УДК 681.3

В. Дерецкий

ПОДХОДЫ И ЗАДАЧИ КОМПОЗИЦИИ СЕРВИСОВ В СЕМАНТИЧЕСКОМ WEB ОКРУЖЕНИИ

Композиция Веб-сервисов вызывает значительный интерес в контексте проблематики семантической Веб-сети. Ее цели заключаются в том, чтобы сделать существующий Веб более эффективным и обеспечить гибкий подход к управлению всеми типами процессов. Ожидаемые программные приложения, построенные на основе композиции сервисов, сделают более продвинутыми такие направления как Веб-коммерция, электронное правительство, электронная наука и другие.
На сегодня композиция сервисов выполняется, как правило, вручную в процессе разработки приложений. Это требует много времени и программирования на достаточно низком уровне.
В этой работе мы рассматриваем модель композиции Веб-сервисов, которая базируется на онтологиях. Подход к генерации композитного сервиса основан на декларативных описаниях. Формальной основой для композиции служат композиционные правила. Эти правила совмещают синтаксические и семантические характеристики сервисов для определения того, являются ли сервисы композитными.

Вступление

Концепция Семантического Веб направлена на расширение существующего Веб [1–4] в части определения и обработки содержания информации. Окончательная цель Семантической сети Интернет заключается в том, чтобы превратить Веб в среду, через которую данные могут быть разделены, понятны и обработаны автоматизированными или автоматическими средствами, т.е. машинами.

Развитие технологий Семантической Веб-сети является приоритетным заданием многих исследовательских проектов. Одним из ключевых проблемных понятий, в исследованиях является понятие Веб-сервиса [5], который представляет собой программную систему, представленную множеством связанных функций, к которым можно обратиться программно через сеть. Веб-сервисы предназначены для решения проблем интероперабельности и повторного использования программ при взаимодействии машина-машина в сети. Понятие сервиса постепенно внедряется в бизнес, государственную деятельность, науку и другие отрасли. Примерами Веб-сервисов в бизнесе является электронная торговля акциями, проверка кредитов, электронные платежи. Веб-сервисы в государственных структурах обеспечивают функции различных социальных фондов.

Широко распространенные стандарты на основе XML, включая: WSDL, SAWSDL, SOAP, UDDI, WSMO и MSSLI [5–9], инициируют интенсивные исследования Веб-сервисов в промышленности и академических проектах. Одним из важнейших назначений Семантической сети Интернет является использование ее как носителя сервисов. Эта новая модель дала бы возможность компаниям значительно уменьшить расходы на развертывание новых решений бизнеса и открытия новых возможностей.

Идентифицируют два типа сервисов: простые и композитные. Простой сервис представляет программную систему, созданную на основе Интернета, которая не использует другие Веб-сервисы для выполнения запроса пользователя. Сложный или композитный сервис определяется тем, что он использует совокупность функций простых или композитных сервисов для выполнения своих заданий. Примером композитного сервиса может быть автомобильный брокер, который получает данные для выполнения работ от агента по продаже автомобилей, финансирования и страхового обслуживания для обеспечения сервиса продажи [9].

Композиция сервисов стала центральной задачей в исследованиях по Веб-

сервисам. В этой области предложено несколько новых подходов [10─12]. Как правило, все они требуют детального программирования, который делает процесс композиции сервисов зависимым от программиста-разработчика сервисов [13].

Разработчики композиции должны определить, какие действия должны быть выполнены композитным сервисом, и не интересоваться, какие сервисы необходимо вовлечь в выполнение и как вовлеченные сервисы должны взаимодействовать. Процесс композиции сервисов включает такие операции: выбор сервисов, встраивание операций, определение сообщений разговора и другие, которые должны быть прозрачными для пользователей. Детализированные описания композитного сервиса должны быть сгенерованными из спецификации композиции.

Семантика является важной и основной составляющей для обеспечения композиции сервисов, в части проверки того, что выбранные для композиции сервисы были корректными и отвечали требованиям композиции. Такие средства могут быть синтаксическими (например, проверка числа параметров, включенных в сообщение, которое посылается или получается сервисом). Они также могут быть семантическими (например, состав бизнес-функций, которые предлагаются сервисной операцией или определение соответствия предметной области). Для определения семантических характеристик Веб-сервисов используется понятие онтологии, которая определяется как концептуализация, основанная на семантической близости терминов в определенной предметной области [14]. Онтология ─ ключевое понятие для формального определения семантики и семантического управления процессами [15, 16]. Онтологии играют центральную роль в Семантической сети Интернет, расширяя синтаксическую интероперабельность сервисов до их семантической интероперабельности [17].

В данной работе рассматриваем подход и модель композиции Веб-сервисов, согласование концептов Веб-сервисов и онтологии, которые являются основой подхода. Для понимания и иллюстрации подхода композиции рассматривается приложение из электронной коммерции B2B «Продажа автомобилей» [18].

Модель композиции Веб-сервисов направлена на решение следующих задач:

определение того, являются ли выбранные сервисы композитно-реализуемыми (Composability) [4]. Композитная реализуемость определяется путем проверки возможностей сервисов взаимодействовать друг с другом. В работе представлена модель композиции, на основе которой выполняется сравнение синтаксических и семантических характеристик Веб-сервисов, которые принимают участие в композиции;

генерация композитного сервиса, при которой сохраняются вышеупомянутые характеристики композиции. Входом предложенной методики являются спецификации композиции высокого уровня. Кроме того, спецификация содержит список операций, которые будут выполнены в процессе композиции.

Рассматривается архитектура и прототип системы композиции сервисов, в основе архитектуры используются такие стандарты Веб-сервисов: WSDL,SAWSDL, UDDI, SOAP, WSMO и др.

1. Семантические спецификации Веб-сервисов

Процесс создания Веб-сервиса требует его описания для того, что бы другие сервисы могли понять его возможности и узнать, как взаимодействовать с ним. Язык WSDL (язык описания Веб-сервисов) предназначен для описания операционных особенностей Веб-сервисов. WSDL стандартизируется в пределах консорциума W3C [6]. Главные лидеры промышленности поддерживают и принимают участие в развитии WSDL. Однако WSDL почти не обеспечивает семантическое описание Веб-сервисов. Она главным образом вклю-

чает конструкции, которые описывают Веб-сервис с синтаксической точки зрения. Чтобы отвечать определению Семантического Веб-сервиса, в работе используется подход, основанный на расширении WSDL семантическими характеристиками. Этот подход положен в основу выбора и композиции Веб-сервисов с использованием стандартов WSDL 2.0, WSDL 4j и SAWSDL [5, 6].

Другой ключевой момент подхода заключается в использовании онтологии для Веб-сервисов, которая определяется с помощью языка DAML+OIL. Он использует объектно-ориентированный подход, описывая онтологию в терминах классов, свойств и аксиом. Язык DAML+OIL основывается на предыдущих стандартах Веб-сервиса и онтологии, типа RDF и RDF-S, и расширяет эти языки примитивами моделирования (например, количество элементов). Для определения онтологии могут использоваться другие онтологические языки, типа OWL, OWL-S, WSMO [14].

Для моделирования онтологии воспользуемся моделью, которая базируется на ориентированных графах, узлы которого представляют концепты онтологии. Множество узлов графа разделяется на «заполненных» и «незаполненных». «Незаполненные» узлы ссылаются на концепты, которые определены языком WSDL (например, имена, связи, входы). «Заполненные» узлы ссылаются на описания, полученные с помощью расширения WSDL семантическими характеристиками. Дуги представляют отношение между понятиями онтологии. Их маркируют количеством элементов соответствующих отношений. Например, дуга «*сервис > операция*» означает то, что сервис имеет одну или больше операций. *Дуга «операция > >вход*» означает то, что операция имеет больше одного входного сообщения. Определение Веб-сервиса заключается в определении каждого концепта соответствующей онтологии.

В соответствии с классическим определением сервиса, рассмотрим три типа участников нашего подхода: провайдеры, композиционные конструкторы (конструкторы) и пользователи.

Провайдеры определяются как объекты, которые специфицируют и предлагают Веб-сервисы для использования. Например, справочное агентство по кредитам, которое предоставляет сервис истории кредитора. Провайдер является ответственным за описание Веб-сервиса, назначая каждому понятию сервиса значение в онтологии.

Конструкторы определяются как объекты, ответственные за определение композитного сервиса (например, компания-посредник по продаже автомобилей). Формируются описание сервисов, на основе которого генерируется композитный сервис. Этот сервис определяется и объявляется в реестре сервисов так, чтобы другие сервисы могли его обнаружить и использовать.

Пользователи сервисов могут быть конечными пользователями (например, посредники по продаже автомобилей) или другие Веб-сервисы, которые вызывают необходимый Веб-сервис.

Для иллюстрации подхода рассмотрим приложение по продаже автомобилей, бизнес-схема которого рассматривается на сайте http://en.wikipedia.org/ wiki/ Lemon_ law. Допустим, что компания предоставляет автомобильному брокеру композитный сервис, который предлагает пакеты продажи автомобилей. Клиенты компании формируют запросы брокеру по продаже. Брокер привлекает другие сервисы для выполнения своей работы, например, сервис по продаже автомобилей (CB), сервис страхования (IN), сервис проверки проблем (LC), сервис финансирования (FI), сервис кредитной истории клиента (CH), сервис регистрации (RG). Приблизительно такая же бизнес-схема используется в каждом автосалоне.

Бизнес-схема процесса продажи автомобиля в нотации IDEF0 показана на рис. 1.

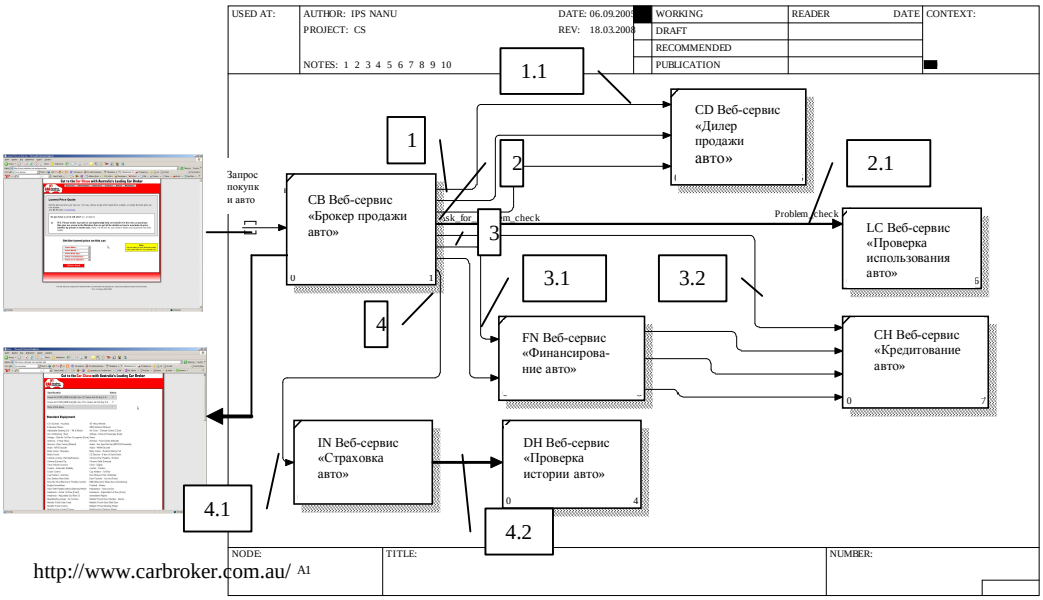

Рис. 1. Пример бизнес-схемы композитного сервиса продажи автомобилей

Типичный сценарий использования сервиса CB следующий: чтобы купить автомобиль определенной модели, клиент должен запустить на выполнение сервис CB, вызывая функцию sendMePriceQuote, для получения ценовой характеристики автомобиля (1). Для получения ценовой характеристики, сервис, очевидно, взаимодействовал бы с сервисом по продаже автомобилей через операцию priceQuote (1.1). Если бы клиент интересовался автомобилем, который уже был в использовании, он проверил бы его состояние и хронологию, вызывая операцию askForProblem Check (2). В свою очередь, эта операция обращается к операции problemCheck (2.1). Клиент должен выполнить оплату за выбранный автомобиль, вызывая операцию applyForFinancing (3). Перед утверждением плана финансирования сервис проверяет платежеспособность клиента, вызывая операцию payingHistory CH (3.1). Если решение по предоставлению кредита положительно, то CB вызывает операцию financingQuote, которая предлагает финансовые сервисы (3.2). Дальше, клиент пригласил бы страховую компанию, используя операцию insuranceQuote (4). Сервис CB, очевидно, вызывал бы операцию applyforInsurance, предложенную страховым сервисом (4.1). В результате, от страхового сервиса DH получил бы параметры страховки, путем выполнения операции drivingRecord (4.2). И наконец, клиент может зарегистрировать автомобиль, используя сервис carRegistry, в результате выполнения которого клиент получит номер государственной регистрации.

1.1. Типы сообщений. Функциональные возможности Веб-сервиса доступны пользователям через запуск операций. Запуск операций сервисов осуществляется посредством передачи сообщений. В операциях используют четыре типа сообщений (рис. 2):

исходное одностороннее сообщение (notification);

входное одностороннее сообщение (one-way);

требование ответа (solicit-response);

запрос-ответ (request-response).

Рассмотрим особенности перечисленных типов сообщений.

Операция посылает «исходное одностороннее сообщение (notification)» сервису, но не ожидает его в ответ. В случае односторонней операции «входное одностороннее сообщение (one-way)» сервис получает входное сообщение, но не посылает никакого сообщения в обратном направлении. Для типа сообщения «требования ответа (solicit-response)» сервис генерирует исходное сообщение для получения соответствующего входного сооб-

щения. В случае сообщения «запрос-ответ (request-response)» сервис получает входное сообщение, обрабатывает его и посылает соответствующее исходное сообщение.

Для определения соответствия параметров сообщений используются типы данных 1), 2), но они не представляют семантику этих параметров. Например, ценовой параметр может быть в долларах США или гривнах, цену возможно представить как полную или цену без НДС. Чтобы моделировать такие ограничения, используют связывание каждого из параметров с его бизнес-ролью. В качестве значений единиц параметров используют стандартные единицы измерения (длина, вес, код денег и так далее). Если параметр не имеет единицы (например, «имя»), тогда этот параметр будет иметь значение "none". Бизнес-роль определяет семантику соответствующего параметра. Значение семантики определяется в соответствии с предопределенной таксономией бизнес-ролей. Примером такой таксономии может быть классификатор товара и услуг.

Сообщение M определяется как кортеж (*P, T, U, R*), где:

P – множество имен параметров;

T:p > DataTypes – функция, назначающая тип данных к каждому параметру. DataTypes – множество типов данных XML;

U:P > единицы измерений – функция, определяющая единицы измерения, которые используются для каждого параметра. Единицы измерения определяются таксономией единиц измерения;

R:P > роли – функция, назначающая бизнес-роль для каждого параметра. Роли определяются таксономией бизнес-ролей.

1.2. Цель и категория. Семантика каждой операции сервиса описывается через цель и категорию. Цель содержит три атрибута: функция, синонимы и специализация.

Функция определяет бизнес-функциональность, обеспечивающаяся операцией сервиса. Примерами функций могут быть "запрос о ценах на автомобили", "заказ на кредитование". Чтобы определить функциональный признак бизнес-ролей параметров функций, используется широкий диапазон таксономий.

Цель операции OP_{ik} определяется кортежем (F_{ik}, SN_{ik}, SP_{ik}), где F_{ik} – бизнес-функции, определенные в пределах данной таксономии, SN_{ik} (синонимы) – множество альтернативных имен функции и SP_{ik} (специализация) – множество характеристик функции OP_{ik}.

Каждая цель предварительно определяется пространством имен в структуре XML, которая соответствует определенной таксономии. Например, операции цели может предшествовать пространство имен, указывающее на таксономию определенного бизнес-классификатора, например, RosettaNet или соответствующего отечественного классификатора товаров и услуг. Синонимы отвечают набору альтернативных имен функции для операции.

Категория операции определяется таким же образом, как и цель. Каждая категория содержит три признака: предметная область, синонимы и специализация. Предметная область представляет область интересов для текущей операции. Примером областей может быть сфера работы "автомобильных дилеров" или "страхования". Таксономии предметных областей определяются принятыми стандартами в бизнесе и промышленности, которые могут использоваться для определения предметной области. Синонимами области "автомобильные дилеры" являются "агенты по продаже автомобилей" или "автомобильные продавцы". Примером специализации, связанной со "страхованием", является {"автомобиль", "дом"}. Это значит, что соответствующая операция страхования обеспечивает страхование автомобиля и страхование дома или квартиры.

Категория операции OP_{ik} определена кортежем (PO_{ik}, SN_{ik}, SP_{ik}), где PO_{ik} – область интересов операции, определяющаяся в пределах выбранной таксономии, SN_{ik} (синонимы) – множество альтернативных областей и SP_{ik} (специализация) – множество характеристик области операции OP_{ik}.

1.3. Качество операций сервисов.

Несколько Веб-сервисов могут обеспечить "похожие" операции в соответствии с описанием их сообщений, целей и категорий. Важно определить качественные характеристики, которые помогут выбирать "лучшие" Веб-сервисы по определенным критериям. Определим три показателя качества для операций сервисов: стоимость, безопасность и секретность. Другие показатели, такие как время доступа, время ожидания ответа на сообщение, при необходимости могут рассматриваться дополнительно.

Показатель качества «стоимость» указывает на количество денежных единиц, которые необходимо потратить для выполнения операции. Секретность и безопасность особенно важны в применениях E-government, где граждане озабочены утечкой их личной информации. Показатель безопасности – булевая переменная, указывающая на то, прошел ли обмен сообщениями между сервером и клиентами надежно (например, используя методики кодировки). Показатель секретности содержит параметры, которые не должны быть доступными для внешних объектов (кроме поставщика сервисов). Если параметр сервиса не включается в набор секретных параметров, то он не подлежит контролю на ограничение секретности. Далее определим понятие качества операции сервиса.

Качество операции OP_{ik} определяется кортежем ($Fees_{ik}$, $Security_{ik}$, $Privacy_{ik}$). $Fees_{ik}$ – количество денежных единиц, которое необходимо потратить на выполнение OP_{ik}. $Security_{ik}$ – булева переменная, определяющая надежность обмена сообщениями при выполнении операции OP_{ik}. $Privacy_{ik}$ – набор входных и исходных параметров, которые являются не доступными для внешних объектов.

1.4. Определение операций Веб-сервисов. Веб-сервисы доступны через операции. Каждая операция идентифицируется именем и текстовым описанием, обобщающим функциональные особенности операции. Операция также определяет способ передачи сообщений, входные и/или выходные сообщения, цель и категорию. Далее представим определение операции сервиса.

Операция сервиса OP_{ik} определена кортежем ($Description_{ik}$, $Mode_{ik}$, In_{ik}, Out_{ik}, $Purpose_{ik}$, $Category_{ik}$, $Quality_{ik}$), где

$Description_{ik}$ – текстовое описание функциональности операции;

$Mode_{ik}$ – тип операции {“one_way”, ”notificatoin”, ”solisit-respons”, "respons-reply"};

In_{ik} и Out_{ik} – тип (входное или выходное) сообщение;

$Purpose_{ik}$ – описывает бизнес-функцию операции;

$Category_{ik}$ – описывает предметную область интересов операции;

$Quality_{ik}$ – определяет качественные характеристики операции.

Рассмотрим пример операции продажи автомобилей. Операция CB:: sendMePriceQuote определена кортежем (*Desc, Vode, B, P, C, Q*), где

Desc = "операция возвращает цену на выбранный автомобиль";

Vode = "solicit-response" – требование ответа.

B представляет типы данных параметров:

In = (*P*, *T*, *U*, *R*), где *P* = {*VIN*, *price*}; *T*(*VIN*) = "positiveInteger"; *T*(*price*) = = "float"; *U*(*VIN*) = "none"; *U*(*price*) = ="доллар США"; *R*(*price*) = "extendedPrice";

Out = (*P*, *T*), где *P* = {*make, model, year, mileage*}; *T*(*make*) = "string"; *T*(*model*)= "string"; *T*(*year*) = "gYear"; *T*(*mileage*) = "positiveInteger"; *U*(*mileage*) = ="км".

P – представляет цель операции и определяется: *P.Function* = "запрос о стоимости"; *P.Specialization* = {"информация для пользователя"}; *P.Synonyms* = ={"стоимость"};

C – категория операции, определенная следующим образом:

C.Domain = "автомобильные дилеры";

C.Synonyms = {"агенты по продаже автомобилей"};

C.Specialization = {"автомобили, которые были в использовании"};

Q – качество операции, определенное следующим образом: *Q.Fees* = 0*; Q.Security* = “false"; *Q.Privacy* = 0 (т. е., нет слишком важной информации).

Взаимодействия с сервисом выполняются согласно протоколу переговоров [7], которые определяются форматом сообщений и протоколом для сервисных операций. Взаимодействие сервисов основано на следующих стандартах: SOAP: (Simple Object Access Protocol), HTTP_Get/Post MIME (Multipurpose Internet Mail Extensions). Сервис одновременно может выполнять несколько переговоров. С каждым сервисом связываются цель и категория. Цель описывает функциональные возможности, которые предлагаются сервисными операциями. Каждый элемент цели сервиса ссылается на функцию определенной операции. Категория описывает предметную область сервиса. Она включает категории всех операций, которые обеспечиваются сервисом. Область, представляющая интерес сложного сервиса, может отличаться от областей, которые представляют интерес его отдельных операций. Например, композитный сервис продажи автомобилей связан с "автомобильными дилерами". Но он включает операции, связанные со "страхованием" (например, insuranceQuote) и "кредитованием" (например, financingQuote).

Формальное определение Веб-сервиса.

Веб-сервис WS_i определяется кортежем ($Description_i$, OP_i, $Bindings_i$, $Purpose_i$, $Category_i$), где

$Description_i$ – текстовое описание функциональных возможностей сервиса;

OP_i – множество операций, которое обеспечивается сервисом WS_i;

$Binding_i$ – набор протоколов переговоров, которые поддерживаются WS_i;

$Purpose_i$ = {$Purpose_{ik}$ (OP_{ik}) | OP_{ik} из OP_i } множество операций цели;

$Category_i$ = {$Category_{ik}$ (OP_{ik}) | OP_{ik} *из* OP_i } *AND* {$Category_i$ (WS_i)} – множество категорий операций WS_i .

В качестве примера сервис по продаже автомобилей (CD) (см. рис. 1) определен кортежем (*Desc, OP, B, P, C*), где

OP = {*priceQuote, testDrive, specialOffers*};

B = {"SOAP"};

P – цель сервиса, определена множеством {цель (priceQuote), цель (testDrive), цель (specialOf fers)};

C – категория сервиса, определена множеством {категория (priceQuote), категория (testDrive), категория (specialOffers)}, категория {category(CD)}.

Элементы категории *CD* определяются следующим образом:

category(CD).domain = “automobile dealers”;

category(CD).synonyms = {“car dealers”, “car sellers”};

category(CD).specialization = {“used cars”}.

2. Моделирование реализуемости композиции Веб-сервисов

Главная проблема композиции Веб-сервисов заключается в определении того, являются ли компоненты сервисов композитными. Например, было бы трудно или совсем невозможно вызывать операцию, если бы не было соответствия между параметрами, которые используются этой операцией (например, типы данных, число параметров), переданные сервису клиентом.

Определим два типа правил композиционности, использующие синтаксические и семантические свойства Веб-сервисов. Синтаксические правила сравнивают типы операций и протокол взаимодействия сервисов. Семантические правила сравнивают число параметров сообщений, бизнес-роли и единицы измерения. Операции семантической композиционности сравнивают семантику сервисных операций.

Композиционность по параметрам качества сравнивает качественные свойства Веб-сервисов. Целостность или непротиворечивость сервисов проверяет, даст ли выбранная комбинация Веб-сервисов ожидаемый результат.

Определение непротиворечивости сервисов осуществляется на уровне множества сервисов, которые вовлекаются в композицию, в отличие от других правил, проверяющие композиционность на уровне операций.

В нашей модели, сервисы вызывают функции внешних сервисов (например, автомобильный брокер), которые, в свою очередь, используют другие внешние сервисы для выполнения своих работ (например, страхование). Мы используем подход, определенный в стандартах WSMO [9], XLANG [19], WSFL [20] и BPEL4WS [21]. Веб-сервисы определяются в терминах WSDL, с расширением семантических возможностей этих средств. Допускается, что каждой операции на стороне сервера однозначно соответствует операция на стороне клиента и наоборот.

2.1. Определение композиционной реализуемости на основе взаимодействия сервисов. Для взаимодействия Веб-сервисов должен обеспечиваться режим, при котором операция в одном сервисе должна быть связана с односторонней операцией в сервисе партнера. Так же, операция «требование-ответа» сопровождается операцией «ответ на запрос» в сервисе партнера. Например, операция *CD::sendMePriceQuote* (требование ответа) приводит к *CD::priceQuote* (ответ на запрос), и *CD::receivespecialoffers* (односторонняя) *CD::specialOffers* (сообщение). Следующее правило определяет способ композиционности и проверяет, имеют ли две операции требуемый режим.

Две операции OP_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik}) и OP_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl}) – композиционные по типу, если обеспечиваются условия соответствия режимов: (I) M_{ik} = ="notification" и M_{jl} = "one-way"; или (II) M_{ik} = "one-way" и M_{jl} = "notification"; или (III) M_{ik} = "solicit-response" и M_{jl} = "request-response"; или (IV) M_{ik} = "request-response" и M_{jl} = "solicit-response".

Допустим, два Веб-сервиса общаются через операции, которые являются композиционными по типу. Поскольку эти Веб-сервисы могут поддерживать разные протоколы переговоров (например, SOAP, HTTP или MIME), важно обеспечить, чтобы они "поняли" друг друга по форматам сообщений и уровням протокола. По крайней мере, один из протоколов, который поддерживается Веб-сервисом, должен быть поддержан другим. Например, сервис ожидает сообщения по протоколу MIME и взаимодействует с другим сервисом, который формирует сообщение по протоколу HTTP.

Следующее правило, названное композиционным связыванием, проверяет, чтобы Веб-сервисы поддерживали по крайней мере один общий протокол связывания.

Композиционность сервисов WS_i = = $(D_i, O_i, B_i, P_i, C_i)$ и WS_j = (D_j, O_j, B_j, Pj, Cj) имеет место, если $B_i \cap Bj \neq 0$.

2.2. Композиционность по сообщениям. Взаимодействие Веб-сервисов осуществляется на основе метода обмена сообщениями («переговоров»). Сообщение состоит из одного или нескольких параметров и каждое из них имеет определенный тип данных. Важно проверить соответствие типов данных каждого из параметров сообщения посылающего сервиса на совместимость с типами данных параметров, которые получает его партнер. Мы рассматриваем два метода проверки совместимости типов данных: непосредственный и косвенный.

Два параметра непосредственно совместимы, если они имеют тот же тип данных. Параметр p косвенно совместим из p', если тип данных параметра p получен из типа данных параметра p'.

Например, параметр с типом *positiveInteger* или типом *short* косвенно совместим с целочисленным параметром. Следует отметить, что непосредственная и косвенная совместимость являются асимметричными.

Расширим понятие совместимости по типу данных таким образом: сообщение M совместимо с сообщением M' по типу данных, если каждый параметр M непосредственно или косвенно совместим с параметром M'.

Отметим, что не все параметры *M'* можно отобразить на параметры *M*. *Входное* сообщение может использовать только подмножество параметров, которые ссылаются на исходное сообщение операции.

Например, автомобильный провайдер не интересуется цветом автомобилей, которые рекламируются как специальные предложения. В этом случае, он определяет входное сообщение *CD::receive-SpecialOffers*, которое составляется из следующих параметров: “make”, “model”, “year”, “mileage”, “price”. Входное сообщение совместимо с сообщением *CD:: specialOffers*, хотя исходное сообщение содержит дополнительный параметр.

Допустим, параметр *p* совместим (непосредственно или косвенно) с параметром *p'*. Чтобы быть совместимыми, параметры *p* и *p'* должны иметь также совместимую семантику. Например, полная цена не должна сравниваться с ценой, которая включает налоги. Также цена в долларах США не должна сравниваться с ценой в гривнах. В результате сравниваем единицы измерения и бизнес-роли *p* и *p'*. Оба параметра должны иметь одни и те же единицы измерения и бизнес-роли.

Правила проверки композиционности по сообщениям сравнивают входные и исходные сообщения каждой пары операций. Идея заключается в проверке того, имеет ли каждая входная операция тип данных, который является совместимым с типом данных соответствующей исходной операции. Единицы измерения и бизнес-роль входа должны быть такими же, как единицы измерения и бизнес-роль соответствующего выхода.

Композиционность по сообщениям. Две операции $OP_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$ и $OP_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl})$ – композиционные по сообщениям если:

1. $\forall\ p \in In_{ik}\ \exists p' \in In_{jl,}$ где *p* − совместимо по типам данных из *p'*, $U(p) = U(p')$ и $R(p) = R(p')$.

2. $\forall\ p \in In_{jl}\ \exists p' \in In_{ik}$, где *p* является типами данных, совместимыми с типами данных из *p'*, $U(p) = U(p')$, и $R(p) = R(p')$.

2.3. Композиционность операционной семантики. Совместимость целей и категорий связанных операций обеспечивается композиционностью операционной семантики, например, функция Cd::sendmepricequote ("запрос о стоимости") отличается от функции CD::testDrive ("пробная поездка"). Было бы некорректно сравнивать эти операции, поскольку они предлагают разные бизнес-функции. Так же операция IN::applyForInsurance не совместима семантически с операцией CB::calculatePayment, поскольку эти операции имеют разные области интересов ("страхование" и "кредитование", соответственно). Чтобы определять совместимость между категориями операции, рассмотрим следующие две:

операции $OP_{IK} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$ и $OP_{JL} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, Cjl, Q_{jl})$ − совместимые если:

1. (C_{ik}.Domain = C_{jl}.Domain) or
(C_{ik}.Domain∈ C_{jl}.Synonyms) or
(C_{jl}.Domain∈ C_{ik}.Synonyms) or
(C_{ik}.Synonyms ∩ C_{jl}.Synonyms ≠ Ø).

2. C_{ik}.Specialization ⊆ C_{jl}.Specialization.

Первое условие проверяет параметры области интересов, которые должны быть подобными или синонимическими. Второе − гарантирует, что OP_{jl} обеспечивает все характеристики категории OP_{ik}. Совместимость между целями определена таким же образом, как между категориями. Семантическая композиционность основана на понятии совместимости по категориям и целям.

Операция OP_{ik} – семантически композиционна с операцией OP_{jl}, если цель операции OP_{ik} совместима с целью операции OP_{jl} и категория операции OP_{ik} совместима с категорией операции OP_{jl}.

Композиционность операционной семантики определим следующим образом. Мы говорим, что операция $OP_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$ – семантически композиционная с операцией $OP_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl})$, если P_{ik} совместимы с P_{jl} , C_{ik} и C_{jl}.

2.4. Качество композиционности. Для определения качества композиционности проверяют свойства качества взаимодействующих операций. Рассмотрим операцию OP_{ik}, привлекающая внешние сервисы для выполнения другой операции OP_{jl}. Проверка композиционности по стоимости подтверждает, что количество денежных единиц, которое тратится на выполнение операции OP_{ik}, по крайней мере, равняется стоимости операции OP_{jl}. Безопасность композиции гарантирует то, что если OP_{ik} использует механизмы безопасности (например, кодировка и строгое выполнение обязательств) для обмена сообщениями, то OP_{jl} использует их также. Авторизация композиции сравнивает особенности авторизации операций OP_{ik} и OP_{jl}. Персональные настройки авторизации OP_{ik} должны быть включены в категорию авторизации, подтвержденными OP_{jl}. Если провайдер OP_{ik} не хочет, чтобы параметр p был обнародован

$p \in Quality_{ik}.privacy$, то

$p \in Quality_{jl}.privacy$.

Качественная композиционность. Мы говорим, что $OP_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$ – качественно композиционная с $OP_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl})$, если:

1. $Q_{ik}.Fees \geq Q_{jl}.Fees$.
2. $(Q_{ik}.Security = \text{true}) \Rightarrow (Q_{jl}.Security = true)$.
3. $Q_{ik}.Privacy \subseteq Q_{jl}.Privacy$.

2.5. Согласование сервисов (machmaking). Следующий важный аспект композиции сервисов заключается в объединении множества сервисов определенным способом для получения новой функциональности. Например, сложно комбинировать сервисы продажи автомобилей с сервисами гостиниц. Однако, объединение сервисов авиалиний и гостиничных сервисов обеспечило бы композицию сервиса подготовки путешествия. Идея заключается в том, чтобы определить правило, названное согласованием композиции, суть которого заключается в проверке того, являются ли композитные сервисы потенциально согласованными. Под согласованием мы имеем в виду способ, по которому композитный сервис обеспечивает новую функциональность. Для этого вводим понятие шаблонов композиции. Шаблон представляет собой граф, построенный на основе использования отношения «следования» рис. 2. Как показано на рис. 3, Веб-сервис WS_i предшествует другому сервису WS_j, если операция WS_i вызывает операцию в WS_j.

Далее формально определим отношение «следования»:

для заданных сервисов $WS_i = (D_i, O_i, B_i, P_i, C_i)$ и $WS_j = (D_j, O_j, B_j, P_j, C_j)$, WS_i следует за WS_j, если $\exists\, OP_{ik} \in O_i$ и $\exists OP_{JL} \in O_j \mid$ (I) (M_{ik} = “*notification”* и M_{jl}= = “*one-way”*); или (II) (M_{ik} = “*solicit-response”* и M_{jl} = “*request-response”*).

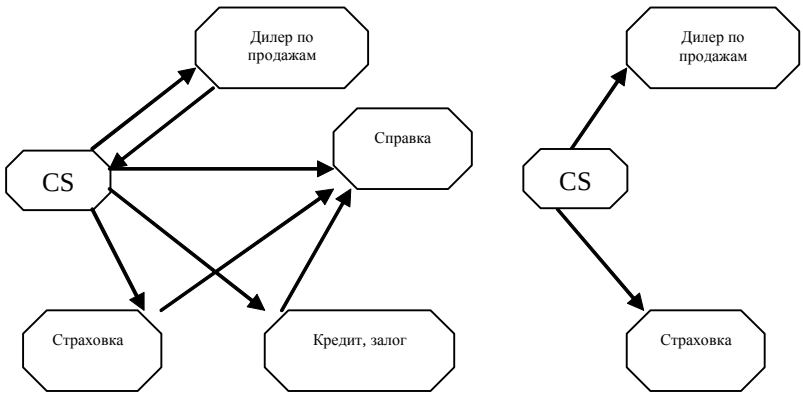

Рис. 2. Шаблон согласования композиции

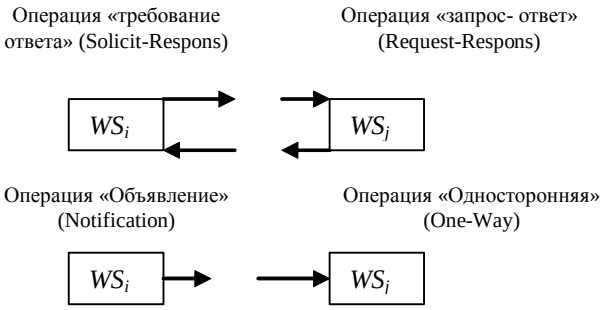

Рис. 3. Типы сообщений сервисов

Шаблон композиции связан с каждым композитным сервисом и представляет общую структуру сервиса. Отношение следования моделируется ориентированным графом (V, E), где V – множество имен категорий сервисов и E – множество дуг. Заданная вершина отвечает композитному сервису и имеет значение "*CS*". Дуга (V_i, V_j) указывает на то, что сервис с категорией V_i предшествует сервису с категорией V_j. На рис. 4 представлена схема композитного сервиса продажи автомобилей. Сервисы «Проверки чеков», «Кредитной истории», и «Истории вождения» представлены вершинами в графе, имеющие одну и ту же категорию «information».

Шаблоны композиции используются для того, чтобы сравнить значение новой функциональности в разных композициях.

Согласованность композиции. Композиция сервисов − согласованная, если граф шаблона нового сервиса является подграфом эталонного шаблона.

Эталонные шаблоны отличаются от реальных. Шаблоны процесса и сами процессы используются в качестве априорной основы для определения композитного сервиса, использующиеся для проверки согласованности композиционного сервиса, когда он сгенерирован. Важно отметить, что правило согласованности сервисов не используется для того, чтобы определить, является ли Веб-сервис композитным. Даже если композиция не является согласованной, конструкторы должны решить, желают ли они рассматривать полученный состав сервисов как "приемлемый".

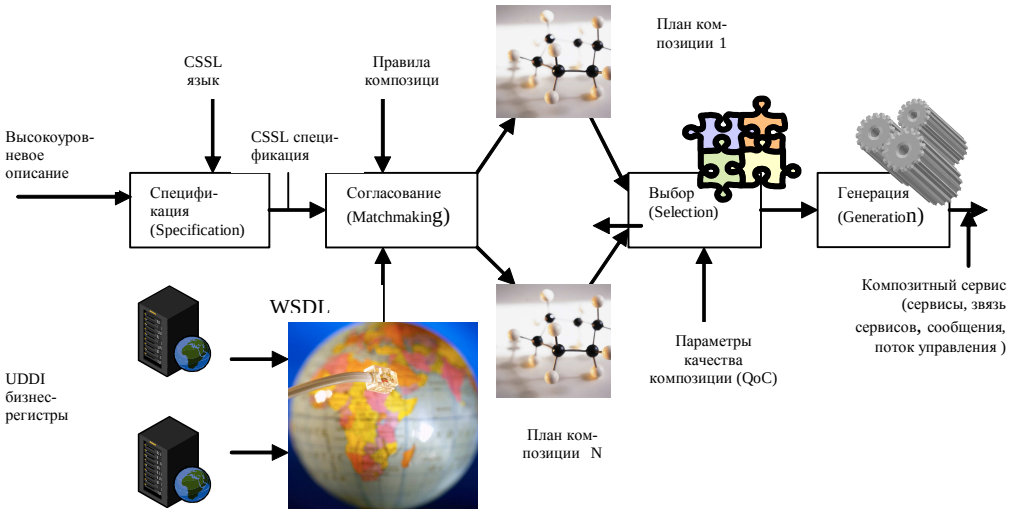

Рис. 4. Схема подхода к композиции сервисов

3. Подход к композиции Веб-сервисов

Подход к реализации композиции Веб-сервисов, основан на модели композиции и состоит из четырех концептуально отдельных фаз: спецификация, определение планов композиции, поиск и выбор сервисов и собственно композиция (см. рис. 4).

Фаза спецификации допускает высокоуровневое описание необходимой композиции, используя язык CSSL (Composite ServiceSpecification Language) [22]. Фаза определения композиции использует правила проверки композиционности для генерации планов композиции, которые отвечают спецификациям. Под планом композиции, подразумеваем список сервисов, определение их взаимодействия друг с другом с целью формирования композитного сервиса. Алгоритм отбора сервисов использует входные спецификации, которые формируются конструктором, и содержатся в репозитории (UDDI [8]). Сервисы специфицированы в нотации WSDL, которая расширена семантическими конструкциями.

Конструкторы на фазе выбора определяют сгенерированный план композитного сервиса на основе показателей качества композиции (например, стоимость). Используя отобранный план, генерируется детализированное описание композитного сервиса. Это описание содержит список внешних сервисов для выполнения работ, схему связей между операциями сервисов и поток управления сервисами. Поток управления формируется множеством операций, которые выполняются сервисами.

Фаза спецификации требует вмешательства конструктора для описания необходимой композиции. На фазе согласования сервисов генерируются планы композиции, основанные на спецификации, которая задана конструктором. Фаза выбора использует параметры ограничения для выбора лучшего плана. Такие параметры определяются конструктором в процессе спецификации. Фаза композиции обеспечивает описание композитного сервиса на заданном языке, например, WSFL [19].

3.1. Фаза спецификации. Для спецификации композитного сервиса используем язык CSSL (*Composite Service Specification Language*), созданный на основе XML. CSSL обеспечивает описание высокого уровня композитного сервиса. Конструкторы должны иметь общую идею или бизнес-модель сервиса. Они не обязаны знать технические детали нового сервиса, характеристики составных сервисов и то, как они взаимодействуют между собой. Есть несколько отличий между языком CSSL и существующими языками композиции сервисов. Язык CSSL определяет модель композиции на основе онтологии для обеспечения семантической поддержки возможностей Веб-сервисов. Большинство существующих языков не учитывают семантические возможности Веб-сервисов. CSSL расширяет язык WSDL таким образом, чтобы обеспечить: описание семантических особенностей Веб-сервисов; спецификацию потока управления между операциями композитного сервиса, что делает композицию сервиса такой же простой, как и определение простых сервисов. Принятый подход позволяет поддерживать рекурсивную композицию сервисов. Сложные сервисы можно рассматривать как сервисы WSDL и следовательно, использовать их как компоненты для новых композиций.

Язык CSSL также допускает спецификации потока управления операциями композитного сервиса. Операции могут выполняться последовательно или параллельно. Например, спецификация сервиса автомобильного брокера показывает, что этот сервис сначала проверяет хронологию оплаты клиентов перед запросом финансирования. Следует отметить, что CSSL также позволяет специфицировать условия потоков управления. Например, сервис автомобильного брокера применяет операцию финансирования, только после проверки, что клиент имеет позитивную хронологию оплат.

3.2. Установление согласованности сервисов. Следующим шагом после обеспечения спецификации CSSL является генерация соответствующих планов композиции с использованием алгоритма согласования. Поскольку число планов, которые будут сгенерированы, может быть большим, конструкторы имеют возможность отбора планов.

Общая схема алгоритма согласования заключается в отображении каждой операции OP_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik}) сложного сервиса WS_i на операции OP_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl},P_{jl}, C_{jl}, Q_{jl}) существующего сервиса WS_j. Алгоритм ищет Веб-сервисы WS_j = (D_j, O_j, B_j, P_j, C_j) так, чтобы P_{ik} и C_{ik} были совместимыми, по крайней мере, с одним элементом P_j и C_j, соответственно. Дальше алгоритм проверяет, связаны ли сервисы композиционно. Веб-сервисы группируются в содружества. Каждое содружество сервисов представляет кластер в соответствии с категорией. Сервисы, имеющие подобные категории и одну и ту же цель, принадлежат определенному содружеству. Использование сервисных содружеств ускоряет процесс выявления необходимого для композиции сервиса. В случае если C_{ik} не совместима с сервисами, принадлежащие содружеству WS_J, эти Веб-сервисы будут устранены из сервисного пространства для этой композиции.

Для каждой пары операций (OP_{ik}, OP_{jl}) проверяется режим композиционности, операция семантической композиционности и сообщения композиционности.

Множество сервисов, для которых устанавливается соответствие, содержит все операции, которые были "встроены" в соответствующую сложную сервисную операцию. Использование этого набора сервисов предшествует генерация планов композиции, которые могут быть выведены из планов, и предварительно были сгенерированы. Например, рассмотрим следующие два плана: *plan*1 = {(*opi*1, *opj*1), (*opi*2, *opj*2)} и *plan*2 = {(*opi*1, *opj*3), (*opi*2, *opj*4)}. Тогда *plan*3 = {(*opi*1, *opj*1), (*opi*2, *opj*4)} и *plan*4 = {(*opi*1, *opj*3), (*opi*2, *opj*2)} могут быть выведены из *plan*1 и *plan*2, поэтому нет необходимости генерировать их заново.

Для иллюстрации алгоритма определения согласованности без потери общности, рассмотрим трафик выполнения операций сервиса автомобильного брокера [18] receiveSpecialOffers.

Шаг 1. Формируется множество переменных *plan*.

Шаг 2. Выполняется поиск компонент сервиса (например, агент по продаже легковых автомобилей) поддерживающих SOAP протокол так, чтобы цель и категория *receiveSpecialOffers* были совместимыми с целью и категорией сервиса.

Шаг 3. Определяются операции сервиса *car_dealer*, которые являются совместимыми с *receiveSpecialOffers*. Алгоритм также проверяет, чтобы эти две операции не использовались вместе.

Шаг 4. Определяется качество композиционности операции *receiveSpecialOffers* и *specialOffers*, имеющие подобные функции ("*price-sales catalogue*") и области ("*automobile dealer*"). Таким образом, операционная семантика является композиционной и обе операции − качественно композиционные.

Шаг 5. Выполняется проверка операции на композиционность по сообщениям. Входы *receiveSpecialOffers* сравниваются с выходами *specialOffers*. Следовательно, эти две операции − композиционные по сообщениям.

Шаг 6. Формируются и сохраняются шаблоны операций. Поскольку обе операции синтаксически и семантически композиционные, информация сохраняется в соответствующем шаблоне. Шаги 1 − 6 итерационно формируют сложные сервисные операции.

Шаг 7. Для всех операций автомобильного брокера проверяется готовность сгенерированного плана.

3.3. Фаза выбора. На фазе выбора генерируются несколько планов композиции. Обеспечевается выбор релевантных планов, осуществляется сравнение значений параметров качества композиции: следования, релевантности и законченн-

ности. Также могут быть определены другие параметры, основанные на показателях стоимости и времени.

Для каждого плана определяется соответствующий шаблон композиции − *Cm*. Допустим, что *CT* – подграф сохраненного шаблона ST_i. Используем функцию *R*, определяющую сохраненные шаблоны; $R(ST_i)$, которые являются подграфами ST_i. Ранжирование *CT* относительно ST_i определяется следующим образом:

$$Ranking(CT, ST_i) = R(ST_i)/\ SUM^{n}_{к\ =\ 1,n}(R(ST_k)),$$

где *n* – число сохраненных шаблонов.

Композиционная релевантность. Обозначается как *CR*, дает приближенное значение композиции. Для этого сравниваются дуги шаблона композиции *CT* с дугами сохраненного шаблона ST_i. $CR(CT,ST_i)$ − отношение дуг *CT*, которые встречаются в ST_i.

Параметр полноты композиции. Определяет отношение операций композитного сервиса с сервисами, которые могут быть не "полностью" композитными. Действительно, если значение *CC* является относительно невысоким (например, 25%), то эти планы исключаются из разработки. В этом случае, конструкторы должны изменить спецификацию (например, изменить типы данных) так, чтобы достичь совместимости с другим сервисом. Для сложного сервиса WS_i:

$$CC(WS_i) = \frac{\left|Composable(O_i)\right|}{\left|O_i\right|}.$$

Планы композиции сортируются согласно их ранжированию. Конструкторы определяют пределы параметров завершенности. Планы будут возвращены конструкторам, если их релевантность и законченность больше, чем установленные ограничения. Параметры могут быть определены в пределах спецификаций CSSL таким образом, чтобы были отобраны и возвращены пользователям "лучшие" планы.

3.4. Фаза генерации описания композитного сервиса. Последняя фаза обеспечивает генерацию описания композитного сервиса. Это описание включает список внешних сервисов, отношения между сервисными операциями, которые определяются сообщениями, потоком управления и потоком данных между сервисами. Важные особенности этой фазы заключаются в настройке и расширении. Настройка относится к способности генерировать сложные сервисные описания на разных языках типа WSFL (Web Services Flow Language) [20], XLANG и BPEL4WS (Business Process Execution Language forWeb Services) [21].

Расширяемость относится к возможности включать дополнительные языки композиции [23]. Структура планов композиции − достаточно абстрактна, чтобы ее использовать на промежуточном уровне между спецификациями CSSL и наиболее распространенными языками композиции Веб-сервисов.

WSFL определяет два комплементарных описания для сложного сервиса: *flow model* и *global model*. *Модель потока* определяет последовательность выполнения операций между составными сервисами. *Глобальная модель* определяет, как взаимодействуют компоненты сервисов. *Модель потока* содержит множество действий. Каждое действие представляет отдельный шаг всех бизнес-целей. Для поиска информации по *businessEntity* каждого Веб-сервиса, сохраненного в регистре UDDI, используем операцию *find*() интерфейса запроса *UDDI*.

Глобальная модель включает множество элементов встроенных *связей*. Каждый элемент связи соответствует отображению в плане композиции. Например, операция *insuranceQuote* связана с операцией *applyForInsurance,* предлагающая сервисом страхования.

Выводы

В этой работе рассмотрен подход к композиции Веб-сервисов, представлена модель, алгоритм, который генерирует сложный сервис на основе спецификации высокого уровня. Модель обеспечивает

проверку композиционной реализованности сервиса и композиционные правила, которые сравнивают синтаксические и семантические параметры Веб-сервисов. Определяется модель "оптимизации" сложного сервиса, основанная на качестве композиционных параметров сервисов.

Развитие технологий семантических Веб-сервисов даст возможность создавать единственное унифицированное представление семантического Веб-сервиса в различных применениях, позволит точно находить необходимую информацию, упрощать корпоративную интеграцию и интеграцию сетевых приложений в распределенном Веб-окружении.

1. *Tim Berners-Lee, James Hendler and Ora Lassila*. The Semantic Web. http://www.semwebcentral.org
2. *Berners-Lee T.* Services and semantics: Web architecture. http://www.w3.org/2001/04/30-tbl
3. W3C Semantic Web. http://www.w3.org/2001/sw
4. *Bussler C., Fensel D., Maedche A.* A conceptual architecture for Semantic Web enabled Web services. SIGMOD Rec − 2002.
5. W3C Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl
6. *Jacek Kopecky, Tomas Vitvar, Carine Bournez, Joel Farrell* SAWSDL: Semantic Annotations for WSDFL and ML Schema. IEEE Internet Computing, Dec. − 2007. − P. 60−67.
7. W3C Simple Object Access Protocol (SOAP). http://www.w3.org/TR/soap
8. W3C Universal description, discovery, and integration (UDDI). http://www.uddi.org.
9. http://www.wsmo.org/
10. *Sheila A. McIlraith, Tran Cao Son, and Honglei Zen*. Semantic Web Services, Stanford University. http://www.ksl.Stanford.edu.
11. Web Services. http://www306.ibm.com/software/solutions/webservices/uddi
12. *Андон П., Дерецкий В.* Проблемы построения сервис-ориентированных прикладных информационных систем в Semantic Web среде на основе агентного подхода, // Проблемы программирования. − 2006. − № 2-3. − С. 493−502.
13. *Brahim Medjahed1, Athman Bouguettaya1, Ahmed K. Elmagarmid* Composing Web services on the Semantic Web // Published online: September 23, 2003. Springer-Verlag.
14. OWL Technical Committee. Web Ontology Language (OWL). http://www.w3.org/TR/2004 /WD-owlref
15. *Andon Ph., Deretsky V.* Control Oriented Ontology and Process Description for Cooperation Agents in Information Retrieval // Sixth International Scientific Conference „Electronic Computers and Informatics ECI'2004". Kosice – Herlany, Slovakia: September 22-24, 2004. − P. 14–18.
16. *McIlraith S.A., Son T.C., Zeng H.* Semantic Web services. IEEE Intell Sys. − 2001. − 16(2). − P. 46–53.
17. *Калиниченко Л.А.* Методология организации решения задач над множественными распределенными неоднородными источниками информации. ИПИ РАН. http://www.rfbr.ru /default.asp?doc_id=20786
18. http://www.carbroker.au/
19. *Florescu D., Grunhagen A., Kossmann D.* XL: an XML Programming Language for Web service specification and composition. In: Proceedings of the WWW, conference, Honolulu, May 2002. − P. 65–76.
20. IBM (2003) Web Services Flow Language (WSFL). http://xml.coverpages.org/wsfl.html
21. A Comparison of XPDL, BPML and BPEL4WS // http://xml.coverpages.org/Shapiro-XPDL.pdf
22. https://www.ifi.uzh.ch/fileadmin/site/teaching/Diplomarbeiten/Abgeschlossene_Diplomarbeiten/Jahrgang_2007/Allemann_Jonas.pdf.
23. *Wooldridge M.* On the Sources of Complexity in Agent Design - In Applied Artificial Intelligence, 14(7), 2000. − P. 623–644.

Получено 15.09.2008

Об авторе:

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

Место работы автора:

Институт программных систем
НАН Украины,
03187, Киев-187, проспект Академика Глушкова, 40.
Тел.: 38 044 526 4342.