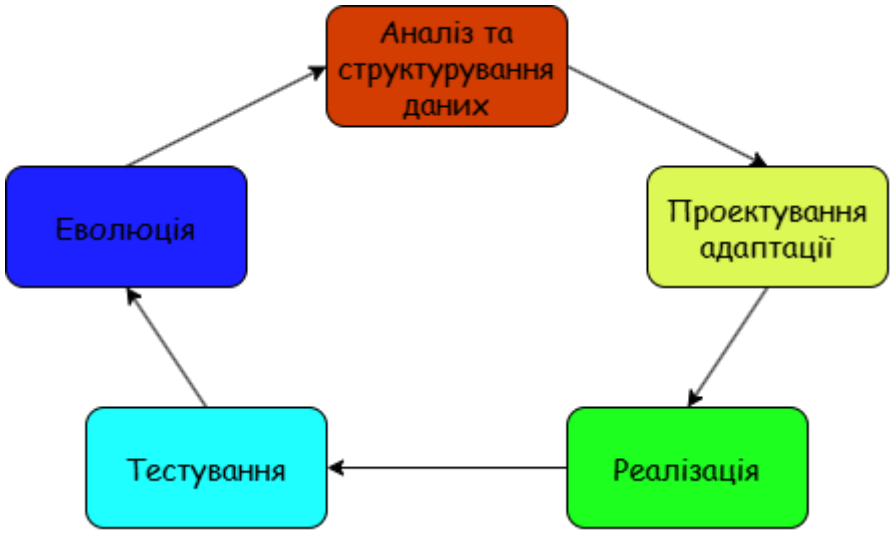

Рис. 1. Життєвий цикл створення контектно-адаптивного інтерфейсу з використанням ІПС

З використанням інтелектуальних програмних систем, цикл стає на перший погляд коротшим. Нам більше не потрібно виділяти окремо час на аналіз та структурування, оскільки такі системи здатні обробляти такі масиви в десятки разів швидше. Замість просто збору даних ми бачимо, як система почне прогнозувати дії користувача. Вже на цьому етапі ми бачимо використання одного з положень для адаптації інтерфейсів. Приклад такої роботи, це коли лише за аналізом часу

18:00 та за тим що Ви вийшли з місця де були весь день, тобто, місця роботи, програма автоматично відкриє навігатор із шляхом додому.

Наступний етап циклу з використанням ІПС ми бачимо теж саме проєктування адаптації, проте операції закладені на цьому етапі кардинально змінюються. Нам більше непотрібно закладати в систему декілька макетів з сіткою, ми досягаємо індивідуального підходу до користувача. Зникають так звані групи, програма стає масштабнішою. Інтерфейс стає Вашим місцем творчості, де Ви самі заправляєте необхідними Вам функціями.

Етап реалізації у звичайній системі, це просто набір однотипних рішень де розробник проєктує усі макети, в варіанті з використанням ІПС, ми бачимо об'єднання роботи етапу реалізації з етапом аналізу. Система стає стабільнішою, оскільки, через елемент навчання, зміни в системі відбуваються лише в потрібний момент, а не постійно.

Тестування стає набагато швидшим етапом, адже ІПС здатні оцінювати виконану роботу відразу після застосування, аналізувати ці дані, та вчитись по них. Одна ІПС здатна розіграти величезну кількість сценаріїв за короткі терміни, за звичайних умов на це буде необхідний величезний людський ресурс, а не менші витрати часу.

На крайньому місці життєвого циклу, ми бачимо етап Еволюції, адже програма навчається після кожної зміни і зміни помітні вже під час наступного використання.

Усі ці зміни в роботі циклу виглядають, як суцільні покращення, які просто необхідно використовувати, але чи справді це так. Адже там ховаються і недоліки. Розглянемо ключові недоліки, з якими ми неодмінно зустрінемось при роботі з життєвим циклом що використовує інтелектуальні програмні системи, які наведені в таблиці 1.

Таблиця 1. Недоліки життєвого циклу з використанням ІПС

Етапи життєвого циклу	**Недоліки**
Аналіз та структурування даних	Приватність користувача
	Навантаження на пристрій
Проєктування адаптації	Відсутність контролю користувача
	Постійні зміни
Реалізація	Недостовірність результатів
	Невизначена стартова точка
Тестування	Відсутність впливу на баги
	Неточні тестування
Еволюція	Помилковий фідбек
	Анулювання роботи ІС

Згідно до таблиці 1, можна зрозуміти, що з використанням ІПС при створенні контекстно-адаптивних інтерфейсів, на кожному етапі життєвого циклу траплятимуться проблеми. На етапі аналізу та структурування даних ми зустрінемось із проблемою приватності – для якісної роботи ІС необхідним є постійний доступ до даних. Мова не про звичайні дані типу години та погоди, для правильної роботи нам необхідний величезний діапазон, який може включати: місцезнаходження, доступ до мікрофону чи камери, аналіз історії дій. Відсутність усієї картини буде вести до неправильного аналізу та некоректного виконання інших кроків циклу. Оскільки даних багато тут випливатиме нова проблема скільки енергії споживатиме пристрій під час цього збору та аналізу. Високий обсяг енерговитрат дорівнює високу навантаженню на прилад, а отже така система не зможе використовуватись усіма верствами населення.

Етап проєктування також ховає проблеми, як наприклад відсутність контролю користувачем. Коли система сама вирішує, які функції важливіші, людина не здатна відчувати контроль над власним пристроєм. Звідси випливає також інша проблема – постійні зміни інтерфейсу. Якщо відбуваються постійні зміни, користувач мусить постійно адаптуватись чи навчатись, що буде схоже на постійний початок роботи з нуля. Як тоді правильно аналізувати дані, якщо після зміни, користувач починає затрачати більше часу на якісь дії, чи не буде це створювати хибну картину про користувача.

Плавно переходимо до проблем недостовірності результатів на етапі реалізації. Якщо ІПС виконують, якісь розрахунки та аналізування, як дізнатись що вони виконані вірно. Буде відсутня можливість відслідковувати логіку тих чи інших змін інтерфейсу. Мало того, що ми не знаємо, на

які пункти зважає система при реалізації, так більше того, з чого вона мусить розпочати. Відсутня стартова точка – для перших логічних реалізацій, системі потрібен час на адаптацію під користувача, це може зайняти години, дні, тижні. А такі затримки приведуть до невдоволення, роздратування та відмови від подібних нововведень.

Етап тестування не видається таким же проблематичним, як інші етапи, але чи справді це так. Як система має реагувати на людський фактор. Якщо телефон спрацював сам у кишені, чи коли було підвисання виконана додаткова дія, чи людина не довго думаючи скачувала, якусь дію інтерфейсу, хоча це мало бути правильним рішенням. Тестування відбуватимуться швидко, через що навіть незначна помилка буде враховуватись і вважатиметься не менш значимим пунктом для аналізу. Також майже неможливий зовнішній вплив в роботу системи, якщо в користувача виникне баг, розробнику, майже неможливо буде відслідкувати, в якому місці та за яких умов це сталося.

Що стосується етапу еволюції, то питання помилкового фідбеку ми вже зачепили, але також загрозу становитиме анулювання роботи ІПС. Мова не про ті випадки коли система буде сприймати помилкові дані за правильні, а про те, що поведінка користувачі може змінюватись. Наприклад після покупки нового телефону старий перейде до когось з родичів, або із зміною роботи в людини зміниться і темп життя і місце і транспорт. В таких ситуаціях, можливо варто закладати, якісь додаткові функції, чи будуть такі функції безпечними чи виконуватиме ІС наступний розвиток правильно.

Отже, після виокремлення усіх проблем, на кожному з етапів життєвого циклу, ми бачимо, що інтегрування ІПС несе велику кількість недоліків, які ускладнюють процес злиття звичної моделі життєвого циклу створення контекстно-адаптивних інтерфейсів із роботами штучного інтелекту. Кожен етап передбачає зміни, які мають в перспективі лише позитивний вплив, але щоб його досягти треба пройти достатньо тернистий шлях. Етап аналізу контексту в різних пристроях, платформах та програмах залучає інтелектуальні програмні системи і показує хороші результати, після вирішення більшою мірою правових питань, він поки недоліки позаду. Що стосується інших етапів, можна зробити висновок, що покладатись лише на штучний інтелект не вийде.

Проєктування адаптація досі залишається проблемою, над якою буде найбільше проблем. Оскільки зараз ми бачимо лише дві паралельні сторони, які практично не поєднуються. Відразу прогнозовані групи та штучний інтелект, який буде головним. Знайти цю тонку межу, щоб об'єднати ці сторони та досягти наближеної гармонії буде достатньо важко.

Усі інші етапи містять страшні проблеми, проте кожна з них підлягає вирішенню, основний критерій для цих етапів, це – час. Час для розігріву штучного інтелекту, терміни для його адаптації до нового користувача чи після зміни користувача чи контексту. Тестування, які будуть забирати багато часу, якщо розробник буде перевіряти роботу виконану штучним інтелектом.

Висновки та перспективи подальшого дослідження. З проаналізованих статей та з виділених проблем, можна зробити висновок, що швидшого переходу від класичної моделі контекстно-адаптивних інтерфейсів, до тих які використовують інтелектуальних програмних системи не буде. З рядом переваг ми розуміємо, що це справді наступний крок, до якого варто йти, але найкращим рішенням буде – поступове розширення. Це допоможе пом'якшити проблеми по типу – невизначеного старту.

Найкращим виходом із цих проблем будуть такі кроки, як, для прикладу, щоб робота інтелектуальних програмних систем мала розпочинатись з опитувань. Користувач зможе вибирати чи варто системі відразу починати аналіз роботи чи можливо користувач почне використання нового пристрою без втручання ІПС та спершу звикне до його стандартних функцій. Так для прикладу, контекст може накопичуватись та аналізуватись, проте варіанти змін стануть доступними уже після досягнення певного проміжку. Динамічне форматування контексту, яке буде переходити в автоматизоване – користувач буде розвиватись і адаптуватись поступово, паралельно із прогнозованими та запропонованими результатами штучного інтелекту. Ще один варіант для уникнення проблем – це те, що користувачеві постійно будуть доступні функції по зміні в роботі інтелектуальній програмній системі і в разі різких змін доступне буде скидання чи призупинення використання контекстно-адаптивного інтерфейсу. Це допоможе для кращого фідбеку розробникам, а також ІПС буде переосмислювати попередні свої кроки, а не починати роботу з нуля.

Поступовий рух удосконалень дозволить чіткіше відслідковувати межу між зручним та нав'язливим програмним середовищем. А чіткіша межа дасть зрозуміти, які аспекти потребують

більшого доопрацювання. Користувачі та розробники будуть залишатися головними і зможуть корегувати роботу інтелектуальних програмних систем.

Розвиток контекстно-адаптивних інтерфейсів з використанням інтелектуальних програмних систем це перспектива, яка дозволить краще зрозуміти, як використовувати штучний інтелект, а також дозволить покращити загальний прогрес суспільства із використанням технологій у всіх напрямках та сферах життєдіяльності.

Зважаючи на усі тенденції розвитку штучного інтелекту та його інтеграції в усіх сферах життя людини, необхідно в першу чергу звернути увагу на проєктування адаптації. Зробити систему здатну підлаштовуватись під індивідуальні потреби, а не зводити користувачів до конкретних груп. Навчити програму виділяти людину, як особистість та в той же час не обмежувати її в бажаннях та можливостях, щоб кожен мав шанс відчуватись усі переваги технологічного процесу, незалежно від технічних, когнітивних, фізичних чи будь яких інших можливостей.

Список бібліографічного опису

1. Створення адаптивного дизайну: принципи та інструменти - IT рейтинг UA. *IT рейтинг України*. URL: https://it-rating.ua/stvorennya-adaptivnogo-dizaynu-printsipi-ta-instrumenti (дата звернення: 14.08.2026).
2. Adaptive Interfaces: Personalized Human-AI Collaboration Through Contextual AR Environments. URL: https://www.researchgate.net/publication/397273231_Adaptive_Interfaces_Personalized_Human-AI_Collaboration_Through_Contextual_AR_Environments (дата звернення: 14.08.2026).
3. CONTEXT-AWARE AI SYSTEMS FOR ADAPTIVE AUGMENTED REALITY EXPERIENCES. URL: https://www.researchgate.net/publication/396444075_CONTEXT-AWARE_AI_SYSTEMS_FOR_ADAPTIVE_AUGMENTED_REALITY_EXPERIENCES (дата звернення: 14.08.2026).
4. AI Adaptive User Interfaces: 20 Updated Directions (2026) - Yenra. *Yenra Research*. URL: https://yenra.com/ai20/adaptive-user-interfaces/ (дата звернення: 15.08.2026).

References

1. Creating Adaptive Design: Principles and Tools – IT Rating UA. IT Rating Ukraine. URL: https://it-rating.ua/stvorennya-adaptivnogo-dizaynu-printsipi-ta-instrumenti (date of access: 14.08.2026).
2. Adaptive Interfaces: Personalised Human-AI Collaboration Through Contextual AR Environments. URL: https://www.researchgate.net/publication/397273231_Adaptive_Interfaces_Personalized_Human-AI_Collaboration_Through_Contextual_AR_Environments (date of access: 14.08.2026).
3. CONTEXT-AWARE AI SYSTEMS FOR ADAPTIVE AUGMENTED REALITY EXPERIENCES. URL: https://www.researchgate.net/publication/396444075_CONTEXT-AWARE_AI_SYSTEMS_FOR_ADAPTIVE_AUGMENTED_REALITY_EXPERIENCES (date of access: 14.08.2026).
4. AI Adaptive User Interfaces: 20 Updated Directions (2026) – Yenra. Yenra Research. URL: https://yenra.com/ai20/adaptive-user-interfaces/ (date of access: 15.08.2026).

Історія статті:
Отримано: 24.05.2026 Доопрацьовано: 06.08.2026 Прийнято до друку: 23.09.2026 Опубліковано: 29.09.2026