แบบจำลองเชิงไวยากรณ์สำหรับทดสอบเว็บเซอร์วิส

Syntax-based Model for Testing Web Services

จุฑาพร อินทะสะระ* และ สุภาภรณ์ กานต์สมเกียรติ*

บทคัดย่อ

ปัจจุบันเทคโนโลยีเว็บเซอร์วิสได้ถูกใช้กันอย่างแพร่หลายในการพัฒนาระบบ ความถูกต้องในการทำงานของเว็บเซอร์วิสเป็นคุณสมบัติที่สำคัญซึ่งต้องคำนึงถึงก่อนตัดสินใจเรียกใช้เว็บเซอร์วิส การทดสอบเป็นกระบวนการหนึ่งที่ใช้ในการตรวจสอบข้อผิดพลาดและประเมินคุณภาพของเซอร์วิสที่เรียกใช้ แต่การสร้างกรณีทดสอบไม่สามารถสร้างจากชุดคำสั่งได้เนื่องจากผู้ใช้มีเพียงเอกสาร WSDL ที่อธิบายถึงการเรียกใช้เซอร์วิสเท่านั้น ดังนั้นงานวิจัยนี้จึงได้เสนอแบบจำลองเชิงไวยากรณ์สำหรับสร้างกรณีทดสอบโดยใช้เอกสาร WSDL ซึ่งเริ่มจากการสกัด XML Schema ที่อธิบายรูปแบบของข้อมูลนำเข้าจากเอกสาร WSDL ต่อจากนั้น XML Schema จะถูกแปลงให้อยู่ในรูปของแบบจำลองเชิงไวยากรณ์เพื่อใช้สร้างกรณีทดสอบ

คำสำคัญ: เว็บเซอร์วิส WSDL การทดสอบ แบบจำลองเชิงไวยากรณ์ XML Schema

Abstract

Web services technology is widely used in distributed system. The correctness of web services is an important property that is able to be realized before invoking the services. Testing is a process to detect faults and evaluate the quality of the software and including the services. WSDL document describes how to invoke the web services and this can also be used for the testing process. This research proposed the syntax-based model for test case generation from WSDL document. The WSDL document is extracted to collect input message in form of XML Schema. The XML Schema is converted into a syntax-based model used for test cases generating.

Keyword: Web Services, WSDL, Testing, Syntax-based Model, XML Schema.

1. บทนำ

สถาปัตยกรรมเชิงบริการ (Service Oriented Architecture: SOA) [1] เป็นแนวคิดในการออกแบบระบบซอฟต์แวร์ที่มีการจัดเตรียมเซอร์วิส (service) ให้กับผู้ใช้โดยการประกาศ (publish) และค้นหา (discover) เซอร์วิส จากแนวคิดดังกล่าวมีเทคโนโลยีที่ช่วยสนับสนุนมากมายซึ่งเทคโนโลยีที่นิยมใช้คือเทคโนโลยีเว็บเซอร์วิส แต่ในการเรียกใช้เว็บเซอร์วิสผู้ใช้จะมั่นใจได้อย่างไรว่าเว็บเซอร์วิสนั้นให้ผลการทำงานที่ถูกต้องและมีความน่าเชื่อถือ ดังนั้นการทดสอบเว็บเซอร์วิสจึงเป็นเรื่องสำคัญและเป็นวิธีการหนึ่งที่ช่วยให้ผู้ใช้เกิดความมั่นใจในเว็บเซอร์วิสที่เรียกใช้ แต่เนื่องจากการใช้เว็บเซอร์วิสเป็นการเรียกใช้ผ่านเครือข่าย ผู้ใช้บริการไม่ได้รับชุดคำสั่ง (source code) จากผู้ให้บริการ ผู้ใช้บริการมีเพียงเอกสาร WSDL เท่านั้น

งานวิจัยนี้จึงได้นำเสนอแบบจำลองเชิงไวยากรณ์สำหรับการสร้างกรณีทดสอบจากเอกสาร WSDL เพื่อทดสอบเว็บเซอร์วิส ขั้นตอนการดำเนินงานจะเริ่มจากการสกัดเอกสาร WSDL ให้เหลือเฉพาะส่วนที่เป็นรูปแบบข้อมูลนำเข้า (input message) ของการดำเนินการ (operation) ที่อยู่ในรูปของ XML Schema จากนั้นสร้างแบบจำลองเชิงไวยากรณ์จาก XML Schema ที่สกัดได้ สุดท้ายสร้างกรณีทดสอบจากแบบจำลองเชิงไวยากรณ์ที่สร้างขึ้น

บทความนี้ประกอบด้วยหัวข้อต่างๆ ดังต่อไปนี้คือ

* *คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์*

หัวข้อที่ 2 เสนอทฤษฎีและงานวิจัยที่เกี่ยวข้อง หัวข้อที่ 3 เสนอแบบจำลองเชิงไวยากรณ์ หัวข้อที่ 4 เสนอกรณีทดสอบ และหัวข้อที่ 5 สรุปและงานในอนาคต

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในส่วนนี้นำเสนอเกี่ยวกับเว็บเซอร์วิส XML Schema และการทดสอบเว็บเซอร์วิส

2.1 เว็บเซอร์วิส (Web Services)

เว็บเซอร์วิส [2] เป็นเทคโนโลยีหนึ่งที่นำมาใช้ในการพัฒนาตามแนวคิดสถาปัตยกรรมเชิงบริการที่อาศัยมาตรฐานเปิด ได้แก่ SOAP (Simple Object Access Protocol) XML (Extensible Markup Language) WSDL (Web Services Description Language) และ UDDI (Universal Description, Discovery and Integration) โดยผู้ให้บริการ (Service Provider) จะใช้เอกสาร WSDL ที่อธิบายรายละเอียดของเซอร์วิสด้วย XML Schema ว่าประกอบด้วยเซอร์วิสอะไรบ้างและสามารถเรียกใช้เซอร์วิสได้อย่างไรมาทำการลงทะเบียนที่ส่วนการลงทะเบียน (Service Broker) ผู้ใช้บริการ (Service Consumer) ค้นหาเซอร์วิสจากส่วนการลงทะเบียนและเรียกใช้เซอร์วิสจากผู้ให้บริการตามเอกสาร WSDL ตัวอย่างเอกสาร WSDL ของเว็บเซอร์วิส weather ที่ใช้สำหรับพยากรณ์อากาศแสดงดังภาพที่ 1

2.2 XML Schema

XML Schema [3] ใช้ในการอธิบายโครงสร้างของเอกสาร XML โดยการกำหนดแท็ก (tag) ต่างๆ ที่คล้ายกับ XML ทำให้สามารถเข้าใจโครงสร้างได้ง่ายและรองรับกับชนิดข้อมูลพื้นฐานต่างๆ ที่ใช้ในภาษาการโปรแกรม โดย XML Schema จะระบุอิลิเมนต์ (element) คุณสมบัติ (attribute) ลำดับ ความสัมพันธ์ จำนวน และชนิดข้อมูล (datatype) ของอิลิเมนต์ในเอกสาร ชนิดข้อมูลที่ใช้ใน XML Schema มี 2 แบบคือ 1) แบบ simple datatype เป็นอิลิเมนต์ที่มีชนิดข้อมูลเป็นชนิดข้อมูลพื้นฐาน (built-in type) หรือชนิดข้อมูลที่ผู้ใช้กำหนดเอง (derived type) โดยการใส่เงื่อนไข (constrain) ให้กับชนิดข้อมูลพื้นฐาน และ 2) แบบ complex datatype เป็นอิลิเมนต์ที่บรรจุอิลิเมนต์ย่อยต่างๆ ไว้ภายใน

2.3 การทดสอบเว็บเซอร์วิส

การทดสอบเว็บเซอร์วิสถือว่าเป็นสิ่งสำคัญที่ต้องคำนึงถึงก่อนที่จะนำเซอร์วิสไปใช้ในระบบต่างๆ ทั้งนี้เพื่อให้เกิดความมั่นใจในผลลัพธ์ของการทำงาน Tsai และคณะ [4] เสนอรูปแบบการทดสอบซอฟต์แวร์เชิงบริการในระดับซิกเนเจอร์ (Signature Level Test Support) ซึ่งเป็นวิธีการหนึ่งในรูปแบบของการทดสอบเซอร์วิสเดี่ยวๆ (Atomic Service Testability) ที่สนับสนุนวิธีการทดสอบแบบกล่องดำ (Black-box Testing) ในหลายลักษณะ เช่น การทดสอบการเชื่อมต่อเซอร์วิส (Service Interface Testing) การทดสอบ SOAP (SOAP Testing) การทดสอบแบบแบ่งส่วน (Partition Testing) การทดสอบแบบสุ่ม (Random Testing) หรือการวิเคราะห์ค่าขอบเขต (Boundary Value Analysis) เป็นต้น

```
<wsdl:definitions
    targetNamespace="http://litwinconsulting.com/webservices/">
<wsdl:types>
 <s:schema elementFormDefault="qualified"
    targetNamespace="http://litwinconsulting.com/webservices/">
  <s:element name="GetWeather">
   <s:complexType>
    <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="City"
                type="s:string"/>
   </s:sequence>
   </s:complexType>
  </s:element>
  <s:element name="GetWeatherResponse">
   <s:complexType>
     <s:sequence>
    <s:element minOccurs="0" maxOccurs="1"
                name="GetWeatherResult" type="s:string"/>
   </s:sequence>
   </s:complexType>
  </s:element>
 </s:schema>
</wsdl:types>
<wsdl:message name="GetWeatherSoapIn">
 <wsdl:part name="parameters" element="tns:GetWeather"/>
</wsdl:message>
<wsdl:message name="GetWeatherSoapOut">
 <wsdl:part name="parameters"
            element="tns:GetWeatherResponse"/>
</wsdl:message>
<wsdl:portType name="WeatherSoap">
 <wsdl:operation name="GetWeather">
  <wsdl:input message="tns:GetWeatherSoapIn"/>
 <wsdl:output message="tns:GetWeatherSoapOut"/>
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding> … </wsdl:binding>
<wsdl:service> … </wsdl:service>
</wsdl:definitions>
```

ภาพที่ 1 *เอกสาร WSDL ของเว็บเซอร์วิส weather*

Hanna และ Munro [5] เสนอแบบจำลองกราฟที่แสดงถึงเงื่อนไขของชนิดข้อมูลแบบ simple datatype และสร้างกรณีทดสอบโดยใช้การวิเคราะห์ค่าขอบเขต (Boundary Value Analysis)

3. แบบจำลองเชิงไวยากรณ์

เอกสาร WSDL เป็นเอกสารที่ระบุรูปแบบข้อมูลนำเข้าของการดำเนินการต่างๆ โดยข้อมูลนำเข้าของแต่ละการดำเนินการจะประกอบด้วยพารามิเตอร์ (parameter) ต่างๆ ซึ่งมีชนิดข้อมูลของพารามิเตอร์ที่ถูกกำหนดโดย XML Schema สำหรับขั้นตอนในการสร้างกรณีทดสอบเริ่มจากการสกัดรูปแบบข้อมูลของแต่ละพารามิเตอร์จากเอกสาร WSDL จากนั้นจะนำอิลิเมนต์ต่างๆ ใน XML Schema ที่สกัดได้ไปสร้างเป็นแบบจำลองเชิงไวยากรณ์ ตัวอย่างของ XML Schema ที่สกัดได้จากเอกสาร WSDL ของเว็บเซอร์วิส weather แสดงดังภาพที่ 2

```
<s:element name="GetWeather">
<s:complexType>
 <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="City"
              type="s:string"/>
 </s:sequence>
</s:complexType>
</s:element>
```

ภาพที่ 2 *XML Schema ของการดำเนินการ GetWeather ของเว็บเซอร์วิส weather*

แบบจำลองเชิงไวยากรณ์ประกอบด้วยโปรดักชั่น (Production) ต่างๆ ซึ่งแต่ละโปรดักชั่นประกอบด้วยสัญลักษณ์ทางซ้ายและสัญลักษณ์ทางขวาของเครื่องหมายเท่ากับ ("=") โดยโปรดักชั่นจะถูกสร้างจนกระทั่งสัญลักษณ์ทางขวาเป็นสัญลักษณ์สิ้นสุดทั้งหมด แต่ละโปรดักชั่นในแบบจำลองเชิงไวยากรณ์สร้างจากอิลิเมนต์ต่างๆ ใน XML Schema ที่สกัดได้

ในการสร้างแบบจำลองไวยากรณ์ได้กำหนดสัญลักษณ์และความหมายของสัญลักษณ์สำหรับการสร้างแบบจำลองเชิงไวยากรณ์แสดงดังตารางที่ 1 และขั้นตอนการสร้างโปรดักชั่นมีรายละเอียดดังนี้

1. สร้างสัญลักษณ์เริ่มต้น (Start Symbol) ของแบบจำลอง

แบบจำลองเชิงไวยากรณ์เป็นสัญลักษณ์ทางซ้ายซึ่งตั้งชื่อตามค่าของแอตทริบิวต์ [name] ของอิลิเมนต์แรก จากนั้น

ตารางที่ 1 *สัญลักษณ์ ความหมาย และตัวอย่างของสัญลักษณ์สำหรับแบบจำลองเชิงไวยากรณ์*

สัญลักษณ์	ความหมายและตัวอย่าง
<>	สัญลักษณ์ไม่สิ้นสุด เช่น <books> หมายถึง สัญลักษณ์ไม่สิ้นสุดชื่อ books
[]	สัญลักษณ์แสดงการมีคำอธิบายข้อมูลในตารางข้อมูล เช่น [employee] หมายถึง สัญลักษณ์สิ้นสุดชื่อ employee ที่มีคำอธิบายข้อมูลในตารางข้อมูล
()	สัญลักษณ์แสดงการมีคำอธิบายข้อมูลในตารางข้อมูลของสัญลักษณ์สิ้นสุดที่เป็นชนิดข้อมูลที่กำหนดมาให้แล้ว เช่น (integer) หมายถึงสัญลักษณ์สิ้นสุดชื่อ integer ที่มีคำอธิบายข้อมูลในตารางข้อมูล
|	สัญลักษณ์แสดงการเลือก เช่น <ISBN>|<price> หมายถึง การเลือกสัญลักษณ์ไม่สิ้นสุดชื่อ ISBN หรือ สัญลักษณ์ไม่สิ้นสุดชื่อ price

สร้างสัญลักษณ์ทางขวาโดยพิจารณาลักษณะของอิลิเมนต์ตามกรณีต่อไปนี้

1.1) กรณีตามด้วยอิลิเมนต์ <complexType>

(1) สร้างสัญลักษณ์ทางขวาเป็น <complexType> (2) สร้างโปรดักชั่นใหม่ที่มีสัญลักษณ์ทางซ้ายเป็น <complexType>

จากนั้นพิจารณาต่อว่าอิลิเมนต์ถัดไปมีลักษณะใดต่อไปนี้

1.1.1) ถ้าอิลิเมนต์ถัดไปเป็นอิลิเมนต์ <sequence>

(1) สร้างสัญลักษณ์ทางขวาเป็น <sequence> (2) สร้างโปรดักชั่นใหม่ที่มีสัญลักษณ์ทางซ้ายเป็น <sequence> (3) สร้างสัญลักษณ์ไม่สิ้นสุดทางขวาเรียงตามลำดับของอิลิเมนต์ <element> ย่อยที่อยู่ถัดจากอิลิเมนต์ <sequence> ตัวอย่างโปรดักชั่นที่สร้างแสดงในตารางที่ 2 บรรทัดที่ 3

1.1.2) ถ้าอิลิเมนต์ถัดไปเป็นอิลิเมนต์ <all>

(1) สร้างสัญลักษณ์ทางขวาเป็น <all> (2) สร้างโปรดักชั่นใหม่ที่มีสัญลักษณ์ทางซ้ายเป็น <all> (3) สร้างสัญลักษณ์ไม่สิ้นสุดทางขวาเรียงตามลำดับของอิลิเมนต์

ตารางที่ 2 *XML Schema และแบบจำลองเชิงไวยากรณ์ของการดำเนินการ GetWeather*

	XML Schema	**Syntax-based Model**
1	<s:element name=”GetWeather”>	
2	<s:complexType>	<GetWeather> = <complexType>
3	<s:sequence>	<complexType> = <sequence> <sequence> = <City>
4	<s:element name=”City” type=”s:string” minOccurs=”0” maxOccurs=”1” />	<City> = null | [City]
5	</s:sequence>	
6	</s:complexType>	
7	</s:element>	

<element> ย่อยที่อยู่ถัดจากอิลิเมนต์ <all> (4) สร้างสัญลักษณ์ “|” (5) สร้างสัญลักษณ์ไม่สิ้นสุดทางขวาซึ่งสลับลำดับระหว่าง
อิลิเมนต์ <element> ย่อยตัวแรกและตัวสุดท้ายที่อยู่ถัดจากอิลิเมนต์ <all>

ตัวอย่างโปรดักชั่นที่สร้างแสดงในตารางที่ 3 บรรทัดที่ 3

ตารางที่ 3 *XML Schema และแบบจำลองเชิงไวยากรณ์ของการดำเนินการ Keyword*

	XML Schema	**Syntax-based Model**
1	<element name=“Keyword”>	
2	<complexType>	<Keyword>= <complexType>
3	<all>	<complexType> = <all> <all> = <key><locale> | <locale><key>
4	<element name=“key” type=“string” />	<City> = null | [City]
5	<element name=“locale” type=“string” minOccurs=“0” />	<locale> = null
6	</all>	
7	</complexType>	
8	</element>	

จากนั้นพิจารณาค่าของแอตทริบิวต์ [type] ในแต่ละอิลิเมนต์ <element> ย่อยในข้อ 1.1.1 – 1.1.3 นั่นคือ ถ้าอิลิเมนต์ย่อยนั้นมีชนิดข้อมูลเป็นชนิดข้อมูลที่กำหนดมาให้แล้วให้ดำเนินการต่อในข้อ 2 ถ้าไม่ใช่ให้ดำเนินการต่อในข้อ 3 หากในอิลิเมนต์ <element> ย่อยที่กำลังพิจารณาไม่ปรากฏแอตทริบิวต์ [type] ให้ดำเนินการตามขั้นตอนดังแสดงในภาพที่ 3 จากนั้นให้พิจารณาอิลิเมนต์ถัดไป กรณีที่เป็นอิลิเมนต์ <complexType> ให้ดำเนินการตามข้อ 1.1 และกรณีที่เป็นอิลิเมนต์ <simpleType> ให้ดำเนินการตามข้อ 3

ตารางที่ 4 *XML Schema และแบบจำลองเชิงไวยากรณ์ของการดำเนินการ Keyword*

	XML Schema	**Syntax-based Model**
1	<element name= “person”>	
2	<complexType>	<person> = <complexType>
3	<choice>	<complexType> = <choice> <choice> = <name> | <car>
4	<element name=”name” type= “string”/>	<name> = [name]
5	<element name=”car” type=”CarType”/>	
6	</choice>	
7	</complexType>	
8	</element>	
9	<element name= “CarType”>	
10	<simpleType>	
11	<restriction base= “string”>	<car> = (string)
12	<enumeration value= “Audi”/>	
13	<enumeration value= “Golf”/>	
14	<enumeration value= “BMW”/>	
15	</restriction>	
16	</simpleType>	
17	</element>	

2. กรณีเป็นชนิดข้อมูลที่กำหนดมาให้แล้ว

(1) สร้างโปรดักชั่นใหม่ที่มีสัญลักษณ์ทางซ้ายเป็นสัญลักษณ์ไม่สิ้นสุดซึ่งตั้งชื่อตามค่าของแอตทริบิวต์ [name]

(2) สร้างสัญลักษณ์ทางขวาโดยพิจารณาจากแอตทริบิวต์

[minOccurs] และแอตทริบิวต์ [maxOccurs] ด้วยขั้นตอนดังแสดงในภาพที่ 3

ขั้นที่ 1 พิจารณาแอตทริบิวต์ [minOccurs] และแอตทริบิวต์ [maxOccurs] ดังนี้

1. ถ้าไม่กำหนดแอตทริบิวต์ [minOccurs] และแอตทริบิวต์ [maxOccurs]

(1) สร้างสัญลักษณ์สิ้นสุดตามค่าของแอตทริบิวต์ [name] ของอิลิเมนต์ <element> อยู่ภายในสัญลักษณ์ “[” และ “]” 1 ครั้ง (2) ดำเนินการต่อในขั้นที่ 2

2. ถ้าค่าของแอตทริบิวต์ [minOccurs] = x และ ค่าของแอตทริบิวต์ [maxOccurs] = y และ 1<=x<y

(1) สร้างสัญลักษณ์สิ้นสุดตามค่าของแอตทริบิวต์ [name] ของอิลิเมนต์ <element> อยู่ภายในสัญลักษณ์ “[” และ “]” x ครั้ง (2) สร้างสัญลักษณ์ “|” (3) สร้างสัญลักษณ์สิ้นสุดตามค่าของแอตทริบิวต์ [name] ของอิลิเมนต์ <element> อยู่ภายในสัญลักษณ์ “[” และ “]” y ครั้ง (4) ดำเนินการต่อในขั้นที่ 2

3. ถ้าค่าของแอตทริบิวต์ [minOccurs] = x หรือ ค่าของแอตทริบิวต์ [maxOccurs] = x หรือ ค่าของแอตทริบิวต์ [minOccurs] เท่ากับค่าของแอตทริบิวต์ [maxOccurs] คือเท่ากับ x แล้ว

3.1 ถ้า x = 0 ให้สร้างสัญลักษณ์สิ้นสุดเป็น null 1 ครั้ง

3.2 ถ้า x >= 1

(1) สร้างสัญลักษณ์สิ้นสุดตามค่าของแอตทริบิวต์ [name] ของอิลิเมนต์ <element> อยู่ภายในสัญลักษณ์ “[” และ “]” x ครั้ง (2) ดำเนินการต่อในขั้นที่ 2

4. ถ้าค่าของแอตทริบิวต์ [maxOccurs] เป็น unbounded แล้ว

(1) สร้างสัญลักษณ์สิ้นสุดเป็น null 1 ครั้ง (2) สร้างสัญลักษณ์ “|” (3) สร้างสัญลักษณ์สิ้นสุดตามค่าของแอตทริบิวต์ [name] ของอิลิเมนต์ <element> อยู่ภายในสัญลักษณ์ “[” และ “]” 1 ครั้ง (4) ดำเนินการต่อในขั้นที่ 2 ขั้นที่ 2 เพิ่มข้อมูลเพื่ออธิบายสัญลักษณ์สิ้นสุดลงในตารางข้อมูล

ภาพที่ 3 *ขั้นตอนการพิจารณาแอตทริบิวต์ [minOccurs] และแอตทริบิวต์ [maxOccurs] เพื่อสร้างแบบจำลองเชิงไวยากรณ์*

3. กรณีเป็นชนิดข้อมูลแบบผู้ใช้กำหนดเอง

(1) สร้างโปรดักชั่นใหม่ที่มีสัญลักษณ์ทางซ้ายเป็นสัญลักษณ์ไม่สิ้นสุดซึ่งตั้งชื่อตามค่าของแอตทริบิวต์ [name]

(2) สร้าง สัญลักษณ์ทางขวาเป็นสัญลักษณ์สิ้นสุดซึ่งตั้งชื่อตามค่าของแอตทริบิวต์ [base] ของอิลิเมนต์ <restriction> อยู่ภายในสัญลักษณ์ “(” และ “)” (3) เพิ่มข้อมูลอธิบายสัญลักษณ์สิ้นสุดลงในตารางข้อมูล กรณีที่มีข้อมูลมากกว่า 1 ข้อมูลให้แยกด้วยเครื่องหมาย “,”

ตัวอย่างโปรดักชั่นที่สร้างใน (1)-(2) แสดงในตารางที่ 4 บรรทัดที่ 11 และคำอธิบายข้อมูลใน (3) ของสัญลักษณ์สิ้นสุดของโปรดักชั่นในตารางที่ 4 บรรทัดที่ 11 แสดงในตารางที่ 5 บรรทัดที่ 2

ตารางที่ 5 *ตารางข้อมูลสำหรับสัญลักษณ์สิ้นสุดของการดำเนินการ person*

	Name	**Data**
1	name	string
2	string	enumeration = Audi, enumeration = Golf, enumeration = BMW

4. ทำการปรับปรุง (optimize) แบบจำลองเชิงไวยากรณ์โดยพิจารณาสัญลักษณ์ไม่สิ้นสุดทางขวาที่เหมือนกันกับสัญลักษณ์ไม่สิ้นสุดทางซ้ายของโปรดักชั่นถัดไปเพื่อรวมโปรดักชั่น ตัวอย่างของแบบจำลองเชิงไวยากรณ์ในตารางที่ 4 ด้านขวาที่ผ่านการทำปรับปรุงแสดงได้ดังภาพที่ 4

```
<person> = <name> | <car>
<name> = [name]
<car> = (string)
```

ภาพที่ 4 *แบบจำลองเชิงไวยากรณ์ของการดำเนินการ person ที่ผ่านการปรับปรุง*

4. กรณีทดสอบ

การสร้างกรณีทดสอบในงานวิจัยนี้ประกอบด้วยขั้นตอนหลัก 2 ขั้นตอน ดังนี้

1) การทำ derivation โดยการนำแบบจำลองเชิงไวยากรณ์ที่ผ่านการปรับปรุงจากขั้นตอนที่ 4 ในหัวข้อ 3 มาทำการ derivation โดยใช้ลูกศร (-->) แทนการ derivation ในแต่ละครั้ง การ derivation เริ่มต้นจากสัญลักษณ์เริ่มต้นของแบบจำลองเชิงไวยากรณ์และทำการ derivation โดยใช้โปรดักชั่นแรกทำการ derivation ก่อน การทำ derivation

จะทำซ้ำๆ โดยเลือกโปรดักชั่นครั้งละ 1 โปรดักชั่นที่มีสัญลักษณ์ไม่สิ้นสุดทางซ้ายตรงกับสัญลักษณ์ไม่สิ้นสุดทางขวาของลูกศรมาใช้ ในการทำ derivation แต่ละโปรดักชั่นในแบบจำลองจะต้องถูกใช้อย่างน้อย 1 ครั้งและต้องทำการ derivation จนกระทั่งสัญลักษณ์ทางขวาของลูกศรเป็นสัญลักษณ์สิ้นสุดทั้งหมด ตัวอย่างการ derivation แสดงได้ดังภาพที่ 5

```
<person> ---> <name><car>
         ---> [name] <car>
         ---> [name] (string)
```

ภาพที่ 5 *การ derivation โดยใช้แบบจำลองเชิงไวยากรณ์ของการดำเนินการ person*

2) การสร้างกรณีทดสอบ โดยพิจารณาจากสัญลักษณ์สิ้นสุดทุกตัว นั่นคือสัญลักษณ์สิ้นสุดแต่ละตัวจะมีกรณีที่ประกอบด้วยค่าที่ถูกต้องและไม่ถูกต้องตามสัญลักษณ์สิ้นสุดนั้น ยกเว้นสัญลักษณ์สิ้นสุด null ซึ่งจะมีเพียงค่าเดียวคือค่าว่าง การสร้างกรณีที่เป็นค่าที่ถูกต้องจะพิจารณาตามสัญลักษณ์สิ้นสุดดังนี้

(1) กรณีสัญลักษณ์สิ้นสุดอยู่ภายในสัญลักษณ์ “[” และ “]” ให้พิจารณาข้อมูลในตารางข้อมูลดังนี้

1.1) ถ้าข้อมูลในตารางข้อมูลเป็นชนิดข้อมูลแบบตัวเลข (Number Type) ให้สร้างค่าที่ถูกต้องโดยใช้ค่าต่อไปนี้คือ ค่าลบ ค่าศูนย์ และค่าบวก

1.2) ถ้าข้อมูลในตารางข้อมูลเป็นชนิดข้อมูลนอกเหนือจากข้อ 1.1 ให้สร้างค่าที่ถูกต้องโดยการสุ่มค่า

(2) กรณีสัญลักษณ์สิ้นสุดอยู่ภายในสัญลักษณ์ “(” และ “)” ให้พิจารณาข้อมูลในตารางข้อมูลดังนี้

2.1) ถ้าข้อมูลในตารางข้อมูลเป็นข้อมูลของเงื่อนไขจำกัดกลุ่มของค่า (Restriction on a set of values) ได้แก่ enumeration ให้สร้างค่าที่ถูกต้องโดยใช้ค่าทุกค่าของเงื่อนไขจำกัดกลุ่มของค่า

2.2) ถ้าข้อมูลในตารางข้อมูลเป็นข้อมูลของเงื่อนไขจำกัดอื่นๆ ให้สร้างค่าที่ถูกต้องตามสัญลักษณ์สิ้นสุดและเงื่อนไขจำกัดนั้น

ดังนั้นจากสัญลักษณ์สิ้นสุดในภาพที่ 4 มีสัญลักษณ์สิ้นสุด 2 ตัวคือสัญลักษณ์สิ้นสุด [name] ของอิลิเมนต์ <name> และสัญลักษณ์สิ้นสุด (string) ของอิลิเมนต์ <car> ซึ่งมีคำอธิบายข้อมูลในตารางที่ 5 ทำให้สามารถสร้างค่าที่ถูกต้องและไม่ถูกต้องตามสัญลักษณ์สิ้นสุดดังแสดงในตารางที่ 6 และจำนวนกรณีทดสอบทั้งหมดได้จากผลคูณคาร์ทีเชียนของจำนวนกรณีทั้งหมดในแต่ละอิลิเมนต์ นั่นคือ 2x4 = 8 กรณี

ตารางที่ 6 *ค่าที่ถูกต้องและไม่ถูกต้องตามสัญลักษณ์สิ้นสุดของการดำเนินการ person*

อิลิเมนต์	ค่าที่ถูกต้องตามสัญลักษณ์สิ้นสุด	ค่าที่ไม่ถูกต้องตามสัญลักษณ์สิ้นสุด
<name>	asdfUlokn234	2
<car>	Audi, Golf, BMW	Toyota

5. สรุปและงานในอนาคต

งานวิจัยนี้เสนอแบบจำลองเชิงไวยากรณ์เพื่อสร้างกรณีทดสอบสำหรับทดสอบเว็บเซอร์วิสโดยใช้เอกสาร WSDL ขั้นตอนเริ่มจากการสกัด XML Schema จากเอกสาร WSDL จากนั้นสร้างแบบจำลองเชิงไวยากรณ์โดยใช้ XML Schema ที่สกัดได้ และนำแบบจำลองเชิงไวยากรณ์ที่ได้ไปสร้างกรณีทดสอบ วิธีการที่นำเสนอนี้ช่วยแก้ปัญหาในกรณีที่ไม่สามารถสร้างกรณีทดสอบจากชุดคำสั่งได้ ในอนาคตจะปรับปรุงขั้นตอนของการสร้างแบบจำลองเชิงไวยากรณ์เพื่อให้มีประสิทธิภาพมากขึ้น เช่น เพิ่มสัญลักษณ์ในส่วนที่เป็นลักษณะของการปรากฏ (occurrence) ของอิลิเมนต์ นอกจากนั้นจะนำหลักการที่ได้นำเสนอไปทดสอบกับเว็บเซอร์วิสที่มีขนาดใหญ่และซับซ้อน

6. เอกสารอ้างอิง

[1] Michael P. Papazoglou et al., “Service-Oriented Computing: State of the Art and Research Challenges,” Computer, v.40 n.11, pp.38-45, November 2007.

[2] W3C, *Web Services Architecture*, *W3C Working Group Note*, February 11, 2004. Available from: http://www.w3.org/TR/ws-arch/

[3] W3C, *XML Schema Part 0: Primer Second Edition*, *W3C Recommendation*, 28 October 2004, Available from: http://www.w3.org/TR/2004/REC-xmlschema-0-20041028.

[4] W. T. Tsai et al., “Testability of Software in Service-Oriented Architecture,” *Proceedings of*

the 30th Annual International Computer Software and Applications Conference (*COMPSAC* 2006), pp. 163-170.

[5] S. Hanna and M. Munro, “An Approach for Specification-based Test Case Generation for Web Services,” *ACS/IEEE International Conference on Computer Systems and Application* (*AICCSA* 2007), pp.16-23.