УДК 621.391

МЕТОД ВЕРИФИКАЦИИ КОМПЛЕКСНЫХ WEB-СЕРВИСОВ

Е.Б. Ткачева, Иссам Саад
Харьковский национальный университет радиоэлектроники
Раед Яхя Абдулгхафур
Одесская национальная академия связи имени А. С. Попова

Abstract *– In the article proposed formalisms for defining rules of Web-services orchestration and choreography that allows to combine elements of a distributed system in a single system. Proposed approach allows to perform formal association of individual system components into a single unit. Proposed verification method for distributed systems based on the model approach. E-networks are models for verification of distributed systems. Web-service's safety check is performed by finding deadlock conditions or any deviation from the specification process. This method allows taking into account the asynchronous nature of complex services and also performing stateful inspection: check for different Web-service consistency, check for compatibility requirements orchestration and choreography for each service. Proposed analysis of Web-service model properties such as boundedness, liveness, reachability, coverability, etcetera. Proposed in the article step by step verification method allows to perform a dynamic verification of the service by changing its structure or constituent elements, as well as repairing or timeout.*

Анотація *– У статті запропоновані формалізми, що визначають правила композиції і узгодження Web-сервісів, що дозволяють об'єднати елементи розподіленої системи в єдине ціле. Наведено метод верифікації розподілених систем, що базується на модельному підході та дозволяє враховувати асинхронну природу комплексних сервісів, а також виконувати динамічну перевірку.*

Аннотация *– В статье предложены формализмы, определяющие правила композиции и согласования Web-сервисов, позволяющие объединить элементы распределенной системы в единое целое. Приведен пошаговый метод верификации распределенных систем, базирующийся на модельном подходе, позволяющий учитывать асинхронную природу комплексных сервисов, а также выполнять динамическую проверку.*

Введение

Основной тенденцией последних лет в области бизнеса стал перенос многих процессов в Интернет. В частности, электронные платежи, интернет-магазины, бронирование мест в гостиницах и т.п. В большинстве своем техническая реализация таких услуг осуществляется с помощью Web-сервисов в рамках парадигмы сервис-ориентированной архитектуры (СОА).

Рост области применения СОА приводит к необходимости увеличения функциональности самих Web-сервисов, следовательно, и усложнения их архитектуры. В частности, все чаще приходится использовать комплексные и распределенные сервисы. Главной целью разработчиков в таком случае становится гибкое и корректное сочетание различных элементарных (единичных) сервисов в композитные (объединенные) сервисы в качестве составных услуг более высокого уровня, не исключая возможности рекурсии, - объединения композитных сервисов в более сложные структуры.

Парадигма СОА как раз и позволяет описать отношения и взаимодействия в среде потребителей услуг (конечных пользователей), поставщиков услуг и сервисных брокеров (связующего звена) и тем самым обеспечивает абстрактную среду функционирования сервисов.

В рамках СОА для предоставления услуг требуемого качества, обеспечения надежности и универсальности, необходимых современным распределенным системам, актуальным становится задача предоставления асинхронного обращения к сервису, а также отправка запросов к нескольким сервисам одновременно [1]. Для гарантирования доступности ресурсов дополнительным условием является возможность управления исключительными ситуациями, определение целостности цикла выполнения сервиса, обработка ошибок и возможных тайм-аутов работы. Также для обеспечения постоянно изменяющихся потребностей пользователей СОА подразумевает гибкую, динамическую и высокоадаптивную работу компонентов распределенных систем.

При разработке распределенных систем в соответствии с требованиями СОА разработчики и пользователи сталкиваются с рядом проблем. Так, не всегда возможна композиция сервисов, предоставляемых разными поставщиками, ряд затруднений возникает при реализации длительных распределенных операций (данная проблема связана с небольшим временем блокирования необходимым для выполнения операций ресурсов). При анализе существующих спецификаций наблюдается отсутствие единых понятий и стандартов относительно объектов и структуры взаимодействия различных компонентов сложного сервиса и т.д.

Основными авторами, посвятившими свое внимание решению указанных выше проблем, а также разработке формального описания взаимодействия сервисов в распределенных системах являются Дж. Алонсо, К. Пельтц, Касати, а также такие корпорации, как W3C, Oracle, Hewlett-Packard и многие другие, а также наши соотечественники К. Самуйлов, В. Курчидис и др.

В данной статье приведен анализ существующих языков описания, применяемых для композиции и взаимодействия сервисов, принципы их совместного функционирования, а также предложена методика верификации распределенных систем, которая основана на модельном подходе.

I. Композиция и взаимодействие Web-сервисов

Под композицией в общем случае подразумевается «моделирование направленных, внутренних бизнес-процессов» [2, 3], а под взаимодействием или согласованием Web-сервисов «спецификация взаимодействий между автономными бизнес-процессами», т.е. определение последовательности условий, при соблюдении которых несколько независимых участников обмениваются сообщениями с целью выполнения некоторой общей задачи (сервиса).

Технологии композиции Web-сервисов во многих случаях до сих пор не имеют единых определений и сценариев применения: отсутствует единый (универсальный) язык описания компонентов сервиса, единые стандарты и спецификации технических характеристик распределенных систем. Так, с точки зрения разработчика можно выделить две позиции для композиции и взаимодействия. Согласно имеющемуся обобщенному подходу [4] данные понятия можно разделить на частные (внутренние) и общие.

Композиция бизнес-процессов относится к частным понятиям и описывает процесс, протекающий между сервисами, как набор атомарных сервисов, которые контролируются одним из участников, принимающих участие в предоставлении сервиса,

и определяет логику выбора процессов. Так, композиция сервисов подразумевает собой способность контролировать исполнение единиц программного обеспечения (например, приложений, компонентов, сервисов) для достижения заданного результата. Композиционными по своей природе являются композитные сервисы. Поэтому построение процесса на основе логики композиции стандартизует представление процесса в масштабе всей организации (участника, предоставляющего сервис) [3, 5].

Взаимодействие или согласование сервисов не имеет участника, носит открытый характер и не имеет централизованного управления. Примером может служить процесс установления соединения различных протоколов прикладного уровня (SIP). Важным при взаимодействии бизнес-процессов является соблюдение верной последовательности их выполнения. В табл. 1 приведены основные компоненты, которые формально представляют понятия согласования и композиции сервисов.

Таблица 1. Согласование и композиция Web-сервисов в распределенных системах

Согласование	Композиция
Общие (открытые) бизнес-процессы. Носит общедоступный характер для всех участников взаимодействия	Частные (внутриорганизационные) бизнес-процессы Носит закрытый характер, доступный только определенному процессу или сервису
Согласование сервисов – их взаимодействие между собой, включающее набор протоколов совместной работы	Композиция сервиса – логическая взаимосвязь между процессами при формировании сервиса – исполняемые процессы
Ключевые компоненты согласования: структура сообщений, асинхронная и синхронная коммуникация сервисов, служебные сообщения.	Ключевые компоненты композиции: участник и его роль, переменные и свойства, определяющие взаимодействие участников, обработчики ошибок, события, логические связки между событиями.

При составлении новых сложных сервисов в распределенных системах разработчики должны учитывать ограничения, накладываемые существующими координационными протоколами взаимодействия конечных пользователей и внутренними правилами функционирования сервисов. На рис. 1 приведен пример согласования (взаимодействия) и композиции Web-сервисов.

Техническими требованиями для корректного выполнения Web-сервисов в соответствии с концепцией СОА являются [1, 3, 4]:

1. Гибкость. Разделение между логикой процесса и Web-сервисами.
2. Простые и структурированные действия. Процесс согласования сервисов должен поддерживать действия как для обращения к другим Web-сервисам, так и для описания семантики процесса. Простое действие можно рассматривать как компонент, взаимодействующий с чем-то вне процесса, в то время как структурированное действие управляет общим выполнением процесса, специфицируя состав и порядок действий.
3. Рекурсивная композиция. Отдельный бизнес-процесс может взаимодействовать с множеством Web-сервисов. Сам процесс может быть представлен как Web-сервис для агрегации в процесс более высокого уровня.

Последовательная обработка транзакций и корреляция запросов. Долго выполняемые сервисы должны обеспечивать транзакционную целостность и управление исключениями, а также открытый доступ к множеству ресурсов.

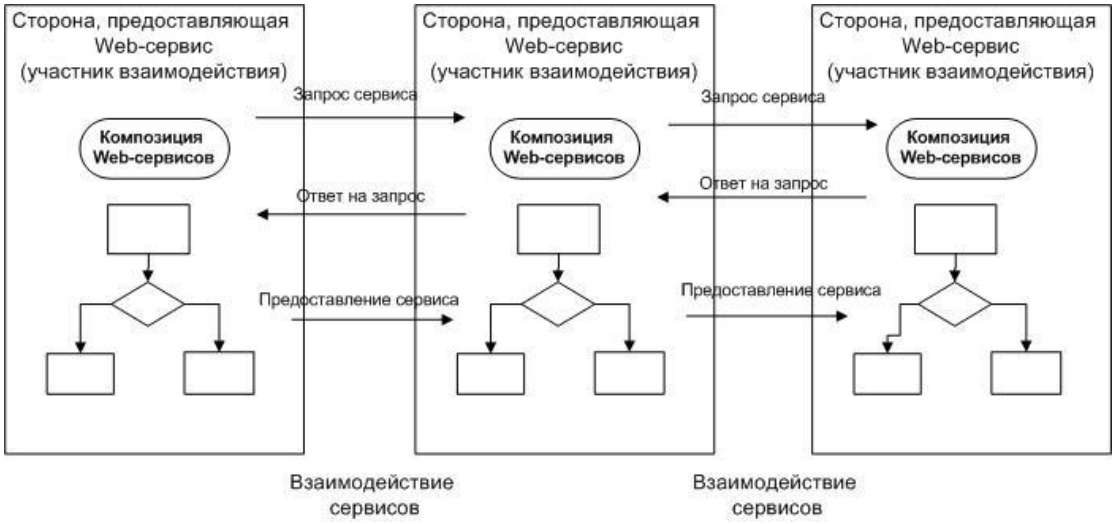

Рис. 1. Пример взаимодействия и композиции и сервисов

II. Обзор существующих решений моделирования бизнес-процессов

Первым из языков, предназначенным для описания последовательных, параллельных и многовариантных потоков работ, является спецификация **BPEL4WS (BPEL)**, которая позволяет моделировать поведение Web-сервисов при взаимодействии бизнес-процессов [2, 4]. Ее составной частью является грамматика на основе XML, которая предназначена для описания логики управления при координации Web-сервисов, участвующих в потоке работ бизнес-процесса.

Структурные действия управляют потоком работ бизнес-процесса в целом, определяя последовательность вызова Web-сервисов. Эти действия также поддерживают выполнение циклов и динамическое ветвление. Два других важных элемента BPEL – партнерские связи и переменные.

Язык WSCI.В общем виде WSCI определяет взаимодействие или обмен сообщениями между Web-сервисами. Спецификация обеспечивает корреляцию сообщений, правила упорядочения, обработку исключительных ситуаций, транзакции и динамическое взаимодействие [5]. Однако интерфейс WSCI описывает участие в обмене сообщениями с позиции лишь одного из партнеров. Согласование сервисов с точки зрения WSCI должно включать в себя набор интерфейсов WSCI – по одному для каждого партнера, участвующего во взаимодействии. В WSCI отдельный процесс не может управлять взаимодействием. Каждое элементарное действие WSCI представляет собой единичный процесс, связанный с определенной операцией WSDL. Спецификация WSCI расширяет возможности WSDL, позволяя упорядочить выполнение доступных операций WSDL. Другими словами, WSDL для каждого доступного сервиса описывает точки входа, а WSCI определяет взаимодействия операций WSDL.

WSCI поддерживает как базовые, так и структурные действия. Тег *action* определяет базовое сообщение для запроса или ответа. Каждое действие определяет соответствующую операцию WSDL и конкретного участника, который ее выполняет. К внешним сервисам можно обратиться с помощью тега *call*. WSCI поддерживает разнообразные структурные действия, включая выполнение циклов, последовательную и параллельную обработку.

Следующий интерфейс WSCI определяет процесс закупки, в который входят два последовательных действия - Receive Order («получить заказ») и Confirm («подтвердить»). Каждое действие отображается в набор абстрактных операций WSDL, а WSCI устанавливает между ними корреляцию. WSCI поддерживает бизнес-транзакции и обработку исключительных ситуаций. Разработчик бизнес-процесса может установить определенный контекст транзакции в рамках интерфейса WSCI, подобный контексту действия в BPEL4WS. Этот контекст определяет группу действий, неудачное завершение любого из которых «откатит» назад всю группу.

Язык BPML.BPML - язык описания бизнес-процессов на основе XML. BPML содержит конструкции для описания действий и потока работ бизнес-процесса [2, 6]. Наряду со структурными действиями для организации ветвления, последовательной и параллельной обработки, циклов и синхронизации процессов обеспечиваются базовые действия для отправки и получения сообщений, вызова сервисов.

Язык был разработан для управления длительными процессами и включает в себя функции долговременного хранения данных. Обмен данными между участниками ведется в формате XML с использованием ролей и определений партнеров, подобных конструкциям BPEL. Также BPML поддерживает рекурсивную композицию, предназначенную для формирования комбинированных бизнес-процессов. BPML поддерживает как длительные, так и кратковременные транзакции, используя для управления правилами компенсации понятие контекста, которое аналогично применяемому в BPEL [7].

III. Согласование и композиция совместной работы Web-сервисов

На сегодняшний день не существует стандартизированных протоколов взаимодействия совместной работы сервисов, которые представлены различными разработчиками. В общем случае протокол совместной работы относится к организации обмена сообщениями между несколькими участниками бизнес-процесса, в то время как композиция сервисов представляет собой частный (закрытый) процесс, подконтрольный отдельному участнику. Язык BPEL поддерживает как исполняемые процессы, так и организацию взаимодействия между ними. BPML и WSCI могут работать вместе так, чтобы BPML моделировал выполнение бизнес-процесса, а WSCI - хореографию Web-сервисов [1, 4, 7].

Формальное представление элементарного Web-сервиса посредством композиции и согласования процессов имеет вид:

$$S : Inp \rightarrow Out ,$$

где S – сервис; Inp – конечное множество входных параметров, влияющих на выполнение бизнес-процесса, $\{Inp1, Inp2, Inp3 \ldots Inpn\}$; Out – конечное множество выходных параметров $\{Out1, Out2, Out3 \ldots Outn\}$.

Расширенное представление сервиса включает в себя описание условий, предшествующих выполнению сервиса (так называемых предусловий – Pre), и изменений предметной области, порождаемых выполнением Web-сервиса. В этом случае во множестве Out выделяется два подмножества: множество $Output$ – непосредственно выходные параметры; множество $Effects$ –эффекты, которые сервис может оказывать на предметную область, изменяя состояние окружающей его среды.

$$S : Pre \to Output \cup Effects . \quad (1)$$

Формула (1) отражает необходимое условие выполнения сервиса. Необходимое и достаточное условие имеет вид:

$$S : Pre \cup Input \to Output \cup Effects . \quad (2)$$

Множества $Inp; Pre; Output; Effects$, приведенные в формуле (2), задаются в спецификации при разработке определенного сервиса и зависят от предметной области предоставляемого сервиса.

С помощью формальных методов композиция сервисов в соответствии с концепцией СОА и спецификацией BPEL языка может быть задана следующим набором выражений:

$P := Stop \mid Skip$ – начало или пропуск процесса композиции сервисов;

$\mid inv!ch \to P$ – вызов сервиса или процесса, который принимает участие в предоставлении сервиса;

$\mid inv?x \to P$ – определение вызываемого сервиса или процесса, который принимает участие в предоставлении сервиса;

$\mid ch!exp \to P$ – входной интерфейс (входное значение обработки запроса);

$\mid ch?x \to P$ – выходной интерфейс (выходное значение);

$\mid x := exp; P$ – назначение сервиса;

$\mid if\ b\ P\ else\ Q$ – условное ветвление сервисов;

$\mid P \Xi Q$ – выбор логического действия при композиции сервисов;

$\mid P \Delta Q$ – прерывание одного сервиса или процесса, который принимает участие в предоставлении сервиса за счет выполнения другого сервиса или процесса;

$\mid P \cap Q$ – взаимодействие сервисов;

$\mid P; Q$ – последовательное выполнение,

где P и Q – компоненты Web-сервиса, которые участвуют в согласовании и целостном предоставлении сервиса, это могут быть элементарные сервисы либо процессы, принимающие участие в предоставлении сервиса; символ Ξ может представлять любою логическую операцию ($\cup, \cap, !, ||, \&$).

Согласование или взаимодействие сервисов может быть описано глобальным протоколом, который регулирует поведение участников взаимодействия друг с дру-

гом. Так, каждый участник взаимодействия имеет свои собственные процессы, при этом согласование направляет процесс взаимодействия, который отвечает выдвигаемым требованиям.

Формально элементы согласования или взаимодействия Web-сервисов могут быть представлены следующим образом:

$X :: = Stop \mid Skip$ – бездействие и прекращение предоставления сервиса;

$\mid svr(A, B, ch\tilde{}) \to X$ – вызов сервиса в соответствии с определенными условиями выполнения;

$\mid ch(A, B, exp) \to X$ – открытие канала связи с определенным интерфейсом;

$\mid x := exp; X$ – назначение сервиса;

$\mid if\ b\ X\ else\ J$ – условие выполнения;

$\mid X \wedge J$ – выбор одного из нескольких сообщений;

$\mid X \leftrightarrow J$ - чередование сообщений;

$\mid X; J$ – последовательное выполнение сообщений,

где X и J – процессы предоставления Web-сервиса, которые учувствуют во взаимодействии, A, B – входные условия.

Любой сервис может быть подвержен ряду критических факторов (например, сервис может быть заблокирован после того, как буфер одного из участников будет переполнен). Формула для описания подобного рода ошибки при выполнении сервиса может быть представлена следующим образом:

$$X(mess1(start) \to (mess1 : RETR(A, B, exp)) \to$$
$$\to (S : messageDel \mid x) U (mess1 : DELE) \to X(mess2\_ch?x) \to Delivery\_data \to (Stop)).$$

Приведенная выше формула описывает процесс информирования о невозможности доставки *сообщения1* по причине заполнения входного буфера.

$$G((m : out \wedge m : in) U \neg(m : out)) \mid\to G(m : out \wedge \neg m' : out \wedge F(m' : out) \to$$
$$\to F(m : in \wedge \neg m' : in) \wedge F(m' : out))). \quad (3)$$

В формуле (3) формально приведено совместное представление композиции и взаимодействия сервисов: требование «канал сохраняет порядок транзакций», описывающее согласование процессов и требование «после получения подтверждения о том, что сообщение доставлено получателю, происходит его удаление из буфера .out».

IV. Верификация Web-сервисов

Верификация Web-сервисов отличается от верификации телекоммуникационных протоколов, так как Web-сервисы по своей природе являются сервис-ориентированными и могут гибко изменять значение параметров в зависимости от требований пользователей в реальном масштабе времени, а также учитывать асинхронный режим взаимодействия и возможность множественного доступа к ресурсам распределенной системы [8].

Основными требованиями при разработке и предоставлении Web-сервисов в распределенной системе являются [3, 4, 9]:

– **совместимость интерфейсов** – сосредоточение внимания при разработке сервиса на семантику кореллирующих сообщений, отслеживание заданной последовательности, приема, передачи и получение сообщений ответов между взаимодействующими процессами участников.

Формально совместимость интерфейсов может быть задана следующим образом:

$$Input: O \to сообщение \cup Input(o) = M(O), где\ \_o - [inv?m], [inv!\ m];$$
$$Output: O \to сообщение \cup Output(o) = M(O), где\ \_o - [inv!\ ch?], [inv!\ ch].$$

где O – процесс согласования сервисов для входного (Input) интерфейса и выходного (Output) соответственно; m – факторы, определяющие тип вызываемого сервиса; ch – вызов сервиса, необходимого для дальнейшей работы

– **безопасность (активность) предоставления сервисов** – предоставление разработчиками гарантий, что композиция процессов, входящих в предоставляемый сервис, не имеет тупиковых состояний, состояний блокировок, не производит несанкционированное увеличение процессов.

Данное требование в большей степени относится к взаимодействию сервисов. Так, взаимодействие X функционирует корректно, если для любого множественного процесса мы имеем одинаковое количество последующих исполняемых процессов:

$$X = \{m(P, A, B), \phi_{input\_output}(o)\} \to \phi_{input}(x) = \phi_{ioutputt}(y), \quad (4)$$

где ϕ – функция, соответствующая условиям предоставления сервиса (логика выполнения процесса o), x и y – переменные, зависящие от входных условий A, B и имеющие следующий тип взаимосвязи $receive[o](x) \Rightarrow reply[o](y)$;

– **живость сервисов** – последовательность процессов должна иметь логическое завершение: сообщения, передающиеся в процессе выполнения сервиса, должны быть обязательно доставлены. При этом должны выполняться оговоренные требования относительно качества предоставляемых сервисов (время задержки, последовательность транзакций).

Формально данное требование может быть представлено следующим образом:

$$M(S) = (S_0 \to S, L, pre), \quad (5)$$

где S - набор состояний сервисов или процессов, предоставляющих композиционный сервис; S_0 – начальное состояние (вызов сервиса). S – набор предоставленных сервисов; $\to \subseteq S_0 \times pre \cup S \times pre \cup L$ – правило взаимодействия сервисов.

Пример взаимодействия и успешного выполнения следующих бизнес-процессов: в процессе осуществления банковского сервиса участнику предоставляется следующий список операций: $o1[?\ getF; !\ ListF\ ]$ и возвращает список следующих транзакций $o2[?\ getCho]$, предоставляя пользователю выбор: $o3[?\ cancel]$ – отказ от обслуживания или $o4[?\ payment]$ – отослать подтверждение об оплате

$$A = (PA = fC : o2g; InA = fo1; o2; o3; o4g; BA(x; y))$$
$$with\ BA(x; y) = rec[o1](x); rep[o1](x); rec[o2](x); scope[time; f(rec[o4](y); inv[C : o2]); (rec[o3](x); empty)g].$$

Для верификации систем рассматриваемого класса зачастую применяется модельный подход. При использовании данного подхода свойства и характеристики, которыми должна обладать распределенная система для выполнения определенной задачи, задаются формально, при этом учитывая структуру и поведенческие свойства процессов. Использование модельного подхода совместно с семантической составляющей, описывающей реальное взаимодействие процессов, позволяет выполнить верификацию распределенных систем в соответствии с основными требованиями. В качестве моделей верификации распределенных систем предложено использовать E-сети.

Так, проверка на безопасность (активность) процессов выполняется нахождением тупиковых состояний или любым отклонением от спецификации процесса может быть сведена к проверке таких свойств, как ограниченность, покрываемость. Проверка живости Web-сервиса на модели E-сети, которые приводят к конечным результатам, которые совпадают с требованиями спецификации [9]. Т.е. композиция сервисов достигает своего завершения, проверка данного свойства может быть сведена к покрываемости и достижимости.

Процесс верификации распределенных систем предусматривает тесное взаимодействие между исполнителем (организацией, предоставляющей процесс) и его разработчиком Web-сервиса, который предоставляет требования к функционированию сервиса. При построении общей модели сервиса важным моментом является представление корректного взаимодействия внутренних и внешних процессов различных участников. На рис. 2 приведена обобщенная схема процесса верификации Web-сервисов.

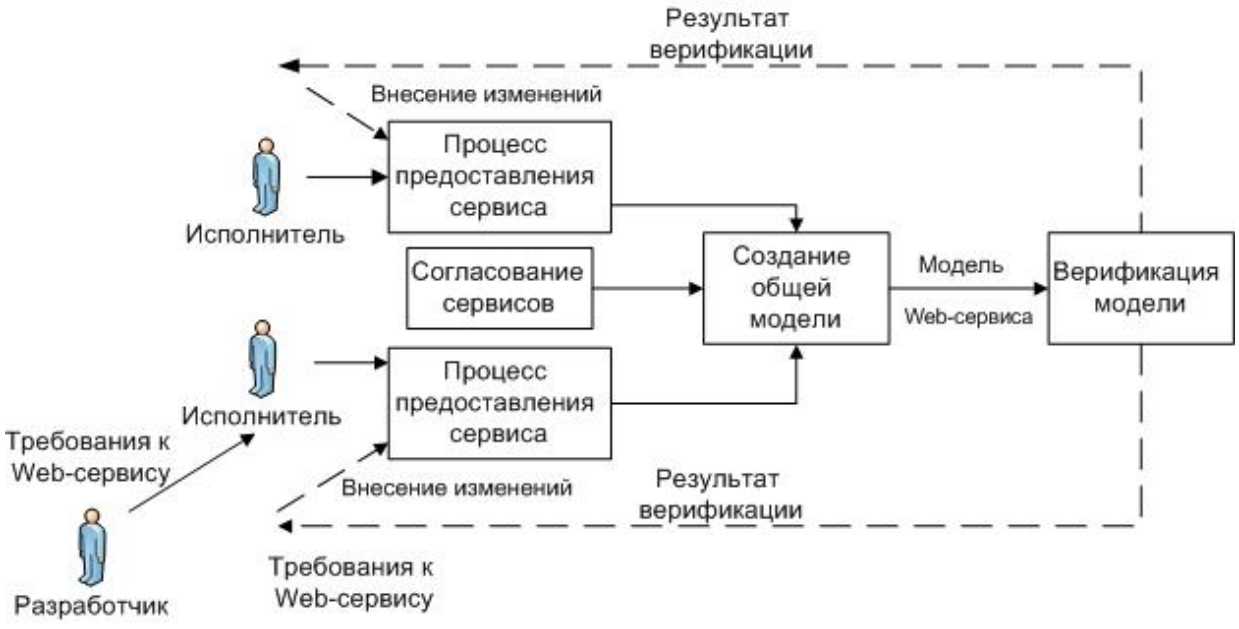

Рис. 2. Верификация Web-сервисов

В процессе верификации Web-сервисов необходимо выполнить следующие шаги:

1. Формальное представление композиции процесса предоставления каждого сервиса.

2. Формальное представление взаимодействия композиции сервисов.

3. Построение обобщенной модели комплексного Web-сервиса на основе Е-сети, которая соответствует требованиям спецификации (модель спецификации).

4. Построение обобщенной модели комплексного Web-сервиса на основе Е-сети, которая соответствует формализированному представлению процессов (модель реализации).

5. Анализ таких свойств моделей, как достижимость, живость, покрываемость, наличие блокировок; поверка соответствия моделей спецификации и реализации.

Верификация сервиса на основе модельного подхода позволяет выполнить проверку на непротиворечивость при предоставлении Web-сервиса; выполнить проверку на совместимость требований композиции и согласования для каждого сервиса.

Выводы

Современные инфокоммуникационные системы представляют собой сложные разнородные многокомпонентные структуры, которые имеют как синхронную, так и асинхронную природу предоставления услуг. Такая природа систем накладывает ряд требований при проектировании и разработке Web-сервисов реализующих данные услуги.

Так, процессы, участвующие в предоставлении услуги, могут быть разделены на частные, которые выполняют определенные логические операции в единичном узле системы и общие, которые определяют процесс взаимодействия между узлами системы.

Первые определяются посредством композиции процессов, вторые – согласования процессов.

Для формального описания требований к Web-сервисам разработан ряд языков спецификации, наиболее используемыми из них являются BPML, BREL, WSDL, WSCI. Однако открытыми остаются вопросы согласования композиции и взаимодействия Web-сервисов, а также их корректного функционирования. Основным методом, позволяющим выполнить проверку корректности функционирования Web-сервисов, является верификация. Для верификации систем рассматриваемого класса предложено использовать модельный подход.

В статье предложены средства описания композиции и согласования Web-сервисов. Предложенный подход позволяет выполнить формальное объединение отдельных узлов системы в единое целое, а также служит основой, для построения обобщенной математической модели, используемой для верификации.

Приведенный в статье пошаговый метод верификации позволяет выполнять динамическую проверку выполнения сервиса при изменении его структуры или составляющих элементов, а также в случае восстановления или тайм-аута.

Список литературы:

1. *Westerman J.* SOA Today: Introduction to Service-Oriented Architecture. [Электронный ресурс]. – Режим доступа: http://www.information-management.com/news/7992-1.html.

2. *Peltz C.* Web services orchestration and choreography // IEEE Computer. – 2003. – Vol. 36, № 10. – P. 6-52.

3. *Alonso G., Casati F., Kuno H., Machiraju E.* Web services—Concepts, architectures and applications.– Berlin Heidelberg: Springer-Verlag, 2008. – 205 p.

4. *Peltz C.* Web services orchestration and choreography—A review of emerging technologies, tools, and standards // Hewlett-Packard Company, 2003. – 7 p.

5. Web Service Choreography Interface (WSCI) 1.0 [Электронный ресурс] / *Airkin A., Askary S., S. Fordin. et al* – W3C Working Group, 2002. – Режим доступа: http://www.w3.org/TR/wsci

6. *Weerawarana S., Francisco C.* Business Process with BPEL4WS: Understanding BPEL4WS, Part 1. Research report, IBM developerWorks [Электронный ресурс] – 2002. – 8 p. – Режим доступа: http://www.ibm.com/developerworks/library/ws-bpelcol1/ws-bpelcol1-pdf.pdf.

7. *Langdon C. S.* The state of Web services // IEEE Computer. –2010. – Vol. 36, №7. – P. 96-99.

8. *Baldoni M., Baroglio C., Martelli A.* Verifying the conformance of web services to global interaction protocols: A first step // International Workshop on Web Services and Formal Methods. – 2005. – P. 196-212

9. *Bravetti M., Guidi C., Lucchi* R. Supporting e-commerce systems formalization with choreography languages // Proceedings of the 2005 ACM symposium on Applied computing. - New York, USA, 2005. – P. 831–835.