УДК 681.3: 519.68

ОНТОЛОГИЧЕСКАЯ МОДЕЛЬ ИНТЕЛЛЕКТУАЛИЗАЦИИ СЕРВИС-ОРИЕНТИРОВАННЫХ ВЫЧИСЛЕНИЙ В РАСПРЕДЕЛЕННОЙ СРЕДЕ ИНТЕРНЕТ

Ю.В. Рогушина, А.Я. Гладун

Институт программных систем НАН Украины
Киев, проспект Академика Глушкова, 40,
тел.: 526 5139, _jjj_@ukr.net
Международный научно-учебный центр информационных технологий и систем НАН Украины и МОН Украины
Киев, проспект Академика Глушкова, 40,
тел.: 526 6344, glanat@yahoo.com

Рассмотрев базовые составляющие сервис-ориентированных вычислений в распределенной среде Интернет (WSDL, UDDI, SOAP) и проанализировав перспективы их развития, можно сделать выводы о том, что автоматизация компоновки Web-сервисов, которая должна обеспечить их значительно более широкое применение, должна базироваться на семантическом описании их функциональных возможностей. Сегодня описание семантики Web-сервисов, как и многих других информационных ресурсов распределенной гетерогенной среды Интернет, связывают с онтологическим подходом к представлению знаний (OWL-S). Однако открытыми остаются вопросы как создания онтологий, адекватно отражающих специфику определенных предметных областей, так и проблемы, связанные со сравнением и установлением соответствий между различными онтологиями. В данной работе предложено несколько подходов к расширению онтологий (в частности, для перехода от онтологии ПрО к онтологии приложения – Web-сервиса), так и к установлению подобия между различными онтологиями.

Considering the base component parts of service-oriented computation in the distributed environment of the Internet (WSDL, UDDI, SOAP) and analysing the prospects of their development, it is possible to conclude that automation of arrangement of Web-services that has to provide them considerably more wide use, must be based on semantic description of their functional possibilities. Today description of semantics of Web-services, as well as many other informative resources of the distributed heterogeneous environment of the Internet, corresponds with the ontological approach to knowledge representation (OWL-S). But questions either of adequate domain ontology creations or problems of ontology comparison and alignment are remain open. In this work a few approaches are offered for ontology development (in particular, for transition from domain ontology to ontology of applied Web-service) and for similarity retrieval between different ontologies).

Развитие информационных технологий связано с вычислениями, распределенными в сети. Возникает необходимость интеграции сервисов, предоставляемых различными разработчиками, и поиска тех сервисов, которые соответствуют потребностям пользователей – не только людей, но и различных компьютерных систем, в частности, мобильных программных агентов. Web рассматривается как набор серверов приложений, обменивающихся информацией (например, в формате XML по протоколу SOAP). Это направление развития Web тесно связано с проектом Semantic Web, целью которого является преобразование Интернет в единую распределенную базу знаний.

Концепция Web--сервисов возникла в конце 90-х годов XX века. Однако, к настоящему моменту эта концепция успела устояться и архитектура, которую она предлагает, стала отраслевым стандартом в сфере IT . Сегодня, говоря о Web-сервисах, почти всегда имеют в виду решения, которые основаны на протоколах SOAP, UDDI и WSDL. Основные источники стандартов: OASIS (oasis-open.org); W3C (w3c.org); Web Services Interoperability Organization (ws-i.org); Global Grid Forum (ggf.org); Distributed Management Task Force (dmtf.org), которые специализируются на различных аспектах проблемы. Например, деятельность рабочих групп WS-Desc и XMLP (W3C) непосредственно связана с разработкой XML-спецификаций для описания Web-сервисов, а Web Security Technical Committee (OASIS) работает над стандартизацией механизма авторизации и аутентификации с использованием XML. В рамках W3C образована рабочая группа Web Services Architecture Working Group, опубликовавшая ряд документов, в том числе глоссарий терминов из области Web-сервисов. В 2003 году была опубликована рабочая версия документа «Архитектура Web-сервисов. В истории развития Web-сервисов выделяют четыре этапа. Первый из них связан с пассивным обслуживание пользователей с помощью скриптов Perl или Shell, встраиваемых в HTML-документы. Второй и третий этапы связывают с языками Java и J2EE (Java 2 Platform, Enterprise Edition). Java-аплеты обеспечили графический, а сервлеты позволили динамически генерировать HTML-страницы, отделив представление страниц от их содержимого. J2EE позволила разработчикам использовать серверы приложений, предназначенные для разделения сервисов и бизнес-логики (HP AS, IBM WebSphere, BEA WebLogic, Sun iPlanet и Oracle 9i AS). Четвертый этап обеспечивает взаимодействие между приложениями на платформах Sun ONE (Open Net Environment) и Microsoft .Net. Сегодня можно говорить о переходе к пятому этапу, связанному с внедрением агентных технологий и динамической сборкой сервисов, а также с реализацией распределенных вычислений.

Web-сервис – набор логически связанных функций, которые могут быть программно вызваны через Интернет. В основе Web-сервисов лежат Интернет-стандарты, которые разрабатываются совместно

компаниями-разработчиками программного обеспечения. Рабочая группа "Web services" консорциума W3C рассматривает Web-сервис как программу, которая идентифицируется по URI, интерфейс которой может быть определен в виде XML-конструкций. Ими предложено следующее определение понятия ″Web-сервис″: "Web-сервис – это реализуемая программными средствами система для поддержки межмашинного взаимодействия через сеть. Интерфейс сервиса описывается на языке, читаемом машиной, например, WSDL. Другие системы взаимодействуют с Web-сервисом способом, указанным в его описании, используя сообщения в стандарте SOAP, передаваемые с использованием HTTP и XML и в сочетании с другими стандартами, относящимися к Web". Кроме того, описание сервиса содержит указание на одну или несколько точек в сети (endpoint), откуда доступен поставщик. Web-сервис предоставляет некоторые функции от имени его поставщика (человека или организации) с помощью соответствующего агента.

Постановка задачи

Цель работы состоит в том, чтобы проанализировав принципы и технологию сервис-ориентированных вычислений, области применения Web-сервисов, и на основе этого предложить методы анализа семантики Web-сервисов, направленные на автоматизацию их компоновки, использующие онтологический анализ.

Стандарты Web-сервисов

Web-сервисы базируются на трех основных Web-стандартах: SOAP (Simple Object Access Protocol) - протоколе для посылки сообщений по протоколу HTTP и другим Интернет-протоколам; WSDL (Web Services Description Language) – языке для описания программных интерфейсов Web-сервисов; UDDI (Universal Description, Discovery and Integration) – стандарте для индексации Web-сервисов. Серверы приложений являются хранилищами Web-сервисов и делают их доступными через протоколы HTTP GET, HTTP POST и HTTP SOAP. Стек технологий, реализующий архитектуру Web-сервисов, показан на рис. 1.

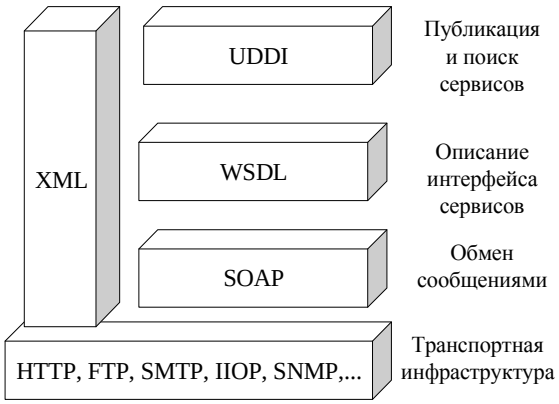

Рис.1. Стек технологий архитектуры Web-сервисов

SOAP (Simple Object Access Protocol) [1] — стандарт передачи сообщений по Интернет, разработанный фирмой Microsoft для удаленного вызова процедур (RPC, Remote Procedure Call) по протоколу HTTP. Он позволяет передавать информацию по сети в формате XML. При этом могут использоваться любая сеть, любой протокол передачи данных, произвольная информация (заказы, прогноз погоды, выписки об уровне товарных запасов и т.д.), разнообразные вычислительные устройства (в том числе мобильные). Спецификация SOAP определяет XML-«конверт» для передачи сообщений, метод для кодирования программных структур данных в формате XML, а также средства связи по протоколу HTTP.

Используют два различных типа SOAP-сообщений: запрос (Request) и ответ (Response). Запрос вызывает метод удаленного объекта, ответ возвращает результат выполнения данного метода.

WSDL (Web Services Description Language) [2] – основанный на XML стандарт, предложенный Консорциумом W3C для описания того, как пользоваться сервисом. Описание Web-сервиса на WSDL содержит все технические детали, необходимые для интеграции Web-сервиса в приложение. Оно включает формат сообщений, операции и не зависит от языка программирования.

Описание сервисов представляет собой XML-документ, состоящий из нескольких элементов, в том числе из описания пространства имен, описания типов и элементов, сообщений, порта, а также возможных операций – запросов и ответов. Они хранятся в WSDL-документах, которые располагаются либо на сервере приложений, либо в специальных XML-хранилищах. WSDL-документ может ссылаться на другие WSDL-документы и документы XML Schema, в которых описаны типы данных Web-сервисов. Для управления WSDL-документами используются XML-хранилища. Внутри WSDL-документа находится URL – адрес Web-сервиса. На данный момент язык WSDL поддерживают продукт от Microsoft – SOAP Toolkit 2.0 (его часть WSDL Generator) и продукт от IBM – WSDL Toolkit. Язык описания Web-сервисов (Web Services Description Language, WSDL) только определяет синтаксис того, как Web-сервис может быть вызван; он не говорит ничего о его семантике. Это вызывает ряд проблем при использовании Web-сервисов.

Стандарт *UDDI* (Universal Description, Discovery, and Integration) [3] предоставляет механизм для обнаружения Web-сервисов. Он обеспечивает метод описания бизнес-процесса и предлагаемых им Web-сервисов на основе XML. UDDI позволяет описать бизнес-процесс любой сложности, разложив его на составные элементы. UDDI формирует бизнес-реестр (UDDI Business Registry), в котором провайдеры Web-сервисов могут регистрировать свои сервисы, а разработчики – искать необходимые им сервисы. Компании сами регистрируют себя в Business Registry, который представляет собой базу данных общего пользования. Технология UDDI дает возможность поиска и публикации нужного сервиса, как человеком, так и программой-клиентом. Можно провести параллель между бизнес-реестром и индексом ИПС. На сегодня существуют отдельные UDDI-реестры таких компаний, как IBM, Microsoft, Ariba и Hewlett-Packard, которые в перспективе будут объединены в единый Web-реестр.

UDDI позволяет описывать, искать, интегрировать и публиковать сервисы. UDDI как универсальный метод описания, обнаружения и интеграции, сам является особым Web-сервисом, который позволяет пользователям и приложениям находить необходимые им сервисы. Это официальный стандарт OASIS, основанный на XML и SOAP. Он поддерживает операции создания, модификации, удаления и поиска элементов всех четырех типов, выше рассмотренных. UDDI не создает самодостаточный сервис, но обеспечивает техническую возможность для поиска требуемых сервисов. Он построен на основе сообщений (соединения осуществляются путем передачи XML-документов) и поддерживает распределенные приложения (даже если справочная база данных относительно централизована).

Сервис-ориентированные вычисления

Сервис-ориентированная архитектура (СОА) – это парадигма проектирования, разработки и управления функциональных модулей (сервисов), каждый из которых доступен через сеть и способен выполнять определенные действия [4]. СОА создает коммуникационную среду для модулей, реализующих прикладную бизнес-логику. Информация о модулях публикуется в такой форме, что их использование не требует знаний об использованных в них решениях и технологиях. От разработчика не требуется знать, как работает программа, необходимо лишь понимать, какие входные и выходные данных нужны, и как вызываются эти программы для исполнения. СОА сокращает время и стоимость реализации проектов.

СОА можно реализовать и без использования Web-сервисов, однако сегодня Web-сервисы рассматриваются в качестве основного средства для создания подобной архитектуры. Понятие архитектуры, ориентированной на сервисы, сложилось в ходе развития концепции Web--сервисов, однако, существуют и другие походы к реализации СОА: Java RMI (от Sun Microsystems), CORBA (от консорциума OMG), DCOM (от Microsoft), DCE (предложенный ассоциацией Open Group), программные агенты.

Все архитектуры, реализованные этими средствами, можно назвать ориентированными на сервисы, но при этом каждая из них определяет собственные форматы и протоколы, механизмы вызовов, интерфейсы для прикладных программ. Именно недостаточная универсальность ограничивает их распространение. В сегодняшней же трактовке СОА под сервисами понимают Web-сервисы, в основе которых лежат общепринятые Интернет-технологии и развитая инфраструктура. Сервис-ориентированные вычисления – новая и динамическая область, имеющая следующие характерные особенности:

- функциональные модули приложения могут быть распределены по множеству вычислительных систем и способны к взаимодействию с использованием локальных или глобальных сетей;
- интерфейс функциональных модулей таков, что их использование не зависит от технологии или платформы, в рамках которой они реализованы;
- возможен динамический поиск и подключение нужных функциональных модулей;
- архитектура базируется на общепринятых отраслевых стандартах.

СОА предполагает наличие трех основных участников: поставщика сервиса, потребителя сервиса и реестра сервисов (рис. 2). Взаимодействие участников выглядит достаточно просто: поставщик сервиса регистрирует свои сервисы в реестре, а потребитель обращается к реестру с запросом.

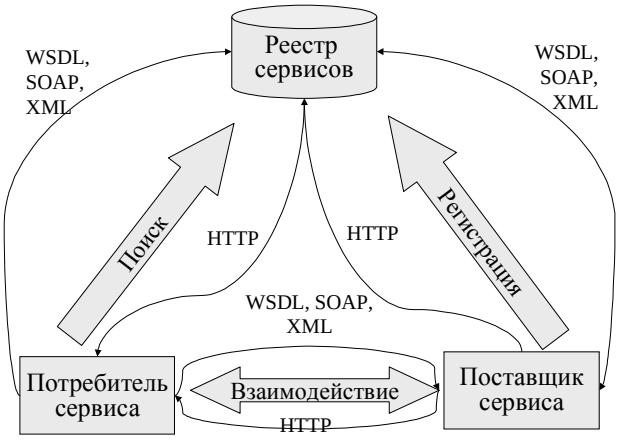

Рис. 2. Основные участники СОА

Взаимоотношения участников включает такие основные аспекты, как публикация сервиса, его поиск, подключение и использование. Для реализации СОА необходимы соглашения о форматах и протоколах взаимодействия; об описании функциональности сервиса, в виде, пригодном для автоматической обработки для определения процесса взаимодействия между клиентом и поставщиком сервиса; о способе обнаружения сервиса. СОА как система сложнее других информационных систем, и поэтому она требует для своего описания не одной, а нескольких моделей. В рамках СОА предложены такие архитектурные модели, как MOM (Message Oriented Model), SOM (Service Oriented Model), ROM (Resource Oriented Model), PM (Policy Model) и MM (Management Model). Наличие разных моделей позволит обеспечить согласованную работу специалистов разных профилей, согласование различных стандартов, образование структуры стандартов.

Стек технологий Web-сервисов вводит иерархию во множество технологий Web-сервисов в соответствии с их функциональным назначением. Стандартными названы технологии, получившие официальный статус стандартов международных консорциумов по разработке IT-стандартов (W3C, OASIS либо WS-I). В действительности, спектр технологий, описывающих аспекты использования Web-сервисов либо СОА, очень широк. *Сервис-ориентированные вычисления* (СОВ) – вычислительная парадигма, которая использует сервисы как фундаментальные элементы для разработки приложений. СОВ базируется на СОА. СОВ обеспечивает ряд операций управления сервисами, использующихся для контроля правильности и функциональных возможностей агрегированных сервисов и поддержки рынков открытых сервисов. Разработка системы СОВ – это процесс поиска, подбора и компоновки сервисов для того, чтобы удовлетворить требованиям пользователя. Компоновка сервиса – наиболее важный вопрос, так как позволяет создавать сервис из существующих частей (многоразовое использование компонент). СОВ обеспечивают способ создания новой архитектуры, которая отображает тенденции компонентов к автономии и разнородности.

Понятие архитектуры, ориентированной на сервисы, сложилось в ходе развития концепции Web-сервисов. Архитектура Web-сервисов является одной из реализаций СОА. Однако, существуют и другие походы к реализации СОА: Java RMI (от Sun Microsystems), CORBA (от консорциума OMG), DCOM (от Microsoft), DCE (предложенный ассоциацией Open Group).

Компоновка Web-сервисов

Возможность компоновки (composability) часто рассматривают как одно из основных преимуществ Web-сервисов. Компоновка Web-сервиса состоит из нахождения набора атомарных сервисов, необходимых для реализации запроса пользователя, и определение порядка их выполнения.

Компоновка Web-сервисов подобна проблеме планирования, которая исследовалась в искусственном интеллекте. Но классические планировщики непригодны для компоновки Web-сервисов, потому что предназначены для работы в статичном окружении (а среда Интернет динамична) и не имеют полной информации о системе, с которой работают (например, о возможностях различных сервисов).

Для автоматической компоновки программы должны уметь отбирать нужные им Web-сервисы и комбинировать их для достижения своих целей. Необходимо установить тесное взаимодействие между сервисами, чтобы получающийся в результате их комбинирования результат был приемлемым решением поставленной задачи. Таким образом можно строить совершенно новые сервисы, комбинируя сервисы, уже имеющиеся в сети. Информация, содержащаяся в реестре UDDI, недостаточна для того, чтобы автоматически выполнить компоновку Web-сервисов, так как необходимы усилия человека для интерпретации семантики этих сервисов. Поэтому необходимо разрабатывать механизмы отображения семантики сервисов, запросов пользователей этих сервисов и их автоматизированного сопоставления с учетом специфики предметной области (ПрО), интересующей пользователя. Для композиции Web-сервисов в динамичной среды необходимо иметь больше синтаксической и статической метаинформации о них. Web-сервисы в гетерогенной среде часто оказываются недействительными, меняют версии или заменяются другими сервисами. Поэтому Web-сервисы нуждаются в описаниях, которые обеспечивают динамический поиск и проверку.

Для описания Web-сервисов используют термины "оркестровка" и "хореография". Под *оркестровкой* понимают то, как сервисы взаимодействуют друг с другом на уровне сообщений, включая бизнес-логику и кооперацию при выполнении сложных процессов в пределах одного предприятия. Происходящие на основе обмена сообщениями взаимодействия включают бизнес-логику и порядок выполнения задач; они могут выходить за границы приложений и организаций, определяя долговременную, транзакционную, многошаговую бизнес-модель. Оркестровка основывается на открытых стандартах для создания бизнес-процессов, включая BPEL4WS (Business Process Execution Language for Web Services), WSCI (Web Service Choreagraphy Interface) и BPML (Business Process Modelling Language). *Хореография* охватывает более широкий круг участников взаимодействия, отслеживаются последовательности сообщений между участниками и источниками. Хореография ассоциируется с публичным обменом сообщениями между множеством Web-сервисов, а не с одним бизнес-процессом, осуществляемым на одном предприятии.

Web-сервисы являются функциональными блоками и соответствуют единицам работ в терминах потоков работ. Для описания систем и приложений, построенных по архитектуре Web-сервисов, на уровне модели бизнес-процесса ведущими IT-компаниями предлагались различные проекты стандартов: Wf-XML (от Workflow Management Coalition), WSFL (IBM Web Services Flow Language), XLANG (Microsoft's XLANG: Business modeling language for BizTalk), PIPs (RosettaNet's Partner Interface Process), а также некоторые другие. Стандарты хореографии и оркестровки опираются на стандартизованное описание Web-сервисов WSDL. К

настоящему моменту наиболее распространены BPEL4WS, подготовленный IBM, Microsoft и BEA Systems, и WSCI **(**Web Service Choreography Interface**)** корпорации Sun Microsystems. WSCI отражает концепцию хореографии сервисов. BPEL4WS предназначен для реализации оркестровки сервисов.

Интеллектуальные Web-сервисы

Подобно тому, как целью проекта Semantic Web является расширением обычной сети WWW, интеллектуальные Web-сервисы (называемые часто семантическими Web-сервисами, SW-сервисами) [5] расширяют понятие обычных Web-сервисов. Хотя программы могут найти некий Web-сервис в реестре UDDI без помощи человека, она не в состоянии понять, как именно им пользоваться и даже просто для чего он предназначен. Язык описания Web-сервисов WSDL даёт инструмент для описания того, каким образом взаимодействовать с тем или иным Web-сервисом, тогда как семантическая разметка предоставляет информацию о том, что и как делает данный сервис. Необходимо снабжать Web-сервисы такими описаниями, чтобы можно было автоматически распознавать их семантику. Ни WSDL, ни UDDI не позволяют программе понять, для чего именно с точки зрения клиента служит тот или иной Web-сервис, его назначение. Кроме того, программы (в частности, программные агенты, представляющие в сети интересы своего пользователя) должны уметь самостоятельно узнавать, каким образом запускать и исполнять данный сервис. Например, если выполнение сервиса представляет собой многошаговую процедуру, то программе требуется знать, как ей следует взаимодействовать с сервисом, чтобы требуемая последовательность шагов осуществилась. Программный агент должен по метаописанию Web-сервиса уметь определять его свойства данного сервиса и следить за его выполнением.

Рассмотрим проблему поиска Web-сервисов с учетом их семантики [6]. Фиксация семантики запросов и поиска сервисов, так же как контекста предложенного взаимодействия с сервисом, требует адекватных средств представлений сервисов и взаимодействий. Для этого могут быть естественно применены онтологии, которые облегчают компоновку сервисов. Наличие явного представления знаний о ПрО, к которой относится сервис, допускает переформулировку запросов контекстно-зависимым способом и переговоры о возможностях этого сервиса. Онтология состоит из терминов, их определений и атрибутов, а также связанных с ними аксиом и правил вывода. *Формальная модель онтологии* O – упорядоченная тройка O = < T, R, F>, где T – конечное множество терминов; R – конечное множество отношений между терминами из T; F – конечное множество функций интерпретации, заданных на терминах и/или отношениях онтологии O.

Для интероперабельного представления онтологий разработан язык OWL и его модификацию для описания сервисов OWL-S [7]. Цель разработки OWL-S состоит в том, чтобы сделать возможным использование логического вывода для Web-сервисов, планирование компоновки Web-сервисов, автоматическое использование сервисов программными агентами. OWL-S обеспечивает декларативные описания свойств Web-сервиса и возможности, которые могут использоваться для автоматического обнаружения сервиса, декларативный API для автоматизированного выполнения Web-сервисов, которые являются необходимыми для Web-сервисов. Функциональные возможности Web-сервиса определены входами, выходами, предварительными условиями, и действиями сервиса. Их обозначают как IOPEs (*inputs, outputs, preconditions, effects*). Функциональные возможности каждого подпроцесса могут быть описаны через его IOPE. Например, для сервиса покупки предварительное условие – это правильный ввод номера кредитной карточки, выход – генерация квитанции, а действие – оплата покупки.

Профиль сервиса строится на основе контента UDDI, описывающем свойства сервиса, необходимые для его автоматического обнаружения, такие, например, как предложение сервиса, его входы и выходы, предварительные условия и дополнительные действия. На основе профиля, который представляет информацию о провайдере, функциональных возможностях, и функциональных атрибутах сервиса, могут быть созданы описания и запросы сервиса. Профиль Web-сервиса дает описание его свойств: категорию сервиса (например, по классификации UNSPSC – международной системы классификации продукции и услуг) и его качественную оценку (скорость, надежность и т.п.).

Алгоритмы нахождения соответствия между запросом и сервисом [8], использующие онтологическое представление знаний, позволяют автоматизировать нахождение семантического подобия между запросом и описанием сервиса, несмотря на синтаксические различия между ними. Для этого запрос согласовывается на основе иерархии понятий ПрО, отображенной в онтологии. Например, запрос об автомобилях соответствует объявлению о транспортных средствах, так как автомобили включены в категорию "транспортные средства". Соответствие между описанием Web-сервиса и запросом обнаруживается, когда все выходы запроса согласованы с выходами описания, и все входы описания – со всеми входами запроса, т.е. когда сервис способен удовлетворить потребности запрашивающей стороны, и запрашивающая сторона обеспечивает все входы согласованных сервисных потребностей в его действиях.

Стандарты WSDL и UDDI не отражают семантику Web-сервисов. Интеллектуальный поиск и автоматическая композиция Web-сервисов могут быть сделаны только с использованием таких более мощных возможностей семантического описания Web-сервисов, как предложено в OWL-S. Этот язык – важный шаг по направлению к более интеллектуальным Web-сервисам. OWL-S (Web Ontology Language for Services) обеспечивают онтологическое описание Web-сервиса. Используя OWL-S, Web-сервис может сообщать о своих функциональных возможностях потенциальным пользователям. Запрос на обслуживание может быть согласован с объявлениями Web-сервисов посредством процесса подбора (matchmaking). Механизм подбора в

OWL-S не специфицирован. OWL-S, как и BPEL4WS, обеспечивает механизм для моделирования бизнес-процессов, но отличается от него по выразительности терминов, представлений, семантики, поддержки поиска и выполнения, обработки ошибок.

Чтобы обеспечить возможность композиции Web-сервисов, необходимо обеспечить пользователя описаниями, говорящими ему, что фактически значат имена типов данных и операций, используемых в синтаксическом описании сервиса. Компоновка Web-сервисов, основанных на доступных в настоящее время описаниях, как, например, WSDL, подвержена ошибкам, так как значения (или семантика) имен входных и выходных параметров Web-сервисов, используемых в этих синтаксических описаниях, не всегда ясны. Можно идентифицировать три вида проблем, возникающих вследствие семантически разнородных описаний во время композиции сервисов. Сейчас обнаружение сервисов зависит от имен входных и выходных описаний и от имен типов данных, представленных в описаниях WSDL (или других подобных метаданных сервиса). В общем предполагается, что, если имена одинаковы, то и переданная информация тоже. Однако это не всегда так. Рассмотрим проблемы разнородности, усложняющие компоновку Web-сервисов.

1. *Разнородность обозначения*. Выход Web-сервиса и вход второго Web-сервиса представлены данными одинаковых типов и ссылаются на одно и то же доменное понятие, но имеют различные метки (имена). Эта проблема может быть разрешена путем аннотации элементов, используемых в описаниях WSDL, с помощью понятий прикладной онтологии. На понятия приложения всегда ссылаются более общие понятия ПрО. Ссылка включает дополнительные ограничения на свойства понятия ПрО, которые ограничивают значение прикладного понятия. Если два прикладных понятия ссылаются на одно доменное понятие и их ограничения идентичны, то значение аннотированных символов одинаково.

2. *Разнородность типа данных*. Выход Web-сервиса и ввод второго Web-сервиса имеют одинаковые имена и ссылаются на одно доменное понятие, но представлены различными типами данных. Эта проблема не может быть просто рассмотрена как синтаксическая разнородность до тех пор, пока значение информации, содержащейся в составном типе, не станет понятным пользователю. При работе с составными типами данных может помочь в поставке правил для преобразования их или для извлечения требуемой информации семантическое описание их структуры и содержимого.

3. *Концептуальная разнородность*. Выход Web-сервиса и вход второго Web-сервиса имеют одинаковые метки (имена), представлены одним и тем же типом данных, но ссылаются на различные доменные понятия (например, расстояние могут быть представлены в разных единицах измерения). Это может привести к созданию составного сервиса, который производят результаты, не планируемые пользователем. Из-за этого в настоящее время большинство составных сервисов формируются вручную, используя основанные на WSDL описания сервисов. Если входы и выходы операций имеют одинаковые имена и типы данных, то пользователь не может сказать, что эти операции ссылаются на различные доменные понятия. Основное предположение, вызывающее эту проблему, заключается в том, что, если некие объекты описаны одинаково, то они должны иметь одинаковое значение.

Решить эти проблемы позволяет использование семантических ссылок *(Semantic Reference System)* для аннотирования символов, т.е. ссылок имен параметров на понятия онтологии ПрО и семантического обоснования этих понятий [9]. Система семантических ссылок имеет трехуровневую архитектуру (рис. 3).

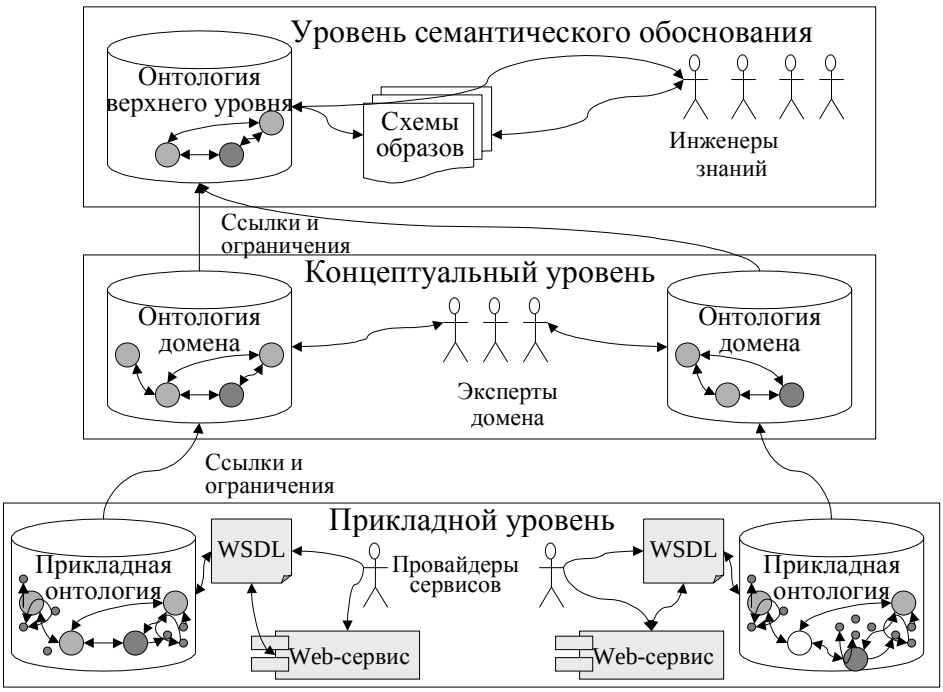

Рис. 3. Архитектура системы семантических ссылок

Доменная онтология обеспечивает согласованное осмысление (концептуализацию) определенной части мира – ПрО. Чтобы не определять одни понятия через другие, необходим уровень семантического обоснования. Семантическое обоснование – это процесс ссылки понятия, описанного на концептуальном уровне, на понятия, значения которых являются общими для всех пользователей и поэтому не требуют дальнейшего определения. Определение значения символа через ограничения, сформулированные для него на концептуальном уровне, и ссылки на схему образов сокращают семантическую двусмысленность. Прикладной уровень предназначен для описаний Web-сервисов, т.е. метки, используемые в WSDL-описании сервиса, фиксируются в прикладной онтологии. Прикладная онтология использует символы, к которым значения уже назначены, и соединяют их с семантически свободными символами из описаний WSDL. Доменная онтология может определять понятия слишком обобщенно для конкретного приложения, а прикладной уровень позволяет ограничить значения понятий и ввести понятия, не содержащиеся в доменной онтологии.

С развитием СОВ были предложены несколько различных подходов к поиску сервисов, основанных на описании сервисов, ссылающихся на онтологии (например, [10]). С этими подходами связано использование механизмов вывода для оценки взаимосвязей категоризации между понятиями онтологии, которые используются для аннотирования описания синтаксиса сервиса. В работе [11] представлены классификации архитектур онтологии, используемых для явного создания семантики источников информации (рис. 4).

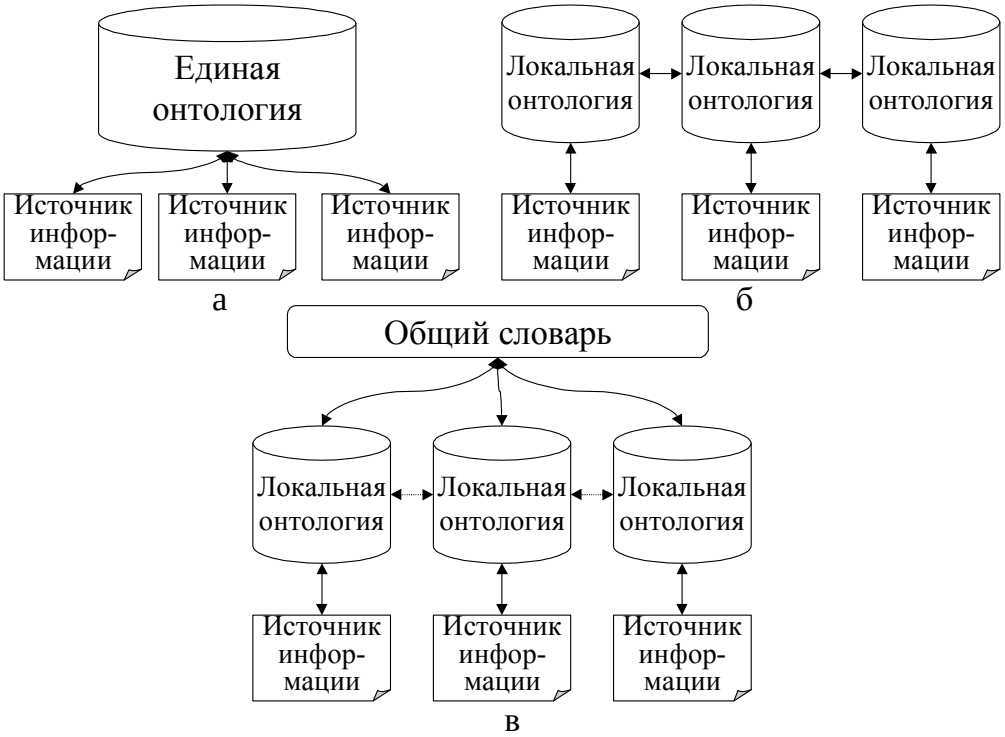

Рис. 4. Различные архитектуры онтологий для информационной интеграции

а – *подходы, основанные на единой онтологии*, используют одну глобальную онтологию, обеспечивающую совместный словарь для спецификации семантики, с которой связаны все источники информации. Такие подходы могут применяться в тех случаях, когда все источники информации, которые надо объединить, используют похожие взгляды на ПрО.

б – в *подходах, основанных на множественных онтологии*, каждый источник информации описан своей собственной онтологией. Так как каждая прикладная онтология может развиваться независимо, отсутствие общего словаря затрудняет их сравнение.

в – *гибридные подходы* подобны подходам, основанным на множественных онтологиях в том, что семантика каждого исходного текста описана ее собственной онтологией, но для сопоставления локальных онтологий формируется общий словарь.

Общий словарь (тезаурус) содержит базовые термины (примитивы) ПрО, которые комбинируются в локальные онтологии для того, чтобы описать более сложную семантику. Иногда общий словарь также является онтологией. Подходы, предложенные в настоящее время для исследования семантики сервисов, используют простые архитектуры, которые могут быть классифицированы как подходы, основанные на единой онтологии. Одна и та же ПрО может иметь несколько онтологий, поскольку информация о ПрО доступна экспертам лишь частично. Любая ПрО характеризуется своей действительностью, т.е. множеством ситуаций, которые имели место в прошлом, имеют место в настоящем и будут иметь место в будущем [12].

При разработке онтологических моделей и использовании их исследователи сталкиваются с тем фактом, что между онтологиями существуют различные отношения. Поэтому возникает проблема излучения таких

отношений между онтологиями одной и той либо разных ПрО. В частности, такая проблема возникает, если запросы пользователей и представленные в реестре Web-сервисы ссылаются на различные онтологии: необходимо установить, соответствуют эти онтологии одной или же различным ПрО, а в том случае, если они отражают один домен, – то какие отношения существуют между их понятиями. Одним из приемов, который для этого используется, является интеграция онтологий, которая определяется как конструирование некоторой онтологии С, формально специфицирующей объединение словарей двух других онтологии А и В.

Соответствие между терминами может быть лишь одним из приемов для установления отношений между онтологиями. Если установлено некоторое соответствие между понятиями двух онтологии, то только этот факт еще не дает права называть соответствующие понятия эквивалентными. Онтологии разных ПрО могут быть похожи друг на друга, а некоторые являются упрощениями других. Для того чтобы быть уверенным, что А и В могут быть интегрированы на некотором уровне, расширение понятий из А и В должно допускать отображение на расширение понятий из С. Интеграция онтологий имеет более глубокий уровень, чем интеграция представлений: интеграция представлений имеет дело с разнородностью формальных языков или схем баз данных, а интеграция онтологий - с разнородностью концептуализации.

Интеграция информации является главной областью применения онтологий. Даже если две системы используют один и тот же словарь, нет никаких гарантий, что они одинаково трактуют некоторую информацию, если не относятся к одной и той же концептуализации. Предполагая, что каждая информационная система (и, в частности, каждый Web-сервис) имеет свою собственную концептуализацию, необходимым условием одинаковой трактовки является перекрытие моделей концептуализации. Если предположить теперь, что эти два множества подразумеваемых моделей аппроксимируются двумя разными онтологиями, то возможен случай, когда две онтологии пересекутся, тогда как подразумеваемые модели нет. Следовательно, представляется более удобным согласиться на объединяющую онтологию верхнего уровня, чем полагаться на соглашения, основанные на пересечении различных онтологии. Таким образом, в процессе сопоставления запроса с описаниями Web-сервисов необходимо отслеживать иерархию онтологий, на которые ссылаются имена их параметров (от онтологий приложений через онтологии ПрО до онтологии верхнего уровня). Чем ниже уровень, на котором установлено соответствие между онтологиями, тем выше вероятность того, что выбранный Web-сервис пертинентен потребностям пользователя, а его вызов приведет к ожидаемым результатам. Интеграцию онтологий можно рассматривать как процесс нахождения общности между двумя различными онтологиями А и В и получения новой онтологии С, которая облегчает взаимодействие между компьютерными системами, основанными на онтологиях А и В (в данном случае – между онтологией, на понятия которой ссылаются имена параметров Web-сервиса, и онтологией ПрО, интересующей пользователя). Новая онтология С может быть использована как посредник между системой, основанной на А, и системой, основанной на В. В зависимости от изменений, которые необходимо сделать, чтобы получить С из А и В, можно различать следующие уровни интеграции: соответствие (alignment), частичная совместимость (partial compatibility), усовершенствование и унификация (unification).

Соответствие – отображение понятий и отношений между двумя онтологиями $A = < T_A, R_A, F_A >$ и $B = < T_B, R_B, F_B >$, которое сохраняет частичный порядок подтипов как в А, так и в В. Если соответствие отображает некоторое понятие $x \in T_A$ или отношение $x \in R_A$ из онтологии А в понятие $y \in T_B$, или отношение $y \in R_B$ в онтологии В, $\exists f(x) = y, \exists g(y) = x$, то говорят, что x и y эквивалентны. Отображение может быть частичным: может быть много понятий в А или В, для которых не существует эквивалентов в другой онтологии. *Частичная совместимость* – это соответствие онтологий А и В, которое поддерживает эквивалентные выводы и вычисления для всех эквивалентных понятий и отношений. Если А и В являются частично совместимыми, то любой вывод или вычисление, выражающие в одной онтологии, только соответствующие понятия и отношения, могут быть транслированы в эквивалентный вывод или вычисление в другой онтологии.

$$\exists f(x) = y, \exists g(y) = x, \exists x_i \in T_A, y_i \in T_B, i = \overline{1,n}, f(x_i) = y_i \Rightarrow \forall \varphi(x_1,...,x_n) \exists \gamma : \gamma(y_1,...,y_n) = \varphi(x_1,...,x_n).$$

Усовершенствование - это соответствие каждого понятия онтологии А некоторому понятию онтологии В, которая называется усовершенствованием А: $\forall x \in T_A \exists y \in T_B$. Каждое понятие из А должно соответствовать эквивалентному понятию из В. Усовершенствование определяет частичный порядок между онтологиями: если В является усовершенствованием А, а С - усовершенствованием В, то С является усовершенствованием А; если две онтологии являются усовершенствованиями друг друга, то они изоморфны.

Унификация – это взаимно-однозначное соответствие всех понятий и отношений в двух онтологиях, которое позволяет любой процесс вывода или вычислений, выраженных в одной онтологии, отображать в эквивалентный процесс вывода или вычислений в другой. $\forall x \in T_A \exists y \in T_B$, $\forall y \in T_B \exists x \in T_A$, $\forall x \in R_A \exists y \in R_B$, $\forall y \in R_B \exists x \in R_A$. Обычным способом унификации двух онтологий является усовершенствование каждой из них в более детальные онтологии, чьи категории взаимно-однозначно эквивалентны. Примером такой онтологии может быть объединение С прикладных онтологий А и В различных Web-сервисов, являющихся расширением одной онтологии ПрО. $C = \langle T_A \cup T_B, R_A \cup R_B, F \rangle$.

Соответствие является наиболее слабой формой интеграции: она требует минимальных изменений, но может поддерживать только ограниченные виды взаимодействий. Она полезна для классификации и информационного поиска, в частности, для поиска Web-сервисов, соответствующих запросу, но она не

поддерживает глубокие выводы и вычисления. Частичная совместимость требует больших изменений, чтобы поддерживать более широкую способность к взаимодействию, хотя могут существовать некоторые понятия или отношения в той или другой системе, которые будут препятствовать полному взаимодействию. Унификация или полная совместимость могут потребовать значительных изменений или полной реорганизации А и В, но в результате может быть получена наиболее полная способность к взаимодействию: все, что может быть сделано в одной, может быть сделано полностью эквивалентным способом и в другой.

Вышеизложенные подходы оставляют открытыми вопросы о том, кто и каким образом формирует онтологии ПрО, с понятиями которых связаны имена параметров Web-сервиса, и как строится описание Web-сервиса в OWL-S (разработчики и провайдеры Web-сервисов не обязаны владеть онтологическим анализом и знать инструментальные средства создания онтологий). В связи с этим актуальной задачей представляется разработка средств и методов автоматизированного формирования онтологий по информационным ресурсам, соответствующим определенному Web-сервису. Важным вопросом является также создание общего словаря (тезауруса) ПрО, обеспечивающего взаимопонимание пользователей и разработчиков Web-сервисов. Кроме того, очень актуальна разработка эффективных алгоритмов сравнения онтологий, которые, возможно, являются различными концептуализациями одной и той же ПрО – для нахождения соответствия между онтологиями пользователей и разработчиков Web-сервисов. В ряде случаев пользователи и разработчики могут воспользоваться готовыми онтологиями, но поиск таких онтологий также является нетривиальной задачей при нестандартном терминологическом базисе. Иногда ПрО приложения настолько специфична, что требует значительного уточнения и расширения для уровня приложения самим пользователем. Предлагается использование методов индуктивного обобщения для автоматизированного извлечения онтологических знаний о ПрО из набора информационных ресурсов релевантных этой ПрО.

Методы сопоставления онтологий

При наличии онтологических описаний как Web-сервисов, так и запросов пользователей возникает проблема сравнения этих описаний. Если Web-сервисы и запросы ссылаются на одну онтологию, то можно легко установить, связаны ли имена параметров атомарных сервисов с одним понятием онтологии или с разными. В противном случае необходимо установить, являются ли понятия различных онтологий эквивалентными (например, синонимами) или находятся в иерархических отношениях (например, являются подклассом).

В общем случае это довольно сложная задача, которая имеет высокую вычислительную сложность. Наш подход к ее решению базируется на предположении, которое в поиске информации используются относительно небольшие и простые за структурой фрагменты онтологий, которые характеризуют семантику конкретных ИР и запросов пользователей.

Алгоритм сравнения таких онтологий состоит из следующих этапов:

1. Построение пересечения терминов онтологий Web-сервиса и запроса $T(O) = T(O_s) \cap T(O_q)$.

2. Если это пересечение не пусто, для каждого термина из T(O) строятся два множества T_s и T_q- термины, которые связанные с ним в каждой онтологии любыми отношениями.

3. Для каждого термина из T(O) строится пересечение множеств T_s и T_q.

4. Анализ типов отношений между терминами из T(O) и пересечения множеств T_s и T_q (все отношения онтологии делятся на 3 типы – иерархические, синонимические и прочие).

5. Строится коэффициент сходства онтологий, который является количественным отображением сходства семантики двух онтологий. При этом учитываются следующие факторы: вхождение одного и того же термина в обе онтологии; то, что два термина находятся в разных онтологиях в одном и том же отношении; то, что два термина находятся в разных онтологиях в отношениях одного типа или разных (например, в иерархическом отношении и отношении синонимии); существуют ли вообще любые отношения (прямые или опосредствованные) между одними и теми же терминами. Для этого используют статистические методы, нечеткую логику, интенсиональные отношения и эмпирические правила.

6. Строится коэффициент подобия запроса и Web-сервиса – аналогично п.5, но учитываются только термины из $T(O) = T(O_s) \cap T(O_q)$, на которые ссылаются имена параметров Web-сервиса. Если полученный коэффициент выше определенного пользователем коэффициента доверия, то считается, что Web-сервис удовлетворяет потребностям пользователя и может использоваться при компоновке составного Web-сервиса.

Алгоритм реализован как программный модуль OntoWeb на языке J2EE 2 SDK Standard Edition Version 1.4.2_03 в среде NetBeansTM IDE 3.6 с использованием библиотеки JENA Semantic Web Framework Version 2.1. При разработке тестовых онтологий применялся редактор Protégé-2000.

Методы индуктивного обобщения

Методы индуктивного обобщения позволяют путем обобщения информации о ПрО, интересующей пользователя, строить множество базовых понятий ее онтологии и устанавливать связи между ними. Путем анализа текстовых документов, релевантных ПрО, достаточно сложно установить тип отношений между понятиями (это требует знаний о синтаксисе и морфологии конкретного естественного языка), однако в большинстве случаев достаточно просто указать пользователю на наличие связи. А e-learning тип он определяет самостоятельно. Индуктивные методы позволяют также дополнять и корректировать онтологии

верхнего уровня и ПрО, чтобы адаптировать ее к потребностям конкретных приложений. Предполагается, что как разработчики Web-сервисов, так и их потенциальные пользователи обладают определенными знаниями о ПрО этих сервисов и имеют несколько информационных ресурсов (ИР) – текстовых либо мультимедийных документов, описывающих ее.

В простейшем случае генерируется тезаурус, который содержит термины, которые входят в состав всех (или многих) документов, которые пользователь считает релевантными ПрО, и их метаописаний. Довольно сложной проблемой при этом является подбор коэффициентов для учета веса вхождение терминов в разные элементы структуры ИР и метаописаний.

Одной из проблем, с которой встречается разработчик онтологии, является необходимость ввести все термины, которые характеризуют определенные объекты ПрО, и правильно установить отношения между ними. Рассмотрим детальнее, как индуктивное обобщение автоматизирует построение онтологии ПрО, к которой относится разработанный Web-сервис.

В результате анализа ПрО формируется I – множество ИР, которые относятся к Про. Предположим, разработчик считает необходимым ввести к онтологии новые понятия: A и его подклассы (экземпляры, свойства и т.п.) $B_1,...,B_n$. В множестве I разработчик выделяет подмножество $I(A): I(A) \subseteq I$. Потом в этом множестве $I(A)$ разработчик выделяет подмножества $I(B_1),...,I(B_n), i = \overline{1,n}$, которые характеризуют определенные понятия ПрО. На эти множества накладываются следующие ограничения: $I(B_i) \subset I(A), i = \overline{1,n}$; $\forall i, j, i = \overline{1,n}, j = \overline{1,n}, I(B_i) \cap I(B_j) = \varnothing$; $\bigcup_{i=1}^{n} I(B_i) = I(A), i = \overline{1,n}$. После этого для любого $IP_j \in A$ из множества A строится словарь $Ont_{I(A)}$ – алфавитный перечень терминов онтологии ПрО О, которые помещаются в любом из этих документов и их метаописаниях. $Ont_{I(A)}$ создается как пересечение множеств онтологических терминов ИР, что входят в соответствующее множество: $Ont_{I(A)} = \bigcap_j Ont_{IP_j}, IP_j \in A$. Аналогично создаются $Ont_{I(B_i)}, i = \overline{1,n}$. Описанием терминов $B_1,...,B_n$ могут быть множества онтологических терминов $O_{I(B_i)}, i = \overline{1,n}$, полученные из $Ont_{I(B_i)}, i = \overline{1,n}$ удалением терминов из $Ont_{I(B_i)}, i = \overline{1,n}$.

Тем не менее эти множества могут пересекаться, тогда как нам нужно найти именно те термины, с помощью которых можно отличить любое B_i от всех других. Используем для этого метод индуктивного обобщения ID3m, обобщающий алгоритм ID3 на случай произвольного количества классов и учитывающую степень доступности информации о значении различных атрибутов [13]. Примеры учебной выборки, по которым осуществляется обобщение, – это ИР из $I(B_1),...,I(B_n), i = \overline{1,n}$. Столбцам учебной выборки отвечают онтологические сроки из множества $T = \bigcup_{i=1}^{n} O_{I(B_i)}$. Значения ячеек – это количество вхождений термина в ИР. Параметром, по которому осуществляется классификация, является принадлежность ИР к определенному классу из $I(B_1),...,I(B_n), i = \overline{1,n}$. Из-за того, что алгоритм ID3m предназначен для обработки не количественных, а качественных данных, значение ячеек превращают в качественные оценки (как правило, используют три значения – "отсутствует", "есть" и "много").

На каждом шаге работы алгоритма вычисляется онтологический термин, который несет более всего информации, которая нужна для классификации ИР, по формуле:

$$C(A_m) = \sum_i \sum_j \frac{C(A_m = a_{mi}, R = R_j)}{T(A_m)} = \max_s C(A_s) = \max_s \sum_i \sum_j \frac{C(A_s = a_{si}, R = R_j)}{T(A_m)},$$ где $C(X,Y)$ – количество информации $C(X,Y) = \sum_i \sum_j p(X = x, Y = y) * \log p(X = x, Y = y)$; $p(X=x,Y=y)$ – возможность общего наступления событий X = x и Y = y; $T(A_m)$ – стоимость получения значение A_m.

Выводы

Рассмотрев базовые составляющие сервис-ориентированных вычислений в распределенной среде Интернет (WSDL, UDDI, SOAP) и проанализировав перспективы их развития, можно сделать выводы о том, что автоматизация компоновки Web-сервисов, которая должна обеспечить их значительно более широкое применение, должна базироваться на семантическом описании их функциональных возможностей. Сегодня описание семантики Web-сервисов, как и многих других информационных ресурсов распределенной гетерогенной среды Интернет, связывают с онтологическим подходом к представлению знаний (OWL-S)[14]. Однако открытыми остаются вопросы как создания онтологий, адекватно отражающих специфику определенных предметных областей, так и проблемы, связанные со сравнением и установлением соответствий между различными онтологиями. В данной работе предложено несколько подходов к расширению онтологий (в частности, для перехода от онтологии ПрО к онтологии приложения – Web-сервиса), так и к установлению подобия между различными онтологиями.

1. *W3C* Recommendation. SOAP Version 1.2.- http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.
2. *Синтаксис* WSDL. – http://allxml.h1.ru/articles/SOAP_rpc_wsdl_2.html.
3. *UDDI* – Universal Description, Discovery and Integration. – http://www.uddi.org/.
4. *Clive* Finkelstein Service-Oriented Architecture (SOA): Methods and Technologies, Addison-Wesley, Sydney: Australia: 2005. - 765p.
5. *Janssen D., Lins A., Schlegel T., Kühner M.,Wanner G.*A Framework for Semantic Web Service Retrieval.- http://www.webservice-kompass.de/fileadmin/publikationen/SSRA-NCWS04.pdf.
6. OWL-S Home Page. – http://www.daml.org/services/
7. *Martin D., Burstein M., Lassila O., Paolucci M., Payne T., McIlraith S.* Describing Web Services using OWL-S and WSDL. – http://www.daml.org/services/owl-s/1.0/owl-s-wsdl.html.
8. *Probst F., Lutzand M.* Giving Meaning to GI Web-сервис Descriptions. – http://musil.uni-muenster.de.
9. *Paolucci M., Kawamura T., Payne T. R., Sycara K.* Semantic Matching of Web-service Capabilities // Proc. of 1st International Semantic Web Conference ISWC2002, 2002. – P. 333 - 347.
10. *Wache H., Vögele T., Visser U. Stuckenschmidt H., Schuster G., Neumann H., Hübner S.* Ontology-Based Integration of Information – A Survey of Existing Approaches // Proc. of IJCAI-01 Workshop: Ontologies and Information Sharing, 2001. – P. 108 - 117.
11. *Клещев А.С., Артемьева И.Л.* Отношения между онтологиями предметных областей. Ч. 1. // Информационный анализ. - 2002. – В.1, С.2, С. 4-9.
12. *Рогушина Ю.В.* Применение методов индуктивного вывода для создания прикладных экспертных систем // Разработка и использование информационных технологий в системах управления. – Киев: Ин-т кибернетики им. В.М. Глушкова АН Украины, 1993. – С. 122 - 128.
13. *Гладун А.Я.* Сервіс-орієнтовані обчислення – нова парадигма інтеграції інтелектуальних послуг в Internet // Економіка і управління, ЄУ, 2005, № 4. – С. 112 - 120.