บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1. แนวคิดและทฤษฎี

2.1.1. เอกซ์เอ็มแอล (XML: eXtensible Markup Language)

เอกซ์เอ็มแอล [8, 9] เป็นภาษาที่ออกแบบเพื่อใช้อธิบายข้อมูล และสามารถอธิบายความหมายของตัวเองได้ โดยสามารถนิยามข้อมูลหรือแท็ก (Tag) ที่ต้องการได้ด้วยตนเอง ซึ่งความแตกต่างระหว่างเอกซ์เอ็มแอลกับเอชทีเอ็มแอล (HTML) คือ เอชทีเอ็มแอลเป็นภาษาที่ออกแบบ เพื่อแสดงผลข้อมูลและสนใจว่าจะแสดงข้อมูลอย่างไร แต่เอกซ์เอ็มแอลจะสนใจว่าข้อมูลอะไรบ้างที่ใช้ติดต่อสื่อสารกันหรือข้อมูลอะไรที่ต้องการเก็บรักษา ซึ่งเอกซ์เอ็มแอลจะไม่ขึ้นอยู่กับซอฟต์แวร์ (Software) และฮาร์ดแวร์ (Hardware) สำหรับการจัดการข้อมูล

โครงสร้างของเอกสารเอกซ์เอ็มแอล ประกอบด้วย 2 ส่วนที่สำคัญ ได้แก่ โครงสร้างทางตรรกะ (Logical structure) และโครงสร้างทางกายภาพ (Physical structure) โดยโครงสร้างทางกายภาพ เอกซ์เอ็มแอลประกอบด้วยหน่วยหลายๆ หน่วย ซึ่งเรียกหน่วยเหล่านี้ว่า เอนทรีส์ (Entries) ในแต่ละเอนทรี (Entry) อาจจะอ้างอิงถึงเอนทรีอื่นๆ ที่ถูกแทรกไว้ในเอกสารเอกซ์เอ็มแอล และเอกสารเอกซ์เอ็มแอลต้องเริ่มต้นด้วยรูทของเอนทรีส์ (Root element) เสมอ ส่วนโครงสร้างทางตรรกะเอกสารเอกซ์เอ็มแอลประกอบด้วยการประกาศ (Declaration) อิลิเมนต์ หมายเหตุ (Comment) อักขระพิเศษ และคำสั่งการประมวลผล (Processing instruction)

กฎไวยากรณ์ของเอกซ์เอ็มแอล (XML syntax rules) มีดังนี้

2.1.1.1. ทุกเอกซ์เอ็มแอลอิลิเมนต์ต้องมีแท็กปิดเสมอ เช่น

```
<p>This is a paragraph</p>
```

2.1.1.2. เอกซ์เอ็มแอลแท็ก มีความหมายแตกต่างกันระหว่างตัวอักษรเล็กและตัวอักษรใหญ่

2.1.1.3. เอกซ์เอ็มแอลอิลิเมนต์ต้องมีการจัดกลุ่มอย่างเหมาะสม เช่น

```
<b><i>This text is bold and italic</i></b>
```

2.1.1.4. เอกสารเอกซ์เอ็มแอลต้องมีรูทอิลิเมนต์ ซึ่งเป็นส่วนเริ่มต้นของทุกอิลิเมนต์

2.1.1.5. ข้อมูลของเอกซ์เอ็มแอลแอ็ตทริบิวต์ (XML attribute) ต้องอยู่ภายใต้เครื่องหมายคำพูด เช่น ตัวอย่างการเขียนเอกสารเอกซ์เอ็มแอลดังรูปที่ 2.1

จากรูปที่ 2.1 แสดงตัวอย่างการเขียนเอกซ์เอ็มแอล โดยในบรรทัดแรกเป็นการประกาศเอกซ์เอ็มแอล ซึ่งกำหนดให้เวอร์ชันของเอกซ์เอ็มแอลเป็นเวอร์ชัน 1.0 และรูปแบบการเข้ารหัส คือ ISO-8859-1 บรรทัดที่ 2 เป็นการประกาศรูทอิลิเมนต์ของเอกสารเอกซ์เอ็มแอล บรรทัดที่ 3 - 6 เป็นอิลิเมนต์ลูกของรูทอิลิเมนต์ และบรรทัดสุดท้าย เป็นการกำหนดปิดของรูทอิลิเมนต์

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this
weekend!</body>
</note>
```

รูปที่ 2.1. ตัวอย่างการเขียนเอกสารเอกซ์เอ็มแอล

2.1.2. เอกซ์เอ็มแอลสคีมา

เอกซ์เอ็มแอลสคีมา [10] เป็นภาษาที่นิยามโครงสร้างข้อมูลของเอกสารเอกซ์เอ็มแอล โดยการนิยามนี้สามารถใช้เป็นข้อตกลงในการแลกเปลี่ยนข้อมูลระหว่างเครื่องมือ ทำให้เกิดความเชื่อมั่นของข้อมูล เช่น เว็บเซอร์วิส อีคอมเมิร์ซ (E-Commerce) เป็นต้น รวมทั้งเอกซ์เอ็มแอลสคีมายังสามารถใช้ตรวจสอบความถูกต้องของข้อมูลในเอกซ์เอ็มแอลได้อีกด้วย ซึ่งกฎไวยากรณ์ของเอกซ์เอ็มแอลสคีมาจะเหมือนกับเอกซ์เอ็มแอลทำให้ง่ายต่อการใช้งาน และสามารถนิยามข้อมูลหรือแท็กที่ต้องการได้เช่นเดียวกับเอกซ์เอ็มแอล

จากรูปที่ 2.2 แสดงตัวอย่างเอกซ์เอ็มแอลสคีมาสำหรับนิยามอิลิเมนต์ของเอกสารเอกซ์เอ็มแอลที่ใช้จากรูปที่ 2.1 ซึ่งจะแบ่งอิลิเมนต์ออกเป็น 2 ประเภท คือ ชนิดข้อมูลเชิงซ้อน (Complex type) และชนิดข้อมูลอย่างง่าย (Simple type) โดยชนิดข้อมูลเชิงซ้อนเป็นเอกซ์เอ็มแอลอิลิเมนต์ที่สามารถประกอบด้วยอิลิเมนต์อื่นๆ หรือแอ็ตทริบิวต์ และชนิดข้อมูลอย่างง่ายเป็นเอกซ์เอ็มแอลที่ประกอบด้วยข้อความอย่างเดียวเท่านั้น เช่น อิลิเมนต์ note จะเป็นชนิดข้อมูลเชิงซ้อน และอิลิเมนต์ to, from, heading และ body เป็นชนิดข้อมูลอย่างง่าย

```
<?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>
```

รูปที่ 2.2. ตัวอย่างเอกสารเอกซ์เอ็มแอลสคีมา

2.1.3. ดับเบิลยูเอสดีแอล

ดับเบิลยูเอสดีแอล [2, 3] เป็นรูปแบบของเอกซ์เอ็มแอลสำหรับอธิบายเว็บเซอร์วิสและการเข้าถึงข้อมูลบนเว็บเซอร์วิส รวมทั้งสามารถระบุตำแหน่งของเว็บเซอร์วิส และโอเปอร์เรชันที่ให้บริการ

รูปแบบของดับเบิลยูสเอสดีแอลแสดงดังรูปที่ 2.3 และมีรายละเอียดต่างๆ ดังนี้

2.1.3.1. <portType>

อิลิเมนต์นี้แสดงรายละเอียดเว็บเซอร์วิส โอเปอร์เรชันที่สามารถทำงาน และแมสเสจที่มีความสัมพันธ์กับเว็บเซอร์วิส

2.1.3.2. <message>

อิลิเมนต์นี้ใช้กำหนดอิลิเมนต์ข้อมูลของโอเปอร์เรชัน

2.1.3.3. <types>

อิลิเมนต์นี้ใช้กำหนดชนิดของข้อมูลที่ใช้ในเว็บเซอร์วิส

2.1.3.4. <binding>

อิลิเมนต์นี้ใช้กำหนดรูปแบบของแมสเสจและรายละเอียดของโพรโทคอล (protocol) สำหรับแต่ละพอร์ต (port)

```
<definitions>
    <types>
        definition of types.....
    </types>
    <message>
        definition of a message....
    </message>
    <portType>
        definition of a port.....
    </portType>
    <binding>
        definition of a binding....
    </binding>
</definitions>
```

รูปที่ 2.3. รูปแบบของดับเบิลยูเอสดีแอล

2.1.4. ดับเบิลยูเอสดีแอล-เอส

เอกสารดับเบิลยูเอสดีแอล-เอส [4, 6] เป็นส่วนที่กำหนดกลไกของความสัมพันธ์ระหว่างซิแมนทิกแอนโนเทชัน (Semantic annotation) ที่อธิบายเว็บเซอร์วิสโดยใช้ดับเบิลยูเอสดีแอล ซึ่งดับเบิลยูเอสดีแอล-เอสได้กำหนดข้อมูลที่เพิ่มเติมขึ้นเพื่อช่วยอธิบายความหมายของดับเบิลยูเอสดีแอลและการทำงานของเว็บเซอร์วิส ได้แก่ ข้อมูลนำเข้า ผลลัพธ์จากการทำงาน และเงื่อนไขในการเรียกใช้งานเว็บเซอร์วิส (Precondition) รวมทั้งเงื่อนไขของผลลัพธ์ที่ได้จากการทำงานของเว็บเซอร์วิส (Post-condition) ทำให้ผู้เรียกใช้งานเว็บเซอร์วิสสามารถรู้รายละเอียดในการทำงานและเงื่อนไขต่างๆ มากกว่าดับเบิลยูเอสดีแอล

อิลิเมนต์และแอ็ตทริบิวต์ที่กำหนดเพิ่มเติมจากดับเบิลยูเอสดีแอล มีรายละเอียดดังนี้

2.1.4.1. modelReference

modelReference เป็นแอ็ตทริบิวต์ที่กำหนดเพิ่มเติม เพื่อใช้กำหนดความสัมพันธ์ระหว่างดับเบิลยูเอสดีแอลและซิแมนทิกโมเดล (Semantic model) ซึ่งสามารถนำไปใช้ภายใต้ชนิดข้อมูลเชิงซ้อน โอเปอร์เรชัน และอิลิเมนต์ที่กำหนดเพิ่มเติมในดับเบิลยูเอสดีแอล-เอส ได้แก่ เงื่อนไขในการเรียกใช้งานเว็บเซอร์วิสและเงื่อนไขของผลลัพธ์ในการทำงานของเว็บเซอร์วิส

2.1.4.2. schemaMapping

schemaMapping เป็นแอ็ตทริบิวต์ที่กำหนดเพิ่มเติม เพื่อใช้ระบุเอกซ์เอสดีอิลิเมนต์ (XSD element) เพิ่มเติม

2.1.4.3. precondition และ effect

precondition และ effect เป็นอิลิเมนต์ที่กำหนดเพิ่มเติม ซึ่งจะสามารถใช้ระบุเป็นอิลิเมนต์ลูกของโอเปอร์เรชันอิลิเมนต์ โดย precondition เป็นส่วนที่กำหนดเงื่อนไขสำหรับเว็บเซอร์วิสเริ่มทำงาน และ effect เป็นส่วนที่กำหนดเงื่อนไขของผลลัพธ์ในการทำงานของเว็บเซอร์วิส

2.1.4.5. category

category เป็นแอ็ตทริบิวต์ที่กำหนดเพิ่มเติม ซึ่งแอ็ตทริบิวต์นี้จะประกอบด้วยข้อมูลของกลุ่มของเซอร์วิส

รูปที่ 2.4 แสดงตัวอย่างเอกสารดับเบิลยูเอสดีแอล-เอสสำหรับเว็บเซอร์วิสที่ให้บริการการสั่งซื้อสินค้า โดยโอเปอร์เรชันชื่อว่า ProcessPurcahseOrder ซึ่งโอเปอร์เรชันนี้จะประกอบด้วยเงื่อนไขในการเรียกใช้งานเว็บเซอร์วิส และเงื่อนไขของผลลัพธ์ในการทำงานของเว็บเซอร์วิส ซึ่งในเอกสารดับเบิลยูเอสดีแอล-เอสจะเรียกว่า effect โดยเงื่อนไขทั้งสองจะถูกอ้างอิงไปยังเอกสารอื่นที่เก็บรายละเอียดของเงื่อนไขไว้

2.1.5. เอสดับเบิลยูอาร์แอล

เอสดับเบิลยูอาร์แอล [11, 12] เป็นภาษาที่ออกแบบมา เพื่อใช้เป็นภาษาของกฎสำหรับซิแมนทิกเว็บ (Semantic web) บนพื้นฐานของอาวล์ (OWL: Web Ontology Language) ซึ่งเอสดับเบิลยูอาร์แอล ประกอบด้วยส่วนของพจน์นำ (Antecedent part) และส่วนของพจน์ตาม (Consequent part) ซึ่งส่วนของพจน์นำจะอ้างอิงถึงอิลิเมนต์บอดี (body) และส่วนของพจน์จะอ้างอิงถึงอิลิเมนต์เฮด (head) โดยเงื่อนไขในส่วนของพจน์นำเป็นจริงแล้วเงื่อนไขของส่วนของพจน์ตามต้องเป็นจริงด้วย และประโยคเงื่อนไขของทั้งสองส่วนนั้นต้องเขียนอยู่รูปของประพจน์เชื่อมที่เป็นบวก (Positive conjunction) ของอะตอม (Atom) เท่านั้น ซึ่งอะตอมสามารถเขียนอยู่ในรูปแบบ C(x) P(x, y) sameAs(x, y) หรือ differentFrom(x, y) โดย C เป็นคำอธิบายอาวล์ (An OWL description) P เป็นคุณสมบัติของอาวล์ (An OWL property) และ x, y เป็นตัวแปร หรือค่าข้อมูลของอาวล์ (OWL data value) โดยสามารถเขียนกฎในเอกสารเอสดับเบิลยูอาร์แอลได้หลายรูปแบบ และมีรายละเอียดในการเขียนกฎแต่ละแบบดังนี้

```
<?xml version="1.0" encoding="iso-8859-1"?>
<definitions name="PurchaseOrder"
    targetNamespace="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
    xmlns="http://www.w3.org/2004/08/wsdl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
    xmlns:xsd1="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
    xmlns:wssem="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
    xmlns:POOntology="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/PurchaseOrder.owl"
    xmlns:Rosetta="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/rosetta.owl">
    <types>
        <xs:import namespace="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
            schemaLocation="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/WSSemantics.xsd"/>
        <xs:import namespace="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
            schemaLocation="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/POBilling.xsd" />
        <xs:import namespace="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
            schemaLocation="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/POItem.xsd" />
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl"
            xmlns="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/purchaseOrder.wsdl">
        <xs:complexType name="processPurchaseOrderRequest">
            <xs:all>
                <xs:element name="billingInfo" type="xsd1:POBilling"/>
                <xs:element name="orderItem" type="xsd1:POItem"/>
            </xs:all>
        </xs:complexType>
        <xs:element name="processPurchaseOrderResponse" type="xs:string"
            wssem:modelReference="POOntology#OrderConfirmation"/>
        </xs:schema>
    </types>
    <interface name="PurchaseOrder">
        <wssem:category name="Electronics" taxonomyURI="http://www.naics.com/" taxonomyCode="443112" />
        <operation name="processPurchaseOrder" pattern="wsdl:in-out"
wssem:modelReference="Rosetta:RequestPurchaseOrder" >
            <input messageLabel ="processPurchaseOrderRequest" element="tns:processPurchaseOrderRequest"/>
            <output messageLabel ="processPurchaseOrderResponse"  element="processPurchaseOrderResponse"/>
            <wssem:precondition name="ExistingAcctPrecond"  wssem:modelReference="POOntology#AccountExists"/>
            <wssem:effect name="ItemReservedEffect" wssem:modelReference="POOntology#ItemReserved"/>
        </operation>
    </interface>
</definitions>
```

รูปที่ 2.4. ตัวอย่างเอกสารดับเบิลยูเอสดีแอล-เอส

2.1.5.1. ไวยากรณ์นามธรรม (Abstract syntax)

ไวยากรณ์นามธรรม เป็นส่วนที่นิยามเพิ่มเติมจากไวยากรณ์นามธรรมของคำอธิบายอาวล์ (The abstract syntax of OWL) และอธิบายไวยากรณ์นี้ในรูปแบบส่วนขยายของบีเอ็นเอฟ (Extended BNF) ดังรูปที่ 2.5 ทำให้กฎที่เขียนตามไวยากรณ์นี้จะอ่านและเข้าใจได้ยาก โดยข้อกำหนดของไวยากรณ์ในรูปแบบส่วนขยายของบีเอ็นเอฟ ได้แก่ ตัวแปรที่เป็นเทอร์มินัล (Terminal) จะเขียนภายใต้เครื่องหมายคำพูด และตัวแปรที่ไม่ใช่เทอร์มินัล (Non-terminal) จะเป็นตัวหนาและไม่อยู่ภายใต้เครื่องหมายคำพูด ทางเลือกของข้อมูลที่กำหนดให้แต่ละตัวแปรจะแบ่งด้วยเครื่องหมายแท่งเชิงตั้ง (Vertical bar: |)

จากรูปที่ 2.5 สัจพจน์ของกฎ (A rule axiom) ประกอบด้วยพจน์นำและพจน์ตาม และสามารถกำหนดการอ้างอิงยูอาร์ไอ (URI reference) ให้กับสัจพจน์ของกฎ เพื่อกำหนดเอกลักษณ์ของกฎ

```
axiom          ::= rule
rule           ::= 'Implies(' [ URIreference ] { annotation } antecedent consequent ')'
antecedent     ::= 'Antecedent(' { atom } ')'
consequent     ::= 'Consequent(' { atom } ')'
atom           ::= description '(' i-object ')'
                 | dataRange '(' d-object ')'
                 | individualvaluedPropertyID '(' i-object i-object ')'
                 | datavaluedPropertyID '(' i-object d-object ')'
                 | sameAs '(' i-object i-object ')'
                 | differentFrom '(' i-object i-object ')'
                 | builtIn '(' builtinID { d-object } ')'
builtinID      ::= URIreference
i-object       ::= i-variable | individualID
d-object       ::= d-variable | dataLiteral
i-variable     ::= 'I-variable(' URIreference ')'
d-variable     ::= 'D-variable(' URIreference ')'
```

รูปที่ 2.5. ไวยากรณ์นามธรรมสำหรับการเขียนเอสดับเบิลยูอาร์แอล

2.1.5.2. ไวยากรณ์ที่ง่ายต่อการเข้าใจสำหรับมนุษย์ (Human readable syntax)

รูปแบบของไวยากรณ์นี้ มีดังนี้

antecedent => consequent

โดยพจน์นำและพจน์ตามเป็นประโยคเชื่อมของอะตอม (Conjunctions of atoms) โดยเขียนในรูปแบบ A1 Λ ... Λ An และรูปแบบของตัวแปรที่ใช้จะมีเครื่องหมายคำถามนำหน้าชื่อของตัวแปรด้วย เช่น ?x แทน ตัวแปร x เป็นต้น

2.1.5.3. ไวยากรณ์รูปแบบเอกซ์เอ็มแอล (XML concrete syntax)

ไวยากรณ์รูปแบบเอกซ์เอ็มแอล เป็นไวยากรณ์ที่เกิดจากไวยากรณ์เอกซ์เอ็มแอลสำหรับอาวล์ด้วยไวยากรณ์เอกซ์เอ็มแอลสำหรับรูทเอ็มแอล (RuleML XML syntax) ซึ่งการเขียนกฎตามไวยากรณ์นี้จะต้องสอดคล้องกับข้อกำหนดของเอกซ์เอ็มแอลสคีมาที่กำหนดขึ้นจากดับเบิลยูเอสสามซี และมีรายละเอียดของอิลิเมนต์และแอ็ตทริบิวต์ที่สำคัญพอสังเขป ดังนี้

1) ruleml:var[axiom] เป็นอิลิเมนต์สำหรับนิยามตัวแปร

2) ruleml:imp[axiom] เป็นอิลิเมนต์สำหรับนิยามพจน์นำและพจน์ตาม โดยนิยามพจน์นำและพจน์ตาม โดยใช้ ruleml:_body และ ruleml:_head ตามลำดับ

3) swrlx:builtin เป็นแอ็ตทริบิวต์ที่ใช้สำหรับอ้างอิงเอลดับเบิลยูอาร์แอลบิวอิน (SWRL built-ins) ที่กำหนดขึ้นโดยดับเบิลยูสามซี

นอกจากนั้นเอสดับเบิลยูอาร์แอลยังกำหนดโมดูลลาร์ (Modular) ที่เป็นดับเบิลยูอาร์แอลบิวอิน เพื่อช่วยในการปฏิบัติการร่วมกันของเอสดับเบิลยูอาร์แอล (The interoperation of SWRL) โดยดับเบิลยูอาร์แอลบิวอิน แบ่งออกเป็น 7 ประเภท ได้แก่ บิวอินสำหรับการเปรียบเทียบ (Built-ins for comparisons) บิวอินทางคณิตศาสตร์ (Math built-ins) บิวอินสำหรับค่าความเป็นจริง (Built-ins for Boolean values) บิวอินสำหรับสายอักขระ (Built-ins for strings) บิวอินสำหรับวันที่ เวลา ระยะเวลา (Built-ins for date, time, duration) บิวอินสำหรับยูอาร์ไอ (Built-ins for URI) และบิวอินสำหรับรายการ (Built-ins for lists)

จากรูปที่ 2.6 แสดงตัวอย่างการเขียนเอกสารเอสดับเบิลยูอาร์แอล สำหรับอธิบายความสัมพันธ์ของครอบครัวของ x1, x2 และ x3 โดย x2 เป็นพ่อแม่ของ x1 และ x3 เป็นพี่ชายของ x2 แล้ว x3 จะเป็นลุงของ x1

Abstract syntax

hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)

Human readable syntax

```
Implies (    Antecedent(hasParent(I-variable(x1) I-variable(x2))
             hasBrother(I-variable(x2) I-variable(x3)))
             Consequent(hasUncle(I-variable(x1) I-variable(x3))))
```

XML concrete syntax

```
<ruleml:imp>
    <ruleml:_rlab ruleml:href="#example1"/>
    <ruleml:_body>
        <swrlx:individualPropertyAtom swrlx:property="hasParent">
            <ruleml:var>x1</ruleml:var>
            <ruleml:var>x2</ruleml:var>
        </swrlx:individualPropertyAtom>
        <swrlx:individualPropertyAtom swrlx:property="hasBrother">
            <ruleml:var>x2</ruleml:var>
            <ruleml:var>x3</ruleml:var>
        </swrlx:individualPropertyAtom>
    </ruleml:_body>
    <ruleml:_head>
        <swrlx:individualPropertyAtom swrlx:property="hasUncle">
            <ruleml:var>x1</ruleml:var>
            <ruleml:var>x3</ruleml:var>
        </swrlx:individualPropertyAtom>
    </ruleml:_head>
</ruleml:imp>
```

รูปที่ 2.6 ตัวอย่างการเขียนเอกสารดับเบิลยูเอสดีแอล

2.1.6. ตารางตัดสินใจ

ตารางตัดสินใจ [13] สามารถนำมาใช้ในการนำเสนอ และวิเคราะห์ความสัมพันธ์ของตรรกะที่มีความซับซ้อน ซึ่งจำนวนของกลุ่มของการทำงานจะขึ้นอยู่กับเซตของเงื่อนไข โดยตารางตัดสินใจ ประกอบด้วย 4 ส่วน ดังตารางที่ 2.1 ดังนี้

2.1.6.1. คอนดิชันสทับ (Condition stubs) คือ เงื่อนไขต่างๆ ของระบบ

2.1.6.2. คอนดิชันเอนทรี (Condition entries) คือ กฎที่เป็นการรวมความสัมพันธ์ต่างๆ ของคอนดิชันสทับ

2.1.6.3. แอ็คชันสทับ (Action stubs) คือ เหตุการณ์ต่างๆ ที่เกิดขึ้นของระบบ

2.1.6.4. แอ็คชันเอนทรีส์ (Action entries) คือ ส่วนที่บอกว่าเมื่อเงื่อนไขต่างๆ สัมพันธ์กันแล้วจะเกิดเหตุการณ์ใดขึ้น

ตารางที่ 2.1. ส่วนประกอบของตารางตัดสินใจ

Condition stubs	Condition entries
Action stubs	Action entries

ประเภทของตารางตัดสินใจ แบ่งออกเป็น 2 ประเภท ดังนี้

1) ลิมิเทดเอนทรี คือ ตารางการตัดสินใจประเภทที่มีค่าที่เป็นไปได้ในส่วนของคอนดิชันเอนทรีเพียง 2 ค่าเท่านั้น คือ จริง (T) และเท็จ (F) ซึ่งในงานวิจัยนี้จะครอบคลุมเฉพาะตารางการตัดสินใจประเภทนี้

2) เอกซ์เทนเด็ดเอนทรี (Extended entry) คือ ตารางการตัดสินใจประเภทที่มีค่าที่เป็นไปได้ในส่วนของคอนดิชันเอนทรีเกิดขึ้นได้หลายค่า

นอกจากนั้นค่าที่เกิดขึ้นในส่วนของคอนดิชันเอนทรีที่มีค่าความจริงเป็นจริง และค่าความจริงเป็นเท็จแล้วนั้น ยังมีค่าอีก 2 ชนิด ได้แก่ ค่าที่เป็นเท็จเสมอ (Must be Fault: F!) และค่าที่ไม่สนใจว่าจะเป็นจริงหรือเท็จ (Don't Care: -) อีกด้วย ซึ่ง F! จะใช้ในกรณีที่เงื่อนไขก่อนหน้าทำให้เงื่อนไขที่ตามมาเป็นเท็จเสมอ เช่นในกรณีที่ตัวแปรมีหลายช่วง $x < 100$ และ $x > 150$ ถ้าในเงื่อนไขของ $x < 100$ เป็นจริงก็จะแสดงว่าในเงื่อนไขของ $x > 150$ เป็นเท็จเสมอ และในกรณีที่ไม่สนใจว่าค่าจะเป็นจริงหรือเท็จ จะใช้ในกรณีที่เมื่อเงื่อนไขก่อนหน้าเป็นจริงหรือเป็นเท็จก็ตาม แล้วทำให้เงื่อนไขที่ต่อๆ มาไม่ว่าจะเป็นจริงหรือเท็จผลลัพธ์ที่ออกมาก็จะมีค่าเหมือนกันจึงใช้เครื่องหมาย "-" ในเงื่อนไขที่เป็นจริงหรือเท็จก็ได้

2.2. เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1. Automatic Test Data Generation for XML Schema-Based Partition Testing [7]

งานวิจัยนี้พัฒนาเครื่องมือแท็กซี่ (TAXI tool) สำหรับสร้างกรณีทดสอบเพื่อทดสอบเว็บเซอร์วิส โดยใช้เทคนิคการทดสอบเอกซ์เอ็มแอลเบสพาร์ทิชัน สำหรับสร้างเอกซ์เอ็มแอลอินสแตนซส์จากเอกซ์เอ็มแอลสคีมา เพื่อนำเอกซ์เอ็มแอลอินสแตนซส์ที่ได้ไปใช้ในการทดสอบเว็บเซอร์วิสที่ต้องการ แต่จำนวนกรณีทดสอบที่สร้างได้นั้นจะมีจำนวนมาก งานวิจัยนี้จึงเสนอวิธีการถ่วงน้ำหนักสำหรับอิลิเมนต์ ซึ่งงานวิจัยจะไม่คำนึงถึงเงื่อนไขในการเรียกใช้งานเว็บเซอร์วิสและเงื่อนไขของผลลัพธ์ในการทำงานของเว็บเซอร์วิส เนื่องจากเป็นข้อจำกัดของเอกสารดับเบิลยูเอสดีแอล

2.2.2. Extending WSDL to Facilitate Web Services Testing [5]

เนื่องจากดับเบิลยูเอสดีแอล ประกอบด้วยข้อมูลของจำนวนข้อมูลนำเข้าและผลลัพธ์ ชนิดของตัวแปรสำหรับแต่ละข้อมูลนำเข้าและผลลัพธ์ ลำดับของข้อมูลนำเข้าและผลลัพธ์ รวมทั้งวิธีการเรียกใช้งานเว็บเซอร์วิส ดังนั้นงานวิจัยนี้เสนอแนวคิดในการกำหนดอิลิเมนต์เพิ่มเติมเพื่อแก้ไขปัญหาข้อมูลที่ได้จากดับเบิลยูเอสดีแอลที่ไม่ครบถ้วนสำหรับการสร้างกรณีทดสอบ ทำให้กรณีทดสอบที่สร้างขึ้นมานั้นอาจจะไม่สามารถทดสอบเว็บเซอร์วิสได้ครบทุกเหตุการณ์ ซึ่งประเภทของอิลิเมนต์เพิ่มเติมที่กำหนดขึ้นในงานวิจัยมีดังนี้

2.2.2.1. การขึ้นต่อกันระหว่างข้อมูลนำเข้าและผลลัพธ์ (Input-output dependency)

อิลิเมนต์ของข้อมูลประเภทนี้ จะช่วยให้สามารถสร้างกรณีทดสอบได้อย่างมีประสิทธิภาพสำหรับการทดสอบแบบถดถอย (Regression testing) โดยช่วยกำจัดกรณีทดสอบที่ไม่จำเป็นออกไปด้วยการวิเคราะห์ความสัมพันธ์ระหว่างข้อมูลนำเข้าและผลลัพธ์

2.2.2.2. ลำดับการทำงาน (Invocation sequence)

อิลิเมนต์ของข้อมูลประเภทนี้ จะแสดงความสัมพันธ์ในการทำงานระหว่างเว็บเซอร์วิส เนื่องจากเว็บเซอร์วิสสามารถเรียกใช้งานเว็บเซอร์วิสอื่นให้ทำงาน สำหรับบางกิจกรรมที่ต้องการ ดังนั้นอิลิเมนต์นี้จึงช่วยให้สามารถติดตามสถานะการทำงานของเว็บเซอร์วิส

2.2.2.3. คำบรรยายหน้าที่เชิงลำดับชั้น (Hierarchical functional description)

อิลิเมนต์ของข้อมูลประเภทนี้ จะแสดงความสัมพันธ์ระหว่างการทำงานระหว่างเว็บเซอร์วิส ทำให้สามารถวิเคราะห์การทำงานของเว็บเซอร์วิสที่มีความสัมพันธ์ขึ้นอยู่ต่อกัน และสามารถสร้างกรณีทดสอบที่ใช้ในการทดสอบเชิงฟังก์ชัน (Functional testing) และการทดสอบแบบถดถอย

2.2.2.4. ข้อกำหนดเชิงลำดับ (Sequence specification)

อิลิเมนต์ของข้อมูลประเภทนี้ จะแสดงข้อกำหนดเชิงลำดับในการทำงานของเว็บเซอร์วิสที่มีการทำงานร่วมกัน เช่นเดียวกับการทำงานของโปรแกรมเชิงวัตถุ

ผลลัพธ์จากงานวิจัยนี้ คือ สามารถสร้างกรณีทดสอบจากการกำหนดอิลิเมนต์เพิ่มเติมลงในดับเบิลยูเอสดีแอล ทำให้ได้กรณีทดสอบที่เหมาะสมกับการทดสอบเว็บเซอร์วิส แต่งานวิจัยดังกล่าวไม่ได้แสดงวิธีการสร้างกรณีทดสอบจากอิลิเมนต์เพิ่มเติม เพียงแต่แสดงรายละเอียดอิลิเมนต์ที่เพิ่มเติมเท่านั้น และไม่คำนึงถึงเงื่อนไขในการเรียกใช้งานเว็บเซอร์วิสและเงื่อนไขของผลลัพธ์ในการทำงานของเว็บเซอร์วิส เนื่องจากเป็นข้อจำกัดของเอกสารดับเบิลยูเอสดีแอลและอิลิเมนต์ที่เพิ่มเติมนั้น ไม่ได้มีมาตรฐานใดมารองรับ

2.2.3. WSDL-Based Automatic Test Case Generation for Web Services Testing [14]

งานวิจัยนี้เสนอแนวคิดการสร้างกรณีทดสอบ สำหรับเว็บเซอร์วิสอัตโนมัติจากดับเบิลยูเอสดีแอล โดยเริ่มต้นจากการเปลี่ยนแปลงข้อมูลภายในดับเบิลยูเอสดีแอลไปอยู่ในรูปแบบโครงสร้างต้นไม้ของดอม (DOM) ซึ่งกรณีทดสอบจะถูกสร้างจาก 2 ส่วน คือ การสร้างข้อมูลทดสอบ (Test data generation) และการสร้างโอเปอร์เรชันทดสอบ (Test operation generation) โดยการสร้างข้อมูลทดสอบจะวิเคราะห์จากชนิดของข้อมูลแมสเสจ (Message data type) ที่สอดคล้องกับมาตรฐานเอกซ์เอ็มแอล และขั้นตอนของโอเปอร์เรชัน (Operation flow) จะวิเคราะห์จากความสัมพันธ์ระหว่างโอเปอร์เรชันที่ขึ้นต่อกัน ซึ่งแบ่งออกเป็น 3 ประเภท ได้แก่ ข้อมูลนำเข้าที่ขึ้นต่อกัน (Input dependency) ผลลัพธ์ที่ขึ้นต่อกัน (Output dependency) และข้อมูลนำเข้าและผลลัพธ์ที่ขึ้นต่อกัน (Input/output dependency) โดยกรณีทดสอบที่สร้างได้นั้นจะอยู่ในรูปแบบของเอกสารเอกซ์เอ็มแอล และเรียกเอกสารนั้นว่าข้อกำหนดการทดสอบเซอร์วิส (Service test specification)

ผลลัพธ์จากงานวิจัยนี้ คือ สามารถสร้างกรณีทดสอบจากดับเบิลยูเอสดีแอล โดยวิเคราะห์ความสัมพันธ์ระหว่างโอเปอร์เรชัน เพื่อกำหนดลำดับการทดสอบ แต่งานวิจัยดังกล่าวจะ

ไม่คำนึงถึงเงื่อนไขในการเรียกใช้งานเว็บเซอร์วิส และเงื่อนไขของผลลัพธ์ในการทำงานของเว็บเซอร์วิส เนื่องจากเป็นข้อจำกัดของเอกสารดับเบิลยูเอสดีแอล

2.2.4. An Approach for Generating Test Cases from Use Cases Based on A Decision Table [15]

งานวิจัยนี้นำเสนอแนวคิดในการสร้างกรณีทดสอบจากยูสเคสด้วยตารางตัดสินใจแบบลิมิเทดเอนทรี โดยสมมติฐานของตารางตัดสินใจที่สร้างขึ้นนั้น คือ ข้อบกพร่องนั้นเกิดขึ้นจากตัวแปรใดตัวหนึ่งเพียงอันเดียว (Single-fault assumption) ซึ่งกรณีทดสอบที่สร้างได้จะครอบคลุมเหตุการณ์ที่เป็นไปได้ทั้งหมด และสามารถทดสอบเหตุการณ์มากกว่า 1 เหตุการณ์ที่เกิดขึ้น

2.2.5. An Approach for Specification-based Test Case Generation for Web Services [16]

งานวิจัยนี้นำเสนอแนวคิดในการสร้างกรณีทดสอบ สำหรับเว็บเซอร์วิสอัตโนมัติจากดับเบิลยูเอสดีแอล โดยเริ่มต้นจากการเปลี่ยนแปลงข้อมูลนำเข้าของแต่ละโอเปอร์เรชันไปอยู่ในรูปแบบโครงสร้างต้นไม้ ต่อจากนั้นจะทำการสร้างกรณีทดสอบ โดยกรณีทดสอบจะเกิดจากการนำกิ่งแต่ละกิ่งของต้นไม้มาทำการสร้างข้อมูลทดสอบด้วยเทคนิคการทดสอบค่าขอบเขต (Boundary value testing) ดังนั้นจำนวนกรณีทดสอบที่สร้างได้จะเท่ากับผลคูณของจำนวนของค่าขอบเขตแต่ละตัวแปรของข้อมูลนำเข้า