УДК 004.415

ОСНОВНІ ПІДХОДИТИ ДО КОМПОЗИЦІЇ ВЕБ-СЕРВІСІВ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

О.В. Новицький

Інститут програмних систем НАН України,
03187, Київ, проспект Академіка Глушкова, 40
alex@zu.edu.ua

Розглянуто деякі підходи та проблематику побудови сервіс-орієнтованої електронної бібліотеки де доступ здійснюватиметься на динамічній основі. Для цього пропонується використовувати композицію веб-сервсів. Зокрема Консорціум W3C для композиції веб-сервсів рекомендує використовувати мову OWL-S, призначену для опису онтології обслуговування, яка покривається веб-сервісами. Подано ідею планування на основі ціле-орієнтованої парадигми для вирішення задач користувача.

This paper considers some approaches and problems relating to setting up a service-oriented digital library where access will be granted on a dynamic basis. For this purpose it is proposed to use a composition of web-services. In particular, for the composition of web-services Consortium W3c recommends to use the OWL-S language, which is intended for description of ontology of services covered by web services. The authors also put forward the concept of planning on the basis of target-oriented paradigms aimed at solving users' problems.

Вступ

Мета науково-технічної розробки "Електронна бібліотека вищого закладу освіти, інтегрована в Європейські освітньо-наукові бібліотечні системи": створити типову електронну бібліотеку, яка б ураховувала особливості процесу навчання у ВНЗ. Йдеться про створення інтеграційного середовища, що дозволило б обмінюватися навчальними ресурсами та посібниками між університетами. Весь проект буде реалізований із використанням Web-технологій та мережі Internet.

Ключем інтероперабельності даної роботи є використання сервіс-орієнтованої архітектури. Складне програмне забезпечення може бути спроектоване на основі наявних сервісів, доступних локально й через Internet. У такому контексті для побудови електронних бібліотек пропонується використовувати Сервіс Орієнтовану Архітектуру Електронної Бібліотеки (Service-Oriented Digital Library architecture - SODL) [1].

Під сервісом ми розуміємо деяку послугу, що задовольняє потреби користувача, таким чином досягаючи деякої цілі. Сервіси реалізуються за допомогою веб-сервісу. Веб-сервіс це конкретний програмний додаток. Основна відмінність між веб-сервісом і сервісом полягає в тому, що деякий абстрагований сервіс може задовольнятися різними веб-сервісами. В свою чергу веб-сервіси можуть між собою об'єднуватися створюючи новий веб-сервіс, цей процес називають композиція веб-сервісів.

Досягнення цілей здійснюватиметься динамічними сервісами тобто ефекти сервісу будуть визначатися середовищем в який даний сервіс поміщено та вимогами користувача.

Для вирішення складних задач, які постають при побудові сервіс-орієнтованої електронної бібліотеки використовують композицію веб-сервсів.

Основні проблеми, що потребують вирішення при побудові електронної бібліотеки ВНЗ:

- підвищення якості навчального процесу шляхом забезпечення вільного доступу студентів до сервісів та ресурсів свого ВНЗ та інших університетів та наукових установ;
- побудова сервіс-орієнтованої електронної бібліотеки на основі технології Semantic Web.

Існує багато способів композиції веб-сервісів [2]. Один з них базується на технології Sematic Web. В рамках цього підходу консорціум W3C рекомендує використовувати мову OWL-S, призначену для опису онтології обслуговування, яка покривається веб-сервісами. OWL-S включає три під онтології: онтологія профілю, моделі процесу та основи. Опис сервісів відрізняється від звичайного опису онтологій. Це пов'язане з тим, що при описі сервісів ми маємо справу з динамічним середовищем. Існує три різновиди OWL-S: OWL-Lite, OWL-DL та OWL-Full. Ми детально розглядатимемо OWL-DL.

Ініціатива OWL-S розвиває OWL для семантичного опису веб-сервісів, містить у собі різні аспекти веб-сервісів у тому числі й функціональні. Щоб описати функціональні можливості сервіси розглядаються як процеси, які мають початкові умови й дії.

Дескриптивна логіка. Для дослідження взаємодії веб-сервісів необхідно виконати формалізацію цього процесу. В основу мови OWL-S покладений математичний апарат дескриптивної логіки**.** Дескриптивна логіка (ДЛ) призначена для того щоб описувати логіку дій і поводження сервісів для досягнення мети залежно від вхідних і вихідних умов.

У роботі ми будемо розглядати сервіси як дії, які мають початкові й кінцеві умови. Ці умови виражаються за допомогою дескриптивної логіки, стан реального світу описують (частково) умовами, які називаються Abox. В ДЛ база знань включає два компоненти Tbox і Abox. Tbox вводить термінологію, тобто словник прикладної області, а Abox містить твердження в термінології даного словника [3]. На додаток до автономних сервісів ми будемо також розглядати просту композицію сервісів, які утворюються з послідовності атомних сервісів [4].

Зупинимося на проблемі можливої композиції сервісів а саме здійснимості, яке перевіряє чи можливо, за поточних і можливо неповних знань про світ, здійснювати обслуговування [5]. Розглянемо композицію на основі ціле-орієнтованої парадигми, тобто виходячи з початкових умов та наявної множини сервісів здійснити композицію. Причому, оскільки веб-сервіси розміщені в семантичному середовищі, то вибирати такі плани композиції, які можуть бути корисними для кінцевого користувача. Тобто на відміну від класичної постановки задачі, від специфікації цілі до пошуку сервісів, які зможуть дану ціль досягти, виходити з того припущення, що наявна множина сервісів може досягнути деякі наперед невідомі цілі, які можуть бути обрані користувачем. Для дескриптивної логіки, яка тут розглянута, і включає всі діалекти мови $ALCQIO$, складність здійснення сервісів виражених у цих діалектах збігається зі складністю судження у звичайній дескриптивній логіці AL, яка розширена так званим номіналом (тобто поняттям одиничного предмета) [5].

Основними поняттями дескриптивної логіки є атомні поняття (atomic concepts) і атомні ролі (atomic roles). Комплексні описи можуть бути індуктивно побудовані з них за допомогою конструкторові понять (concept constructors). Надалі ми будемо використовувати букви A, B для позначення атомних понять і R для позначення атомних ролей і букви C, D для опису понять. Мови опису відрізнять конструктори, які вони представляють. Мова AL є мінімальною мовою, яка становить практичний інтерес, інші мови це розширення AL. Ця мова описує поняття завдяки наступному синтаксису таб.1 [3], [6].

Таблиця 1. Основні оператори дескриптивної логіки

		Позначення	Опис
		A	Атомне поняття
		$\top$	Універсальне поняття
C, D	$\rightarrow$	$\bot$	Основне поняття
		$\neg A$	Автономне заперечення
		$C \sqcap D$	Перетин
		$\forall R.C$	Обмежене значення
		$\exists R.\top$	Обмежена езкзистанціальна квантифікація

Щоб визначити формальну семантику AL, понять розглянемо інтерпретації I які складаються з непустого набору Δ^I (домена інтерпретації) і інтерпретуючої функції, яка визначає для кожного атомного поняття A набір $A^I \subseteq \Delta^I$ і для кожної атомної ролі R бінарне відношення $R^I \subseteq \Delta^I \times \Delta^I$.

Ми говоримо що два поняття C, D еквівалентні й записуємо $C \equiv D$, якщо $C^I \equiv D^I$ для всіх інтерпретацій I [3].

Слід сказати, що в повній мірі OWL-DL описується дескриптивною логікою діалекту *SHOIN(D)*.Для наших результатів будемо використовувати (без обмеження застосування) дескриптивну логіку $ALCQIO$ і всі інші її діалекти. Це спровоковано тим, що DL $ALCQIO$ формує ядро OWL-DL. Додаткові транзитивні конструкції можуть бути легко додані за винятком транзитивних ролей.

Означення це тотожність у формі (1).

$$A \equiv C, \tag{1}$$

де A ліва частина це атомне поняття (роль). Обмежену множину називають термінологією або Tbox T якщо визначення однозначні, тобто жодне атомне поняття не проходить через ліву частину означення. Ім'я понять які зустрічаються з ліва означення в T називаються означеними в T усі інші називаються примітивами в T. Tbox T називають ациклічним, якщо немає ніяких циклічних залежностей між означеннями, тобто рекурсія означення понять завжди закінчується. Цей процес називається розширенням в Tbox. Семантика означень визначена в такий спосіб: інтерпретація є моделлю Tbox T якщо виконуються наступні умови (2):

$$A^I = C^I \text{ виконується для всіх } A = C \text{ в } T. \tag{2}$$

Будь-яка інтерпретація примітивних понять та ролей може бути унікально розширена на нециклічний Tbox. [1].

Використовуючи поняття C й ролі s можна утворити твердження наступного вигляду:

$$C(a) \text{ і } s(b,c). \tag{3}$$

Перший вид називається твердження поняття, тобто твердження приналежності a до C (інтерпретація), наступний вид називається твердженням ролі, тобто твердження, що c перебуває у певній ролі s до b.

Інтерпретація є моделлю в Abox A якщо задовольняються всі твердження в Abox A (4):

$$a^I \in C^I \quad ((a^I, b^I) \in s^I, (a^I, b^I) \notin s^I). \tag{4}$$

Якщо інтерпретація I задовольняє аксіому (твердження) α то це записують у вигляді $I \models \alpha$. Визначаємо бінарні відношення $\equiv$ та $\preccurlyeq$ на моделях I, I' у A для T.

Запис $I \equiv I'$ значить, що дві моделі рівні тобто (5):

$$a^I = a^{I'},\ \tilde{N}^I = \tilde{N}^{I'} \text{ і } R^I = R^{I'}. \tag{5}$$

Запис $I \% I'$ свідчить, що одна модель є нижчої для іншої тобто:

$$a^I = a^{I'},\ \tilde{N}^I \subseteq \tilde{N}^{I'} \text{ і } R^I \subseteq R^{I'}.$$

Формальний опис сервісів. Уведемо формальний опис веб-сервісів. Для спрощення будемо розглядати так звані основні сервіси (ground services), тобто сервіси в яких вхідні параметри індивідуальні імена. Параметричні сервіси в, яких вхідні параметри є змінні, можна розглядати як компонування всіх індивідуальних імен. Для задач планування більше логічним буде розгляд параметричних сервісів.

Означення. Нехай T нециклічний Tbox. Атомним сервісом для нециклічного Tbox T будемо називати вираз $S = (pre, sname, inputs, outputs, post, pref)$, де pre – обмежене множина тверджень в Abox, що утворює попередні статичні (і динамічні) умови; $sname$ – ідентифікаційне ім'я автономного сервісу; $inputs$ - область визначення класу вхідних параметрів сервісу, що формується у вигляді $A(a)$ або $r(a,b)$: де A - примітивне поняття в T; r - ім'я ролі a,b індивідуали; $outputs$ – область визначення класу вихідних параметрів сервісу, що формується в такому ж вигляді, як і $inputs$; $post$ – обмежене множина пост умов у формі P/E: де P - твердження в Abox; а E – примітивне поняття з посиланням на Tbox T тобто $A(a)$, $\neg A(a)$, $r(a,b)$ або $\neg r(a,b)$ [7].

Семантично запис P/E означає, що якщо P істинне перед виконанням сервісу то й E має бути істинне після виконання сервісу, pre - обмежене множина тверджень заснованих на Abox і Tbox і відображає особисті вимоги користувача, вони подаються у формі $A(a)$ або $r(a,b)$, де A - примітивне поняття з посиланням на T, r - роль a,b індивідуали.

Формальна семантика сервісів може бути визначена на основі переходу до інтерпретацій. Сервіс S може трансформувати інтерпретацію (6) I до I' [8] ($I \Rightarrow_S^{T\ A} I'$), якщо C примітивне поняття й R ім'я ролі причому.

$$C^{I'} := \begin{pmatrix} C^I \subseteq \{c^I \mid \varphi / C(c) \in post, I \models \varphi\} \setminus \\ \{c^I \mid \varphi / \forall C(c) \in post, I \models \varphi\} \end{pmatrix};$$

$$R^{I'} := \begin{pmatrix} R^I \subseteq \{(a^I, b^I) \mid \varphi / R(a,b) \in post, I \models \varphi\} \setminus \\ \{(a^I, b^I) \mid \varphi / \forall R(a,b) \in post, I \models \varphi\} \end{pmatrix}. \tag{6}$$

Тоді здійснення й проектування сервісів можливо представити таким чином [8]:

здійснення: сервіс S виконується в A для T, якщо $I \models P$ у всіх моделях I в A у відношенні до T і $I \Rightarrow_S^{T\ A} I'$;

проектування: твердження a є наслідком застосування S в A у відношенні до T якщо для кожної моделі I в A та T і для всіх I' з $I \Rightarrow_S^{T\ A} I'$ маємо $I' \models a$.

1. Семантичний опис сервісів

OWL-S не містить у собі абстрактні конструкції для того щоб явно описати повідомлення. Скоріше абстрактний зміст визначений неявним способом, через властивість входів і виходів автономного процесу.

Загалом кажучи, ServiceProfile необхідний для виявлення сервісу, а ServiceModel і ServiceGrounding разом подають інформацію про те, як використовувати знайдений сервіс рис.1.

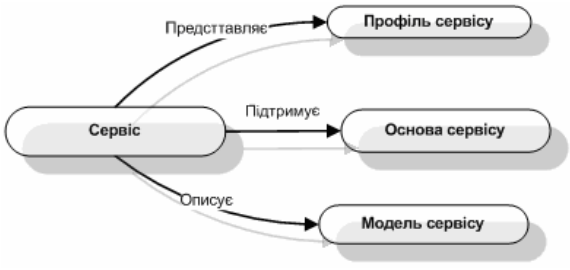

Рис. 1. Структура сервісу

Основа сервісу визначає деталі про те як одержати доступ до сервісу. Це головним чином протокол і формат повідомлення, порядок надходження, транспортування й адресація. У відмінності від ServiceProfile і ServiceModel які є абстрактними представленнями тільки ServiceGrounding звертається до конкретних рівнів специфікації [9].

Планування сервісів. Більшість досліджень, присвячених проблемі композиції веб-сервісів, ведуться в галузі штучного інтелекту. Головна проблема планування може бути описана, як кортеж чотирьох $\langle St, St_0, G, Act, Con \rangle$, де St – множина всіх можливих станів світу; $St \subset St_0$ – множина початкових станів світу; $G \subset St$ – множина цілей світу, яких система планування намагається досягти; Act – множина дій, яку може використовувати планувальник для зміни одного стану світу на інший і відношення $Con \subseteq St \times Act \times St$ відображає попередні умови та ефекти для виконання кожної дії.

В термінах веб-сервісів St_0 та G – початкові стани та цілі, які специфіковані для вибору веб-сервісів; Act – множина доступних сервісів; Con - функція зміни стану кожного сервісу [2].

Мова OWL-S – єдина мова опису веб-сервісів, яка має безпосередній зв'язок з плануванням у штучному інтелекті. Зміна стану внаслідок виконання веб-сервісу, визначається через передумови та властивості ефектів у профілі сервісу OWL-S. Передумови визначають логічні умови, виконання яких є обов'язковим для запиту сервісу. Ефекти – це результат успішного виконання сервісу. Оскільки OWL-S базується на дескриптивній логіці. Тому мова наслідує потужні засоби, які надає ця основа.

Основні методи планування веб-сервісів у рамках штучного інтелекту: числення ситуацій; мова Planning Domain Definition Language (PDDL); планування на основі правил; автоматичне доведення теорем тощо.

Постає проблема вибору правильного методу планування з урахуванням особливостей поставленого завдання та предметної галузі.

Різні методи забезпечують різний рівень автоматизації в оформленні закономірності сервісу, тому складно виділити найбільш сприйнятливий спосіб. Поряд із цим веб-сервіси є складними конструктивними елементами, і завдання автоматичної композиції може бути розв'язане лише для побудови основних етапів реалізації складного розподіленого програмного забезпечення. Проте при плануванні композиції веб-сервісів в рамках підходу Semantic Web, необхідно будувати семантичне середовище в яке веб-сервіси будуть поміщуватися.

2. Побудова семантичного середовища для веб-сервісів

Покажемо приклад опис сервісів мовою OWL-S. Як вищезазначено сервіс складається із профіль сервісу (ServiceProfile), модель процесу (ServiceModel) і основи (ServiceGrounding).

Модель взаємодії сервісів в семантичному середовищі спроектована в програмі Protege з використанням плагина OWL-S Editor.

Щоб більш детально описати як взаємодіяти із сервісом його представляють як процес. Відповідно до специфікації OWL-S процес є підкласом ServiceModel рис. 2.

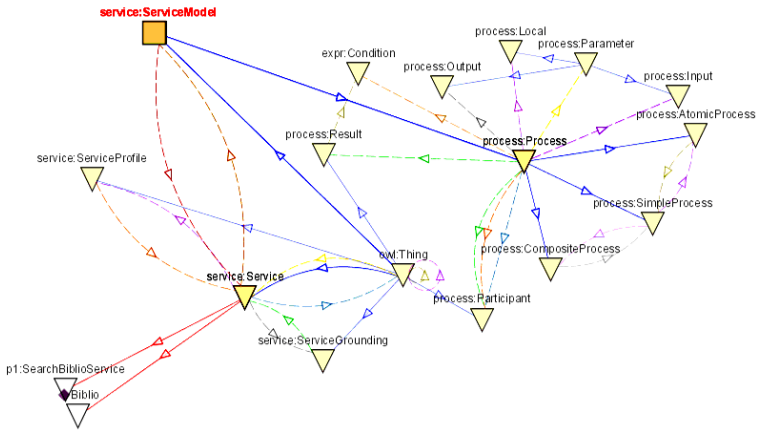

Рис. 2. Представлена структура ServiceMode

Не обмежуючи загальності для експерименту було побудовано 3 атомних процеси.

Наприклад ми маємо ситуацію, коли користувач котрий має логін і пароль доступу до електронної бібліотеки, здійснює пошук інформації (використовуючи розширений пошук) та зберігає результати пошуку для подальшого використання. Для цього потрібні наступні сервіси:

S_1 – сервіс управління доступом (7), S_2 – сервіс пошуку (8), S_3 – сервіс збереження результатів пошуку (9).

$$S_1:\quad sname_1 = \text{User\_AtomicProcess},\quad input_1 = \{Person(a)\},\quad outputs_1 = \begin{Bmatrix} Person(a), \\ \forall isRegister.DigitalLibraty(a) \end{Bmatrix} \tag{7}$$

$$S_2:\quad pre_2 = \{notFindBaseFiled(a,b)\},\quad sname_2 = FullSearch,\quad input_2 = KeyWord(c),\quad output_2 = \begin{Bmatrix} KeyWord(c), \\ \exists isObtained.DigitalLibrary(c) \end{Bmatrix},\quad pref_2 = \{sortOut(d)\} \tag{8}$$

$$S_3:\quad pre_4 = \{Person(a)\},\quad sname_4 = SaveSearch,\quad input_4 = \{Person(a), Object(o)\},\quad output_4 = \{preservation(a,o)\},\quad pref_4 = \{select(o)\} \tag{9}$$

Наведемо приклад автономного сервісу User_AtomicProcess який призначений для санкціонування користувача й у випадки проходження користувачем реєстрації привласнювати йому унікальний номер, який пов'язаний із сесією браузера. Семантика даного сервісу буде мати таку структуру рис. 3.

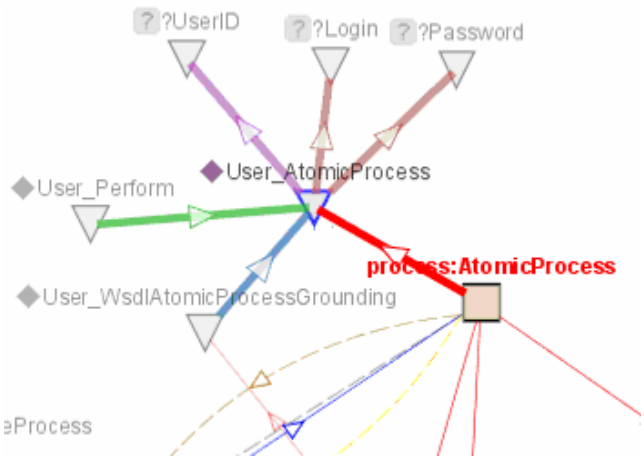

Рис. 3. Ієрархія ServiceModel

Даний сервіс вимагає параметри входу Password і Login, які, у свою чергу, належать класу Person_Password і Person_Login. Вихідним параметром після виконання процесу є UserID, який належить класу Register_Person, і визначає як клас Person які зареєстровані в Digital Library. Відповідне відношення на основі формул (1)-(3) показане на рис. 4 [10].

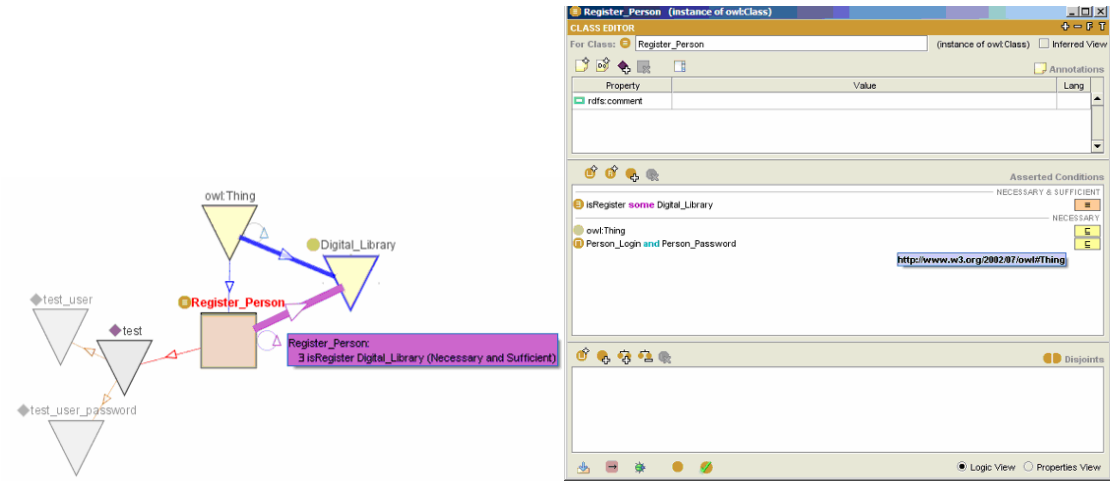

Рис. 4. Визначення класу Register_Person

Зупинимося на деяких етапах реалізації, а саме це процес побудова композиційного сервісу. Оскільки сервіс S_2 перебуває в незалежному відношенні до сервісів S_1, S_3 тобто:

$$S_2 + (S_1 + S_3) = (S_1 + S_3) + S_2 .$$

То його порядок нам байдужий. У той же час сервіс S_3 вимагає попереднього виконання сервісів S_1 і S_2 тобто повинні виконуватися умови:

$$S_1 \to S_2 \to S_3 \text{ або } S_2 \to S_1 \to S_3 .$$

Виходячи з логічності побудова архітектури обслуговування нами обраний випадок коли $S_1 \to S_2 \to S_3$.

Послідовність процесів визначається потоком даних між усіма процесами які беруть участь у композиції. На рис. 5 продемонстрований потік даних між процесами S_1 S_2 S_3.

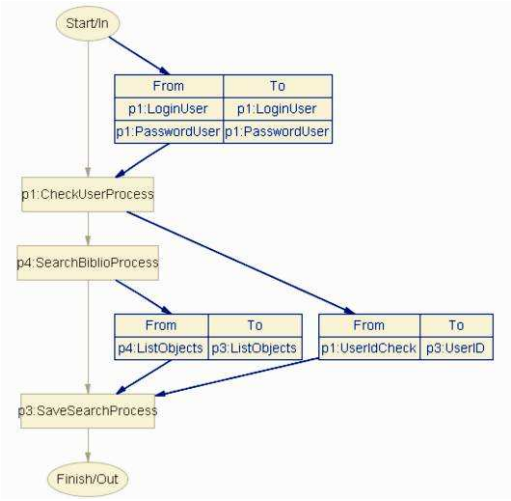

Рис. 5. Потік даних між процессами

Ще один важливий компонент це основа сервісу – ServiceGrounding. Центральна функція основи OWL-S показати яким способом абстрактні входи й висновки атомного (або складного) процесу, мають бути реалізовані на повідомленнях, які несуть ті входи й виходи в деякому певному форматі. Тому що єдиним добре розробленим форматом для повідомлень є WSDL[11], то OWL-S підтримує основу на базі цієї специфікації [9]. Ми створили основу з використанням WSDL інтерфейсів.

Висновок

Ми привели початкові відомості про дескриптивну логіку, а також про сферу її використання. У даній роботі показано приклад використання дескриптивної логіки в побудові й розумінні концепції Semantic Web для електронної бібліотеки. Цей підхід ефективний, так як дозволяє створювати семантику програмного забезпечення не прив'язуючись до конкретної мови прикладного програмування. Водночас дозволяє ефективно повторно використовувати атомні сервіси не тільки рядовим програмістам, але й спеціалізованим програмним агентам. Опис атомних сервісів засобами дескриптивної логіки й наступна реалізація цього опису на OWL-S

надає машиночитаємий інтерфейс для автоматичної композиції сервісів. Що дозволяє вирішувати задачі які постають перед агентом в автоматичному або напівавтоматичному режимі.

Виділено атомні сервіси які можна взяти за основу для побудова наукової електронної бібліотеки.

Однак при композиції сервісів постає ряд проблем які потребують подальшого вирішення, зокрема співставлення сервісів не тільки на семантичному рівні, а також і на рівні повідомлень та якісного вибору планів які досягають цілі в деяких певних ситуаціях.

1 *Yves Petinot, C. Lee Giles, Vivek Bhatnagar, Pradeep B.* Teregowda, Hui Han, Isaac Councill. A Service-Oriented Architecture for Digital Libraries.

2 *J. Rao, X. Su.* A Survey of Automated Web Service Composition Methods. In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, San Diego, California, USA, July 6th, 2004.

3 *The* description logic handbook: theory, implementation, and applications Edited by Franz Baader Deborah L. McGuinness Daniele Nardi Peter F. Patel-Schneider.

4 *Lirong* Qiu, Fen Lin, ChanglinWan, ZhongzhiShi. Semantic WebServices Composition Using AIplanning of Description Logics.

5 *Description* Logics: Foundations for Class-based Knowledge Representation Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini Dipartimento di Informatica e Sistemistica Universit'a di Roma “La Sapienza” Via Salaria 113, I-00198 Roma, Italy.

6 *Description* Logics: Foundations for Class-based Knowledge Representation Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini Dipartimento di Informatica e Sistemistica Universit'a di Roma “La Sapienza” Via Salaria 113, I-00198 Roma, Italy.

7 *Yingjie Li1, Xueli Yu1, Lili Geng1, Li Wang.* Research on Reasoning of the Dynamic Semantic Web Services Composition. Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on Volume , Issue , 18-22 Dec. 2006 Page(s):435 – 441.

8 *Franz Baader, Maja Milicic, Carsten Lutz, Ulrike Sattler, Frank Wolter.* Integrating Description Logics and Action Formalisms for Reasoning about Web Services. LTCS-Report 05-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005.

9 *OWL*-S: Semantic Markup for Web Services http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

10 *Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M., Colucci, S.* Fully automated Web services orchestration in a resource retrieval scenario. Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on Publication Date: 11-15 July 2005.

11 *WebServices* Description Language (WSDL)1.1 http://www.w3.org/TR/2001/NOTE-wsdl-20010315.