Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Бакалавърска програма
„Софтуерно инженерство”

Предмет: XML технологии за семантичен Уеб

Зимен семестър, 2020/2021 год.

Тема 35: Преглед на езика WSDL

Есе

Автори:

Симона Любенова, фак. номер 62258

Ати Мускова, фак. номер 62308

ноември, 2020

София

Съдържание

1 Въведение

WSDL е XML базиран език, който се използва за описание на уеб услуги. Той дефинира XML граматика за описване на мрежови услуги като колекции от комуникационни крайни точки, които са способни да обменят съобщения. С помощта на WSDL могат да се описват отделни услуги, поддържаните от тях процедури, както и приеманите и връщаните от тях съобщения. Той е една от основните концепции за уеб услуги. Езикът WSDL е неразделна част от Universal Description, Discovery and Integration (UDDI), като е базиран на XML световен бизнес регистър.

Идеята за UDDI се поддържа основно от Microsoft, IBM, Arabia. UDDI има два раздела – регистър на всички метаданни на уеб услуга и набор от дефиниции на типа WSDL порт за манипулиране и търсене в този регистър. Целта е компютрите да могат сами да намерят услугите, които са им необходими, както и да намерят точни инструкции за начина, по който могат да комуникират с услугата.

2 Характеристики и използване на езика

WSDL често се използва в комбинация със SOAP и XML схема за предоставяне на уеб услуги през Интернет. Свързваща се с уеб услуга клиентска програма може да чете WSDL, за да определи какви функции са налични на сървъра. Всички използвани специални типове данни са вградени в WSDL файла под формата на XML схема. След това клиентът може да използва SOAP, за да извика някоя от функциите, които са изброени в езика WSDL.

2.1 Дефиниции

Основните елементи, които съдържа езикът WSDL, са:

- **Definitions –** Това е основният елемент на всички WSDL документи. Той дефинира името на уеб услугата, декларира множество пространства от имена, които са използвани в останалата част от документа, и съдържа всички описани елементи на услугата
- **Data types –** Контейнер за дефиниции на абстрактни типове, които се използват в съобщенията под формата на XML схеми
- **Message –** Определение на данните под формата на съобщение, което е представено или като цял документ, или като аргументи, които трябва да бъдат съпоставени с извикване на метод
- **Operation –** Определение на операцията за съобщение като именуване на метод, опашка от съобщения или бизнес-процес, което ще приеме и обработи съобщението
- **Port type –** Набор от операции, съпоставени с една или повече крайни точки, които дефинират събирането на операции за обвързване
- **Binding (обвързване) –** Конкретните формати на протокол и данни за операциите и съобщенията, които са дефинирани за определен тип порт

- **Port** – Комбинация от обвързващ и мрежов адрес, който осигурява целевия адрес на комуникацията на услугата
- **Service** – Колекция от свързани крайни точки, които обхващат дефинициите на услугата във файла

Към тези основни елементи спецификацията WSDL дефинира и помощни елементи:

- **Documentation** – Този елемент се използва за осигуряване на разбираема документация и може да бъде включен във всеки друг елемент на WSDL
- **Import** – Използва се за добавяне на други WSDL документи или XML схеми

Пример:

```
<definitions name = "HelloService"
   targetNamespace =
"http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name = "Hello_Binding" type = "tns:Hello_PortType">
      <soap:binding style = "rpc"
         transport = "http://schemas.xmlsoap.org/soap/http"/>
      <operation name = "sayHello">
         <soap:operation soapAction = "sayHello"/>
         <input>
            <soap:body
               encodingStyle =
"http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </input>

         <output>
```

```
            <soap:body
               encodingStyle =
"http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </output>
      </operation>
   </binding>

   <service name = "Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding = "tns:Hello_Binding" name = "Hello_Port">
         <soap:address
            location = "http://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

- **Definitions –** HelloService
- **Type –** Използване на вградени типове данни, дефинирани в XML схема
- **Message:**
  - sayHelloRequest − firstName parameter
  - sayHelloresponse − greeting return value
- **Port Type –** sayHello операция, която се състои от заявка и услуга за отговор
- **Binding –** Указание за използване на транспортния протокол SOAP HTTP
- **Service –** Услуга, достъпна на адрес http://www.examples.com/SayHello/
- **Port –** Свързва обвързването с URI http://www.examples.com/SayHello/ , където може да се осъществи достъп до работещата услуга

2.2 Основни характеристики

Езикът WSDL съдържа четири примитива за предаване, които крайната точка може да поддържа. Това са:

- **One-way (еднопосочен):** Крайната точка получава съобщение
- **Request-response (заявка-отговор):** Крайната точка получава съобщение и изпраща отговор
- **Solicit-response (искане-отговор):** Крайната точка изпраща съобщение и получава отговор
- **Notification (известие):** Крайната точка изпраща съобщение

Езикът WSDL дефинира обвързване към формати на съобщения и протоколи:

- Крайни точки, които са дефинирани чрез обвързващ конкретен мрежов протокол и формат на съобщението до абстрактни операции и съобщения

- Може да опише всяка крайна точка, независимо от основния мрежов протокол и формат на съобщението
- Определя как да се намери крайната точка за дадена услуга

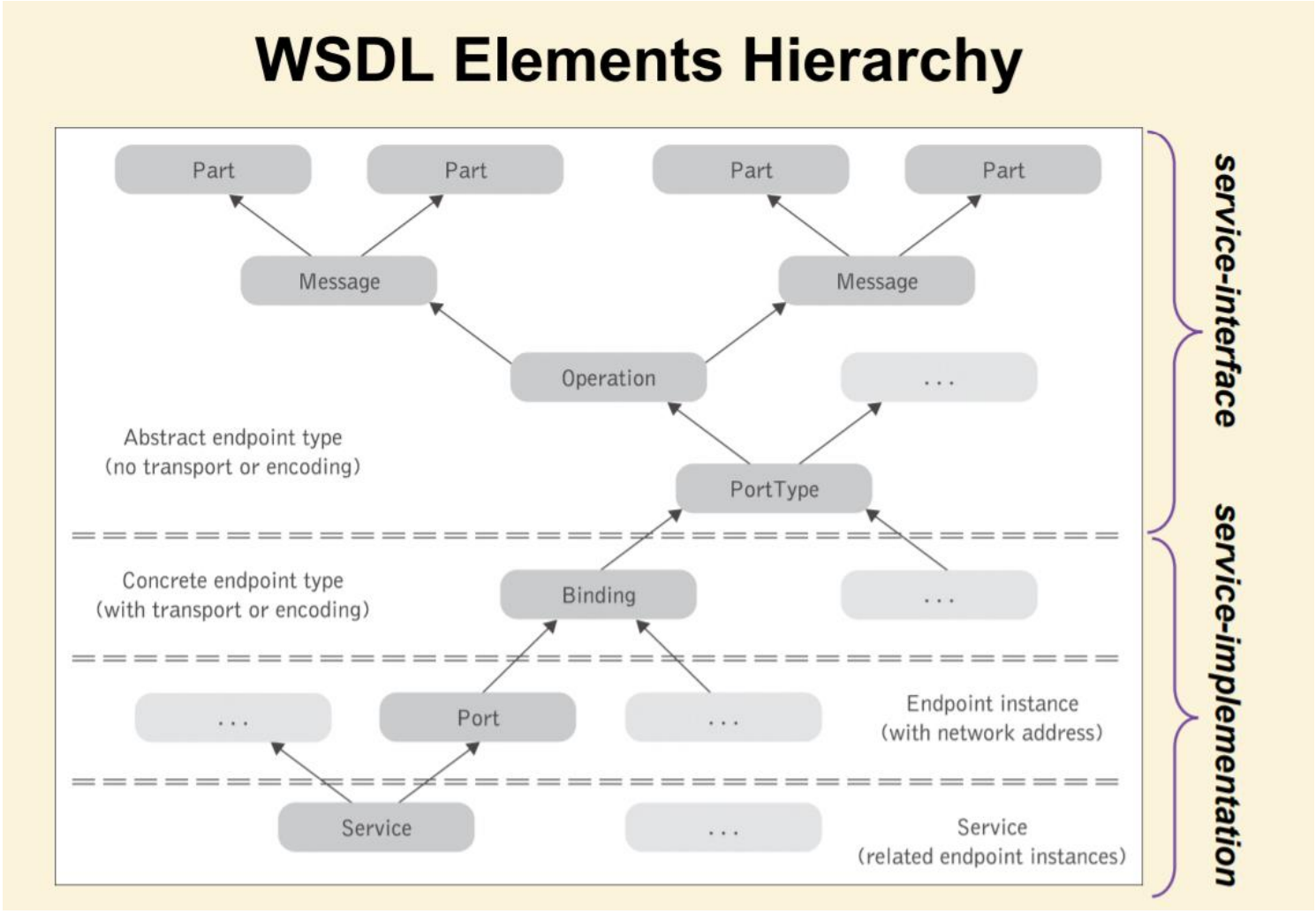
Един WSDL документ може да бъде разделен на „абстрактни“ и „конкретни“ части, които по конвенция често се дефинират в два или повече файла.

Абстрактните определения са:

- <types> – дефиниции на типа данни
- <message> – параметри за операция
- <operation> – абстрактно описание на действия за услуги
- <portType> – набор от дефиниции на операцията

Конкретните определения са:

- <binding> – оперативни обвързвания
- <port> – асоцииране на крайна точка със свързване
- <service> – местоположение за всяко свързване



2.2.1 <types>

Елементът <types> включва дефиниции на типа данни, които са от значение за обменяните съобщения. За максимална оперативна съвместимост и неутралност на платформата WSDL предпочита използването на XSD като каниочна тип система и я третира като вътрешна система.

```
<definitions .... >
<types>
<xsd:schema .... />*
</types>
</definitions>
```

2.2.2 <message>

Елементът <message> описва полезния товар на съобщение, използвано от мрежата обслужване. Съобщението се състои от елементи <part>, които са свързани с <types> елементи.

```
<!-- message elements that describe input and output parameters for the
    PurchaseOrderService -->
<!--input message  -->
<wsdl:message name="POMessage">
      <wsdl:part name="PurchaseOrder" type="tns:POType"/>
      <wsdl:part name="CustomerInfo" type="tns:CustomerInfoType"/>
</wsdl:message>
<! -- outputput message  -->
<wsdl:message name="InvMessage">
      <wsdl:part name="Invoice" type="tns:InvoiceType"/>
</wsdl:message>
```

RPC-style message

```
<!-- message element that describes input and output parameters -->
<wsdl:message name="POMessage">
      <wsdl:part name="PurchaseOrder" element="tns:PurchaseOrder"/>
</wsdl:message>
```

Document-style message

Услугата PurchaseOrder дефинира два елемента <message>, описващи параметрите и върнатите стойности на тази услуга.

- POMessage описва входните параметри на услугата
- InvMessage представлява параметрите за връщане (изход)

2.2.3 <portType>, <operation> елементи

Елементът <portType> определя абстрактен тип и неговите операции, но не е изпълнение. Той е логическо групиране на <operation> в уеб услуга. Описва видовете операции, които уеб услугата поддържа – режим на съобщения и полезни товари, без да се посочва интернет протокол или използван физически адрес. Елементът <portType> е основен за WSDL описание. Останалата част от елементите в дефиницията са по същество подробности.

Елементът <operation> представлява методите, изложени от услугата – включват името на метода, подадените и изведените параметри. Типичен елемент <operation> се състои от най-много един <input> или <output> елемент и произволен брой елементи по подразбиране.

2.2.4 <binding>, <port>, <service> елементи

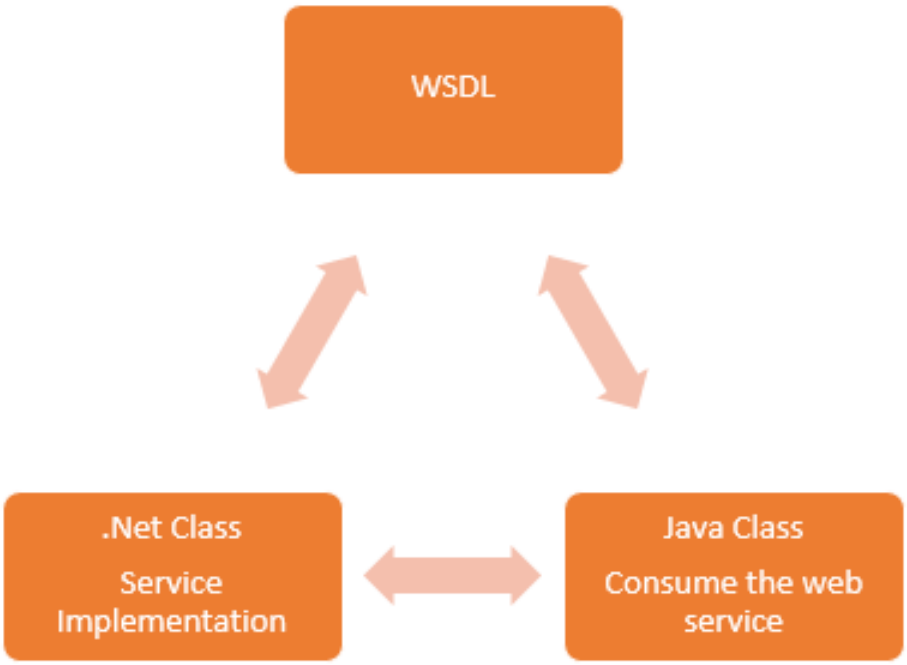
Централният елемент на описанието на изпълнението е <binding> елемент. Този елемент указва как трябва да бъде клиентът и уеб услугата обмен на съобщения. Клиентът използва тази информация за достъп до мрежата обслужване. Елементът <binding> съдържа информация за това, как елементите от абстрактна услуга интерфейс (<portType> елемент) се превръщат в конкретно преобразуване на определена комбинация от конкретни протоколи като например SOAP или HTTP.

Елементът <port> определя местоположението на уеб услуга и ние можем да го възприемем като URL, където можем да намерим определената услуга. Елементът <port> свързва крайна точка – например местоположение на мрежов адрес или URL адрес със специфичен WSDL <binding> елемент. Възможно е два или повече елемента <port> да присвоят различни URL адреси на един и същ <binding> елемент. Това може да е полезно за балансиране на натоварването или за неуспешни цели.

Елементът <service> съдържа колекция (обикновено една) от WSDL <port> елементи. Всеки елемент <service> е именуван и всяко име трябва да е уникално сред всички услуги в WSDL документ.

2.3 Въведение в използването на езика

Един WSDL файл е написан в обикновен XML. Причината за това е да може файлът да се чете от всеки език за програмиране. Ако клиентско приложение е написано на .Net, то ще разбере XML файла. По същия начин, ако клиентското приложение е написано на езика за програмиране Java, то също ще може да интерпретира WSDL файла.



Файлът WSDL е това, което свързва всичко заедно. От диаграмата се забелязва, че може да се създаде уеб услуга на езика .Net. Ето тук услугата се прилага. Ако не разполагаме с файла WSDL и искаме клас Java да използва уеб услугата, ще са ни необходими много усилия, с които да постигнем тази цел. Но сега файлът WSDL, който е в XML, може да бъде разбран от всеки език за програмиране и може лесно да се накара Java клас да консумира услугата .Net. По този начин усилията за писане на код са значително намалени.

WSDL файловете са от основно значение за тестване на SOAP базирани услуги. SoapUI използва WSDL файлове, за да генерира тестови заявки, твърдения и фалшиви услуги. WSDL файловете дефинират различни аспекти на SOAP съобщенията:

- Дали някой елемент или атрибут е разрешен да се появява няколко пъти
- Необходимите или незадължителни елементи и атрибути
- Конкретен ред на елементите, ако е необходим

Файлът WSDL се създава, когато уеб услугата е изградена, на който и да е език за програмиране. Тъй като той е доста сложен за генериране, всички редактори като Visual Studio за .Net и Eclipse за Java автоматично създават файл WSDL.

Пример за създаден WSDL файл във Visual Studio:

```
<?xml version="1.0"?>
<definitions name="Tutorial"
                targetNamespace=http://Guru99.com/Tutorial.wsdl
        xmlns:tns=http://Guru99.com/Tutorial.wsdl
        xmlns:xsd1=http://Guru99.com/Tutorial.xsd
        xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
        xmlns="http://schemas.xmlsoap.org/wsdl/">
   <types>
                <schema targetNamespace=http://Guru99.com/Tutorial.xsd
        xmlns="http://www.w3.org/2000/10/XMLSchema">

        <element name="TutorialNameRequest">
        <complexType>
                <all>
                <element name="TutorialName" type="string"/>
               </all>
            </complexType>
       </element>
       <element name="TutorialIDRequest">
                <complexType>
                <all>
                <element name="TutorialID" type="number"/>
               </all>
            </complexType>
       </element>
```

```
   </schema>
 </types>
 <message name="GetTutorialNameInput">
        <part name="body" element="xsd1:TutorialIDRequest"/>
 </message>
 <message name="GetTutorialNameOutput">
        <part name="body" element="xsd1:TutorialNameRequest"/>
 </message>
 <portType name="TutorialPortType">
        <operation name="GetTutorialName">
        <input message="tns:GetTutorialNameInput"/>
        <output message="tns:GetTutorialNameOutput"/>
   </operation>
  </portType>
  <binding name="TutorialSoapBinding" type="tns:TutorialPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
         <operation name="GetTutorialName">
                <soap:operation soapAction="http://Guru99.com/GetTutorialName"/>
        <input>
                <soap:body use="literal"/>
           </input>
        <output>
   <soap:body use="literal"/>
 </output>
 </operation>
 </binding>

 <service name="TutorialService">
        <documentation>TutorialService</documentation>
    <port name="TutorialPort" binding="tns:TutorialSoapBinding">
        <soap:address location="http://Guru99.com/Tutorial"/>
    </port>
 </service>
</definitions>
```

2.4 Ограничения при използването на езика

Повечето WSDL файлове се приемат по време на потребление, но някои файлове могат да причинят проблеми:

- Не се поддържат кодирани със SOAP масиви и структури. Това означава, че RPC-кодираните и кодираните с документи уеб услуги със сложни входни или изходни параметри не работят – например Amazon Web Services API и Google Web Services API попадат в тази категория

- Всички операции трябва да бъдат от един вид – тоест всички RPC-кодирани или всички doc/literal
- Разглеждат се само SOAP операции – MIME и HTTP операциите се игнорират
- Не са разрешени претоварени операции
- Еднопосочните операции не са разрешени
- WSDL файл не може да има едновременно <wsdl:include> и <wsdl:types>, но може да се използва <xsd:include> вътре в <wsdl:types>. Няма ограничение за броя на елементите <xsd:include>, които могат да се използват
- WSDL файл не може да има повече от един елемент <wsdl:include>

3 Сравнителен анализ

Съществуват много и различни технологии, които се използват за създаването понякога на един и същ продукт. Всяка технология се различава от останалите по нейните основни приоритети при изпълнение и функции, като всяка допринася за подобряването на крайния продукт. Решихме да направим сравнителен анализ между WSDL и SOAP, както и WSDL и WADL. Изборът ни на сравнение с точно този протокол и език е мотивиран от факта, че според повечето хора те са или едно и също нещо, или са взаимно заменяеми, без това да променя по някакъв начин структурата на продукта.

3.1 Критерии за сравнение

Критериите по които ще сравняваме WSDL и SOAP са:

- Дефиниция – искаме първо да представим основните разлики между двете технологии.
- Архитектура – определяща слоевете на продукта, както и по-нататъшното му развитие
- Леснота (Simplicity) при използване на двете технологии – представя ясно основните разлики при използването на двете технологии, както и техните предимства в различните ситуации
- Неутралност – тоест дали промяна на технологиите, с които се работи, оказват влияние върху работата им
- Производителност – максималната достижима производителност при използването на двете технологии
- Начин на действие – какви комуникационни канали използват двете технологии

- Независимост – дали и до каква степен двете технологии зависят от версиите и моделите на останалите програми
- Поддръжка – при евентуален проблем колко лесно може да получим помощ

Критериите, по които ще сравняваме WADL vs WSDL

- Дефиниция – сравнение относно основна цел на работа
- Предназначение – сравнение относно използването им
- Начин на работа – какво е необходимо даден клиент да направи
- Работа на ниско ниво – на какво машинно ниво работят
- Леснота на работа – сложността при първоначално запознаване
- Обвързаност с други технологии – дали зависят от други технологии
- Сложност относно дизайн
- Начин на оторизация – как точно се извършва при всяка
- Използване на шаблони – механизми на работа с шаблони
- Обхват на сравнение – коя от двете е по-обхватна
- Препоръчани интерфейси – кои са по- подходящи
- Поддръжка на външни технологии – с кои технологии работят заедно
- Удовлетворение на потребителски очаквания – оценка на потребители
- Предоставени допълнителни ресурси
- Генериране

3.2 Сравнение със SOAP

Таблица 1: Сравнение на WSDL със SOAP

Критерии	**WSDL**	**SOAP**
Дефиниция	WSDL е в основата си базиран на XML език за дефиниране на интерфейс на различни функционалности на уеб услуги	SOAP е основно XML-базирана спецификация на протокола за съобщения, която се използва за обмен на различна и структурирана информация при внедряването на уеб услуги в компютърни мрежи
Архитектура	WSDL архитектурата има три основни слоя/елемента: Types, Binding, Operations	SOAP има четири слоя архитектура: Header, Body, Envelope, Fault
Леснота	В случай на WSDL, той обработва различни сложни ситуации и запитвания, за да създаде правилните изходи на ниво машина. И така, това е основно усъвършенствана версия на кодирането и други различни подходи. По този начин той е много по-сложен от SOAP	В случай на SOAP, кодирането е началната стъпка от програмирането на сложни заявки и е по-лесно от програмирането.
Неутралност	В случай на WSDL, той не осигурява поддръжка на повечето протоколи като SOAP	В случай на SOAP, той осигурява поддръжка и напълно работещ за повечето протоколи като HTTP, JSM, SMTP и т.н.

Производителност	В случай на WSDL, той осигурява малко по-бърза комуникация и изпълнение, отколкото SOAP интерфазата и комуникацията на уеб сървъри.	От гледна точка на производителността, SOAP може да бъде малко по-бавен от WSDL поради фундаменталното актуализиране и големите процеси
Начин на действие	В случай на WSDL, той се комуникира директно чрез уеб сървъри и по този начин процесът на свързване не е толкова плавен, колкото в SOAP.	В случай на SOAP, съществуващата защитна стена и прокситата могат лесно да бъдат свързани поради по-лесната транзакция и комуникация през HTTP сървъри.
Независимост	Това не е случаят с WSDL и има зависимост за различни модели на програмиране.	В случай на SOAP, той осигурява поддръжка за всички или повечето от програмните модели.
Поддръжка	WSDL също предоставя широка гама от обществена и платена подкрепа. Обикновено всички версии на WSDL използват за осигуряване на дългосрочна поддръжка на клиенти.	Има и голяма подкрепа от общността за SOAP и неговите потребители.

3.3 Сравнение с WADL

Таблица 2: Сравнение на WSDL с WADL

Критерии	**WSDL**	**WADL**
Дефиниция	Език за описание на уеб услугата	Език за описание на уеб приложението
Предназначение	XML за описание на базирани на SOAP уеб услуги	XML за описание на RESTful уеб услуги
Начин на работа	Клиентът може да зареди WSDL файл и да се запознае с методите, които уеб услугата може да извика, какви са аргументите, които методът очаква и кой тип данни връща	Клиентът може да зареди WADL файл и да има достъп до пълната функционалност на уеб услугата.
Работа на ниско ниво	Машиночетимо описание с текущата версия 2.0	WADL е REST еквивалент на езика за описание на уеб услугата на SOAP
Лесната на работа	В сравнение с WADL, WSDL е труден за писане и разбиране	WADL е лек, лесен за разбиране и писане
Обвързаност с други технологии	Той има повече обвързване със SMTP услугите	Той няма обвързване със SMTP сървъри
Сложност относно дизайн	Сложен по дизайн	Прост по дизайн
Удостоверяване	Удостоверяването е включено за уеб услуги	Не е включено удостоверяване за уеб услуги
Използване на шаблони	Потребителят трябва да дефинира XML входното съобщение, за да използва механизма на URI Templete	Той изисква прост механизъм на URI шаблон

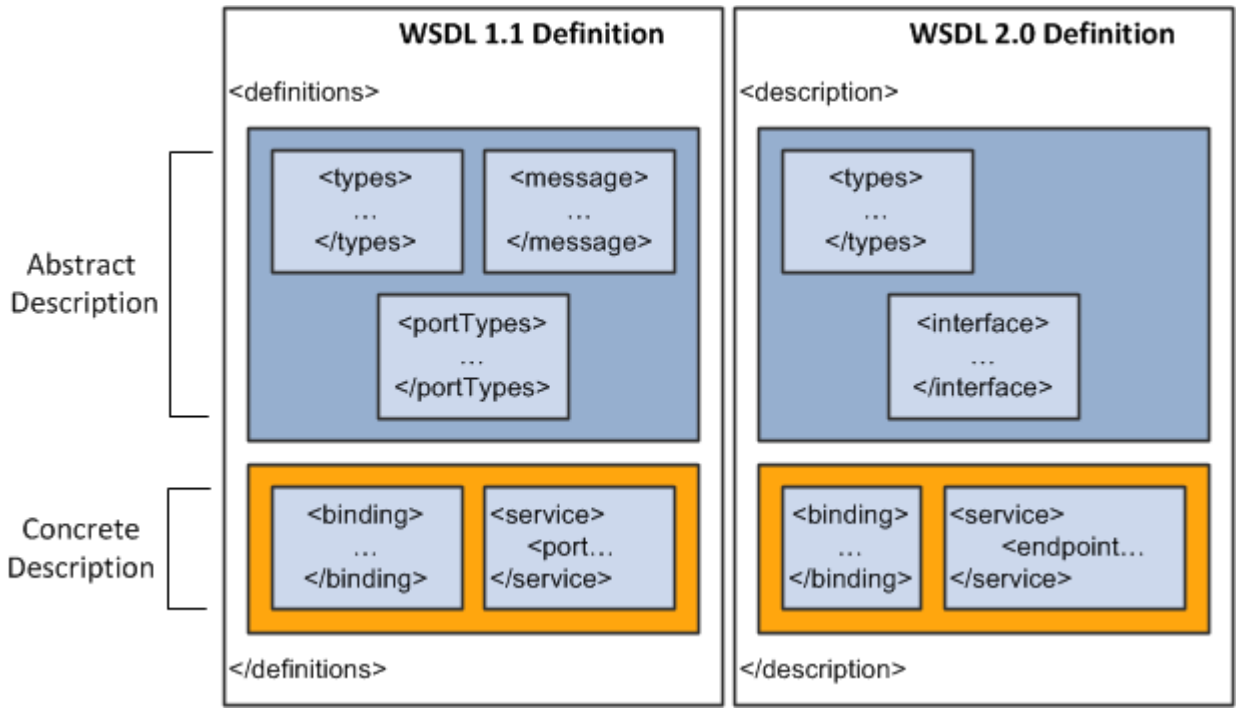
Обхват на сравнение	Той има по-голям обхват в сравнение с WADL	Той има ограничен обхват в сравнение с WSDL
Препоръчани интерфейси	W3C препоръчва WSDL	W3C не препоръчва WADL интерфейс
Поддръжка на външни технологии	WSDL 1.1 поддържа само GET и POST методи, докато WSDL 2.0 също поддържа HTTP протоколи	WADL стриктно поддържа REST услуга, само HTTP протоколи
Удовлетворение на потребителски очаквания	WSDL е по-гъвкав от WADL	WADL покрива всички очаквания на потребителите с изключение на Authentication, което е толкова просто, колкото REST
Предоставени допълнителни ресурси	WSDL дефинира всички основни елементи, които могат да бъдат дефинирани 0 или повече пъти с изключение на <types>. Не е необходимо всички дефиниции да бъдат в един и същ XML, един XML може да подобри други. Оттук идва концепцията за абстрактни и конкретни части	WADL дефинира както ресурси, така и представителство, както и методи за манипулиране на ресурси
Генериране	WSDL се генерира автоматично чрез? WSDL заедно с URL адреса на услугата като уеб референция в проекта, който генерира клиенти / заглушки от вас.	Представянето на WADL се състои от две функции за метода POST и една за метода GET, към този метод са прикрепени XSD файлове, за да се опише XML представяне на входни и изходни обекти и да се генерира WADL формат на REST услугата URI може да се използва

4 Примери на използване

4.1 Пример 1

Интеграционният брокер може да предоставя WSDL документи във формат WSDL 1.1 или WSDL 2.0. По подразбиране съветникът за предоставяне на уеб услуги, използван за предоставяне на WSDL документи, генерира WSDL във формат 1.1 и функции и опция за генериране на WSDL документи във формат WSDL 2.0.

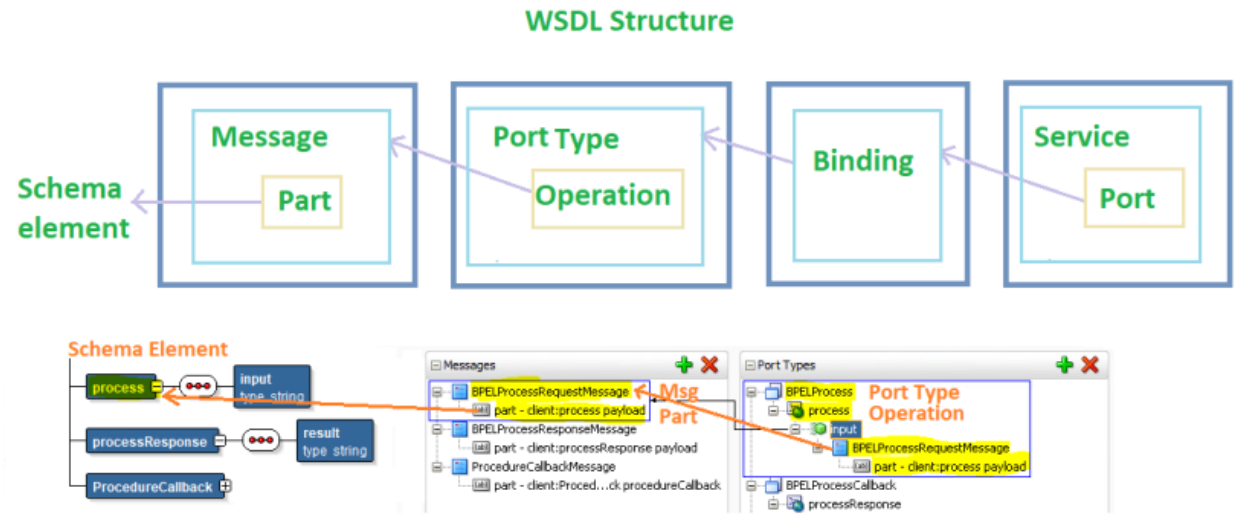
Следващата графика илюстрира разликите между WSDL 1.1 и WSDL 2.0 документи в абстрактния и конкретния слой:



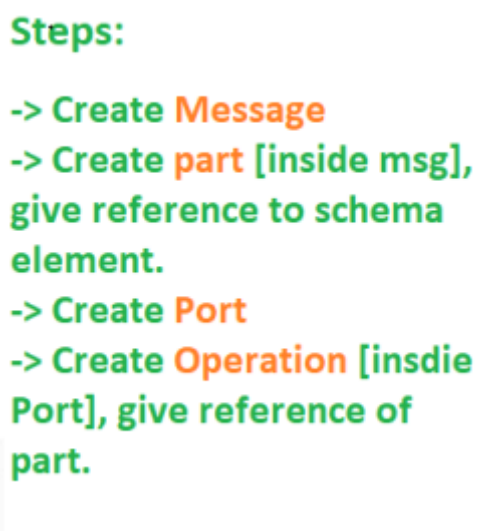
графика 1 (разлика между абстрактния и конкретния слой на спецификациите на WSDL 1.1 и WSDL 2.0)

4.2 Пример 2

Този пример показва графично създаването WSDL структура на документ, както и представянето му чрез source, използвайки Abstract WSDL:



фигура 1 (представяне на свързаността на Schema елементите)



фигура 2(формалните стъпки за създаване на структурата от фигура 1)

```
<wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
         xmlns:client="http://xmlns.oracle.com/JulyApplication/CorrelationProject/BPELProcess"
         xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype">
        <import namespace="http://xmlns.oracle.com/JulyApplication/CorrelationProject/BPELProcess"
             schemaLocation="../Schemas/BPELProcess.xsd"/>
    </schema>
</wsdl:types>
<wsdl:message name="BPELProcessRequestMessage">
    <wsdl:part name="payload" element="client:process"/>
</wsdl:message>
<wsdl:message name="BPELProcessResponseMessage">
    <wsdl:part name="payload" element="client:processResponse"/>
</wsdl:message>
<wsdl:message name="ProcedureCallbackMessage">
    <wsdl:part name="procedureCallback" element="client:ProcedureCallback"/>
</wsdl:message>
<wsdl:portType name="BPELProcess">
    <wsdl:operation name="process">
        <wsdl:input message="client:BPELProcessRequestMessage"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="BPELProcessCallback">
    <wsdl:operation name="processResponse">
        <wsdl:input message="client:BPELProcessResponseMessage"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="ProcedureCallback">
    <wsdl:operation name="procedureCallback">
        <wsdl:input message="client:ProcedureCallbackMessage"/>
    </wsdl:operation>
</wsdl:portType>
</wsdl:definitions>
```

фигура 3(Abstract WSDL Source)

5 Добри практики и методи за използване

Това са най-добрите практики според нас, които е необходимо задължително да се спазват при използване на WSDL.

- Contrct First Development (договаряне на първа разработка) – тази практика дава възможност на разработчика да използва пълните възможности на XML (ограничения, модели и други), като от там е доста лесно да се генерира код за всеки език. Другата причина е, че схемата в <wsdl:types> е договорът между двете системи, които си комуникират. Ако разработчикът създаде WSDL от код, то може да се въведат технически зависимости от конкретния програмен език.

- Document/literal style – Валидацията на RPC кодирания SOAP може да бъде сложна, XPATH заявките и XSLT трансформациите са невъзможни. RPC също създава проблеми с валидирането на XML. Някои SOAP двигатели изпускат дефинирани от схемата пространства на имена, което затруднява валидирането. Така, използвайки

Document/ literal style валидирането, тялото на SOAP е напълно обработено като XML документ, който може да бъде проверен, трансформиран и заявен по стандартен начин.

- Отделяне на опасенията – отделете дефиницията на схемата от самия WSDL файл, като използвате директивите <import...> и <include...>. Това насърчава повторното изпозлване на схеми и разделянето на пространствата от имена в различни .xsd файлове.

- При описването на интерфейса за услуги да се използва WSDL и XSD, вместо Java. Това е така, защото типът порт на дефиницията на услугата е WSDL.

- Интерфейсите от тип WSDL порт обикновено трябва да съдържат повече от една операция. Операциите, дефинирани като част от един интерфейс на услугата, трябва да бъдат семантично свързани с данните, по които работят.

- При проектирането на нова услуга не трябва да се смесват синхронна и асинхронна семантика на показване в един интерфейс от тип WSDL порт. Ако е изгодно да се дефинират две семантики, трябва да се дефинират отделни интерфейси за синхронни и асинхронни покани.

6 Заключение и очаквано бъдещо развитие

Защо WSDL?

WSLD файлът е написан на XML. Това е оргромно предимство, понеже така може да бъде прочетен от всеки програмен език. Така, ако приложението на клиента е написано на .Net, би разбрало XML файла по абсолютно същия начин, както ако приложението на клиента е било написано чрез езика за програмиране Java. Така почти всички устройства са в състояние да интерпретират WSDL файла.

7 Разпределение на работата

Всички точки са разгледани, коментирани, обсъдени и съставени от двамата автори. Есето е продукт на съвместна работа с равно разпределени и положени усилия. Разпределение:

- Точки 1 и 2 са написани от Симона Любенова
- Точки 3, 4 и 5 са написани от Ати Мускова
- Точки 6, 7 и 8 са написани и от двете

8 Използвани литературни източници

1. Пенчев, Й.. Български синтаксис. Пловдив, 1993, стр. 99.
2. https://www.w3.org/TR/wsdl.html
3. https://www.tutorialspoint.com/wsdl/wsdl_introduction.htm
4. https://www.w3schools.com/xmL/xml_wsdl.asp
5. https://www.educba.com/soap-vs-wsdl/
6. https://www.educba.com/wadl-vs-wsdl/
7. https://www.guru99.com/comparison-between-web-services.html
8. https://www.geeksforgeeks.org/difference-between-soap-and-wsdl/
9. https://www.talkwalker.com/blog/top-competitor-analysis-tools
10. https://www.researchgate.net/publication/324106370_Web_Applications_and_Web_Services_A_Comparative_Study
11. https://stackoverflow.com/questions/282448/wsdl-best-practices