УДК 681.3

В.А. Дерецький

ПІДХОДИ ДО КОМПОЗИЦІЇ СЕРВІСІВ В СЕМАНТИЧНОМУ WEB СЕРЕДОВИЩІ

Технології мережних сервісів дозволяють програмним системам забезпечувати гнучкий і динамічний механізм інтеграції автономних систем програмного і інформаційного забезпечення в розподіленому мережному середовищі з ціллю вирішення задач користувача. Розробка та моделювання застосувань і інструментів для напів та автоматичної композиції і аналізу сервісів потребує врахування їх семантичних і динамічних властивостей. Ця робота представляє короткий огляд основних понять, моделей і засобів, що лежать в основі композиції сервісів.

Вступ

Парадигма мережних сервісів спрямована на забезпечення побудови та гнучкої, динамічної взаємодії різнорідних сервісів у мережі. Мережний сервіс – це програмна система яка ідентифікується URL, має інтерфейс який використовує засобами XML. Сервіс може бути доступним іншим програмним системам. У свою чергу ці системи можуть взаємодіяти з мережним сервісом (W3C) [1].

Модель мережного сервісу традиційно складається з трьох частин, постачальника сервісу, реєстратора сервісу і користувача сервісу. Постачальник сервісу створює або пропонує мережний сервіс, він же описує мережний сервіс у стандартному форматі XML і публікує його в центральній службі реєстрації. Служба містить додаткову інформацію про постачальника сервісу, наприклад, адреса і контактні реквізити компанії яка забезпечує сервіс та інші технічні деталі сервісу. Користувач сервісу отримує інформацію від служби реєстрації і використовує отриманий службовий опис, щоб зв'язати сервіс з іншими сервісами, або виконати визначені функції сервісу.

Для того, щоб досягти коректної комунікації сервісів, що виконуються на різних платформах і написаних на різних мовах програмування, потрібні стандарти. Створені стандарти SOAP, WSDL, BPEL та інші, розроблені технології та засоби фірми IBM – WebSphere Toolkit, фірми Sun-Open Net Environment and JiniTM Network technology, фірми Microsoft-.Net та фірми Novell-One Net initiatives, HP має розробку e-speak, фірма BEA-WebLogic Integration та багато інших [2]. Окремі зусилля спрямовані на дослідження та створення надбудови до зазначених стандартів та засобів, включаючи такі як DAML-S/OWL-S [3–5], які спрямовані на забезпечення підвищення рівня інтелектуальності сервісів, зокрема, автоматичного пошуку, автоматичної композиції, забезпечення ресурсами та моніторинг веб-сервісів, що спільно працюють для досягнення визначених користувачем цілей.

Композиція веб-сервісів обумовлена ситуацією, коли запит клієнта неможливо виконати єдиним сервісом, але можливо виконати за допомогою об'єднання певних доступних сервісів у мережі.

Окремі задачі композиції сервісів можна сформулювати таким чином:

визначити моделі та методи для моделювання мережних сервісів та їх композиції;

визначити моделі та методи для підтримки автоматизованої та автоматичної композиції;

визначити моделі та методи для забезпечення управління складовими мережних сервісів у процесі їх виконання;

забезпечити поєднання новітніх підходів та засобів з існуючими стандартами мережних сервісів.

Мета цієї роботи – огляд моделей та засобів, що становлять основу композиції сервісів та пов’язаних з нею задач.

Для виконання запиту користувача сукупністю мережних сервісів їх перш за все необхідно знайти серед чисельної кількості сервісів у мережі та далі забезпечити

управління і координацію їх виконання. Управління взаємодією сервісів, або оркестровка, має забезпечуватись потоками інформації, що міститься в повідомленнях, якими обмінюються сервіси.

Загальну модель композиції сервісів, можна представити у трьох вимірах, кожен з яких має свою метрику [6]. Вимір складності композитного сервісу визначає інформаційну місткість мов і моделей представлення сервісу. Наприклад, мова OWL-S в цьому вимірі має не високу міру, тому що засобами цієї мови сервіс описується тільки параметрами вхідної та вихідної інформації. Мова WSDL, що описує структурну інформацію сервісу засобами XML, більш інформативна, але також використовує тільки вхідну і вихідну інформацію. Другий вимір описує внутрішню структуру сервісів на рівні процесів та представляється такими моделями, як BPEL, CSP і p-Calculus, модель Міля, Римська модель, мережі Петрі [7–10]. Ці моделі описують стани сервісу і відповідно послідовність активностей сервісу. Третій вимір – це здатність описувати "семантику". В цьому вимірі мова OWL–S описує властивості сервісу на рівні входів та виходів, а також конкретизує, як сервіс взаємодіє з абстрактною моделлю "реального світу" [11]. Представлення сервісів у такому «кластері» моделей робить вищезазначені засоби дуже потужними.

У роботі розглянемо деякі з перелічених стандартів та моделей, спрямованих на вирішення задач композиції сервісів. Робота структурована таким чином. Розділ один містить огляд особливостей окремих стандартів, які використовуються в задачах композиції мережних сервісів. Розділ два містить опис задач та підходів до композиції мережних сервісів. Розділ три містить опис підходів до моделювання та аналізу автоматичної композиції мережних сервісів з використанням OWL-S та PSL моделей. Розділ чотири містить опис моделей, що лежать в основі автоматичної композиції. Розділ п'ять містить стислий підсумок та напрямок подальших досліджень моделей, методів композиції та аналізу мережних сервісів.

1. Особливості стандартів мережних сервісів

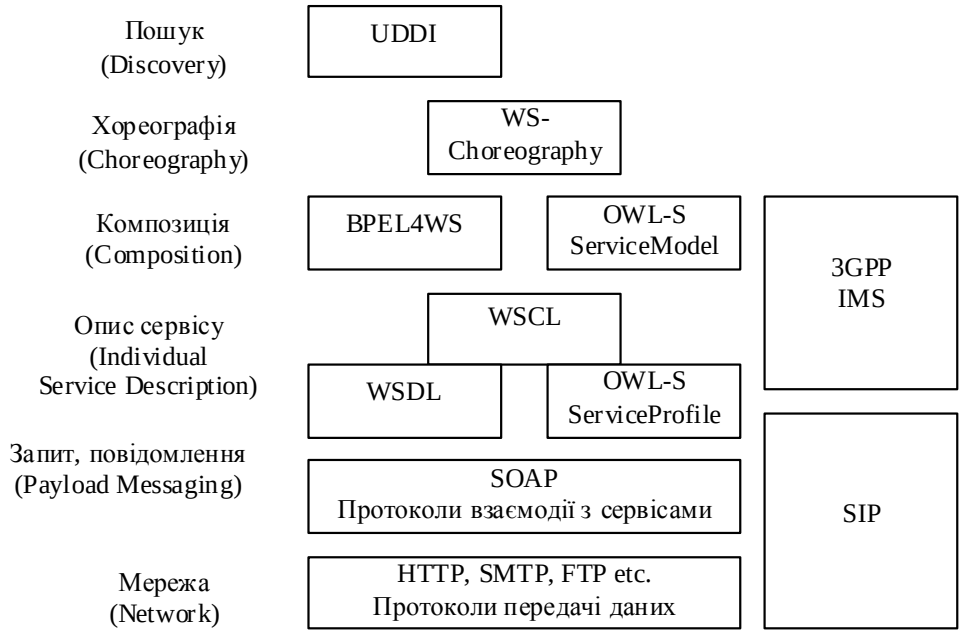
Для розуміння парадигми мережних сервісів у контексті мети композиції розглянемо систему стандартів. Основна причина їх створення, як наприклад, SOAP і WSDL, полягає в забезпеченні певного ступеня гнучкості при об'єднанні мережних сервісів для створення більш складних комбінацій з урахуванням динаміки середовища. Основна мета впровадження стандарту UDDI – створення засобів, які мають забезпечити автоматизований пошук і, тим самим, полегшити створення складових для композиції мережних сервісів. Стандарт BPEL забезпечує основу для ручної специфікації композиції мережних сервісів, використовуючи процедурну мову, яка забезпечує координування активностей мережних сервісів.

Більш потужні можливості мережних сервісів створюються шляхом залучення засобів DAML-S, OWL-S [11]. Мета використання цих засобів полягає в забезпеченні зручних для читання машиною описів мережних сервісів, які дозволять автоматизувати пошук, ведення переговорів, ініціювання та виконання сервісів, здійснювати моніторинг сервісів у процесі їх активності. Мова OWL-S – мова онтологій, що використовується для опису мережних сервісів у термінах їх входів, виходів, передумов і умов отримання результатів, а також процесу виконання. Важливим в цьому стандарті є те, що OWL-S забезпечує формальний механізм для моделювання "реального світу" і опису того, як взаємодіють між собою окремі мережні сервіси. Стандартами також забезпечуються механізми для моделювання специфікації сервісів у нотації OWL-S.

Модель існуючих стандартів мережних сервісів представляють шаровою структурою, яку показана на рис. 1. Мережні сервіси взаємодіють за допомогою передачі повідомлень у форматі XML з використанням типів. Стандарт SOAP використовується як протокол комунікації і підпису для мережних сервісів, які визначає

стандарт WSDL. Функціональні описи мережних сервісів визначаються з використанням вищих горизонтальних стандартів, наприклад, таких як, BPEL, WSCL, BPML, OWL-S.

Стандарт WSDL можна розглядати як продовження традиційних засобів, що специфікують вхід-вихід у розподіленому обчисленні до peer-to-peer структур [12]. Сервіс у даному випадку розглядають і як сервер



1. Відкриті стандарти та засоби, що покладені в основу архітектури SOA

Запит XML і мережні рівні стандартів забезпечують основу для інтероперабельності або взаємодій між сервісами. Базовим стандартом для опису кожного сервісу є мережна мова опису сервісу WSDL [1]. Стандарт WSDL описує мережний сервіс через набір зовнішніх операцій. Вони можуть бути представлені через повідомлення, або описом множини повідомлень, які можуть бути посланими або отриманими сервісом. WSDL конкретизує "реакцію" при отриманні повідомлення. Якщо реагуюча дія оголошена "односторонньою", сервіс не повертає відповідь, інакше він діє за правилом «запит-відповідь». Тип повідомлення, який повертається, також декларується. Таким чином описуються дії, які посилають повідомлення, що виходять від сервісу. Дії "оповіщення" посилають повідомлення без очікування відповіді, водночас як «дія опитування» чекає відповіді, з типом відповіді, що конкретизується цією дією. Повідомлення, що надходять, та відповіді визначаються в конкретній XML схемі, яка використовується в повідомленнях сервісу. (через реагуючі дії), і як клієнт (через упереджуючі дії).

Модель сервісів, що викладена засобами WSDL, по суті не містить опису станів виконання сервісу. Стандарт WSDL 2.0 пропонує обмежену нотацію стану, яка дозволяє специфікувати визначені зразки повідомлень, які сервіс має задовольнити. Поки що внутрішній стан сервісу не визначається, але деяким режимам при виконанні сервісу потрібно контролювати цей стан (наприклад, щоб взаємодіяти коректно з іншими сервісами). Мова WSCL [13] призначена для визначення вхідного та вихідного повідомлення сервісу, за суттю є обмеженим кінцевим автоматом над азбукою типів повідомлень. Такий автомат у контексті сервісів називається розмовою.

Необхідно зазначити, що WSCL надзвичайно добре співпрацює з WSDL. Опис WSDL разом з переговорами сервісів WSCL використовують семантику сервісу, але тільки на період, коли зберігається стан внутрішнього виконання сервісу («короткою пам'яттю»).

Стандарти WSDL і WSCL тільки визначають мережні сервіси. Для композиції сервісів необхідна мова на відповідному рівні абстракції. Мова BPEL розроблена як для «ручного» програмування мережних сервісів, так і для їх специфікації. Ідентифікують дві топології сервісів: “peer-to-peer” – кожен з кожним, і “hub-and-spoke” – з медіатором (посередником). Медіатори – це сервіси-посередники, які забезпечують координацію активностей інших мережних сервісів. Первинна мета BPEL була такою, щоб забезпечити мову для конкретизації поведінки сервісу посередника, а не як засіб, що призначений для універсального програмування.

У противазі до BPEL, WS-хореографія [13] намагається конкретизувати на глобальному рівні спосіб поведінки складових сервісів. WSCDL робить наголос на аспектах “хореографії”: ролі сервісів, інформації, що передається між сервісами, каналами, які забезпечують потоки інформації.

У вершині стандартів визначений стандарт UDDI, який забезпечує реєстрацію сервісів у різних застосуваннях [2]. Сервіси, що зареєстровані, можуть бути знайденими, якщо їх реєстрація забезпечена цим стандартом.

2. Задачі та моделі композиції сервісів

Ідентифікують такі проблеми композиції сервісів.

Координація мережних сервісів. Інфраструктура мережного сервісу задовольняє вимогам простої взаємодії мережних сервісів. Але для композитного сервісу та складних сервісних застосувань необхідно координувати послідовність операцій, що виконуються різними сервісами для того, щоб гарантувати коректне та надійне їх спільне виконання. Потрібні нові протоколи і абстрактні моделі, які забезпечать точну координацію сервісів. Необхідно забезпечити моделювання і спростити розробку композитних мережних сервісів у частині їх координації. У цьому напрямку спрямовані зусилля виробників на створення стандартів, наприклад, WS-координація фірми IBM або WS-CF фірми Sun [14–16].

Управління транзакціями взаємодії сервісів. Для реалізації схеми координації сервісів застосовують протоколи управління транзакціями (WS-транзакції), які забезпечують короткотривалі операції, які викликані атомарними процесами, такими, що забезпечують довготривалі операції, що викликані бізнес процесами. Проблема полягає у тому, що в разі виконання довготривалої операції не завжди можливо гарантувати такі властивості композитного сервісу як атомарність, цілісність і стійкість виконання операцій. У цьому сенсі необхідно розширити можливість протоколу транзакції. Наприклад, будувати WS-транзакції щодо схеми WS-координації для протоколів централізованої та одноранговой транзакції [14].

Моделі активності сервісу описують приховані та явні транзакції між станами сервісу. Для моделювання транзакцій, використовуються властивості активації, які описують засоби запуску транзакцій, властивості операції які конкретизують результат аварійного завершення сервісу і властивості резервуваня ресурсів протягом даного часу.

У цьому напрямку розроблені абстрактні моделі активності сервісу та моделі завершення, вони містять операції компенсації дій у випадку аварійного завершення виконання. Наприклад, викликається функція, яка відміняє виконання сервісу або резервування ресурсу.

Контекст взаємодії сервісів. Термін контекст має багато визначень. Щодо мережних сервісів, контекст визначається як деяка додаткова інформація, яка використовується мережним сервісом або кількома сервісами, щоб забезпечити необхідну поведінку при виконанні сервісу. Контекст може містити інформацію, наприклад, ім'я користувача, адреса, розташування, пристрої, які використовуються, програмне забезпечення, засоби комунікації тощо. Контекст може визначати середовище виконання сервіса та розширюватися новими типами інформації у будь-який час (динаміка середовища) без змін інфраструктури композитного сервісу.

Стандарт WS конкретизує контекст у частині його сумісного використання сервісами та засоби управління контекстом [16].

Моделювання розмови. Модель розмови забезпечує динамічне зв'язування сервісів, створення схеми композиції сервісу, перевірку коректності композиції сервісу.

Для реалізації розмови мережних сервісів використовуються засоби WSCL (Web Services Conversation Language), WSCI специфікація (Web Service Choreography Interface), а також WS-координація і WS-транзакції. Розроблені абстракції моделюються взаємодії сервісів, наприклад, з використанням машин Миля [6, 16–18]. Взаємодія здійснюється через асинхронні повідомлення з підтримкою черг повідомлень.

Контроль за виконанням сервісів. Моніторинг. Засоби визначення стану композитного сервісу в процесі його виконання дають змогу втрутитися в процес виконання з метою забезпечення коректності та надійності виконання сервіса щодо зміни середовища. Відрізняють централізоване і розподілене виконання композиційних веб-сервісів. Централізоване виконання подібне парадигмі клієнт-сервер. У даному випадку сервер є центральний планувальник, який контролює процес виконання складових сервісів. Наприклад, засоби e-flow працюють за принципами централізованого планування [19].

Розподілена парадигма припускає, що зв’язані мережні сервіси розділяють контекст їх виконання. Кожний з хостів, на якому виконується сервіс, має власного координатора, який співпрацює з координаторами решти хостів, щоб гарантувати правильний порядок виконання складових сервісів. Використовується також гібридна форма, яка включає розподілену та централізовану парадигми управління. В цьому випадку сервіс може бути координатором, та контролювати не один, а набір композитних мережних сервісів.

Пошук мережних сервісів. Для пошуку мережних сервісів пропонуються моделі, засновані на обмеженнях основної моделі пошуку мережних послуг – UDDI. Стандарт UDDI містить тільки функціональні характеристики і в загальному випадку не повністю використовуються. Розширення існуючої моделі пошуку здійснюється шляхом додавання додаткових сутностей сервісу. В цьому випадку процес публікації, пошуку і зв'язування мережного сервісу такий самий, але збільшуються можливості пошуку. Коли користувач шукає сервіс, він може крім функціональних характеристик використати додаткову інформацію і отримати більш релевантний результат пошуку.

Модель специфікації сервісу містить тако ж якісні параметри сервісу, які використовуються для пошуку: маштабованість; ємність; потужність (час відповіді, затримка, пропускна здатність); надійність; гнучкість; виключення; точність; спосіб підтримки транзакцій; стандарти, що підтримуються; стабільність; вартість; повнота; безпека (аутентифікація, конфіденційність, можливість трасування, аудит, криптування даних, безвідкатність, тощо).

Якісні параменти сервісу орієнтовані на застосування, що побудовані на P2P архітектурах.

3. Підходи до композиції сервісів

Умовно підходи до композиції сервісів розділяються на два основних класи – статичний і динамічний. Статичний підхід до композиції є еквівалент ручної композиції, яка виконується на етапі проектування сервісу. Динамічні стратегії композиції мережних сервісів використовують час.

Розрізняють п'ять категорій композиції: що орієнтується на модель і на бізнес-правила, декларативна, автоматична і семантичних веб-сервісів композиції;

Статична та динамічна композиція сервісів. Статична композиція використовується на етапі проектування, коли плануються архітектура і проект системи. Вибираються компоненти, які зв'язуються разом, далі вони компілюються і розгортаються для виконання. Такий підхід відмінно працює доти, доки компоненти сервісів не змінюються. Такі засоби як Biztalk фірми

Microsoft і WebLogic фірми Bea – це приклади статичних засобів композиції.

Якщо бізнес-процеси змінюються, надають нові послуги або замінюють старі, можуть виникнути несумісні випадки. У такому разі, це веде до зміни архітектури програмного забезпечення й необхідності використовувати інші сервіси або навіть замінити систему.

В ідеальному випадку процеси сервісу мають прозоро адаптуватись до середовища, яке змінюється, до вимог середовища та користувача з мінімальним ручним втручанням.

Розвинутою платформою яка сфокусована на динамічну композицію сервісів є засоби Web Services Composition Platform фірми Sun, засоби e-flow фірми HP та інші [15, 17].

Композиція сервісів, що заснована на моделях управління (Model driven service composition). Процесс розробки композитного сервісу можливо представити чотирма етапами: визначення сервісу, планування, конструювання, виконання.

Початковою фазою визначення сервісу є абстрактна метамодель сервісу, яка моделює компоненти сервісу та відношення між ними. Цілями такого моделювання є вибір елементів композитного сервісу та правил їх композиції.

Далі композиція сервісу зосереджується на формуванні моделі управління потоками робіт.

Композиція сервісів, яка заснована на бізнес-правилах. Вона визначається шляхом додавання бізнес-правил, що регламентують: перед- та після- умови; опис подій, що мали місце при виконанні процесів композиції сервісу; потоку процесів; визначення та блокування активностей; визначення повідомлень, що містять вхідну та вихідну інформацію; опис ролей у процесі композиції сервісів.

Правила композиції сервісів можливо моделювати з використанням мови OCL (Object Constraint Language). Бізнес-правила можна класифікувати таким чином:

- структурні правила для формування процесів та структури;
- правила для планування пріоритету процесів у композиції;
- правила управління даними та повідомленнями;
- правила поведінки для управління процесами та забезпечення цілісності композитного процесу;
- правила використання ресурсів, вибору сервісів, провайдерів;
- правила для визначення виключень у поведінці сервісів;
- правила ролей для управління учасниками, залученими у сервісі;
- правила повідомлення для регулювання використовування інформації;
- правила для управління поведінкою композиції сервісу відповідно щодо очікуваних або непередбачених подій та інші.

Для опису абстракції високого рівня використовують мову UML (Unified Modelling Language), яка забезпечує опис бізнес-процесів, засоби підключення до інших стандартів, таких як BPEL4WS. Для формування бізнес-правил та опису потоку процесів використовується мова OCL, засобами якої описуються умови вибору сервісів та їх зв’язування, формування структури композитного сервісу і формування плану композиції сервісу.

Декларативна композиція сервісів. Більшість підходів засновані на тому, що спочатку мають бути створені бізнес-процеси. Декларативний підхід суттєво відрізняється від підходів, що засновані на бізнес-правилах та має дві фази: початкова фаза полягає в визначенні цілей композиції та конструювання планів для кожної цілі. Наступна фаза полягає у тому, що для кожного плану здійснюється пошук відповідних сервісів та будується схема потоку робіт.

Перша фаза реалізується на основі PDDL (Planning Domain Definition Language) та засобів XML Web services Request Language, які мають забезпечити машино читані семантичні об’єкти та специфікацію абстракції поведінки сервісу. Наступна фаза може бути реалізована на основі існуючих мовних засобів моделювання таких як BPEL.

З точки зору архітектури декларативний підхід потребує перегляду концепції опису сервісів та реєстру сервісів. Наприклад, засоби WSDL не забезпечують опис необхідних атрибутів сервісу, а засоби UDDI не містять інформацію про те, що робить сервіс. Цільовий, або декларативний підхід забезпечується мовою опису композитного сервісу, яка відображає запити користувачів, визначає нові типи ресурсів, функції та їх відносини на основі унікальних імен URI. Для реалізації такого підходу необхідно розробити модель координації сервісів та універсальний протокол взаємодії.

Автоматична композиція сервісів. На відміну від ручної композиції сервісів, автоматична композиція потребує використання онтологій [16]. Відповідно щодо сервісів, онтологія визначається, як множина об'єктів, які розділяють ту ж саму предметну область, та правила, за якими сервіси можуть бути описані та доступні. Онтологічні мови, такі як RDF, DAML-S (DARPA Agent Markup Language for Web services), DAML+OIL, або OWL (Ontology Web Language) забезпечують формальні специфікації і можуть описувати значення для складних класів властивостей. Засоби DAML-S забезпечують механізм для онтологічної організації сервіса. В якості типів онтологій, необхідних для композиції сервісів є такі: метрик, одиниць виміру, одиниць коштів, властивостей, методів та інші.

Модель автоматичної композиції сервісу потребує семантичного опису сервісу щодо його можливостей до взазаємодії [20].

WSDL не забезпечує семантичного опису сервісу. На рис. 2 показані розширення онтології сервісу семантичними характеристиками. Світлі об'єкти визначають існуючі характеристики, сірі об'єкти визначають додаткові семантичні характеристики сервісу.

Операції сервісів, що містяться у вхідних та вихідних повідомленнях, визначаються своїм найменуванням та іншими характеристиками. Після розширення онтології семантичними характеристиками стає можливим використати її у моделі композиції сервісів. Для цього вводять синтаксичні та семантичні правила.

Синтаксичні правила визначають спосіб композиції сервісу. Вважають, що два сервіси є придатними до композиції, коли, з одного боку, кожен клієнт або сервер на яких розміщені сервіси, мають бути зв'язані каналами зв'язку, а з другого боку сервіси мають бути погоджені з відповідним транспортним протоколом обміну повідомленнями. Семантичні правила забезпечують композиційність повідомлень, композиційність операційної семантики, якісну композиційність та композиційність надійності.

Композиційність повідомлень, що надходять від клієнта, заснована на визначенні типів повідомлень, які використовуються в параметрах запуску сервіса і мають бути сумісними. Повідомлення сумісні, якщо вони сумісні за типами даних.

Операційна композиційність має місце тоді, коли предметні області інтересів сумісні або синонімічні і, якщо операції забезпечують ті ж сімі характеристики.

Якісна композиційність має місце коли сервіси мають тіж самі якісні характеристики, наприклад, безпеки або прав доступу до ресурсів.

Для композиції сервісів вводять поняття схеми композиції, яка забезпечує композиційну надійність. Композиційна схема зв'язана з композиційним сервісом та використовується для порівняння додаткових значень. Наприклад, два незалежні сервіси, такі як замовлення на видання книжок та замовлення авіаквитків можуть бути композитними в разі, коли автори книг замовляють квитки для перельоту на місце їх видання для узгодження справ.

Процес автоматичної композиції включає наступні процедури: специфікації, розмітки, вибору, генерації. Для цього використовують різні розширення WSDL. Правила композиції інтегруються на фазі визначення. Вони використовуються для генерації композиційних планів, які задовольняють вимогам композиції. Здійснюється вибір найбільш відповідних композиційних планів, визначаються обмеження.

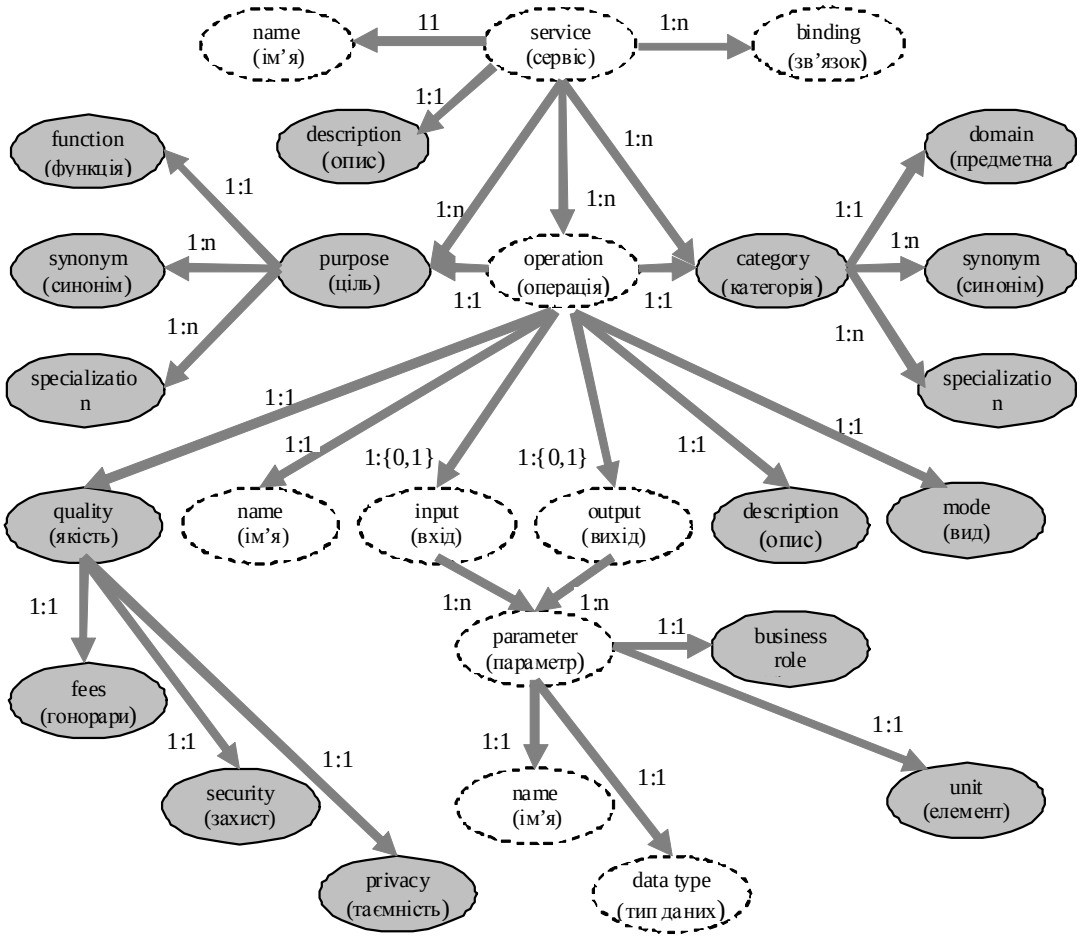


Рис. 2. Розширення онтології WS

Далі генерується результуючий композиційний сервіс.

4. Моделі, що лежать в основі автоматичної композиції

Теоретичне вивчення автоматичного синтезу композиції мережних сервісів знаходиться на початковій стадії. Моделі, що лежать в основі автоматичної композиції, мають свої корені в штучному інтелекті, логіці, обчислення ситуацій, теорії автоматів і тому залежать від вибраної філософської основи. Початкові результати з автоматичної композиції мережних сервісів групуються навколо трьох моделей: (a) модель OWL-S [5], яка включає модель атомарних сервісів та модель їх взаємодії з абстракцією "реального світу", (b) Римська модель [17], яка використовує абстрактну мову для представлення сервісів шляхом опису потоку процесів, що сформовані відповідними сервісами, і (c) модель машини Миля [6], яка сфокусована на управлінні потоками процесів.

Для представлення основних аспектів композиції введемо деякі обмеження. При взаємодії мережних сервісів, має бути забезпечена повна топологія сервісів та визначений протокол розмови між ними. Взаємодія сервісів має бути заснована або на використанні сервіса-посередника, або на використанні структури peer-to-peer (кожен з кожним). Якщо взаємодія сервісів заснована на використанні посередника, то це означає, що існує один мережний сервіс, який названо посередником, і який контролює дії решти сервісів, визначених у моделі. В одноранговій моделі кожен сервіс взаємодіє з іншими безпосередньо.

Слід розрізняти типи композиції, які призначені для разового або багаторазового використовування: у першому ви-

падку алгоритм композиції викликається і виконується кожного разу, коли клієнт ідентифікує цільовий сервіс. У другому випадку алгоритм композиції сервісів може бути виконаний один раз, а викликатися кілька разів тими самими або іншими клієнтами.

Синтез сервіса-посередника на основі атомарних складових сервісів з використанням моделі OWL-S заснований на побудові WorkFlow схеми (наприклад, календарний графік процесу, мережі Petri, складовий OWL-S сервіс тощо) [18–20], яка становить основу моделі управління, в якій атомарні сервіси викликаються для того, щоб координуватись для досягнення конкретної мети в межах визначеної конструкції. Ця WorkFlow схема має координуватися посередником і на практиці може використовувати стандарт BPEL.

Задачі синтезу посередника досліджуються також з використанням римської моделі, в якій складові сервіси є загалом неатомарні і мають кожний свою схему потоку робіт, на відміну від моделі OWL-S, в якій атомарні сервіси абстрагуються від внутрішньої структури.

Більш детально зупинимось на використанні OWL-S моделей. Розглянемо основні аспекти моделі OWL-S та деякі результати композиції, що отримані з використанням цієї моделі, а також підходу, заснованому на онтології числення предикатів першого порядку для мережних сервісів. Моделі OWL-S відіграють основну роль в процесі моделювання взаємодії мережних сервісів з “реальним світом”.

Базовий блок побудови моделі OWL-S – це нотація “процесу”, який включає як атомарні, так і складні процеси. Процеси в нотації OWL-S конкретизовані з точністю до входів, виходів, вхідних умов і умов результатів. Розглянемо приклад моделі OWL-S для двох атомарних сервісів, узятих з області комерції (купівліпродажу товарів).

У прикладі клієнт має використовувати сервіс Select_goods для перевірки наявності певної кількості товару, і якщо результат його задовольняє, то зарезервувати цей товар для послідуючої покупки. Перевірити свої фінансові можливості та придбати зарезервований товар, використовуючи сервіс Purchase_goods. У нотації OWL-S ці сервіси мають вигляд, який показано на рис. 3.

Після успішного виконання сервісу *Select_goods* видається повідомлення, що ці товари будуть зарезервовані для покупки (наприклад, протягом 10 хв.). У випадку успішного резервування клієнт може звернутися до сервісу купівлі *Purchase_goods*, щоб фактично зробити покупку, використовуючи банківський рахунок для проплати. Сервіс *Purchase_goods* виконуватиметься, поки резервування дійсне, але результат цього виконання залежить від того, чи є достатні фінансові ресурси на рахун-

Select_goods
Input: goods_name, quantitu
Output: price, reservation_id
Pre-condition: the quantity of goods is available for sale
Conditional effects:
If true, then modify world state to reglect the fact that this goods is being for sale

Purchase_goods
Input: reservation id, billable_account
Output: confirmation_id
Pre-condition: reservation_id is valid and has not expired
Conditional effects:
If enough money ib billable_account then transfer of $$ to goods owner and transfer of goods to buyer
If not ehough money then make reservation_id void

Рис. 3. Приклад композиції сервісу в нотації OWL-S

ку. Складовий сервіс у нотації OWL-S може бути створений за допомогою об'єднання цих двох атомарних сервісів, використовуючи послідовну конструкцію. У даному випадку виконання складового сервісу може залежати від умов виконання кожного атомарного сервісу.

Для забезпечення формальної основи моделювання взаємодії сервісів з «реальним світом» використовується підхід, заснований на численні ситуацій, що моделює факти, які можуть виникнути в різних "ситуаціях" при виконанні сервісів [19]. Для специфікації сервісів використовується мова PSL – Process Specification Language [21]. ISO стандартизувало цю мову, яка об'єднує аспекти числення ситуації. PSL має рівневу структуру з PSL-ядром та багатьма розширеннями для моделювання процесів. PSL-ядро забезпечує предикати першого порядку, які можуть використовуватися для опису основних блоків процесів та їх виконання.

Основним класом у мові PSL є клас *activity – активність,* яка може розглядатися як модель процесу, вона може бути нульовою або складатися з кількох активностей. Предикат *occurrence_of (occ, a)* використовується для конкретизації процесу, а саме, що *occ* –результат активності *a***.** Час, в даному випадку, представляється як параметр запиту на виконання процесу, в якому враховується початок та кінець процесу.

Модель застосування створюється за допомогою об'єднання аксіоматики PSL з предикатами та пропозиціями, щоб сформувати теорію (логіку першого порядку). Модель цієї теорії може бути подана як дерево, вузли якого відповідають атомарним активностям, з переходами від однієї атомарної активності до іншої. Шлях у цьому дереві може розглядатися як можлива послідовність кроків виконання композитного сервісу.

Підхід до побудови проекції моделі OWL-S на PSL детально розглядається в [22].

Використаємо мову PSL для опису OWL-S нотації у прикладі покупки товарів. У цьому прикладі ми можемо визначити наступні відношення:

Available_goods(goods_name, qty, price) - запит наявності товару певної кількості

Reserved_goods(goods_name, qty, price, res_id) - резервування акцій

Purchase_reservation(res_id, start_time) - придбання зарезервованих акцій

Billable_account(acct_id, current_balance) - контроль наявності коштів, їх проплата

Використаємо вищезгадані відношення для того, щоб конкретизувати поведінку атомарних сервісів, кожен з яких можна розглядати як атомарну активність в PSL нотації. Наприклад, якщо відношення *select_goods* задаються двома параметрами *s, q* – (*назва, кількість*), то відношення *available_goods* представляється трійкою параментрів – *назва, кількість за запитом, ціна*.

$r = (s,\ q',\ p)$ при умові $q \geq q'$,

де *r*- відношення «запит наявності товару», *q* – кількість доступного , *p* – ціна, *q'* – кількість за запитом.

Результат у даному випадку завжди існує, якщо виконується вхідна умова.

На основі цього результату, може бути побудована активність резервування, яка має вигляд (*s, q – q', p, t*), де *s*– назва*; q* – кількість*; p* – ціна*; q'* – кількість за запитом; *t* – час резервування.

Є два підходи в контексті числення ситуацій. Згідно з першим підходом, відношення, наприклад, *available_goods*, перетворюють за допомогою додавання нового поля, в нашому прикладі це значення часу резервування. Згідно з другим підходом, що використовується в PSL відношення перетворюють у функцію, скажімо, *f_available_goods*, яка має три аргументи. Предикати PSL, володіння (hold) і попереджування (prior) використовуються, щоб зв'язати істинні значення з умовами використання цієї функції.

Сукупність складових для побудови атомарних сервісів, їх конструкції визначаються в засобах GOLOG [22, 23], і відповідають графічному стилю проектування, але в OWL-S вони розглядаються як декларативні обмеження в специфікації мережного сервісу.

Ми зазначили два ключових підходи, які стосуються автоматичної композиції з використанням моделей для OWL-S і OWL-S-подібних сервісів. Перший розроблений у численні ситуації, де аспекти моделі OWL-S представляються засобами PSL. Мета композиції конкретизується в термінах загального результату, який досягається на всіх функціях, починаючи від деякого початкового стану, який включає значення вхідного аргументу. Інший підхід полягає у перетворенні атомарних процесів OWL-S у схему WorkFlow, яка відповідає всім можливим послідовностям активностей процесів.

Обчислювальна складність визначення існування композиції є експоненціальною. Для зменшення складності можуть застосовуватися різні евристики.

Інший підхід до композиції сервісів у контексті моделі OWL-S розроблений в [11, 20, 22]. У цьому підході використовується мова GOLOG. Загальна програма може послідовно конкретизуватися, починаючи з атомарних сервісів OWL-S. Для побудови конкретної схеми, яка потрібна замовникові, можуть бути написані додаткові обмеження. Далі виконується композиція, яка відповідає гілці дерева схеми, і яка задовольняє певним властивостям. Для знаходження таких гілок використовується інтерпретатор ConGolog.

Висновки

Парадигма мережних сервісів ініціює дослідження та вирішення нових задач, серед яких є пошук відповідних моделей і абстракцій для представлення мережних сервісів та їх “поведінки”, і таких, що можуть підтримувати ефективне створення алгоритмів композиції та аналізу сервісів. Інший аспект торкається засобів, які виконують статичну та динамічну перевірку специфікацій мережних сервісів як на етапі компіляції, так і на етапі їх виконання з метою контролю цілісності застосувань.

Друга категорія задач полягає у тому, щоб прозоро забезпечити маніпулювання даними в контексті парадигми мережних сервісів і пов’язаних з нею стандартами. Існуючі стандарти і чисельні дослідження в даний час зосереджуються перш за все на моделі процесів, а не на потоці даних і маніпулюванні даними.

Потрібні методи та засоби для моделювання перетворення даних, що відбувається в мережному сервісі, і які дозволять здійснювати міркування про поведінку даних при виконанні складового мережного сервісу. Для цього необхідно створити моделі композиції сервісів, які були б більш відповідними для застосувань, і які спрямовані на обробку великих обсягів даних.

1. *Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D.* (2004) Web Services Architecture // W3C Working Group Note 11, February, W3C Technical Reports and Publications, http://www.w3.org/TR/ws-arch/
2. *Alonso G., Casati F., Kuno H., and Machiraju V.,* Web Services. Concepts, Architectures and Applications. Springer-Verlag, 2004. – P. 354.
3. *Berardi D., Calvanese D., De Giacomo G., Hull R., and Mecella M.*. Automatic composition of web services in Colombo // In Proc. of 13th Itallian Symp. on Advanced Database Systems, June 2005. http://www.ai.sri.com/WSS2005/final-versions/WSS2005_PP-Berardi.pdf
4. *Berardi D., Calvanese D., De Giacomo G., Hull R., and Mecella M*. Automatic Composition of Transition-based Semantic Web Services with Messaging // Technical report, University of Rome, ”La Sapienza”, Roma.– – Italy: 2005, April. – P. 613–624.
5. *Berardi Daniela, Calvanese Diego, De Giacomo Giuseppe, Hull Richard, Mecella_Towards Massimo.* AutomaticWeb Service Discovery and Composition in a Context with Semantics, Messages, and Internal Process Flow (A Position Paper). http://www.w3.org/2005/04/FSWS/Submissions/21/berardi-et-al-for-W3C-Frameworks-SWSs.pdf

6. *Hull R., and Su J.* Tools for Composite Web Services: A Short Overview. http://www.cs.toronto.edu/~libkin/dbtheory/ hullsu.pdf
7. *Bloomberg J.* The seven principles of service-oriented development // XML & Web Services, 3(5):32–33, 2002. Business Process Execution Language for Web Services (BPEL), Version 1.1. http://www.ibm.com /developerworks/library/ws-bpel
8. *Milner R.* Communicating and Mobile Systems: The pi – calculus. Cambridge University Press, 1999. – P. 161.
9. *Hull Richard.* Web Services Composition: A Story of Models, Automata, and Logics // http://ieeexplore.ieee. org/iel5/10245/32665/01530770.pdf?arnumber=1530770&htry=0
10. *Fu X., Bultan T., and Su J.* Analysis of interacting BPEL web services//In Proc. Int. World Wide Web Conf. (WWW), May 2004. http://citeseer.ist.psu.edu/706792.html
11. Semantic Web Services Ontology (SWSO). http://www.daml.org/services/swsf/1.0/swso/
12. *Haas H.* Architecture and future of web services: from SOAP to semantic web services// http://www.w3.org/2004/Talks/0520-hh-ws/, May 2004.
13. Web Services Choreography Description Language Version 1.0 (W3C Working Draft). 2004, December. http://www.w3.org/TR/ 2004/WD-ws-cdl-10-20041217/
14. *Freund T. and Storey T.* (2002a, 2002b) Transactions in the World of Web Services, Part 1. An Overview of WS-transaction and WS-coordination // http://www.106.ibm.com/developerworks/ webservices/ library/ws-wstx1
15. *Sun H., Wang X., Zhou B. and Zou P.* (Research and Implementation of Dynamic Web Services Composition//APPT 2003, LNCS 2834, Springer-Verlag Berlin Heidelberg. – 2003. – P. 457–466.
16. *Андон П., Дерецький В.* Проблеми побудови сервіс-орієнтованих прикладних інформаційних систем в semantic web середовищі на основі агентного підходу // Проблеми програмування. –2006. –№ 2–3. – P. 493-503.
17. Gösta Grahne and Victoria Kiricenko: Process Mediation in an Extended Roman Model. http://events.deri.at/ mediate2005/documents
18. Hamadi Rachid, Benatallah Boualem. A Petri Net-based Model for Web Service Composition. http://crpit.com/confpapers/CRPITV17Hamadi.pdf
19. *Bock Conrad, Gruninger Michael.* PSL:Asemantic domain for Flow models // Softw Syst Model. – 2005. – № 4. – P. 209 - 231.
20. *McIlraith Sheila, Cao Son Tran.* Adapting Golog for Composition of Semantic Web Services. http://citeseer. ist.psu.edu/ mcilraith01adapting.html
21. *McIlraith Sheila A., Cao Son Tran, and Zeng Honglei*. Semantic Web Services. Stanford University. http://portal.acm.org/citation.cfm?id=630625 &dl=&coll=&CFID=15151515&CFTOKEN =6184618
22. *Gr¨uninger Michael, Kopena Joseph B.* Planning and the Process Specification Language. http://portal.acm.org/citation.cfm?id=958677
23. *Lesperance Yves, Kelley Todd G., Mylopoulos John, and S.K. Yu Eric.* Modeling Dynamic Domains with ConGolog// In Conference on Advanced Information Systems Engineering. – 1999. – P. 365-380.

Отримано 13.04.2007

Про автора:

Дерецький Валентин Олександрович,
кандидат фізико-математичних наук,
провідний науковий співробітник.

Місце роботи автора:

Інститут програмних систем
НАН України, 03187, Київ-187,
проспект Академіка Глушкова, 40.
Тел. (044) 526 4342
E-mail:dva@isofts.kiev.ua