Issue 9, 2021

https://doi.org/10.23939/sisn2021.09.061

УДК 004.4

АДАПТИВНА ПРОГРАМНА СИСТЕМА НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ ДЛЯ ЛЮДЕЙ З КОГНІТИВНИМИ ПОРУШЕННЯМИ

Дмитро Федасюк[1], Ілля Луцик[2]

Національний університет "Львівська політехніка"
[1] dmytro.v.fedasyuk@lpnu.ua ORCID 0000-0003-3552-7454
[2] illia.lutsyk.mnpz.2019@lpnu.ua ORCID 0000-0001-7117-5332

Запропоновано метод створення адаптивної програмної системи для допомоги людям з когнітивними порушеннями, оснований на використанні онтологічної моделі предметної області.

Проаналізовано специфіку створення програмних засобів для допомоги людям з когнітивними порушеннями. Розкрито особливості використання онтологічного підходу для формування адаптивної функціональності та графічного інтерфейсу й проаналізовано їхні переваги порівняно з класичними методами. Встановлено, що за використання вказаного методу немає потреби у разі зміни бізнес-логіки здійснювати перекомпіляцію та повне розгортання програмної системи. Спроєктовано онтологічну модель предметної області, що дасть змогу налаштовувати систему під потреби конкретного користувача. Запропоновано архітектуру програмної системи на основі онтологічної моделі предметної області, яка враховує можливість персоналізації компонент системи та інтерфейсу користувача без необхідності повторного розгортання системи. Розкрито процес адаптації мобільного застосунку на основі даних про порушення здоров'я користувача із використанням онтологічної моделі предметної області.

Результатом дослідження є розроблення програмної системи, що реалізовує запропонований процес адаптації та дає змогу модифікувати мобільний застосунок під потреби конкретного користувача, використовуючи онтологічну модель предметної області.

Ключові слова: онтологічний підхід; онтологічна модель; адаптивна програмна система; мобільний додаток; когнітивні порушення.

Вступ

У сучасному світі зростає необхідність створення адаптивних програмних систем, які дають змогу забезпечити гнучке налаштування залежно від потреб користувача. Зокрема, особливу увагу варто приділити програмному забезпеченню для людей із когнітивними порушеннями чи обмеженими можливостями. Часто внаслідок розладів здоров'я чи вікових змін такі особи не можуть брати повноцінну участь у всіх сферах життєдіяльності [1]. Це спричинено тим, що для них характерні проблеми із пам'яттю, увагою, орієнтацією у просторі, підвищена втомлюваність, що утруднює виконання поставлених завдань.

Через свої особливості люди з когнітивними порушеннями не можуть ефективно використовувати наявні програмні рішення. Більшість мобільних додатків не пропонують розширюваність та персоналізацію відповідно до розладів здоров'я людини. Тому актуальне створення алгоритму адаптації для формування персоналізованого функціонала мобільного застосунку, а також розроблення на його основі

спеціалізованої адаптивної програмної системи, що дала б змогу організовувати власний час, запам'ятовувати та планувати повсякденні дії із урахуванням порушення здоров'я користувача.

Аналіз останніх досліджень та публікацій

На вирішення вищезазначеної проблеми спрямована низка досліджень українських та іноземних вчених. Зокрема, принципи застосування мобільних технологій для людей з когнітивними порушеннями досліджували Х. Гомес, К. Аламан, Г. Монторо та ін. [2]. У цьому дослідженні автори подали програмну систему, що допомагає таким особам адаптуватись до сучасних умов праці й підвищити їхню продуктивність. Запропонована система містить дві компоненти. Перша компонента – AssisT-Task використовує QR-коди, відсканувавши які, люди отримують інтерактивні посібники, що допомагають їм виконувати завдання. Друга компонента AssisT-In надає допомогу в переміщенні на робочому місці користувачам з когнітивними вадами за допомогою вказівок, отриманих під час сканування відповідних QR-кодів. Ця система ефективна для людей з когнітивними порушеннями, оскільки інтерактивність вказівок та кількість кроків визначаються залежно від потреб користувача. Проте це рішення не забезпечує повної адаптивності, оскільки динамічно визначається лише інформативність вказівок та кроків під час виконання завдань. На нашу думку, доцільно також забезпечити динамічну модифікацію як інтерфейсу, так і функціональності системи, оскільки це розширить можливості та допоможе покращити опіку осіб з когнітивними порушеннями.

Класичні системи дають змогу вирішувати фіксоване коло проблем, які наявні у певної категорії осіб. Проте, якщо стан здоров'я людини погіршиться або потрібно буде ще доповнити функціонал через інші причини, то програмні засоби не здатні динамічно реагувати на зміни. Вирішенням цієї проблеми є використання адаптивних програмних систем, які забезпечують можливість модифікації під потреби конкретних користувачів.

У такому напрямі продовжили роботу Хав'єр Гомес та Герман Монторро. В своєму дослідженні науковці спроєктували адаптивний програмний застосунок, який допомагає створювати або прокладати маршрути для людей із когнітивними порушеннями. Система розділяє маршрут на атомарні операції/інструкції, а на роздоріжжі або перехрестях користувачу демонструють зображення для орієнтації у просторі [3]. Ця система забезпечує вищу ефективність порівняно зі стандартними програмами, адже, за результатами дослідження, кількість осіб, що дісталися до місця призначення, збільшилася вдвічі. Проте, незважаючи на позитивні результати, подане рішення спрямоване лише на вирішення однієї проблеми, а саме – допомогти орієнтуватися в просторі людям із когнітивними порушеннями. Також доцільно зважати на інші порушення здоров'я у таких осіб, зокрема, проблеми із зором чи слухом. У таких випадках необхідно враховувати можливість зміни й інтерфейсу користувача, для забезпечення зручного користування програмним засобом.

Отже, під час розроблення адаптивних програмних засобів необхідно враховувати можливість модифікації функціональності системи та графічного інтерфейсу під потреби користувача. Наприклад, для людей із порушенням зору доцільно збільшити текст та надати голосовий супровід, а для людей з пониженим слухом – додати текстовий опис мультимедійних елементів.

Особливості розроблення адаптивних інтерфейсів мобільних додатків для людей з обмеженими можливостями описано у дослідженнях А. Поцілуйка [4] та А. Дворянкіна [5]. В роботі автори зазначили, що основна проблема непристосованості інтерфейсу під потреби користувачів полягає у тому, що розробники не знають вимог і рекомендацій щодо створення інтерфейсів для людей із різними видами порушень. Внаслідок цього багато часу витрачається на пошук вимог до інтерфейсу для того чи іншого порушення у користувача.

Отже, адаптація програмної системи до нових вимог вимагає врахування знань фахівців, які є експертами у визначеній предметній області, та, як правило, не є фахівцями з програмування. Це, своєю чергою, призводить до вагомих ризиків у розробленні програмних продуктів, спричинених помилками у формулюванні вимог до системи під час її концептуалізації [6].

Вирішити проблему адаптації програмної системи до змін середовища можна із застосуванням онтологічного моделювання, принципи використання якого розкрито у роботі Є. В. Бурова [7]. На відміну від класичних підходів моделювання, у онтологічному будують формальну модель предметної області, яку можна повторно використати в інших програмних рішеннях. Кінцевим критерієм успішного проєктування онтології є ефективність та результативність вирішення проблем реального життя.

Використання онтологічного підходу для формування адаптивного графічного інтерфейсу для користувачів з обмеженими можливостями розкрито в роботі [8]. Автори запропонували двоетапний метод адаптації інтерфейсу для користувачів з обмеженими можливостями на основі семантичних правил онтології з подальшим використанням шаблонів інтерфейсу. Онтологічні правила вибору дають змогу визначити оптимальні характеристики графічного інтерфейсу системи для користувача з різними видами порушення здоров'я. Проте подана система не враховує можливості динамічної зміни функціональності під час її використання.

Формулювання цілі статті

Здійснений аналіз наукових досліджень щодо створення систем для допомоги людям з когнітивними порушеннями підтверджує актуальність розроблення адаптивного програмного забезпечення для таких осіб, що допоможе у вирішенні повсякденних завдань, дасть змогу покращити якість життя та вдосконалити догляд за хворими. Така система ґрунтуватиметься на онтологічній моделі предметної області, що дасть змогу прийняти рішення про необхідність динамічної зміни поведінки чи вигляду системи залежно від порушень здоров'я конкретного користувача. Крім того, поєднання підходів адаптації графічного інтерфейсу та функціонала, із використанням цього підходу, дасть змогу підвищити ступінь налаштованості програмного забезпечення для конкретного користувача.

Метою дослідження є створення нового підходу до динамічного формування та адаптації функціональності та графічного інтерфейсу за допомогою онтологічної моделі предметної області та подальша реалізація на його основі програмної системи для допомоги людям з когнітивними порушеннями.

Онтологічний підхід до розроблення програмних систем

Онтологічний підхід дає змогу в розгорнутому вигляді подавати семантичну модель предметної області, яка досліджується, у вигляді ієрархії концептів і множини відношень, які поєднують встановлені концепти і терміни [9, 10].

Онтологія є кортежем, що складається з множин: концептів, понять, атрибутів і відношень. Головною метою онтології є класифікація індивідів (екземплярів) сутностей, що є фактичними елементами нижнього рівня. Як поняття вибирають абстрактні набори або колекції об'єктів. Для зберігання специфічної інформації про об'єкт використовують атрибути. Відношення дають змогу визначити залежності між об'єктами онтології. У результаті повного опису об'єктів і їх властивостей предметна область буде представлена як складна ієрархічна база знань, над якою можна буде здійснювати "інтелектуальні" операції, такі як семантичний пошук і визначення цілісності та достовірності даних [11, 12].

Створення онтології основане на аналізі предметної області, що загалом визначається послідовністю таких етапів:

1. Визначення цілей і сфери застосування програми.
2. Проєктування загальної концептуальної структури предметної області. Цей етап передбачає визначення основних понять предметної області, їх властивостей, зв'язків між ними, а також створення абстрактних класів для підтримки наслідування властивостей і зв'язків, посилання чи введення допоміжних онтологій, віднесення екземплярів за концептами. Цей етап поки що практично неможливо автоматизувати, всі дії повинна здійснювати людина.
3. Збереження отриманої онтології як базової для подальшого розширення.
4. Розширення онтології предметної області із додаванням концептів, зв'язків та об'єктів до рівня деталізації, необхідного для забезпечення вимог, які ставлять перед онтологією, щоб використати її для розв'язування задач предметної області.
5. Верифікація створеної онтології та розгортання її в середовищі, де вона буде використовуватися.

Вибір методології проєктування програмних систем на основі онтологічного моделювання зумовлений такими чинниками. По-перше, створення програмного продукту на основі фіксованого набору вимог за допомогою традиційних методів проєктування програмних систем призводить до створення систем, у яких не передбачена реакція на зміну вимог та зовнішніх чинників [4, 13]. По-друге, врахування оновлених вимог, як правило, призводить до необхідності створення нової версії програмного продукту, що передбачає виконання тривалих та ресурсоємних етапів аналізу, дизайну, кодування, тестування та впровадження нової версії.

Недоліки використовуваних модельно-орієнтованих підходів до побудови програмних систем значною мірою пояснюються складністю як предметної області, так і відповідних моделей. Ці недоліки можна усунути, реалізуючи програмну систему як набір простих інтерпретованих моделей, що взаємодіють між собою. Створення програмних систем на основі онтологічного моделювання порівняно з традиційними методами має такі переваги:

- зникає необхідність у разі зміни бізнес-логіки здійснювати перекомпіляцію та повне розгортання програмної системи – необхідно модифікувати тільки ті моделі, що реалізують цю логіку;
- використання онтологічних моделей дає змогу накопичувати та повторно використовувати знання про методи та варіанти адаптації системи до зміни вимог.

Важливою складовою програмної системи для допомоги людям похилого віку з когнітивними порушеннями є онтологічна модель предметної області, яка враховуватиме можливість формування функціонала та графічного інтерфейсу користувача. Подібно до підходу, висвітленого в роботі [14], адаптація програмної системи ґрунтується на використанні сформованих семантичних правил онтологічної моделі предметної області. Порівняно із традиційним підходом, таке рішення дасть змогу вдосконалити взаємодію користувача із системою, покращити виконання щоденних завдань для людей із когнітивними порушеннями та модифікувати програмне забезпечення з урахуванням особливостей конкретної особи.

Інтерфейс користувача в системах, основаних на знаннях, повинен допомагати користувачеві будь-якого рівня підготовленості. Такий інтерфейс має бути інтелектуальний, тобто повинен реалізовувати функції допомоги та взаємодії, що передбачатимуть автоматичну побудову програми (зміну графічного інтерфейсу, приєднання нового модуля), з подальшими вказівками щодо використання нових функцій програмного забезпечення.

Роботу системи та інтерфейс користувача необхідно спроєктувати так, щоб забезпечити можливість взаємодії відповідно до кваліфікації чи завдань користувача [15]. В такому випадку доцільно визначити такі три основні режими роботи:

- Автоматичний режим – передбачає автоматичне визначення необхідних налаштувань та динамічну модифікацію програмного засобу на основі онтологічної моделі, залежно від особливостей когнітивних порушень користувача.
- Меню доцільних варіантів вибору – в цьому режимі для користувача буде динамічно визначено перелік варіантів, найчастіше використовуваних для цього типу задач.
- Повне меню – надає всі можливі варіанти вибору рішень на конкретному етапі для поставлених завдань. Розраховано на опікуна або особу з легкими когнітивними порушеннями (віковими змінами); для програмного засобу визначаються оптимальні налаштування для коректної роботи з системою.

Отже, використання онтологічної моделі дасть змогу забезпечити можливість відокремлення логіки функціонування програмної системи від механізму її реалізації.

Онтологічна модель предметної області

Формалізовано онтологічну модель можна подати як сукупність множин концептів, атрибутів, відношень та інших додаткових елементів предметної області. Отже, онтологічну модель адаптивної системи для допомоги людям з когнітивними порушеннями можна визначити як кортеж, що складається з шести елементів:

$$O_{sys} = <C_{sys}, R_{sys}, F_{sys}, Rl_{sys}, Ind_{sys}, Prop_{sys}>, \quad (1)$$

де $C_{sys} = \{c_i \mid i = 1,...,I\}$ – множина сутностей предметної області системи допомоги літнім людям з когнітивними порушеннями; $R_{sys} = \{r_j \mid j = 1,...,I\}$ – множина відношень між сутностями предметної області; $R_{sys} = \{$ has, is-a,has_impact_on, uses, needModule, needElement$\}$; $F_{sys} = \{f_k \mid k = 1,...,K\}$ – множина функцій інтерпретації. $F_{sys} = \varnothing$; $Ind_{sys} = \{ind_l \mid l = 1,...,L\}$ – множина екземплярів сутностей (понять) предметної області; $Rl_{sys} = \{rl_q \mid q = 1,...,Q\}$ – множина семантичних правил; $Prop_{sys} = \{prop_u \mid u = 1,...,U\}$ – множина властивостей, що характеризують сутності (поняття) предметної області.

Створена онтологія містить сутності $(c_i \in C_{sys})$, які формують ієрархію відношень:

- Impairment (Порушення) – у сутності зберігається інформація про вид порушення:
  - CognitiveImpairment (когнітивні порушення);
  - HearingImpairment (порушення слуху);
  - VisualImpairment (порушення зору).
- Software:
  - Module (Модуль) – сутність містить набір компонентів та модулів, що наявні в системі:
    - ♦ ARModule (Модуль доповненої реальності) – у сутності міститься інформація про налаштування модуля, ступінь деталізації елементів;
    - ♦ MagnifyingGlassModule (Модуль збільшуваного скла) – сутність містить інформацію про модуль збільшуваного скла, його налаштування та параметри використання;
    - ♦ OrganizerModule (Модуль планування задач) – сутність містить інформацію про модуль планування задач та параметри його налаштування.
  - UserInterface (Інтерфейс користувача):
    - ♦ MultimediaElement (Мультимедійний елемент) – використовується для опису мультимедійних елементів;
    - ♦ TextElement (Текстовий елемент) – сутність описує налаштування відображення текстових елементів залежно від потреб користувача.
- User:
  - ElderlyPerson (людина похилого віку) – сутність містить інформацію про особу з когнітивними порушеннями;
  - Guardian (опікун) – сутність містить інформацію про опікуна.

Для додаткового опису кожної зі створених сутностей додано набір властивостей, що описують відповідні характеристики.

Властивості сутностей формують ієрархію:

- MagnifyingGlassDataProperty – містить інформацію про властивості модуля “збільшуване скло”:
  - TextToSpeech_Volume – вказує необхідний рівень гучності для озвучення розпізнаного тексту;
  - TextToSpeechEnabled – вказує, чи необхідна функція “Text-To-Speech”;
  - NeedTextRecognition – вказує, чи необхідно здійснювати розпізнавання тексту для подальшого озвучення.
- ARModuleDataProperty – містить інформацію про властивості модуля доповненої реальності:
  - HelpOrientation – властивість вказує, чи необхідно вмикати допомогу для орієнтації в просторі;
  - Mode – вказує режим функціонування модуля доповненої реальності. Наприклад: “Допомога орієнтації в просторі”, “Створення асоціацій з об’єктами”.

- CognitiveImpairmentDataProperty – містить інформацію про властивості сутності CognitiveImpairment:
  - AbstractThinking – властивість вказує, чи є в людини похилого віку проблеми з абстрактним мисленням;
  - OrientationInSpace – властивість вказує, чи є у людини похилого віку проблеми з орієнтуванням у просторі;
  - ProblemWithMemory – властивість вказує, чи є у людини похилого віку проблеми з пам’яттю;
  - SpeechImpairment – властивість вказує, чи є порушення мови в людини похилого віку.
- HearingImpairmentDataProperty – містить інформацію про властивості сутності HearingImpairment:
  - HearingLevel – властивість вказує на рівень слуху особи.
- MultimediaDataProperty – містить інформацію про властивості мультимедійних об’єктів:
  - NeedAdditionalControls – властивість визначає необхідність додаткових елементів управління для роботи із мультимедійними елементами;
  - Volume – властивість визначає рекомендований рівень гучності мультимедійних елементів.
- TextDataProperty:
  - MaxTextSize – властивість вказує максимально можливий розмір тексту для певного елемента;
  - NeedTextToSpeech – властивість визначає, чи необхідно озвучувати текстовий елемент;
  - TextSize – визначає рекомендований розмір текстових елементів.
- UserDataProperty – містить інформацію про властивості користувача, його дані та роль;
- VisionImpairmentDataProperty – містить інформацію про властивості VisionImpairment.

Сформовану онтологічну модель предметної області, що враховує адаптивність компонент системи та графічного інтерфейсу користувача, подано на рис. 1.

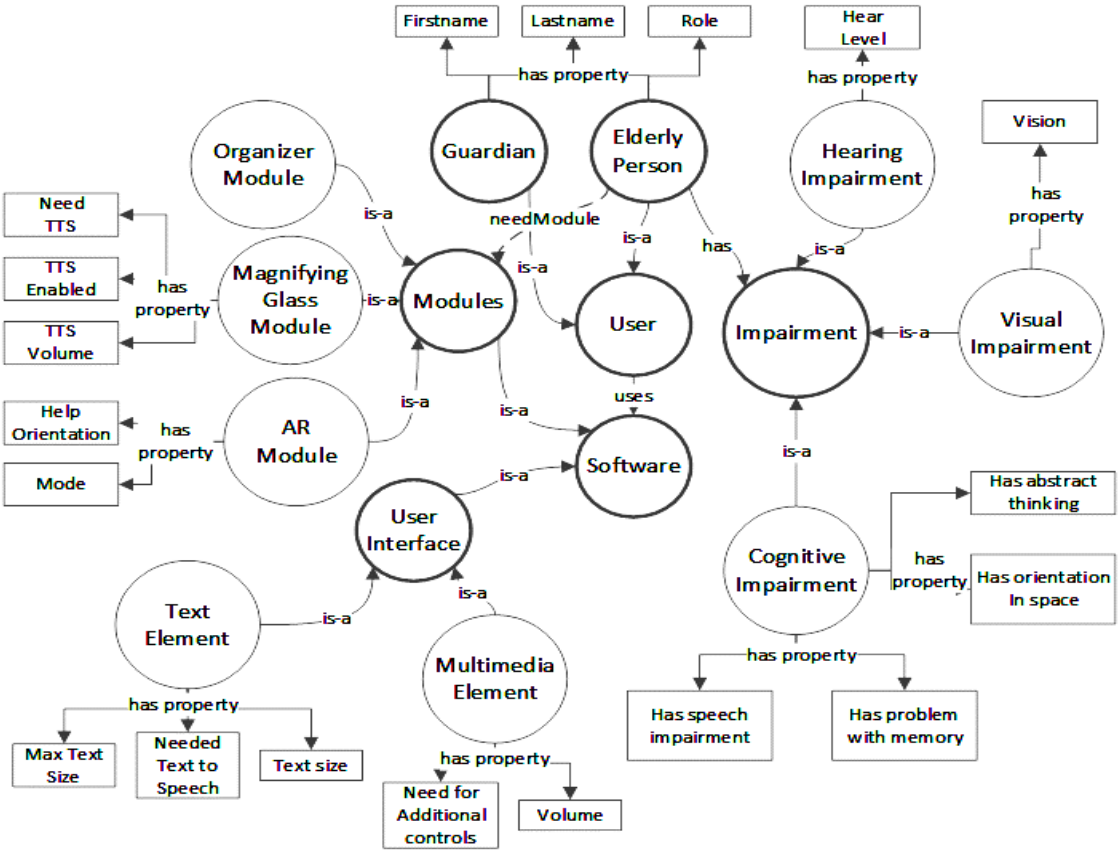

Рис. 1. Онтологічна модель адаптивної програмної системи

Враховуючи визначені властивості та відношення між сутностями, створюють множину SWRL-правил, що дають змогу прийняти рішення про необхідність використання певного модуля чи налаштувань. Мову SWRL (мова опису правил семантичної мережі) використано для створення правил, що дає змогу створити логічну імплікацію між тілом правила та його висновком, що складаються із окремих атомів. Атоми в SWRL правилі представлені як описи сутностей та їх властивості, що, своєю чергою, складаються зі змінних, екземплярів сутностей чи, власне, даних [16].

Наприклад, для забезпечення доступу до модуля організації задач опікуну та особі з когнітивними порушеннями сформовано таке правило:

$$\begin{aligned} &\text{User}(?\text{user}) \cap \text{Software}(?\text{software}) \cap \text{uses}(?\text{user}, ?\text{software}) \cap \\ &\text{OrganizerModule}(?\text{organizer}) \Rightarrow \text{needModule}(?\text{user}, ?\text{organizer}), \end{aligned} \tag{2}$$

де ?user, ?software, ?organizer – змінні SWRL-правила; uses, needModule – відношення між сутностями предметної області.

Формування налаштувань текстових елементів мобільного додатка, на основі даних про користувача із порушенням зору, здійснюється за допомогою сформованого правила:

$$\begin{aligned} &\text{ElderlyPerson}(?\text{person}) \cap \text{VisualImpairment}(?\text{impairment}) \cap \\ &\text{has}(?\text{person}, ?\text{impairment}) \cap \text{Vision}(?\text{impairment}, ?\text{vision}) \cap \\ &\text{swrlb:greaterThan}(?\text{vision}, 5) \cap \text{swrlb:lessThan}(?\text{vision}, 8) \Rightarrow \\ &\text{needElement}(?\text{person}, \text{cog:BigText}), \end{aligned} \tag{3}$$

де ?person, ?impairment, ?vision, ?mglassModule, ?textElement – змінні SWRL-правила; has, needElement – відношення між концептами предметної області; BigText – екземпляр концепту TextElement.

За допомогою зв'язків між сутностями та логічних правил, визначених для онтологічної моделі, формуються висновки про необхідність використання певних модулів системи, а також про параметри їх візуалізації та подання. Це рішення дає змогу сформувати основу для роботи процесу адаптації мобільного застосунку залежно від потреб конкретного користувача.

Архітектура адаптивної програмної системи

Архітектура програмного засобу для допомоги людям з когнітивними порушеннями, як вже зазначено вище, повинна забезпечити можливість динамічної модифікації та персоналізації графічного інтерфейсу та функціональності на основі даних про особу. Для забезпечення визначеного рівня адаптивності доцільно створювати систему із використанням принципів трирівневої клієнт-серверної та plugin-архітектури.

Принцип проєктування програмних засобів на основі трирівневої клієнт-серверної архітектури полягає у відокремленні вигляду системи від рівнів бізнес-логіки та даних [17]. Презентаційний рівень дає користувачу змогу взаємодіяти з функціональністю системи та відповідає за подання даних. Рівень бізнес-логіки відповідає за формування функціональних можливостей системи, а також за опрацювання даних, отриманих від користувача. Такий поділ дає змогу унеможливити втрату або пошкодження даних, у разі, якщо буде сформовано хибний або некоректний запит. Крім того, таке рішення дає змогу замінити або вдосконалити кожен рівень незалежно один від одного. Використання трирівневої клієнт-серверної архітектури гарантуватиме високу надійність та безпеку, масштабованість та конфігурованість, а також низькі вимоги до продуктивності.

Презентаційний рівень представлений у вигляді мобільного додатка та спрямований на безпосередню взаємодію із користувачем. Для забезпечення його розширюваності доцільно використати принципи plugin-архітектури [18]. Відповідно до вказаного підходу в системі виділено два види компонент:

- основна – містить основну функціональність та є фактично ядром програмного засобу, до них буде динамічно доєднуватися додаткова функціональність під час роботи програмного засобу;
- додаткова (plugin) – містить додаткову функціональність, що доповнює або розширює базову компоненту. Плагін можна додавати та видаляти в будь-який момент, і це не вплине на роботу інших плагінів.

Це забезпечить можливість повторного використання, розширюваності та взаємозамінності, оскільки кожен модуль буде реалізовано незалежно один від одного, а комунікація між модулями відбуватиметься безпосередньо через основну (батьківську) компоненту.

Рівень бізнес-логіки призначений для опрацювання даних користувача. Інформація, що надходить від презентаційного рівня, опрацьовується та надсилається на рівень даних, де відбувається синхронізація бази даних та онтологічної бази знань. На основі інформації, отриманої з рівня даних, формуються персоналізовані налаштування графічного інтерфейсу та список необхідних функціональних модулів.

Для відображення запропонованої архітектури програмної системи використано діаграму розгортання (рис. 2). Відповідно до вказаних принципів спроєктовано три рівні програмної системи:

- мобільний застосунок, що відповідає за надання інформації кінцевому користувачу і забезпечує можливість динамічної зміни графічного інтерфейсу та функціональності програми;
- вебсервер, що опрацьовує HTTPS-запити, отримані із презентаційного рівня, та виконує синхронізацію інформації між базою даних, використовуючи прикладний програмний інтерфейс pyodbc, та онтологічною базою знань;
- рівень даних представлений платформою, що містить базу даних, в якій зберігається інформація про користувачів та їх завдання, а також онтологічну базу знань, що використовується для формування персоналізованих налаштувань.

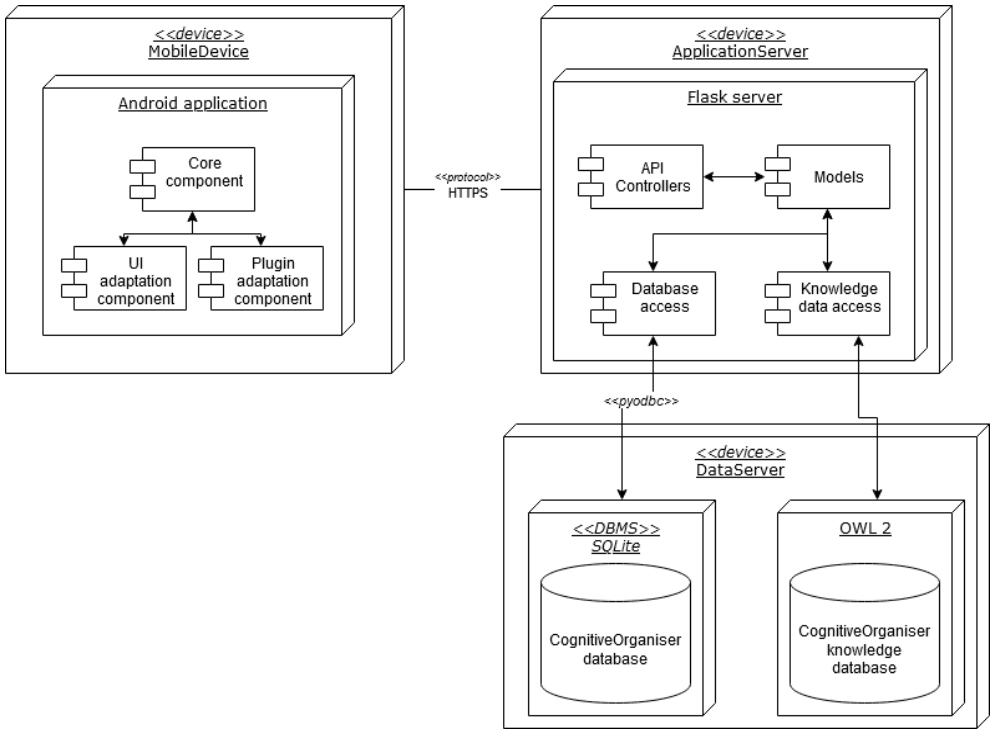

Рис. 2. Запропонована архітектура програмної системи

Метод адаптації програмного засобу

Спроєктована архітектура програмної системи забезпечує основу для персоналізації та налаштування мобільного застосунку на основі даних про користувача. Динамічне формування та модифікація функціональності та графічного інтерфейсу відбуваються у визначеній послідовності кроків.

Процес адаптації розпочинається з авторизації користувача в системі. Якщо користувач ще не має профілю, йому буде запропоновано зареєструватися в системі. Після цього відбувається синхронізація наявної інформації про особу між базою даних та онтологічною базою знань. Якщо в онтологічній базі знань немає інформації про користувача, відповідний запис формується та записується у відповідній сутності.

Після успішної авторизації користувач надсилає запит на сервер для динамічного формування персоналізованих налаштувань. Сформований запит містить JSON-файл, у якому вказано інформацію про порушення здоров'я користувача. Сервер опрацьовує отримані дані та переправляє їх до рівня даних, де відбувається синхронізація між базою даних та онтологічною базою знань. За необхідності в онтологічній моделі створюються сутності та зв'язки між ними, а також властивості, які відповідають інформації про порушення здоров'я відповідного користувача.

У разі успішного опрацювання інформації про користувача формуються персоналізовані налаштування системи. На цьому кроці визначають ключові характеристики шрифтів для різних елементів графічного інтерфейсу, необхідність технології перетворення тексту на голосове повідомлення для конкретних текстових елементів, а також налаштування мультимедійних елементів системи. Крім формування параметрів графічного інтерфейсу, аналізується необхідність використання додаткових модулів (наприклад, модуля "збільшуване скло" чи модуля "доповнена реальність"). Формалізовано алгоритм формування персоналізованих налаштувань подано на рис. 3.

```
ADAPTATION(Impairment, User)
 1  ontologyUser = GET-USER(User)
 2  for each impairment i ∈ Impairment
 3       impType = CREATE-INDIVIDUAL(i)
 4       properties = GET-DATA-PROPERTIES(i)
 5       for each property prop ∈ properties
 6            SET-DATA-PROPERTY(impType, prop)
 7       SET-OBJECT-PROPERTY(ontologyUser, impType)
 8  START-REASONER()
 9  settings = GET-SYNC-SETTINGS(ontologyUser)
10  return settings
```

Рис. 3. Фрагмент псевдокоду для формування налаштувань під потреби користувача

Персоналізовані налаштування надсилаються на мобільний застосунок, де паралельно відбувається адаптація графічного інтерфейсу користувача та функціональності системи.

Модифікація графічного інтерфейсу відбувається у такій послідовності:

1. Визначаються характеристики елемента, які необхідно змінити.
2. Формується модифікація вказаного елемента з урахуванням персоналізованих налаштувань.

Паралельно до зміни графічного інтерфейсу відбувається адаптація функціональності застосунку, яка визначена у такій послідовності кроків:

1. Визначаються характеристики модуля, який необхідно адаптувати.
2. Якщо визначений модуль відсутній, у такому випадку формується запит до сервера, щоб отримати файл із потрібним модулем. Отриманий модуль встановлюється та реєструється в мобільному застосунку.
3. Відбувається модифікація параметрів модуля відповідно до визначених характеристик.

Після успішної адаптації параметрів графічного інтерфейсу та функціональності мобільний застосунок оновлюється та застосовуються вказані зміни. Процес адаптації програмного застосунку під потреби конкретного користувача подано на рис. 4.

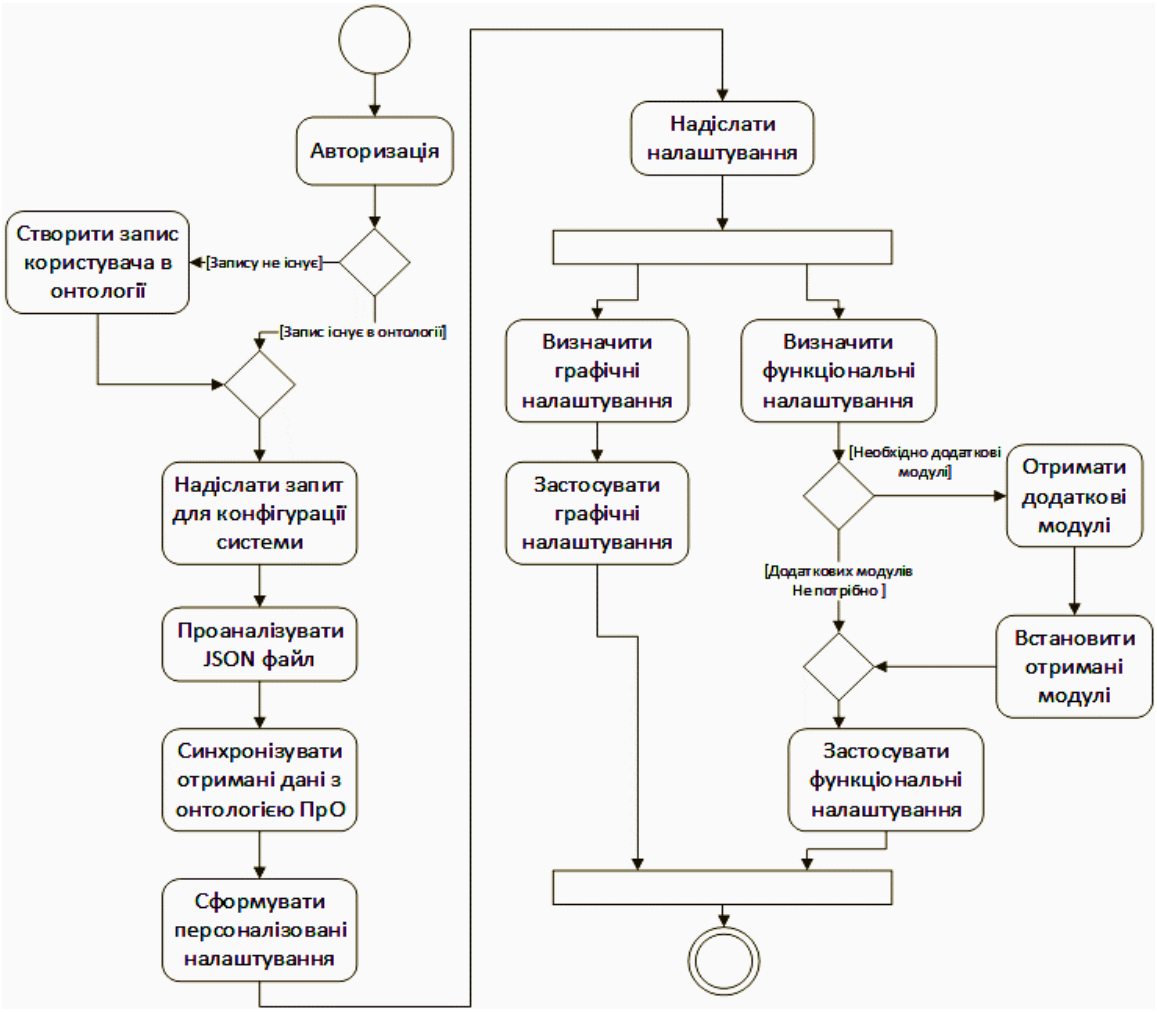

Рис. 4. Схема процесу адаптації ПЗ під потреби користувача

Використання поданого методу забезпечить можливість динамічного формування функціональності системи та модифікації параметрів графічного інтерфейсу під потреби конкретного користувача без необхідності повторного розгортання та конфігурування всієї системи.

Практична реалізація програмної системи

Адаптивна програмна складається із мобільного застосунку, що забезпечує взаємодію користувача із системою, а також вебсервера, що дає змогу опрацьовувати дані користувача, формувати персоналізовані налаштування, а також синхронізувати інформацію між базою даних та онтологічною базою знань.

Мобільний застосунок Cognit-Assist реалізований для платформи Android із використанням стандартних бібліотек та мови програмування Kotlin. Додатково для забезпечення взаємодії з рівнем бізнес-логіки використано бібліотеку Retrofit2, яка надає можливість доступу до прикладного програмного інтерфейсу (API). Для доступу до внутрішньої бази даних пристрою використано бібліотеку Room.

Мобільний застосунок Cognit-Assist містить такі компоненти:

- Core компонента – містить функції планування повсякденних завдань, комунікації із вебсервером для доступу та обміну інформацією.
- Компонента Plugin-adaptation – функцію для налаштування, пошуку (за допомогою Intent-фільтрів) та встановлення додаткових модулів, отриманих після виконання адаптації програмного забезпечення.
- Компонента UI-adaptation – функцію, параметри та користувацькі (custom) класи для налаштування та адаптації графічного інтерфейсу системи.

Рівень бізнес-логіки (прикладний програмний інтерфейс) відповідає за оброблення інформації, що надходить до сервера, та оформлення відповіді, надісланої клієнтові. Сервер реалізовано за допомогою мови програмування Python та вебфреймворку Flask із дотриманням вимог та принципів архітектурного стилю REST (Representational State Transfer).

Для забезпечення доступу створеного веб-ресурсу до онтологічної моделі предметної області використано модуль Owlready2, який застосовується для створення онтологічно орієнтованого програмного забезпечення. Запис інформації, отриманої від користувача, у внутрішнє сховище відбувається за допомогою модуля SQLAlchemy, який дає змогу взаємодіяти з реляційними базами даних. Синхронізація даних, що отримані з мобільного застосунку, та інформації з онтологічної бази знань відбувається за допомогою Pellet reasoner.

Адаптація здійснюється після верифікації даних користувача та їх реєстрації в онтологічній базі знань. Для формування персоналізованих налаштувань системи створюються екземпляри відповідних концептів онтологічної моделі, а також встановлюються зв'язки між створеними концептами предметної області. На основі семантичних правил відбувається прийняття рішень про необхідність призначення конкретних параметрів адаптації.

Результатом виконання вказаного методу є множина налаштувань, які надалі використовуються для персоналізації системи. Під час модифікації мобільного застосунку виділяються два окремих паралельних процеси: персоналізація графічного інтерфейсу та модифікація функціонала системи.

Налаштування зберігаються у секції SharedPreferences мобільного застосунку та використовуються для налаштування користувацьких елементів графічного інтерфейсу. Використання SharedPreferences забезпечує зберігання примітивних даних вигляду "ключ-значення" без використання зовнішньої бази даних, що дає змогу уникнути затримок під час виконання довготривалих SQL-запитів.

Адаптація функціональних характеристик додатка відбувається після завантаження та встановлення необхідних додаткових модулів. Після успішної інсталяції здійснюється пошук наявних у системі модулів за допомогою Intent-фільтрів. Отримана під час пошуку інформація (дані про модуль, його назва та ім'я головної активності (Activity) реєструється у мобільному застосунку.

Отже, створено новий підхід до динамічної адаптації програмної системи, зокрема її функціональності та графічного інтерфейсу, із використанням онтологічної моделі предметної області. Реалізований метод забезпечує можливість модифікації вказаних характеристик мобільного застосунку без необхідності повторної реконфігурації та розгортання, що, своєю чергою, дає змогу розширювати систему функціональними модулями під час її роботи.

Висновки

У роботі проаналізовано наукові дослідження, спрямовані на розроблення програмного забезпечення для допомоги людям із когнітивними віковими порушеннями. Встановлено, що наявні системи не забезпечують універсальності використання, а програмні рішення дають змогу адаптувати тільки інтерфейс або функціональність. Вирішенням цієї проблеми є використання програмних систем, які надають можливість адаптувати функціонал та графічний інтерфейс мобільного застосунку залежно від потреб користувача.

Аналізування відомих методів проєктування та розроблення адаптивних програмних систем дає підстави стверджувати, що використання онтологічного підходу забезпечує можливість відокремлення логіки функціонування програмної системи від механізму її реалізації.

На основі аналізу предметної області спроєктовано онтологічну модель адаптивної програмної системи для людей з когнітивними порушеннями, для якої визначено концепти та відношення між ними, що дають змогу створювати правила для динамічного конфігурування програмного забезпечення.

Спроєктовано архітектуру системи відповідно до принципів трирівневої клієнт-серверної та plugin-архітектури. В поєднанні з онтологічною моделлю предметної області це дасть змогу персоналізувати інтерфейс користувача та функціональність мобільного застосунку залежно від особливостей порушень здоров'я конкретного користувача.

Реалізований метод адаптації мобільного застосунку, на основі спроєктованих архітектури та онтологічної моделі, дає змогу визначити оптимальні налаштування для конкретного користувача

залежно від порушень його здоров'я. Зокрема, для людей із порушеннями зору формуються налаштування, що забезпечать зміну характеристик текстових елементів та можливість їх озвучення. Запропонований процес адаптації також дає змогу виконувати пошук та встановлювати додаткові функціональні модулі для мобільного застосунку.

Використання запропонованого методу забезпечить можливість динамічного формування функціональності системи та модифікації параметрів графічного інтерфейсу під потреби конкретного користувача без необхідності повторного розгортання та конфігурування всієї системи.

Список літератури

1. Steinhubl, S. R., Muse, E. D., & Topol, E. J. (2015). The emerging field of mobile health. *Science Translational Medicine*, 7(283), 283–288. Retrieved from: https://doi.org/10.1126/scitranslmed.aaa3487.

2. Gómez, J., Alamán, X., Montoro, G., Torrado, J. C., & Plaza, A. (2014). AmICog – mobile technologies to assist people with cognitive disabilities in the work place. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *2*(4), 9–17. Retrieved from: https://doi.org/10.14201/ADCAIJ201324917.

3. Gomez, J., Montoro, G., Torrado, J. C., & Plaza, A. (2015). An adapted wayfinding system for pedestrians with cognitive disabilities. *Mobile Information Systems*, *2015*. Retrieved from: https://doi.org/10.1155/2015/520572.

4. Поцелуйко, А. С. (2017). Разработка адаптивных интерфейсов мобильных приложений для людей с ограниченными возможностями. *Смотр-конкурс научных, конструкторских и технологических работ студентов Волгоградского государственного технического университета: тезисы докладов*, Волгоград, 189–190.

5. Дворянкин, А. М., Романенко, Р. Р., & Поцелуйко, А. С. (2015). Разработка алгоритма адаптации интерфейсов для людей с ограниченными возможностями. *Известия Волгоградского государственного технического университета*, 14, 49–55.

6. Буров, Є. В. (2013). Ефективність застосування онтологічних моделей для побудови програмних систем. *Математичні машини і системи*, 1, 44–55.

7. Burov, Y., Mykich, K., & Karpov, I. (2020). Building a Versatile Knowledge-Based System Based on Reasoning Services and Ontology Representation Transformations. *Proceedings of 2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, 2, 255–260.

8. Поцелуйко, А. С., Романенко, Р. Р., Кульцова, М. Б. (2017). Метод адаптации графического интерфейса для людей с ограниченными возможностями на основе онтологической модели пользователя и паттернов интерфейса. *Известия ВГТУ*, 1. 89–97.

9. Литвин, В. В., Демчук, А. Б., & Войчишен, М. М. (2011). Метод побудови інтелектуального агента на основі онтології предметної області. *Вісник Національного університету "Львівська політехніка"*, 715, 215–224.

10. Dyvak, M., Kovbasistyi, A., & Melnyk, A. (2019). Recognition of Relevance of Web Resource Content Based on Analysis of Semantic Components. *Advanced Computer Information Technologies (ACIT): Conference proceedings*, 297–302. Retrieved from: https://doi.org/10.1109/ACITT.2019.8779897.

11. Шаров, С. В., Лубко, Д. В., & Осадчий, В. В. (2015). Вибір моделі представлення знань у системі ІСІКС. *Системи обробки інформації*, 11, 108–111.

12. Литвин, В. В. (2010). Автоматизація процесу розвитку базової онтології на основі аналізу текстових ресурсів. *Вісник Національного університету "Львівська політехніка"*.

13. Lytvyn, V., Burov, Y., Vysotska, V. & Hryhorovych V. (2020). Knowledge Novelty Assessment During the Automatic Development of Ontologies. *IEEE Third International Conference on Data Stream Mining & Processing (DSMP)*, 372–377, Retrieved from: https://doi.org/10.1109/DSMP47368.2020.9204124.

14. Поцелуйко, А. С., Кравец, А. Г., & Кульцова, М. Б. (2019). Персонализация интерфейсов мобильных приложений на основе паттернов интерфейсов для людей с ограниченными возможностями. *Прикаспийский журнал: управление и высокие технологии*, 3, 17–27.

15. Піднебесна, Г. А. (2014). Онтологічний підхід до конструювання інтерфейсу користувача в системах індуктивного моделювання. *Індуктивне моделювання складних систем*, 6, 117–125.

16. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21(79), 1–31.

17. Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley.

18. Ameller, D., Collell, O., & Franch, X. (2013). The Three-Layer architectural pattern applied to plug-in-based architectures: the Eclipse case. *Software: Practice and Experience*, 43(4), 391–402.

References

1. Steinhubl, S. R., Muse, E. D., & Topol, E. J. (2015). The emerging field of mobile health. Science Translational Medicine, 7(283), 283–288. Retrieved from: https://doi.org/10.1126/scitranslmed.aaa3487.

2. Gómez, J., Alamán, X., Montoro, G., Torrado, J. C., & Plaza, A. (2014). AmICog – mobile technologies to assist people with cognitive disabilities in the work place. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *2*(4), 9–17. Retrieved from: https://doi.org/10.14201/ADCAIJ201324917.

3. Gomez, J., Montoro, G., Torrado, J. C., & Plaza, A. (2015). An adapted wayfinding system for pedestrians with cognitive disabilities. *Mobile Information Systems, 2015*. Retrieved from: https://doi.org/10.1155/2015/520572.

4. Potseluyko, A. S. (2017). Development of adaptive interfaces for mobile applications for people with disabilities *Review-competition of scientific, design and technological works of students of the Volgograd State Technical University: Abstracts*, Volgograd, 189–190.

5. Dvoryankin, A. M., Romanenko, R. R., & Potseluyko, A. S. (2015). Development of an algorithm for adapting interfaces for people with disabilities. *Bulletin of the Volgograd State Technical University*, 14, 49–55.

6. Burov, Y. V. (2013). The effectiveness of ontological models for building software systems. *Mathematical Machines and Systems*, 1, 44–55.

7. Burov, Y., Mykich, K., & Karpov, I. (2020). Building a Versatile Knowledge-Based System Based on Reasoning Services and Ontology Representation Transformations. *Proceedings of 2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, 2, 255–260.

8. Potseluyko, A. S., Romanenko, R. R., Kultsova, M. B. (2017). A method for adapting a graphical interface for people with disabilities based on the ontological model of the user and interface patterns. *Izvestia VSTU*, 1, 89–97.

9. Lytvyn, V. V., Demchuk, A. B., & Voychyshen, M. M. (2011). A method of constructing an intellectual agent based on the ontology of the subject area. *Bulletin of the Lviv Polytechnic National University*, 715, 215–224.

10. Dyvak, M., Kovbasistyi, A., & Melnyk, A. (2019). Recognition of Relevance of Web Resource Content Based on Analysis of Semantic Components. *Advanced Computer Information Technologies (ACIT) : Conference proceedings*, 297–302.Retrieved from: https://doi.org/10.1109/ACITT.2019.8779897.

11. Sharov, S. V., Lubko, D. V., & Osadchyy, V. V. (2015). Choice of knowledge representation model in ISICS system. *Information processing systems*, 11, 108–111.

12. Lytvyn, V. V. (2010). Automation of the process of basic ontology development based on the analysis of text resources. *Bulletin of the Lviv Polytechnic National University*.

13. Lytvyn, V., Burov, Y., Vysotska, V. & Hryhorovych V. (2020). Knowledge Novelty Assessment During the Automatic Development of Ontologies. *IEEE Third International Conference on Data Stream Mining & Processing (DSMP)*, 372–377, Retrieved from: https://doi.org/10.1109/DSMP47368.2020.9204124.

14. Potseluyko, A. S., Kravets, A. G., & Kultsova, M. B. (2019). Personalization of mobile application interfaces based on interface patterns for people with disabilities. *Caspian Journal: Management and High Technologies*, 3, 17–27.

15. Pidnebesna, H. A. (2014). Ontological approach to user interface design in inductive modeling systems. *Inductive modeling of complex systems*, 6, 117–125.

16. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, *21*(79), 1–31.

17. Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley.

18. Ameller, D., Collell, O., & Franch, X. (2013). The Three-Layer architectural pattern applied to plug-in-based architectures: the Eclipse case. *Software: Practice and Experience*, *43*(4), 391–402.

ADAPTIVE SOFTWARE SYSTEM BASED ON ONTOLOGICAL APPROACH FOR PEOPLE WITH COGNITIVE IMPAIRMENTS

Dmytro Fedasyuk[1], Illia Lutsyk[2]

Lviv Polytechnic National University
[1] dmytro.v.fedasyuk@lpnu.ua ORCID 0000-0003-3552-7454
[2] illia.lutsyk.mnpz.2019@lpnu.ua ORCID 0000-0001-7117-5332

The paper presents a method of creating an adaptive software system to help people with cognitive impairments, based on the use of an ontological model of the subject area.

The specifics of creating software tools to help people with cognitive impairments are analysed. The features of using the ontological approach for the formation of adaptive functionality and graphical interface are revealed and their advantages over traditional methods are analysed. It was found that when using this method, there is no need to recompile and fully deploy the software system in the event of a change in business logic. An ontological model of the subject area has been designed, which will make it possible to customize the system for the needs of a particular user. The architecture of a software system based on an ontological model of the subject area is proposed, which takes into account the possibility of personalizing the components of the system and the user interface without the need to re-deploy the system. The process of adaptation of a mobile application based on data about health disorders of the user using an ontological model of the subject area is disclosed.

The result of the research is the development of a software system that implements the proposed adaptation process and allows to modify a mobile application for the needs of a specific user, using an ontological model of the subject area.

Key words: ontological approach; ontological model; adaptive software system; mobile application; cognitive impairment.