УДК 004.728.8

ПРОТОКОЛИ ДЛЯ ІНТЕГРАЦІЇ МОБІЛЬНИХ ПРИСТРОЇВ ІЗ СТАНЦІЯМИ НАДАННЯ ПОСЛУГ ЧЕРЕЗ БЕЗДРОТОВИЙ ЗВ'ЯЗОК

К.І. Яценко

Київський національний університет імені Тараса Шевченка,
Київ, проспект Академіка Глушкова, 2,
kyatsenko@gmail.com

Запропоновано новий метод в області інформаційних технологій по наданню локальних послуг через безшовну інтеграцію мобільних пристроїв. Підхід базується на класі протоколів XDEP (XML data exchange protocol). Основним завданням цього підходу є заміна існуючих методів надання інформації, які функціонують завдяки спеціалізованим пристроям. Пропонується перенести цю функціональність на пристрої масового використання – персональні електронні записники, мобільні телефони, MP3 плеєри та інші.

A new approach in the area of information technologies of providing local services via seamless integration of mobile devices has been proposed. The approach resides on the basis of XDEP (XML data exchange protocol). The solution is intended to substitute existing methods of information services providing, which reside on a specialized hardware. The functionality is to migrate to the commonly used devices like PDAs, Mobile Phones, MP3 players, etc.

Вступ

У сучасних динамічних умовах розвитку інформаційних технологій, рівень інтеграції мобільних пристроїв, персональних комп'ютерів та серверів через електронні канали є високим. Кожний мобільний пристрій містить персональну інформацію, яку користувачі потребують для вирішення атомарних питань. Персональні комп'ютери акумулюють інформацію, яку користувачі збирають на протязі деякого часу. Серверні системи представляють засіб довготривалого зберігання інформації від різних користувачів. Отже, інформацію, яку користувачі потребують в різні періоди часу можна класифікувати на три категорії:

- використання для дії (зберігається на мобільних пристроях);
- використання для діяльності (набір дій, зберігається на персональному комп'ютері);
- використання для ведення хронології та планування (набір різних діяльностей, які зберігаються на серверній платформі).

За цією класифікацію, дані зберігаються та організовуються на різних фізичних платформах. Такий підхід до зберігання інформації потребує постійної синхронізації обладнання. Логічний процес взаємодії різних програмно-апаратних комплексів, що розглядається в даній роботі, пропонує уніфікований метод його організації.

Клас протоколів XDEP (XML Data Exchange Protocol) вперше представлений під час конференції ICTA 2007 [1]. Головною ідеєю цього класу є представлення даних в універсальному форматі, який системи, за різними логічною та фізичною будовами, могли б інтерпретувати швидко та безпомилково. Підхід до побудови структури цього класу протоколів був схожий до структури веб-сервісів [2] розроблених W3C. Базою для веб-сервісів є метаопис даних, який легко інтерпретується будь-якою програмою [1].

Для того, щоб висвітлити наступний етап еволюції XDEP розглянемо наступний приклад. У Парижі, Франції, як альтернативу використання транспорту на бензині, комунальні служби запропонували використання велосипедів (послуга *Velolib*) за дуже символічну платню. Передбачається, що така послуга розвантажить автомобільний трафік та позитивно вплине на навколишнє середовище. Однак, з метою попередження можливості не цільового використання або крадіжки велосипедів, мандрівник має валідувати кредитну картку, перед тим, як скористатися новою послугою громадського транспорту. Велосипед можна взяти безкоштовно на півгодини, після чого, за кожні півгодини починаючи з одного євро, ціна прогресує в арифметичній прогресії. Платформи (місця, де можна отримати та здати велосипед) знаходяться по всьому місту, отже зміна велосипеду не представляє складнощів, та може бути виконана дуже швидко. Проте, мандрівники змушені кожного разу, під час заміни велосипеда повторювати операцію з введенням персональних даних. Такий процес може займати для однієї людини близько 3 хвилин, що складає 10 % від потенційного часу використання послуги. Враховуючи те, що завжди кількість автоматичних пунктів (екрани з клавіатурою) на платформі не перевищує 1, в разі черги, економія часу при використанні таких послуг стає дуже сумнівною. На поточний момент єдиною можливістю вирішення цієї проблеми є встановлення додаткових автоматичних пунктів (хоча, в цьому випадку виникають складнощі з синхронізацією дій мандрівників, які хочуть взяти один і той самий велосипед).

Водночас, у деяких країнах (таких, наприклад як Україна) кількість користувачів мобільних телефонів перевищила 100 % популяції. Це означає, що кожна людина має як мінімум один мобільний телефон. Як результат кожний користувач має постійно із собою персональну консоль із можливостями вводу та виводу

даних. Інтерфейс для представлення даних для використання типу *дії* (або представлення атомарних послуг), може бути перенесений на будь-який персональний пристрій. Цей підхід відображає наступний крок еволюції XDEP.

Структура XDEP

Рис. 1 структурно показує кон'юнктуру використання протоколів. Кожний протокол передбачений для дії у ролі медіатора між різними клієнтськими та серверними платформами. Завдяки текстовому формату, протокол може бути легко адаптований під різні системи, що дозволяє його широке використання.

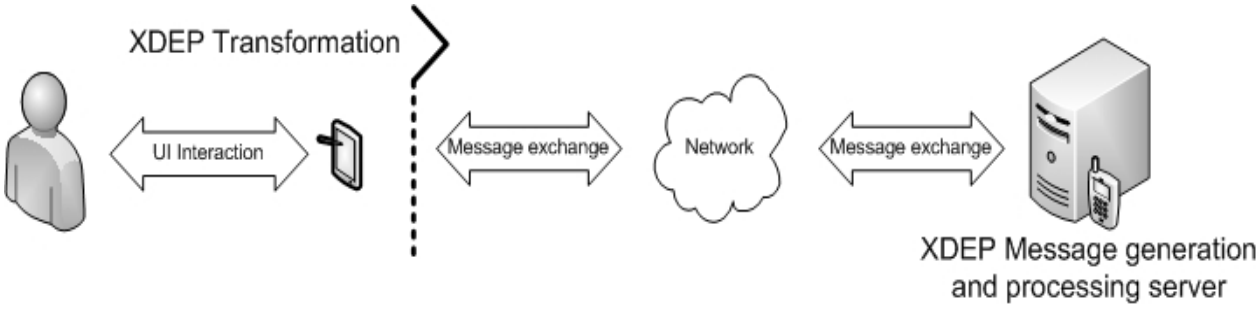

Рис. 1

Для проекту Fenestra на конкурсі Microsoft Imagine Cup 2007 [3] автори удосконалили XDEP для того, щоб надати можливість людям із ускладненнями зору отримувати інформаційні послуги використовуючи звичайні портативні пристрої. Інформація із серверної програми подавалась на мобільний пристрій з голосовими можливостями в форматі XDEP, яка представлялась користувачеві через голосовий інтерфейс. Високорівневу архітектуру цієї системи показано на рис. 2.

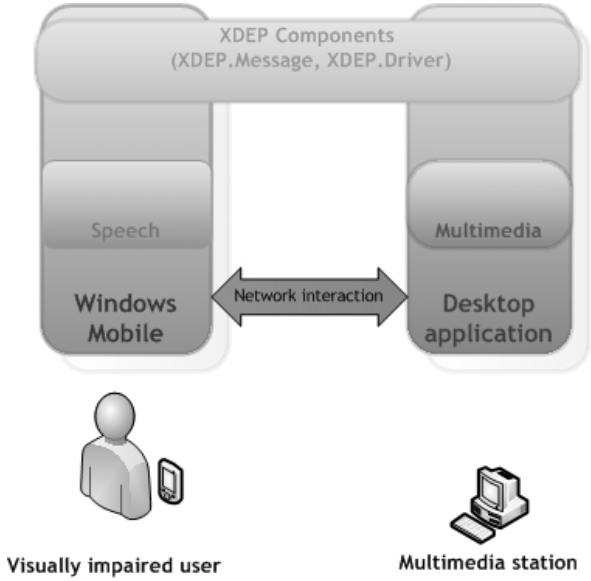

Рис. 2

Під час реалізації цього проекту було проведено аналіз ефективності конструювання цієї системи, використовуючи так звані Smart Client архітектури, а також багатошарові системи. Вибір зупинено на другому варіанті, так як він гарантував можливість розширення клієнтської бази приладів та пришвидшення роботи із великими масивами даних. Отже, система була розбита на наступні програмні модулі:

- спільні модулі для серверної так клієнтської частини являли собою так звані шаблони (Domain Patterns), які описували загальну структуру повідомлень в машинному представленні (набір класів). Ця бібліотека включалась одночасно в обидві програми;
- модулі, що розміщювались тільки на клієнтському пристрої:
  - Speech Interface;
  - XDEP.Client.

Перший модуль реалізовував комунікацію з сервером при обробці голосових команд. Основними задачами цього модуля були: захват звуку, його передача на сервер, та програвання отриманих звукових файлів. Другий

модуль займався безпосередньо генеруванням та інтерпретацією системних повідомлень між приладом та серверною програмою.

Що стосується серверної частини, то в ній був реалізований модуль інтерфейсу доступу до голосових операції та набір бібліотек який відповідав за виконання команд протоколу. Завдяки використанню властивості рефлексії серверна програма могла бути розширена додатковими командами, через додавання спеціалізованих бібліотек які реалізовували заданий програмний інтерфейс через оновлення файлу конфігурації. Таким чином досягався ефект розширення програми в режимі реального часу. Графічно, модульну архітектуру системи показано на рис. 3.

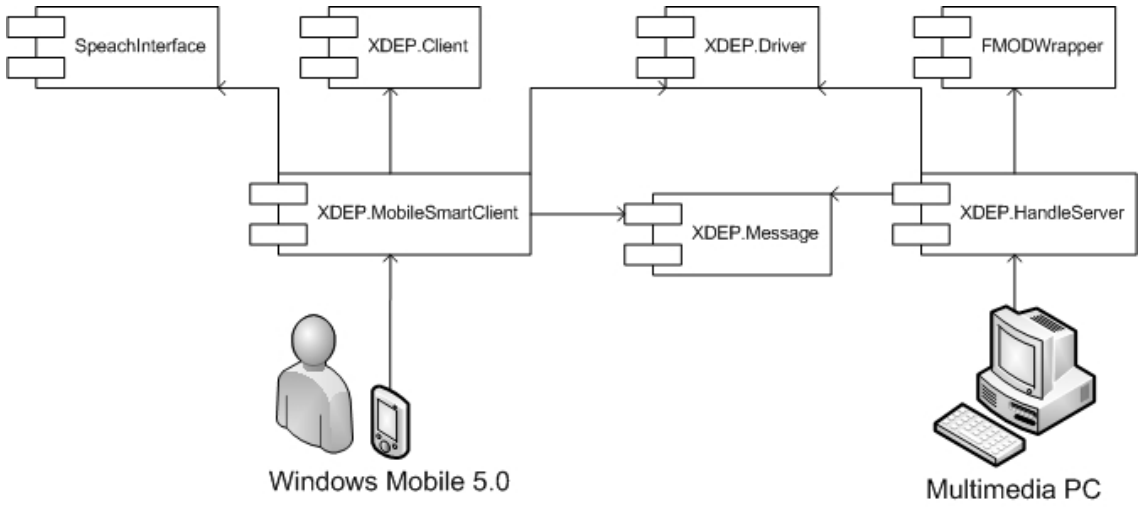

Рис. 3

Особливістю цього рішення складало обладнання, яке було використано для реалізації системи. А саме, для повноцінного функціонування системи було використане таке обладнання: Fujitsu-Siemens Loox 520 32MB, Pocket PC Linksys WRT54G wireless adapter, Laptop Compaq nx9030.

Сумарна вартість використаного обладнання складала приблизно 750 доларів США. Причому, більшу частину було витрачено на персональній комп'ютер, який не зважаючи на слабку потужність успішно виконував аналіз голосових команд одночасно працюючих з протоколом. Тобто, ми змогли отримати можливість побудови багатофункціональних рішень із незначними фінансовими затратами.

Повертаючи до технічної реалізації протоколу розглянемо наступні приклади. Далі представлено одну із команд протоколу XDEP для проекту Fenestra:

```
<?xml version="1.0" encoding="utf-8"?>
<XDEPDriver ProtocolName =”Fenestra” ServerName=”Fenestra Server”
startcommand=””>
  <Commands>
    <Command>
      <Name>protocol syntax</Name>
      <IsStatic>false</IsStatic>
      <IsDriverDescription>true</IsDriverDescription>
      <IsSilent>false</IsSilent>
      <Description>gets the protocol description</Description>
      <Parameters>
        < param >
          <ParamName>Protocol</ParamName>
          <PredecessorParamReference />
          <ParamType>String</ParamType>
          <ParamDefaultValue = “” />
          <Direction>Out</Direction>
        </ param >
      </Parameters>
      <Successors>
      </Successors>
    </Command>
    <Command>
      <Name>music play</Name>
```

```
      <IsStatic>true</IsStatic>
      <IsSilent>false</IsSilent>
      <Description>If you want to listen to some music</Description>
      <Parameters>
      </Parameters>
    </Command>
    <Command>
      <Name>music stop</Name>
      <IsStatic>true</IsStatic>
          <IsDriverDescription>false</IsDriverDescription>
          <IsSilent>false</IsSilent>
      <Description>If you want to stop listening to the music</Description>
      <Parameters>
      </Parameters>
    </Command>
  </Commands>
</XDEPDriver>
```

Цей приклад демонструє три команди зі структурою, що самоописується. Одна з цих команд `protocol syntax` – базова команда для всіх протоколів, побудованих за допомогою правил XDEP. Дві інші (`music play` – команда, яка запускає програвач музики, та `music stop`, яка зупиняє аудіотрек) є специфічними командами даного прикладу.

При розгляді XDEP необхідно приділити додаткову увагу іменуванню команд, їх опису та параметрам. Здавалося б, досить великий опис команд збільшує їх розмір, і таким чином збільшується мережевий трафік. Однак, враховуючи швидкість росту потужності комп'ютерних мереж, цей факт можна ігнорувати. На противагу, такий підхід дозволяє інтегрувати різні прилади, навіть ті, які орієнтовані для людей з проблемами зору. Як результат клієнтські програми можуть успішно використовувати аудіо інтерфейси, що дозволяє не змінювати представлення даних (метаданих) та адаптувати команди для приладів з різним інтерфейсом.

Наступним кроком вивчення протоколу буде розгляд опису команд в загальному випадку.

```
<?xml version="1.0" encoding="UTF-8"?>
<XDEPDriver ProtocolName="TheNameOfTheCurrentProtocol"
      ServerName ="TheNameOfTheServerDevice"
      startcommand="NameofTheStartingcommand">
<Command name="CommandName" isStatic="boolean"
      description="userfriendlydescription">
<parameters>
            <param
             paramName="paramName"
             predecessorParamReference="paramName"
             paramType= "integer|boolean|string|list"
       maximumValues =”integer value”
       minValues=”integer value”
       validationRule=”regular expression”
             paramDefaultValue= "thevalue"
             direction="in|out|inout">
            <values>
                              <value key="value" value="value"/>
                  </values>
                  </param>
            </parameters>
            <successors>
            <successor commandName="CommandName"
                      autocall="bool"/>
            </successors>
    </Command>
```

Комунікація між клієнтськими та серверними програмами виконується, в командному режимі. Далі пропонується загальний синтаксис запитів та відповідей протоколу. Запит виглядає так:

```
  <XDEPInlineCommand commandName="CommandName">
      <parameters>
        <param paramName="paramName" paramValue="thevalue"/>
```

```
  </parameters>
</XDEPInlineCommand>
 Відповідь:
<!--Response sent to the client's command-->
<XDEPInlineResponse commandName="commandName">
  <param
          paramName="paramName"
          paramDefaultValue= "thevalue">
    <!--in case the value is a list-->
    <values>
      <value key="value" value="value"/>
    </values>
  </param>
</XDEPInlineResponse>
```

Перший запит, що відправляється до серверної команди має бути `ProtocolSyntax,` для того, щоб отримати опис всіх запитів та відповідей, доступних в конкретній серверній реалізації протоколу, та порядок їх виконання.

```
 <ProtocolSyntax>
  <parameters>
    <param paramName="devicename" param
  Value="the_name_of_the_device"/>
  </parameters>
</ ProtocolSyntax >
```

Оскільки протокол вимагає виконання команди `ProtocolSyntax`, вона має бути закодована в клієнстську програму. Відповідь на цей запит надає користувачу синтаксис протоколу. Такий підхід дозволяє в універсальній формі:

- конструювати інтерфейси користувача, не прив'язуючись до конкретного візуального представлення даних;
- визначати кроки виконання програми.

Ідея цього підходу нагадує принципи роботи HTTP. Однак, існує три суттєві особливості, що відрізняють XDEP від HTTP та веб-сервісів:

- команди добре структуровані, що дозволяє їх ефективно оброблювати машиною;
- інтерфейс користувача конструюється під час роботи програми;
- чітко визначені кроки виконання та порядку обміну командами.

Третя особливість вимагає додаткового розгляду. Ідентифікація порядку виконання команд досягається через теги `startcommand, successors` та `predecessorParamReference` (див. синтаксис протоколу). Перший тег використовується для визначення першої команди циклу виконання програми. Цю команду клієнт-програма має відправити першою на сервер. Це є початкова точка виконання програми. Наступний тег `successors` надає перелік команд, які можуть бути виконані після поточної команди. Важливо зауважити, що презентація цієї інформації для користувача повністю залежить від клієнт-програми. Тег `successor` містить параметр `autocall`, який визначає – чи виконується наступна команда після закінчення виконання поточної команди автоматично. Така функціональність необхідна в разі, коли програма має виконувати деякий послідовний набір команд. Останній тег `predecessorParamReference` використовується для визначення параметрів попередньої команди, дані з якої треба передати в поточний параметр. Таке представлення допомагає передати дані з однієї команди до іншої в разі, якщо ці дані надходять з сервера. Ця особливість ставить XDEP окремо від протоколів, які не підтримують поняття стану [4]. Вона досягається через перенесення логіки послідовності виконання програми на клієнтську частину. Цей підхід виражає поняття сесії, який навіть сьогодні не підтримується у веб-сервісах.

Для того, щоб чітко зрозуміти зміст використання цієї функціональності розглянемо ще раз проект Fenestra. У зв'язку з тим, що клієнт програма працює в аудіо режимі, після виконання кожної з команд користувач від персональної консолі отримує голосове повідомлення про результат виконання запиту та переліком можливих наступних команд. Користувач голосом обирає команду для виконання. Як тільки пристрій успішно розпізнавав голосову команду, користувачу пропонувалось вказати дані для параметрів запиту. В разі, якщо параметри вимагали визначених даних, їх перелік зачитувався користувачеві. Коли всі голосові повідомлення інтерпретувались однозначно, клієнтська програма відправляла команду на сервер і процес починався спочатку. Табл. 1 містить перелік всіх тегів та коментарі до них, які використовуються в даній програмі.

Обробка помилок. XDEP стандартизує обробку помилок та надає наступний перелік стандартних помилок. Коди від 0000 – 0999 є зарезервованими кодами класу. В табл. 2 наведено коди найбільш характерних помилок.

Таблиця 1

Назва тегу	Опис
XDEPDriver	Базовий тег для опису протоколу
ProtocolName	І'мя протоколу
ServerName	І'мя серверу
startcommand	Стартова команда
Command	Мета опис конкретної команди
Name	І'мя команди
IsStatic	Визначає чи доступна команда в будь-який момент часу
Description	Опис команди
Parameters	Мета опис параметрів
ParamName	І'мя параметру
predecessorParamReference	Посилання на параметр з минулої команди
paramType	Тип параметру *integer|boolean|string|list;*
paramDefaultValue	Значення по-замовчуванню
maxValues	Максимальна довжина значення параметру
minValues	Мінімальна довжина значення параметру
validationRule	Правило верифікації значення визначене через регулярні відносини
Direction	Визначає як оброблюється параметр. Можливі варіанти: in|out|inout
Values	Тег з переліком перед визначених значень параметру
Key	Ключ переліку
Value	Значення переліку
Successors	Перелік команд, які доступні користувачеві після закінчення обробки поточної команди
CommandName	І'мя команди
Autocall	Чи автоматично викликається команда

Таблиця 2

Код помилки	Опис
0000	Помилка відсутня
0001	Невідомий запит
0002	Помилка формату атрибутів
0003	Помилка параметрів атрибутів
0004	Помилковий формат полів
0005	Відсутні закриваючі теги параметрів
0800	Неможливо відкрити сесію
0801	Користувач або пароль невірні
0802	Невідомий ідентифікатор сесії
0998	Серверна помилка
0999	Невідома помилка

Кожна команда вимагає введення різних параметрів цілого, строкового, або інших типів. Додатково, деякі параметри можуть мати не одне значення, а їх набір.

Повернемось ще раз до прикладу з велосипедними станціями з точки зору перспективи розробки протоколу на базі XDEP. Можна сміливо заявити, що 100 % користувачів кредитних карток володіють мобільним телефоном. Припустимо, що замість автоматичних пунктів, керівництво міста облаштувала платформи Bluetooth® серверами.

Актор (мандрівник) наближається до пункту з орендою велосипедів. На його мобільному телефоні встановлена програма які підтримує функції протоколів XDEP. Ця програма автоматично фіксує серверну платформу через Bluetooth® канали. В разі, якщо користувач вперше користується послугою, програма вимагає ввести деталі кредитної картки (можливо, якщо користувач повністю довіряє інформаційним системам, він вкаже номер кредитної картки, який програма зможе використовувати всюди), в іншому випадку, програма автоматично відправляє дані про взятий у поточний момент велосипед (час, номер та платформа оренди). Таким чином актор може продовжити свою подорож не змінюючи велосипед та не втрачаючи час на формальні процедури. До того, такий підхід дозволить знизити вартість обладнання нових платформ та їх майбутнє обслуговування. В разі, якщо користувач не має відповідної програми на своєму пристрої, вона може бути завантажена через ті ж безпроводові канали.

Висновки

Розглянуто еволюцію класу протоколів XDEP, який призвів до нової можливості конструювання інтерфейсів користувачів незалежно від типу та функціонального призначення мобільних приладів. Основною метою цього рішення була автоматизація процесів, необхідних для надання різноманітних послуг для різних груп користувачів. Ця мета була досягнута через розробку класу платформо-незалежних протоколів. У роботі представлено два приклади, один з яких - програмний проект Fenestra розроблений для конкурсу Microsoft Imagine Cup 2007, а інший – розглянутий в перспективі застосування для вирішення недосконалостей існуючої системи *Velolib*.

Під час розгляду представленого підходу, автори зробили акцент на перевагах протоколу з точки зору користувачів. Однак, структурність протоколу також мотивує його технологічне застосування. Згідно агенції досліджень *Juniper Research*, кількість користувачів, які будуть активно використовувати їх мобільні пристрої до 2011 року складе 54 мільйони. А кількість їх трансакцій складатиме 11,5 $ мільярдів [5]. Це відкриває нові простори для застосування протоколу. Варто зазначити, що проекти використання бездротових методів передачі інформації, використовуючи магнітні носії, наприклад, карти, вже впроваджуються і на Україні. Останнім продуктом масового споживання таких розробок став Київський Метрополітен, якій ввів в використання безконтактні проїзні квитки.

1. *Doroshenko A., Yatsenko K.* Protocols for Mobile Devices Integration in Heterogeneous Environments, ISTA, 2007.
2. *Graham S., Davis D., Simeonov S., Daniels G.* Building Web Services with Java : Making Sense of XML, SOAP, WSDL, and UDDI (2ndEdition) (Developer's Library), 2004.
3. http://imaginecup.com/MyStuff/MyTeam.aspx?TeamID=4541
4. *Gundavaram S.* CGI Programming on the World Wide Web (Nutshell Handbook), 1996
5. *Foggon D., Maharry D., Ullman C., Watson K.* Programming Microsoft .NET XML Web Services (Pro-Developer), 2003.