СЕРВИСНО-ОРИЕНТИРОВАННЫЕ ТЕХНОЛОГИИ РЕАЛИЗАЦИИ ИНФОРМАЦИОННЫХ СИСТЕМ

Сервисно-ориентированные архитектуры (COA) и Web-сервисы

- Сервисно-ориентированные архитектуры. Сервисно-ориентированная архитектура СОА (service-oriented architecture, SOA) — это подход к созданию ИС, основанный на использовании сервисов или служб (service). Далее сервис и служба рассматриваются как синонимы. СОА — это, в первую очередь, интеграционная архитектура, использование которой позволяет обеспечить гибкую интеграцию ИС. При использовании СОА приложения взаимодействуют, вызывая сервисы, входящие в состав других приложений. Сервисы объединяются в более крупные последовательности, реализуя бизнес-процессы, которые могут быть доступны как сервисы.
- СОА можно рассматривать так же как подход к построению слабосвязанных (loosely coupled) систем, реализующих механизмы асинхронного взаимодействия. К слабосвязанным системам обычно относятся такие системы, как электронная почта и системы очередей сообщений.

Переход на СОА-архитектуры позволяет решать следующие задачи:

- уменьшать сроки освоения и внедрения новых ИТ-систем, быстро создавать новые ИТ-системы на базе уже существующих;
- уменьшать суммарную стоимость владения ИТ-продуктом и стоимость их интеграции;
- увеличить срок жизни ИТ-систем за счет возможности их оперативной модернизации;
- использовать гибкие модели ценообразования путем передачи разработки детализированных бизнес-модулей сторонним производителям (аутсорсинг);
- уменьшать стоимости работ по интеграции, необходимой при слиянии и поглощении компаний;
- реализовывать бизнес-процессы на уровне, не зависящем от приложений и платформ для поддержки процессов

- СОА — это интеграционная архитектура, основанная на концепции сервисов (служб).
- Бизнес-функции и инфраструктурные функции, которые необходимы для построения распределенных систем, реализуются как сервисы, которые в сочетании или по отдельности обеспечивают прикладную функциональность либо другим сервисам, либо приложениям, взаимодействующим с конечным пользователем.

- Концепция СОА предполагает использование единого механизма взаимодействия служб. Этот механизм строится на основе концепции свободных связей и должен поддерживать использование формальных интерфейсов.
- СОА привносит преимущества, которые дают слабосвязанность и инкапсуляцию, в интеграцию на уровне предприятия.

- Сервисы можно рассматривать как строительные блоки, которые могут использоваться для построения как сервисов более высокого уровня, так и законченных распределенных ИТ-систем. Сервисы могут вызываться независимо внешними или внутренними потребителями для выполнения элементарных функций либо могут объединяться в цепочки для формирования более сложных функций и для быстрого создания новых функций.
- При использовании СОА организации могут создавать гибкие КИС, которые позволяют оперативно реализовывать быстро изменяющиеся бизнес-процессы и многократно использовать одни и те же компоненты в рамках одной ИТ-системы, в рамках семейств продуктов и в независимых ИТ-системах.

- Сервисом можно назвать любую дискретную функцию, которая может быть предложена внешнему потребителю. В качестве сервиса может выступать как отдельная бизнес-функция, так и набор функций, которые образуют бизнес-процесс.

- **Сервисы, ориентированные на использование в составе СОА, должны обладать следующими свойствами:**
- представлять собой многократно используемые бизнес-функции;
- определяться с помощью формальных, не зависящих от реализации интерфейсов;
- наличие протоколов связи, обеспечивающих прозрачность местонахождения и инвариантность по отношению к языку и платформе.

Замечание

- Хотя в качестве сервиса может выступать любая бизнес-функция, однако крайне желательно, чтобы имелась возможность повторного использования бизнес-функций одним или разными приложениями. Примером может служить система записи сообщений о событиях в log-файлы, которая присутствует практически в каждом приложении.

- Во время выполнения каждый сервис размещается в одном, и только в одном месте и удаленно вызывается всеми клиентами, которые используют данный сервис. Достоинство подобного подхода состоит в том, что изменения в интерфейс или реализацию сервиса, например изменение кода реализации или настроек, нужно вносить только в одном месте.

Web-сервисы

- В самом общем виде понятие Web-сервиса можно определить как сервис (услугу), которая предоставляется через WWW с использованием языка XML и протокола HTTP.
- Практически все ведущие ИТ-компании положительно относятся к использованию Web-сервисов, поэтому Web-сервисы можно использовать в качестве механизма интеграции приложений, реализованных на любых платформах. Существует много разных определений понятия Web-сервиса.

Определение, которое дается консорциумом W3C:

- Web-сервис представляет собой приложение, которое идентифицируется строкой URI. Интерфейсы и привязки данного приложения описываются и обнаруживаются с использованием XML-средств. Приложения взаимодействуют посредством обмена сообщениями, которые пересылаются с использованием интернет-протоколов

- Web-сервис — это новая парадигма реализации сервисов через Web. Иногда говорят о Web-сервисе как об атрибуте Web третьего поколения.
- При этом к первому поколению относят статический Web, использующий преимущественно статический HTML, а ко второму — интерактивный Web, использующий такие технологии, как PERL, ASP, JSP, и компоненты, используемые для построения распределенных приложений, которые работают по принципу черного ящика.

- Web-сервис в полной мере и практически без всяких ограничений поддерживает кроссплатформенность, независимость от языка программирования, хорошо работает через файерволы.
- Для обмена данными Web-сервис использует такие протоколы как XML, HTTP, TCP/IP, т.е. протоколы, которые поддерживаются практически всеми производителями.

- Web-сервис оказал существенное влияние на подходы к построению распределенных систем, поскольку они обладают такими ценными свойствами, как самоописываемость, их легко создавать размещать, регистрировать и находить в репозитариях. Для того чтобы работать с сервисами, достаточно знать их интерфейсы. Унаследованные приложения могут быть относительно легко оформлены в виде Web-сервисов.

- Уже в начале XXI в. появились тысячи общедоступных сервисов. Большое число производителей, включая всех основных производителей ИТ-систем, предлагают инструментальные средства для разработки Web-сервисов. К наиболее успешно используемым можно отнести следующие инструментальные средства для создания Web-сервис: Apache Tomcat Next Generation Web service — AXIS, Microsoft.NET Web service studio based on IIS server, IBM Web Sphere Web service, BEA Web Logic Workshop, Java Web Service Development Pack (JWSDP), Mind Electric GLUE и многие другие. Значительное количество из них — свободно распространяемые. Web-сервисы, в частности, можно создавать с помощью таких средств разработки, как NetBeans и Eclipse.

- Web-сервисы представляют собой самостоятельные модульные приложения, которые могут быть описаны, опубликованы, размещены и вызваны как локально, так и удаленно. Web-сервисы могут инкапсулировать как простейшие бизнес-функции типа «запрос-ответ», до полномасштабных взаимодействий бизнес-процессов. Службы могут создаваться заново или строиться на основе существующих приложений методом «обертывания»

Свойства Web-сервисов

- Все Web-сервисы являются самодостаточными, т.е. с клиентской стороны не требуется никакого дополнительного программного обеспечения кроме языка программирования, поддерживающего работу с XML и HTTP, а на серверной стороне требуется только HTTP-сервер, поддерживающий работу с посланиями;
- являются самоописываемыми, поскольку метаданные передаются вместе с сообщением и не требуется никаких внешних хранилищ метаданных

- могут быть опубликованы, обнаружены и вызваны через Интернет. Причем для этого используют простые установившиеся стандарты, такие как HTTP и существующую сетевую инфраструктуру;
- являются модульными, т. е. простые Web-сервисы могут объединяться в более сложные, причем это может быть сделано разными способами, например, с использованием рабочих процессов или через прямой вызов Web-сервисов из других бизнес-процессов;
- инвариантны к способу реализации, т.е. клиент и сервер могут быть реализованы в разных средах с использованием разных языков программирования, причем для клиента не имеет значения, на какой платформе реализован сервер, и наоборот;

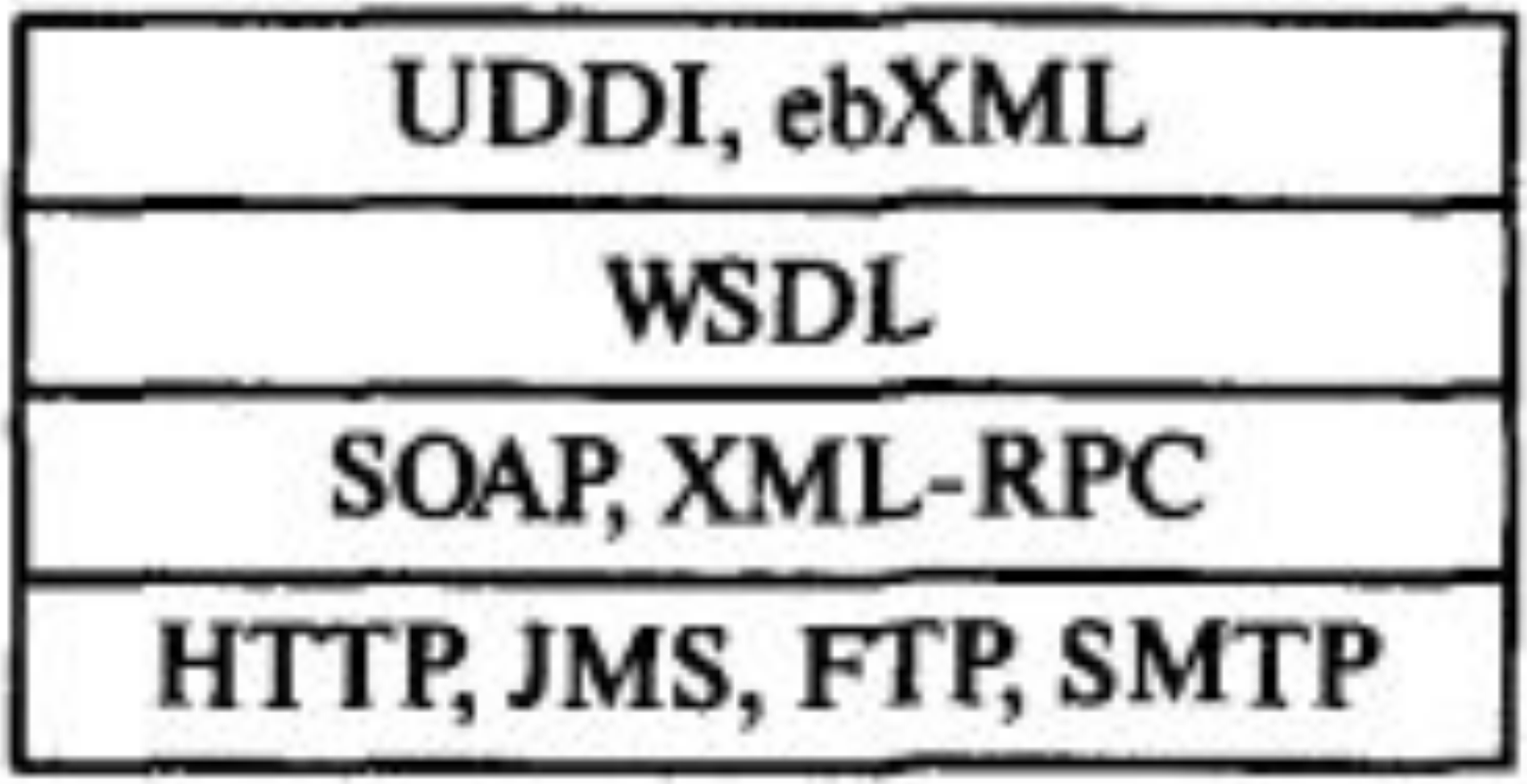
- открыты и основаны на стандартах, технической основой Web-сервисов являются XML и HTTP, значительная часть технологии Web-сервисов создана с использованием проектов с открытым исходным кодом;
- имеют свободные связи, по сравнению с другими, в частности, компонентными технологиями для Web-сервисов требуется более простой уровень координации, который позволяет осуществлять достаточно гибкую реконфигурацию для обеспечения интеграции нужных сервисов;
- являются динамическими, поскольку имеется возможность обнаруживать службы в процессе функционирования, при этом имеется возможность их модификации, которая не влияет на работу клиентов;
- являются действенным средством интеграции унаследованных приложений.

- Различие между СОА и Web-сервисами состоит в том, что СОА представляет собой общую архитектуру интеграции приложений, а Web-сервисы — один из методов реализации СОА. Между Web- сервисами и СОА существует много логических связей. Web-сервисы представляют собой модель на основе открытых стандартов, которая может быть обработана автоматически и позволяет создавать формальные описания, не зависящие от интерфейсов служб. Web-сер- висы предоставляют механизмы коммуникации, обеспечивающие прозрачность местоположения и возможность взаимодействия

- Наиболее важными компонентами архитектуры Web-сервисом являются провайдер сервиса (север) и пользователь сервиса (клиент)
- *Провайдер* должен опубликовать (зарегистрировать сервис в репозитарии, который может быть реализован, в частности, как UDDI-реестр (Universal Description, Discovery, and Integration (UDDI) registry). UDDI-реестр внешне выглядит как Web-сервис и в нем хранится информация о зарегистрированных сервисах.

- ***Клиент*** может обращаться к UDDI-реестру с запросами о месте нахождения отдельных сервисов и способах обращения к ним. Следует отметить, что для обращения к Web-сервисам клиент не обязательно должен предварительно обращаться к тому или иному репозитарию. Если клиенту известны местонахождение сервиса и его интерфейс, он может обращаться к сервису напрямую. Множество протоколов, используемых для работы с Web-сервисами, составляют стек

Стек протоколов, используемых для работы с Web-сервисами



- На нижнем уровне находятся транспортные протоколы, отвечающие за транспортировку сообщений (HTTP, JMS, FTP, SMTP). Протоколы, определяющие форматы сообщений (SOAP, XML-RPC), располагаются на следующем уровне. На третьем уровне располагается протокол WSDL (Web Services Description Language, язык описания Web-служб). На четвертом уровне находятся протоколы, определяющие механизмы обнаружения Web-сервисов: UDDI, ebXML. Кроме перечисленных, имеется большое число вспомогательных протоколов, которые часто называют WS*

Язык XML при работе с Web-сервисами

- **Язык XML** (extensible Markup Language) является основным при работе с Web-сервисами.
- XML можно рассматривать как надмножество HTML. В XML пользователь может вводить и использовать собственные теги.
- Любой XML-документ регламентируется метаданными, которые находятся в специальном файле.

- Примеры метаданных: можно указать, что в году ровно 12 месяцев, в месяце может быть от 28 дней до 31 дня, в почтовом индексе должно быть ровно 6 цифр и т.д.
- Для представления метаданных используется либо Document Type Definition (DTD), либо XSD файла. DTD является устаревшим. В 2001 г. консорциум W3C (WWW Consortium) рекомендовал описывать структуру документов XML на языке описания схем XSD.

- Основным достоинством XML-файлов является то, что имеется возможность их машинной обработки. Практически все современные языки программирования имеют средства для работы с XML. XML используется в качестве универсального формата обмена данными, в качестве средства для хранения данных (XML базы данных), а также в разного рода конфигурационных файлах и файлах размещения (deployment descriptors)

простейший XML-документ, содержащий записи о студентах и их рейтинге (среднем балле):

```
<?xml version="1.0" encoding="Windows-1251"?>
<students>
        <student id=123405>
            <name>Иванова Ю.А.</name>
            <rating>4.5</rating>
        </student>
            <student id=123410>
            <name>Соколов М.М.</name>
        <rating>4.3</rating>
</student>
    </students>
```

- Документ XML начинается с пролога, в котором указывается версия языка XML, способ кодировки (по умолчанию — UTF-8) и ряд других элементов.
- Все элементы XML-документа находятся внутри корневого элемента (root element), в данном случае — это элемент <students>. Имя корневого элемента должно совпадать с именем документа. Внутри корневого элемента имеются два вложенных элемента типа <student>. Каждый студент имеет уникальный идентификатор (id), в качестве которого может выступать, например, номер группы или номер студента по списку. В рассматриваемом примере идентификатор помещен в качестве атрибута в открывающийся тег. Внутри каждого элемента типа <student> имеется два вложенных элемента: <name> и <rating>. В первом указаны фамилия и инициалы, а во втором — средний балл.

XSD Schema выглядит следующим образом:

```
<?xml version="1.0" encoding=" Windows-1251"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
            targetNamespace=http://www.myuniver.ru"
            xmlns="http://www.myuniver.ru">
      <xsd:element name = "students">
```

```
            <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="student"
                minOccurs="1"
                maxOccurs="unbounded"/>
            </xsd:sequence/>
            </xsd:complexType/>
</xsd:element>
     <xsd:element name="student">
     <xsd:complexType>
     <xsd:sequence>
     <xsd:element ref="name"/>
     <xsd:element ref="rating"/>
     </xsd:sequence>
     </xsd:complexType>
     </xsd:element>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="rating" type="xsd:float"/>
</xsd:schema>
```

- Использование пространств имен (namespace) является средством предотвращения коллизии имен. Для этого имена тегов и атрибутов снабжают префиксом, который отделяется от имени двоеточием.
- Префикс имени связывается с идентификатором, определяющим пространство имен. Все имена тегов и атрибутов, префиксы которых связаны с одним и тем же идентификатором, образуют одно пространство имен. Идентификатор пространства имен имеет форму URI, причем такой адрес, как http://www.myuniver.ru/2011/mns, может не соответствовать никакому реальному адресу, а представлять собой просто уникальный идентификатор. Каждый пользователь может создавать любое количество собственных пространств имен и пользоваться общедоступными пространствами имен. В рассматриваемом примере в качестве разделяемого пространства имен выступает xsd:schema, которому соответствует пространство имен xmlns:xsd=http://www. w3.org/2001/XMLSchema.
- В данном пространстве имен определены теги, относящиеся к XSD Schema.
- Появление имени тега без префикса в документе, использующем пространство имен, означает, что имя принадлежит пространству имен по умолчанию (default namespace).

В рамках схемы можно определять новые типы данных, например:

```
<element name = "student"
<complexType>
      <element name = "name" type = "xsd: string" />
      <element name = "rating" type = "xsd: float" />


    </complexType>
</element>
```

Протокол XML-RPC. Непосредственным предшественником Web-сервисов являлся протокол XML-RPC.

- Это очень простой протокол, предназначенный для вызова удаленных процедур. В отличие от традиционного RPC для вызова удаленной процедуры используются XML-послания. Ответ приходит также в форме XML-послания.
- Рассмотрим простейший пример.
- Пусть имеется удаленная процедура, осуществляющая поиск синонимов в словаре. Для обращения к процедуре используется строка вида public String getSynonym (String word). В качестве аргумента используется исходное слово (word). Процедура возвращает слово — синоним исходного.

Запрос в рассматриваемом случае выглядит следующим образом:

```
<?xml version="1.0"?>
<methodCall>
      <methodName>getSynonym</methodName>
      <params>
            <param>
                  <value>
                        <string>самолет</string>
                  </value>
            </param>
      </params>
</methodCall>
```

- Корневым является элемент <methodCall>, в который вложен элемент <methodName>, в теле которого записывается имя вызываемой процедуры. Параметры вызываемой процедуры помещаются в элемент <params>. В элемент <params> вложены нуль или несколько элементов <pagam>, в которые помещаются параметры вызываемой процедуры.
- В рассматриваемом примере требуется получить синоним слова «самолет».

Ответ также поступает в форме XML-сообщения:

```
<?xml version="1.0"?>
<methodResponse>
      <params>
            <param>
                  <value>аэроплан</value>
            </param>
      </params>
</methodResponse>
```

- Если произошла ошибка при выполнении запроса, то в ответном пакете указывается код ошибки. XML-RPC позволяет использовать такие типы данных как целые строки, вещественные, структуры, массивы.
- Механизм XML-RPC был создан в середине 1990-х гг., однако не получил широкого распространения. Причины этого достаточно понятны и состоят в следующем:

- запросы, выполняемые в рамках XML-RPC, только отчасти являются самоописываемыми, поскольку пространства имен в нем не определены;
- XML-RPC подобно классическому RPC не поддерживают работу с объектами;
- при работе с XML-RPC обнаруживаются проблемы, вызванные использованием XML. На стороне клиента необходимо сформировать запрос. На стороне сервера необходимо его разобрать. То же самое требуется сделать с ответом. Все это усложняет код и требует существенных временных затрат;
- сообщения, отправляемые в формате XML, за счет тегов имеют большую избыточность.