УДК 681.3

В.А. Дерецкий

ПОДХОД К ПРОГРАММИРОВАНИЮ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ СЕМАНТИЧЕСКИХ ВЕБ-СЕРВИСОВ

Семантические Веб-сервисы пока не достигли желаемого уровня зрелости в части обеспечения интероперабельности и автоматизации. Средства поддержки SWS ограничены, наиболее продвинутые структуры, такие как WSMO, требуют дополнительного исследования и работ для построения средств программирования SWS. С принятием SAWSDL появляются дополнительные инструментарии и расширения языков и архитектур SWS. В работе представлены технологии SWS, которые не являются завершенными решениями, но определяют модели применения и исследования семантических Веб-сервисов, направленные для создания новой методологии программирования в распределенных и динамических окружениях.

Введение

Семантические Веб-сервисы являются дополнительным шагом в направлении развития и усовершенствования уже традиционных Веб-сервисов, которые определяются как сервисы, поддерживающие основанную на SOAP или RESTful (Representation State Transfer) архитектуру [1].

Сегодня существуют тысячи традиционных Веб-сервисов. Однако, возможность объединить их в полезные композиции требует сложной ручной работы. Разработчик должен исследовать каждый сервис на его функциональное соответствие или семантику, а затем решить, как извлечь эту полезность через заданный синтаксис и протоколы. Такая ручная работа строго ограничивает возможность композиции. Большинство композиций ограничено интеграцией нескольких сервисов, таких как Амазон, YahooMaps или GoogleMaps. Кроме того, ограничиваются "договорные соглашения" по различным интерфейсам Веб-сервисов. Интерфейсы могут быть изменены или удалены поставщиками сервисов в любое время, а разработчики, которые используют эти сервисы, не знают, какие изменения были внесены. В результате этого запросы к ним больше не работают. Эта ситуация делает уязвимым любой сервис. Семантическая информация непосредственно направлена на решение этих ограничений, используя однородный, машиночитаемый способ взаимодействия семантических Веб-сервисов в Веб.

SWS обеспечивает динамическую машинную обработку в части поиска, запуска, переговоров и композиции Веб-сервисов с целью достижения некоторой цели пользователей. Потоковая модель обработки запросов (WorkFlow), начиная от создания запроса, завершая заключительной поставкой результатов, является достаточно нетривиальным процессом и требует сложных структур данных.

Сегодня WorkFlow, основанные на «шине» сервисов предприятия (ESB), используют сервис-ориентированную архитектуру (SOA), применяя высокоуровневое ручное вмешательство для того, чтобы построить поток данных. Неспособность сервисов использовать в своих интересах конкурентное превосходство поставщика сервисов, динамическое определение предпочтений одного сервиса относительно другого является недостатком традиционных архитектур. Определение сервисов, которые доступны через Веб, является начальным шагом; последующие задачи состоят в развертывании этих сервисов и определении соответствующих метаданных, которые обеспечивают обработку сервисов машинным способом.

В этой работе рассмотрены некоторые главные технологии SWS, предлагаемые и используемые сегодня. SWS – это развивающаяся область, где не был достигнут баланс выразительности и полноценности. Вначале исследуем основные признаки семантических Веб-сервисов, а

затем три предложенных решения: семантическая разметка для Веб-сервисов (OWL-S), онтологическое моделирование Веб-сервисов (WSMO), и семантические аннотации для WSDL и схемы XML (SAWSDL). Каждое из решений содержит некоторые возможности SWS, но каждый из них не стал полностью неоспоримым решением в этой области. Эти три технологии формируют модель понимания и развития SWS. Их использование связано с тем, чтобы позволить решить некоторые ранее отмеченные проблемы; однако, следует отметить, что и семантическая сеть, и семантические Веб-сервисы получают развитие от их частичного или полного применения.

1. Этапы и модели жизненного цикла семантических Веб-сервисов

Не существует единственного определения Веб-сервиса из-за того, что существует много сторон восприятия, каждая из которых определяет свое понятие Веб-сервиса. Консорциум W3C определяет Веб-сервис [2] как часть архитектуры, которая включает следующее определение:

Веб-сервис представляет собой систему программного обеспечения, разработанную для поддержки взаимодействий машина-машина по сети. Он имеет интерфейс, представление в машинно-обрабатываемом формате (определенно как WSDL). Другие системы взаимодействуют с Веб-сервисом в соответствии с его описанием, используя SOAP, передаваемое с использованием HTTP протокола с преобразованием в последовательную форму XML, используя другие связанные с сетью стандартами.

Это определение Веб-сервиса не покрывает архитектуру RESTfull, но если сосредоточиться на первой части предложения, то определение соответствует: Веб-сервис – система программного обеспечения, разработанная для того, чтобы поддержать взаимодействие машина-машина по сети. Кроме того, сервис должен что-то выполнить как часть такого взаимодействия: потреблять, обеспечивать или преобразовывать данные, или возможно переключать события.

Главная проблема с традиционными Веб-сервисами состоит в том, что они определяются своим синтаксисом и совершенно не определяются семантикой своих операций. Сервисы могут декларативно перечислять информацию о необходимых входах, выходах, и то как они будут управлять коммуникациями. Но этой информации недостаточно чтобы определить, какие данные или операции Веб-сервисов основаны исключительно на синтаксисе. Например, Веб-сервис BearNews мог бы определить в WSDL метод, который учитывает параметр dateTime как входной и возвращает список заголовков, которые являются последними новостями Bears.

Вторичная проблема Веб-сервисов состоит в том, что они не поддерживают достаточный уровень машинной автоматизации. Разработчики могут закодировать Веб-сервисы и развернуть их, например, в TomCat. Кодировать и издавать сервисы в некотором реестре, используя UDDI таким образом, чтобы результат одного Веб-сервиса (поставщик) передать как вход другому Веб-сервису (потребитель), создавая процесс, известный как оркестровка сервисов, хореография сервисов или композиция сервисов. На разработчиков и персонал ИТ возложена функция управления и поддержки рабочих структур Веб-сервисов.

На рис. 1 представлена схема жизненного цикла Веб-сервиса, в которой определены этапы, в которых используются семантические описания.

Далее рассмотрим основные этапы жизненного цикла Веб-сервиса, представленные на рис. 1: запрос, поиск, переговоры, обработка ошибок, мониторинг и композиция.

Поиск. Сервисы используют информационные сервера для саморегистрации и нуждаются в информации о других доступных сервисах, которые обеспечивают такие же функциональные возможности как и запрашиваемый сервис через глобальную сеть зарегистрированных сервисов. Для этой роли используют програм-

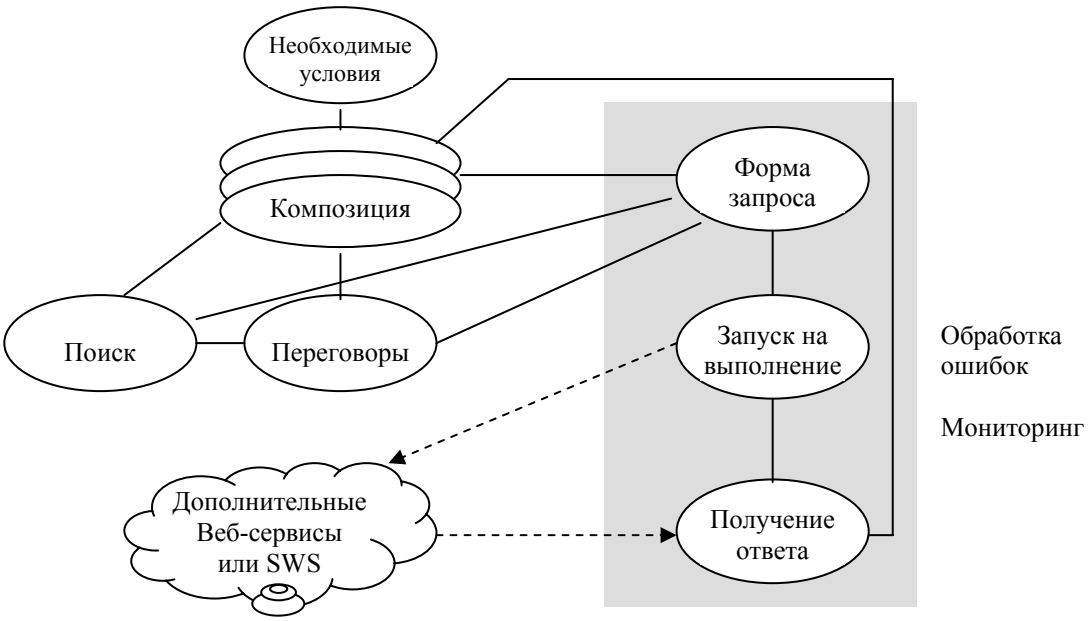

Рис. 1. Жизненный цикл Веб-сервиса

мные агенты поиска. Такую информацию мог бы обеспечить семантический сервис, который должен знать, когда и как зарегистрировать себя, какие темы должны быть включены и организованы, и где можно найти метаданные сервиса. Существующие технологии не обеспечивают такие возможности. Например, реестры UDDI не являются столь общепринятыми, как первоначально предполагалось, а централизованная природа UDDI не всегда согласовывается с децентрализованной природой Интернет.

Запрос. Он определяет описание вызова функциональности сервиса на выполнение. Он включает знания о том, где сервис размещен (разрешимый URIs, детализированный в WSDL, и полученный через UDDI), что должно быть на входе, и что должно получиться в результате выполнения сервиса. Для достижения желаемого результата сервис может потребовать выполнения ряда дополнительных операций кроме единственного вызова метода. Эти шаги могли бы быть декларативно определены и сохранены для машинной интерпретации, и включать семантику, которая добавляется для того, чтобы определить, когда нужно вызывать процедуры, как интерпретировать результат, как использовать нерегулярные данные и др.

Переговоры. Они используют широкий набор подходов и протоколов, от трастов до контрактов. Используются переговоры товарообмена, переговоры соглашения стоимости, сервисного согласования (SLAs), качества данных, количества данных и т. п. Внедрение SWS или агентов, которые управляют переговорами с использованием алгоритмов принятия решений сервисами в интересах их клиентов. Например, предположим, что пользователь хочет получить сведения о погоде каждые 5–10 минут, если нет чрезвычайных ситуаций, таких как ураган. В случае чрезвычайных ситуаций пользователь хочет получать информацию о погоде каждые 60 секунд. Это становится возможным, если использовать семантику, чтобы получить дополнительную информацию и определить, какие сервисы могут обеспечить этот уровень требований. Другой пример использования сервисами условий стоимости, которые не могут быть определены только с использованием синтаксиса сервиса.

Обработка ошибок. С использованием семантики обработка ошибок может стать намного эффективней. Синтаксический подход на основе определения списка ошибок или исключений, когда возникает состояние ошибки, то для сервиса предопределяется ряд альтернативных

действий путем вызова сервисов, которые обеспечат процедуры предупреждения или устранения ошибок. Подход к обработке ошибок с использованием SWS может усилить условия обработки ошибок для получения различных входов, (обеспечивающих поиск дополнительных данных) и повторить запрос сервиса, таким образом, чтобы без ручного вмешательства получить безошибочную работу.

Мониторинг. Информация об успешном или ошибочном выполнении сервиса, производительности, объеме данных и других индикаторах используется для обеспечения обратной связи и усовершенствования сервиса. Семантическая разметка этой информации не может быть выполнена только синтаксическими средствами. Кроме того сервисы больших предприятий могли бы включать семантику с использованием онтологического посредничества, представляя согласованную семантику высокоуровневых сервисов.

Композиция. Композиция сервисов наиболее популярная и часто исследуемая методология программирования с использованием SWS. Чтобы продемонстрировать цель композиции, рассмотрим пару специализированных сервисов, которые свяжем в модели WorkFlow. Рассмотрим пример, в котором клиент купил товар, выбрал способ доставки, принял наиболее простой способ оплаты. Поставщик товара должен резервировать, находить и упаковывать товар, отправить его, затем обеспечить оплату, управляя продвижением товара на каждом шаге процесса. Допускается повторное использование любой из этих операций в виде единого технологического процесса, инкапсулированного в единую систему. SOA направлена на реализацию такой возможности, но сервисная инфраструктура требует регулярного участия квалифицированного персонала. Архитектуры SWS направлена на уменьшение зависимости технологии от человеческого вмешательства. Программные агенты используют декларативную семантику, которая позволяет им реагировать, контролировать, приспосабливать и выполнять сервисы, которые вручную потребовали бы многих часов информационной поддержки.

Семантические Веб-сервисы активно развиваются. Они не обеспечивают все то, что было обещано; однако, они продолжают привлекать многочисленные академические исследования. В этом контексте рассматриваются модели OWL-S, WSMO и Семантические аннотации для WSDL и XML схемы (SAWSDL). Использование WSDL и других базовых стандартов осуществляется во всех отмеченных технологиях.

2. Реализация семантических Веб-сервисов

Исследуем три подхода, используемые в технологии Веб-сервисов. Каждая технология строит онтологии для того, чтобы представить семантику, используемую на этапах жизненного цикла Веб-сервисов. Эти решения являются первым шагом, необходимым для формирования сервисов, которые смогут искать, вызывать и мониторить другие Веб-сервисы в автоматическом режиме. Онтологии отличаются не только семантикой, но и выразительностью и легкостью интеграции Веб-сервисов на основе стандартов типа WSDL.

Онтологии выражают операции сервисов через семантику. Семантические методы могут быть использованы в таких операциях как поиск, навигация, формирование и выполнение запросов к сервису. Используя возможности SPARQL, уровень выразительности в соответствии со сложностью сервисов выражает не входы и выходы сервиса, а его надежность, детали и т. п. Возможности ограничены только семантическим описанием.

Принятие онтологий и создание соответствующих сервисов все еще остаются предметом исследования. Неизвестно, приведет ли технология к желаемому результату или они будут заменены другими архитектурами. В любом случае, важно понять эти подходы, потому что они помогают выбрать путь к лучшему управлению с широкими возможностями SWS, усилить тысячи сервисов, предлагаемых в Интернете.

Семантическая разметка Веб-сервисов. OWL-S [3] не являются акронимом для OWL, а – средство для семантической разметки Веб-сервисов. OWL-S представляет собой онтологию верхнего уровня, которая представлена в соответствии с тремя высокоуровневыми понятиями [4].

На рис. 2 представлены основные компоненты, реализуемые OWL. Эти три основных концепта представляют семантику Веб-сервиса в виде классов, связанных с классом «Service». Класс ServiceProfile представляет то, что делает сервис. Класс ServiceGrounding представляет то, как получить доступ к сервису, при этом класс ServiceModel определяет работу сервиса.

Профиль сервиса. Класс Профиль сервиса (ServiceProfile) и его подклассы описывают то, что делает данный сервис. SWS ищет в сети различные сервисы, и профиль сервиса обеспечивает необходимые данные таким образом, чтобы клиент смог определить, соответствует ли сервис требованиям. Профиль может включать такую информацию, как параметры сервиса (serviceParameters) или ссылка на любой вид свойств сервиса, категории сервиса (serviceCategory), метод для классификации информационных систем. Класс Profile определен в [5]. На рис. 3 показаны свойства и ссылки, которые являются частью профиля.

Модель сервиса. Она является подклассом процесса и позволяет потенциальным клиентам понять, как сервис работает и как он должен использоваться. Модель обеспечивает абстрактные детали, такие как параметры пред- и пост- условий. Они были разработаны для того, чтобы взаимодействовать с простыми одношаговыми и композитными сервисами. Класс процесса определен в [6].

Граудинг сервиса. ServiceGrounding представляет собой процесс перехода от абстрактного профиля сервиса (ServiceProfile) и классов модели сервиса (ServiceModel) в конкретную среду реализации сервиса. Единственной основой, реализованной для OWL-S является стандарт WSDL, полученный через класс WsdlGrounding. ServiceGrounding включает отображения прямых свойств, входов и выходов между онтологией и WSDL, дополненное отображением, связывающим типы данных. WsdlGrounding определен в [7].

Все три из представленных классов определены в отдельном файле OWL-S, структура которого представлена в [8]. OWL-S реализует композицию сервисов через класс CompositeProcess. Этот класс представляет возможность выполнять Веб-сервисы гибким способом, включая последовательный или параллельный режимы использования условий.

Java-инструментарий OWL-S, названный OWL-S API, свободно доступен на сайте [9].

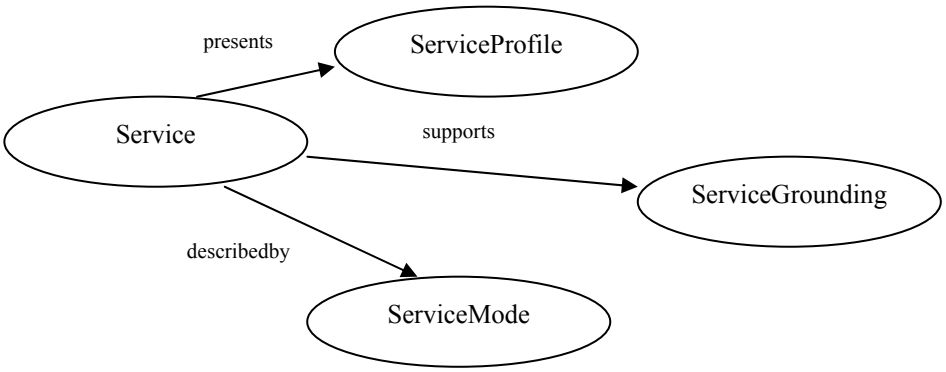

Рис. 2. Высокоуровневое представление OWL-S

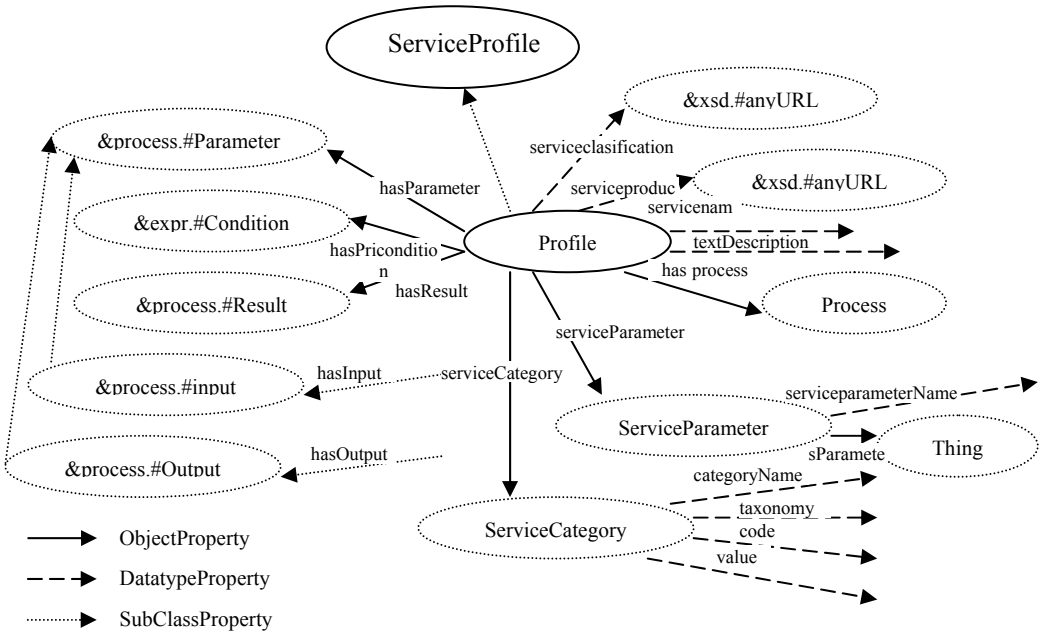

Рис. 3. RDF-схема представления профиля в соответствии с OWL-S

3. Онтологии моделирования Веб-сервисов (WSMO)

Онтология WSMO использует подобный подход детализации семантических Веб-сервисов как и OWL-S. Эта технология является более широкой, так как она соответствует требованиям автоматизации сложных взаимодействий в композициях Веб-сервисов. Составными частями WSMO являются язык моделирования Веб-сервисов (WSML), среда выполнения Веб-сервисов (WSMX), которые были представлены в материалах W3C [10 – 12].

WSMO разделяется на четыре основных компонента: онтологии, описания Веб-сервисов, цели и посредники. Онтологии WSMO представляются классами и свойствами, которые связывают все её понятия друг с другом в реализации WSMO. Последние описывают информацию о сервисах (функциональную и нефункциональную), которая может обеспечивать реализацию WSMO. Цели определяют то, что ожидают получить пользователи с помощью Веб-сервисов WSMO. Посредники представляются как мосты этих компонент (например, отражают связи онтологий или связывают цели). WSMO не основан на OWL-S, он имеет свой собственный язык, названный языком моделирования Веб-сервисов (WSML).

WSML может соответствовать конструкциям OWL-DL, RDF или XML для обеспечения гибкости. Реализация WSMO осуществляется на основе среды выполнения Веб-сервисов (WSMX). Средства WSMX доступны на сайте SourceForge в [13]. Далее кратко рассмотрим основные возможности WSMX.

Среда WSMX может быть выполнена средствами Eclipse или через интерфейс командной строки (используя Java 1.6 и Ant 1.7.0 или выше). В этом случае Eclipse будет служить средой разработки. Как только дистрибуция загружена и развернута, необходимо создать новый Java проект, основанный на wsmx-0.5-src. Необходимо проверить путь к классам Ant, он должен включать Ant-wsmx.jar и три необходимых jax*.jars (представляют Runtime Ant).

После запуска на выполнение проекта будет загружен ряд Веб-сервисов, онтологий, целей и посредников. Будет подготовлено окружение к развертыванию SWS. По умолчанию портал будет установлен по адресу http://localhost:8081 и управление будет осуществляться через JMX MBeans (Java Management Extensions Management Beans).

SWS, который развернут в WSMX обеспечивает выполнение следующих тех-

нологических процессов:
– использование существующего Веб-сервиса на основе WSDL или создание нового сервиса, если он не существует;
– создание модели WSMO для поддержки SWS;
– создание онтологии, которая описывает Веб-сервис;
– создание цели, которая определяет объект SWS;
– создание второй онтологии, которая содержит описание входов, соответствующих указанной цели;
– создание адаптеров сервисов для понижения к физическому и подъема к концептуальному уровню. Адаптер отслеживает понижение и подъем отображения и используется для того, чтобы преобразовать WSMO в RDF или XML и наоборот.

Более детальная информация относительно этих шагов представлена в документации WSMX [14]. Использование WSMO направлено на создание спецификаций WSMO-Lite и MicroWSMO. WSMO-Lite использует RDFS и SAWSDL, MicroWSMO покрывают семантические аннотации в RESTful Веб-сервисах [15].

Семантические аннотации для WSDL. В результате аннотации формируется структура SWS, которая представляет SAWSDL. Последний содержит семантические аннотации WSDL и XML схемы. SAWSDL обеспечивает общий метод связи объектов WSDL, таких как декларации элементов и определение типов, с некоторой концептуальной моделью, базируемой на OWL-S, RDF, RIF, WSML или других. Также поддерживает ссылки на такие атрибуты как подъем, понижение и детализация. В итоге, SAWSDL направлен на обеспечение интероперабельности. Схема SAWSDL ограничена тремя простыми атрибутами: схема, элемент и тип.

Пример SAWSDL. Рассмотрим пример, в котором мы используем существующий WSDL и создадим связь между некоторой операцией Веб-сервиса и онтологией, которая определяет некоторую модель запроса:

```
<wsdl:operation name="operationname"
        sawsdl:modelReference="http://www.
semwebprogramming.org#Request">
        <wsdl:input element="input"/>
        <wsdl:output element="output"/>
</wsdl:operation>
```

Мы можем так же использовать modelReference, чтобы отобразить названия параметров:

```
<xsd:element name="RequestService">
    <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="itemCode"
type="xsd:string"
        sawsdl:modelReference=
                "http://www.semwebprogram
        ming.org#Parm1"/>
        <xsd:element name="date"
type="xsd:string"
        sawsdl:modelReference="
                http://www.semwebprogrammi
        ng.org#Parm2"/"/>
        <xsd:element name="qty"
type="xsd:float"
        sawsdl:modelReference="
                http://www.semwebprogrammi
        ng.org#Parm3"/"/>
         </xsd:sequence>
      </xsd:complexType>
</xsd:element>
```

При таком отображении элементы Parm1, Parm2, и Parm3 ссылаются на модель онтологии и обеспечивают снятие омонимии этого сервиса на клиентской машине.

Request2Ont.xslt, в запросе SPARQL используется для того, чтобы собрать данные из базы данных knowledgebase, firstParam и secondParam.

```
<xsd:element name = "Request">
        <xsd:complexType
sawsdl:liftingSchemaMapping =
        "http://semwebprogramming.org/Requ
est2Ont.xslt">
            <xsd:sequence>
                <xsd:element name =
        "firstParam" type = "xsd:string"/>
```

```
                <xsd:element name =
            “secondParam“ type = “xsd:string“/>
                </xsd:sequence>
            </xsd:complexType>
</xsd:element>
```

Этот пример показывает метод усиления существующей технологии семантической структурой.

Средства SAWSDL. Кодирование WSDL и XSLT основано на SAWSDL4J, представляющее собой открытое API, который обеспечивает Java объекты для SAWSDL.

SAWSDL4J доступен на [16]. Другой API – Woden4SAWSDL обеспечивает возможности разметки WSDL 2.0, которая позволяет получить SAWSDL по заданому WSDL. Средство доступно по адресу [17].

Есть также редакторы SAWSDL, например, Radiant [18] и WSMOStudio [19]. Оба представляют собой программные расширения для Eclipse и открыты для применения.

Заключение

Представленные в этой работе технологии SWS не являются завершенными решениями. Все они определяют модели применения семантических Веб-сервисов и направлены на их программирование при решении задач автоматизации.

Семантические Веб-сервисы пока не достигли желаемого уровня в части интероперабельности и автоматизации. Средства поддержки SWS ограничены, наиболее продвинутые структуры, такие как WSMO, требуют дополнительного исследования и работ для построения средств программирования SWS. С принятием SAWSDL появляются дополнительные инструментарии и расширения языков и архитектур SWS .

1. *Андон П.І., Дерецкий В.О.* Проблеми побудови сервіс-орієнтованих прикладних інформаційних систем в Semantic web середовищі на основі агентного підходу // Проблеми програмування. − 2006. − № 2-3. − С. 493–502.
2. http://www.w3.org/TR/ws-gloss/
3. http://www.w3.org/Submission/OWL-S
4. http://www.daml.org/services/owl-s/1.2/
5. http://www.daml.org/services/owl-s/1.1/Profile.owl
6. http://www.daml.org/services/owl-s/1.1/Process.owl
7. http://www.daml.org/services/owl-s/1.1/Grounding.owl
8. http://www.daml.org/services/owl-s/1.1/Service.owl
9. http://code.google.com/p/owl-s/
10. http://www.w3.org/Submission/WSMO/
11. http://www.w3.org/Submission/WSML/
12. http://www.w3.org/Submission/WSMX)
13. http://sourceforge.net/projects/wsmx
14. http://www.wsmx.org/papers/documentation/WSMX Documentation.pdf
15. *Дерецкий В., Богданова М., Горошанский С.* Подход к разработке программных приложений с использованием семантических Веб-сервисов // Проблеми програмування. − 2009. − № 4. − С. 59–70.
16. http://lsdis.cs.uga.edu/projects/meteor-s/opensource/sawsdl4j/
17. http://lsdis.cs.uga.edu/projects/meteor-s/opensource/woden4sawsdl/
18. http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1
19. http://www.wsmostudio.org/

Получено 10.03.2010

Об авторах:

Дерецкий Валентин Александрович,
кандидат физико-математических наук,
ведущий научный сотрудник.

Место работы:

Институт программных систем
НАН Украины.
03187, Киев-187,
Проспект Академика Глушкова, 40.
Тел.: 38 044 526 4342.
e-mail: dva@isofts.kiev.ua.