УДК 681.3; 004.8

О. Ю. Бочкарьов, К. В. Бакай
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

КОНТЕКСТНО-ЗАЛЕЖНИЙ ПРОГРАМНИЙ СЕРВІС ДЛЯ КОРИСТУВАЦЬКОГО КОНТРОЛЮ ЗА ВИКОНАННЯМ ЗАВДАНЬ

Розглянуто проблему розробки контекстно-залежного програмного сервісу для користувацького контролю за виконанням завдань. Запропоновано структуру та алгоритми роботи сервісу. Наведено архітектуру програмного рішення для прототипу сервісу.

Ключові слова: контекстно-залежний програмний сервіс, контроль за виконанням завдань

CONTEXT-AWARE PROGRAM SERVICE FOR THE TASKS EXECUTION USER CONTROL

The problem of developing context-aware program service for the tasks execution user control is considered. The structure and algorithms of the service are proposed. The program architecture of the service prototype is provided.

Keywords:context-aware program service, tasksexecution user control

1. Проблема контекстно-залежного контролю за виконанням завдань

В роботі розглядається задача побудови контекстно-залежного програмного сервісу для користувацького контролю за виконанням завдань, які організовані у вигляді лінійного списку, створеного користувачем. Передбачається, що цільовою апаратною платформою для сервісу є мобільні обчислювальні пристрої (МОП), зокрема МОП з ОС Android. До базової функціональності сервісу можна віднести: 1) додавання та видалення завдань зі списку, 2) редагування параметрів завдань, 3) користувацький контроль за виконанням завдань на основі інформації про їх стан та відміток про виконання, 4) формування контекстно-залежного списку актуальних завдань (САЗ).

Реалізація 4-ої функції передбачає, що в роботі сервісу враховується так званий контекст (context), тобто різні відомості про поточний стан оточення користувача, які сервіс отримує самостійно за допомогою відповідного апаратно-програмного забезпечення МОП. Відтак ідея сервісу полягає в тому, що завдання, які сервіс відображає для користувача в першу чергу (САЗ), найбільше відповідають різним аспектам тої ситуації, в якій зараз знаходиться користувач. Тобто САЗ містить деяку підмножину тих завдань, які є найбільш актуальними для користувача на теперішній час і при теперішніх умовах. Дослідження та розробка подібних сервісів, які отримали назву контекстно-залежних (context-aware) [1,2] розпочалися на початку 90-их років разом з виникненням перших прототипів МОП. На даний час завдяки бурхливому розвитку технологій мобільних обчислень (mobile computing) та широкому розповсюдженню МОП серед користувачів, дослідження та розробки в галузі контекстно-залежних обчислень стають все більш актуальними.

Для того, щоб мати доступ до контекстно-залежного контролю за виконанням завдань потрібно отримати звідкись цей контекст. Апаратне забезпечення сучасних МОП, ОС Android та інтернет-сервіси надають багато можливостей для цього. Наприклад, можна отримати поточний час і дату, інформацію про день тижня, інформацію про погоду через інтернет з відповідного сервісу, показники різних вбудованих сенсорів та ін. Усього цього достатньо, щоб сформувати користувацький контекст.

2. Алгоритм формування списку актуальних завдань (A1)

Розглянемо список актуальних завдань $A = \{q\}_x$, який складається з $x < n$ завдань, у який відбираються актуальні завдання із загального списку всіх завдань $Q = \{q\}_n$. Рішення про попадання завдання в САЗ приймається на основі його динамічного пріоритету d(q), який перераховується або із заданим інтервалом, або по факту звернення користувача до САЗ. Перерахунок динамічних пріоритетів (рис.1) здійснюється для всіх завдань $\{d(q)\}_n$, після чого завдання сортуються по спаданню величини динамічного пріоритету і в САЗ відбираються перші x завдань з найбільшими d(q).

Рис. 1. Алгоритм формування списку актуальних завдань (A1)

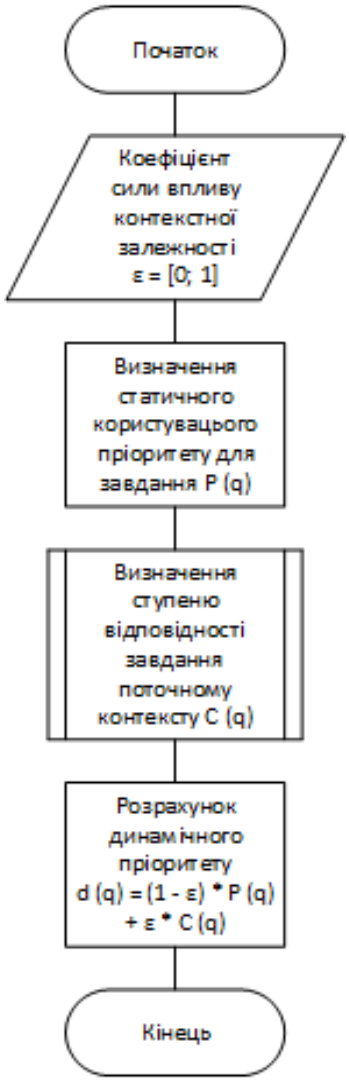

Рис. 2. Алгоритм розрахунку динамічного пріоритету (A2)

3. Алгоритм розрахунку динамічного пріоритету завдання (А2). Динамічний пріоритет розраховується (рис.2) за формулою:

$$d(q) = (1 - \varepsilon) \cdot P(q) + \varepsilon \cdot C(q), \quad (1)$$

де P(q) - фіксований (статичний) пріоритет завдання, який задається користувачем, C(q) - індикатор відповідності завдання поточному контексту (в більш загальному випадку ця велични може відображати ступінь відповідності завдання q поточному контексту), $\varepsilon \in [0;1]$ - коефіцієнт, яким регулюється сила впливу контекстної залежності на потрапляння завдання в САЗ (в базовому варіанті ε=const). Якщо, наприклад, ε= 0.5, то пріоритет завдання P(q) заданий користувачем і індикатор відповідності завдання поточному контексту C(q) мають однаковий «вплив» на формування значення динамічного пріоритету d(q). Відповідно, якщо ε= 0.2, то пріоритет заданий користувачем буде мати більший «вплив», ніж залежність від контексту, а в разі ε= 0.8 - навпаки.

Для зручності розрахунків та реалізації сервісу призначимо P(q) наступний діапазон значень $P(q) \in [0;100]$. При цьому для користувача можна залишити більш просту схему пріоритетів (наприклад, від 1-го до 5-ти), відображаючи ці значення в діапазон [0; 100] всередині програми. Відповідно, якщо, наприклад, для користувача буде існувати 5 рівнів пріоритету, а для розрахунків ми візьмемо діапазон значень [0; 100], то коли користувач обере 1 рівень, то значення P(q) буде дорівнювати 0, якщо 2 рівень, то значення P(q) буде 25, якщо 3 рівень, то значення P(q) буде 50, якщо 4 рівень, то значення P(q) буде 75, якщо 5 рівень, то значення P(q) буде 100.

Значення C(q) в базовому варіанті індикаторне: якщо завдання q відповідає поточному контексту, то C(q) = 100 (тобто дорівнює максимальному значенню P(q)); якщо завдання q не відповідає поточному контексту, то C(q) = 0 (тобто дорівнює мінімальному значенню P(q)). У більш складному випадку значення C(q) може відображати ступінь відповідності завдання q поточному контексту і бути величиною в деякому діапазоні, наприклад $C(q) \in [0;100]$ (тобто в тому ж діапазоні що і P(q)).

4. Алгоритм визначення ступеню відповідності завдання поточному контексту (А3). Значення {C(q)} для всіх завдань визначаються в блоці прийняття рішення про відповідність завдання поточному контексту.Алгоритм А3 визначення C(q) складається з наступних кроків (рис.3):

1) Цикл по всіх параметрах контексту: для кожного параметра визначається його значення, виходячи з показів «бортових» сенсорів і аналізу інших джерел інформації (годинник, календар, погода, місце розташування і т.д.). Приклади параметрів і значень: <час доби> = «ранок», <день> = «вихідний», <погода> = «сонячно», <місце розташування> = «вдома».

2) Цикл по всім завданням $Q = \{q\}_n$: обнулити значення C(q).

3) Цикл по множині правил if-then $U = \{u\}_m$, якими задається відповідність завдань контексту.

Шаблон правила u: якщо<параметр1 = y & параметр2 = z & …>, тодля всіх завдань {q} з <#тег1 & #тег2 & …> C(q):=100. Зауважимо, що можуть одночасно спрацювати декілька правил, у яких виконались умови рівності параметрів контексту до поточних значень, і всім відповідним завданням буде встановлено C(q):=100, що підніме їх динамічний пріоритет. Однак повстає проблема - машина сама по собі не може зрозуміти, що означає кожне завдання по назві і відповідно не може застосувати до нього певні правила. Є декілька

варіантів того, як уникнути це обмеження, та зробитит так, щоб програма почала розуміти «суть» завдання: 1) користувач при формуванні завдання вибирає теги (із заданого набору), які характеризують зміст цього завдання; 2) у тексті назви або опису завдання користувач позначає потрібні слова хештегом; 3) програма виконує граматичний розбір назви та опису завдання і знаходить слова-теги (також використовуючи попередньо задану «опорну множину» тегів).

Отже, тепер ми можемо сформулювати правила, які будуть забезпечувати взаємодію сервісу з поточним контекстом. Цих правил може бути багато і їх можна змінювати або додавати нові з чероговими оновленнями сервісу. Як варіант, можна дати користувачеві повний або частковий доступ до цих правил (додавання, видалення, редагування). Приклади правил відповідності завдання поточному контексту:

u_1: **якщо**<час_доби = вечір & день = вихідний & розташування = вдома>, **то**для всіх завдань q з <#подивитися & #фільм> C (q): = 100.

u_2: **якщо**<час = робочий_час_магазинів & день = будній & розташування = поряд_з_магазином_де_продається_X & сума_на_рахунку > ціна (X)>, **то** для всіх завдань q з <#купити & #X> C(q) : = 100.

u_3: **якщо**<час_доби = день & день = вихідний & розташування = вдома & погода = сонячно>, **то** для всіх завдань q з <#гуляти & #відпочинок> C(q) : = 100.

u_4: **якщо**<час_доби = ранок & день = останній у місяці & розташування = вдома>, **то** для всіх завдань q з <#інтернет & #рахунки> C(q) : = 100.

u_5: **якщо**<час_доби = день & день = будній & розташування = офіс>, **то** для всіх завдань q з <#робота> C(q) : = 100.

u_6: **якщо**<час_доби = вечір & день = день перед екзаменом & розташування = вдома>, **то** для всіх завдань q з <#сесія & #екзамен> C(q) : = 100.

u_7: **якщо**<час_доби = день & день = будній & розташування = магазин>, **то** для всіх завдань q з <#продукти> C(q) : = 100.

5. Алгоритм навчання вибору правил відповідності завдання поточному контексту (A4). Оскільки запропонований сервіс є персоналізованим, то виникає потреба у реалізації навчання вибору правил відповідності завдання поточному контексту (яке буде «налаштовувати» вибір тих чи інших правил відповідності під потреби даного користувача). Для реалізації алгоритму навчання в даній роботі обрано один з метододів навчання з підкріпленням (reinforcement learning), а власне метод нерівноцінного вибору дій (softmax action selection) [3,4]. Зауважимо, що у варіанті без навчання ми в пункті 3 алгоритму А3 - визначення C(q) (цикл по правилам if-then) використовуємо всі правила, які у нас є. В той же час у варіанті з навчанням ми кожному правилу призначаємо ймовірність його використання. Тобто в циклі за правилами if-then буде розглядатися тільки підмножина всіх наявних правил, яка буде формуватись алгоритмом навчання з підкріпленням. Відповідно для кожного нового спрацьовування алгоритму А3 буде формуватись нова підмножина правил {u}.

Для побудови алгоритму введемо такі величини:

1) Підкріплення r(t) показує наскільки поточна підмножина правил {u} влаштувала користувача при його черговому зверненні до САЗ. Є різні способи розрахунку підкріплення r(t): a) r(t) визначається явно користувачем (like / dislike, r(t) = {+ 1, -1}); b) r(t) формується на основі аналізу кількості виконаних завдань з поточного САЗ (чим більше, тим краще, r(t) =

[0; x]); с) якщо користувач звернувся до загального списку (тобто його не «задовільнив» САЗ і він почав шукати «цікавіші» завдання), то r(t) = -1, інакше r(t) = +1.

2) v(u) = {0;1} - дві дії: не використовувати (v(u)=0) або використовувати (v(u)=1) правило.

3) k(u,1) - лічильник загального числа виконаних дій v(u)=1 (k(u,0) - те саме для v(u)=0).

4) R(u,1) - сума всіх підкріплень за реалізацію дії v(u)=1 (R(u,0) - те саме для v(u)=0).

5) Q(u,1) - оціночна вага (action value) дії v(u)=1 (Q(u,0) - те саме для v(u)=0).

6) p(u,1) – ймовірність вибору дії v(u)=1 (p(u,0) - те саме для v(u)=0); p(u,1)+p(u,0)=1.

7) x – випадково згенероване число в діапазоні [0,1].

Алгоритм А4 складається з наступних кроків (рис.4):

1) Отримати величину підкріплення r(t).

2) Цикл по всім правилам $U = \{u\}_m$, для кожного правила:

якщо v(u) = 1 (правило u використовувалося),

то k(u,1) = k(u,1) + 1; R(u,1) = R(u,1) + r(t); Q(u,1) = R(u,1) / k(u,1);

інакше (правило u не використовувалося):

k(u,0) = k(u,0)+1; R(u,0) = R(u,0) + r(t); Q(u,0) = R(u,0) / k(u,0).

3) Відобразити оціночну вагу Q(u,1) у ймовірність вибору дії p(u,1) (при цьому p(u,0)=1-p(u,1)).

4) Згенерувати випадкове число x в діапазоні від 0 до 1.

5) Якщо x < p(u,1), то v(u) = 1, інакше v(u) = 0.

6) Всі правила {u}, для яких v(u) = 1, включити в підмножину використовуваних правил.

Ймовірність вибору дії v(u)=1 розраховується за формулою:

$$p(u,1) = \{\exp (Q(u,1) / \tau)\} / \{\exp (Q(u,0) / \tau)\}, \quad (2)$$

де τ - коефіцієнт яким регулюється величина випадкової складової у виборі наступної дії: $\tau > 0$, τ=const (чим більше значення, тим більше випадкова складова).

6. Структура сервісу та архітектура програмного рішення

В структурі сервіса (рис.5) можна виділити настуні основні блоки: 1) блок, який визначає поточний контекст, в який входять такі параметри: година дня, день тижня, погода, місцероташування, покази сенсорів МОП та вхід «ручного» управління користувача (передбачається, що користувач може уточнювати поточний контекст встановлючи значення деяких параметрів); 2) після визначення поточного контексту відповідна інформація зберігається в моделі біжучого контексту, звідки дані потрапляють на сервер та одночасно використовуються локально блоком прийняття рішень; 3) з сервера на мобільний додаток сервісу, крім користувацьких даних, ще приходить інформація про спільний контекст та необхідні службові дані; 4) генератор завдань призначенний для створення нових завдань (в тому числі на основі аналізу поточного контексту); 5) у блоці прийняття рішень реалізовані алгоритми А1, А2, А3; алгоритм А4 реалізований в блоці навчання з підкріплененям.

Розширена структура контекстно-залежного програмного сервісу та архітектура програмного рішення зображені на рис. 6 та рис. 7. Прототип сервісу розроблений на платформі Android та складається з двох частин: клієнтської та серверної. Основна функціональність сервісу, розглянута в даній роботі, реалізована у клієнтській частині.

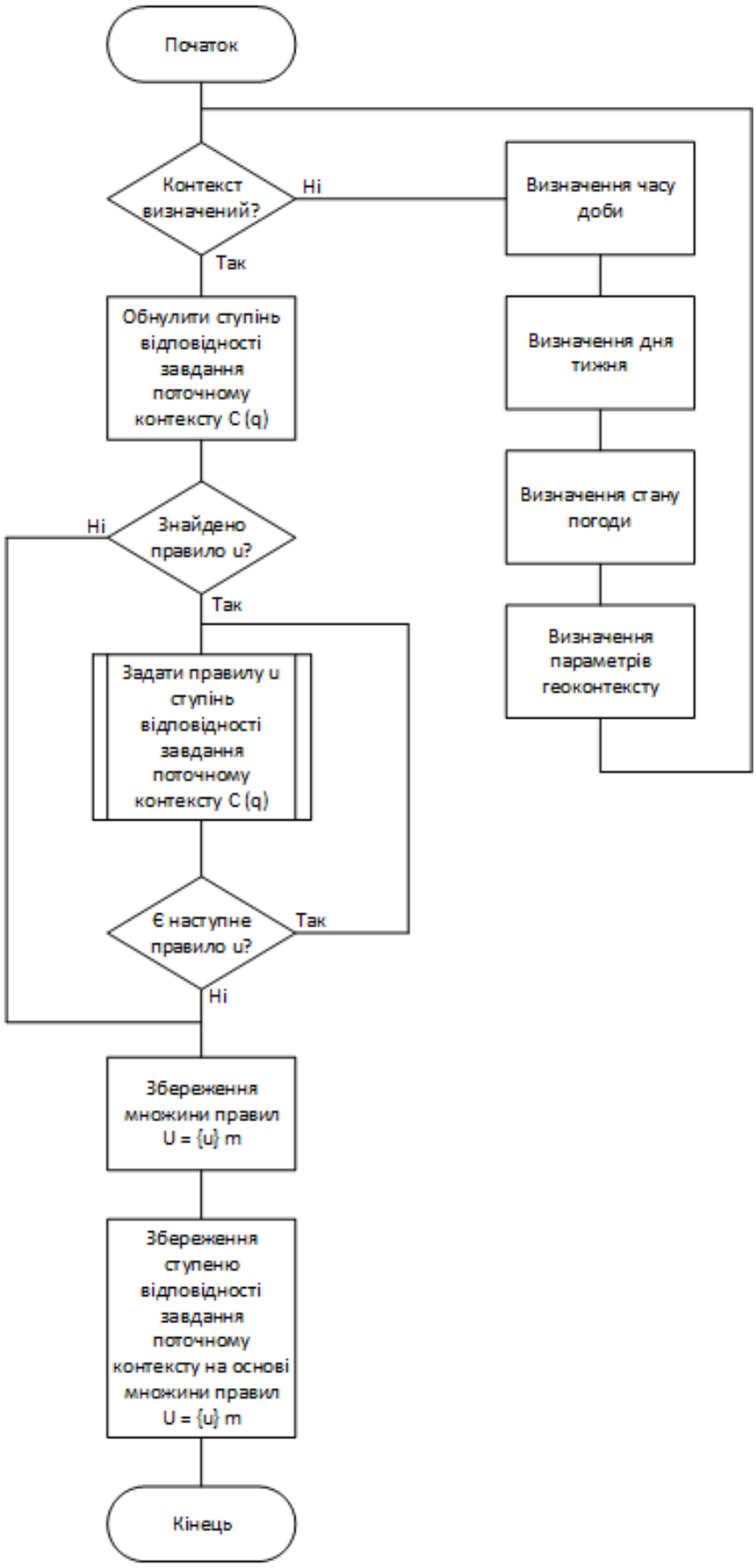

Рис. 3. Алгоритм визначення відповідності завдання поточному контексту (А3)

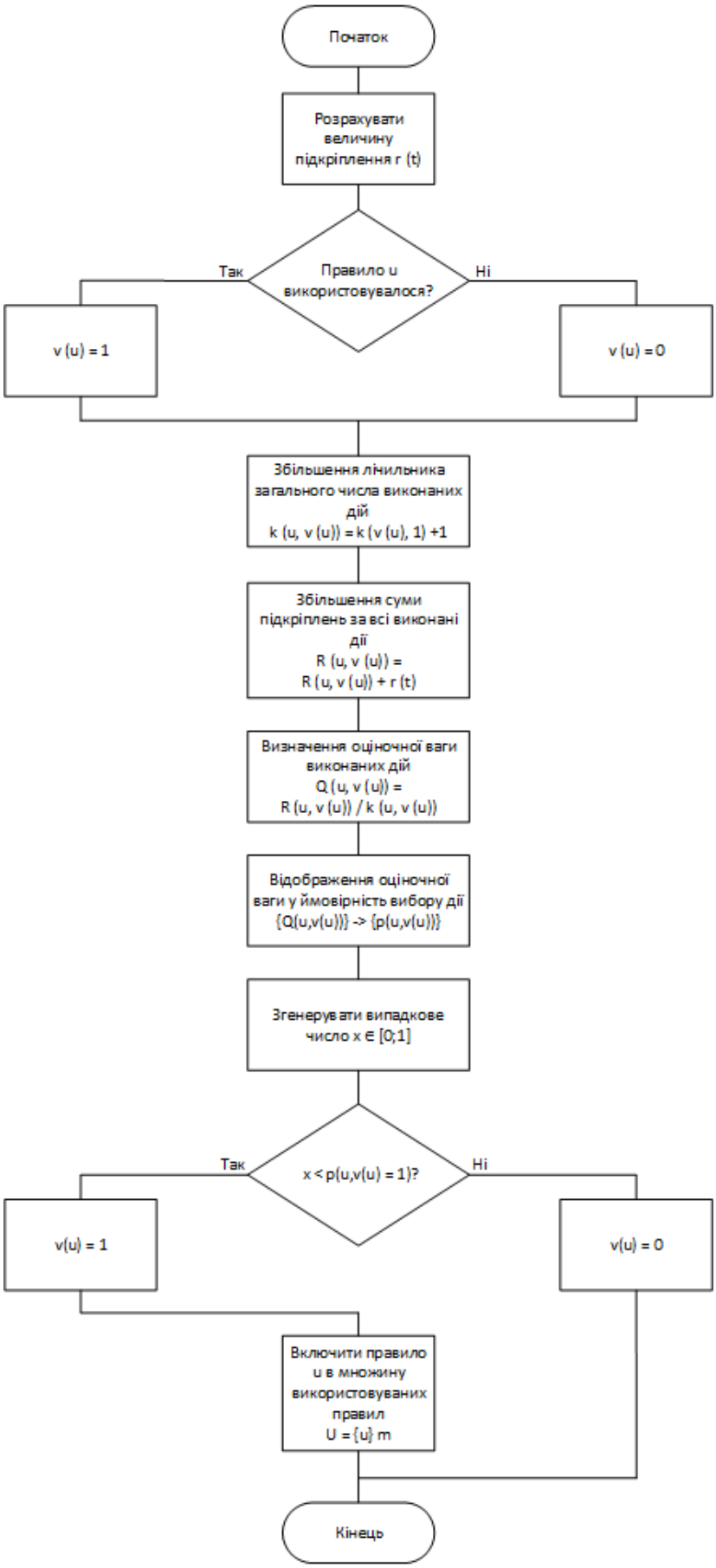

Рис. 4. Алгоритм навчання вибору правил відповідності завдання поточному контексту

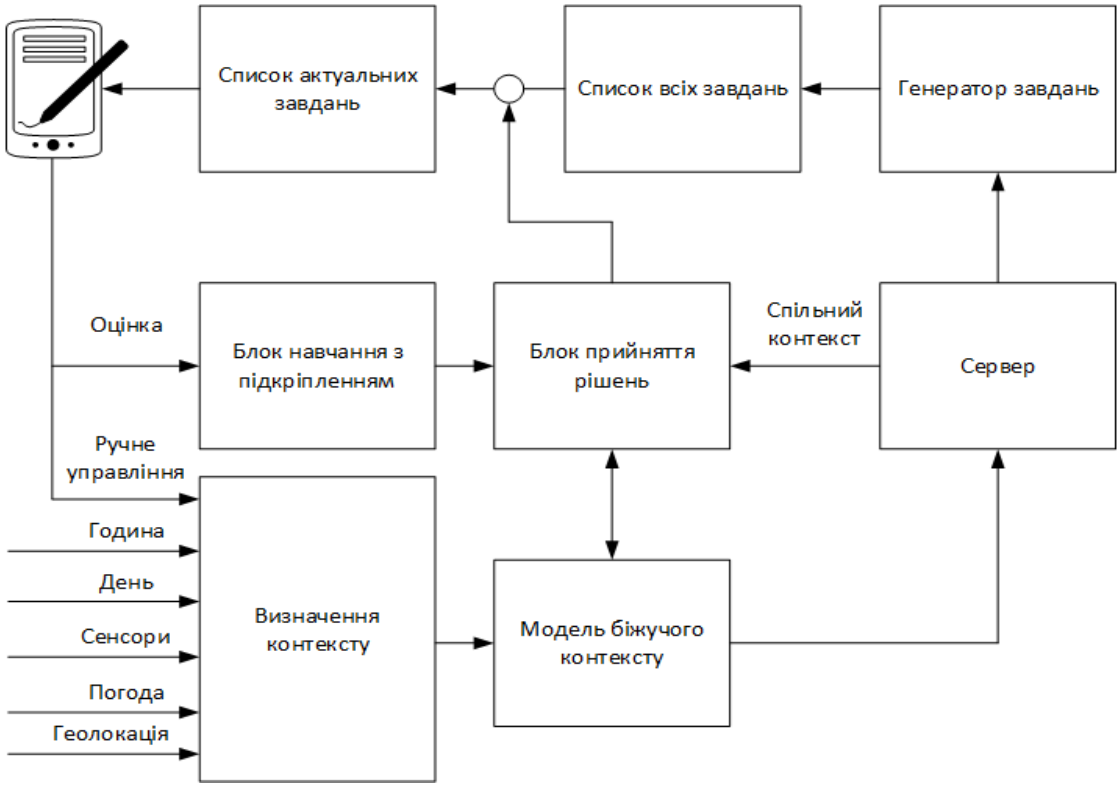

Рис. 5. Структура контекстно-залежного програмного сервісу

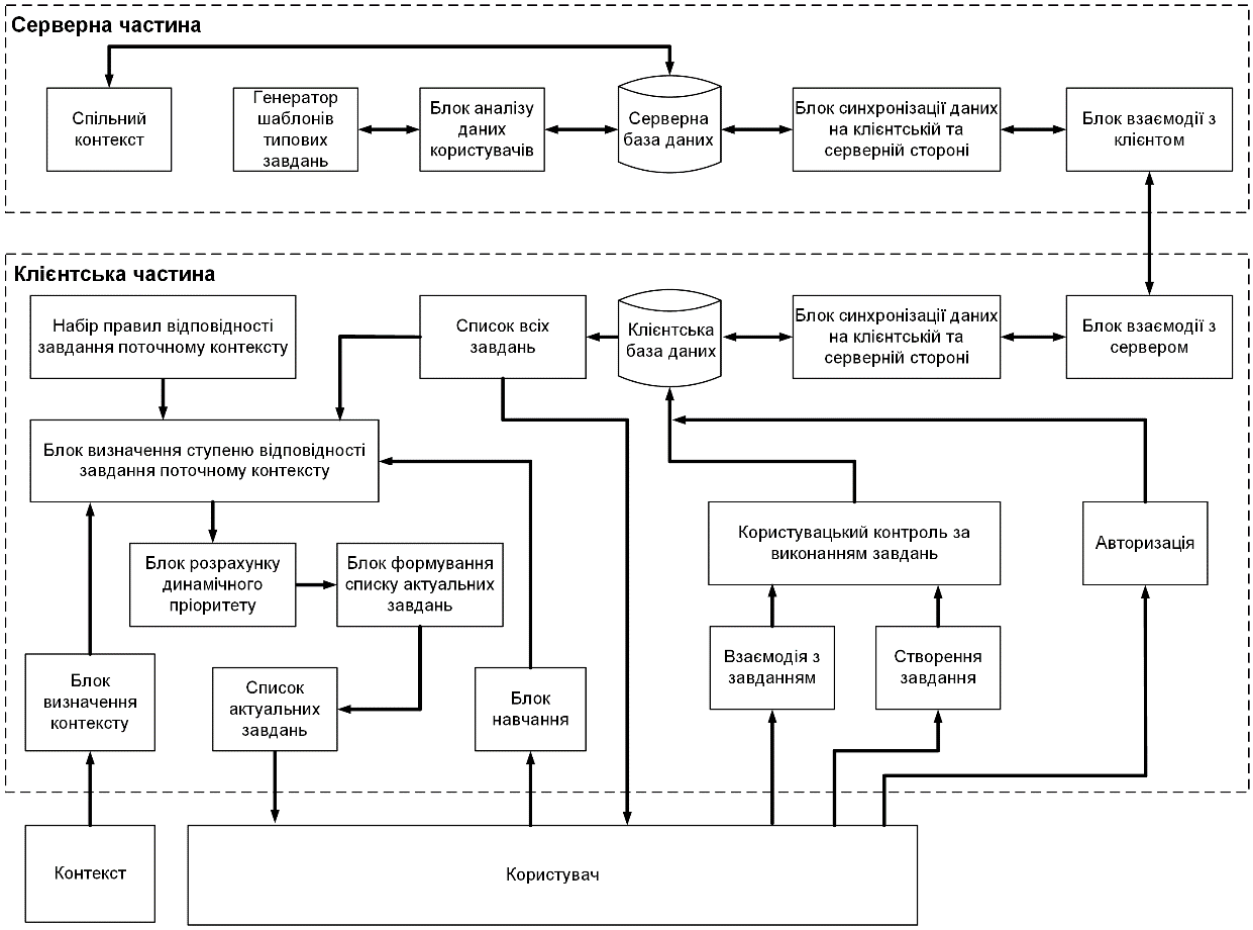

Рис. 6. Розширена структура контекстно-залежного програмного сервісу

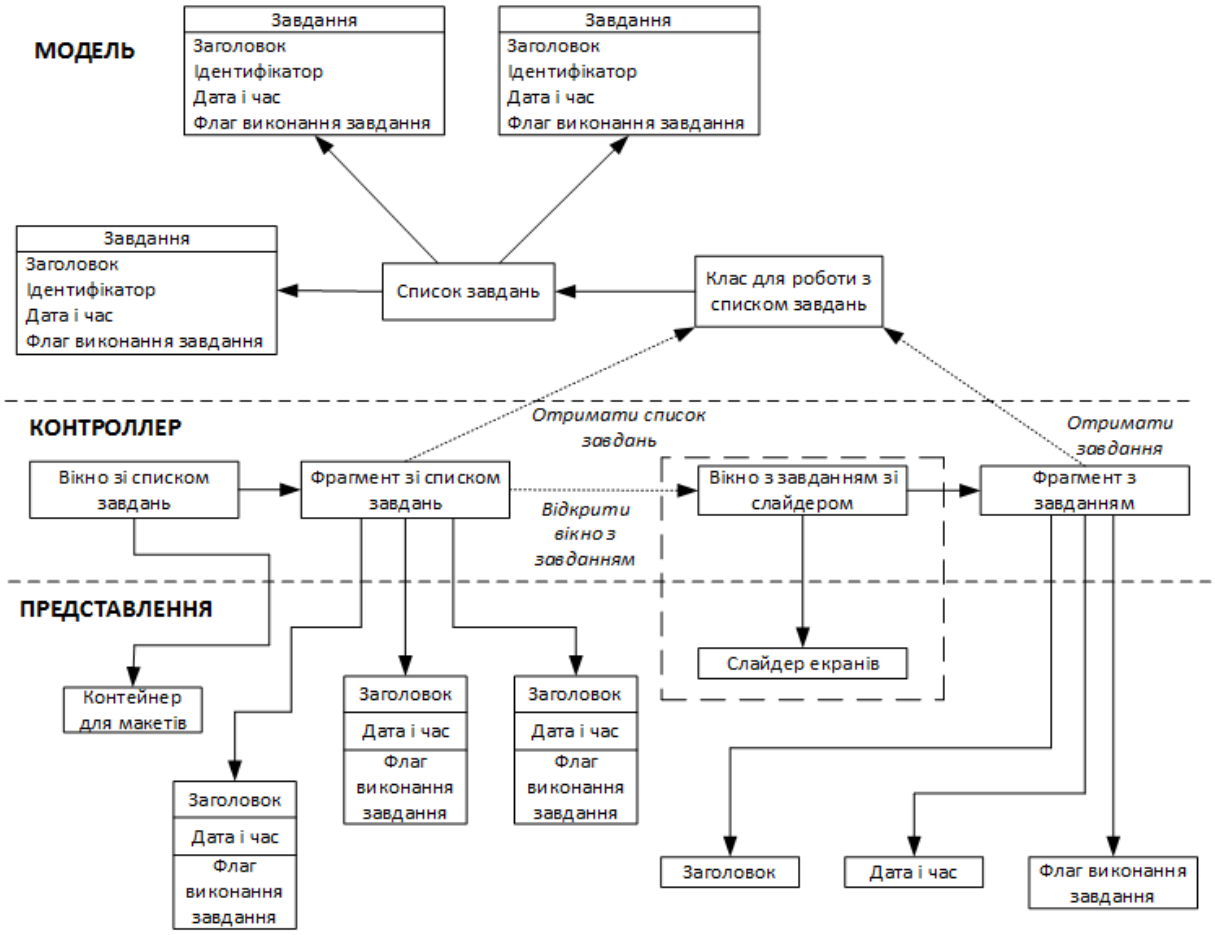

Рис. 7. Архітектура програмного рішення

1. Bill N. Schilit, Norman Adams, Roy Want. Context-Aware Computing Applications, Mobile Computing Systems and Applications, 1994р. – с. 85-90. 2. Dey A. K. Understanding and Using Context, Personal and Ubiquitous Computing, 2001р. – с. 4-7. 3. Richard S. Sutton, Andrew G. Barto. Reinforcement Learning: An Introduction, The MIT Press, 1998р. – с. 334. 4. Csaba Szepesvari. Algorithms for Reinforcement Learning, Morgan and Claypool Publishers, 2009р. – с. 104.

Наукові результати, подані у цій статті, було отримано в рамках дослідницького проекту ДБ/КІБЕР з реєстраційним номером 0115U000446, 01.01.2015 – 31.12.2017, фінансово підтриманим Міністерством освіти та науки України.