УДК 004.75

Б. В. Булах,
аспірант

WORKFLOW-СЦЕНАРІЇ ДЛЯ СЕРВІСНО-ОРІЄНТОВАНИХ СИСТЕМ МОДЕЛЮВАННЯ

Розглянуто завдання поєднання сервісно-орієнтованого підходу та workflow-концепції як архітектурного рішення для розподілених програмних систем математичного моделювання. Запропоновано підхід, який полягає у застосуванні стандартизованих засобів реєстрації та компоновки веб-сервісів, розширених підтримкою семантичного опису, для реалізації таких систем.

Ключові слова: сервісно-орієнтована архітектура, веб-сервіси, потоки робіт (workflow), семантичні технології.

Архітектура комплексів моделювання постійно розвивається як адекватна реакція на нові запити спільноти користувачів. Локальні версії програмних пакетів свого часу були доповнені клієнт-серверними можливостями, що надало можливість інженерам винести розрахунки на окремі обчислювальні сервери, а також колективно працювати над проектами. Сьогодні ж до програмних рішень у галузі інженерних обчислень (і не тільки) висуваються все нові вимоги, в числі яких: використання високопродуктивних обчислень, доступність через мережу Інтернет, легкість масштабування, розширюваність, сумісність із різними платформами, організація відмовостійкої роботи та ін.

Для відповіді на ці актуальні запити сьогодення програмним комплексам вже недостатньо можливостей традиційної клієнт-серверної архітектури, із цілісними серверним та клієнтським варіантами програмного забезпечення. Більшу гнучкість пропонує популярна нині парадигма сервісно-орієнтованої архітектури [1], концептуальні особливості якої дозволяють спростити організацію та підтримку складних розподілених систем (що, втім, все одно вимагає ретельного проектування архітектури таких систем для досягнення бажаних результатів), за рахунок упорядкування процесу розробки та функціонування, інтенсивнішого повторного використання коду, незалежності від конкретних платформ, слабким зв'язкам між компонентами.

Метою статті є розглянути існуючі підходи до організації складних сценаріїв інженерних обчислень у сервісно-орієнтованих середовищах, дослідити підхід, що дозволяє швидке проектування обчислювальних сценаріїв у вигляді потоків робіт для вирішення міждисциплінарних завдань із орієнтацією на прийняті веб-стандарти та з гнучкими можливостями розширення.

Розглядаючи сучасні поширені програмні засоби для інженерних та наукових обчислень, можна відзначити, що попри постійні вдосконалення, часто вони не є сервісно-орієнтованими за своєю природою. Так, добре відомий пакет Mathematica [2] залишається вірним традиційній архітектурі із «ядром» та клієнтською частиною, і максимальний «рівень декомпозиції» залишається на рівні окремих бібліотек. Хоча в пакет включено можливості із організації високопродуктивних обчислень (версія gridMathematica) та специфічні інтерфейси для роботи з іншими програмами (MathLink) як і підтримку більшості операційних систем. Орієнтація на архітектуру, збудовану навколо «ядра обчислень», має свої переваги у продуктивності і не вимагає революційних змін у програмному коді.

Однак тоді, коли важливим є відкритість архітектури, використання сторонніх розробок, модульність, розподіленість функціоналу на різних обчислювальних ресурсах, СОА може бути гнучкішим рішенням. Про перспективність СОА як архітектурного шаблону для систем інженерних та наукових розрахунків свідчать, наприклад, такі факти, як прийняття СОА як архітектуру для програмного забезпечення проміжного шару систем грід-обчислень (проекти Globus, Unicore, ARC[3], gLite) або ж розвиток систем швидкого композиціювання прикладних сервісів, таких як Taverna Project [4]. До того ж слід зазначити, що СОА-підхід здатен спростити завдання поєднання засобів різних пакетів від різних розробників, наприклад, для вирішення міждисциплінарних завдань, оскільки він базується на використанні стандартних відкритих інтерфейсів та протоколів.

Севісно-орієнтований підхід в архітектурі комплексів моделювання. Можна коротко узагальнити суть сервісно-орієнтованого підходу такими положеннями: стандартні інтерфейси компонентів (сервісів), описані у відкритих контрактах; прихована внутрішня логіка максимально автономних сервісів; декомпозиція на сервіси має враховувати перспективи максимального повторного використання; уникнення підтримки внутрішнього стану сервісів для забезпечення кращої масштабованості; стандартні шляхи пошуку сервісів їх споживачами; здатність до композиції сервісів задля виконання складних завдань.

Стандартною де-факто реалізацією СОА нині є веб-сервіси. Веб-сервіси реалізують засади СОА на базі ряду відкритих стандартів: стандарту опису сервісів WSDL, стандарту на реєстрацію та пошук UDDI, стандарту на взаємодію (передачу повідомлень) SOAP.

Веб-сервіси є достатньо перевіреним, зрілим рішенням для побудови розподілених систем, в тому числі і таких специфічних, як системи грід-обчислень. Нині більшість міжнародних грід-проектів перейшли або переходять на сервісно-орієнтовану архітектуру програмного забезпечення проміжного шару: Globus Toolkit 4, Unicore 6, Nordugrid ARC 11 [3], EGEE gLite 3 тою чи тою мірою використовують стандартні або спеціалізовані веб-сервіси (останні дістали назву грід-сервісів та дотримуються специфікацій WSRF). Практика показала, що веб-сервіси непогано узгоджуються із динамічною природою грід-систем, що оперують пулом гетерогенних розподілених грід-ресурсів: так, доступ до грід-завдань, сховищ даних тощо може бути організований через стандартні інтерфейси службових веб-сервісів; крім того, службові сервіси можуть взаємодіяти між собою для «колективного» виконання покладених на них функцій з управління завданнями, передачею даних, моніторингу та ін.

Подібний принцип побудови функціоналу як множини самостійних веб-сервісів, очевидно, може бути застосований і для програмних середовищ з інженерних обчислень. Така декомпозиція їх обчислювальних «ядер» дозволить надати доступ до їх функціоналу стороннім клієнтам, забезпечити простий динамічний механізм розширень за рахунок додаткових сумісних сервісів від сторонніх розробників, а також дістати вигоду від здатності веб-сервісів до компонування (див. далі).

Втім, не слід переоцінювати можливості СОА та вважати веб-сервіси панацеєю від проблем інтеграції та функціональної сумісності компонентів системи. СОА є лише шаблоном, за яким може бути збудована інфраструктура, як успішна, так і невдала. Веб-сервіси вирішують питання сумісності лише на синтаксичному рівні, спираючись на стандарти опису інтерфейсів, однак для досягнення повної сумісності між споживачем сервісу та самим сервісом необхідна організація сумісності і на семантичному рівні, тобто забезпечення однозначного розуміння усіма учасниками множини використаних понять, позначень, для чого потрібні додаткові засоби. Без цього, наприклад, проблемною є організація автоматичного пошуку сервісів програмними агентами.

Перехід на СОА також означає, як правило, додаткові накладні витрати на передачу повідомлень, виклик сервісів, а головне — змушує переносити традиційні програми на нові «рейки», що не завжди можливо та доцільно.

Методологія workflow у системах моделювання. Обчислювальні задачі, що вирішують інженери або дослідники, часто є складними сценаріями, які залучають чимало дій та мають складну логіку виконання із розгалуженнями, циклами, переходами за умовою та ін. Тож очевидною є потреба у наданні відповідного автоматизованого інструментарію, який здатен полегшити завдання проектування та виконання таких складних обчислювальних сценаріїв. Існує кілька підходів для досягнення зазначеної мети — від опису сценарію скриптовими мовами до представлення сценарію як орієнтованого графу, вершинами якого є базові дії, а дуги визначають порядок їх виконання та маршрути передач даних між кроками. Останнє представлення дістало назву потоку робіт (англ. *workflow* [5; 6]).

Достатнього поширення workflow-ідеологія набула в таких галузях, як обробка зображень, сигналів, інтелектуальна обробка даних (data mining), біомедицина та ін., де процес обробки даних відносно легко вибудовується із окремих, визначених процедур (генерація вхідних даних, фільтрація, аналіз, узагальнення, візуалізація та ін.). Завдання моделювання складних систем також можуть бути представлені у вигляді послідовності кроків (підготовка даних, побудова математичної моделі,

рішення отриманих систем рівнянь, оптимізація, візуалізація тощо). Прикладами мультидисциплінарних workflow-середовищ інженерних обчислень є такі проекти, як Triana Workflows [7] та Kepler Project [8].

В рамках workflow-моделі користувач може вільно компонувати обчислювальні кроки у певні маршрути, при цьому в ролі кроків можуть виступати локальні програми, віддалені програмні об'єкти, веб-сервіси чи інші програмні компоненти, описані належним чином та включені у бібліотеку конкретного середовища. На жаль, нині не існує визнаних стандартів на обчислювальні workflow-сценарії, а тому компоненти різних проектів, як правило, не сумісні між собою.

Окремо слід відзначити необхідність контролю коректності проектних рішень користувача, а також належної інформаційної підтримки проектувальника. Так, часто певні послідовності кроків у сценаріях моделювання не мають сенсу (побудова математичної моделі без вхідних даних), або, залежно від реалізації, є неможливими для виконання, хоча формально можуть бути побудовані. У разі міждисциплінарного середовища від користувача просто неможливо вимагати пам'ятати призначення та специфіку використання всіх наявних компонентів, тож завдання контролю та підказки має взяти на себе середовище проектування.

Сценарії числових експериментів як маршрути виконання веб-сервісів. Як вирішення вищезгаданої проблеми сумісності компонентів різних workflow-проектів можна запропонувати перевірене рішення — використання веб-сервісів як складових потоку робіт. Прикладом подібної системи може бути архітектура середовища Taverna Workbench [4]. Цей проект початково орієнтований на використання веб-сервісів як складових кроків робочих потоків (хоча деякі прості операції над даними можна виконувати локально).

До складу подібної системи входять такі елементи: графічний редактор (з більшою чи меншою свободою дій користувача, що визначається прийнятою моделлю робочого потоку), бібліотека доступних сервісів (із здатністю динамічного поповнення), менеджер виконання потоку (відповідальний за автоматичне виконання складеного користувачем за допомогою редактору опису потоку), візуалізатор результатів.

Перевагою на користь реалізації базових операцій потоків робіт як веб-сервісів може слугувати наявність стандартів та відкритих засобів не лише із розробки самих сервісів, а й з їх композиції (напр. WS-BPEL [9]). Такий підхід дозволяє максимально використати стандартні рішення та готовий інструментарій. Це відрізняє зазначений підхід, наприклад, від прийнятого у проекті Taverna, в рамках якого розроблені власні опис потоку та менеджер його виконання, що не спираються на стандарти OASIS. Загальна архітектура такої системи показана на рис.1.

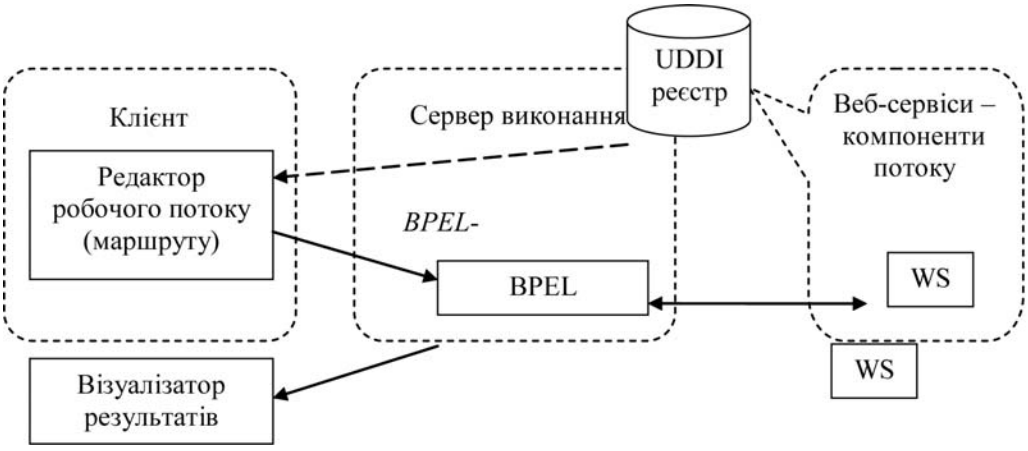

Рис. 1 — Архітектура BPEL-орієнтованої системи управління робочими потоками

Змішаний підхід на основі семантичного реєстру. Слід зазначити, що завдання представлення маршруту обчислень винятково з веб-сервісів як BPEL-сценарію є непростою і має свої вади, серед яких виділимо такі:

— мова BPEL базується на представленні робочого потоку як ациклічного графу та накладає додаткові обмеження на потік контролю, що означає прив'язку графічного редактора до специфіки цієї мови;

— без додаткових засобів завдання поєднання входів та виходів веб-сервісів перетворюється на трудомісткий розбір структур XML-повідомлень;

— декомпозиція програм як набору веб-сервісів, як уже зазначали, не завжди технічно можлива;

— занадто висока деталізація (у значенні ступеня декомпозиції) сервісів неоправдано підвищує інтенсивність мережевого обміну, тоді як низька деталізація значно обмежує свободу проектувальника;

— недосвідчений користувач може апріорі не знати, які веб-сервіси у якому порядку йому слід використати для досягнення своєї мети, що зводить нанівець усю зручність від графічного представлення.

Тож пропонуємо дещо модифікувати вищеописаний BPEL-підхід таким чином, щоб вирішити або згладити вищезгадані недоліки. Для цього пропонуємо відокремити шар веб-сервісів та шар компонентів потоку. Це призведе до таких змін в архітектурі системи.

Вводяться абстрактні складові кроки потоку замість реальних веб-сервісів. Потік представляється послідовністю «об'єктів-дій», кожен з яких описаний та занесений до бібліотеки компонентів. Перед виконанням абстрактний потік «транслюється» у BPEL-сценарій та передається на виконання до стандартного BPEL-процесора, далі виконання відбувається аналогічно попередньому варіанту.

Наявність подібного кроку «трансляції» (мається на увазі саме уточнення абстрактного потоку, а не переклад графічного представлення BPEL-сценарію у його текстове представлення, що було характерне для попереднього підходу) дозволяє, по-перше, дещо відокремитись від конкретики мови BPEL при створенні зручного графічного редактора потоків та не перевантажувати сам клієнтський редактор (відображення графу потоку на конкретну мову системи виконання потоку, таку як BPEL, стає завданням транслятора), по-друге, приховати складнощі роботи із повідомленнями (завдяки введенню абстрактних типів вхідних та вихідних даних, які далі неявно транслюються в конкретні операції з XML), по-третє, відійти від необхідності представлення усіх кроків потоку тільки як веб-сервісів. Останнє зауваження означає, що одному «об'єкту-дії» робочого потоку може відповідати один або кілька веб-сервісів, або навпаки — один веб-сервіс моделюється кількома об'єктами робочого потоку.

Реєстр компонентів потоку. Вищеописаний підхід, з одного боку, вирішує проблеми із складністю безпосереднього компонування веб-сервісів, з іншого — вимагає наявності окремого реєстру для «об'єктів-дій» та правил їх відображення на реальні веб-сервіси. Пропонуємо для описів абстрактних компонентів потоку використати стандартні мови семантичного опису: RDF, OWL-DL[10]. Запропонована розширена архітектура наведена на рис. 2.

Як уже зазначалося, питання автоматизованого пошуку та компонування складових потоку для workflow-систем є надзвичайно важливим, оскільки звільняє користувача від необхідності пам'ятати призначення конкретних компонентів та допустимі способи їх поєднання.

Стандартним вирішенням завдання реєстрації та пошуку веб-сервісів використовують реєстр UDDI. Втім він є відносно простим каталогом, що надає здебільшого інформацію про синтаксичні особливості виклику сервісів (тобто містить WSDL-контракти сервісів, які описують порядок, формат взаємодії із сервісом, але не характеризують його призначення, особливості його функціонування та поєднання з іншими сервісами, тобто — не містять семантичну [11] інформацію).

Чимало досліджень були спрямовані на вирішення завдання семантичного пошуку та композиції веб-сервісів [12; 13; 14; 15] та семантичне розширення стандартного UDDI-реєстру [16; 17]. Найбільш популярними підходами є використання структур даних UDDI, таких як tModel, для збереження семантичних описів сервісів; пропонуємо різні схеми відображення семантичних описів на різних мовах (OWL-S, WSMO) на поля базових структур UDDI.

Одним з підходів є т. зв. «зовнішнє» розширення реєстру, коли сам інтерфейс UDDI-реєстру лишається без змін, а для операції із семантичними даними надається додатковий API [18].

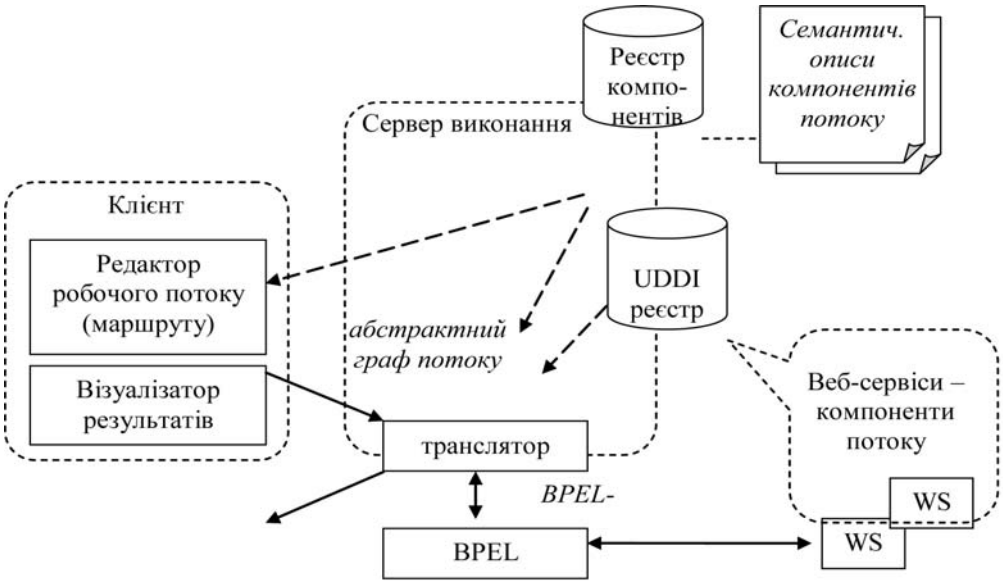

Рис. 2 — Розвиток архітектури системи управління робочими потоками

Що стосується окремого реєстру для представлень абстрактних компонентів потоків, то лише він може бути реалізований як семантичний, залишаючи стандартний реєстр веб-сервісів без змін для використання його передбаченим способом (як розвиток «зовнішнього» підходу до розширення UDDI). Процедура реєстрації нових компонентів потоку перетворюється на двокрокову: реєстрація веб-сервісів у UDDI-реєстрі із подальшою реєстрацією пов'язаних компонентів потоку у семантичному реєстрі та правил їх відображення на веб-сервіси з реєстру UDDI.

Загальний *сценарій роботи* із системою можна описати так.

1. Розробник нового веб-сервісу розгортає його на своїх ресурсах та реєструє у UDDI-реєстрі системи. Розробник також реєструє семантичні описи пов'язаних із його сервісом компонентів потоку (завдання, входи, виходи, допустимі правила композиції) в термінах прийнятої онтології у семантичному реєстрі. Коли абстрактні компоненти відображаються на веб-сервіси нетривіально, розробник надає сервіс трансляції опису абстрактного потоку на конкретний BPEL-потік для таких компонентів.

2. Графічний редактор (або інший програмний агент), що використовує прийняту онтологію компонентів потоку, дозволяє редагування абстрактного потоку робіт, використовуючи при цьому правила композиції з семантичного реєстру та автоматично поєднуючи входи та виходи компонентів згідно з цими правилами. Використання правил композиції дозволяє як уникнути недопустимих комбінацій компонентів, так і видавати підказку користувачу за можливими варіантами розбудови потоку. Програмні агенти потенційно можуть використати семантичну інформацію з реєстру для автоматичної композиції потоку під задані умови.

3. Згенерований у п. 2 абстрактний опис потоку у визначеному форматі передається транслятору, завданням якого є переклад цього опису на мову конкретної системи управління потоками робіт (в цьому разі — стандартну мову WS-BPEL). Транслятор використовує стандартні правила перекладу, однак якщо він не може перекласти описи деяких компонентів на BPEL-сценарій (наприклад, коли поєднанню абстрактних компонентів фізично відповідає різна кількість веб-сервісів їх реалізації), він шукає у реєстрах спеціалізований сервіс-транслятор для цього типу компонентів (зареєстрований у п. 1).

4. Результат роботи транслятора у вигляді BPEL-сценарію, де абстракні «об'єктидії» замінені викликами веб-сервісів, передається для виконання стандартному менеджеру управління BPEL-потоками (англ. *BPEL-engine*). Результати виконання потоку повертаються користувачу.

Практична реалізація та перспективи. Для практичної реалізації описаного підходу було використано такі засоби: контейнер веб-сервісів Apache Axis2 + jUDDI

реєстр, BPEL-процесор Apache ODE 1.3 [19]. Для перевірки семантичного підходу вирішувалось завдання контролю допустимих переходів між кроками потоку, для чого була розроблена тестова RDF-база знань, що підтримує SPARQL-запити, реалізована засобами пакету Jena [20]. Дослідження показало наявність численних відкритих засобів для розробки компонентів запропонованої архітектури. Це дозволяє продовжити експерименти включенням до системи спеціалізованих сервісів моделювання.

Висновки. Було розглянуто переваги та недоліки сервісно-орієнтованого підходу та workflow-концепції при проектуванні систем організації числових експериментів, в тому числі — засобів математичного моделювання. Запропоновано підхід, що дозволяє поєднати СОА та workflow-сценарії у системі, що максимально спирається на стандартні рішення. Запропоновано розвиток підходу, який полягає у залученні семантичних технологій для розширення можливостей системи проектування сценаріїв обчислень і забезпечення комфортнішої роботи користувача у ній, та описано шляхи його практичної реалізації.

Рассмотрена задача сочетания сервисно-ориентированного подхода и workflow-концепции как архитектурного решения для распределенных программных систем математического моделирования. Предложен подход, который заключается в применении стандартизированных средств регистрации и компонования веб-сервисов, расширенных поддержкой семантического описания, для реализации таких систем.

Ключевые слова: сервисно-ориентированная архитектура, веб-сервисы, потоки работ (workflow), семантические технологии.

Література

1. *T. Erl.* Service-Oriented Architecture: Concepts, Technology & Design. — New York: Prentice Hall/PearsonPTR. — 2005. — 792 p.

2. Wolfram Mathematica: Technical Computing Software. — http://www.wolfram.com/mathematica/

3. Advanced Resource Connector. — http://www.nordugrid.org/arc/

4. Taverna Workflow Management System. — http://www.taverna.org.uk/

5. *V. Curcin, M. Ghanem.* Scientific workflow systems — can one size fit all? // Proceedings of Biomedical Engineering Conference CIBEC 2008. — Cairo International. — 2008. — P. 1–9.

6. *I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields* (Eds.). Workflows for e-Science. Scientific Workflows for Grids. — Springer. — 2007. — 530 p.

7. Triana — Open Source Problem Solving Software. — http://www.trianacode.org/

8. The Kepler Project. — https://kepler-project.org/

9. Web Services Business Process Execution Language. — docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

10. OWL 2 Web Ontology Language. — http://www.w3.org/TR/owl2-overview/

11. *Петренко А. І.* Семантичний Грід для науки і освіти / А. І. Петренко, Б. В. Булах, В. С. Хондар. — К. : Політехніка, 2010. — 184 с.

12. *Андон Ф.* Роль семантики в интеграции приложений на основе веб-сервисов / В. Дерецкий // Проблеми програмування. — № 2–3. — 2010. — С. 469–479.

13. *Рогушина Ю. В.* Онтологическая модель интеллектуализации сервис-ориентированных вычислений в распределенной среде Интернет / Ю. В. Рогушина, А. Я. Гладун // Проблеми програмування. — № 2–3. — 2006. — С. 526–536.

14. *Дерецкий В.* Подходы и задачи композиции сервисов в семантическом окружении / В. Дерецкий // Проблеми програмування. — № 4. — 2008. — С. 45–59.

15. *Кашалкин Д. Ю., Курчидис В. А.* Семантическая композиция сервисов / Д. Ю.Кашалкин, В. А.Курчидис // Вестник РГРТУ. — Вып. 23. — 2008. — С.84–90.

16. *D. Kourtesis, I. Paraskakis, A. Friesen* et al. Web service discovery in a semantically extended UDDI registry: the case of FUSION // Proc. of the 8th IFIP Working Conference on Virtual Enterprises PRO-VE'07, Guimaraes, Portugal, 10–12 September 2007. — 2007. — P. 547–554.

17. *N. Srinivasan, M. Paolucci, K. Sycara.* An efficient algorithm for OWL-S based semantic search in UDDI // Lecture notes in computer science.- Springer. — Vol.3387. — 2005. — P.96–110.

18. *J. C. Goodwin, D. J. Russomanno, J. Qualls.* Survey of Semantic Extensions to UDDI: Implications for Sensor Services // Proceedings of the International Conference on Semantic Web and Web Services SWWS 2007, Las Vegas, Nevada. — 2007. — P. 16–22.

19. Apache ODE (Orchestration Director Engine). — http://ode.apache.org/

20. Jena — A Semantic Web Framework for Java. — http://jena.sourceforge.net/