บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 เว็บเซอร์วิส (Web Services)

เว็บเซอร์วิส [1, 2, 3, 4, 5] คือ โปรแกรมประยุกต์ซึ่งทำงานอย่างใดอย่างหนึ่งในลักษณะให้บริการ โดยจะถูกเรียกใช้งานจากโปรแกรมประยุกต์อื่น ๆ ผ่านเครือข่ายอินเทอร์เน็ต และได้กลายเป็นมาตรฐานของดับเบิลยูทรีซี (W3C : World Wide Web Consortium) [22] เพื่อให้ระบบที่มีความหลากหลายสามารถทำงานร่วมกันได้ การให้บริการของเว็บเซอร์วิสจะมีเอกสารที่อธิบายคุณสมบัติของบริการกำกับไว้ และมีการนำเสนอให้แก่สาธารณะรับทราบ ผู้ใช้บริการสามารถค้นหาบริการเว็บเซอร์วิสและเรียกใช้บริการได้โดยไม่ต้องรู้ที่อยู่จริงของโปรแกรมประยุกต์นั้น

2.2 สถาปัตยกรรมของเว็บเซอร์วิส

สถาปัตยกรรมของเว็บเซอร์วิสประกอบด้วย 3 ส่วนคือ

2.2.1 ผู้ให้บริการ (Service Provider) เป็นผู้สร้างเว็บเซอร์วิสและประกาศคำอธิบายของบริการนั้นไปยังตัวแทนผู้ให้บริการ

2.2.2 ตัวแทนผู้ให้บริการ (Service Brokers) เป็นผู้เก็บข้อมูลของการให้บริการจากผู้ให้บริการเพื่อเผยแพร่แก่สาธารณะ

2.2.3 ผู้ขอบริการ (Service Requesters) เป็นผู้ค้นหาบริการที่ต้องการใช้จากตัวแทนผู้ให้บริการ เมื่อพบบริการที่ต้องการก็จะทำการติดต่อขอใช้บริการไปยังผู้ให้บริการ

2.3 เทคโนโลยีแกนหลักสำคัญของเว็บเซอร์วิส

เทคโนโลยีที่เป็นแกนหลักสำคัญของเว็บเซอร์วิสประกอบด้วย 3 ส่วนคือ

2.3.1 โซพ (SOAP : Simple Object Access Protocol) [23] เป็นโพรโทคอลที่ใช้โครงสร้างพื้นฐานของภาษาเอ็กซ์เอ็มแอลเป็นหลัก มีความยืดหยุ่นสำหรับการแลกเปลี่ยนข้อมูลระหว่างโปรแกรมประยุกต์ และสามารถทำงานร่วมกับโพรโทคอลอื่น ๆ ได้หลายชนิด เช่น เอ็ชทีทีพี (HTTP) เอสเอ็มทีพี (SMTP) เป็นต้น ซึ่งสามารถทำงานได้ในทุกระบบปฏิบัติการ

2.3.2 ดับเบิลยูเอสดีแอล (WSDL : Web Services Description Language) [24] เป็นภาษาที่อธิบายคุณลักษณะการให้บริการของเว็บเซอร์วิสและวิธีการติดต่อขอรับบริการจากเว็บเซอร์วิสโดยใช้ไวยากรณ์ของภาษาเอ็กซ์เอ็มแอล ดับเบิลยูเอสดีแอลเป็นภาษาที่อยู่ในความดูแลขององค์กรดับเบิลยูทรีซี

ตัวอย่างเอกสารดับเบิลยูเอสดีแอลของเว็บเซอร์วิสชื่อ StockQuote ซึ่งทำหน้าที่ตรวจสอบราคาซื้อขายสินค้า [24] แสดงดังรูปที่ 2.1

```
1  <?xml version="1.0"?>
2  <definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote.wsdl"
         xmlns:tns="http://example.com/stockquote.wsdl"
         xmlns:xsd1="http://example.com/stockquote.xsd"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns="http://schemas.xmlsoap.org/wsdl/">
3     <types>
4       <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
5         <element name="TradePriceRequest">
6           <complexType>
7                 <element name="tickerSymbol" type="string"/>
8           </complexType>
9         </element>
10        <element name="TradePrice">
11          <complexType>
12                <element name="price" type="float"/>
13          </complexType>
14        </element>
15      </schema>
16    </types>
17    <message name="GetLastTradePriceInput">
18      <part name="body" element="xsd1:TradePriceRequest"/>
19    </message>
20    <message name="GetLastTradePriceOutput">
21      <part name="body" element="xsd1:TradePrice"/>
22    </message>
```

รูปที่ 2.1 เอกสารดับเบิลยูเอสดีแอลของเว็บเซอร์วิส StockQuote

```
23    <portType name="StockQuotePortType">
24      <operation name="GetLastTradePrice">
25        <input message="tns:GetLastTradePriceInput"/>
26        <output message="tns:GetLastTradePriceOutput"/>
27      </operation>
28    </portType>
29    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
30      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
31      <operation name="GetLastTradePrice">
32        <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
33        <input>
34          <soap:body use="literal"/>
35        </input>
36        <output>
37          <soap:body use="literal"/>
38        </output>
39      </operation>
40    </binding>
41    <service name="StockQuoteService">
42      <documentation>My first service</documentation>
43      <port name="StockQuotePort" binding="tns:StockQuoteBinding">
44        <soap:address location="http://example.com/stockquote"/>
45      </port>
46    </service>
47  </definitions>
```

รูปที่ 2.1 เอกสารดับเบิลยูเอสดีแอลของเว็บเซอร์วิส StockQuote (ต่อ)

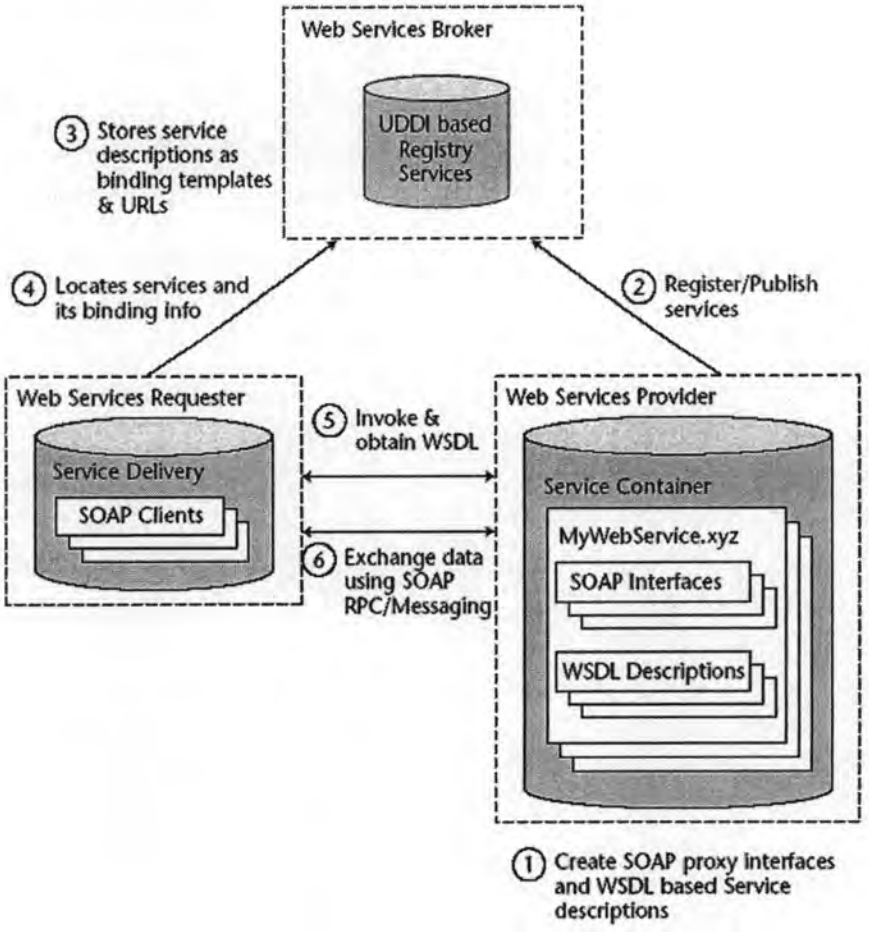
จากรูปที่ 2.1 สามารถอธิบายตัวอย่างเอกสารดับเบิลยูเอสดีแอลของเว็บเซอร์วิส StockQuote ได้ดังนี้

- บรรทัดที่ 41 ชื่อบริการของเว็บเซอร์วิสคือ StockQuoteService
- บรรทัดที่ 44 ตำแหน่งที่อยู่ของเว็บเซอร์วิสที่ให้บริการคือ http://example.com/stockquote

- บรรทัดที่ 24 ชื่อเมท็อดที่ให้บริการคือ GetLastTradePrice
- บรรทัดที่ 25 ประเภทข้อมูลที่นำเข้าของเมท็อด GetLastTradePrice คือ tns:GetLastTradePriceInput จะเชื่อมโยงไปยังบรรทัดที่ 17, 18, 5 และ 7 ซึ่งประกอบด้วยข้อมูลนำเข้าคือ tickerSymbol มีประเภทข้อมูลเป็น string
- บรรทัดที่ 26 ประเภทข้อมูลที่นำออกของเมท็อด GetLastTradePrice คือ tns:GetLastTradePriceOutput จะเชื่อมโยงไปยังบรรทัดที่ 20, 21, 10 และ 12 ซึ่งประกอบด้วยข้อมูลนำออกคือ price มีประเภทข้อมูลเป็น float

2.2.3 ยูดีดีไอ (UDDI : Universal Description, Discovery, and Integration) [25] เป็นมาตรฐานในการค้นหาบริการเว็บเซอร์วิส เปรียบเสมือนฐานข้อมูลขนาดใหญ่ที่เก็บรายละเอียดของเว็บเซอร์วิสที่ให้บริการทางธุรกิจและรอให้ผู้ใช้บริการมาค้นหาบริการ ส่วนผู้ให้บริการต้องนำข้อมูลเกี่ยวกับเว็บเซอร์วิสของตนไปประกาศไว้ที่ยูดีดีไอ

กระบวนการขั้นตอนที่เกิดจากเทคโนโลยีที่เป็นแกนหลักสำคัญข้างต้นก่อให้เกิดการทำงานภายใต้สถาปัตยกรรมของเว็บเซอร์วิสแสดงดังรูปที่ 2.2 [3]

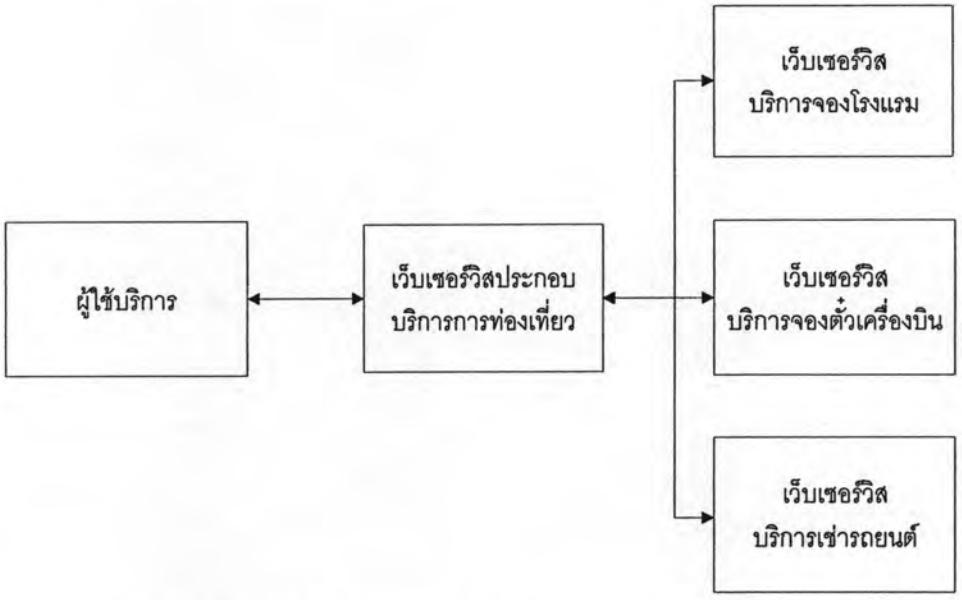


รูปที่ 2.2 กระบวนการขั้นตอนการทำงานของเว็บเซอร์วิส

2.4 เว็บเซอร์วิสประกอบ (Composite Web Services)

เว็บเซอร์วิสประกอบ [6, 7] คือ การรวมบริการของเว็บเซอร์วิสหลายอันเข้าไว้ด้วยกัน โดยมีลำดับและความสัมพันธ์ในการเรียกใช้บริการของเว็บเซอร์วิสต่างๆ ซึ่งสามารถสร้างและควบคุมการทำงานได้โดยใช้ภาษาบีเพล

รูปที่ 2.3 แสดงตัวอย่างบริการการท่องเที่ยวซึ่งเป็นเว็บเซอร์วิสประกอบ ที่เกิดจากการเรียกใช้เว็บเซอร์วิสต่างๆ ได้แก่ เว็บเซอร์วิสบริการจองโรงแรม เว็บเซอร์วิสบริการจองตั๋วเครื่องบิน และเว็บเซอร์วิสบริการเช่ารถยนต์



รูปที่ 2.3 เว็บเซอร์วิสประกอบที่เกิดจากการรวมเว็บเซอร์วิสต่าง ๆ

2.5 บีเพล (BPEL : Business Process Execution Language)

บีเพล [6, 7, 8] คือ ภาษาที่มีโครงสร้างพื้นฐานเป็นภาษาเอ็กซ์เอ็มแอล และเป็นมาตรฐานอยู่ภายใต้การดูแลขององค์กรโอเอซิส (OASIS : Organization for the Advancement of Structured Information Standards) [9] บีเพลเกิดจากการรวมกันของภาษาดับเบิลยูเอสเอฟแอล (WSFL : IBM's Web Service Flow Language) [26] ดับเบิลยูเอสซีไอ (WSCI : BEA Systems' Web Services Choreography Interface) [27] และเอ็กซ์แลง (XLANG) [28] ซึ่งใช้สำหรับการสร้างเว็บเซอร์วิสประกอบ

เว็บเซอร์วิสประกอบที่สร้างด้วยภาษาบีเพลโดยใช้เครื่องมือออราเคิลบีเพลดีไซเนอร์ประกอบด้วยแฟ้มข้อมูลต่อไปนี้

2.5.1 แฟ้มข้อมูลบีเพล (นามสกุลคือ .bpel) ใช้อธิบายกระบวนการทำงานของเว็บเซอร์วิสต่างๆ ที่มาประกอบกันเพื่อทำงานร่วมกัน

2.5.2 แฟ้มข้อมูลดับเบิลยูเอสดีแอล (นามสกุลคือ .wsdl) ใช้อธิบายส่วนต่อประสานของเว็บเซอร์วิสประกอบ (จะอยู่ภายในไดเร็กทอรีเดียวกันกับแฟ้มข้อมูลบีเพล)

2.5.3 แฟ้มข้อมูลเอ็กซ์เอ็มแอล (bpel.xml) ใช้บอกตำแหน่งที่อยู่ของเอกสารดับเบิลยูเอสดีแอลของเว็บเซอร์วิสที่มาประกอบกัน (จะอยู่ภายในไดเร็กทอรีเดียวกันกับแฟ้มข้อมูลบีเพล)

2.6 โครงสร้างของภาษาบีเพล

โครงสร้างของภาษาบีเพลตามข้อกำหนดของภาษาบีเพลรุ่นที่ 1.1 แสดงได้ดังรูปที่ 2.4

```
01  <process name="ncname" targetNamespace="uri"
        queryLanguage="anyURI"?
        expressionLanguage="anyURI"?
        suppressJoinFailure="yes|no"?
        enableInstanceCompensation="yes|no"?
        abstractProcess="yes|no"?
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
02      <partnerLinks>?
03          <partnerLink name="ncname" partnerLinkType="qname"
                myRole="ncname"? partnerRole="ncname"?>+
04          </partnerLink>
05      </partnerLinks>
06      <partners>?
07          <partner name="ncname">+
08              <partnerLink name="ncname"/>+
09          </partner>
10      </partners>
11      <variables>?
12          <variable name="ncname" messageType="qname"?
                type="qname"? element="qname"?/>+
13      </variables>
14      <correlationSets>?
15          <correlationSet name="ncname" properties="qname-list"/>+
16      </correlationSets>
```

รูปที่ 2.4 โครงสร้างของภาษาบีเพลรุ่นที่ 1.1

```
17          <faultHandlers>?
18              <catch faultName="qname"? faultVariable="ncname"?>*
19                  activity
20              </catch>
21              <catchAll>?
22                  activity
23              </catchAll>
24          </faultHandlers>
25          <compensationHandler>?
26              activity
27          </compensationHandler>
28          <eventHandlers>?
29              <onMessage partnerLink="ncname" portType="qname"
                    operation="ncname" variable="ncname"?>
30                  <correlations>?
31                      <correlation set="ncname" initiate="yes|no"?>+
32                  <correlations>
33                  activity
34              </onMessage>
35              <onAlarm for="duration-expr"? until="deadline-expr"?>*
36                  activity
37              </onAlarm>
38          </eventHandlers>
39          activity
40  </process>
```

รูปที่ 2.4 โครงสร้างของภาษาบีเพลรุ่นที่ 1.1 (ต่อ)

จากรูปที่ 2.4 โครงสร้างส่วนต่างๆ ของภาษาบีเพลที่งานวิจัยนี้พิจารณานำมาใช้งาน สามารถอธิบายได้ดังนี้

- บรรทัดที่ 1 ป้ายระบุ <process> คือจุดเริ่มต้นของกระบวนการ แสดงชื่อบริการเว็บเซอร์วิสประกอบ
- บรรทัดที่ 2 – 5 ป้ายระบุ <partnerLink> คือการกำหนดเว็บเซอร์วิสที่นำมาประกอบกันเพื่อการเรียกใช้งาน

- บรรทัดที่ 11 – 13 ป้ายระบุ <variable> คือการกำหนดตัวแปรที่เกี่ยวข้องเพื่อการใช้งาน
- บรรทัดที่ 39 โทเค็น *activity* คือกิจกรรมที่เกิดขึ้นในกระบวนการทำงานสำหรับงานวิจัยนี้ ในส่วนของโทเค็น *activity* จะพิจารณาคำสั่ง 6 คำสั่งที่ใช้สำหรับสร้างวิถีการทำงานพื้นฐานสำหรับบีเพล ได้แก่

1) คำสั่ง <invoke> ใช้สำหรับเรียกใช้บริการเว็บเซอร์วิสย่อย สามารถเรียกได้ทั้งแบบสมวารและอสมวาร (สำหรับงานวิจัยนี้ เรียกใช้เฉพาะแบบสมวาร) โครงสร้างของคำสั่งนี้แสดงได้ดังรูปที่ 2.5

2) คำสั่ง <receive> ใช้สำหรับรับคำสั่งจากไคลเอนต์ (Client) เพื่อเรียกใช้เว็บเซอร์วิสประกอบ โครงสร้างของคำสั่งนี้แสดงได้ดังรูปที่ 2.6

3) คำสั่ง <assign> ใช้สำหรับจัดการข้อมูลของตัวแปรที่เกี่ยวข้อง โครงสร้างของคำสั่งนี้แสดงได้ดังรูปที่ 2.7

4) คำสั่ง <reply> ใช้สำหรับส่งค่ากลับให้ไคลเอนต์หลังจากเรียกใช้บริการเว็บเซอร์วิสประกอบ โครงสร้างของคำสั่งนี้แสดงได้ดังรูปที่ 2.8

5) คำสั่ง <switch> ใช้สำหรับกำหนดทางเลือกในการทำงานของกระบวนการ โครงสร้างของคำสั่งนี้แสดงได้ดังรูปที่ 2.9

6) คำสั่ง <while> ใช้สำหรับกำหนดการวนซ้ำในการทำงานของกระบวนการ โครงสร้างของคำสั่งนี้แสดงได้ดังรูปที่ 2.10

```
<invoke partnerLink="ncname" portType="qname" operation="ncname"
        inputVariable="ncname"? outputVariable="ncname"?
        standard-attributes>
        standard-elements
        <correlations>?
            <correlation set="ncname" initiate="yes|no"?
                pattern="in|out|out-in"/>+
        </correlations>
        <catch faultName="qname" faultVariable="ncname"?>*
            activity
        </catch>
```

รูปที่ 2.5 โครงสร้างของคำสั่ง <invoke>

```
        <catchAll>?
            activity
        </catchAll>
        <compensationHandler>?
            activity
        </compensationHandler>
</invoke>
```

รูปที่ 2.5 โครงสร้างของคำสั่ง <invoke> (ต่อ)

```
<receive partnerLink="ncname" portType="qname" operation="ncname"
        variable="ncname"? createInstance="yes|no"?
        standard-attributes>
        standard-elements
        <correlations>?
            <correlation set="ncname" initiate="yes|no"?>+
        </correlations>
</receive>
```

รูปที่ 2.6 โครงสร้างของคำสั่ง <receive>

```
<assign standard-attributes>
    standard-elements
    <copy>+
        from-spec
        to-spec
    </copy>
</assign>
```

รูปที่ 2.7 โครงสร้างของคำสั่ง <assign>

```
<reply partnerLink="ncname" portType="qname" operation="ncname"
      variable="ncname"? faultName="qname"?
      standard-attributes>
      standard-elements
      <correlations>?
            <correlation set="ncname" initiate="yes|no"?>+
      </correlations>
</reply>
```

รูปที่ 2.8 โครงสร้างของคำสั่ง <reply>

```
<switch standard-attributes>
      standard-elements
      <case condition="bool-expr">+
            activity
      </case>
      <otherwise>?
            activity
      </otherwise>
</switch>
```

รูปที่ 2.9 โครงสร้างของคำสั่ง <switch>

```
<while condition="bool-expr" standard-attributes>
      standard-elements
      activity
</while>
```

รูปที่ 2.10 โครงสร้างของคำสั่ง <while>

2.7 ตัวอย่างแฟ้มข้อมูลบีเพล

รูปที่ 2.11 แสดงตัวอย่างแฟ้มข้อมูลบีเพลซึ่งเป็นเว็บเซอร์วิสประกอบที่ให้บริการการขออนุมัติเงินกู้ (LoanApproval) [11] โดยมีการเรียกใช้เว็บเซอร์วิสต่างๆ ได้แก่ เว็บเซอร์วิสลูกค้า (Customer) ซึ่งให้ข้อมูลเกี่ยวกับลูกค้าและวงเงินที่จะขอกู้ เว็บเซอร์วิสผู้ประเมิน (Assessor) ซึ่งทำหน้าที่ประเมินความเสี่ยงในการให้กู้เงิน และเว็บเซอร์วิสผู้อนุมัติ (Approval) ซึ่งทำหน้าที่พิจารณาอนุมัติเงินกู้ในกรณีที่เป็นการกู้วงเงินสูงหรือมีความเสี่ยงสูง การทำงานเริ่มจากการรับอินพุตจากผู้ใช้บริการ แล้วเรียกใช้งานเว็บเซอร์วิสต่างๆ แล้วรอผลลัพธ์ที่ได้ สามารถอธิบายการทำงานของบีเพลแต่ละส่วนได้ดังรูปที่ 2.12

```
01  <process name="LoanApproval" targetNamespace="http://acm.org/samples"
    suppressJoinFailure="yes" xmlns:tns="http://acm.org/samples"
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
    xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
    xmlns:ora="http://schemas.oracle.com/xpath/extension"
    xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
02      <partnerLinks>
03          <partnerLink name="client" partnerLinkType="tns:LoanApproval"
    myRole="LoanApprovalProvider"/>
04          <partnerLink name="Customer" partnerLinkType="tns:Customer"
    partnerRole="CustomerProvider"/>
05          <partnerLink name="Assessor" partnerLinkType="tns:Assessor"
    partnerRole="AssessorProvider"/>
06          <partnerLink name="Approval" partnerLinkType="tns:Approval"
    partnerRole="ApprovalProvider"/>
07      </partnerLinks>
08      <variables>
09          <variable name="input" messageType="tns:LoanApprovalRequestMessage"/>
10          <variable name="output" messageType="tns:LoanApprovalResponseMessage"/>
11          <variable name="cInput" messageType="tns:CustomerRequestMessage"/>
12          <variable name="cOutput" messageType="tns:CustomerResponseMessage"/>
13          <variable name="aInput" messageType="tns:AssessorRequestMessage"/>
```

รูปที่ 2.11 ตัวอย่างแฟ้มข้อมูลบีเพลเว็บเซอร์วิส LoanApproval

```
14        <variable name="aOutput" messageType="tns:AssessorResponseMessage"/>
15        <variable name="pInput" messageType="tns:ApprovalRequestMessage"/>
16        <variable name="pOutput" messageType="tns:ApprovalResponseMessage"/>
17    </variables>
18    <sequence name="main">
19        <receive name="receiveInput" partnerLink="client" portType="tns:LoanApproval"
operation="process" variable="input" createInstance="yes"/>
20        <assign name="assignCustomerID">
21            <copy>
22                <from variable="input" part="payload"
query="/tns:LoanApprovalRequest/tns:input"></from>
23                <to variable="cInput" part="payload"
query="/tns:CustomerRequest/tns:name"/>
24            </copy>
25        </assign>
26        <invoke name="invokeCustomer" partnerLink="Customer" portType="tns:Customer"
operation="process" inputVariable="cInput" outputVariable="cOutput"/>
27        <switch name="checkAmount">
28            <case
condition="(bpws:getVariableData('cOutput','payload','/tns:CustomerResponse/tns:amount')
&amp;lt; 10000)">
29                <sequence>
30                    <assign name="assignCustomerData">
31                        <copy>
32                            <from variable="cOutput" part="payload"
query="/tns:CustomerResponse/tns:data"></from>
33                            <to variable="aInput" part="payload"
query="/tns:AssessorRequest/tns:data"/>
34                        </copy>
35                    </assign>
36                    <assign name="assignCustomerAmount">
37                        <copy>
```

รูปที่ 2.11 ตัวอย่างแฟ้มข้อมูลบีเพลเว็บเซอร์วิส LoanApproval **(ต่อ)**

```
38                          <from variable="cOutput" part="payload"
query="/tns:CustomerResponse/tns:amount"></from>
39                          <to variable="aInput" part="payload"
query="/tns:AssessorRequest/tns:amount"/>
40                      </copy>
41                  </assign>
42                  <invoke name="invokeAssessor" partnerLink="Assessor"
portType="tns:Assessor" operation="process" inputVariable="aInput"
outputVariable="aOutput"/>
43                  <switch name="checkRisk">
44                      <case
condition="(bpws:getVariableData('aOutput','payload','/tns:AssessorResponse/tns:result') =
'Low')">
45                          <sequence>
46                              <assign name="assignApproved">
47                                  <copy>
48                                      <from expression="'Yes'"></from>
49                                      <to variable="output" part="payload"
query="/tns:LoanApprovalResponse/tns:result"/>
50                                  </copy>
51                              </assign>
52                          </sequence>
53                      </case>
54                      <otherwise>
55                          <sequence>
56                          <assign name="assignCustomerData">
57                              <copy>
58                                  <from variable="cOutput" part="payload"
query="/tns:CustomerResponse/tns:data"></from>
59                                  <to variable="pInput" part="payload"
query="/tns:ApprovalRequest/tns:data"/>
60                              </copy>
```

รูปที่ 2.11 ตัวอย่างแฟ้มข้อมูลบีเพลเว็บเซอร์วิส LoanApproval (ต่อ)

```
61                          </assign>
62                          <assign name="assignCustomerAmount">
63                              <copy>
64                                  <from variable="cOutput" part="payload"
    query="/tns:CustomerResponse/tns:amount"></from>
65                                  <to variable="pInput" part="payload"
    query="/tns:ApprovalRequest/tns:amount"/>
66                              </copy>
67                          </assign>
68                          <invoke name="invokeApproval" partnerLink="Approval"
    portType="tns:Approval" operation="process" inputVariable="pInput"
    outputVariable="pOutput"/>
69                          <assign name="assignResult">
70                              <copy>
71                                  <from variable="pOutput" part="payload"
    query="/tns:ApprovalResponse/tns:result"></from>
72                                  <to variable="output" part="payload"
    query="/tns:LoanApprovalResponse/tns:result"/>
73                              </copy>
74                          </assign>
75                      </sequence>
76                  </otherwise>
77              </switch>
78          </sequence>
79      </case>
80      <otherwise>
81          <sequence>
82              <assign name="assignCustomerData">
83                  <copy>
84                      <from variable="cOutput" part="payload"
    query="/tns:CustomerResponse/tns:data"></from>
85                      <to variable="aInput" part="payload"
    query="/tns:AssessorRequest/tns:data"/>
```

รูปที่ 2.11 ตัวอย่างแฟ้มข้อมูลบีเพลเว็บเซอร์วิส LoanApproval (ต่อ)

I23065825

```
86                  </copy>
87                </assign>
88                <assign name="assignCustomerAmount">
89                  <copy>
90                    <from variable="cOutput" part="payload"
query="/tns:CustomerResponse/tns:amount"></from>
91                    <to variable="aInput" part="payload"
query="/tns:AssessorRequest/tns:amount"/>
92                  </copy>
93                </assign>
94                <invoke name="invokeApproval" partnerLink="Approval"
portType="tns:Approval" operation="process" inputVariable="pInput"
outputVariable="pOutput"/>
95                <assign name="assignResult">
96                  <copy>
97                    <from variable="pOutput" part="payload"
query="/tns:ApprovalResponse/tns:result"></from>
98                    <to variable="output" part="payload"
query="/tns:LoanApprovalResponse/tns:result"/>
99                  </copy>
100               </assign>
101             </sequence>
102         </otherwise>
103         </switch>
104         <reply name="replyOutput" partnerLink="client" portType="tns:LoanApproval"
operation="process" variable="output"/>
105     </sequence>
106 </process>
```

รูปที่ 2.11 ตัวอย่างแฟ้มข้อมูลบีเพลเว็บเซอร์วิส LoanApproval (ต่อ)

	Outline of the process	BPEL
01	Reference to the Web Services	2 – 7
02	Variables Declaration	8 – 17
03	Receive input from Client	19
04	Assign input to cInput	20 – 25
05	Invoke Customer Service with cInput and receive cOutput	26
06	Switch Case (cOutput.amount < 10000)	27 – 28
07	Assign cOutput to aInput	30 – 41
08	Invoke Assessor Service with aInput and receive aOutput	42
09	Switch Case (aOutput = "low")	43 – 44
10	Assign "Yes" to output	46 – 51
11	Otherwise	54
12	Assign cOutput to pInput	56 – 67
13	Invoke Approval Service with pInput and receive pOutput	68
14	Assign pOutput to output	69 – 74
15	End Switch Case	77
16	Otherwise	80
17	Assign cOutput to pInput	82 – 93
18	Invoke Approval Service with pInput and receive pOutput	94
19	Assign pOutput to output	95 – 100
20	End Switch Case	103
21	Reply output to Client	104

รูปที่ 2.12 กระบวนการทำงานของบีเพล (จากรูปที่ 2.11)

2.8 การทดสอบซอฟต์แวร์ (Software Testing)

การทดสอบซอฟต์แวร์ [12, 13] คือ กระบวนการที่แสดงให้เห็นว่าพฤติกรรมของโปรแกรมตรงกับพฤติกรรมที่ได้คาดหวังไว้ โดยใช้กรณีทดสอบ (Test Cases)

การทดสอบซอฟต์แวร์มีกลยุทธ์ของการทดสอบ 2 แบบ คือ การทดสอบแบบกล่องดำ และการทดสอบแบบกล่องขาว แบ่งระดับของการทดสอบเป็น 3 ระดับ คือ การทดสอบระดับหน่วย การทดสอบแบบบูรณาการ และการทดสอบระบบ การทดสอบแต่ละแบบและทฤษฎีที่เกี่ยวข้องกับการทดสอบซอฟต์แวร์มีรายละเอียดดังนี้

2.8.1 การทดสอบแบบกล่องดำ (Black Box Testing)

การทดสอบแบบกล่องดำ [12, 13] คือ วิธีการทดสอบโดยผู้ทดสอบไม่รู้โครงสร้างภายในของโปรแกรม ดังนั้นจึงเป็นการทดสอบฟังก์ชันการทำงาน (Function Testing) ว่าสามารถทำงานได้ตามข้อกำหนดความต้องการของโปรแกรมหรือไม่ โดยสามารถออกแบบและสร้างกรณีทดสอบได้จากการกำหนดอินพุตของโปรแกรมเพื่อทดสอบการทำงานเพื่อให้ได้เอาต์พุตตามที่ได้คาดหวังไว้

2.8.2 การทดสอบแบบกล่องขาว (White Box Testing)

การทดสอบแบบกล่องขาว [12, 13] คือ วิธีการทดสอบโดยผู้ทดสอบรู้โครงสร้างภายในของโปรแกรม (Structure Testing) และสามารถออกแบบและสร้างกรณีทดสอบได้จากรหัสคำสั่งของโปรแกรม ผู้ทดสอบเป็นผู้เลือกกรณีทดสอบเพื่อทดสอบให้ครอบคลุมการทำงานตามที่ต้องการ เช่น การทดสอบเพื่อครอบคลุมทุกข้อความสั่ง (Statement Testing) การทดสอบเพื่อครอบคลุมทุกคำสั่งแยกทาง (Branch Testing) การทดสอบเพื่อครอบคลุมทุกวิถีการทำงาน (Path Testing) และการทดสอบวิถีการทำงานพื้นฐาน (Basis Path Testing) เป็นต้น

การเปรียบเทียบวิธีการทดสอบระหว่างแบบกล่องดำและแบบกล่องขาวแสดงได้ดังตารางที่ 2.1

ในงานวิจัยนี้ผู้วิจัยเลือกวิธีการทดสอบวิถีการทำงานพื้นฐานมาใช้ในการทดสอบเว็บเซอร์วิสประกอบที่สร้างด้วยภาษาบีเพล ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อถัดไป

ตารางที่ 2.1 การเปรียบเทียบวิธีการทดสอบระหว่างแบบกล่องดำและแบบกล่องขาว

Test Strategy	Knowledge Sources	Methods
Black box	Requirements document Specifications Domain knowledge Defect analysis data	Equivalence class partitioning Boundary value analysis Cause and effect graphing Error guessing
White box	High-level design Detailed design Control flow graphs Cyclomatic complexity	Statement testing Branch testing Path testing Data flow testing Mutation testing Loop testing

2.8.3 กราฟควบคุมสายงาน

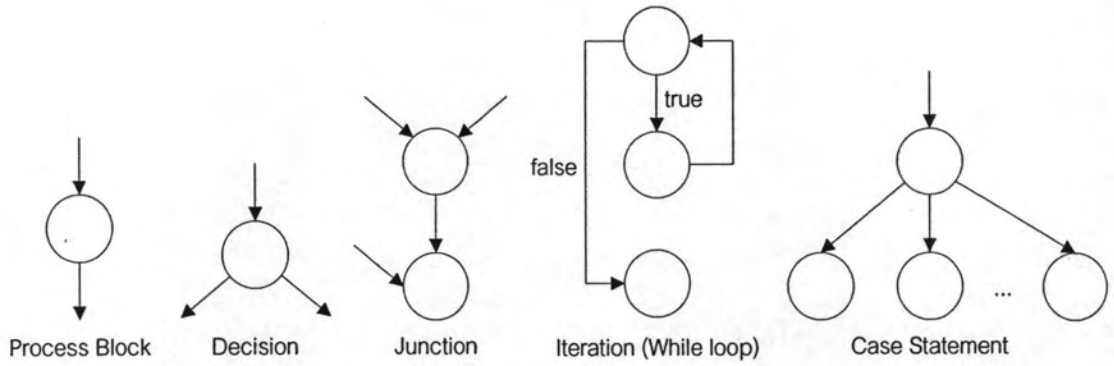
กราฟควบคุมสายงาน [12, 13] คือ ตัวแทนของโครงสร้างการควบคุมการทำงานของโปรแกรม มีลักษณะเหมือนกับผังงาน (Flowchart) ซึ่งมีส่วนประกอบ 5 ส่วนคือ

1) บล็อกกระบวนการ (Process Block)
2) การตัดสินใจ (Decision)
3) จุดต่อ (Junction)
4) ทางเลือก (Case Statement)
5) การวนซ้ำ (Iteration)

ส่วนประกอบต่าง ๆ แสดงได้ดังรูปที่ 2.13 และตัวอย่างการสร้างกราฟควบคุมสายงานจากรหัสคำสั่งของโปรแกรมแสดงได้ดังรูปที่ 2.14

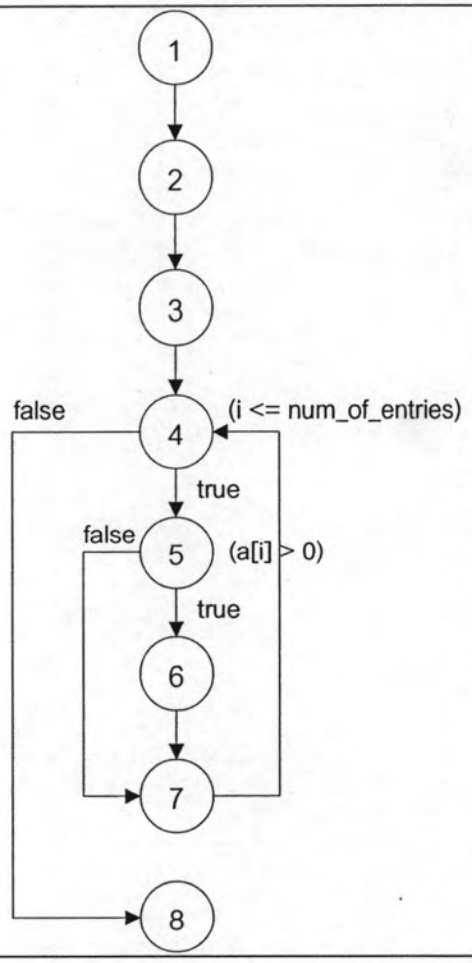
ในงานวิจัยนี้ใช้ส่วนประกอบของกราฟควบคุมสาย 3 ส่วนได้แก่ บล็อกกระบวนการ การตัดสินใจ และการวนซ้ำ


หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 2.13 ส่วนประกอบของกราฟควบคุมสายงาน

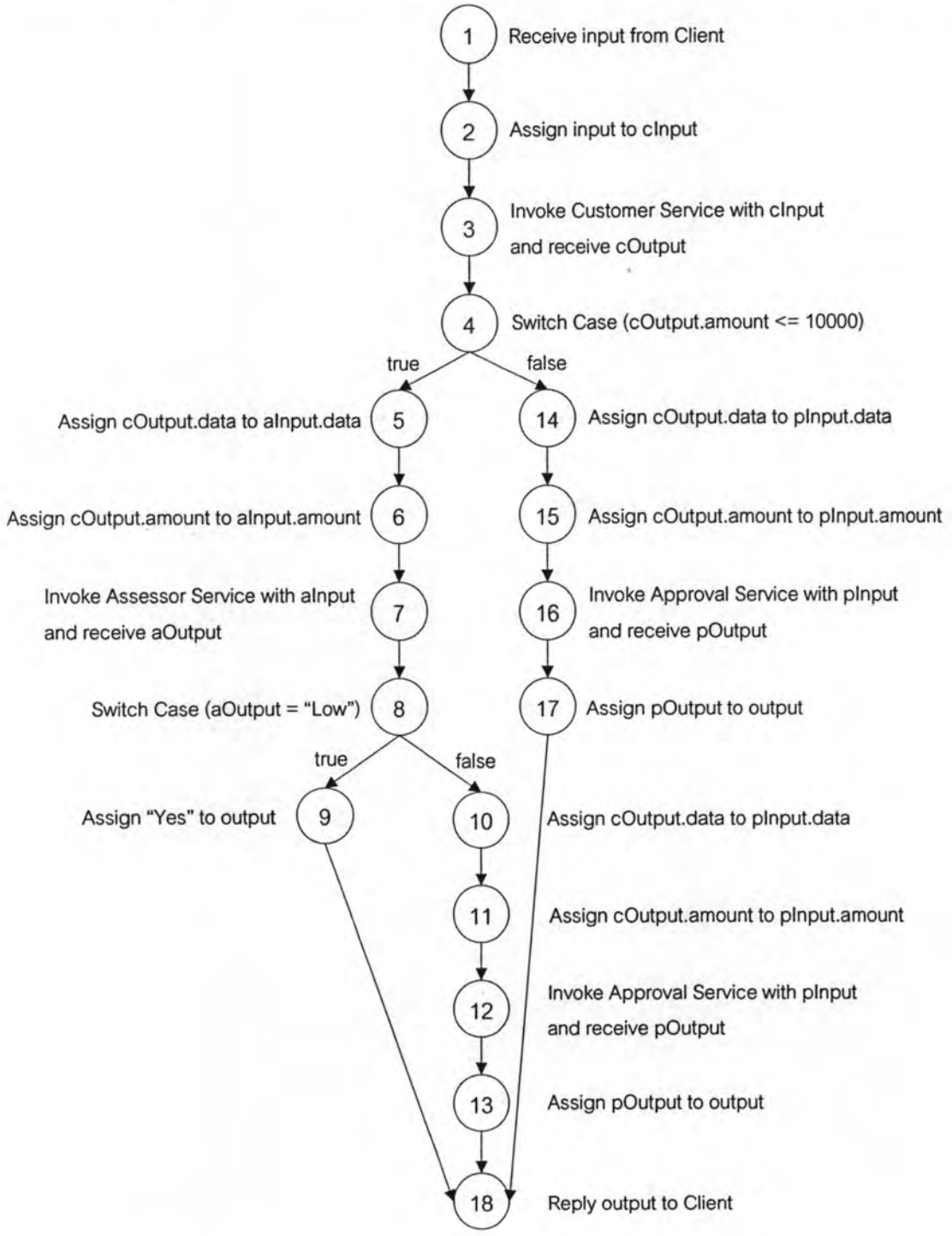
รหัสคำสั่งของโปรแกรม	กราฟควบคุมสายงาน

1. pos_sum(a, num_of_entries, sum)
2. sum = 0
3. int i = 1
4. while (i<= num_of_entries)
5. if a[i] > 0
6. sum = sum + a[i]
end if
7. i = i + 1
end while
8. end pos_sum
	

รูปที่ 2.14 ตัวอย่างการสร้างกราฟควบคุมสายงานจากรหัสคำสั่งของโปรแกรม

จากรูปที่ 2.15 แสดงกราฟควบคุมสายงานที่สร้างจากเอกสารจากบีเพลตามตัวอย่างแฟ้มข้อมูลบีเพลในรูปที่ 2.11

ในงานวิจัยนี้ผู้วิจัยจะสร้างกราฟควบคุมสายงานจากเอกสารบีเพลของเว็บเซอร์วิสประกอบแล้วนำไปสร้างกรณีทดสอบสำหรับทดสอบวิถีการทำงานพื้นฐานของเว็บเซอร์วิสประกอบ



รูปที่ 2.15 กราฟควบคุมสายงานที่สร้างจากเอกสารบีเพลตามตัวอย่างแฟ้มข้อมูลบีเพลในรูปที่ 2.11

2.8.4 การทดสอบวิถีการทำงานพื้นฐาน (Basis Path Testing)

การทดสอบวิถีการทำงานพื้นฐานเป็นวิธีการทดสอบการทำงานของโปรแกรมวิธีหนึ่งภายใต้การทดสอบแบบโครงสร้าง และเป็นวิธีการทดสอบแบบกล่องขาว คือสามารถมองเห็นการทำงานภายในของโปรแกรม การทดสอบวิถีการทำงานทำได้โดยการพิจารณากราฟควบคุมสายงาน (Control Flow Graph) ที่สร้างได้จากรหัสคำสั่งของโปรแกรม

ในงานวิจัยนี้ผู้วิจัยใช้มาตรวัดไซโคลเมติกของแมคเคบ (McCabe's Cyclomatic Complexity – V(G)) [29] ในการกำหนดจำนวนกรณีทดสอบที่ต้องสร้างขึ้นเพื่อทดสอบวิถีการทำงานพื้นฐาน (Basis Path Testing) โดยมีสูตรในการคำนวณจากกราฟควบคุมสายงานคือ

V(G)	=	E – N + 2
V(G)	คือ	จำนวนวิถีอิสระ (Independent Paths) หรือจำนวนกรณีทดสอบ
E	คือ	จำนวนเส้นเชื่อมของกราฟ
N	คือ	จำนวนจุดต่อของกราฟ

วิถีอิสระของกราฟ หมายถึง วิถีของกราฟที่มีเส้นเชื่อมของกราฟอย่างน้อยหนึ่งเส้นเชื่อมที่ไม่เคยผ่านมาก่อน

ผลที่ได้จากการวัดคือจำนวนวิถีอิสระของกราฟหรือจำนวนกรณีทดสอบสำหรับการทดสอบวิถีการทำงานพื้นฐานของเว็บเซอร์วิสประกอบ

จากกราฟควบคุมสายงานในรูปที่ 2.15 สามารถคำนวณหาจำนวนกรณีทดสอบได้ดังนี้

V(G)	=	E – N + 2
	=	19 – 18 + 2
	=	3

จากการคำนวณจะได้กรณีทดสอบทั้งหมด 3 กรณีทดสอบเพื่อทดสอบวิถีการทำงานพื้นฐานได้แก่

วิถีที่ 1 คือ 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 18

วิถีที่ 2 คือ 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 11 –12 – 13 – 18

วิถีที่ 3 คือ 1 – 2 – 3 – 4 – 14 – 15 – 16 – 17 – 18

2.8.5 การทดสอบระดับหน่วย (Unit Testing)

การทดสอบระดับหน่วย คือ การทดสอบการทำงานของซอฟต์แวร์ในระดับมอดูล (Module) เพื่อทดสอบการทำงานของแต่ละมอดูลว่าสามารถทำงานได้ถูกต้องหรือไม่

2.8.6 การทดสอบระดับบูรณาการ (Integration Testing)

การทดสอบระดับบูรณาการ คือ การทดสอบระดับบูรณาการ คือ การทดสอบส่วนต่อประสานระหว่างมอดูลภายในซอฟต์แวร์ ซึ่งจะทดสอบหลังจากที่ได้ทดสอบระดับหน่วยแล้ว โดยสามารถแบ่งกลวิธีในการทดสอบได้หลายแบบ เช่น การทดสอบจากบนลงล่าง (Top Down Testing) เป็นการทดสอบส่วนต่อประสานจากมอดูลระดับบนลงไปมอดูลระดับล่าง การทดสอบจากล่างขึ้นบน (Bottom Up Testing) เป็นการทดสอบส่วนต่อประสานจากมอดูลระดับล่างขึ้นไปมอดูลระดับบน และการทดสอบแบบบิ๊กแบง (Big Bang Testing) เป็นการทดสอบส่วนต่อประสานของการรวมทุกมอดูล เป็นต้น

2.8.7 การทดสอบระบบ

การทดสอบระบบ คือ การทดสอบว่าซอฟต์แวร์มีการทำงานเป็นไปตามความต้องการของระบบ

2.8.8 กรณีทดสอบ

กรณีทดสอบ คือ เอกสารสำหรับทดสอบการทำงานของซอฟต์แวร์ ซึ่งใช้สำหรับแสดงให้เห็นพฤติกรรมของโปรแกรม ประกอบด้วย เงื่อนไขก่อน (Pre-conditions) อินพุต เอาต์พุตที่คาดว่าจะได้รับ และเงื่อนไขหลัง (Post-conditions)

2.9 เอ็กซ์เอ็มแอล (XML: eXtensible Markup Language)

เอ็กซ์เอ็มแอล [10] เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถนิยามความหมายของข้อมูลได้ (Data Definition) เอกสารเอ็กซ์เอ็มแอลมีลักษณะเป็นข้อความธรรมดาแล้วปิดล้อมด้วยแท็ก (Tag) หรืออีลิเมนต์ ซึ่งแท็กต่างๆ นั้นผู้ใช้จะเป็นคนกำหนดขึ้นมาเอง และสามารถนำเอกสารเอ็กซ์เอ็มแอลนี้ไปใช้เป็นต้นแบบในการนิยามข้อมูล เพื่อใช้ในงานต่างๆ ได้อีกด้วย ซึ่งเอกสารเอ็กซ์เอ็มแอลนี้ต้องอยู่ในรูปแบบที่ถูกต้อง (Well-Formedness) ดังนี้

2.9.1 เอกสารเอ็กซ์เอ็มแอล จะมีอีลิเมนต์แม่ (Root Element) ได้เพียงอีลิเมนต์เดียวเท่านั้น ซึ่งทำหน้าที่คลุมอีลิเมนต์อื่นทั้งหมด จากรูปที่ 2.16 อีลีเมนต์แม่คือ <testsuite></testsuite>

2.9.2 ทุกๆ อีลิเมนต์จะต้องมีแท็กเปิด และแท็กปิดเหมือนกัน เช่น <testcase></testcase> ดังรูปที่ 2.16

2.9.3 ห้ามระบุแท็กเหลื่อมซ้อนกัน ถ้าแท็กไหนเปิดก่อนจะต้องปิดทีหลัง เช่นมีแท็ก <input> ก่อน <output> จะต้องปิดแท็ก </input> ก่อนแท็ก </output> ดังรูปที่ 2.16

2.9.4 ชื่อแท็กตัวอักษรตัวพิมพ์เล็ก และพิมพ์ใหญ่ถือว่าไม่เหมือนกัน (Case-Sensitive) เช่น <input> กับ <INPUT> จะไม่เหมือนกัน

2.9.5 ค่าของแอตทริบิวต์ (Attribute) ต้องอยู่ในเครื่องหมายคำพูด Single Quote (') หรือ Double Quote (") อย่างใดอย่างหนึ่ง เช่น <testcas no="1"> ดังรูปที่ 2.16

2.9.6 ถ้ามีเอนทิที (Entity) ใดที่นำมาใช้จากเอกสารอื่นต้องทำการอ้างอิง (Reference) ไว้ก่อนนำมาใช้

```
<?xml version="1.0"?>
<testsuite service="LoanApproval">
    <testcase no="1">
        <input>
            <variable name="cOutput" service="Customer">
                <element name="name" type="string" value="Peter">
            </variable>
        </input>
        <output>
            <variable name="output" service="LoanApproval">
                <element name="result" type="string" value="Yes">
            </variable>
        </output>
    </testcase>
</testsuite>
```

รูปที่ 2.16 ตัวอย่างเอกสารเอ็กซ์เอ็มแอล

2.10 งานวิจัยที่เกี่ยวข้อง

2.10.1 การเพิ่มเติมเอกสารดับเบิลยูเอสดีแอลเพื่อช่วยในการทดสอบเว็บเซอร์วิส (Extending WSDL to Facilitate Web Services Testing) [14] โดย W. T. Tsai, Ray Paul, Yamin Wang, Chun Fan และ Dong Wang

งานวิจัยนี้ได้นำเสนอข้อมูลที่ต้องการเพิ่มเติมในเอกสารดับเบิลยูเอสดีแอลเพื่อช่วยในการทดสอบเว็บเซอร์วิส โดยแบ่งประเภทข้อมูลเป็น 4 ประเภทคือ

1) ความสัมพันธ์ระหว่างอินพุตเอาต์พุต (Input-output Dependency) เป็นข้อมูลที่ช่วยในการทดสอบแบบถดถอย (Regression Testing) และการทดสอบสายงานข้อมูล (Data Flow Testing) และช่วยลดการสร้างกรณีทดสอบที่ไม่จำเป็น

2) ลำดับการร้องขอ (Invocation Sequence) เป็นข้อมูลเกี่ยวกับลำดับในการเรียกใช้เมท็อด (Method) ช่วยในการสร้างการทดสอบวิถีการทำงาน และการทดสอบสายงานข้อมูล

3) ลำดับชั้นของฟังก์ชัน (Hierarchical Functional Description) เป็นข้อมูลความสัมพันธ์ในลักษณะโครงสร้างต้นไม้ ช่วยให้การทดสอบเป็นไปแบบอัตโนมัติได้

4) ข้อกำหนดลำดับการประมวลผล (Sequence Specifications) เป็นข้อมูลสำหรับกำหนดลำดับในการประมวลผล

สรุปได้ว่า ข้อมูลที่มีอยู่ในเอกสารดับเบิลยูเอสดีแอลตามมาตรฐานไม่เพียงพอต่อการทดสอบเว็บเซอร์วิสจึงต้องมีการเพิ่มข้อมูลข้างต้นเพื่อช่วยในการทดสอบเว็บเซอร์วิส โดยข้อมูลที่เพิ่มจะช่วยลดระยะเวลาในการทดสอบและช่วยให้สามารถทำการทดสอบแบบอัตโนมัติได้

อย่างไรก็ตามผลจากการวิจัยนี้ยังไม่ได้มีการนำไปทดสอบใช้งานให้เห็นจริง เนื่องจากยังไม่มีเครื่องมือที่สนับสนุนการทำงาน

2.10.2 การพัฒนาเครื่องมือสำหรับทดสอบการทำงานของเว็บเซอร์วิส (On the Development of Software Tools for Testing Web Service) [15] โดย Tsung-Teng Cheng และ Chih-Hsiung Fu

งานวิจัยนี้นำเสนอเครื่องมือสำหรับทดสอบการทำงานของเว็บเซอร์วิส ได้แก่ การทดสอบการใช้งานซีพียู (CPU) เวลาตอบสนอง (Response Time) และการทดสอบระดับหน่วย (Unit Testing) โดยใช้ข้อมูลที่เพิ่มเติมในเอกสารดับเบิลยูเอสดีแอลจากงานวิจัย [14]

จากการศึกษางานวิจัยนี้ พบข้อจำกัดดังนี้

1) การทดสอบระดับหน่วยจะต้องทำการแก้ไขรหัสคำสั่งของโปรแกรมเพื่อทดสอบฟังก์ชันการทำงานของเว็บเซอร์วิสที่ต้องการทดสอบ

2) การทดสอบต้องติดตั้งเซ็นเซอร์อ็อบเจกต์ (Sensor Object) ไว้ที่แม่ข่ายของผู้ให้บริการเว็บเซอร์วิสเพื่อใช้ตรวจจับการใช้งานซีพียู และเวลาตอบสนองในการทดสอบ

3) ไม่สามารถทดสอบการทำงานของเว็บเซอร์วิสประกอบได้เนื่องจากไม่ได้พิจารณาว่าเว็บเซอร์วิสที่นำมาทดสอบเป็นเว็บเซอร์วิสประกอบหรือไม่

2.10.3 โคโยตี : เอ็กซ์เอ็มแอลเฟรมเวิร์คสำหรับการทดสอบเว็บเซอร์วิส (Coyote: An XML-Based Framework for Web Service Testing) [16] โดย W. T. Tsai, Ray Paul, Weiwei Song และ Zhibin Cao

งานวิจัยนี้ได้นำเสนอโครงสร้างภาษาเอ็กซ์เอ็มแอลสำหรับการทดสอบเว็บเซอร์วิส โดยแบ่งการทำงานออกเป็น 2 ส่วนคือ

1) ตัวหลักทดสอบ (Test Master) ทำหน้าที่ในการแปลงเอกสารดับเบิลยูเอสดีแอลไปเป็นกรณีทดสอบในรูปแบบของภาษาเอ็กซ์เอ็มแอล

2) เครื่องทดสอบ (Test Engine) ทำหน้าที่ในการทดสอบเว็บเซอร์วิสโดยนำกรณีทดสอบที่ได้จากตัวหลักทดสอบมาทดสอบการทำงาน

อย่างไรก็ตามการแปลงเอกสารดับเบิลยูเอสดีแอลไปเป็นกรณีทดสอบยังไม่มีกฎเกณฑ์ที่ชัดเจน และไม่สามารถทดสอบการทำงานของเว็บเซอร์วิสประกอบได้

2.10.4 การรับรองส่วนประกอบซอฟต์แวร์ (Software Component Certification) [30] โดย Morris, J., Lee, G., Parker, K., Bundell, G.A. และ Lam, C.P.

งานวิจัยนี้กล่าวถึงการทดสอบส่วนประกอบซอฟต์แวร์ โดยมีตัวหลักทดสอบ (Test Master) ทำหน้าที่ในการแปลงเอกสารดับเบิลยูเอสดีแอลไปเป็นกรณีทดสอบในรูปแบบของภาษาเอ็กซ์เอ็มแอล งานวิจัยนี้แบ่งงานออกเป็น 2 ส่วน ได้แก่

1) นำเสนอข้อกำหนดทดสอบมาตรฐาน (Standard Test Specifications) สำหรับการสร้างข้อกำหนดทดสอบ (Test Specification) เพื่อนำไปใช้ทดสอบการทำงานของส่วนประกอบซอฟต์แวร์ โดยมีรูปแบบตามหลักไวยากรณ์ของภาษาเอ็กซ์เอ็มแอลดังรูปที่ 2.17 ซึ่งประกอบด้วยข้อมูล ข้อกำหนดทดสอบ (<TestSpecification>) ชุดของข้อมูลทดสอบ (<TestSet>) กลุ่มของข้อมูลทดสอบ (<TestGroup>) การดำเนินการ (<Operation Name>) และการเรียกใช้เมท็อด (<MethodCall>) เป็นต้น

2) นำเสนอเครื่องมือที่ใช้ทดสอบการทำงานของส่วนโปรแกรมโดยทำการทดสอบตามข้อกำหนดทดสอบที่ได้สร้างไว้

ในที่นี้ผู้วิจัยให้ความสนใจเฉพาะในส่วนของข้อกำหนดทดสอบมาตรฐาน โดยเห็นว่าสามารถนำมาประยุกต์ใช้ในการสร้างกรณีทดสอบสำหรับทดสอบการทำงานของเว็บเซอร์วิสประกอบได้ และจากการศึกษางานวิจัยชิ้นนี้พบข้อจำกัดคือ ผู้ทดสอบจะต้องสร้างข้อกำหนดทดสอบขึ้นเอง โดยไม่มีเครื่องมือที่ช่วยสนับสนุนในการสร้าง

```
<TestSpecification>
    <TestSet Name="...">+
        <TestGroup>
        <Invariant DataType="...">*
        <Operation Name="..." Pre="op_name">*
            <Invariant>*
            <Constructor>*
            <MethodCall Target="...">*
        <Operation>
        <Invariant>
...
```

รูปที่ 2.17 อีลีเมนต์ของข้อกำหนดทดสอบมาตรฐานจากงานวิจัย