УДК 681.3

В. Дерецкий

ПОДХОД К КОМПОЗИЦИИ ВЕБ-СЕРВИСОВ НА ОСНОВЕ СПЕЦИФИКАЦИИ ФУНКЦИОНАЛЬНОЙ СЕМАНТИКИ

Главная задача композиции Веб-сервисов состоит в идентификации готовых к использованию Веб-сервисов через процедуры поиска, согласования и оркестровки управления выбранными сервисами в соответствии с целевой спецификацией. В данной работе мы предлагаем и исследуем модель композиции Веб-сервисов через поиск сервисов в соответствии с требованиями функциональной спецификации целевого сервиса. Рассматривается общий алгоритм композиции с учетом или без запуска на выполнение составляющих сервиса.

Введение

Веб-сервисы представляют собой доступные в сети автономные программные системы, которые описываются функциональными и нефункциональными свойствами, могут быть размещены и доступны через стандартизированные интерфейсы [1]. Композиция Веб-сервисов обеспечивает механизм получения новой функциональности путем сбора многократно выполняемых сервисов для их совместного выполнения, и в дальнейшем рассматривать его как новый сервис. Автоматизированная композиция сервисов актуальна для многих областей, в том числе для электронной коммерции, электронной науки и др. Композиция Веб-сервисов была исследована в различных аспектах и во многих исследованиях, где были предложены различные модели композиции [1]. Эти модели основаны на таких подходах как AI-планирование [2, 3], композиция сервисов на основе моделей потоков работ WorkFlow [4−7], композиция на основе переговоров [8] и композиция на основе интерактивных диалоговых процедур [9−11].

В данной работе мы рассмотрим проблему композиции сервисов на основе спецификации целевого сервиса. Целевой сервис может рассматриваться как модель композиции сервисов, в которой каждая активность может быть рассмотрена как логическое представление реального сервиса. Такая модель композиции сервиса сформировалась на основе подхода к интерактивной композиции сервисов [9−11] и научных процессов потоков работ – WorkFlow [6, 12, 13]. В этой модели есть две технологические составляющие: поиск сервисов, для того чтобы идентифицировать существующие сервисы, которые могут использоваться, а оркестровка сервисов, чтобы собирать годные к использованию сервисы, для реализации целевого сервиса. В работах [10, 11] рассматриваются проблемы сообщества сервисов, которые основаны на том, что (1) составные сервисы для композиции уже известны и (2) каждый из них не содержит семантики. Работа [9] направлена на усовершенствование метода, путем использования OWL-S, для описания семантики сервиса. В настоящей работе, мы разрабатываем подобный подход с использованием семантики WSMO для сервисов, и исследуем проблему композиции в части поиска сервисов для композиции целевого сервиса.

Представим спецификацию целевого сервиса в виде ациклического графа активностей с потоком данных и условиями. Проблема композиции "схемы" целевого сервиса состоит в реализации схемы активностей существующих сервисов, которая осуществляется через запросы к рееструсервисов и семантического согласования (match-making) составных сервисов. Мы также рассматриваем проблему композиции по "образцу", для которой назначение сервиса для специфицированной цели вычисляется путем запуска на выполнение в соответствии с запросом.

Представление целевого сервиса в виде ациклического графа мотивировано процессами WorkFlow [6, 12, 13]. Процессы композиции для научных и бизнес областей в основном ациклические. Предположим, что существующие сервисы не имеют состояния, но имеют семантику OWL-S. Такое представление позволяет не только избегать сложных задач оркестровки рассмотренных в [9, 11], но также определяет проблему, применимую ко многим приложениям SOA [7].

Семантическое согласование (matchmaking) исследуется в контексте поиска сервисов [14], в части использования семантического согласования в целевом композитном сервисе, в случае, когда один сервисный запрос порождает запросы в других сервисах. Фактически, композиция сервисов и поиск сервисов связаны между собой. Поисковые запросы, исходящие от целевого сервиса и его семантических условий зависят от того, будут ли найдены нужные или квалифицированные сервисы. С другой стороны, найденные для активации сервисы, в свою очередь, могут усиливать возможность поиска других нужных сервисов для реализации других активностей. Интеграция поиска и композиции сервисов в одной структуре обеспечивают новый подход, как для проблемы поиска сервисов, так и для проблемы композиции сервисов.

В этой работе исследован алгоритм композиции сервисов, построенный на основе отождествления функциональных условий, показан общий алгоритм композиции с различными стратегиями.

1. Автоматизированная композиция сервисов

Существует три основных составляющих композиции сервисов: спецификация цели, поиск доступных сервисов, разработка алгоритма, который может "склеивать", составляющие части вместе [1].

Предлагаемый подход является естественным продолжением подхода использованного в диалоговых сервисах [10−11] и процессах WorkFlow в e-science [6, 12, 13] и состоит в конкретизации цели сервиса, представленного в виде множества состояний Веб-сервисов, используя модель потока данных. Цели сервиса определяются действиями или активностями, содержат потоки данных и взаимодействия с окружением. Реализация цели сервиса состоит в поиске релевантных сервисов, которые могут быть найдены через поиск сервисов в реестре. Есть два главных отличия между нашим подходом и работами [9−11]. Во-первых, мы предполагаем, что управляющий поток целевого сервиса ациклический и, во-вторых, сервисы в хранилище сервисов не имеют состояния и поэтому, проблема композиции состоит в поиске и назначении сервиса из реестра для каждой активности в целевом сервисе. В работе [15], сервисы выбираются для каждой активности в соответствии со спецификацией цели и основаны на нефункциональных свойствах сервиса. Мы рассматриваем алгоритмы композиции сервисов, которые основаны на функциональных свойствах, в частности на пред- и постусловиях активностей целевого сервиса.

Модель целевого сервиса, представленная в форме ациклического графа, принята по нескольким причинам. В форме WorkFlow представляются многие научные и бизнес-процессы. Композиция сервисов обычно имеет схему WorkFlow, включая активности, управляющие потоки и потоки данных. Каждый образец схемы WorkFlow должен быть настроен на определенный запрос сервиса. Композитный сервис должен искать и запускать отдельные сервисы из реестра, чтобы удовлетворить динамическим требованиям в соответствии с запросом, который поступает на вход сервиса. Например, бизнес-процесс приобретения продукта должен выбирать различные базы данных продукта и компании доставки согласно различным требованиям, поступающим от клиентов. Преимущество нашей модели состоит в спецификации основных функциональных требований, выделенных в логические условия, которые учитываются на уровне поиска сервисов. Ациклические WorkFlow целевого сервиса являются особо актуальными для e-science, так как

большинство из научных технологических процессов являются ациклическими.

На рисунке и в табл. 1 показан пример целевого сервиса "агент продажи автомобилей", бизнес-схема которого рассмотрена в [5]. Каждая активность имеет входные/выходные дуги, обозначенные как e_i в вершинах, и каждая входная дуга имеет входное условие, обозначенное как g_i. Активность, может иметь множество входных дуг и множество выходных дуг, указывая различные способы запуска активностей и различные пути следования. Для каждой входной и выходной пары дуг, определено логическое условие выхода, обозначенное как $P(e_i, e_j)$.

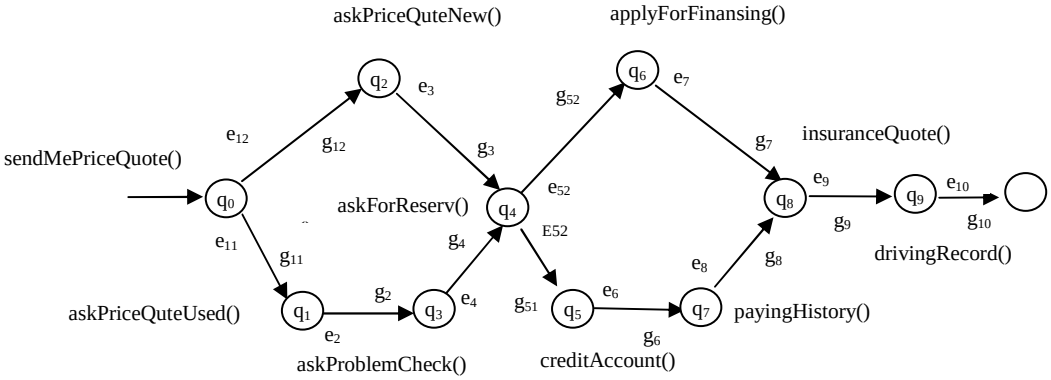


Рисунок. Ациклический граф целевого сервиса «агент продажи автомобилей»

Целевой сервис представлен ациклическим конечным графом с начальными и конечными вершинами. Каждая вершина за исключением начальной и конечной представляет собой активность. Каждая активность описывается своим входом/выходом и соответствующими логическими условиями. Каждый переход, отмеченный дугами, обозначает поток информации из одной активности к следующей. Кроме того, для каждого перехода определена защита в виде начального условия последующей активности. Для каждой активности в целевом сервисе, указано логическое условие выхода, для каждой пары входных и выходных дуг этой активности. Для данного образца входной дуги, активность может выполниться, если удовлетворено начальное условие; после выполнения активности, производится образец выходной дуги и фиксируется условие выхода для данной пары входной и выходной дуг.

В технической реализации, мы предполагаем, что набор функций окружения известен для всех сервисов (целевого и определенных в реестре), например, функции Now(), Date() – определяют текущие время и дату.

Определение 2.1. Целевой сервис представляется в виде потока данных (WorkFlow) $(Q, q_0, F, E, \Delta, P)$, где

Q – (конечное) множество активностей (состояний), $q_0 \in Q$ – начальная активность, $F \subseteq Q$ – набор конечных активностей,

E – (конечное) множество дуг,

Δ – множество переходов (q, g, e, q') такое, что

$q, q' \in Q$ являются состояниями,

$q \notin F, q' = q_0$,

$e \in E$ есть дуги,

g – начальное условие над e, и

граф $(Q, \{(q, q') \mid (q, g, e, q') \in \Delta\})$ является ациклическим.

P обозначает выходные условия активностей и является частичной моделью E x E такой, что, если есть последовательные переходы, помеченные как e_1 и e_2, то $P(e_1, e_2)$ – условие над выходными дугами и функциями окружения.

Таблица 1. Переменные, пред- и пост- условия модели целевого сервиса «агент продажи автомобилей»

Обозначение	Переменные и условия
e1 =	[modYear,vehMake,vehMod,transmitionType,extColor,intColor, engType,powerEngine, vehPriceQuote, vehDistance]
e2 =	[modYear,vehMake,vehMod,transmitionType,extColor,intColor, engType,powerEngine, vehPriceQuoteUsed,**idVehUsed**, vehDistance]
e3 =	[modYear,vehMake,vehMod,transmitionType,extColor,intColor, engType,powerEngine, vehPriceQuoteNew,idVehNew, vehDistance]
e4 =	[modYear,vehMake,vehMod,transmitionType,extColor,intColor, engType,powerEngine, vehPriceQuoteUsed,**idVehUsed**, vehDistance]
e5 =	[modYear,vehMake,vehMod,transmitionType,extColor,intColor, engType,powerEngine, vehDistance]
e6 =	[idUser,vehMod,vehPriceQuoteUsed]
e7 =	[idUser,vehMod,vehPriceQuoteUsed]
e8 =	[idUser,vehMod,vehPriceQuoteUsed]
e9 =	[modYear,vehMake,vehMod,transmitionType,extColor,intColor, engType,powerEngine, idUser,vehMod,vehPriceQuoteUsed, vehDistance]
g11 =	e1.modYear = {2000,2006} ∧ E1.vehMake = USED ∧ vehPrice = {6000,7000} ∧ vehDist < {200}
g12 =	e1.modYear = {2007,2009} ∧ E1.vehMake = NEW ∧ vehPrice = {10 000, 15 000} ∧ vehDist < {100}
g2 =	TRUE
g4 =	e1.modYear = {2000,2006} ∧ E1.vehMake = USED ∧ vehPrice = {6000,7000} ∧ vehDist < {200}
P(e1,e2) =	e2.modYear ∈ e1.modYear and ∧ e2.vehPrice ∈ e1.vehPriceQuote ∧ e2.vehDist ∈ e1.vehDist
P(e1,e3) =	e3.modYear ∈ e1.modYear and ∧ e3.vehprice ∈ e1.vehPriceQuote ∧ e3.vehDist ∈ e1.vehDist
P(e2,e4) =	e4.modYear ∈ e2.modYear and ∧ e4.vehprice ∈ e2.vehPriceQuote ∧ e4.vehDist ∈ e2.vehDist
P(e3,e4) =	e4.modYear ∈ e3.modYear and ∧ e4.vehprice ∈ e3.vehPriceQuote ∧ e4.vehDist ∈ e3.vehDist

Типичный сценарий использования сервиса следующий: чтобы купить автомобиль определенной модели, клиент должен запустить на выполнение сервис, вызывая активность (q_0) *sendMe PriceQuote*. Для получения ценовой характеристики, сервис, очевидно, взаимодействовал бы с сервисом по продаже автомобилей через активности *priceQuoteNew* (q_1), *priceQuoteUser* (q_2). Если бы клиент интересовался машиной, которая уже была в использовании, клиент проверил бы ее состояние и хронологию, вызывая операцию *askFor ProblemCheck* (q_3). В свою очередь, такая операция обращается к операции *problemCheck*. Клиент должен зарезервировать автомобиль для проплаты, вызывая операцию *applyForReserv* (q_4). Перед утверждением плана финансирования, сервис проверит платежеспособность клиента, вызывая операцию *paying History*. Если решение по предоставлению кредита положительно, то вызывается операция *financingQuote*, которая предлагает финансовые сервисы. Дальше, клиент пригласил бы страховую компанию, используя операцию insuranceQuote, которая, в свою очередь, вызывает операцию *applyforInsurance* предложенную страховым сервисом. В результате, от страхового сервиса получил бы параметры страховки путем выполнения операции *drivingRecord*.

На примере (рисунок) сервис "агент продажи автомобилей" требует атрибуты в соответствии с входными дугами, такие как модель автомобиля (vehMod), год

выпуска (vehYear). Целевой сервис гарантирует резервирование автомобиля данной модели для оплаты и данной целевой квоты, которое выражается как vehPrice $\in$ [6000, 7000] во входном условии $g_{1,1}$ и $g_{1,2.}$ Агент страхования вначале проверяет кредитную историю, если она существует и положительна, то используются льготные условия кредита; в противном случае, используются общие условия кредита. В примере рассмотрим только активность резервирования автомобиля для проплаты.

1.1. Модель Веб-сервиса

Рассмотрим некоторые ключевые понятия, используемые в задачах моделирования сервисов, представим модель сервиса, которую будем использовать для решения проблемы композиции.

Семантическое расширение Веб-сервисов обеспечивает увеличение возможностей, прежде всего поиска сервисов и автоматизации композиции. Средства описания семантики сервисов такие: OWL-S [3], WSDL-S [16], SWSO [17], WSMO [18] являются лидирующими в данном направлении. Хотя текущий UDDI явно не поддерживает семантику Веб-сервисов, подходы к его семантическому расширению рассматриваются в [16, 19].

Предлагаемая модель Веб-сервиса объединяет конструкции, для описания функциональной семантики сервисов. Веб-сервисом является активность с входными и выходными данными, пред- и пост-условиями. Активность может быть определена как действие в WSDL-S [16], или атомарной активностью в OWL-S [20]. Пред- и пост- условия определяются над входными и выходными данными, и взаимодействиями с "внешним миром".

Определение 2.2. Сервис представляет собой кортеж (e^i, e^o, P, Q),

где e^i, e^o – входные и выходные дуги; P, Q – соответственно пред- и пост-условия.

Сервис может быть запущен, если пред- условие P принимает значение «истина». Он использует образец дуги e^i, и производит образец дуги e^o. В результате выполнения сервиса так же должно удовлетворяться пост- условие Q.

Входные и выходные данные сервиса определяются дугами (envelop). Дуга представляет собой набор (кортеж) имен атрибутов со связанными типами; образец дуги – значение атрибутов соответствующих типов. Окружение, в котором выполняются сервисы, моделируется как набор функций окружения. Сервис взаимодействует с окружением, запуская эти функции. Например, функция *Now()* – функция окружения, через которую сервис может получить значение текущего времени.

Пред- условие сервиса – условие, которое должно удовлетворяться перед тем как сервис запускается. Входное условие определяется над входными атрибутами и функциями окружения. Пост- условие определяется над входными и выходными атрибутами, функциями окружения, описывающие взаимоотношение между входными и выходными данными и изменение окружения после того как сервис будет выполнен.

Определение 2.3. Условием поиска сервиса в реестре является кортеж (I, O, P, Q), где I – набор входных атрибутов; O – набор выходных атрибутов; P, Q – входные и выходные условия. Сервис S = = (e^i, e^o, Ps, Qs) удовлетворяет условиям поиска (I, O, P, Q), если выполняются следующие условия: (1) $e^i \subseteq$ I; (2) O $\subseteq$ e^o; (3) $P \rightarrow Ps$; (4) P и $Qs \rightarrow Q$.

Реестр сервисов представляет собой конечное множество сервисов с функцией поиска. Поиск сервисов – процесс поиска в реестре, в результате выполнения которого получается набор сервисов, которые удовлетворяют условиям поиска.

1.2. Задачи композиции сервисов

Проблема композиции сервисов широко представлена в литературе. Предложены различные модели композиции, например, Латинская модель [10, 11], модель переговоров [8], модели WorkFlow [4, 6, 12], модели состояния [15], AI-планирование [2, 3], модель Коломбо [9] и другие.

Наша модель композиции сервисов отличается от них средствами семантического описания целевого сервиса, семантической разметкой каждой активности целевого сервиса и ее семантическим описанием логических условий в реестре сервисов.

Входом для композиции сервисов являются целевой сервис и реестр сервисов. Композиция сервисов осуществляется для каждого запроса целевого сервиса. Для достижения цели целевого сервиса осуществляется выполнение запроса. Задача композиции состоит в поиске релевантных сервисов, назначаемых для активностей в целевом сервисе. Определим ключевые понятия алгоритма композиции.

Определение 2.4. Пусть $G = (Q, q_0, F, E, \Delta, P)$ является целевым сервисом. Реализация целевого сервиса G над реестром R – схема от $Q - (\{q_0\} \cup F)$ для R, т. е., для каждой активности, кроме начальных и конечных состояний осуществляется поиск и назначается на выполнение сервис из реестра R. Реализация L – выполнима, если для каждой активности $q \in Q - (\{q_0\} \cup F)$, определяется следующее равенство: $L(q) = (e_i, e_o, Ps, Qs)$.

Предположим, что $e^i_1,...,e^i_n$, $1 \le j \le n$ – входные дуги, а $e^o_1,...,e^o_m$, $1 \le k \le m$ – выходные дуги для q, g_j – начальные условия для e^i_j, и $P(e^i_j, e^o_k)$ $(1 \le j \le n, 1 \le k \le m)$ – выходные условия , тогда

$e^i \subseteq \cap_{1 \le j \le n} e^i_j$, $e^o \supseteq U_{j,k}(e^o_j - e^i_k)$,

$V_{1 \le j \le n}\, g_j \rightarrow P$, $g_j \wedge Q \rightarrow \wedge_{j,k} P(e^i_j, e^o_k)$.

В определении 2.4, реализация целевого сервиса выполняется, если для каждой активности входные дуги и предусловия назначенные сервису должны гарантироваться целевым сервисом, к тому же, выходные дуги и выходные условия целевого сервиса должны покрывать назначенный сервис.

Пусть задана активность q_i, ее входные и выходные условия. Сервис-кандидат может быть найден, если выполняются условия $(\cap_j e^i_j, \cup_{j,k}(e^o_j - e^i_k), \vee_j g_j, \wedge_{j,k} P(e^i_j, e^o_k))$ хотя бы в одном из зарегистрированных сервисов в R.

Сформулируем следующие задачи композиции сервисов:

– пусть задан целевой сервис G и реестр сервисов R, необходимо найти выполнимую реализацию сервиса G над R;

– пусть задан целевой сервис G, реестр сервисов R и среда выполнения целевого сервиса V, необходимо найти выполнимую реализацию сервиса G над R для среды выполнения V.

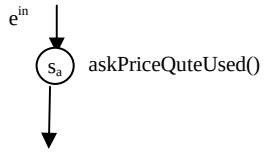
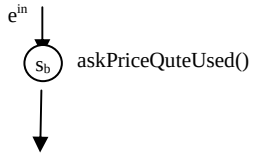
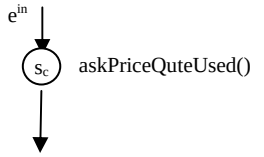
2. Стратегии композиции

Для обеспечения выполнимой реализации целевого сервиса рассмотрим методы, которые могут преобразовать начальные условия активностей в целевом сервисе, тем самым осуществлять более полный и точный поиск необходимых сервисов.

Для получения выполнимой реализации используется следующий подход: для каждой активности q в предоставленном целевом сервисе, мы формируем условия поиска $(\cap_j e^i_j, U_{j,k}(e^o_j - e^i_k), V_j g_j, \wedge_{j,k} P(e^i_j, e^o_k))$ в реестре сервисов. Отметим, что поисковые критерии обеспечивают последовательность действий. Главная проблема состоит в том, что при поиске сервисов в реестре есть возможность пропустить сервисы-кандидаты для данной активности и, поэтому выполнимая реализация не может быть найдена, хотя она существует.

Для иллюстрации этой проблемы мы рассмотрим целевой сервис (рисунок) "агент продажи автомобилей". В табл. 2 приведены три доступных в реестре сервиса, которые могут быть потенциальными кандидатами для активности резервирования автомобиля для последующей оплаты. Сервис (a) в табл. 2 обеспечивает резервирование автомобиля для только в ценовом диапазоне [3000, 5000]; сервис (b) обеспечивает резервирование автомобиля только в городе расстояние к которому превышает 600 миль, доставка которого может быть слишком дорогой; сервис (c) обеспечивает резервирование автомобиля только в ценовом диапазоне [8000, 12000].

Таблица 2. Сервисы-кандидаты резервирования для оплаты использованных автомобилей

№	Сервисы-кандидаты	Параметры и условия сервисов
a)		e^{in} (modYear,vehMake,vehMod,transmitionType, extColor, intColor,engType,powerEngine, vehPrice, vehDistance) P = modYear ∈ {2005} ∧ vehMake ∈ USED ∧ vehPrice ∈ {3000,5000} ∧ vehDistance ∈ {100,400}
b)		e^{in} (modYear,vehMake,vehMod,transmitionType, extColor, intColor,engType,powerEngine, vehPrice, vehDistance) P = modYear < {2006} ∧ vehMake ∈ USED ∧ vehPrice > {6000} ∧ vehDistance ∈ {600,1000}
c)		e^{in} (modYear,vehMake,vehMod,transmitionType, extColor, intColor,engType,powerEngine, vehPrice, vehDistance) P = modYear ∈ {2005,2007} ∧ vehMake ∈ USED ∧ vehPrice ∈ {8000,12000} ∧ vehDistance ∈ {50,100}

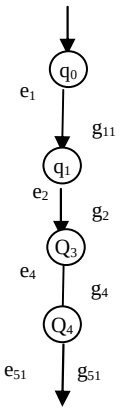
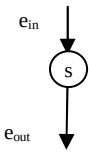
Рассматривая схему активностей целевого сервиса, представленных на рисунке и доступные сервисы в табл. 2, подход будет не в состоянии идентифицировать сервисы для активности резервирования использованного автомобиля, потому что начальное условие g_4 в целевом сервисе не соответствует условиям любого из трех сервисов. Это следует из факта, что g_3 и g_4 требует, чтобы сервис обеспечивал ценовой интервал [6000, 7000] в ближайшем ($\leq$ 200 миль) городе.

После проверки начальных условий g_{11} в целевом сервисе "агент продажи автомобилей", видим, что ценовой интервал фактически ограничивается интервалом между [5000, 8000]. Комбинируя с выходными условиями $P(e_4, e_{51})$, можно вывести, что ценовой интервал находится между [5000, 8000]. Как показано в табл. 3, левая часть показывает оригинальный целевой сервис, и начальное условие g_4 может быть «сжато» в условии g_4^1, включая входы/выходы предыдущих активностей. Если g_4^1 используется для поиска сервисов, то сервис (a) в табл. 2 становится сервисом-кандидатом.

Начальные условия активности могут быть сжаты с помощью использования пост- условий выбранного сервиса для предыдущей активности. В табл. 3 сервис (b), предназначен для активности резервирования автомобиля, существуют выходные условия активности, которые можно заменить более ограниченными пост- условия. В данном случае, ценовой интервал может быть ограничен [6000, 7000], тогда сервис (a) и (c) в табл. 2 становятся сервисами-кандидатами.

Композиция сервисов имеет место после того как осуществлен запуск целевого сервиса. Входные условия, поступающие от запроса при запуске сервиса, могут передаваться через целевой сервис путем преобразования входных условий. В табл. 3 (c), ограничение e_1 распространяется на активность резервирования автомобиля и начальное условие «сжимается» от g_4 к g_4^1. В данном случае, оба сервиса (a) и (c) в табл. 2 соответствуют сервису-кандидату.

Таблица 3. Схема «уплотнения» целевого сервиса

Исходный целевой сервис	Параметры и условия целевого сервиса	Выбранный сервис для q_1	Параметры и условия преобразованного целевого сервиса
	e_1 (…modYear, vehMake,…, vehPrice, vehDistance…) g_{11} = e1.modYear $\subseteq$ {2005} $\wedge$ e1.vehMake $\subseteq$ USED $\wedge$ $\wedge$ vehPrice $\subseteq$ {3000,6000} P(e1,e2) = e2.modYear $\in$ e1.modYear and $\wedge$ e2.vehPrice $\in$ e1.vehPriceQuote $\wedge$ e2.vehDist $\in$ e1.vehDist E_{51} (…modYear,vehMake,…, vehPrice, vehDistance…) G51 = e1.modYear $\subseteq$ {2005} $\wedge$ $\wedge$ E1.vehMake = USED $\wedge$ vehPrice $\subseteq$ {3000,5000} P(e4,e2) = e4.modYear $\in$ e2.modYear and $\wedge$ e4.vehPrice $\in$ e2.vehPriceQuote $\wedge$ $\wedge$ e4.vehDist $\in$ e2.vehDist		e^{in}(…vehYear,vehMake,…, vehPrice, vehDist…) P=… e^{out} =(…vehPrice, vehDist…) Q = … vehPrice $\subseteq$ {5000,8000} $\wedge$ vehDist $\subseteq$ {50,150} $\wedge$ … e_2 (…modYear,vehMake,…, vehPrice…) g2 = e1.modYear $\subseteq$ {2005} $\wedge$ e1.vehMake $\subseteq$ USED $\wedge$ vehPrice $\subseteq$ {3000,5000} $\wedge$ vehDist $\subseteq$ {50,150} $\wedge$ … e_4 (…modYear,vehMake,…, vehPrice…) g4 = e1.modYear $\subseteq$ {2005,2007} $\wedge$ E1.vehMake $\subseteq$ USED $\wedge$ vehPrice $\subseteq$ {5000,8000} $\wedge$ vehDist $\subseteq$ {50,150} …

В результате, полученный запрос при запуске целевого сервиса, может запустить выбранный сервис таким образом, что атрибуты выходных дуг были связаны с конкретными значениями. Эти фиксированные значения условий могут быть добавлены к выходным условиям активностей. В табл. 3 запускается назначенный сервис-кандидат и атрибут *veh PriceQute* в выходной дуге e_2 связан со значением [5000, 8000], тогда любой сервис резервирования автомобиля, который обеспечивает сервисы в полученном ценовом интервале, становится кандидатом для активности резервирования автомобиля.

Выполняя преобразование начальных условий, можно осуществить поиск релевантных сервисов для соответствующих активностей. Мы представляем две стратегии преобразования условий целевого сервиса.

Первая стратегия направлена на решение проблемы создания схемы сервисов. Она основана на уплотнении входных условий каждой активности в целевом сервисе, используя условия предыдущих активностей и пост- условия выбранных сервисов. Начальные условия и выходные условия предыдущих активностей уплотняют начальные условия следующих в пути активностей. Как только для активности назначен сервис из реестра, выходные условия заменяются пост- условиями выбранного сервиса.

Вторая стратегия направлена на решение проблемы композиции образца. Ограничения при запуске целевого сервиса, а также условия от предыдущих активностей в целевом сервисе. Как только

активность была назначена сервису из реестра, выходные условия не только заменяются постусловиями выбранного сервиса, но так же запускается выбранный сервис. Аргументы на выходных дугах активности связывают с конкретными значениями, полученными в результате выполнения сервиса. Полученные условия добавляются к выходным условиям активности.

2.1. Алгоритм композиции сервисов

Алгоритм композиции сервисов представлен следующим образом:

входом для алгоритма является целевой сервис и множество ограничений для запуска *GS*;

выходом является реализация целевого сервиса *L*.

1. Присоединить ограничения *C* к целевому сервису *GS* как условия запуска g_0 $L = 0$.

2. $Q_a = \emptyset$, где Q_a − очередь активностей.

3. for each $q' \in succeed(q_o)$ $append(Q_a, q')$.

4. while (!$empty(Q_a)$) $q = get(Q_a)$;

GenerateTightenedCondition(*q*) преобразование начальных условий для каждой активности;

Q_s = set of services discovered for *q*;

if $Q_s = 0$, return FAIL.

5. nondeterministically select service *S* from Q_s.

6. add $q \rightarrow S$ to *L*.

7. for each $q' \in succeed(q)$

if $q' \notin Q_a$, append(Q_a, q'').

8. *UpdateCondition*(*q*) обновление условий.

9. endwhile.

В предложенной модели композиции, методы уплотнения интегрируются в композицию сервисов. Мы опишем алгоритм композиции сервисов с преобразованием условий, как вышепоказано, и рассмотрим свойства алгоритма композиции. Ограничения условий, полученные при запуске целевого сервиса, прилагаются к целевому сервису. Так как целевой сервис является ациклическим, алгоритм проходит через активности в целевом сервисе с начала до конца. Для каждой активности q_i, осуществляется уплотнение начальных условий q_i, используя условия из предыдущих активностей, и первичные начальные условия заменяются уплотненными. Для преобразования используется процедура *GenerateTightenedCondition*(q_i). На следующем шаге, алгоритм выполняет поиск сервиса в реестре для активности q_i. Сервис выбирается из множества найденных сервисов для заданной активности. В результате, выходные условия активности обновляются в процедуре *Update Condition*(q_i).

Стратегии преобразования условий используют различные активности на шаге *UpdateCondition*(). Первая стратегия работает по схеме композиции сервисов и использует пост- условия выбранных сервисов для замены существующих условий активности. Вторая стратегия работает на композиции образца сервисов, т. е. в результате запуска целевого сервиса, запускается выбранный сервис для активности и условия ограничиваются результатом выполнения сервиса, который добавляется к выходным условиям.

Процедура *GenerateTightened Condition*() производит преобразование начальных условий для каждой активности, используя начальные условия и выходные условия предыдущих активностей.

Заключение

Автоматизированная композиция Веб-сервисов − практическая проблема. Через формулировку проблемы композиции мы рассматриваем много связанных вопросов для дальнейшего изучения. Одно из направлений исследования – развитие алгоритмов полноты и эффективности композиции, которые могут использовать семантические свойства специфицированных предметных областей данных. На общем уровне важным является вопрос, как процессы поиска сервисов могут совместно использовать процессы запуска сервисов.

1. *Hull R., Su J.* Tools for composite web services: A short overview // SIGMOD Record. −2005. −N 34(2). − P. 86–95.
2. *Narayanan S., McIlraith S.* Simulation, verification and automated composition of web services. In Proc. Int. World Wide Web Conf. (WWW). 2002.
3. *Narayanan S., McIlraith S.* Analysis and simulation of web services, // Computer Networkds. − 2003. −N 42. − P. 675–693.
4. *Aalst W.* On the automatic generation of workflow processes based on product structures // Computer in Industry.−1999.− N 39.− P. 97–111.
5. *Андон П., Дерецкий В.* Проблемы построения сервис-ориентированных прикладных информационных систем в Semantic web среде на основе агентного подхода // Проблемы программирования. −2006. − N 2-3. − С. 493–502.
6. *Tsalgatidou A., Athanasopoulos G., Pantazoglou M., et al*. Developing scientific workflows from heterogeneous services // ACM Sigmod Record. − 2006. − N 2. − P. 22–28.
7. *Sirin E., Parsia B., Hendler J.* Template-based composition of semantic web services. In AI symposium on agents and the semantic web.− 2005.
8. *Fu X., Bultan T, Su J.*. Conversation protocols: A formalism for specification and verification of reactive electronic services, // Theoretical Computer Science. − 2004. − N 328(1-2).− P. 19–37.
9. *Berardi D., Calvanese D., De Giacomo G. et al.* Automatic composition of transition-based semanticweb services with messaging // In Proc. 31st Int. Conf.on Very Large Data Bases (VLDB).− 2005.− P. 613–624.
10. *Berardi D., CalvaneseD., De Giacomo G., et al.* Automatic composition of e-services that export their behavior // In Proc. 1st Int. Conf. on Service OrientedComputing (ICSOC).− 2003.− P. 43–58.
11. *Gerede C., Hul lR., Ibarra O., J. Su.* Automated composition of e-services: Lookaheads.In Proc. 2nd Int. Conf. onService-Oriented Computing (ICSOC). − 2004.
12. *Ludaescher B., Altintas I., Berkley C., et al*. Scientific workflow management and the kepler system // J. of Concurrency and Computation: Practice & Experience. Special Issue on Scientific Workflows.
13. *Rao J., Su X.* A survey of automated web service composition methods // In Proc. of the 1st Int.Workshop on Semantic Web Services and Web Process Composition.− 2004.
14. *Elgedawy I., Tari Z., Winikoff M.* Exact functional context matching for web services, // In Proc. Int. Conf. on ServiceOriented Computing (ICSOC).− 2004. − P. 143–152.
15. *Zeng L., Benatallah B., Ngu A., et al*. Qos-aware middleware for web services composition // IEEE Transactions on Software Engineering. − 2004. − N 30(5). − P. 311–327.
16. W3C. Web services semantics – WSDL-S 1.0. 2005.Nov. http://www.w3.org/Submission/ WSDL-S/.
17. Web Service Modeling Ontology. http: //www.wsmo.org/.
18. *Battle S. et al*. Semantic Web Services Ontology (SWSO) Version 1.0. http://www. daml.org/services/swsf/1.0/swso/.2005.
19. *Luo J., Montros B., Kim A. et al.* OWL-S support to the existing UDDI infrastructure // In Proc. 4th IEEE Int. Con. on Web Services (ICWS). − 2006.
20. *Andon Ph., Deretsky V.* Control Oriented Ontology and Process Description for Cooperation Agents in Information Retrieval // Sixth International Scientific Conf. „Electronic Computers and Informatics ECI'2004". X Kosice – Herlany, Slovakia, September 22–24. − 2004. − P. 14–18.

Получено 18.03.2009

Об авторе:

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

Место роботи автора:

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