УДК 004.75:004.423.4

Д.Ю. Кашалкин, В.А. Курчидис

СЕМАНТИЧЕСКАЯ КОМПОЗИЦИЯ СЕРВИСОВ

Рассматриваются основные понятия и архитектурные элементы семантической сервис-ориентированной архитектуры (ССОА), используемые для решения задачи интеграции компонентов в распределенных информационных системах на семантическом уровне. Проведено формальное описание архитектурных элементов, которое положено в основу предлагаемого метода решения задачи семантической композиции сервисов. Метод использует семантический реестр ССОА, функциональная модель которого представлена в виде сети Петри. Решение основывается на обнаружении и селекции набора сервисов, семантически соответствующего формализованному запросу пользователя.

Введение. Одним из перспективных направлений развития ИТ отрасли является разработка и исследование принципов, методов и средств построения адаптивных распределенных информационных систем. Под адаптивностью понимается такое свойство информационной системы (ИС), которое при изменении условий окружающей среды позволяет динамично переконфигурировать элементы системы для решения определенных задач [1]. Динамическая реконфигурации элементов системы во многом зависит от степени решения проблемы интеграции распределенных разнородных компонентов ИС.

Сервис-ориентированная архитектура (СОА) [2,3] представляет собой один из подходов к интеграции компонентов в распределенных информационных системах. В основе архитектуры лежит идея представления бизнес-логики приложения в виде отдельной (от логики представления) службы. Развитие сервис-ориентированного подхода рассмотрено в работе [4], в которой предлагается использование формализованной семантики при описании основных архитектурных компонентов (сервисов) и среды их взаимодействия, что приводит к семантической сервис-ориентированной архитектуре (ССОА). Задача интеграции в ССОА сводится к реализации процессов поиска, выбора и композиции компонентов информационных систем. Базовые понятия об этих типах процессов и их реализации представлены в работе [4].

Семантическая композиция является наиболее сложнореализуемым процессом в ССОА и заключает в себе как обнаружение, так и селекцию необходимых компонентов. Фактически, решение задачи композиции ведет к созданию сложных (зависящих от выполнения других) компонентов, способных выполнять некоторую новую семантическую функциональность в рамках данной архитектуры. Для реализации процесса композиции компонентов ССОА в работе предлагается метод, который позволяет обнаружить и выбрать набор сервисов, совместное выполнение которых обеспечивает семантическое соответствие формализованному запросу пользователя.

Формализованное представление элементов ССОА, необходимых для композиции сервисов. ССОА базируется на следующих основных элементах [4]:

- компоненты: агенты пользователя, сервисы, посредники;
- коннекторы: протоколы передачи сообщений, менеджер сервисов (например, реестр), онтологии (сервиса, пользователя, доменные);
- данные: сообщения.

В семантической сервис-ориентированной архитектуре могут быть организованы следующие типы процессов: опубликование, обнаружение, селекция, композиция, вызов [4]. Опубликование и вызов представляют собой процессы, которые не требуют использования специальных алгоритмов. Обнаружение сервиса представляет собой поиск сервиса на базе определения семантического соответствия между описанием запроса и описанием опубликованного сервиса. Селекция сервиса выполняется в случае, если критерию запроса удовлетворяет более чем один сервис.

Композиция представляет собой более сложный процесс по сравнению с обнаружением и селекцией и осуществляется в случае, если для выполнения пользовательского запроса на семантическом уровне недостаточно вызова одно-

го сервиса, а необходимо выполнение нескольких сервисов. Задача композиции состоит в определении набора сервисов, согласованное взаимодействие которых обеспечивает новую семантическую функциональность, эквивалентную запросу пользователя.

Решение задачи композиции основывается на использовании следующих элементов ССОА: пользователь, реестр, машина логического вывода, онтологии. Взаимодействие элементов ССОА при решении задачи композиции сервисов представлено в виде схемы на рисунке 1.

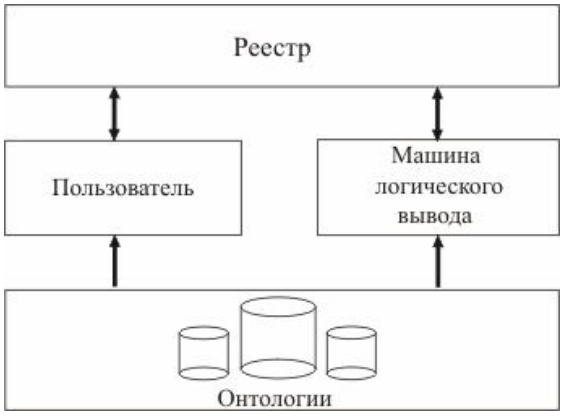

Рисунок 1 - Взаимодействие элементов ССОА при решении задачи композиции сервисов

Можно предложить следующий сценарий взаимодействия элементов.

1. Провайдер публикует сервис в реестре, анализируются связи публикуемого сервиса с другими сервисами реестра, происходит сохранение описания сервиса в БД реестра.

В ССОА сервис S_i формально представляется как некоторая функция, правило, позволяющее преобразовать набор входных параметров в выходные [4]. В терминах архитектуры сервис - это событийно-управляемый компонент: он реагирует на входные сообщения, выполняет внутренние действия и формирует выходные сообщения. Формализованное описание сервиса имеет вид:

$S_i : Inp_i \to Out_i$, где Inp_i – конечное множество входных параметров $\left\{Inp_i^1, Inp_i^2, ..., Inp_i^n\right\}$, а Out_i - конечное множество выходных параметров $\left\{Out_i^1, Out_i^2, ..., Out_i^m\right\}$.

Расширенное представление сервиса включает в себя описание условий, предшествующих выполнению сервиса (так называемых предусловий - $\Pr e$), и изменения состояния предметной области, порождаемых выполнением сервиса:

$S_i : \Pr e_i \to Output_i$ т $Effects_i$.

В этом представлении в множестве Out_i выделяются два подмножества:

- $Output_i$ - непосредственно выходные параметры;

- $Effects_i$ - эффекты, которые сервис может оказывать на предметную область, изменяя состояние окружающей его среды.

Расширенное представление отражает необходимое условие выполнения сервиса. Необходимое и достаточное условие имеет вид:

$S_i : \Pr e_i$ т $Inp_i \to Output_i$ т $Effects_i$.

Множества In_i, $\Pr e_i$, $Output_i$, $Effects_i$ задаются в терминах онтологий [5,6,7], которые определяют понятия и отношения предметной области. Для задания онтологий используется формализм дескриптивной логики [8].

2. Пользователь задает некоторую цель, конечный результат, описывает целевой сервис, то есть тот сервис, который он хочет найти. Описание конечной цели пользователя задается в том же формате, что и описание сервиса:

$S_g : Inp_g \to Output_g$.

3. На основании запроса пользователя осуществляется поиск целевого сервиса. В случае успешного обнаружения сервиса возвращается ссылка на его описание, в противном случае осуществляется попытка составить требуемый сервис из имеющихся описаний. Определение этих сервисов и порядок их запуска представляют собой цель композиции. Поиск нужных сервисов происходит на основе определения семантического соответствия существующих описаний сервисов запросу пользователя, поэтому композиция несет семантическое содержание. В случае успешного решения задачи композиции возвращается ссылка на описание последовательности сервисов. Результат композиции может быть удобно представлен с помощью языков описания процессов. Для обнаружения сервисов и их композиции предлагается использовать семантический реестр.

С функциональной точки зрения семантический реестр можно представить в виде множества взаимодействующих сервисов, то есть как систему со сложной динамической структурой, анализ поведения которой удобно осуществлять с помощью моделирования. Если рассматривать сервисы, как события в системе, а наличие входных данных сервисов – как условия выполнения событий, то динамика поведения такой системы может быть удобно отражена с помощью сети Петри:

$R = (P, S, I, O)$, где $P = \{p_1, p_2, ..., p_n\}$, $(n \geq 0)$– множество позиций, моделирующих условия, т.е. $p_i = \{Inp_{Si}^j \atop Out_{Si}^j$;

$S = \{S_1, S_2, ..., S_k\}$, $(k \geq 0)$ - множество переходов (сервисов);

$P \cap S = \varnothing$;

I – входная функция $I: P \to S$;

O – выходная функция $O: S \to P$.

Сеть, построенная таким образом, является свободно помеченной с некоторой функцией помечения $\sigma: S \to \sum$, где $\sum$- алфавит. Схема работы сервиса в сети: сервис S_i получает сообщение, выполняет некоторые действия, посылает сообщение. После этого сервис готов к принятию новых сообщений. Выполнение действий сервисом возможно только при условии наличия входных параметров, что соответствует наличию фишек в соответствующих позициях p_i сети R. Поэтому процесс выполнения сервиса в сети R представляет собой запуск разрешенного перехода.

Предлагается выделить следующие архитектурные элементы реестра: анализатор публикаций, хранилище сервисов, парсер запроса пользователя, блок работы с сетью Петри, транслятор в один из языков описания процессов, в частности в BPEL (рисунок 2). Используя такую модель семантического реестра, можно строго определить композицию сервисов, проверить ее выполнимость.

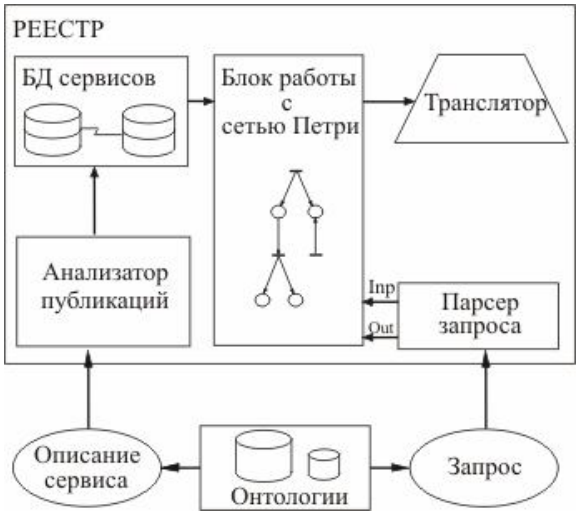

Рисунок 2 – Структура семантического реестра

Определение 1. Формально семантический реестр представляет собой множество семантических сервисов $S = \{S_1, S_2, ..., S_k\}$, между которыми могут существовать отношения выводимости.

Определение 2. Все взаимосвязанные сервисы реестра S образуют множество $S' = \{S_1', S_2', ..., S_{k'}'\}$, $S' \subseteq S$. Два сервиса S_i, $S_j : S_i, S_j \in S'$; $i, j \in [1, k']$, $i \neq j$, определяются как взаимосвязанные, если они связаны отношением выводимости или отношением частичного вывода.

Определение 3. Два сервиса S_i, $S_j : S_i, S_j \in S'$; $i, j \in [1, k']$, $i \neq j$, $S_{i,j} : Inp_{i,j} \to Out_{i,j}$, $Inp_{i,j} = \{Inp_{i,j}^1, Inp_{i,j}^2, ..., Inp_{i,j}^{n_{i,j}}\}$, $Out_{i,j} = \{Out_{i,j}^1, Out_{i,j}^2, ..., Out_{i,j}^{m_{i,j}}\}$ связаны отношением выводимости (сервис S_j выводится из S_i), если множество/подмножество выходных данных сервиса S_i может служить в качестве всего множества входных данных сервиса S_j (рисунок 3), то есть выполняется условие: $\forall p \in [1, n_j] : \exists l \in [1, n_i] : Inp_j^p \phi Out_i^l$.

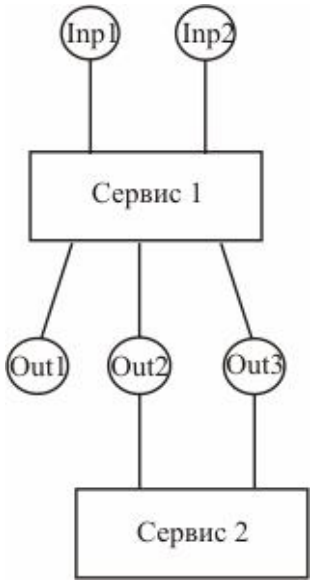

Рисунок 3 – Отношение выводимости между двумя сервисами

Определение 4. Два сервиса $S_i, S_j : S_i, S_j \in S', i, j \in [1, k'], i \neq j$, $S_{i,j} : Inp_{i,j} \to Out_{i,j}$, $Inp_{i,j} = \{Inp_{i,j}^1, Inp_{i,j}^2, ..., Inp_{i,j}^{n_{i,j}}\}$, $Out_{i,j} = \{Out_{i,j}^1, Out_{i,j}^2, ..., Out_{i,j}^{m_{i,j}}\}$ связаны отношением частичного вывода (сервис S_j частично выводится из S_i), если множество/подмножество выходных данных сервиса S_i может служить в качестве подмножества (но не всего множества) входных данных сервиса S_j, то есть выполняется условие: $\exists p \in [1, n_j] : \exists l \in [1, n_i] : Inp_j^p \phi Out_i^l$ (рисунок 4).

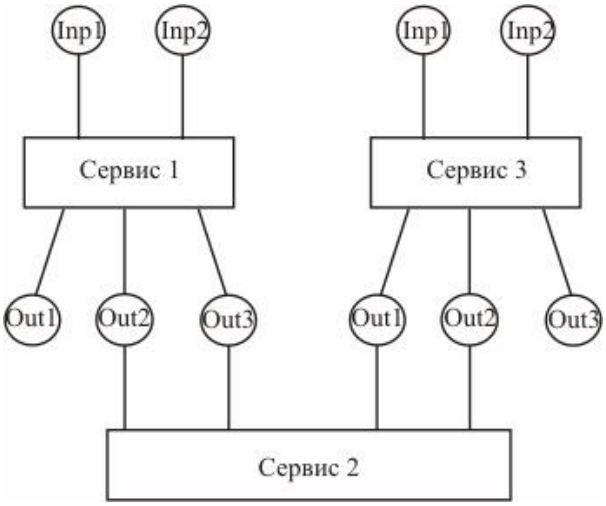

Рисунок 4 – Отношение частичного вывода между сервисами

При размещении сервиса в реестре и описании его входных и выходных параметров происходит анализ сервисов, содержащихся в реестре, на отношение выводимости нового сервиса и уже опубликованных в реестре. Определение взаимосвязанных сервисов на стадии их опубликования в реестре позволяет задать машину вывода, которая при инициализации входных данных целевого сервиса позволяет вывести возможные композиции сервисов.

Определение 5. Композицией множества сервисов $S_{comp} = \left\{ S^1_{comp}, S^2_{comp}, ..., S^k_{comp} \right\}, S_{comp} \in S'$,

$S_{comp} : Inp_{comp} \to Out_{comp}$,

$Inp_{comp} = \left\{ Inp^1_{comp}, Inp^2_{comp}, ..., Inp^{n_{comp}}_{comp} \right\}$,

$Out_{comp} = \left\{ Out^1_{comp}, Out^2_{comp}, ..., Out^{m_{comp}}_{comp} \right\}$ по отношению к целевому сервису $S_g : Inp_g \to Output_g$,

$Inp_g = \left\{ Inp^1_g, Inp^2_g, ..., Inp^{n_g}_g \right\}$,

$Out_g = \left\{ Out^1_g, Out^2_g, ..., Out^{m_g}_g \right\}$ называется совокупность сервисов семантического реестра (метасервис) таких, что выполняются условия:

1) $\forall S^i_{comp} \in S_{comp}$:

$Inp_{S^i_{comp}} \phi\ Inp_{comp} \Rightarrow Inp_{S^i_{comp}} \phi\ Inp_g$;

2) $Out_{S^i_{comp}} \phi\ Out_{comp} \Rightarrow$

$\Rightarrow (Inp_g \phi\ Out_{S^i_{comp}})$ т $(Out_{S^i_{comp}} \phi\ Inp_g)$.

При этом сервисы $S_{i\text{-}1}, S_i$ и S_i, S_{i+1} являются взаимосвязанными $\forall i \in [2, k\text{-}1]$. Графически композицию сервисов можно изобразить в виде схемы, представленной на рисунке 5.

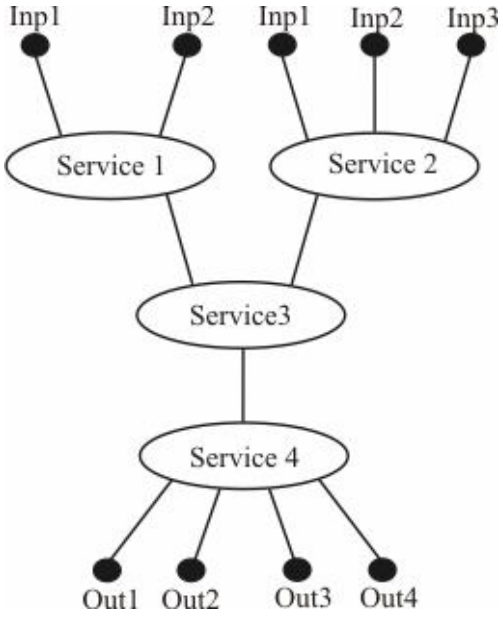

Рисунок 5 – Композиция сервисов

Выполнение композиции представляет собой инициализацию входных параметров метасервиса, собственно выполнение метасервиса и возврат его выходных данных.

Описанная формализация архитектурных элементов ССОА может быть положена в основу метода решения задачи композиции семантических сервисов.

Метод решения задачи композиции сервисов. Предлагается проводить решение задачи композиции в несколько этапов.

1. Определение начальной маркировки сети Петри семантического реестра.

2. Построение и анализ дерева достижимости с целью поиска сервиса(ов) удовлетворяющих запросу пользователя и определения последовательности выполнения сервисов (в случае если цель пользователя может быть выполнена реализацией нескольких сервисов).

3. Завершение алгоритма представляет собой возврат пользователю ссылки на сервис или последовательность сервисов.

Определение начальной маркировки сети Петри семантического реестра. Этап определения начальной маркировки необходим для определения сервисов, которые пользователь может запустить, используя множество начальных параметров Inp_g. Эти данные используются для построения пространства состояний сети Петри и его анализа на предмет семантического соответствия позиций сети R цели пользователя Out_g. Начальную маркировку можно записать в виде:

$M_{нач} = \left\{ \mu_1, \mu_2 ..., \mu_n \right\}$, $\mu_i \in \{0,1\}$, $i = 1, ..., n$,

где μ_n - маркировка позиции p_i, т.е. $\mu : P \to N$, где N - множество неотрицательных целых чисел.

Определение семантического соответствия данных, которыми обладает пользователь, и данных, необходимых для запуска сервисов, сводится к решению задачи выводимости из Inp_g дескриптивного логического выражения, описывающего в терминах предметной области условия запуска сервиса, соответствующего позиции p_i в сети, моделирующей реестр. В свою очередь в формализме дескриптивной логики задача выводимости сводится к задаче категоризации. Поэтому задача $Inp_g \models p_i$ сводится к задаче Inp_g т p_i.

При решении данной задачи возможно несколько вариантов.

1. Пользователь может ввести данные, которые категоризуют входные данные сервиса S_i, то есть Inp_g т $I(S_i)$. В этом случае пользователь вводит более общие данные.

2. Пользователь может ввести данные, которые категоризуются входными данными сервиса S_i, то есть $I(S_i)$ т Inp_g. В этом случае пользователь не может ввести все необходимые данные.

3. Пользователь может ввести те данные, которые необходимы для запуска сервиса, то есть Inp_g т $I(S_i)$ и $I(S_i)$ т Inp_g.

4. Сервис использует входные данные, которые пользователь не может ввести, то есть $(Inp_g \text{ у } \neg I(S_i)) \text{ ф } \bot$.

На этапе определения начальной маркировки реестра необходимо проверить множество всех входных позиций переходов $p_i : p_i \in I(S_j), \forall i, \forall j$ на отношение категоризации Inp_g и p_i и провести маркировку фишек. Тогда

$$\mu_i = \begin{cases} 0, \text{если } \forall j : \text{I}np_{g_j} : (\text{I}np_{pi} \text{ ф } \text{I}np_{g_j}) \vee ((\text{I}np_{pi} \text{ у } \text{I}np_{g_j}) \text{ф} \bot); \\ 1, \text{если } \exists j : (\text{I}np_{g_j} \text{ ф } \text{I}np_{pi}) \vee (\text{I}np_{g_j} \equiv \text{I}np_{pi}). \end{cases}$$

Построение дерева достижимости. Следующий этап решения задачи композиции состоит в построении дерева достижимости T сети Петри R, анализ которого будет использоваться на следующем этапе для поиска целевого сервиса или определения сервисов, участвующих в композиции. Целесообразно задать следующие правила построения дерева T:

1)определяется начальная маркировка сети R (корневая вершина дерева T) и разрешенные переходы;

2) любая вершина M_i дерева T представляет собой некоторую маркировку $(\mu_1,...,\mu_n)$ позиций сети R, полученную запуском какого-либо разрешенного перехода, удалением фишек из входных позиций этого перехода и помещением фишек в его выходные позиции;

3) если переход S_i разрешен, то после его выполнения переход S_j также будет разрешен, при условии, что соответствующие сервисы S_i, S_j связаны отношением вывода;

4) если переход S_i разрешен, а соответствующие сервисы S_i, S_j и S_h, S_j связаны отношением частичного вывода, то после выполнения перехода S_i переход S_j также будет разрешен, при условии, что переход S_h уже был выполнен на одном из этапов построения дерева T; в противном случае переход S_j будет запрещен.

Анализ дерева достижимости. После построения дерева T происходит анализ его вершин, конечной целью которого является обнаружение целевого сервиса или сервисов, необходимых для композиции. Соответственно выбираются только те вершины, семантическое описание которых соответствует описанию выходных параметров сервиса пользователя. Выбор осуществляется на основании оценочной функции F, которая для каждой вершины M_i определяет степень семантического соответствия выходным данным целевого сервиса всех тех позиций $p_j : \forall_j \ \mu(p_j) = 1, \mu(p_j) \in M_i$. Предлагается F задавать следующим образом:

$$F(M_i) = \sum_{j=1}^{n} V(\mu_j),$$

где

$$V(\mu_j) = \begin{cases} 0, \exists\, q : \ (Out_g^q \equiv Out_p^j) \vee (Out_g^q \text{ ф } Out_p^j); \\ 1, \exists\, q : \ Out_p^j \text{ ф } Out_g^q; \\ 2, \forall\, q : \ Out_p^j \text{ у } Out_g^q \text{ ф} \bot. \end{cases}$$

При таком задании оценочной функции выбор вершин с минимальным значением F гарантирует получение маркировки сети, которая наилучшим образом удовлетворяет запросу пользователя.

Анализ вершин дерева T с использованием оценочной функции может быть выполнен на основании жадного алгоритма [9]. Перед запуском алгоритма создается список целей пользователя $Goals = \{Out_g^1,...,Out_g^k\}$, список отобранных вершин $Selected = \{M_s^1,...,M_s^h\}$ и список посещенных вершин Visited = $Visited = \{M_v^1,...,M_v^h\}$.

Работа алгоритма осуществляется следующим образом:

1) для текущей вершины M_i рассчитывается оценочная функция вершин-потомков $F(M_j)$;

2) выбираются вершины-потомки M_j с минимальным значением функции $F(M_j)$;

3) вершина M_i помещается в список *Visited* ;

4) проверяются правила:

- если для вершины $p_i : p_i = mark^{-1}(\mu_j)$, $\exists j:\ (Out_g^j \equiv Out_p^i)$ т $(Out_g^j$ ф $Out_p^i)$, тогда из списка *Goals* исключается Out_g^j, а вершина M_i заносится в список *Selected* ;

- если для вершины $p_i : p_i = mark^{-1}(\mu_j)$, $\exists j:\ Out_p^i$ ф Out_g^j, тогда из списка *Goals* исключается Out_g^j и заносится новый элемент $Out_g^j - Out_p^i \equiv Out_g^j$ у $\neg Out_p^i$,, а вершина M_i заносится в список *Selected* ;

- если ни одно из вышеперечисленных правил не выполняется, то M_i не заносится в список *Selected* .

Завершение алгоритма. Завершение алгоритма происходит, когда все цели пользователя выполнены, то есть список *Goals* пуст, а список *Selected* имеет вид: $Selected = \{M_1, M_2, ..., M_h\}$. Следует отметить, что вершины данного списка в силу особенностей его заполнения обладают следующим свойством:

$\forall M_i\ \exists \left\{S_i^1, S_i^2, ..., S_i^k\right\} : I(S_i^1)$ ф Inp_g и

$(O(S_i^k)$ ф $Out_g) \vee (Out_g$ ф $O(S_i^k))$,

где $\left\{S_i^1, S_i^2, ..., S_i^k\right\}$ - последовательность переходов, выполнение которых приведет состояние сети R от разметки (корневой вершины дерева достижимости T) до разметки, обозначенной вершиной M_i. Очевидно, что при замене переходов $\left\{S_i^1, S_i^2, ..., S_i^k\right\}$ на соответствующие сервисы оказываются взаимосвязанными $S_{i\text{-}1}, S_i$ и S_i, S_{i+1} $\forall i \in [2, k\text{-}1]$. Таким образом, последовательность сервисов $\left\{S_i^1, S_i^2, ..., S_i^k\right\}$ представляет собой композицию сервисов, а множество таких композиций (совокупность последовательностей переходов для вершин дерева T, включенных в список *Selected*) представляет собой метасервис, выполнение которого означает реализацию целевого сервиса. То есть по построенному списку *Selected* определяются композиции сервисов, реализующие целевой сервис. Семантические описания сервисов, участвующих в композиции, транслируются в язык описания процессов, например, BPEL. Ссылка на описание возвращается пользователю.

Заключение. В настоящее время существуют подходы, позволяющие моделировать семантические веб-службы (semantic web services) [10, 11, 12]. Однако данные подходы являются только лишь основой для интеграции некоторых технологий Semantic Web и Web Services и не предоставляют средств для решения задачи композиции семантических сервисов на архитектурном уровне.

Представленный метод решения задачи композиции сервисов в семантической сервис-ориентированной архитектуре не привязан к какой-либо конкретной технологии и использует следующие основные элементы ССОА: сервис, пользователь, реестр, машина логического вывода, онтологии. В основе метода лежит моделирование реестра взаимосвязанных сервисов в виде сети Петри и анализ дерева достижимости данной сети с помощью машин логического вывода на предмет соответствия семантического описания сервисов формализованному представлению запроса пользователя. Для анализа дерева достижимости разработан алгоритм, использующий оценочную функцию, которая характеризует степень семантического соответствия запроса пользователя и элементов дерева.

Предложенный метод решения задачи композиции семантических сервисов апробирован на классе задач дистанционного контроля знаний [13]. В этой работе рассматриваются вопросы построения распределенных систем дистанционного контроля знаний учащихся на базе семантической сервис-ориентированной архитектуры, а также некоторые новые эффекты и возможности применения предлагаемого подхода.

Использование предложенного метода обеспечивает решение задачи интеграции компонентов распределенных информационных систем на семантическом уровне, что способствует повышению гибкости и адаптивности информационных систем.

Библиографический список

1. *Michalewicz Z.* Adaptive Business Intelligence. Springer, 2007. 249 с.

2. *Stojanovic Z.* Service-oriented Software System Engineering. Idea Group Publishing, 2005. 435 с.

3. *Черняк Л.* SOA – шаг за горизонт // Открытые системы. 2003. №09. С.31-35.

4. *Кашалкин, Д.Ю, Курчидис В.А.* Принципы построения семантической сервис-ориентированной архитектуры // Модел. и анализ информ. систем. 2007. Т. 14. №1. С. 48-53.

5. *Кашалкин Д.Ю, Лататтуев А.Н.* Принципы создания метаописаний ресурсов во всемирной сети // Модел. и анализ информ. систем. - 2005. Т. 12. №2. С. 12-16.

6. *Taniar D.* Web semantics and ontology. Idea Group Publishing, 2006. 423 с.

7. *Fensel D*. Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer, 2003. 104 с.

8. *Baader F*. The description logic handbook. Cambridge University Press, 2003. 573 с.

9. *Люггер Д*. Искусственный интеллект. М., 2003. 848 с.

10.OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S.

11. WSMO: Web Services Modeling Ontology. http://www.w3.org/Submission/WSMO.

12. SWSF:Semantic Web Services Framework. http://www.w3.org/Submission/2005/07/.

13. *Курчидис, В.А., Кашалкин Д.Ю., Назанский А.С.* Вопросы построения систем дистанционного контроля знаний с применением онтологий // Матер. 2-й науч.-методич. конф. преподав. математ. фак. и фак. инф. и вычисл. техн. / Яросл. гос.ун-т. – Ярославль : ЯрГУ. 2007. 102-105 с.