УДК 004.652

І. І. Кушнірецька[1], О. І. Кушнірецька[1], А. Ю. Берко[2]
Національний університет "Львівська політехніка",
[1]кафедра інформаційних систем та мереж,
[2]кафедра загальної екології та екоінформаційних систем

ПРОЕКТУВАННЯ СИСТЕМИ ДИНАМІЧНОЇ ІНТЕГРАЦІЇ СЛАБОСТРУКТУРОВАНИХ ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЇ MASH-UP

Описано проектування системи динамічної інтеграції слабоструктурованих даних з використанням технології Mash-Up. Охарактеризовано функціональні вимоги до системи динамічної інтеграції даних на основі технології Mash-Up. Запропоновано алгоритм побудови онтологічної моделі всіх систем, що інтегруються та алгоритм отримання інформаційних ресурсів із системи, що інтегрується. Розглянуто архітектуру та принципи роботи системи Shares and Discounts Mashup динамічної інтеграції слабоструктурованих даних.

Ключові слова: динамічна інтеграція, технологія Mash-Up, проектування Mash-Up системи, архітектура Mash-Up системи.

This paper describes the designing of semi-structured data dynamic integration system using Mash-Up technology. The functional requirements to dynamic data integration system based on Mash-Up technology have been characterized. The algorithm of ontological models construction of all integrated systems and the algorithm of information resources getting from the integrated system has been proposed. The architecture and functional principles Shares and Discounts Mashup system of semi-structured data dynamic integration has been considered.

Key words: dynamic integration, Mash-Up technology, Mash-Up system designing, Mash-Up system architecture.

Вступ. Загальна постановка проблеми

Сьогодні, зважаючи на активний розвиток Web 2.0, вже накопичено та невпинно збільшується кількість інформаційних ресурсів різноманітного характеру і змісту. Постає проблема оперативної, динамічної інтеграції даних, виділяючи при цьому зміст інформації та зберігаючи семантику даних. Характер і складність можливих для використання методів вирішення цієї проблеми залежить насамперед від рівнів та способів інтеграції, властивостей окремих джерел даних і всієї сукупності джерел. Найперспективнішим сьогодні напрямом розроблення систем динамічної інтеграції даних вважається проектування систем, що працюють, використовуючи технологію Mash-Up динамічної інтеграції даних [1]. Сьогодні такі системи активно розвиваються і дають користувачам змогу динамічно збирати, використовувати і передавати Web-ресурси. Метою цих систем є створення нових корисних аплікацій із доступних інформаційних ресурсів.

Проектування системи динамічної інтеграції слабоструктурованих даних з використанням технології Mash-Up вимагає врахування доволі великої кількості різноманітних факторів, до яких, наприклад, належать: організація задання можливих запитів користувача, динамічність отримання колажів даних, формування відповіді на запит користувача, визначення спільних рис інформаційних ресурсів, особливості систем, що інтегруються, тощо. Відомими розробниками систем інтеграції даних на основі технології Mash-Up можна назвати Microsoft, Google, Yahoo, тощо. Зважаючи на активний розвиток Mash-Up систем та на їх величезні функціональні

можливості, особливо актуальною є задача розроблення підходів, технологій, архітектурних рішень і гнучких інструментальних засобів проектування систем динамічної інтеграції слабоструктурованих даних, що працюють на основі технології Mash-Up та надання рекомендацій опрацювання інформаційних ресурсів у таких системах.

Зв'язок висвітленої проблеми із важливими науковими та практичними завданнями

У зв'язку зі стрімким зростанням українських і світових інформаційних ресурсів методи динамічної інтеграції інформації на основі технології Mash-Up та опрацювання інформаційних ресурсів у системах на основі цієї технології перебувають у фокусі дослідників федеративних середовищ і представляють науково-практичний інтерес для розробників сучасних розподілених інтелектуальних інформаційних систем, але численні проблеми все ж залишаються невирішеними. На жаль, поки немає чітко визначених рекомендацій до побудови систем динамічної інтеграції даних на основі технології Mash-Up. Такі проекти реалізуються переважно завдяки власним ідеям та рішенням.

Аналіз останніх досліджень та публікацій

Розроблення методів інтеграції інформаційних ресурсів – одна з найбільш актуальних проблем в галузі інформаційних систем. Особливо велику увагу вона стала привертати із розвитком систем інтеграції даних, що працюють, використовуючи технологію Mash-Up [1]. Mash-Up системи являють собою новий тип інтерактивних веб-додатків, об'єднуючи вміст з декількох сервісів або ресурсів в новий сервіс або ресурс даних. Появу Mash-Up систем спричинило розроблення додатків використання AJAX1 технології. Постійно збільшується кількість веб-інтерфейсів для вилучення вмісту з популярних веб-сайтів. ProgrammableWeb2 вже у 2007 році налічував понад 1700 Mash-Up систем і близько трьох нових Mash-Up систем додаються кожного дня. Багато Mash-Up систем анотують інформацію з географічних даних для візуалізації інформації на карті, наприклад, розташування обраних ресторанів або об'єктів нерухомості.

Використання технології Mash-Up забезпечує створення нових споживчих сервісів [2]. Крім того, на основі кожного Mash-Up додатка можна створити новий оригінальний сервіс, що надасть безліч нових можливостей для створення наступних Mash-Up систем. За допомогою технології Mash-Up можна створювати персоналізовані інформаційні сервіси на базі існуючих. Наприклад, технології Mash-Up додаток, який надає користувачеві доступ до декількох інтернет-магазинів, може пропонувати більш придатні товари на основі його попередніх дій або інформації про користувача в соціальних мережах. Mash-Up системи новин можуть дозволити користувачеві вибирати тільки ті теми, які його цікавлять.

Інтеграція контенту в Mash-Up системах, як правило, динамічна, тобто вона відбувається під час виконання, на основі введених користувачем даних. Для досягнення мінімального часу виконання більшість Mash-Up систем підтримують тільки достатньо прості види інтеграції даних. Наприклад, вони часто використовують стандартизовані ідентифікатори об'єктів (наприклад, широта/довгота географічних позицій, або унікальні номера продукту, такі як EAN або ISBN) для легкого взаємозв'язку різних джерел або послуг. Запит доступу до джерел даних або пошукових систем є, як правило, на основі ключових слів, тегів або категорій імен, але без обширної пост-обробки, для перегляду результатів з різних джерел. Отже, сучасне використання Mash-Up технології потребує покращеної підтримки для динамічної інтеграції даних в гетерогенних об'єктах даних.

Як правило, схема Mash-Up інтеграції даних така: виконання завдання, визначеного користувачем, перетворення даних до рівня сервісу та представлення готового колажу інформаційних ресурсів (рис. 1).

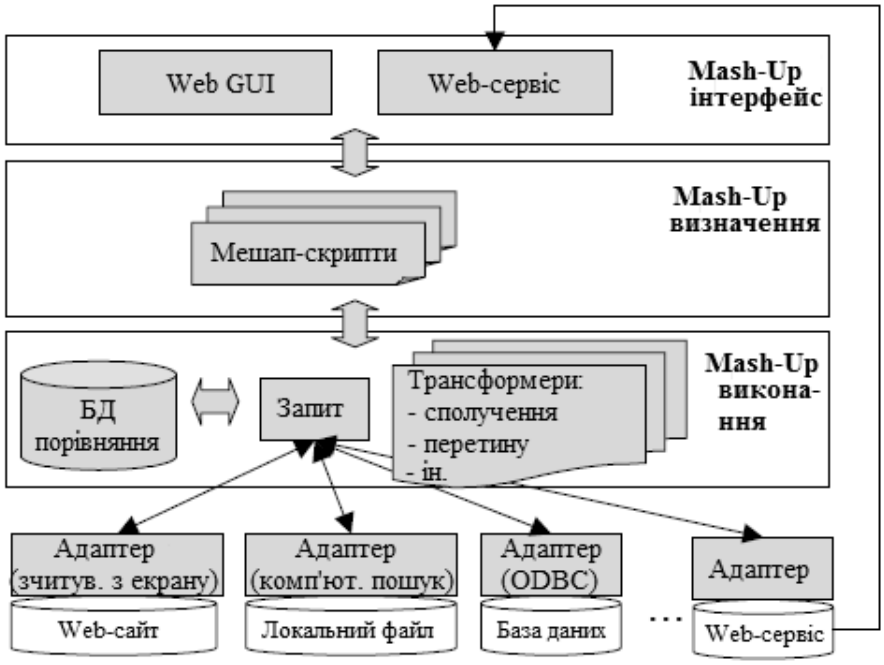

Рис. 1. Схема Mash-Up інтеграції даних[2]

Розглянемо деякі приклади сучасних найвідоміших Mash-Up систем.

IFTTT [3] (“if this, then that” – “якщо це, тоді то”) – це Mash-Up сервіс, який дозволяє користувачам підключатися до різних веб-додатків (наприклад, Face book, Evernote, LinkedIn, Dropbox і т.д.) за допомогою простих умовних операторів, відомих як “рецепти” і створювати просту автоматизовану послідовність операцій, яка запускається при виконанні певної дії. IFTTT дозволяє користувачам створювати і обмінюватися “рецептами”, які відповідають судженню: “якщо це, тоді те” “це” частина рецепта, яка називається тригером. Доволі простий у користуванні та складається всього із трьох закладок:

– Tasks (Завдання) – це список ваших активних завдань.

– Recipes (Рецепти) – це список найпопулярніших завдань, які ви можете використовувати як свої, тобто це щось схоже на заготовки завдань.

– Channels (Канали) – це список підтримуваних сервісів, на момент дослідження їх 54 (Наприклад, Twitter, Evernote, Google Calendar, LinkedIn, Google RSS Reader, Gmail, WordPress і ін.).

Google Alerts [4] – сервіс від пошукового гіганта. Ця система працює на основі ідеї моніторингу результатів пошукового запиту відповідно до часових змін. Фактично можна налаштувати “Алерти” на появу нових результатів за запитом. Система вміє фільтрувати результати і відбирати найбільш релевантні дані.

У списку параметрів сповіщення:

– сам запит (підтримується також синтаксис пошукових запитів Google);

– тип запиту (все, новини, блоги, відео, обговорення, книги);

– частота оновлення повідомлень (у режимі реального часу, раз на день, раз на тиждень);

– фільтр кращих результатів або всіх;

– відправка результатів на e-mail або у вигляді RSS-фіду.

Wappwolf [5] – Mash-Up сервіс для роботи з файлами. Він схожий за ідеєю з IFTTT, але з ухилом на обробку файлів. Єдина подія тут – додавання до папки хмарного сховища (підтримуються Dropbox, Google Drive, SkyDrive, Box) файла, а ось дій тут може бути багато:

– синхронізація з іншими хмарними сховищами Box, SkyDrive, Google Drive, а також з FTP-сервером.

– різноманіття простих операцій для зображень: зміна розміру, переведення у відтінки сірого, поворот, додавання водяного знака.

– операції для звукових файлів: конвертація в інший формат.

– операції для текстових файлів: конвертація в PDF, формати електронних книг, завантаження на Kindle, роздруківка через хмарний принтер Google.

– операції для всіх типів файлів: додавання в архів до архіву, перейменування, шифрування/дешифрування.

Розглядаючи принципи роботи Mash-Up систем, їх можна поділити на такі основні групи:

– формати і доступ до даних;

– внутрішня модель даних;

– робота з вхідним та вихідним потоком інформації (зіставлення даних, забезпечення функціональних можливостей операторів потоків даних, оновлення даних тощо);

– функціональні можливості (розширюваність, розповсюдження тощо).

У Mash-Up додатку, користувач може об'єднувати дані, описані в різних форматах. Наприклад, веб-формат використовується для публікації часто оновлюваної інформації, наприклад, записів у блогах, сайтах новин тощо; табличний формат підходить для опису моделей даних на основі таблиць, таких як CSV-файли або електронні таблиці; формат на основі розмітки (наприклад, HTML і XML) зазвичай часто використовується для публікації даних; мультимедійний контент, такий як відео-, аудіо- та зображення, стають все поширенішими. Ці типи даних можуть бути доступні користувачеві з різних джерел даних. Найпоширенішими джерелами даних можуть бути традиційні системи керування базами даних, локальні файли, які доступні у файловій системі власника, веб-сторіноки, веб-сервіси і веб-додатки. Для полегшення вилучення веб-даних провайдери часто отримують їх вміст через веб-API.

Мета Mash-Up додатка – об'єднання різних ресурсів (даних у нашому випадку) для створення нового додатка. Ці ресурси приходять зазвичай з різних джерел, в різних форматах і транспортують різну семантику. Для підтримання своєї роботи кожен Mashup-інструмент використовує свою внутрішню модель даних. Внутрішня модель даних є єдиною глобальною схемою, що являє собою уніфіковане представлення даних. Внутрішня модель даних Mash-Up системи може бути або у вигляді графу, або на основі об'єктів.

Графічно-орієнтована модель: на основі графу ми будуємо модель, основану на мові XML і всьому тому, що безпосередньо пов’язано з XML. Тобто модель може містити чистий XML, RDF, RSS і т.д. Більшість Mashup-додатків використовують модель у вигляді графу як внутрішню модель даних. Це, звичайно, мотивується тим, що більшість сьогоднішніх даних, в основному в Інтернеті, таких як RSS-канали, є доступні для цього формату моделі даних. Тобто всі дані, які використовуються Mash-Up інструментами в цій категорії, трансформують вхідні дані до вигляду XML перед їх обробкою.

Об'єктно-орієнтована модель: в цьому випадку внутрішня модель даних формується у вигляді об'єктів (у класичному сенсі об'єктно-орієнтоване програмування). Об'єкт є екземпляром класу, який визначає особливості елемента, зокрема характеристик елемента (його атрибутів, полів або властивостей) і поведінки елемента (методи). Слід зазначити, що в цьому випадку немає жодного явного перетворення, виконаного системою, як у випадку графічно-орієнтованої моделі, але програміст повинен визначити структуру об'єкта відповідно до її даних.

Робота з вхідним та вихідним потоком інформації у Mash-Up системі динамічної інтеграції даних характеризується набором таких характеристик:

1. Зіставлення даних. Щоб створити екземпляр внутрішньої моделі даних із зовнішнього джерела даних, Mash-Up-додаток мусить забезпечити стратегію за попередньо заданими відповідностями між внутрішньою моделлю даних і бажаними джерелами даних. Цього досягають відображенням даних. Відображення даних – це процес, необхідний для визначення відповідності між елементами моделі даних джерела і внутрішньою моделлю даних. Відображення даних може бути:

– ручне, коли всі відповідності між внутрішньою моделлю даних і моделлю даних джерела даних вручну вказує, одну за однією, розробник програми. У цьому випадку система повинна потім надати деякі засоби для користувача, щоб розробити трансформацію.

– напівавтоматичне, коли система використовує метадані (наприклад, поля імена і типи) для пропозицій певних можливих конфігурацій відображення. Звичайно, користувач повинен підтвердити ці пропозиції і зазвичай виправити деякі з них.

– автоматичне, коли всі відповідності між двома моделями даних автоматично генеруються без втручання користувача. Це складна проблема в області інтеграції даних. Оскільки площа Mash-Up знаходиться в "початковій стадії", цей тип відображення поки ще повністю не підтримується будь-яким інструментом Mash-Up. Слід зазначити, що процес відображення даних може зажадати проміжного кроку, тобто крок загортання для перетворення початкового формату на внутрішній формат, наприклад, від CSV для XML. Відображення в наявних сьогодні засобах Mash-Up робиться тільки на рівні схеми даних, а семантичний аспект не розглядається й досі.

2. Оператори потоків даних дають змогу виконувати операції або на структурі даних (аналогічно до мови визначення даних/операторів у реляційній моделі), або на самих даних (змісті) (за аналогією до мови обробки даних/операторів у реляційній моделі). Розглянемо оператори і вирази мов програмування, що надаються інструментами для обробки та інтеграції даних. Оператори потоку даних дають змогу:

– реструктурувати схему вхідних даних, наприклад, додавати нові елементи, нові атрибути до елементів;

– виконувати перетворення на наборах даних, таких як: витягнення певної частини інформації, поєднання конкретних елементів, які задовольняють задану умову, зміна значень деяких елементів;

– будувати новий набір на основі інших наборів даних, використовуючи операції перетину, об'єднання або різниці над даними (за аналогією як у базах даних).

Реалізація операторів потоку даних значно залежить від головної мети інструменту, тобто інтеграції чи візуалізації. Деякі оператори реалізуються в різних системах по-різному, і надана інтерпретація результату також відрізняється. Основні оператори, орієнтовані на інтеграцію даних, реалізовано в таких операціях: Union (об'єднання), Join (з'єднання), Filter (фільтрування) і Sort (сортування) (таблиця). Докладніше описано наведені оператори в [6].

Таблиця 1

Загальні основні оператори Mash-Up систем

Оператор	Опис
Union	Об'єднує два набори даних в один. Результуючий набір містить всі дані з усіх наборів, що беруть участь в об'єднанні.
Join	З'єднує різні набори даних відповідно до умови.
Filter	Вибирає конкретну підмножину (сутностей і атрибутів) з вихідної підмножини.
Sort	Представляє вибрані дані в певному порядку.

3. Оновлення даних. У деяких випадках, наприклад, фондового ринку, дані зазвичай динамічно генеруються і постійно оновлюються. Багато стратегічних рішень, особливо на підприємствах, як правило, приймають відповідно до останніх статусу/значення даних. Саме тоді важливо, щоб система поширювала оновлені джерела даних для відповідного користувача(ів). Існують дві основні стратегії, що стосуються статусу даних у джерелі, залежно від мети користувача:

– стратегія витягнення даних (pull strategy) основана на формуванні частих і повторюваних запитів від клієнта, акцент робиться на періодичності і частоті витягнення даних. Ці періодичність і частоту витягнення даних переважно вибирають якомога нижчими, ніж середня частота оновлення даних у джерелі. Свіжість даних залежить від періодичності і частоти витягнення даних, тобто що вищою є періодичність і частота витягнення даних, то свіжішими будуть дані і навпаки. Одним з головних недоліків високої частоти оновлення даних є те, що на сервері можуть бути згенеровані непотрібні запити.

– стратегія вміщення даних (push strategy), за якої клієнт не посилає запити, але повинен зареєструватися на сервері. Реєстрація необхідна, щоб вказати/ідентифікувати дані, що представляють інтерес. Отже, сервер посилає дані клієнту, коли відбувається зміна на стороні сервера. Основним недоліком цієї моделі є те, що клієнт може бути зайнятий виконанням інших завдань, коли відправляється інформація, і тому існує можливість затримки в момент обробки інформації.

Ще одним важливим параметром для оновлення даних є те, як система управляє інтервалом витягнення даних. Ми можемо визначити дві можливі стратегії для вирішення даної проблеми: глобальна стратегія і локальна стратегія.

Для глобальної стратегії інтервал витягнення даних встановлюється для всієї програми. Це передбачає, що джерела даних мають такий самий інтервал оновлення даних. Тобто, до джерел даних посилається запит у той самий проміжок часу, що відповідає одному з інструментів Mash-Up системи. В результаті користувач підтримує кращий зв'язок з деякими джерелами (ті, що мають низький інтервал оновлення порівняно з іншими), ніж з іншими (ті, що мають високий інтервал оновлення порівняно з іншими).

За локальною стратегією для кожного джерела даних є свій власний інтервал оновлення. Цей інтервал оновлення, як передбачається, відповідає одному з джерел даних. У результаті кращий зв'язок потрібно зберігати із кожним джерелом даних. На те, щоб встановити інтервал оновлення, кожна компонента джерела має параметр "інтервал оновлення". Після перевищення часу дані зі вказаного URL перезавантажуються.

4. Вихідні дані Mash-Up систем. Розглянемо вихідні дані з погляду вимірювання їх розміру, оскільки користувач може бути зацікавлений в експорті даних Mash-Up результату (потоку даних) до приведення їх до іншого формату, щоб використовувати їх для подальшої обробки, а не просто візуалізації. Тобто, ми можемо виділити дві основні категорії вихідних даних: людино-орієнтовані вихідні дані і вихідні дані, орієнтовані на обробку.

Людино-орієнтовані вихідні дані призначені для людської інтерпретації, наприклад візуалізація на карті, на HTML сторінці тощо. Тобто для цієї категорії вихідні дані можна розглядати як "кінцевий продукт" всього процесу.

Вихідні дані, орієнтовані на обробку, переважно орієнтовані на використання машинами. Це потрібно у випадку, коли необхідно використовувати дані, що підлягають подальшій обробці, наприклад, для вилучення знань. Слід зазначити, що ця категорія може на деякій стадії містити першу категорію, наприклад, вихідні дані у вигляді RSS можуть бути як візуалізовані на HTML сторінці, так і використані в інших додатках для інших задач обробки даних.

Представлення вихідних даних залежить від головної мети системи. Одні системи можуть забезпечувати багаті динамічні візуалізації колажів додатків, а інші прагнуть до агрегації і маніпулювання даними, які можна використовувати з іншими додатками. Вихідні дані експортується в RSS, Atom або XML інтерпретацію додаванням інформації заголовка і вмісту в ньому конкретної інформації, потім інформація перетворюється на кортежі послідовностей у зазначений тип вихідної подачі інформації.

Розглянемо такі функціональні можливості систем динамічної інтеграції даних, як розширюваність та розповсюдження.

Розширюваність визначає здатність системи підтримувати додаткові, зазвичай, визначені користувачем, функціональні можливості. Може бути два можливі способи визначити і використовувати ці функціональні можливості. Функціональність може бути або вбудована всередині інструменту, тобто відповідний код цієї функціональності додається до інструменту з використанням конкретної мови програмування, або зовнішньою, тобто викликати відповідну послугу, яка містить таку функцію. Розширення переважно залежить від архітектури і напрямленості інструменту. У деяких випадках розширення може бути зроблено вбудовуванням коду необхідної функціональності в інструменті; в інших випадках сервіси викликаються як сервіси REST, SOAP тощо. Крім того, ця функція по-різному використовується різними інструментами. Справді, в одному випадку, додана функція/сервіс спільна з усією функціональністю, яка використовується інструментом. В іншому випадку розширення відображається лише для конкретного користувача.

Mash-Up додатки основані на технології Web 2.0, де люди можуть напрочуд легко створювати, коментувати інформацію і обмінюватися нею. Питання безпеки та конфіденційності для обміну в цьому величезному мережевому середовищі, безумовно, є великою проблемою. Це завдання стає ще важче вирішити, поки є своєрідна цільова публічність з Web 2.0, а більшість користувачів зовсім не фахівці в галузі обчислювальної техніки та безпеки. Тому, визначаючи модальність, яку система пропонує для спільного використання ресурсів, вона не може гарантувати конфіденційність і безпеку створених Mash-Up додатків. Це доволі складна область у створенні Mash-Up систем, і багато роботи ще належить виконати. Крім того, вирішуючи проблему конфіденційності і безпеки Mashup-додатків, потрібно знайти відповіді на такі запитання:

1) що таке розповсюдження даних у Mash-Up системі?
2) як воно відбувається?
3) хто ті користувачі, які братимуть участь в процесі?
4) для кого представлений ресурс може бути корисним?

Представлення ресурсу може бути різним, наприклад: тільки для читання (користувач може читати всі дані, але не може їх змінювати чи дописувати щось), для читання/запису (користувач може читати і змінювати дані), без доступу (користувач не може читати чи змінювати дані). Користувачами, які можуть брати участь в процесі, можуть бути, наприклад, будь-хто, певна група людей або конкретний користувач. Слід зазначити, що для кожного користувача може бути різна політика доступу до розповсюдження інформації. Це наприклад:

– повний доступ, тобто доступ до читання вихідного коду, даних і виведених результатів;
– частковий доступ, тобто доступ на читання вихідного коду;
– без доступу, де Mash-Up дані не розповсюджуються.

Виділення проблем

Процес створення системи динамічної інтеграції даних на основі технології Mash-Up вимагає детального аналізу всіх важливих характеристик проектування і функціонування таких систем, а також попереднього розроблення рекомендацій проектування системи такого типу та детального опису опрацювання інформаційних ресурсів у такій системі, використовуючи при цьому правильну методологію. Отже, актуальною є задача розроблення нових підходів, технологій, архітектурних рішень і гнучких інструментальних засобів для динамічної інтеграції слабоструктурованих даних у web-середовищі. Тому опис проектування системи динамічної інтеграції слабоструктурованих даних з використанням технології Mash-Up представляється перспективним і конструктивним для вирішення поставлених проблем.

Формулювання мети

Мета роботи полягає у розробленні рекомендацій для створення системи динамічної інтеграції даних на основі технології Mash-Up та семантичного опрацювання інформаційних ресурсів у такій системі.

Аналіз отриманих наукових результатів

У [7] на основі аналізу діяльності Mash-Up систем розрізняють такі стани діяльності:

1. Реєстрація. За успішної реєстрації – перехід до другого стану, за неуспішної – повернення знову до початку реєстрації.

2. Авторизація: якщо все пройшло успішно, рухаємося далі, якщо ні – повертаємося до початку авторизації.

3. Формування завдання. Якщо завдання сформовано відповідно до правил системи, рухаємося далі, якщо ні – повертаємося до початку третього стану.

4. Формування джерел для Mash-Up.

5. Пошук потрібної інформації у відібраних джерелах. Якщо результати пошуку задовільні – йдемо далі, якщо ні – повертаємося назад до пошуку.

6. Витягнення інформації і перехід до наступного стану.

7. Зберігання отриманої інформації у вигляді сервісу.

8. Візуальне представлення готового Mash-Up.

Найважливішими станами при роботі Mash-Up системи, згідно з [7], є пошук потрібної інформації (п’ятий стан), витягнення знайденої інформації (шостий стан) та її зберігання у вигляді сервісу (сьомий стан). Для покращення та вдосконалення отримання результату для роботи сьомого стану в [7] запропоновано покрокову процедуру визначення структури і змісту вхідних інформаційних ресурсів, яку можна інтерпретувати як метод визначення структури і змісту отриманої вхідної інформації. Застосуванням цього методу можна підвищити якісні показники результату п’ятого стану. Оскільки кожен наступний стан залежить від попереднього – це підвищить продуктивність наступних двох важливих станів діяльності. Тому при проектування системи динамічної інтеграції даних на основі технології Mash-Up було використано згаданий метод. Взаємозв'язки між функціональними та операційними вимогами до систем інтеграції даних та різними областями архітектури – такими, як прикладні системи і технологічна архітектура – показано на рис. 2.

Функціональні вимоги до прикладної системи описують ту цінність, яку представляє система з погляду реалізації функцій організації (бізнес-цінність). Архітектура додатків, по суті, є архітектурою всіх автоматизованих сервісів, які забезпечують і реалізують такі функціональні вимоги, включаючи інтерфейси до бізнес-програм та інших прикладних систем. Вона описує структуру додатків і те, як ця структура реалізує функціональні вимоги організації.

Операційні (або експлуатаційні) вимоги до програмної системи специфікують такі аспекти, як надійність, керованість, продуктивність, безпека, сумісність. І це далеко не повний список. Прикладами операційних вимог є можливість доступу до системи тільки авторизованих користувачів, рівень доступності прикладної системи 99, 99 % часу тощо.

Технологічна архітектура є архітектурою інфраструктури апаратного та програмного забезпечення, яка забезпечує роботу прикладних систем і виконання операційних (не функціональних) вимог, що представляються до архітектури прикладних систем та інформації. Вона описує структуру і взаємозв'язки між використовуваними технологіями, і те, як ці технології забезпечують виконання операційних вимог організації.

Отже, функціональні вимоги забезпечуються архітектурою додатків, операційні вимоги забезпечуються технологічною архітектурою.

Хоча, звичайно, слід зробити одне зауваження. Вдала технологічна архітектура може забезпечувати безпеку, доступність, надійність та інші операційні вимоги, але якщо систему спроектовано так, що вона не використовує переваг технологічної архітектури, вона все одно не функціонуватиме, і її буде складно впроваджувати і супроводжувати. Аналогічно, належно спроектована структура прикладної системи, яка точно відповідає вимогам бізнес-процесів і зібрана з багаторазово використовуваних компонент із застосуванням найсучасніших технологій, може не відповідати реальній конфігурації використовуваного апаратного та системного програмного забезпечення. Наприклад, наявні сервери можуть не підтримувати роботу компонент системи, конфігурація і топологія мережі – не гарантувати потреби в потоках інформації. Це показує, що все-таки є істотний взаємозв'язок між архітектурою додатків і технологічною архітектурою: вдала технологічна архітектура повинна бути побудована з урахуванням підтримки прикладних систем, що виконують важливу роль у роботі організації. Отже, архітектура додатків повинна ефективно використовувати технологічну архітектуру, щоб забезпечити належний рівень відповідності всім операційним вимогам.

Згідно з ISO 15926-7 [8], якщо сервіс забезпечує процедуру опрацювання та обміну даними, він повинен виконувати такі завдання:

– приймати потік даних, що архівуються у процесі інформаційного обміну між інформаційними системами та/або з федеральними інформаційними ресурсами в режимі реального часу;

– розархівовувати отримані дані;

– визначати приналежність даних, аналізуючи спільні ознаки;

– опрацьовувати розархівовані дані формату xml і розміщувати їх у реляційній базі даних веб-сервісів;

– логувати інформацію на етапі передавання, розархівування, опрацювання та збереження.

– забезпечувати фіксацію часу передавання, цілісності та автентичності даних, зазначення їх авторства та можливості надання відомостей, що дають змогу простежити історію руху даних.

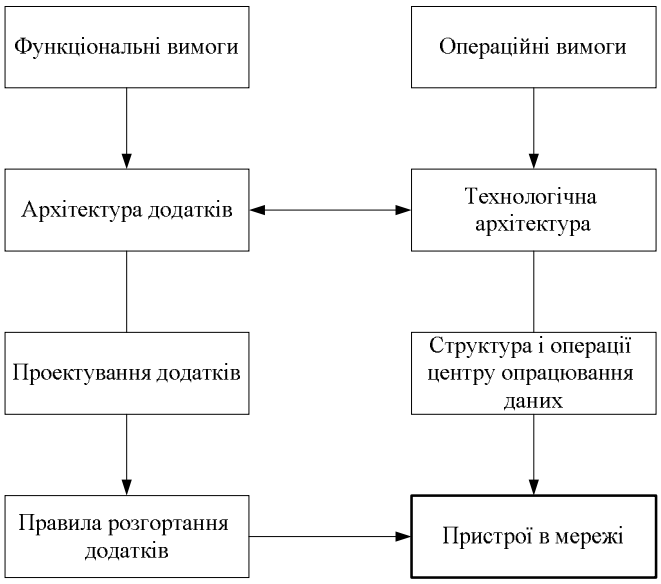

Рис. 2. Взаємозв'язки функціональних і операційних вимог з архітектурою додатків і технологічною архітектурою

Було розроблено функціональні вимоги до системи динамічної інтеграції даних на основі технології Mash-Up, які зображено на рис. 3.

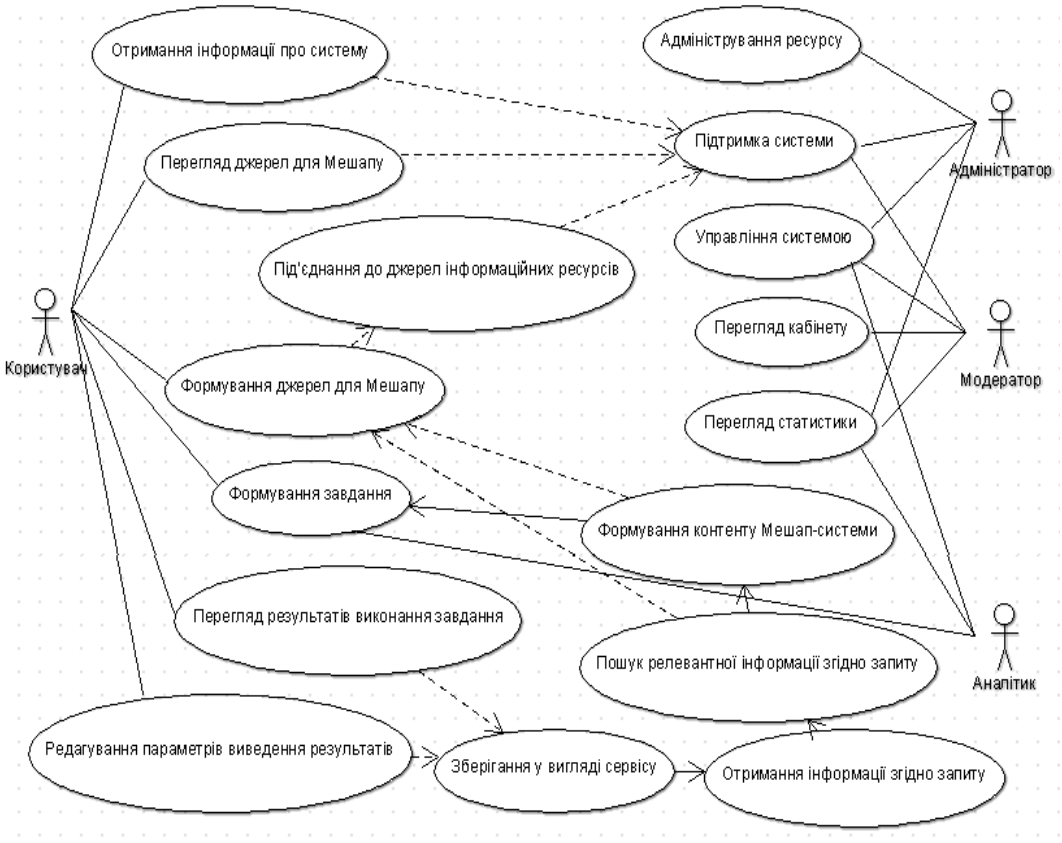

Рис. 3. Функціональні вимоги до системи динамічної інтеграції даних на основі технології Mash-Up

Використовуючи метод визначення структури і змісту отриманої вхідної інформації [14], спроектовано Shares and Discounts Mashup – систему пошуку знижки на купівлю товару, отримання послуги тощо. Система автоматизовано відображає інформацію, що зберігається в гетерогенних, інформаційних web-системах, що інтегруються, в онтологічну модель, яка пізніше

використовується для пошуку інформаційних ресурсів згідно з запитом користувача. Реалізація системи динамічної інтеграції слабоструктурованих даних у web-системах являє собою сукупність завершених модулів, які можна використати для побудови інших систем. Функціональна декомпозиція програми визначає функції як абстрактні операції в термінах задачі, а не в деталях їх реалізації. Згідно з методом визначення структури і змісту отриманої вхідної інформації [7] було створено алгоритм побудови онтологічної моделі всіх систем, що інтегруються (рис. 4). Будуючи будь-який алгоритм, першочерговим завданням є визначення вхідних та вихідних даних. Вхідними даними для алгоритму отримання структури даних системи як онтологічної моделі є: структурні схеми баз даних систем, що інтегруються; онтологія предметної області. Вихідними даними є загальна онтологічна модель, яка описує структуру інтегрованих систем у межах її предметної області і зв'язки між елементами різних систем. Таку модель моделюють засобами мови RDF, її розширенням RDFs та із застосуванням мови OWL.

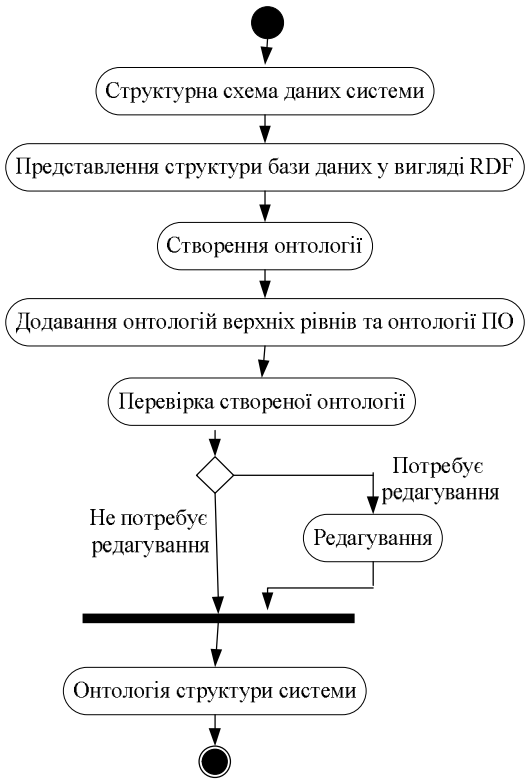

Рис. 4. Схема алгоритму побудови онтологічної моделі всіх систем, що інтегруються

Опишемо роботу алгоритму побудови онтологічної моделі всіх систем, що інтегруються. І для цього введемо деякі позначення понять.

Нехай ми маємо деяку схему бази даних системи: $S = \{T_1,...,T_m\}$, де $T_1,...,T_m$ – таблиці схеми бази даних системи S. $T_i = \{A_1,...,A_k\}$, $i = \overline{1,n}$, де $A_1,...,A_k$ – атрибути таблиць схеми бази даних. $R = \{R_1,...,R_z\}$ – зв'язки між концептами онтології.

Алгоритм побудови онтологічної моделі всіх систем, що інтегруються містить в собі 6 основних кроків, а саме:

1. Представлення структури бази даних у вигляді RDF, тобто послідовне відображення схеми S в RDF формат.

$$T_i \to T(RDF)_i,\ A_j \to A(RDF)_j,\ i = \overline{1,n},\ j = \overline{1,k}, \quad (1)$$

де $T(RDF)_i$ – концепти онтології, описані за допомогою RDF; $A(RDF)_j$ – властивості концептів в онтології.

2. Додавання семантичних властивостей та створення онтології. Цей крок реалізується за допомогою використання процедури визначення спільних рис елементів бази даних і додавання зв'язків між ними.

3. Додавання онтологій верхніх рівнів та онтології предметної області. Реалізуємо цей крок за допомогою мови OWL, використовуючи команду owl: import. Завдяки правилу транзитивності в RDF, додаткові онтології розширюють предметні області і додають нові концепти і властивості.

4. Перевірка створеної онтології. Цей крок реалізується перевіркою і аналізом витягнутої онтології на "зв'язність", тобто ми перевіряємо, чи не бракує ніде семантичних зв'язків. Якщо так, тоді переходимо до п'ятого кроку, якщо ні – переходимо до шостого кроку.

5. Редагування витягнутої онтології за допомогою редактора онтології (Protégé) і додавання зв'язків між концептами. Далі повертаємося до кроку 4.

Зберігання отриманої загальної онтології структури системи у файл чи сховище метаданих у форматі RDF. Алгоритм отримання інформаційних ресурсів із системи, що інтегрується (рис. 5), передбачає використання раніше отриманої глобальної онтологічної метамоделі систем, що інтегруються, що містить онтології верхнього рівня і онтологію предметної області.

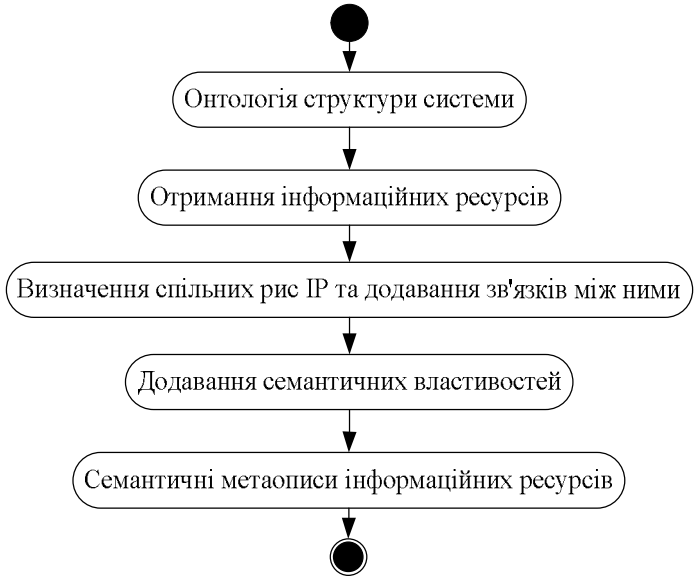

Рис. 5. Схема алгоритму отримання інформаційних ресурсів із системи, що інтегрується

Вхідними даними алгоритму є:

– інформація про збережені у базах даних ресурси;
– глобальна онтологічна метамодель.

Вихідними даними як результатом роботи алгоритму є метамодель, що об'єднує понятійну і змістовну частини онтології. Отже, в такій онтології об'єднується як інформація про структуру інтегрованих систем, так і метадані збережених в них об'єктів, описані термінами з понятійної частини. Отримана метамодель може бути використана як єдиний інтерфейс для семантичної інтеграції даних розподілених систем та забезпечення їх інтероперабельності.

Робота алгоритму отримання інформаційних ресурсів із системи, що інтегрується, складається із таких кроків:

1. Отримання інформації про збережені інформаційні ресурси із системи, що інтегрується, з використанням глобальної онтологічної мета-моделі. Тобто отримуємо кожен кортеж із кожної таблиці певної схеми бази даних.

2. Визначення спільних рис отриманих кортежів таблиць інформаційних ресурсів і додавання семантичних зв'язків між ними.

3. Додавання семантичних властивостей, використовуючи семантичний аналізатор, який працює на основі дескриптових логік та імпортованій онтології.

Результатом є онтологія метаданих інформаційних ресурсів і структури системи, що зберігається у сховище метаданих у форматі RDF.

Архітектура інформаційної системи – концепція, що визначає модель, структуру, виконувані функції і взаємозв'язок компонентів інформаційної системи. Архітектуру спроектованої системи Shares and Discounts Mashup показано на рис. 6.

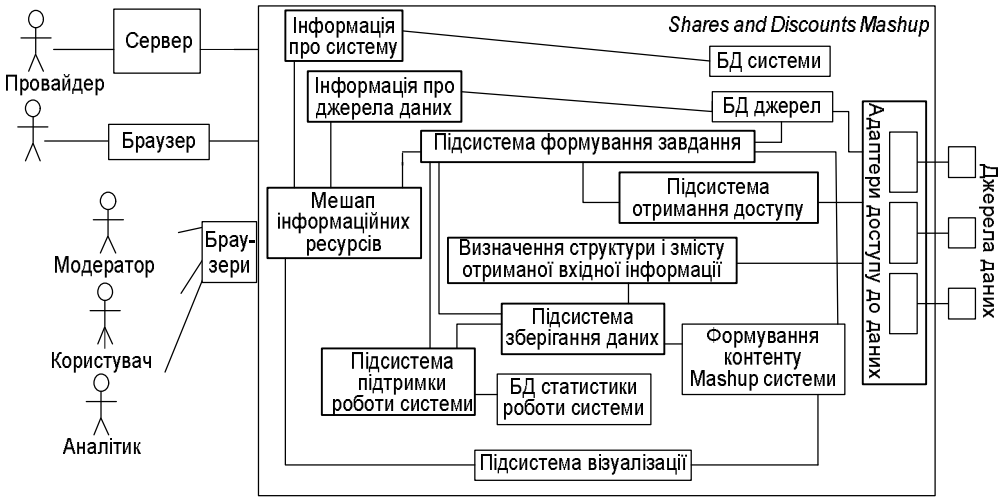

Рис. 6. Архітектура системи динамічної інтеграції слабоструктурованих даних Shares and Discounts Mashup

Вхідними даними системи динамічної інтеграції даних на основі технології Mash-Up Shares and Discounts Mashup є джерела інформаційних ресурсів, до яких має доступ система та запит користувача. Користувач задає запит і отримує відповідь на нього у вигляді колажу із інформаційних ресурсів доступних джерел, які відповідають заданому запиту.

При проектуванні інтерфейсу системи було враховано ергономічні характеристики, простоту та компактність відображення інформації на екрані, зручність доступу до основних органів керування режимами роботи системи, забезпечення надійності роботи з системою. Інтерфейс є доволі простим і зрозумілим для будь-якого користувача. Навіть без детальних інструкцій кожен може зрозуміти, що робить система Shares and Discounts Mashup. Все виконано в тонах, які є не виснажливими для людського ока. Інтерфейс написано з елементами синтаксису HTML, оскільки HTML якнайкраще підходить для оформлення web-сторінок. Саму назву системи ненав’язливо, але все ж досить помітно виділено в лівому верхньому куті сторінки (рис. 6). Основне місце фону займає надпис “Sale”, що безпосередньо характеризує систему пошуку купонів знижок. Навігація по системі здійснюється завдяки використанню пунктів меню або просто гортаючи сторінку вниз. Оскільки система має вигляд web-сайту, потрібно передбачити для нього хоча б якийсь невеликий опис: його необхідність, призначення та функції. Для цього виділено пункт меню із назвою, що не потребує пояснень: “Про нас”. Пункти меню розміщено горизонтально у верхній частину сайту зліва направо. Основні пункти меню – це: “Як це працює?”, “Пошук”, “Партнери”, “Про нас”, “Контакти” (див. рис. 7).

Натиснувши на пункт меню “Як це працює?”, Ви перейдете у частину сайту з описом правил роботи із системою. Пункт меню “Пошук” відповідає рядку пошуку, де користувач може ввести запит на пошук необхідного купону на знижку і система видасть Mash-Up результатів – інформаційні ресурси із доступних джерел, які якнайповніше відображають відповідь на користувацький запит. Пункт меню “Партнери” містить перелік можливих джерел для пошуку інформаційних ресурсів із можливістю безпосередньо перейти на сайт-джерело і ознайомитися із ним детальніше. Такими джерелами є такі відомі сайти купонів на знижки: Pokupon, Biglion, Gara, Goodplace, Groupon, KupiSkidku тощо. Загалом пошук здійснюється більше ніж на двадцяти різних сайтах купонів на знижки. Пункт меню “Про нас” містить опис системи. Пункт меню “Контакти” – зворотний зв’язок для відгуків, пропозицій користувачів.

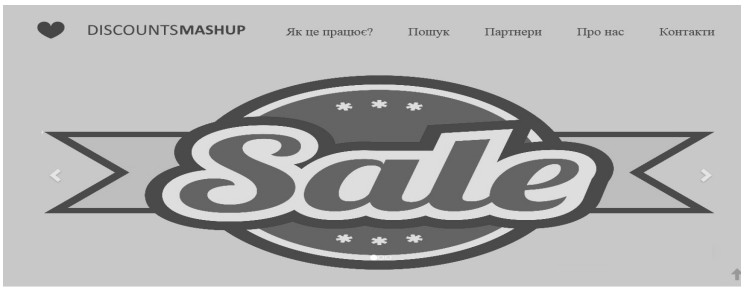

Рис. 7. Вигляд інтерфейсу системи

Приклад роботи із системою: користувач вводить у пошуковий рядок “Спа-процедуры”. Оскільки майже всі українські сайти купонів на знижки описані російською мовою, умовою для введення запиту є російська мова. Результат на цей запит ми можемо побачити на рис. 8.

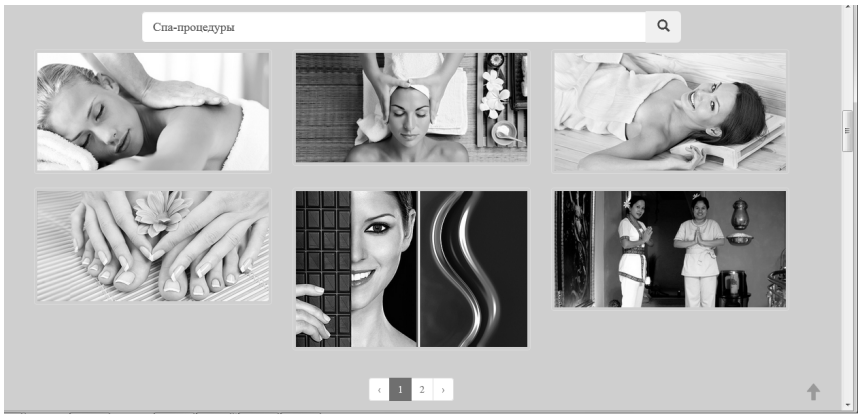

Рис. 8. Результат роботи системи

Абсолютно проста і зрозуміла схема роботи цієї системи дає змогу користувачу із будь-яким рівнем володіння комп’ютерними технологіями виконати пошук купону на знижку купівлі необхідного товару чи отримання певної послуги легко і швидко. Знайшовши потрібну знижку, користувач може безпосередньо перейти до джерела купону на знижку і ознайомитися із акцією детальніше.

Висновки та перспективи подальших наукових розвідок

Для розв’язання задачі динамічної інтеграції слабоструктурованих даних розглянуто проектування системи динамічної інтеграції слабоструктурованих даних Shares and Discounts Mashup. Показано взаємозв'язки функціональних і операційних вимог з архітектурою додатків і технологічною архітектурою в системах динамічної інтеграції даних. Описано функціональні вимоги до системи динамічної інтеграції даних на основі технології Mash-Up. Охарактеризовано процес опрацювання інформаційних ресурсів у системи динамічної інтеграції слабоструктурованих даних за допомогою алгоритму побудови онтологічної моделі всіх систем, що інтегруються, і алгоритму отримання інформаційних ресурсів із системи, що інтегрується.

Описано архітектуру системи динамічної інтеграції слабоструктурованих даних Shares and Discounts Mashup. Розглянуто функціонування спроектованої системи динамічної інтеграції слабоструктурованих даних Shares and Discounts Mashup.

Подальші дослідження будуть присвячені науковому пошуку розширення сфери застосувань розроблених і запропонованих методів при проектуванні систем, що працюють на основі технології Mash-Up динамічної інтеграції даних.

1. Kushniretska I. I. Application of MashUp Technology for Dynamic Integration of Semi-Structured Data. / I. I. Kushniretska, A. Y. Berko // Матеріали VI Міжнародної конференції молодих вчених CSE-

2013. – Львів, Вид-во Нац. ун-ту "Львівська політехніка", 2013, P. 220–221. 2. Fisher T. An overview of current approaches to mashup generation / T. Fischer, F. Bakalov, A. Nauerz // Proceedings of the International Workshop on Knowledge Services and Mashups, 2009, P. 157-158. 3. About IFTTT [Електронний ресурс] / IFTTT Inc. // Режим доступу: URL: https://ifttt.com/wtf. 4. Сповіщення. Слідкуйте за новим цікавим вмістом в Інтернеті [Електронний ресурс] / Google Inc. // Режим доступу: URL: https://www.google.com/alerts. 5. Automate your Dropbox [Електронний ресурс] / Wappwolf Inc. // Режим доступу: URL: http://wappwolf.com/dropboxautomator. 6. Giusy L. Mashups for data integration: An analysis. / L. Giusy, H. Hakim // Technical Report UNSW-CSE-TR-0810, 2008, P. 68–69. 7. Кушнірецька І. І. Визначення структури і змісту вхідних інформаційних ресурсів для роботи Мешап-системи / І. І. Кушнірецька, О. І. Кушнірецька, А. Ю. Берко // Технологический аудит и резервы производства. – 2014. – № 6/3(20). – С. 4–9. 8. ISO 15926-7: Data integration, sharing, exchange, and hand-over between computer systems. Part 7: Implementation methods for the integration of distributed systems: Template methodology, 2007.

УДК 681.513

І. А. Лур'є[1], В. В. Осипенко[2], В. І. Литвиненко[1], М. А.Таиф[1], Н. В. Корніловська[1]
[1]Херсонський національний технічний університет,
[2]Національний університет біоресурсів і природокористування України

ГІБРИДИЗАЦІЯ АЛГОРИТМУ ІНДУКТИВНОГО КЛАСТЕР-АНАЛІЗУ З ВИКОРИСТАННЯМ ОЦІНКИ ЩІЛЬНОСТІ РОЗПОДІЛУ ДАНИХ

Запропоновано нову техніку кластеризації, в основу якої покладено два методи: щільнісний алгоритм DBSCAN та індуктивний алгоритм об'єктивної кластеризації. Експерементально доведено, що комбінацією двох цих методів дозволяє вирішити проблему розпізнавання кластерів різної нелінійної форми та значно підвищити точність при розпізнаванні складних об'єктів.

Ключові слова: щільнісний метод кластеризації DBSCAN, об'єктивний алгоритм кластеризації, індуктивні методи самоорганізації моделей, МГУА, гібридні методи кластеризації.

In this article proposed a new clustering technique, which is based on two methods: density algorithm DBSCAN and inductive objective clustering algorithm. Experimentally proved that the combination of two these methods can solve the problem of recognition of clusters of different nonlinear form, and greatly increase the accuracy in the detection of complex objects.

Key words: Density-based spatial clustering, DBSCAN, objective clustering algorithm, inductive methods of self-organization models, GMDH, hybrid clustering methods.

Вступ

Задача кластеризації – окремий випадок задачі навчання без вчителя, що зводиться до розбиття наявної множини об'єктів даних на підмножини так, щоб елементи однієї підмножини істотно відрізнялися деяким набором властивостей від елементів всіх інших підмножин. Існує багато різних алгоритмів кластеризації. Деякі з них поділяють множину на заздалегідь відому кількість кластерів, деякі автоматично вибирають кількість кластерів. Алгоритм DBSCAN (Density Based Spatial Clustering of Applications with Noise) – щільнісний алгоритм для кластеризації