БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Слюсар В.В.* Системы MIMO: принципы построения и обработка сигналов // Электроника: Наука, Технология, Бизнес. – 2005. – № 8. – С. 52-58.
2. *Рыжов В.П.* Выбор базиса разложения сигналов на основе критерия минимума среднего риска // Радиотехника. – 1983. – № 9. – С. 51-53.
3. *Тихонов В.И.* Статистическая радиотехника. 2-е изд., перераб. и доп. – М.: Радио и связь, 1982. – 624 с.
4. *Лезин Ю.С.* Оптимальные фильтры и накопители импульсных сигналов. – М.: Сов. радио, 1969. – 448 с.
5. *Харкевич А.А.* Борьба с помехами. 3-е издание. – М.: КД Либроком, 2009. – 275 с.
6. *Гоноровский И.С.* Радиотехнические цепи и сигналы. – 5-е изд., испр. и доп. – М.: Дрофа, 2006. – 719 с.

Дулин Михаил Игоревич
Учебный военный центр при федеральном государственном образовательном учреждении высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: tiski@list.ru.
347928, г. Таганрог, пер. Некрасовский, 44.
Тел.: 88634371690.

Dulin Michael Igorevich
The educational military centre at federal state educational institution of the higher vocational training «Southern federal university» in Taganrog.
E-mail: tiski@list.ru.
44, Nekrasovsky, Taganrog, 347928, Russia.
Phone: +78634371690.

УДК 681.03.06

И.Г. Данилов

СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА КАК ОСНОВА ДЛЯ ПОСТРОЕНИЯ СОВРЕМЕННЫХ РАСПРЕДЕЛЁННЫХ СИСТЕМ

Рассматриваются основные SOA-технологии и их взаимосвязь с современными облачными и грид системами.Сервис-ориентированная архитектура (SOA) возникла в результате потребности крупных ИТ-предприятий в подходе к созданию крупных корпоративных программных продуктов, основанном на промышленной платформе интеграции с многократным использованием функциональных элементов информационных технологий. Данный подход был адаптирован при реализации таких парадигм распределённого программирования, как грид и облачные вычисления.

Сервис-ориентированная архитектура; веб-сервисы; облачные вычисления; грид-технологии.

I.G. Danilov

SERVICE-ORIENTED ARCHITECTURE AS A BASIS FOR MODERN DISTRIBUTED SYSTEMS DESIGN

This paper describes some basic SOA-technologies and theirs relationship with modern cloud and grid systems. Service-oriented architecture (SOA) is the result of large IT companies needs in the approach for creation of large corporate software products, based on industrial integration platform with multiple reusable functional IT-elements. This approach was adopted for the implementation of such distributed programming paradigms, as grid and cloud computing.

Service-oriented architecture; web services; cloud computing; grid computing.

Введение. Сервис-ориентированная архитектура (от англ. ***S**ervice-**O**riented **A**rchitecture, SOA*) это такая архитектура приложения, в которой компоненты или «сервисы», имея согласованные общие интерфейсы, используют единые правила (контракты) для определения того, как вызывать сервисы и как они будут взаимодействовать друг с другом [1].

Основу SOA составляют принципы многократного использования функциональных элементов ИТ, ликвидации дублирования функциональности в ПО, унификации типовых операционных процессов, обеспечения перевода операционной модели компании на централизованные процессы и функциональную организацию на основе промышленной платформы интеграции [2].

Компоненты программной системы могут быть распределены по разным узлам сети, и предлагаются как независимые, слабо связанные и, возможно заменяемые сервисы-приложения. Программные комплексы, разработанные в соответствии с SOA, часто реализуются как набор веб-сервисов, интегрированных при помощи известных стандартных протоколов (SOAP, WSDL, и т. п.).

Рассмотрим основные технологии, лежащие в основе построения систем с использованием веб-сервисов.

XML. XML (от англ. *e**X**tensible **M**arkup **L**anguage*) – простой формат данных, разработанный для обеспечения хранения и передачи информации в структурированном виде. Фактически – это текст, одновременно хорошо читаемый человеком и пригодный для автоматического анализа вычислительными машинами. Одним из основных достоинств XML является его расширяемость за счёт создания **словарей**. В настоящее время существуют спецификации, с помощью которых можно описать не только свой собственный словарь, но и любой XML-документ вообще. Одна из таких спецификаций – XML Schema [3] или ***XML S**chema **D**efinition* (XSD). Для примера рассмотрим простой XML-документ, описывающий небольшую записку-напоминание:

```
<?xml version="1.0"?>
<note>
<to>Игорь</to>
<from>Вася</from>
<heading>Напоминание</heading>
<body>Не забудь о нашей завтрашней встрече!</body>
</note>
```

Для данного примера XSD схема будет выглядеть следующим образом:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
<xs:element name="note">
  <xs:complexType>
   <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

По сути XML Schema описывает некоторые правила (словарь), которому соответствует некоторое подмножество XML документов. Любой XSD документ содержит корневой элемент (в нашем случае *xs:schema*) и любое количество вложенных. В примере элемент *note* имеет комплексный (сложный) тип – *complexType* – и содержит несколько элементов простого строкового типа: *to*, *from*, *heading*, *body*. Тег *xs:sequence* управляет порядком дочерних элементов, которые должны присутствовать в результирующем XML-документе в том же порядке, в котором объявлены в схеме.

WSDL. Чем сложнее и масштабнее становится распределённая система, тем важнее становится возможность описания коммуникационных взаимодействий её частей в некотором определённом структурированном виде. Для этой цели применяется WSDL (от англ. ***W**eb **S**ervices **D**escription **L**anguage*) – язык описания веб-сервисов, основанный на XML. WSDL-документ описывает сервисы как набор сетевых точек конечного доступа или **портов**. Определённые типы портов – это наборы абстрактных **операций**. Конечные сетевые точки доступа и **сообщения**, которыми они обмениваются, описываются без привязки к специфичным сетевым протоколам и форматам данных, что упрощает их дальнейшее повторное использование. Конкретный же коммуникационный протокол и форматы данных образуют так называемое **связывание** для порта определённого типа. Порт определяется путем объединения сетевого адреса и связи, а наборы портов определяют сервис в целом. Обобщив, можно сказать, что для описания сетевых сервисов в WSDL-документе используется следующий набор элементов [4]:

- *типы* – контейнер для описания типа данных, использующий некоторую систему типов (например, XSD);
- *сообщения* – абстрактное, основанное на типах описание данных, с помощью которых осуществляется коммуникация;
- *операция* – абстрактное описание определённого действия, поддерживаемого сервисом;
- *тип порта* – абстрактный набор операций, поддерживаемых одной или несколькими точками доступа;
- *связывание* (связь) – конкретный протокол и спецификация формата данных для определённого типа порта;
- *порт* – одиночная конечная точка доступа, определяемая как комбинация связи и сетевого адреса;
- *сервис* – набор связанных конечных точек доступа.

Следует понимать, что, сам по себе, WSDL не является новым языком описания типов, а представляет собой способ описания формата сообщений, поддерживающий стандарт XML Schema, как базовую систему типов. Также имеется возможность использования других языков описания типов через систему расширений.

WSDL определяет общий механизм связывания, используемый для объединения конкретного протокола или формата данных/структуры и абстрактного сообщения, операции или конечной точки доступа. В дополнении к ним определяется набор расширений механизмов связывания для следующих протоколов и форматов сообщений:

- SOAP 1.1;
- HTTP GET/POST;
- MIME.

На практике сам WSDL, механизмы связывания и расширения реализованы в виде словарей XML с использованием XML Schema.

SOAP. Чаще всего в качестве коммуникационного протокола, используемого при построении распределённых систем на основе веб-сервисов, применяется ещё

один XML-словарь: протокол SOAP. Данный протокол предназначен прежде всего для обмена структурированными сообщениями, а также для удалённого вызова процедур или доступа к веб-сервисам.

В качестве примера [5] рассмотрим запрос к серверу *GetStockPrice*. Данный запрос имеет единственный параметр *StockName* — название компании, цену акций которой мы хотим узнать. Пространство имен XML определено в http://www.example.org/stock.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

В ответе через параметр *Price* возвращается запрашиваемая цена акций:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>
</soap:Envelope>
```

Протокол предоставляет возможность для коммуникации между приложениями вне зависимости от платформы, на которой они запущены, технологий и языка их реализации. SOAP может использоваться поверх различных протоколов прикладного уровня, но чаще всего используется поверх HTTP, который поддерживается всеми Интернет браузерами и серверами.

SOAP-сообщение [6] – обычный XML документ, который содержит следующие элементы:

- элемент *Envelope*, который определяет, что данный XML документ является SOAP-сообщением;
- элемент *Header* – заголовок, содержащий опциональную специфичную для приложения информацию;
- элемент *Body*, содержащий основную информацию сообщения;
- элемент *Fault*, содержащий информацию об ошибках и статусную информацию.

Облачные вычисления и технологии грид. В общем случае, как грид, так и облачные системы, были созданы для обеспечения доступа пользователей к определённым компьютерным ресурсам через сеть. В случае с грид, владельцами этих ресурсов являются различные организации, каждая из которых вносит свой собственный вклад в построение разделяемой информационной среды. При этом пользователями могут являться как владельцы системы, так и сторонние лица, вместе образующие новый уровень абстракции – *виртуальные организации* (ВО) [7]. Ка-

ждая ВО имеет доступ к определённой части ресурсов, используемых в рамках проекта. Ресурсы облачных систем, как правило являются собственностью одной компании, менее гетерогенны, в отличие от грид, и сдаются внаём пользователям по правилу «плати за то, что используешь». При этом пользователи информационных облаков получают абсолютно прозрачный доступ к среде и не имеют никакого представления об её архитектуре и промежуточном программном обеспечении (ППО), с помощью которого она построена. Использование грид подразумевает определённые знания о системе и ППО. Ключевой технологией, лежащей в основе обеих компьютерных парадигм является сервис-ориентированная архитектура (SOA). В настоящее время и облачные, и грид системы строятся на основных принципах SOA, с использованием соответствующего инструментария, описанного выше. Возможная распределённая конфигурация таких систем, реализованных на базе веб-сервисов, показана на рис. 1.

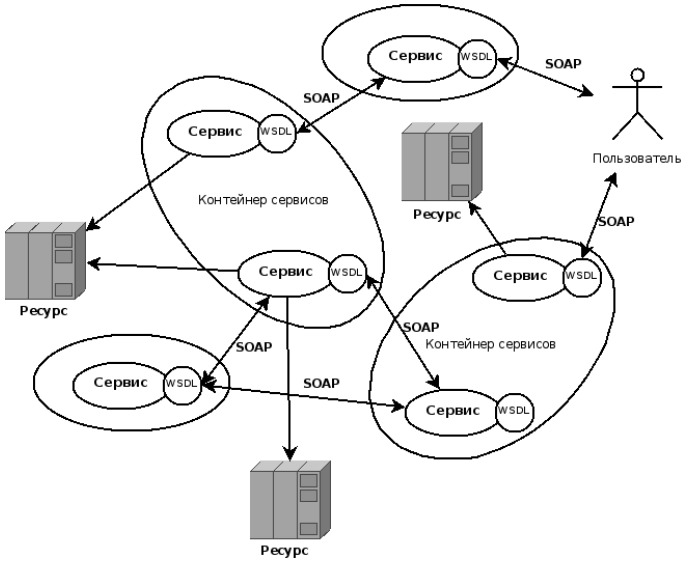

Рис. 1. Возможная конфигурация облачной или грид системы, построенной с применением технологии веб-сервисов

В свою очередь, конечным пользователям предоставляется специфичный набор сервисов: инфраструктура, платформа и приложение как сервис (*IaaS*, *PaaS*, *SaaS*) – в случае облачных технологий; сервисы контроля и мониторинга ресурсов, управления заданиями и данными и т.д. (например, *OGSA*) – в случае с грид технологиями. Традиционно грид приложения создаются с помощью давно разработанных технологий параллельного и распределённого программирования, однако в настоящее время все больше приложений для грид создаётся в виде сервисов. В случае облачных сред пользователь либо имеет возможность создавать собственную платформу разработки и разворачивания приложений (IaaS), либо пользуется платформой, предоставленной провайдером облачных услуг. Ключевыми проблемами являются: сложность получаемых программно-аппаратных систем и недостаточная масштабируемость – для грид, проблемы безопасности, доступности сервисов, зависимости от провайдера услуг – для облачных систем. Общей проблемой для обоих подходов является проблема мониторинга и сравнительного анализа выполнения распределённых приложений, которая в настоящее время очень актуальна [8].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Богданов А.В., Станкова Е.Н., Мареев В.В.* Сервис-ориентированная архитектура: новые возможности в свете развития Grid технологий // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы", 2008. – 32 с.
2. Сервис-ориентированная архитектура // Википедия. [2010-2010]. URL: http://ru.wikipedia.org/wiki/Сервис-ориентированная_архитектура (дата обращения: 26.05.2010).
3. XML Schema // Консорциум W3C. [2000-2007]. URL: http://www.w3.org/XML/Schema (дата обращения: 26.05.2010).
4. Web Services Description Language (WSDL) 1.1 // Консорциум W3C. [2001-2001]. URL: http://www.w3.org/TR/wsdl (дата обращения: 26.05.2010).
5. SOAP Tutorial // Refsnes Data. [1999-2010]. URL: http://www.w3schools.com/soap/ (дата обращения: 26.05.2010).
6. SOAP Version 1.2 Part 0: Primer (Second Edition) // Консорциум W3C. [2007-2007]. URL: http://www.w3.org/TR/soap/ (дата обращения: 26.05.2010).
7. *Ian Foster, Carl Kesselman, Steve Tuecke* The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal of Supercomputing Applications, 15(3): 2001. Pages: 200;
8. *Хашковский В.В.*, *Лутай В.Н.*, *Юрченко В.В.* Сравнительная оценка времени выполнения программ на различных платформах // Известия ЮФУ. Технические науки. – 2009. – № 12 (101). – С. 176-180.

Данилов Игорь Геннадьевич
Технологический институт федерального государственного автономного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: vainamon@hpcmop.ru.
347928, г. Таганрог, пер. Некрасовский, 44.
Тел.: 88634371673.

Danilov Igor Gennadievich
Taganrog Institute of Technology – Federal State-Owned Educational Establishment of Higher Vocational Education “Southern Federal University”
E-mail: vainamon@hpcmop.ru.
44, Nekrasovskiy, Taganrog, 347928, Russia.
Phone: +78634371673.