Семантическое расширение сервисных описаний

Л.Б. Шереметов, К. Санчес

Аннотация. Статья посвящена решению задачи обнаружения и композиции Вэб сервисов (ВС) с неполной информацией. В ней описаны два алгоритма, которые позволяют генерировать расширения описания сервиса клиента (определенного на языке OWL-S), расширяя тем самым границы поиска. Характеристики алгоритмов изучены на основании анализа доступных баз данных описаний ВС и онтологий с целью их сравнения с известными алгоритмами.

Ключевые слова: Вэб сервисы, семантическое описание, обнаружение сервисов, динамическая композиция сервисов.

Введение

В последнее время рынок информационных технологий стал свидетелем повышенного интереса к Вэб сервисам (ВС), развитие которых связано с поиском эффективной и гибкой интеграции бизнес-процессов и информационных систем предприятий. В этой связи особое внимание уделяется масштабной концепции динамической композиции сервисов (ДКС), в которой в любой момент, в любой ситуации возможно предоставить наилучший сервис, удовлетворяющий запросу пользователя [1]. ДКС позволяет объединять по запросу в реальном времени "полностью совместимые" взаимодействующие компоненты программного обеспечения либо удаленно выполнять сложные вычислительные задачи для портативных малопроизводительных устройств, тем самым позволяя строить распределенные слабосвязанные гибкие и гетерогенные системы программного обеспечения, понижая, таким образом, цену разработки приложений и, в тоже время, увеличивая их возможности [2].

Вместе с тем, техническая реализация этой концепции, которая должна включать обнаружение сервисов, их динамическую адаптацию и планирование, находится в начальном состоянии и пока состоит из ряда иллюстративных примеров. Обнаружение ВС, направленное на поиск сервисов, соответствующих запросу пользователя, все еще является открытой задачей, так как большинство текущих работ, ориентированных как на синтаксическое (как, например, в UDDI), так и на семантическое описании сервисов, фокусируются на поиске сервиса с точным соответствием запросу, что не всегда возможно. Существует несколько предложений объединения синтаксических (регистрируемых в XML-реестре Универсального Описания, Обнаружения и Интеграции - UDDI[1]) и семантических свойств сервисов [3]. С одной стороны, они основаны на улучшении существующих синтаксических техник обнаружения, а с другой – на семантическом расширении сервисных запросов. Такое расширение, в свою очередь, может быть достигнуто различными методами: генерированием сервисных описаний на естественном языке для специальных предметных областей с целью выбора наиболее подходящего сервиса в случае неопределенностей [4], генерированием запросов с использованием комбинаций синонимов параметров [5] и т.д. Однако большинство этих

[1] Сокращение от англоязычного *Universal Description, Discovery and Integration*

подходов учитывают только прямое соответствие между входами-выходами-предусловиями-эффектами[2] либо требуют развития семантических UDDI [3]. В первом случае значительно уменьшается множество потенциально пригодных сервисов, а в последнем приводит к семантическим расширениям запросов порядка нескольких миллионов.

Таким образом, для успешного перехода к сервисным системам требуемого качества необходима адаптация существующих методов при одновременной разработке новых подходов к семантической интеграции сервисов. Для того чтобы подобная интеграция стала возможной, необходимо гарантировать семантическое взаимодействие сервисных интерфейсов, в основе которого лежит формальная модель, позволяющая описать сервисы, идентифицировать сходства между ними и, наконец, скомпоновать сложные сервисы. В [6] авторами была предложена формальная модель соответствия сервисов с неполной информацией. Эта модель решает задачи проверок подобия между сервисными описаниями для сервисной классификации и выбора наиболее похожего сервиса, который удовлетворяет запросу. В этой статье предложен подход к обнаружению похожих сервисов с неполной информацией, основанный на расширениях OWL-S описаний сервиса клиента на основе использования онтологий предметной области. Подобная техника расширения запроса широко применяется в механизмах Вэб поиска, где эффективно используются онтологии [7]. Разработанный метод проиллюстрирован на примере из нефтяной области, в которой онтологии в последнее время находят все более широкое применение [8]. Так, например, лексико-семантические отношения использовались для разработки таблиц соответствия словарных элементов при интеграции производственной и финансовой подсистем ОАО «Татнефть» [9]. Цели этой статьи: развить и проанализировать алгоритмы для расширения сервисных описаний и показать, как они могут быть использованы для ДКС.

1. Синтаксическое и семантическое сходство сервисов

UDDI является сервисом, позволяющим регистрировать, удалять и искать сервисы, описанные на языках WSDL или OWL-S, основанных на XML. Другими словами, сервисный домен образован клиентами C и провайдерами Φ. В интерфейсах сервисов обычно специфицируются функциональные аспекты — вводные и выводные параметры, предусловия и эффекты выполнения (IOPE). Описание сервисного профиля (информация, которую требует сервис и которую он может предоставить) на OWL-S представляется 4-кортежем $s_i=(I_i, O_i, P_i, E_i)$, где I_i и O_i означают соответственно требуемые входы (I) и предоставляемые выходы (O), P_i – предусловия, которые должны сохранятся для функционирования сервиса, E_i – эффекты выполнения сервиса.

Допустим, что $S=\{s_1,...,s_{|S|}\}$ есть множество простых сервисов, предлагаемых Φ, где каждый сервис s_i принадлежит также некой категории ST_i. Сложный сервис cs состоит из поднабора сервисов, $cs=S''$. Эти сервисы, сконфигурированные определенным образом (параллельным, последовательным или смешанным), предлагают набор выходов O'' для данного набора входов I'', если и только если выполняются все предварительные условия P'' сервисов, которые образуют сложный сервис, $cs=\{s_1,...,s_{|cs|}\}$. Следует отметить, что сложные сервисы относятся к неунитарному множеству ST'' классификаций (или категорий) аналогично простым сервисам $S''\subseteq S$, $O''\subseteq O$ и $I''\subseteq I$.

Запрос клиента s_C генерируется с целью нахождения сервиса s_Φ, зарегистрированного провайдером в UDDI, или генерации сложного сервиса sc_{Φ_i}, в случае, если простой сервис не найден. Говорят, что UDDI находит сервис s_Φ, удовлетворяющий условиям равенства $s_\Phi=s_C$, если входы, выходы и категории сервисов в точности одинаковы, $I_C=I_\Phi$, $O_C=O_\Phi$ и $ST_C=ST_\Phi$. Задача выбора сервиса из множества найденных формулируется на основе некоего критерия выбора z. Определим

[2] Сокращение от *Input, Output, Preconditions, Effects*.

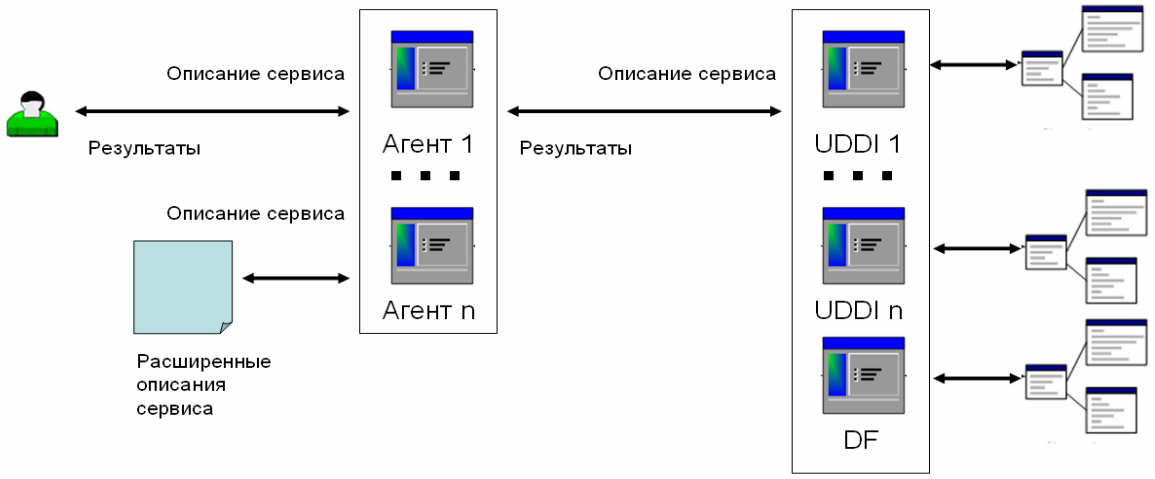


Рис. 1. Архитектура реализации подхода к обнаружению сервисов, основанного на расширении описаний

этот критерий как картезианское произведение условий $z = \prod_{l \in (1,\ldots n)} Q_l$. Кроме того, необходимо задать натуральное число f, $f \in \mathrm{N}$, $f \neq 0$ - максимальное число опций, которые клиент желает получить. Если $f = 1$, на UDDI должен быть выбран единственный сервис, наилучшим образом удовлетворяющий условиям z.

Тем не менее, найти сервис на UDDI не всегда возможно по многим причинам: например, такого сервиса не существует, сервис недоступен в настоящее время либо есть неопределенность в описании сервиса. Как было упомянуто выше, чтобы решить эту задачу, должны быть сгенерированы семантически сходные описания с целью нахождения сервиса, наиболее сходного с запрашиваемым. Эти описания могут быть сгенерированы самим UDDI (другими словами, семантическим UDDI). Недостатки такого подхода состоят в том, что не обеспечивается совместимость с традиционными и широко используемыми в приложениях синтаксическими UDDI, а количество сгенерированных семантически сходных описаний может быть огромным (Раздел 5). В предложенном подходе используется традиционный UDDI (Рис. 1). Для решения семантических задач введен агент (или агенты) брокер, который при поступлении запроса генерирует расширенные сервисные описания и затем посылает их в UDDI. На обратном пути этот агент получает сервисные описания от UDDI для фильтрации и классификации на основе степеней семантического сходства. Вместе с тем, как показано ниже, для случая пошагового алгоритма (Раздел 3.2), эта задача может быть даже опущена, так как предварительное фильтрование происходит во время генерации описаний.

2. Типы сходства сервисов

Существуют три формы сравнения сервисов: равенство, неравенство и сходство, которое, в свою очередь, можно представить тремя способами (операционное, входов и/или выходов). Их можно задать посредством двухзначных соотношений (Табл. 1).

Табл. 1. Бинарные отношения сравнения сервисов

Название отношения	Символ	Характеристики
Равенство	$=$	Рефлексивное, антисимметричное, передаваемое
Операционное сходство	$\sim$	Рефлексивное, антисимметричное, передаваемое
Сходство входов	$\approx$	Рефлексивное, антисимметричное, передаваемое
Сходство выходов	$\cong$	Рефлексивное, антисимметричное, передаваемое
Неравенство	$\neq$	Рефлексивное, антисимметричное, передаваемое

От форм сравнения сервисов перейдем к определению типов сходства между ними. В этой статье предложены три типа сходства сервисов. Наряду с традиционным полным сходством, определены типы сходства с избыточной и недостающей информацией, что делая возможным нахождение тех сервисов, которые соответствуют (возможно, в некоторой степени) желаемым параметрам запроса. При дальнейшем описании подхода без потери общности ограничимся рассмотрением только входов и выходов. Сравнивая запрошенные и предложенные сервисы, определим следующие типы множеств: $\ddot{I} = I_C - I_\Phi$, $\ddot{O} = O_\Phi - O_C$, $\underset{..}{I} = I_\Phi - I_C$ и $\underset{..}{O} = O_C - O_\Phi$, представляющие избыточные и недостающие входы и выходы соответственно. Возможные типы сходства представлены в Табл. 2 и проиллюстрированы на Рис. 2.

Табл.2. Типы сходства сервисов

Тип сходства	Описание
а) Полное сходство сервисов	$I_C = I_\Phi$ & $O_C = O_\Phi$ & $ST_C = ST_\Phi$ & $\ddot{I} = \ddot{O} = \underset{..}{I} = \underset{..}{O} = \varnothing$
б) Полное сходство входов и выходов I&O	$I_C = I_\Phi$ & $O_C = O_\Phi$ & $\ddot{I} = \ddot{O} = \underset{..}{I} = \underset{..}{O} = \varnothing$
в) Сходство с избыточными входами	$I_\Phi \subseteq I_C$ & $\ddot{I} \neq \varnothing$ & $\ddot{O} = \varnothing$
г) Сходство с избыточными выходами	$O_C \subseteq O_\Phi$ & $\ddot{I} = \varnothing$ & $\ddot{O} \neq \varnothing$
д) Сходство с избыточными I&O	$I_\Phi \subseteq I_C$ & $O_C \subseteq O_\Phi$ & $\ddot{I} \neq \varnothing$ & $\ddot{O} \neq \varnothing$
е) Сходство с недостающими выходами	$O_\Phi \subseteq O_C$ & $\underset{..}{I} = \varnothing$ & $\underset{..}{O} \neq \varnothing$
ж) Сходство с недостающими входами	$I_C \subseteq I_\Phi$ & $\underset{..}{I} \neq \varnothing$ & $\underset{..}{O} = \varnothing$
з) Сходство с недостающими I&O	$I_C \subseteq I_\Phi$ & $O_\Phi \subseteq O_C$ & $\underset{..}{I} \neq \varnothing$ & $\underset{..}{O} \neq \varnothing$

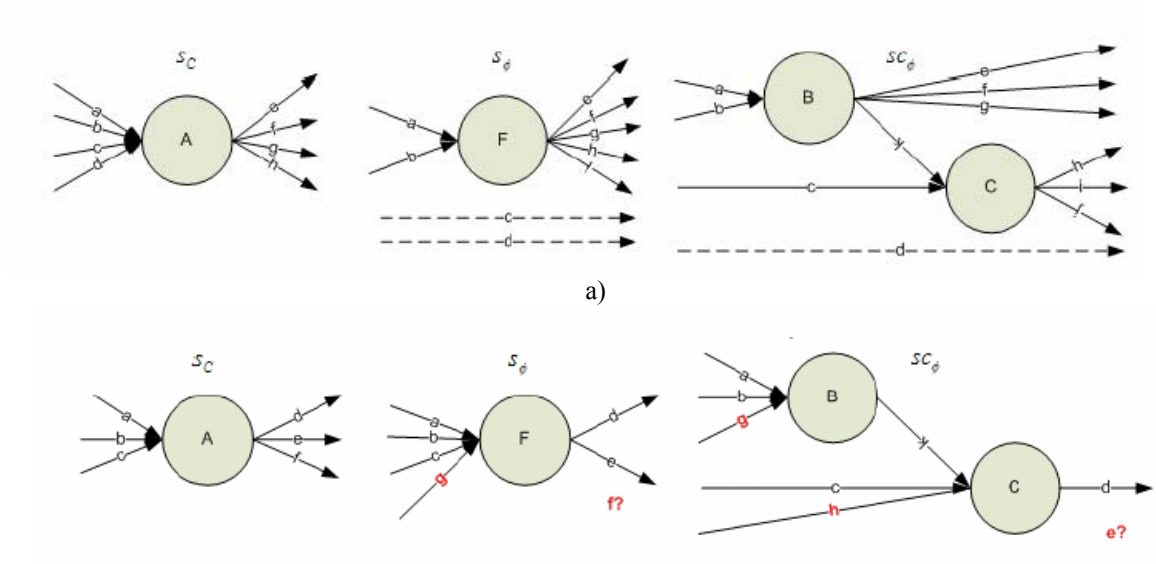


Рис. 2. Типы сходства искомого s_c и найденных простого S_Φ и составного SC_Φ сервисов

а) сходство с избыточными входами и выходами (*c, d, i, j*)
б) сходство с недостающими входами и выходами (*g, h, f, e*)

Рассмотрим пример. Предположим, что задачей является получение данных геофизических исследований некой нефтяной скважины конкретного месторождения. Пользователь запрашивает сервис *LogWebService*, требующий идентификатор ствола скважины и дату *(Wellbore, Date)* и возвращающий данные из бурового журнала по минеральному составу породы (*Drilling_Log,*

Mineral_Composition). Отметим, что предусловия и эффекты, которые в примере не учитываются, впоследствии могут образовать другой критерий отбора.

LogWebService*(Wellbore wellbore_input, Date date_input, Drilling_Log dl_output, Mineral_Composition mc_output)*

Условные найденные описания сервисов соответствуют определенным типам сходства с запросом пользователя (Табл. 1):

а,б) **LogWebService**(*Wellbore wellbore_input, Date date_input, Drilling_Log dl_output, Mineral_Composition mc_output)*
в) **LogWebService**(*Wellbore wellbore_input, Drilling_Log dl_output, Mineral_Composition mc_output*)
г) **LogWebService**(*Wellbore wellbore_input, Date date_input, Drilling_Log dl_output, Mineral_Composition mc_output, Company company_output*)
д) **LogWebService**(*Wellbore wellbore_input, Drilling_Log dl_output, Mineral_Composition mc_output, Company company _output*)
е) **LogWebService**(*Wellbore wellbore_input, Date date_input, Drilling_Log dl_output*)
ж) **LogWebService**(*Wellbore wellbore_input, Date date_input, Depth depth_input, Drilling_Log dl_output, Mineral_Composition mc_output*)
з) **LogWebService**(*Wellbore wellbore_input, Date date_input, Depth depth_input, Drilling_Log dl_output*)

Например, описание (ж) частично соответствует запросу, однако запрашивает дополнительные входные данные по глубине участка скважины (*Depth depth_input)*, с которого получены данные. Как следствие, это описание относится к типу сходства с недостающими входами. Как следует из примера, найденные описания используют и другие дополнительные параметры (не отраженные в запросе), такие как название компании, проводившей съем данных (*Company*).

Упорядочение по предпочтению для типов сопоставления определено следующим образом: *Полное сходство > Полное сходство входов и выходов (I&O)> Сходство с избыточными входами > Сходство с избыточными выходами > Сходство с избыточными I&O > Сходство с недостающими выходами > Сходство с недостающими входами > Сходство с недостающими I&O > Несовпадение.* Понятие точного совпадения между входами и выходами в следующем разделе расширено на понятия их семантического сходства.

2.1. Семантическое сходство входов и выходов

Чтобы определить сходство между сервисами, необходимо определить сходство элементов, которые описывают сервис, другими словами, входов, выходов и категорий, к которым они принадлежат. Так как описания сервисов основаны на OWL-S, семантическое сходство этих элементов определяется с использованием понятий *Эквивалентности* и *Производной,* определенных в Дескриптивной Логике (ДЛ) [10]. Во время сравнения входных и выходных множеств они должны сравниваться поэлементно для каждого множества (например, $i_{C_1} = i_{\Phi_1}$ или $o_{C_1} = o_{\Phi_1}$). Считается, что все элементы множества равны (другими словами $I_C = I_\Phi$ и $O_C = O_\Phi$), если на них может быть определено одно из следующих двух отношений, определенных в [10], *эквивалентности* $\equiv$ *и включения* $\sqsupseteq$. Каждое отношение определяет тип сходства в соответствии с ДЛ. Каждый вход или выход представляется как частный случай концепта OWL из конкретной онтологии. Сопоставление между этими концептами относится к одному из следующих типов: *соответствие эквивалентности* или *соответствие включения.*

Пусть s_C – запрашиваемый сервис с входами I_C и выходами O_C. Так как i_{C_n} - один из входов сервиса s_C, где $i_{C_n} \in I_C$ и $1 \le n \le |I_C|$, считается, что вход i_{Φ_n} соответствует запрашиваемому входу, если

$i_{C_n} \equiv i_{\Phi_n}$ или $i_{\Phi_n} \sqsupseteq i_{C_n}$. Соответствие выходов определено аналогично: $o_{C_m} \equiv o_{\Phi_m}$ и $o_{C_m} \sqsupseteq o_{\Phi_m}$, где $o_{C_m} \in O_C$ и $1 \le m \leqslant |O_C|$.

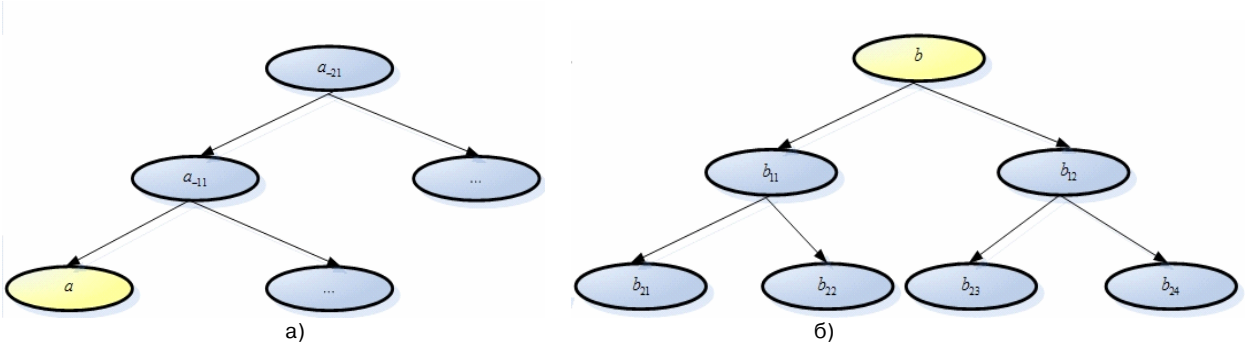
2.2. Измерение степени сходства

Сходство сервисов измеряется с помощью их параметров. Степень сходства определяется на основе вычисления дистанции между концептами онтологии, в которой параметры сервисного описания являются концептами. При этом возможность использования лексической единицы *A* вместо лексической единицы *B* определяется входимостью *A* в иерархию гипонимов *B* либо наоборот (является ли *A* частным случаем *B* или наоборот) [9]. Несмотря на то, что в последнее время были предложены различные алгоритмы измерения дистанции между двумя концептами онтологии [11], в данной работе будем использовать наиболее распространенную метрику определения расстояния как наикратчайшего пути на семантической сети, введенную еще Роем Рада (Roy Rada) и его коллегами [12].

Расстояние между входами и выходами измеряется следующим образом. Вход i_{C_m} имеет нулевое расстояние до входа i_{Φ_m}, если оба входа описаны одним и тем же концептом. С другой стороны, i_{C_m} имеет расстояние -1 до входа i_{Φ_m}, если i_{Φ_m} - предок концепта i_{C_m}. Выход o_{C_n} имеет нулевое расстояние до выхода o_{Φ_n}, если оба выхода описаны одним и тем же концептом. С другой стороны, o_{C_n} имеет расстояние, равное 1 до выхода o_{Φ_n}, если этот выход - потомок концепта o_{C_n}. Таким же образом можно вычислить расстояния для любого найденного уровня потомка или предка, применяя техники восхождения и нисхождения, соответственно для входов и выходов. Этот подход проиллюстрирован на Рис. 3 для случая, когда онтология имеет древовидную структуру. Для того чтобы рассчитать расстояние между сервисными описаниями клиента и поставщика, рассчитываются соответствующие расстояния для каждого входа и выхода, затем абсолютные значения этих расстояний суммируются.

Чтобы проиллюстрировать вышесказанное, рассмотрим концепты из онтологии по нефтяным скважинам (Табл. 3), которые представляют собой сервисные параметры и их предков (для входов) и потомков (для выходов). Учитывая, что входной параметр *Date* не имеет предков, так же как выход *Drilling_Log* не имеет потомков, приводя пример сервисного описания, семантически сходного с запрошенным клиентом:

и) **LogWebService**(*Oil_Well ow_input, Date date_input, Drilling_Log DL_output, Lithology litho_output*)



а) б)

Рис. 3. Схема определения схожести параметров заданного описания сервиса s_c с одним входом *a* и одним выходом *b*
а) параметр входа *a* и его предки (восхождение)
б) параметр выхода *b* и его потомки (нисхождение)

Табл. 3. Предки и потомки для входных и выходных параметров
параметры запрашиваемого сервиса выделены темным фоном

Вещь (Thing)			
Скважина (Hole)			
Буровая скважина (BoreHole)			
Нефтяная скважина (Oil_well)	Вещь (Thing)		
(Ствол скважины - WellBore	Дата - Date	Буровой журнал - Drilling_Log	Минеральный Состав - Mineral_Composition)
		Ничего (Nothing)	Литология (Lithology)
			Ничего (Nothing)

Сервисное описание *i* семантически схоже с первоначальным запросом, поскольку клиент, рассчитывающий на концепцию *Wellbore*, может сделать прогноз по отношению к концепции *Oil_Well*. С другой стороны, клиент ищет *Mineral_Composition* и получает *Lithology,* которая является более специфичной концепцией. Соответствие включения (случай *u*) сгенерирует (-1,0,0,1) или степень сходства, равную 2.

2.3. Определение нечетких степеней сходства

Для определения семантического сходства между сервисами рассмотрим способ, основанный на теории нечетких множеств. При этом степени принадлежности каждой функции помогают определить сходство между двумя сервисами с целью использования этой информации при выборе сервисов. В рамках данной статьи рассмотрим лишь сходство входов, что, однако, не приводит к потере общности рассуждений.

Определим функцию принадлежности $\mu_{i_{C_m}}(i_{\Phi_m})$, которая задает степень сходства входа i_{Φ_m} найденного сервиса по отношению к входу запрошенного сервиса i_{C_m}, следующим образом (Рис. 4):

$$\mu_{i_{C_m}}(i_{\Phi_m}) = \begin{cases} \dfrac{2}{(1+e^{parent\_average^{*}-i\_updistance})^{-1}} & if \quad i_{C_m} = i_{\Phi_m}, \\ 0 & otherwise \end{cases} \qquad (1)$$

где *parent_average* - усредненная максимальная глубина онтологии, значение которой было определено на основе анализа 70-ти онтологий, доступных на Swoogle [15], в среднем один концепт содержит 90 потомков в вертикальном направлении дерева;

i_updistance - расстояние между входом описания клиента i_{C_m} по отношению ко входу найденного сервиса i_{Φ_m}, полученное с использованием техники восхождения, как было описано в предыдущем разделе.

Функция принадлежности μ_{I_C}, которая определяет три степени сходства (низкая, средняя и высокая) между множествами входов I_C сервиса s_C и I_Φ сервиса s_Φ, определена следующим образом (Рис. 5):

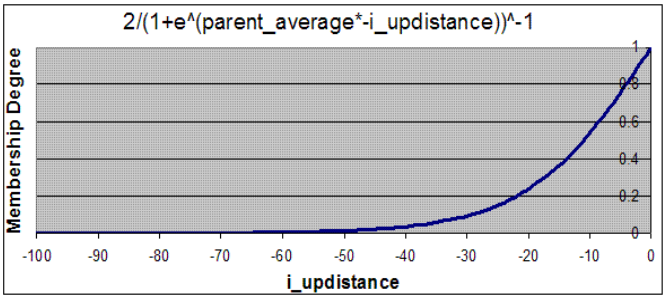


Рис. 4. Функция принадлежности, определяющая степень сходства входа i_{Φ_m} найденного сервиса по отношению к входу запрошенного сервиса i_{C_m}.

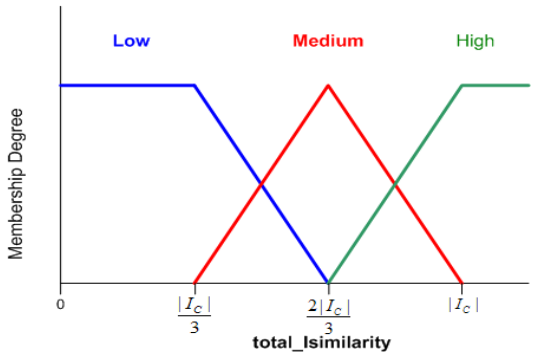


Рис. 5. Функция принадлежности, определяющая степень сходства входов I_C и I_Φ сервисов s_C и s_Φ соответственно

$$\mu_{I_C}(I_\Phi) = \{\mu_{I_{C_L}}, \mu_{I_{C_M}}, \mu_{I_{C_H}}\}, \tag{2}$$

где

$$\mu_{I_{C_L}}(I_\Phi) = \begin{cases} 1 \quad if \quad \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) < \frac{|I_C|}{3} \\ \frac{\left(\frac{|I_C|}{3}*2\right) - \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right)}{\left(\frac{|I_C|}{3}*2\right) - \left(\frac{|I_C|}{3}\right)} \quad if \quad \frac{|I_C|}{3} \le \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) \le \left(\frac{|I_C|}{3}*2\right) \\ 0 \quad if \quad \left(\frac{|I_C|}{3}*2\right) < \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) \end{cases}$$

$$\mu_{I_{C_M}}(I_\Phi) = \begin{cases} \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) - \frac{|I_C|}{3} \quad if \quad \frac{|I_C|}{3} \le \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) \le \left(\frac{|I_C|}{3}*2\right) \\ \frac{|I_C| - \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right)}{3} \quad if \quad \left(\frac{|I_C|}{3}*2\right) \le \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) < |I_C| \\ 0 \quad if \quad \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) < \frac{|I_C|}{3} \quad or \quad |I_C| < \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) \end{cases}$$

$$\mu_{I_{C_H}}(I_\Phi) = \begin{cases} \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) - \left(\frac{|I_C|}{3}*2\right) \quad if \quad \left(\frac{|I_C|}{3}*2\right) \le \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) \le |I_C| \\ 1 \quad if \quad |I_C| \le \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) \\ 0 \quad if \quad \left(\sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})\right) < \left(\frac{|I_C|}{3}*2\right) \end{cases}$$

Таким образом, $total\_Isimilarity = \sum_1^{|I_C|} \mu_{i_{C_m}}(i_{\Phi_m})$.

3. Генерация расширенных описаний сервисов

В случае, когда не удается обнаружить сервис, имеющий полное сходство описания с запрашиваемым клиентом, необходимо сгенерировать расширенные сервисные описания. Ниже описаны два алгоритма расширения описаний. Первый включает генерацию всех возможных (в пределах предметной онтологии) описаний сходных сервисов, поиск таких сервисов, а затем оценку, классификацию и выбор наиболее сходных сервисов. Этот метод называется исчерпывающей генера-

цией описаний. Второй алгоритм включает генерацию запросов в зависимости от степени сходства последовательно, начиная с наиболее сходного запроса и заканчивая наименее сходным. При этом в случае нахождения сервисов, их можно выбрать непосредственно, без дополнительной оценки и классификации. Этот метод называется пошаговой генерацией.

3.1. Исчерпывающая генерация сервисных описаний

Уравнение (3) определяет количество сходных сервисных описаний #*q,* задающих расширение сервисного описания по входам q_e и выходам q_s.

$$\#q = q_e * q_s, \text{ где } q_e = \begin{cases} \prod_{j=1}^{|I|}(a_{i_j}+1) & if \quad 1 \le |I| \\ 0 & otherwise \end{cases} \text{ и } q_s = \begin{cases} \prod_{k=i}^{|O|}(d_{o_k}+1) & if \quad 1 \le |O| \\ 0 & otherwise \end{cases}, \tag{3}$$

при этом a_{i_j} представляет собой количество предков для каждого входа i_j, где $i_j \in I$, а d_{o_k} является количеством потомков для каждого выхода o_k, $o_k \in O$. Алгоритм исчерпывающей генерации описаний приведен ниже:

1: Предположим, что существует сервис s_C с соответствующими j входами I_C и k выходами O_C
2: Если $k > 0$, тогда
3: **REPEAT** для каждого выхода *k*
4: Получить всех потомков m для o_k с использованием программного интерфейса средства разработки онтологии
5: Если $m = 0$, тогда **BREAK**
6: **REPEAT** для каждого потомка *m* выхода o_k
7: Сгенерировать копию сервисного описания s_C', замещающего выход o_k потомком *m*
8: **END REPEAT** *m*
9: **END REPEAT** *k*
10: **REPEAT** для каждого входа *j*
11: Получить предка *l* для I_j.
12: Если I_j не имеет предков, тогда **NEXT**
13: Сгенерировать копию сервисного описания s_C'", замещающего вход i_γ предком *l*
14: **END REPEAT** *j*

Как можно видеть, на первом этапе, если выход o_k имеет *m* потомков, генерируются новые *m* запросы. На втором этапе для *n* потомков второго выхода получены новые сервисные описания (*m*+1)+[(*m*+1)**n*] и т.д. С другой стороны, если каждый вход i_r имеет *l* предков, будут сгенерированы новые запросы *f*+(*f* * *l*), где *f* - количество запросов, сгенерированных для входов.

Вернемся к примеру. Предки и потомки всех параметров входов и выходов для требуемого сервисного описания приведены в Табл. 2. Так как у *Date* и *Drilling_Log* нет предков и соответственно потомков, должны быть рассмотрены дополнительные концепты: *Thing* в качестве предка входов и *Nothing* в качестве потомка для выходов. Эти концепты используются для определения описаний с недостающими входами и выходами (случай 1б ниже, где *Nothing* опущен). Так как литология (*Lithology)* является единственным потомком *Mineral_Composition,* а вход *Wellbore* имеет 3-х предков, для этого примера число расширенных описаний равно [(3+1)*(0+1)]*[(0+1)*(1+1)]= [4]*[2]= 8. Для примера могут быть сгенерированы следующие сервисные описания в соответствии с уравнением (3):

1а) **LogWebService**(*Wellbore wellbore_input, Date date_input, Drilling_Log dl_output, Lithology litho_output*)
1б) **LogWebService**(*Wellbore wellbore_input, Date date_input, Drilling_Log dl_output*)
2) **LogWebService**(*Oil_Well ow_input, Date date_input, Drilling_Log dl_output, Mineral_Composition mc_output*)
3) **LogWebService**(*Oil_Well ow_input, Date date_input, Drilling_Log dl_output, Lithology litho_output*)
4) **LogWebService**(*BoreHole bh_input, Date date_input, Drilling_Log dl_output, Mineral_Composition mc_output*)
5) **LogWebService**(*BoreHole bh_input, Date date_input, Drilling_Log dl_output, Lithology mc_output*)
6) **LogWebService**(*Hole hole_input, Date date_input, Drilling_Log dl_output, Mineral_Composition mc_output*)
7) **LogWebService**(*Hole hole_input, Date date_input, Drilling_Log dl_output, Lithology litho_output*)

С целью уменьшения комбинаторной сложности этого алгоритма мы можем использовать степени сходства, пошагово генерируя расширенные сервисные описания.

3.2. Пошаговая генерация сервисных описаний

Уравнение (4) определяет количество сходных сервисных описаний $\#q^D$, задающих расширение сервисного описания по входам $q_e^{D_e}$ и выходам q_s^{Ds}:

$$\#q^D = q_e^{D_e} * q_s^{D_s}, \text{ где } q_e^D = \begin{cases} \prod_{j=1}^{|I|}(a_{i_j}^{D_i}+1) & if \quad 1 \le |I| \\ 0 & otherwise \end{cases} \text{ и } q_s^D = \begin{cases} \prod_{k=i}^{|O|}(d_{o_k}^{D_k}+1) & if \quad 1 \le |O| \\ 0 & otherwise \end{cases}, \tag{4}$$

при этом $a_{i_j}^{D_i}$ представляет собой количество предков для каждого входа i_j на уровне D_i, $i_j \in I$, а $d_{o_k}^{D_k}$ является количеством потомков для каждого выхода o_k на уровне D_k, $o_k \in O$. D представляет собой степень сходства, $D=e+s$, где e и s являются суммарными количествами перемещений по онтологии (Рис. 3) для параметров входов и выходов соответственно (Табл. 2). Алгоритм пошаговой генерации описаний приведен ниже:

1. Предположим, что существует сервис s_c с соответствующими j входами I_C и k выходами O_C
2. **REPEAT** для каждого уровня D_i
3. Если $k > 0$, тогда
4. **REPEAT** для каждого выхода k
5. Получить потомков для o_k уровня D_i с использованием программного интерфейса средства разработки онтологии
6. Если o_k не имеет потомков, тогда **NEXT**
7. Сгенерировать копии сервисного описания s_C', замещающие выход o_k потомками
8. **END REPEAT** k
9. Если $j>0$, тогда
10. **REPEAT** для каждого входа j
11. Получить предка для I_j уровня D_i с использованием программного интерфейса средства разработки онтологии.
12. Если I_j не имеет предков, тогда **NEXT**
13. Сгенерировать копию сервисного описания s_C''', замещающего вход i_r предком l
14. **END REPEAT** j
15. Осуществить поиск сервисов, соответствующих сгенерированным расширенным описаниям
16. Если сервис найден, то **BREAK**
17. **END REPEAT** i

Степени сходства, определенные в разделе 2.2, могут значительно уменьшить количество расширенных сервисных описаний, сохраняя лишь те, которые имеют наиболее высокие сходства. Для спецификации сервиса с p параметрами должны быть найдены все их комбинации с желаемой степенью сходства. Например, если ищется нулевая степень сходства, тогда получим $(0_1,0_2,0_3,\ldots,0_p)$. Для получения сходства, равного 1, существует больше вариантов: $(1_1,0_2,0_3,\ldots,0_p)$, $(0_1,1_2,0_3,\ldots,0_p)$, …, $(0_1,0_2,0_3,\ldots,1_p)$. Основное отличие алгоритма пошаговой генераций от алгоритма исчерпывающей генерации заключается в получении только тех предков и потомков, которые соответствуют уровням сходства, определенным для каждого параметра. При этом условие продолжения поиска будет выглядеть следующим образом: если не найдено никакого сервиса для данного уровня, тогда анализируется следующий уровень.

Для уровня 1 из примера варианты следующие: $(-1_1,0_2,0_3,0_4)$ и $(0_1,0_2,0_3,1_4)$, которые сгенерируют описания соответствующие описаниям (2) и (1a). Минимальный уровень сходства вычисляется как $(\max abs(p_1), \max abs(p_2), \max abs(p_3),\ldots, \max abs(p_p))$, который, для примера, соответствует $(\max abs(-3), \max abs(0), \max abs(0), \max abs(1))=4$. Единственным описанием, соответствующим этому уровню, является (7).

4. ДКС на основе соответствия описаний с недостающей информацией

Алгоритмы, описанные выше, помогают расширить описания сервисов для их обнаружения. Согласно [13], ДКС может рассматриваться как расширение техник обнаружения сервисов. Рассмотрим случай, когда найдены только сервисы с недостающей информацией. В этом случае классификация типов недостающей информации (Табл. 1) следующая: *Сходство с недостающими выходами > Сходство с недостающими входами > Сходство с недостающими I&O*. Композиция сервиса sc_A, где $1 \le i \le |sc_A|$, включает четыре шага: выбор сервиса, генерирование расширенных сервисных описаний, обнаружение сервиса и оценка соответствия сервисных описаний. Алгоритм композиции приведен ниже:

1. Выбрать сервисы s_{Φ_i} или sc_{Φ_i}, которые имеют отношение к запросу клиента sc_i:
   - имеют большее количество выходов (недостающие выходы);
   - имеют меньшее количество входов (недостающие входы);
   - имеют большее количество выходов и меньшее количество входов (недостающие I&O).

   Добавить s_{Φ_i} или каждый элемент sc_{Φ_i} к sc_A.
2. Расширить запрос клиента $s_{C_{i+1}}$ таким образом, чтобы входы $I_{C_{i+1}}$ полностью соответствовали исходному запросу, $I_{C_{i+1}} = I_C$, а выходы $O_{C_{i+1}}$:
   - $O_{C_{i+1}} = \underline{O}_i$, недостающие выходы сервиса s_{Φ_i} либо sc_{Φ_i} относительно sc_i;
   - $O_{C_{i+1}} = \underline{I}_i$, недостающие входы сервиса s_{Φ_i} либо sc_{Φ_i} относительно sc_i.

   Это означает, что новый запрос будет искать сервис s_{Φ_i} или sc_{Φ_i}, а $s_{\Phi_{i+1}}$, либо каждый элемент $sc_{\Phi_{i+1}}$ прибавляются к sc_A.
3. Применить один из алгоритмов обнаружения сервисов, описанных выше.
4. Оценить сходство: если существуют сервисы, соответствующие sc_A, и тип сходства между s_C и sc_A относится к:
   - сходству с недостающими выходами, недостающими входами или недостающими I&O, тогда вернуться к шагу 1;
   - полному сходству или сходству с избыточной информацией, тогда композиция закончена.

Алгоритм заканчивается тогда, когда достигнуто одно из следующих условий: не существует сервисов, предлагающих недостающие входы и выходы, число попыток или время истекли.

5. Результаты экспериментов

Чтобы проверить выполнимость предложенных алгоритмов, сначала было оценено количество похожих описаний, которые удастся сгенерировать. Целью первого эксперимента стало получение оценки среднего количества входов и выходов. Были выбраны 70 Вэб сервисов с 339 методами из доступных для использования в течение месяца на сайте XMethods [14]. Для описания параметров применялись различные предметные онтологии, что обеспечило возможность сгенерировать их семантическое описание на OWL-S. Эти сервисы относились к различным категориям, как, например, Вэб поиск, мультимедийный поиск, кодирование информации, имитационное моделирования и т.д. Хотя модальное значение числа входов и выходов равно единице (Рис. 6), средние значения числа входов и выходов равны: $2.84 \cong 3$ и $1.58 \cong 2$ соответственно.

В следующем эксперименте для оценки количества описаний сервисов согласно формулам (3) и (4) были проанализированы 70 онтологий, наугад выбранных из Swoogle [15] и описанных на OWL. Их анализ позволил получить среднее количество предков и потомков относительно концепта. Как показано на Рис. 7, средние количества потомков и предков равны $90.21 \cong 90$ и $3.83 \cong 4$ соответственно.

Теперь определим количественные оценки разработанных алгоритмов. Согласно (3), общее количество сходных сервисных описаний равно [(4+1)*(4+1)*(4+1)]*[(90+1)*(90+1)]= [125]*[8281]=**1 035 125**. Однако для второго алгоритма, рассчитанного до 10 степеней сходства (d), количество сгенерированных запросов следующее: d0=0, d1=20 720, d2=51 726, d3=103 512, d4=207 025, d5=**269 132**, d6=207 025, d7=103 512, d8=51 726, d9=20 720, d10=0 (Рис. 8). Другими словами, в наиболее неблагоприятном случае (d5) количество расширенных описаний снижено до 25%. Отметим для сравнения, что число описаний, сгенерированных алгоритмом, описанным в [16], для тех же исходных данных равно **19 501 004 534 375** или в 72 458 884 раза больше.

Рис. 9 показывает экспериментальные результаты по времени вычисления предложенного алгоритма. Эксперименты проводились на ноутбуке с процессором Pentium M (1600МГц) и ОЗУ 512МБ. Экстраполяция полученных результатов показывает, что для алгоритма расширенной генерации сервисных описаний в наиболее неблагоприятном случае (d5) время обнаружения будет равно примерно 30 минутам. Естественно, это значение может быть значительно снижено в случае

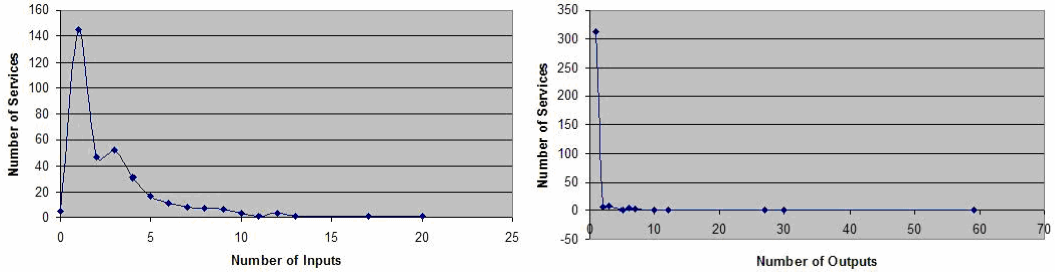


Рис. 6. Распределения количества входов и выходов сервисов зарегистрированных на XMethods [9]

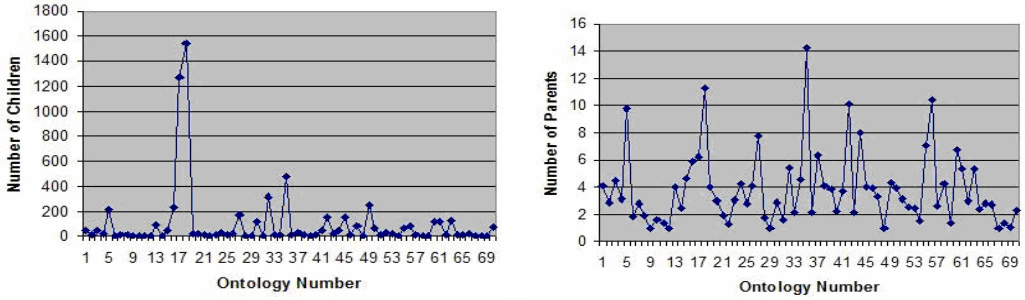


Рис. 7. Характеристики (число предков и потомков) онтологий из Swoogle [10]

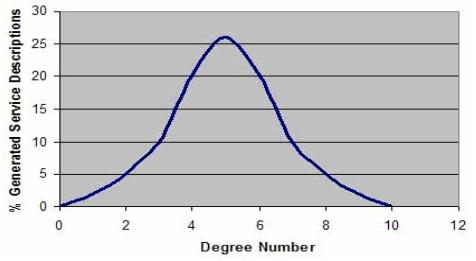


Рис. 8. Распределение сервисных описаний сгенерированных на основе степени сходства

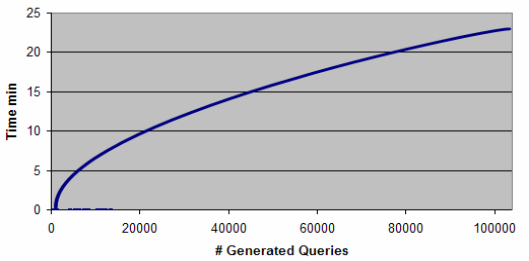


Рис. 9. Расход времени алгоритма на расширение сервисного описания (один агент)

использования более мощного вычислительного оборудования или при параллельной генерации запросов. Вместе с тем отметим, что значительную часть времени занимает вызов интефейса Protégé, использованного для поиска на онтологиях.

Заключение

На сегодняшний день в области развития веб-сервисов существует определенный разрыв между концепцией и реальностью. Это обусловлено тем, что мы еще находимся на ранней стадии эпохи сервисов, стадии концепций, когда сообщество еще пытается понять, в чем состоят суть и основные характеристики сервис-ориентированного подхода. Данная статья направлена на поиск эффективных решений динамической композиции сервисов с использованием семантики описаний их функциональных аспектов.

В статье описан подход к обнаружению сервисов, который основан на новых типах сходства сервисов (полного, с избыточной и недостающей информацией) и двух алгоритмах для исчерпывающей и пошаговой генерации сходных сервисных описаний на OWL-S. Однажды сгенерированные, эти расширенные описания могут быть обработаны традиционным синтаксическим UDDI или профильтрованы на основе их семантического сходства с тем, чтобы снизить неэффективные вычисления. Основными преимуществами предложенного подхода являются: работа с традиционным синтаксическим UDDI, увеличение сгенерированных исполняемых сервисных описаний на 25 - 40% по сравнению с известными методами [16][3], уменьшение числа сгенерированных описаний с низкими степенями сходства на 75% в наиболее неблагоприятном случае.

В настоящее время предложенная модель применяется для ДКС, в частности, для случаев обнаружения сходства в условиях недостающей информации и низкой степени соответствия. Модель используется с целью сокращения численности групп информационно-технологической поддержки при построении корпоративных сервисов, которые выступают в качестве компонентов межкорпоративных служб (например, управление сетями поставок).

Литература

1. Dong J., Paul R. A., Zhang L.-J., High-Assurance Service-Oriented Architectures. Computer, IEEE Computer Society, 41(8): 27-28, August 2008.
2. Zhu, F., Mutka, M.W., Ni, L.M.: Service discovery in pervasive computing environments. Pervasive Computing, pp. 81 --90. IEEE CS Press, Washington, DC, USA (2005)
3. Srinivasan N., Paolucci M., Sycara K.: An Efficient Algorithm for OWL-S based Semantic Search in UDDI. Semantic Web Services and Web Process Composition. LNCS, vol. 3387, pp. 96--110. Springer/Heidelberg (2005)
4. Dourdas, N., Zhu, X., Maiden, N.A.M, Jones, S., Zachos, K.: Discovering Remote Software Service that Satisfy Requirements: Patterns for Query Reformulation. Advanced Information Systems Engineering. LNCS, vol.4001, pp. 239-254. Springer Verlag (2006)

[3] Несмотря на то, что эти методы генерирует значительно большее число описаний, как показано в разделе 6, многие описания не могут быть вычислены, так как содержат неполную информацию

5. Ziembicki, J.I.: Distributed Search in Semantic Web Service Discovery. Master of Mathematics Thesis. University of Waterloo, Canada (2006)
6. Sánchez, C., Sheremetov, L.: A Model for Semantic Service Matching with Leftover and Missing Information. In: 8th Int. Conf. on Hybrid Intelligent Systems, September 10-12th, Barcelona, Spain. IEEE CS Press, Washington, DC, USA (2008)
7. Bhogalb, J., Macfarlanea, A., Smith, P.: A review of ontology based query expansion. Information Processing & Management, 43(4): 866-886. Pergamon Press, NY, USA (2007)
8. Energistics: The energy standards resource centre. URL: http://www.energistics.org/posc/General_Stds.asp?SnID=381164644
9. Биряльцев Е.В., Гусенков А.М. Онтологии реляционных баз данных. Лингвистический аспект. Компьютерная лингвистика и интеллектуальные технологии. Труды международной конференции «Диалог 2007», / Под ред. Л.Л. Иомдина, Н.И. Лауфер, А.С. Нариньяни, В.П. Селегея. - Бекасово, 30 мая - 3 июня 2007 г., М.: Изд-во РГГУ. стр. 50-53.
10. Nardi, D., Brachman, R.: An Introduction to Description Logics. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider (eds.), The Description Logic Handbook: Theory, Implementation and Applications, pp. 5--44. Cambridge Univ. Press, UK (2003)
11. Cordì V., Lombardi P., Martelli M., Mascardi V. An Ontology-Based Similarity between Sets of Concepts. In Proceedings of WOA 2005, F. Corradini, F. De Paoli, E. Merelli and A. Omicini eds. Pitagora Editrice Bologna, ISBN 88-371-1590-3, pp. 16-21, 2005.
12. Rada R., Mili H., Bicknell E., Blettner M. Development and application of a metricon semantic nets. IEEE Transaction on Systems, Man and Cybernetics 19(1), pp. 17–30, 1989.
13. Kalasapur, S., Kumar, M., Shirazi, B.A.: Dynamic Service Composition in Pervasive Computing. Parallel and Distributed Systems, vol.1818, pp. 907--918. IEEE CS Press, Washington, DC, USA (2007)
14. XMethods, URL: http://www.xmethods.net
15. Swoogle: the semantic web search engine and metadata service provider, URL: http://swoogle.umbc.edu
16. Sycara, K., Paolucci, M., Ankolekar, A. and Srinivasan, N.: Automated Discovery, Interaction and Composition of Semantic Web Services, *J. of Web Semantics*, 1(1): 27-46, Elsevier, (2003)

Шереметов Леонид Борисович. Старший научный сотрудник Санкт-Петербургского Института информатики и автоматизации Российской академии наук (СПИИРАН). Окончил Ленинградский кораблестроительный институт в 1982 году. Кандидат технических наук (1990г.). Член редакционных советов нескольких научных журналов, ассоциированный редактор журналов IEEE Transactions on Systems, Man and Cybernetics, Part C и "Computación y Sistemas". Член технических комитетов «Компьютерная поддержка коллективной работы в проектировании», IEEE Systems, Man and Cybernetics Society и «Искусственный интеллект и экспертные системы», IASTED. Автором более 250 научных публикаций. Область научных интересов – многоагентные системы, гибридные интеллектуальные системы, поиск данных и архитектуры ориентированные на семантические сервисы. E-mail: lsheremetov@mail.ru

Санчес-Санчес Кристиан. Стипендиат по гранту молодых ученых (Postdoctoral Position) Центра передовых научных исследований Национального политехнического института (НПИ), Мексика. (CINVESTAV - Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional). Окончил Высшую школу информатики НПИ в 2003 году. В 2008 году защитил диссертацию на получение степени доктора философии (PhD) в Мексиканском нефтяном институте по специальности «Информатика и прикладная математика». Область научных интересов – семантический Вэб, многоагентные системы, архитектуры ориентированные на семантические сервисы. Автор 5 научных трудов. E-mail: christiansanchezs@gmail.com.