УДК 004.051 **DOI: 10.20998/2079-0023.2021.01.15**

Р. О. ГАМЗАЄВ, М. В. ТКАЧУК

ЗАСТОСУВАННЯ МЕТОДІВ І ТЕХНОЛОГІЙ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ДЛЯ КОНФІГУРУВАННЯ ДИНАМІЧНИХ ЛІНІЙОК ПРОГРАМНИХ ПРОДУКТІВ

У статті проведено аналіз існуючих підходів до вирішення задачі динамічного конфігурування у лінійках програмних продуктів (ЛПП). ЛПП це набір програмних систем, що мають спільні і варіабельні компоненти і використовують набір парадигм і методів до розробки. Завдяки використанню яких надаються можливості налаштовувати програмні рішення відповідно до потреб кінцевих користувачів. Було показано, що для вирішення цієї проблеми доцільно використовувати методи і технології побудови сучасних рекомендаційних систем (РС). Проведено аналіз існуючих підходів і методів побудови РС, були розглянуті такі можливі методи як: кластеризація, марковский процес прийняття рішень, факторизація матриць. В результаті огляду інтелектуальних методів розробки РС та дослідження функціональних можливостей технологій реалізації РС у проектах з відкритим кодом, для подальшого використання саме в задачах конфігурування динамічних ЛПП запропоновано метод N-вимірної контекстно-залежної тензорної факторизації та інструментальна система CARSkit. Розроблені функціональні вимоги та запропонована архітектура прототипу РС, яка уможливлює автоматизацію конфігурування програмних компонентів у системах «Розумний будинок» (РБ), і яка може бути програмно реалізована засобами системи CARSkit та алгоритмами обробки консолідованих даних на мові Python. Ця реалізація дозволяє побудувати процес відстеження змін у зовнішньому середовищі і передавати інформацію в РБ і після аналізу вхідних даних обробляти в РС для відстеження змін у контекстної інформації. Під час подальших досліджень заплановано проведення обчислювальних експериментів з урахуванням специфіки систем «Розумний будинок» і застосування кількісних метрик для оцінки ефективності алгоритмів тензорної авторизації для прогнозування динамічних змін програмних компонентів в цих системах.

Ключові слова: рекомендаційна система, лінійка програмних продуктів, варіабельність, динамічне конфігурування, архітектура

Р. А. ГАМЗАЕВ, Н.В. ТКАЧУК

ПРИМЕНЕНИЕ МЕТОДОВ И ТЕХНОЛОГИЙ РЕКОМЕНДАЦИОННЫЙ СИСТЕМ ДЛЯ КОНФИГУРИРОВАНИЯ ДИНАМИЧЕСКИХ ЛИНЕЕК ПРОГРАММНЫХ ПРОДУКТОВ

В статье проведен анализ существующих подходов к решению задачи динамического конфигурирования в линейках программных продуктов (ЛПП). ЛПП представляют набор программных систем, имеющих общие и вариабельные компоненты и использующие набор парадигм и методов к разработке. Благодаря использованию которых предоставляются возможности настраивать программные решения в соответствии с потребностями конечных пользователей. Было показано, что для решения этой проблемы целесообразно использовать методы и технологии построения современных рекомендательных систем (РС). Проведен анализ существующих подходов и методов построения РС, были рассмотрены такие возможные методы как: кластеризация, марковский процесс принятия решений, факторизация матриц. В результате рассмотрения интеллектуальных методов разработки РС и исследования функциональных возможностей технологий реализации РС в проектах с открытым кодом, для дальнейшего использования именно в задачах конфигурирования динамических ЛПП предложен метод N-мерной контекстно-зависимой тензорной факторизации и инструментальная система с открытым исходным кодом CARSkit. Разработаны функциональные требования и предложена архитектура прототипа РС, которая делает автоматизацию конфигурирования программных компонентов в системах «Умный дом» (УД), а также может быть программно реализована средствами системы CARSkit и алгоритмами обработки консолидированных данных на языке Python. Эта реализация позволяет построить процесс отслеживания изменения во внешней среде и передавать информацию в УД и после анализа входных данных обрабатывать в РС для отслеживания изменения во контекстной информации. В качестве дальнейших исследований запланировано проведение вычислительных экспериментов с учетом специфики систем «Умный дом» и применения количественных метрик для оценки эффективности алгоритмов тензорной факторизации для прогнозирования динамических конфигураций программных компонентов в этих системах.

Ключевые слова: рекомендационная система, линейка программных продуктов, вариабельность, динамическое конфигурирование, архитектура

R. O. GAMZAYEV, M.V. TKACHUK

USING METHODS AND TECHNOLOGIES OF RECOMMENDATION SYSTEMS FOR DYNAMIC SOFTWARE PRODUCT LINES CONFIGURATION

Software product lines (SPL) dynamic configuring process could use methods for recommendation system (RS) elaboration. An overview and analysis of such methods was done in this paper. SPL represent a set of software systems that have common and variable functional components and use a set of paradigms and methods for development. In the classical static SPL the process of configuring performed before executing and performing in the operation environment (OE), in contrast dynamic software product lines performs after executing in the OE. Through the use of which it is possible to customize software solutions in accordance with the needs of end users. The following possible methods to build RS were considered: clustering, Markov decision-making process, matrix factorization. According to the review of the intelligent RS method development and researching of the functionalities of such systems in some open-source projects it was proposed to use N-dimensional context-dependent tensor factorization method and CARSkit tool system. Functional requirements and software architecture of the RS were developed. It allows to automatize software components configuration in the „Smart Home" (SH) systems that could be implemented with CARSkit software toolkit and algorithms implemented with programming language Python. This implementation allows to build a process for tracking changes in the external environment and transfer information to the SH system and, after analyzing the input data, process it in the RS to track changes in the context information. In the future research some additional quantitative experiments will be performed considering the specifics of the SH systems, additionally quantitative metrics will be used for efficiency assessment of the tensor factorization algorithms to predict the dynamic configurations of software components in these systems.

Keywords: recommendation system, software product line, variability, dynamic configuration, architecture.

Вступ. Постійне ускладнення функціональних задач, які мають бути вирішені шляхом використання програмних систем (ПС) у різних предметних областях (ПрО) передбачає можливість побудови та подальшого ефективного використання вже не окремих ПС, а цілих їх взаємопов'язаних сукупностей, які в сучасній програмній інженерії отримали назву лінійок програмних продуктів (software product line). Такі лінійки

програмних продуктів (ЛПП) мають спільний набір загальних функцій (компонентів), які налаштовуються відповідно до потреб користувачів у певному контексті їх функціонування і які націлені на забезпечення варіабельності (variability) ЛПП, що в свою чергу передбачає можливість конфігурування та налаштування іх певних функцій та окремих параметрів [1]. У так званих статичних ЛПП визначення властивостей варіабельності відбувається ще до процесу розміщення та виконання у відповідному операційному середовищі, тобто на етапі їх архітектурного проектування шляхом побудови відповідної моделі варіабельності, наприклад, FODA-моделей та деяких ін. [2]. Але внаслідок структурної складності та функціональної насиченості більшості ЛПП, в таких системах досить часто виникають проблеми із продуктивністю і масштабованістю, і тому в сучасній програмній інженерії активно розробляються та досліджуються динамічні ЛПП (Dynamic Software Product Lines) [2], в яких важливу роль відіграють механізми адаптації та конфігурування варіабельних функцій і властивостей безпосередньо на етапі виконання та розміщення їх окремих компонентів. Для вирішення цих актуальних науково-практичних задач необхідно застосування нових інтелектуальних методів розробки ЛПП, і зокрема, цікавим напрямком таких досліджень може бути вивчення можливостей сучасних рекомендаційних систем (recommendation systems), які використовують знання-орієнтовані методи і технологіях ефективної фільтрації консолідованої інформації і забезпечують вибір конфігурацій персоналізованих продуктів для різних груп їх користувачів у відповідності з динамічними змінами в навколишньому середовищі.

Слід зазначити, що дослідження у цьому напрямку проводяться вже досить активно і, зокрема, проблема адаптації ЛПП у відповідь на зміну контексту висвітлена в [3]. В ній традиційні варіабельні моделі функцій ЛПП доповнюються набором логічних правил (продукцій), які явно визначають, за якої умови має відбуватися її переконфігурація. В [4] також розглянуті декілька підходів до динамічного налаштування властивостей ЛПП, це застосування методів машинного навчання, що дозволяє виявити явно не передбачувані зміни контексту та створювати нові правила адаптації. Другий підхід, в саме: еволюція, фокусується на забезпеченні механізма постійного моніторигу змін у вже існуючих конфігураціях ЛПП, а потім – прогнозування нових, на основі обробки цих даних, при цьому в [4] зазначено, що для складних ЛПП ці два підходи доцільно використовувати комбіновано. В [5] запропонована доменна модель для аналізу альтернативних механізмів варіабельності в ЛПП для мобільних застосунків із використання мови Java і показана можливість використання деяких методів імплементації варіабельності, які є специфічними для цієї мови програмування. В той же час можна констатувати, що можливості використання саме рекомендаційних систем ще недостатньо висвітлені в існуючих публікаціях за темою конфігурування динамічних ЛПП.

Метою цієї статті є аналіз інтелектуальних методів і технологій побудови рекомендаційних систем (РС) для вирішення задач самоконфігурування ДЛПП, дослідження можливостей застосування вже існуючих проєктів реалізації РС з відкритим кодом, розробка архітектури прототипу такої РС для застосування в процесах побудови ДЛПП в предметній області «Розумний будинок», а також визначення подальших задач для програмної реалізації та експериментального дослідження працездатності та ефективності запропонованого підходу.

Формальне визначення та узагальнена функціональна структура рекомендаційних систем. Сучасні РС представляють собою окремий клас систем інтелектуальної обробки даних, які застосовуються для того, щоб передбачити, які саме інформаційні об’єкти або сервіси: певні товари, послуги, медійний контент тощо, будуть цікаві різним групах їх користувачів [6]. Для цього РС обробляють додаткову, як правило, накопичену раніше, ретроспективну інформацію стосовно їх профілів (або контекстів), тобто їх вже існуючих уподобань, можливих супутніх запитів, додаткових потреб та ін.

Формальне визначення РС можна подати у наступний спосіб [6]: нехай S – множина користувачів системи, I – множина всіх елементів, які підпадають під категорію їх переваг, $R \subseteq I$ – ранжируваний список підмножини таких елементів, а r – це певний елемент в списку R. Проблема надання рекомендації полягає в тому, щоб вибрати $r \in R$ таким чином, аби воно задовольняло вимого відповідних користувачів $s \subseteq S$.

Нехай E – метрика оцінки задоволеності окремого користувача, яка приймає значеня z, де z – деяке дійсне число. Тоді якщо f визначає функцію рекомендації r елементів для s користувачів, то проблема формування рекомендації може бути сформульована таким чином:

$$f(r, s) = E \rightarrow \max$$

Питання класифікації сучасних РС заслуговують окремої уваги, і воно не є предметом розгляду у цьому дослідженні, але навіть стислий огляд вже існуючих публікацій за цією проблематикою [6–8] дає підставу вважати, що одним з найбільш поширених типів РС є системи на основі спільної фільтрації (collaboratibe filtering).

Спільна фільтрація (СФ) – це трьох ступеневий процес, що починається зі збору користувацької інформації, потім будується матриця для розрахунку асоціацій і, нарешті, надається найбільш вірогідна рекомендація. Її основне припущення полягає в наступному: ті, хто однаково оцінював будь-які предмети в минулому, схильні давати схожі оцінки інших предметів і в майбутньому [8]. На спрощеній діаграмі класів (рис. 1) деякої узагальненої РС зображено 4 основні сутності, що відображають фундаментальні відносини, необхідні для виконання її основних функцій.

Головний клас RecommendationSystem агрегує в собі генеруючий клас RecommendationAlgorithm для алгоритмів надання рекомендацій, і клас бази даних DataBase, що містить інформацію щодо всіх груп

користувачів та їх оцінок, і клас User, який має свій унікальний ідентифікатор і який надає свої оцінки певних об'єктів (сервісів).

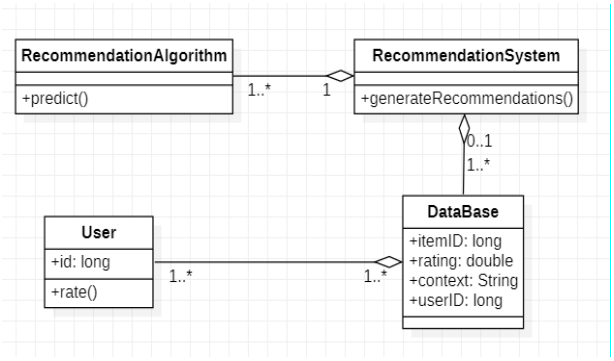

Рис. 1. Спрощена діаграма базових класів РС

Аналіз інтелектуальних методів розробки рекомендаційних систем. Кластерізація. Ця група традиційних методів побудови кластерів схожих об'єктів шляхом вимірювання подібності за допомогою таких показників, як відстань Мінковського та кореляція Пірсона [9]. Для двох об'єктів даних $X = (x_1, x_2, x_3 \dots x_n)$ та $Y = (y_1, y_2, y_3 \dots y_n)$ відстань Мінковського визначається як:

$$d(X,Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q},$$

де n – номер розмірності об'єкта, x_i, y_i значення i-го виміру об'єкта X та Y відповідно,

q – ціле додатне число.

Коли $q = 1$, d – відстань Манхеттена; коли $q = 2$, d – евклідова відстань.

Методи кластеризації можна класифікувати на три категорії: методи розділення, методи на основі щільності та ієрархічні методи. Методи на основі щільності зазвичай шукають щільні кластери об'єктів, розділених розрідженими областями, що представляють інформаційний шум. Методи ієрархічної кластеризації створюють декомпозицію набору об'єктів даних, використовуючи певні критерій. У більшості ситуацій кластеризація є проміжним етапом, і результуючі кластери використовуються для подальшого аналізу.

Марківський процес прийняття рішень. Модель Марківського процесу рішення (Markov Decision Process – MDP) це модель для послідовних стохастичних проблем прийняття рішень, яка часто використовується застосунках, де агент (користувач) впливає на навколишнє середовище своїми діями. MDP можна визначити як чотири набори:$\langle S, A, R, tr \rangle$, де S – набір станів, A – набір дій, R – функція реального значення винагороди для кожної пари стану–дії, і tr – ймовірність переходу між кожною парою станів, що обумовлюються кожною дією, або інакше кажучи, функція переходу станів [10]. Стан $s \in S$ містить всю відповідну інформацію про стан середовища. Дії викликають зміни стану, а вплив дій на стани фіксується функцією переходу. Функція переходу призначає розподіл ймовірностей по кожній парі(стану, дії). Отже, $tr(s_1, \alpha, s')$ – це ймовірність здійснення переходу від стану s до стану s', коли виконується α. Нарешті, функція винагороди присвоює дійсне значення кожній парі (стан, дія), яке описує винагороду (або вартість) виконання цієї дії у такому стані. Часто винагорода є лише функцією стану, і, отже, є мірою бажаності досягнення кожного стану.

Факторизація матриць. Алгоритми матричної факторизації (Matrix Factorization – MF) працюють шляхом розкладання певної матриці конфігурації V взаємодії користувача з елементом його контента на добуток двох прямокутних матриць меншої розмірності: $\boldsymbol{W}$ і $\boldsymbol{H}$, що уможливлює дослідження прихованих факторів для кожного користувача та елемента даних (див. рис. 2).

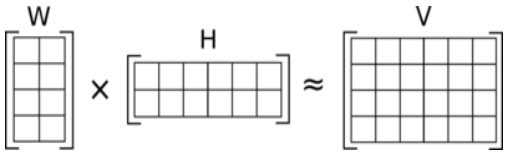

Рис. 2. Схема розкладання матриці конфігурацій

Основне припущення полягає в тому, що як користувачі, так і елементи можуть моделюватися зменшеною кількістю факторів. Формально матриця конфігурації $\boldsymbol{V}$ та її факторизація представляється наступним чином [11]:

$$\boldsymbol{V} = \begin{bmatrix} c_{1_1} & \cdots & c_{1_m} \\ \vdots & \ddots & \vdots \\ c_{n_1} & \cdots & c_{n_m} \end{bmatrix} = \boldsymbol{H}_{n \times h} \cdot \boldsymbol{W}_{h \times m},$$

де $\boldsymbol{V}$ визначається як набір конфігурацій $\boldsymbol{V} = \{\vec{c}_1, \dots, \vec{c}_n\}$;

вибрані функції кодуються як «1», невибрані – як «0», та невизначені функції як «–1».

Крім того, $\boldsymbol{H}$ є латентною матрицею конфігурацій і $\boldsymbol{W}$ – латентною матрицею функцій, де n – кількість конфігурації в $\boldsymbol{V}$, m – кількість функцій, h – число латентних розмірів. Окремим випадком підходу MF є N-вимірна тензорна факторизація [12]. Факторизація N-мірного тензора при ранзі розкладання k формує N матриць, що складаються з k стовпців, які представляють відображення кожного окремого виміру тензора на k фактор–вимірів семантичного простору. Схематично це можна зобразити на рис. 3.

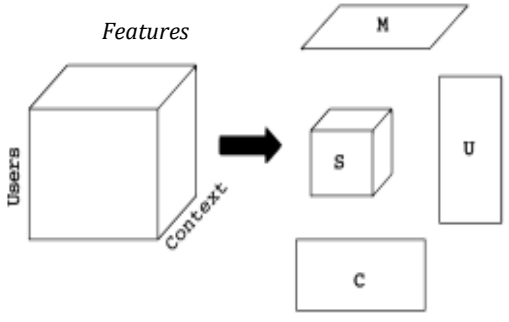

Рис. 3. Схема розкладання 3-вимірного тензора

Тензорна факторизація як найкраще підходить для вирішення задачі управління конфігураціями динамічних SPL, в яких наявна проблема експоненціально зростаючого конфігураційного простору [12].

Огляд технологій реалізації рекомендаційних систем на прикладах існуючих проектів з відкритим кодом. Для розробки власного прототипу РС доцільно провести попередній аналіз деяких існуючих проєктів створення таких систем, з метою виявлення їх характерних технологічних особливостей, переваг та недоліків.

Проєкт Racoon Recommendation Engine. Ця система [13] представляє собою простий у використанні механізм рекомендацій на основі СФ, який використовує відстань (міру) Жаккарда для визначення схожості між користувачем та його k найближчими сусідами в масиві відповідних контекстів, і що дозволяє вимірювання двійкових рейтингових даних (тобто в термінах «подобається» / «не подобається»). Готовний модуль Racoon можна поєднати з будь-якою БД, оскільки його реалізація не залежить від конкретної структури даних про користувачів і він використовує тільки їх унікальні ідентифікатори. Racoon повністю реалізовано на мові Javascript, а для роботи з БД використовує асинхронні, неблокуючі функції платформи Node.js. Рекомендації та рейтинги накопичуються у проміжному noSQL – сховищі даних Redis, яке зберігає в оперативній пам'яті всі набори відповідних даних.

Проєкт LensKit for Python. Це інструментарій [14] з відкритим кодом використовується для побудови, дослідження та вивчення особливостей функціонування РС у різних предметних галузях. Спочатку реалізований як Java-фреймворк, остання версія LensKit містить також інструментальний пакет на мові Python, який забезпечує реалізацію декількох стандартних алгоритмів СФ, обчислення метрик оцінки якості отриманих результатів, а також прості у реалізації API – інтерфейси, що дозволяють легко використовувати його в інших застосунках. Так, наприклад, інтерфейс Algorithm, заснований на шаблонах, надає додаткові методи: Predictor реалізує передбачення уподобань користувачів; Recommender знаходить топ-N рекомендацій та повертає їх рейтинговий список, разом із відповідними оцінками, а метод CandidateSelector дозволяє знайти об'єкт – кандидат для рекомендації, коли в алгоритмі попередньо не надано жодного набору кандидатів для рекомендації.

Типовий експеримент із тестовою РС, яка створена з використанням LensKit for, складається з трьох основних етапів: 1) підготовка навчальних та тестових наборів з контекстами користувачів, 2) тренування обраного алгоритму СФ та формування вихідних рекомендацій; 3) обчислення їх метрик якості.

Проект Surprise. Цей інструмент [15] для побудови РС є бібліотекою скриптів Python для побудови та аналізу алгоритмів СФ. Surprise пропонує колекцію модулів-оцінювачів (estimator) для прогнозування результатів СФ, серед яких реалізовані як класичні так і модифіковані алгоритми на основі подібності (similarity-based), а також алгоритми, які засновані на факторизації матриць: такі, як SVD (Single Value Decomposition) або MF (Matrix Factorization). Система підтримує вбудовані метрики якості результатів СФ, а також інструменти для вибору моделі та автоматичного пошуку гіперпараметрів СФ. У функціоналі Surprise всі алгоритми є похідними від базового класу *AlgoBase*, де реалізовані деякі ключові методи: наприклад, прогнозування (predict), тренування (fit) та перевірка (test), які можна використати для реалізації будь-якого власного алгоритму прогнозування.

Проєкт CARSkit. Ця система [16] представляє собою Java-інструментарій із відкритим вихідним кодом для роботи із контекстно-залежними РС, вона реалізує сучасні алгоритми СФ та надає стандартну платформу для їх розгортання. Функції CARSKit забезпечують процес гнучкого конфігурування РС за допомогою головного файлу властивостей (property file), який включає параметри налаштування певного алгоритму СФ, опис джерел його вхідних та вихідних даних, посилання на методи оцінки, тощо.

Результати цього стислого огляду функціональних можливостей проектів РС з відкритим вихідним кодом представлені в табл. 1.

На цій основі можливо зробити мотивований висновок, що для експериментальної реалізації власної РС доцільно обрати саме інструментарій CARSkit [16].

Для подальшої роботи з актуальною версією CARSkit достатньо клонувати її git-репозиторій до локальної директорії на комп'ютері розробника цільової РС (див.рис. 4) і таким чином отримати доступ до всіх пакетів, файлів та архівів, головним з яких є виконуючий архів CARSkit.jar.

Таблиця 1 – Порівняння розглянутих проєктів РС

Назва	Підтримувані мови програмування			Наявність детальної документації	Можливість імплементації власних алгоритмів	Підтримка проекту розробниками на даний момент
	Java	Javascript	Python			
Racoon	–	+	+	–	–	–
LensKit	+	–	+	+	–	+
Surprise	–	–	+	+	+	+
CARSkit	+	–	–	+	+	+

```
$ git clone https://github.com/irecsys/CARSKit
Cloning into 'CARSKit'...
remote: Enumerating objects: 2896, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 2896 (delta 0), reused 1 (delta 0), pack-reused 2893
Receiving objects: 100% (2896/2896), 38.92 MiB | 4.00 MiB/s, done.
Resolving deltas: 100% (2018/2018), done.
```

Рис. 4. Клонування репозиторію кода проєкта CARSkit

Розробка архітектури прототипу рекомендаційної системи для конфігурування програмних компонентів систем «Розумний будинок. Опис вимог до програмного забезпечення прототипу РС. В роботі [17] представлена одна з можливих доменних моделей варіабельних компонентів умовної системи «Розумний будинок (РБ)», і і враховуючи її функції та властивості, можна запропонувати наступний сценарій управління конфігураціями її компонентів, який наведено на рис. 5 у вигляді UML-діаграми прецедентів.

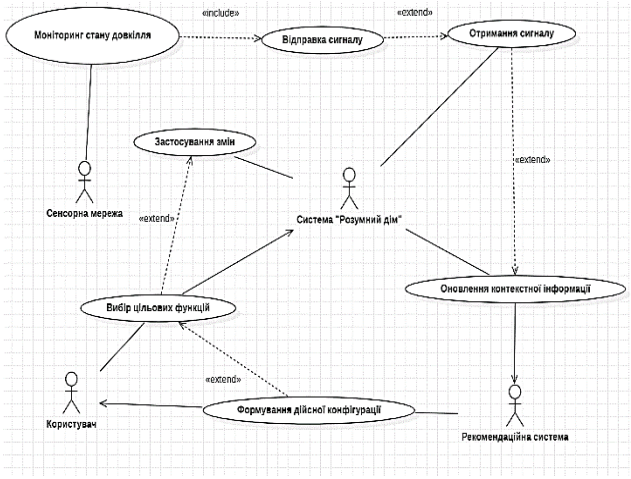

Рис. 5. Сценарій взаємодії РС, системи «РД» та користувачів

При цьому сенсорна мережа відслідковує зміни у зовнішньому середовищі і постійно передає інформацію до системи «РБ»; після аналізу вхідних сигналів система «РБ» передає їх безпосередньо до РС для відслідковування змін контекстної інформації, а потім алгоритм надання рекомендацій РС генерує для користувача певні конфігурації налаштувань функцій та властивостей «РБ»; і в кінцевому рахунку користувач обирає найбільш прийнятну для себе конфігурацію функції і встановлює її у системі «РБ» через відповідні інтерфейси (або цей етап може бути повністю автоматизованим, в залежності від рівня інтелектуальності пристроїв самої системи «РБ»).

Крім того, з метою можливості вдосконалення роботи РС може бути передбачено застосування в ній певної БД прецедентів, які в подальшому також будуть використовуватися для надання рекомендацій: див. відповідну UML-діаграму послідовностей на рис. 6.

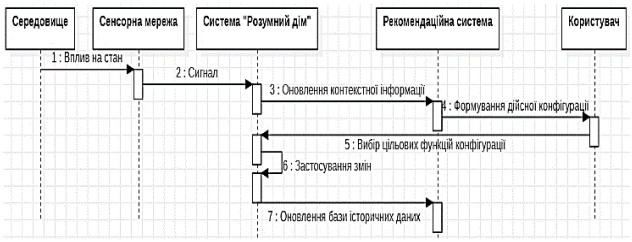

Рис. 6. Діаграма взаємодії РС, «РД» та користувача з використання БД

Маючи функціональні вимог до цільового прототипу РС, які представлені діаграмами на рис. 5 и рис. 6 відповідно, можна перейти до розробки її архітектури із використання інструментальних можливостей проекту CARSkit (див. вище).

Розробка архітектури прототипу РС. Запропонована архітектура для прототипу РС представлена на рис. 7 у вигляді UML діаграми пакетів (package diagram). Пакет Data включає в себе три модулі: Structure, Setting та Processor, які, відповідно, містять основні класи реалізації структур даних (матриці та тензори), класи для базових операцій обробки даних і класи для налаштувань певних системних параметрів.

Пакет Algorithms поділяється на два інших пакети: пакет Baseline, який містить основні рекомендаційні алгоритми спільної фільтрації (UserKNN, SVD++ та ін.) та алгоритми ранжування (RankSGD, RankALS тощо), а також пакет CARS, що надає контестно-залежні класи трансформації та адаптації основних алгоритмів [16]. Крім цих двох пакетів, функціональна архітектура запропонованої РС включає в себе також: пакет Generic, який містить класи, за інтерфейсами яких можуть бути реалізувані додаткові алгоритми рекомендацій; пакет Eva, що надає відповідні класи для розрахунку різноманітних метрик оцінки якості рекомендацій, і нарешті, пакет Main, в якому інкапсулюються класи для завантаження та ініціалізації поточного сеансу роботи з прототипом РС.

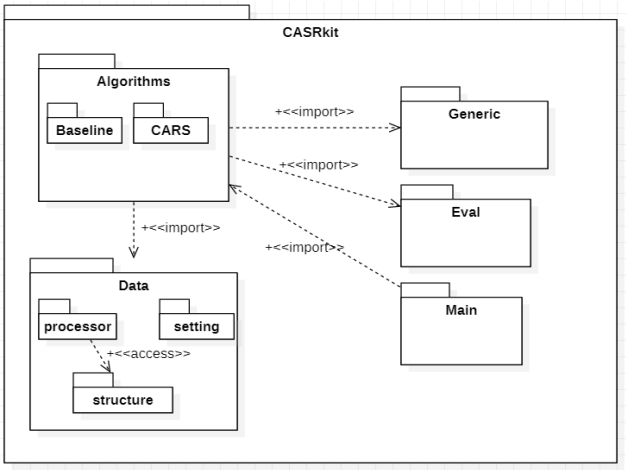

Рис. 7. Діаграма основних функціональних пакетів РС

Висновки та подальші дослідження. На основі проведеного аналізу існуючих підходів до вирішення задачі динамічного конфігурування програмних компонентів у лінійках програмних продуктів (ЛПП) було

показано, що для вирішення цієї проблеми доцільно використання методів та технологій побудови сучасних рекомендаційних систем (РС). Результатом подальшого огляду інтелектуальних методів розробки РС та дослідження функціональних можливостей деяких технологій реалізації РС у проектах з відкритим кодом, було запропоновано для подальшого використання саме в задачах когфігурування динамічних ЛПП обрати метод *N-вимірної контекстно-залежної тензорної факторизації* та інструментальний засіб *CARSkit.* Для практичної реалізації прототипу відповідної РС в роботі була розроблена її функціональна архітектуру, яка уможливлює автоматизацію конфігурування програмних компонентів у системах типу «Розумний будинок». В подальшому ця запропонована архітектура РС має бути програмно реалізована, для чого можуть бути використані функціональні можливості як самого засобу *CARSkit*, так і розроблені додаткові інструментальні засоби для реалізації алгоритмів обробки консолідованих даних на мові Python [18].

Особливу увагу в подальшому слід приділити плануванню обчислювальних експериментів з урахуванням специфіки ресурсів контекстних даних в системах типу «Розумний будинок» та вибору відповідних метрик для оцінки ефективності алгоритмів тензорної факторизації для прогнозування динамічних конфігурацій програмних компонентів в цих системах.

Список літератури

1. Reinhartz-Berger I., Sturm A., Clark T., Cohen S., Bettin J. *Domain Engineering: Product Lines, Languages, and Conceptual Models*. Heidelberg: Springer, 2013. 420 p.
2. Metzger A., Pohl K. Software Product Line Engineering and Variability Management: Achievements and Challenges. *Proceedings of FOSE'14 Conference*, May 31 – June 7, 2014. Hyderabad: India. P. 70–84.
3. Eleutério J. *A Comparative Study of Dynamic Software Product Line Solutions for Building Self-Adaptive Systems*. 2017. 30 p.
4. Sharifloo A. *Learning and Evolution in Dynamic Software Product Lines*. 2016. 8 p.
5. Gamzayev R. O., Karaçuha E., Tkachuk M. V. et al. An Approach to Assessment of Dynamic Software Variability in Mobile Applications Development. *Вісник Харківського національного університету імені В.Н. Каразіна, Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. 2018. № 40. С. 14–25.
6. Sohail S., Siddiqui J., Ali R. Classifications of Recommender Systems: A Review. *Journal of Engineering Science and Technology Review*. 2017. Vol.10(4). P.132–153.
7. Xiaoyuan Su, Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*. Volume 2009, January 2009. 4 p.
8. Lytvyn V., Vysotska V., Shatskykh V. et al. Design of A Recommendation System Based on Collaborative Filtering and Machine Learning Considering Personal Needs of The User. *Eastern-European Journal of Enterprise Technologies*. 2019. Vol. 4/2 (100). P. 6–28.
9. Баргесян А. А., Куприянов М. С. *Методы и модели анализа данных OLAP и Data Mining*. СПб. : БХВ-Петербург, 2004. 331 с.
10. Shani, G., Heckerman, D., Brafman, R. An MDP-Based Recommender System. *Journal of Machine Learning Research*. 2005. Vol. 6. P. 1265–1295.
11. Symeonidis, P., Zioupos, A. *Matrix and Tensor Factorization Techniques for Recommender Systems*. SpringerBriefs in Computer Science, 2016. 102 p.
12. Pereira J. et al. N-dimensional Tensor Factorization for Self-Configuration of Software Product Lines at Runtime Gothenburg. SPLC '18: *Proceedings of the 22nd International Systems and Software Product Line Conference*. 2018. Vol. 1. P. 87–97.
13. *Racoon Recommendation Engine Github* [Електронний ресурс]. URL:https://github.com/guymorita/recommendationRaccoon#recommendationraccoon-raccoon (дата звернення: 01.05.2021).
14. Michael D. Ekstrand. LensKit for Python. *Dept. of Computer Science, Boise State University*. Idaho, USA, September 3, 2020.
15. *Surprise documentation* [Електронний ресурс]. URL: https://surprise.readthedocs.io/en/stable/index.html (дата звернення: 12.06.2021).
16. Zheng, Y., Mobasher, B. CARSKit: A Java-Based Context-aware Recommendation Engine. *Proceedings of the 15th IEEE International Conference on Data Mining*, NJ, USA, 2015.
17. Гамзаєв Р. О., Ткачук М. В., Товстокоренко О. Т. Застосування методів доменного моделювання для підтримки варіабельності програмного забезпечення в розробці систем «Розумний будинок». *Інформаційні системи та технології: Матеріали статей 9-ї Міжнародної наук.-техн. конференції, Харків, 17–20 листопада 2020 року.* Харків: ХНУРЕ, 2020. С. 217–220.
18. *Python documentation* [Електронний ресурс]. URL: https://www.python.org/about/ (дата звернення 01.05.2021).

References (transliterated)

1. Reinhartz-Berger I., Sturm A., Clark T., Cohen S., Bettin J. *Domain Engineering: Product Lines, Languages, and Conceptual Models*. Heidelberg, Springer, 2013. 420 p.
2. Metzger A., Pohl K. Software Product Line Engineering and Variability Management: Achievements and Challenges. *Proceedings of FOSE'14 Conference*, May 31 – June 7, 2014, Hyderabad, India. pp. 70–84.
3. Eleutério J. *A Comparative Study of Dynamic Software Product Line Solutions for Building Self-Adaptive Systems*, 2017. 30 p.
4. Sharifloo A. *Learning and Evolution in Dynamic Software Product Lines*, 2016. 8 p.
5. Gamzayev R. O., Karaçuha E., Tkachuk M. V. et al. An Approach to Assessment of Dynamic Software Variability in Mobile Applications Development. *Visnyk Kharkivs'koho natsional'noho universytetu imeni V.N. Karazina, Seriya «Matematychne modelyuvannya. Informatsiyni tekhnolohiyi. Avtomatyzovani systemy upravlinnya»*. 2018, no 40, pp. 14–25.
6. Sohail S., Siddiqui J., Ali R. Classifications of Recommender Systems: A Review // *Journal of Engineering Science and Technology Review*. 2017, vol.10(4), pp.132–153.
7. Xiaoyuan Su, Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, Volume 2009, January 2009. 4 p.
8. Lytvyn V., Vysotska V., Shatskykh V. et al. Design of A Recommendation System Based on Collaborative Filtering and Machine Learning Considering Personal Needs of The User. *Eastern-European Journal of Enterprise Technologies*. 2019, vol. 4/2 (100), pp. 6–28.
9. Bargesyan A. A., Kupriyanov M. S. *Metody i modeli analiza dannykh OLAP i Data Mining*. SPb, BKhV-Peterburg, 2004. 331 p.
10. Shani, G., Heckerman, D., Brafman, R. An MDP-Based Recommender System. *Journal of Machine Learning Research*, 2005, vol. 6, pp. 1265–1295.
11. Symeonidis, P., Zioupos, A. *Matrix and Tensor Factorization Techniques for Recommender Systems*. SpringerBriefs in Computer Science, 2016. 102 p.
12. Pereira J. et al. N-dimensional Tensor Factorization for Self-Configuration of Software Product Lines at Runtime Gothenburg. SPLC '18: *Proceedings of the 22nd International Systems and Software Product Line Conference*. 2018, vol. 1, pp. 87–97.
13. *Racoon Recommendation Engine Github* [Електронний ресурс]. URL:https://github.com/guymorita/recommendationRaccoon#recommendationraccoon-raccoon (дата звернення: 12.06.2021).
14. Michael D. Ekstrand. LensKit for Python. *Dept. of Computer Science, Boise State University*. Idaho, USA, September 3, 2020..
15. *Surprise documentation* URL:https://surprise.readthedocs.io/en/stable/index.html (accessed 01.05.2021).
16. Zheng, Y., Mobasher, B. CARSKit: A Java-Based Context-aware Recommendation Engine. *Proceedings of the 15th IEEE International Conference on Data Mining*, NJ, USA, 2015.

17. Hamzayev R. O., Tkachuk M. V., Tovstokorenko O.T. Zastosuvannya metodiv domennoho modelyuvannya dlya pidtrymky variabel'nosti prohramnoho zabezpechennya v rozrobtsi system «Rozumnyy budynok». *Informatsiyni systemy ta tekhnolohiyi: Materialy statey 9--yi Mizhnarodnoyi nauk.-tekhn. konferentsiyi, Kharkiv, 17–20 lystopada 2020 roku*. Kharkiv, KhNURE Publ., 2020, pp. 217–220.
18. *Python documentation* URL: https://www.python.org/about/ (accessed 01.05.2021).

Надійшла (received) 11.05.2021

Відомості про автора / Сведения об авторе / About the Author

Гамзаєв Рустам Олександрович – кандидат технічних наук, доцент; докторант кафедри моделювання систем і технологій, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків–22, Україна, 61022; ORCID: https://orcid.org/0000–0002–2713–5664; e–mail: rustam.gamzayev@ karazin.ua

Ткачук Микола Вячеславович – доктор технічних наук, професор; завідувач кафедри моделювання систем і технологій, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків–22, Україна, 61022; ORCID: https://orcid.org/0000–0003–0852–1081; e–mail: mykola.tkachuk@karazin.ua

Гамзаев Рустам Александрович – кандидат технических наук, доцент; докторант кафедры моделирования систем и технологий, Харьковский национальный университет имени В.Н. Каразина, площадь Свободы, 6, Харьков, Украина, 61022; ORCID: https://orcid.org/0000–0002–2713–5664; e–mail: rustam.gamzayev@ karazin.ua

Ткачук Николай Вячеславович – доктор технических наук, профессор заведующий кафедрой моделирования систем и технологий, Харьковский национальный университет имени В.Н. Каразина, площадь Свободы, 4, Харьков, Украина, 61022; ORCID: https://orcid.org/0000–0003–0852–1081; e–mail: mykola.tkachuk@karazin.ua

Gamzayev Rustam Olexandrovich – PhD, associate professor; post–doctorate at the Department of Systems and Technologies Modeling, Kharkiv National University named after V.N. Karazina, Maidan Svobody, 6, Kharkiv, Ukraine, 61022; ORCID: https://orcid.org/0000–0002–2713–5664; e–mail: rustam.gamzayev@ karazin.ua

Tkachuk Mykola Vyacheslavovich – doctor of technical sciences, professor; head of the Department of Systems and Technologies Modeling, Kharkiv National University named after V.N. Karazina, Maidan Svobody, 6, Kharkiv, Ukraine, 61022; ORCID: https://orcid.org/0000–0003–0852–1081; e–mail: mykola.tkachuk@karazin.ua