วิธีการสร้างเครื่องประมวลผลตรรกะทางธุรกิจ

นางสาวปิยนุช โตสงวน

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555

AN APPROACH FOR CONSTRUCTING BUSINESS LOGIC ENGINE

Miss Piyanuch Tosanguan

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Software Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2012

หัวข้อวิทยานิพนธ์	วิธีการสร้างเครื่องประมวลผลตรรกะทางธุรกิจ
โดย	นางสาวปิยนุช โตสงวน
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.ภัทรชัย ลลิตโรจน์วงศ์)

ปิยนุช โตสงวน : วิธีการสร้างเครื่องประมวลผลตรรกะทางธุรกิจ. (AN APPROACH FOR CONSTRUCTING BUSINESS LOGIC ENGINE) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร. ธาราทิพย์ สุวรรณศาสตร์, 93 หน้า.

สาเหตุสำคัญของการแก้ไข ปรับเปลี่ยนโปรแกรมประยุกต์ เป็นผลมาจากการเปลี่ยนแปลงความต้องการ และการเปลี่ยนแปลงของกฎธุรกิจ ทั้งในระหว่างขั้นตอนการออกแบบ ขณะกำลังทำการพัฒนา และหลังจากที่โปรแกรมได้นำไปใช้งานจริงแล้ว ซึ่งการพัฒนาซอฟต์แวร์แบบดั้งเดิม กฎธุรกิจจะถูกรวมเข้าไปอยู่ในโค้ดของโปรแกรมประยุกต์ เมื่อต้องมีการเปลี่ยนแปลงแก้ไข ทำให้ใช้ระยะเวลานาน และมีค่าใช้จ่ายสูง แม้เป็นการเปลี่ยนแปลงเพียงเล็กน้อย ดังนั้นจึงเกิดแนวความคิดในการแยกกฎธุรกิจให้เป็นอิสระออกจากโค้ดของโปรแกรมประยุกต์ขึ้นมา

งานวิจัยนี้นำเสนอวิธีการกำหนดกฎให้อยู่ในรูปแบบของฟังก์ชัน ด้วยภาษาเอกซ์เอ็มแอล โดยมีไวยากรณ์ในการเขียนกฎที่คล้ายคลึงกับการเขียนโปรแกรม พร้อมกันนี้ยังได้พัฒนาเครื่องประมวลผลสำหรับกฎดังกล่าว ซึ่งมีลักษณะเป็นส่วนประกอบที่สามารถนำไปใช้ทำงานร่วมกับโปรแกรมประยุกต์อื่นได้ โดยไม่ผูกติดกับสถาปัตยกรรม จากผลการทดสอบแสดงให้เห็นว่าเครื่องประมวลผลทำงานได้อย่างถูกต้อง สามารถฝังตัวอยู่ในโปรแกรมประยุกต์ได้ทั้งแบบที่ทำงานอยู่บนเครือข่าย และแบบที่ไม่ได้ทำงานอยู่บนเครือข่าย (สแตนด์อโลน) ทำให้บรรลุวัตถุประสงค์ในการแยกตรรกะทางธุรกิจออกจากตรรกะของโปรแกรมประยุกต์

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่ออ. ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2555

5270778721 : MAJOR SOFTWARE ENGINEERING

KEYWORD : BUSINESS DATA PROCESSING / BUSINESS LOGIC / EXECUTION ENGINE / XML

PIYANUCH TOSANGUAN : AN APPROACH FOR CONSTRUCTING BUSINESS LOGIC ENGINE. ADVISOR : ASSOC.PROF. TARATIP SUWANNASART, Ph.D., 93 pp.

A major cause of modifications in software applications is attributed to changes in requirements and business rules during design, development, and maintenance. Traditional software development includes business rules directly into the application code. Maintenance of these applications leads to escalation in time and cost for even small changes. Therefore, the concept to isolate the business rules from the application is introduced.

This thesis introduces an approach for defining rules as functions using XML. The rules have syntax which is similar to programming syntax. Furthermore, it also describes the implementation of an execution engine for these rules. This engine is a pluggable component which can operate within another application. It is not tightly coupled with the architecture of an application. The testing results showed that the execution engine works correctly and can be embedded in any application both online and offline (standalone) to achieve the separation of business logic from application logic.

Department: Computer Engineering Student's Signature .....

Field of Study: Software Engineering Advisor's Signature .....

Academic Year: 2012

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ สำเร็จลุล่วงได้ด้วยความช่วยเหลืออย่างดียิ่งจาก รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้สละเวลาให้คำปรึกษา คำแนะนำ และแนวทางในการทำงานวิจัย ด้วยความเมตตาและเอาใจใส่มาโดยตลอด และขอกราบขอบพระคุณ รองศาสตราจารย์ ดร. วิวัฒน์ วัฒนาวุฒิ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร. อาทิตย์ ทองทักษ์ และผู้ช่วยศาสตราจารย์ ดร.ภัทรชัย ลลิตโรจน์วงศ์ คณะกรรมการสอบวิทยานิพนธ์ ที่กรุณาสละเวลาให้คำแนะนำ และข้อเสนอแนะในการพัฒนางานวิจัย

ขอขอบคุณบุคลากรในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ทุกท่าน ที่ให้ความช่วยเหลือ และอำนวยความสะดวกในการดำเนินการ ทั้งในเรื่องการศึกษา และการสอบวิทยานิพนธ์

สุดท้ายนี้ ขอกราบขอบพระคุณบิดา มารดา ที่คอยให้กำลังใจ ด้วยความห่วงใยตลอดระยะเวลาในการศึกษาและทำงานวิจัย จนกระทั่งวิทยานิพนธ์ฉบับนี้เสร็จสมบูรณ์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

หน้า

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ขั้นตอนในการพัฒนาซอฟต์แวร์ที่มีส่วนสำคัญเป็นอย่างมาก ต่อระยะเวลาในการพัฒนาก็คือขั้นตอนการรวบรวมและวิเคราะห์ความต้องการ เพื่อให้ได้มาซึ่งข้อกำหนดความต้องการ ที่ต้องนำไปใช้ในการออกแบบซอฟต์แวร์ต่อไป ซึ่งการได้มาของความต้องการในความเป็นจริงนั้น ส่วนใหญ่มักจะเกิดความล่าช้า และมีการเปลี่ยนแปลงอยู่บ่อยครั้งอย่างหลีกเลี่ยงไม่ได้ โดยถ้าหากการเปลี่ยนแปลงความต้องการนั้น เกิดขึ้นในช่วงที่ได้เริ่มทำการพัฒนาซอฟต์แวร์ไปแล้ว ผู้พัฒนาต้องทำการแก้ไขโปรแกรม ซึ่งอาจก่อให้เกิดข้อผิดพลาดในโปรแกรมมากยิ่งขึ้น แล้วถ้าหากการพัฒนาซอฟต์แวร์เข้าสู่ขั้นตอนการทดสอบด้วยแล้ว ยิ่งทำให้เกิดความสูญเสียทรัพยากรและมีค่าใช้จ่ายเพิ่มมากขึ้น เพราะต้องใช้ทั้งแรงงานในการพัฒนาและการทดสอบใหม่อีกครั้ง นอกจากนี้ยังทำให้การพัฒนาซอฟต์แวร์ให้เสร็จทันตามกำหนดระยะเวลาเดิมเป็นไปได้ยาก

การเปลี่ยนแปลงกฎธุรกิจ (Business rules) ก็เป็นอีกปัจจัยหนึ่ง ที่ทำให้ต้องมีการแก้ไขโปรแกรม ซึ่งการพัฒนาซอฟต์แวร์โดยทั่วไป หลังจากที่ได้นำซอฟต์แวร์ไปใช้งานจริงแล้ว การปรับเปลี่ยนหรือแก้ไขใดๆ จะเข้าสู่กระบวนการบำรุงรักษา (Maintenance) โดยเมื่อทำการแก้ไขโปรแกรมเสร็จเรียบร้อยแล้ว ต้องมีการคอมไพล์ (Compile) และติดตั้งซอฟต์แวร์ (Deployment) นั้นใหม่ จึงจะสามารถใช้งานจริงได้ ถึงแม้ว่าจะเป็นการแก้ไขเพียงเล็กน้อยก็ตาม

จากปัญหาข้างต้น จะสังเกตได้ว่าสาเหตุหลักที่ทำให้ต้องมีการปรับเปลี่ยน หรือแก้ไขโปรแกรมนั้น เกิดจากการเปลี่ยนแปลงของกฎธุรกิจเป็นสาเหตุหลัก ด้วยเหตุนี้จึงเกิดแนวคิดในการแยกกฎธุรกิจ ซึ่งมักมีการเปลี่ยนแปลงอยู่เป็นประจำ ออกจากตรรกะทางโปรแกรม และโค้ดโปรแกรมที่ยุ่งเหยิง โดยนำเอาหลักการของระบบจัดการกฎธุรกิจ (BRMS: Business Rule Management System) [1] ที่มีฟังก์ชันในการจัดเก็บ จัดการ และเชื่อมต่อให้โปรแกรมอื่นสามารถเรียกใช้งาน เพื่อประมวลผลกฎที่มีอยู่ผ่านเครื่องประมวลผลกฎ (Business Rules Engine) มาประยุกต์ใช้ เพื่อทำให้กฎธุรกิจมีความเป็นอิสระ ไม่ติดอยู่กับโปรแกรม ดังนั้นเมื่อมีความต้องการที่จะแก้ไข เปลี่ยนแปลง เพิ่มเติม กฎธุรกิจ หรือกฎอื่นๆ ก็สามารถทำได้อย่างอิสระ โดยที่ไม่กระทบกับโปรแกรมที่มีการติดต่อใช้งานกับกฎเหล่านี้

เครื่องประมวลผลกฎที่เป็นที่รู้จัก อย่าง เจบอสรูส์ (JBoss Rules) [2] และไอล็อกเจรูส์ (ILOG JRules) [3] เป็นระบบวิสาหกิจ (Enterprise system) ซึ่งต้องทำงานอยู่บนแอพพลิเคชันเซิร์ฟเวอร์ (Application server) และใช้เครือข่ายในการเชื่อมต่อ โดยกฎมักจะถูกใช้งานผ่านบริการ (Service) จึงทำให้ไม่สามารถนำไปใช้งานกับโปรแกรมแบบสแตนด์อโลน (Standalone) ได้ ส่วนเครื่องประมวลผลกฎที่ไม่ต้องทำงานอยู่บนเครือข่าย เช่น เอ็นเอกซ์บีอาร์อี (NxBRE) [4] และซิมเพิลรูลเอนจิน (Simple Rule Engine) [5] ก็มีไวยากรณ์ในการเขียนกฎที่ไม่คุ้นเคยเหมือนกับการเขียนโปรแกรม

งานวิจัยนี้ เป็นการนำเสนอรูปแบบการเขียนกฎที่ใกล้เคียงกับการเขียนโปรแกรม พร้อมกับนำเสนอเครื่องประมวลผล ที่สามารถฝังตัวอยู่ในโปรแกรมที่ต้องการประมวลผลกฎเหล่านี้ ซึ่งผู้วิจัยเลือกใช้ ภาษาเอกซ์เอ็มแอล (XML: Extensible Markup Language) เป็นภาษาที่ใช้ในการเขียนกฎ เนื่องจากเห็นว่าเป็นภาษาที่คุ้นเคย สามารถเรียนรู้ได้เร็ว ทั้งยังสามารถอ่านและทำความเข้าใจได้ เหมาะกับการนำมาใช้ในการอธิบายกฎ โดยเครื่องประมวลผลนี้มีความสามารถในการติดต่อกับฐานข้อมูล เพื่อดึงข้อมูลมาใช้งานภายในกฎ สามารถใช้คำสั่งของภาษาโปรแกรม และมีการนำความสามารถของภาษาเอกซ์เอ็มแอล คือ เอกซ์พาธ (XPath) และเอกซ์เอสแอลที (XSLT) มาใช้งานร่วมด้วย นอกจากนี้ได้นำเอาข้อดีของเครื่องประมวลผลกฎ เช่น เอซีทีเอลบิซซิเนสรูลเอนจิน (ACTL Business Rule Engine) [6] และเฟล็กซ์รูล (FlexRule) [7] ที่สามารถใช้งานกฎในหลากหลายรูปแบบ คือ สามารถเรียกใช้งานกฎที่เก็บอยู่ในฐานข้อมูล ไฟล์ดีแอลแอล (DLL) และกฎที่อยู่ในรูปแบบของไฟล์เอกสาร มาเป็นความสามารถให้กับเครื่องประมวลผลนี้ด้วย ซึ่งเครื่องประมวลผลที่พัฒนาขึ้นนี้ สามารถนำไปใช้งานได้กับโปรแกรมอื่น ได้ทั้งแบบที่ทำงานอยู่บนเครือข่าย และแบบสแตนด์อโลน

1.2 วัตถุประสงค์ของการวิจัย

1) เสนอรูปแบบการเขียนกฎด้วยภาษาเอกซ์เอ็มแอล ที่มีไวยากรณ์ในการเขียนกฎใกล้เคียงกับการเขียนโปรแกรม

2) พัฒนาเครื่องประมวลผล สำหรับประมวลผลกฎที่เขียนอยู่ในรูปแบบตามวิธีการที่นำเสนอ เพื่อเป็นทางเลือกในการนำไปใช้ในการพัฒนาซอฟต์แวร์

1.3 ขอบเขตของการวิจัย

1) กฎต้องเขียนอยู่ในรูปแบบที่ถูกต้องตามโครงสร้าง และใช้ตัวดำเนินการที่กำหนดไว้ ตามข้อกำหนดในการเขียนกฎ

2) ไวยากรณ์ในการเขียนกฎที่ใกล้เคียงกับการเขียนโปรแกรม หมายถึง การทำงานแบบเรียงตามลำดับ (Sequential) การตรวจสอบเงื่อนไข (Condition) และการทำงานวนซ้ำ (Loop)

3) กฎสามารถเก็บอยู่ในฐานข้อมูล หรือเป็นไฟล์เอกซ์เอ็มแอล ที่อยู่ภายใต้โฟลเดอร์โดยมีชื่อเดียวกันกับกฎ

4) เอกซ์เอสแอลที ที่ใช้ในการแปลงค่าข้อมูลส่งกลับ ต้องถูกต้องตามไวยากรณ์ของภาษา

5) เอกซ์เอสแอลที จะถูกใช้งานก็ต่อเมื่อเก็บอยู่ในรายการ (Record) เดียวกันในฐานข้อมูล หรืออยู่ภายใต้โฟลเดอร์ของกฎนั้นๆ

6) กฎสามารถติดต่อกับฐานข้อมูล ซีควลเซิร์ฟเวอร์ (SQL Server) ซีควลเซิร์ฟเวอร์คอมแพค หรือเอสคิวแอลซีอี (SQL CE) และออราเคิล (Oracle) ได้เป็นอย่างน้อย

7) ไฟล์ดีแอลแอลที่กฎสามารถเรียกใช้งานได้ ต้องอิมพลีเมนต์อินเทอร์เฟซตามที่เครื่องประมวลผลกฎกำหนด

8) เครื่องประมวลผลอยู่ในรูปแบบไฟล์ดีแอลแอล ซึ่งพัฒนาด้วยภาษาวิชวลเบสิกดอตเน็ต (Visual Basic .NET)

9) การเปลี่ยนแปลงตรรกะภายในกฎ ที่ไม่กระทบต่อโปรแกรมที่เรียกใช้งาน คือ กฎที่มีพารามิเตอร์เหมือนเดิม (อินเทอร์เฟซไม่เปลี่ยน)

10) การทดสอบ สร้างโปรแกรมมาเรียกใช้งานกฎผ่านเครื่องประมวลผล โดยต้องทดสอบให้ครบคลุมทุกการดำเนินการตามไวยากรณ์ในการเขียนกฎ แล้วเปรียบเทียบผลลัพธ์ของการทำงาน ว่าถูกต้องเหมือนกับการเขียนอยู่ในโปรแกรม

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) เครื่องประมวลผล ที่ทำให้สามารถแยกตรรกะทางธุรกิจออกจากโปรแกรม เพื่อใช้เป็นเครื่องมือช่วยในการจัดการกับปัญหาเรื่องความต้องการที่มีการเปลี่ยนแปลงบ่อยได้

2) สามารถนำรูปแบบไวยากรณ์ในการเขียนกฎ และเครื่องประมวลผลนี้ ไปใช้เป็นทางเลือกในการพัฒนาซอฟต์แวร์ได้

3) เพิ่มความน่าเชื่อถือให้กับโปรแกรมภายในองค์กร ที่มีความต้องการในการตรวจสอบเงื่อนไขบางอย่างที่เหมือนกัน เนื่องจากสามารถใช้งานกฎเดียวกันร่วมกันได้ จึงแน่ใจได้ว่าผ่านการตรวจสอบจากเงื่อนไขเดียวกันจริงๆ

1.5 ขั้นตอนการวิจัย

1) ศึกษาเครื่องประมวลผลกฎและงานวิจัยที่เกี่ยวข้อง
2) ศึกษามาตราฐานเอกซ์เอ็มแอล เอกซ์เอสดี (XSD) เอกซ์พาธ และเอกซ์เอสแอลที
3) กำหนดขอบเขตความสามารถของเครื่องมือ
4) ออกแบบโครงสร้างที่ใช้เป็นไวยากรณ์ในการเขียนกฎ
5) พัฒนาเครื่องมือตามที่ได้ออกแบบไว้
6) สร้างตัวอย่างกฎสำหรับการทดสอบ
7) ทดสอบเครื่องมือที่พัฒนา
8) สรุปผลการวิจัยและข้อเสนอแนะ
9) จัดทำรายงานวิทยานิพนธ์

1.6 บทความวิชาการที่ได้รับการตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ ได้รับการตีพิมพ์เป็นบทความวิชาการในหัวข้อเรื่อง “An Approach for Defining Rules as Functions in Rule-Based Software Development” ซึ่งได้รับการคัดเลือกเพื่อนำเสนอ และตีพิมพ์ในงานประชุมวิชาการ “Seventh International Conference on Digital Information Management (ICDIM 2012)” ระหว่างวันที่ 22-24 สิงหาคม พ.ศ. 2555 ณ เขตปกครองพิเศษมาเก๊า สาธารณรัฐประชาชนจีน

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 กฎธุรกิจ

กฎธุรกิจ คำนี้ถูกนำไปใช้กันอย่างกว้างขวางจนทำให้ความหมายถูกบิดเบือนไป จึงทำให้มีความเข้าใจที่แตกต่างกันไปในแต่ละกลุ่มคน เช่น ในด้านการลงทุน อาจกล่าวว่า ยอดรวมของผลกำไรไม่ควรน้อยกว่า 10 เปอร์เซ็นต์ต่อปี คือกฎ ซึ่งในความเป็นจริงแล้วไม่ใช่ หากแต่คือเป้าหมาย [8] คำนิยามของกฎธุรกิจโดย [9] หมายถึงข้อความที่เป็นการกำหนดหรือข้อจำกัดในมุมมองทางธุรกิจ ซึ่งมีจุดมุ่งหมายเพื่อควบคุมหรือกำหนดพฤติกรรมของธุรกิจ โดยกฎธุรกิจต้องเป็นส่วนย่อยที่เล็กที่สุด ที่มีผลลัพธ์เป็นจริงหรือเท็จ ประเภทของกฎธุรกิจ สามารถแยกได้เป็น 4 ประเภท คือ คำนิยามของคำศัพท์ทางธุรกิจ (Definitions of business terms) ข้อเท็จจริง (Facts) ข้อจำกัด (Constraints) และการได้มา (Derivations)

ในบริบทของการจัดความต้องการ กฎธุรกิจมักจะเกี่ยวข้องกับกระบวนการทางธุรกิจ [10] ดังต่อไปนี้

- ข้อจำกัด (Limitations) เช่น ผู้โดยสารได้รับอนุญาตให้มีกระเป๋าเดินทางได้เพียง 2 ใบเท่านั้น
- กฎการตรวจสอบ (Validation Rules) เช่น การถ่ายโอนบัญชีไม่สามารถทำได้ถ้าหากบัญญชีนั้นสร้างหลังวันที่ 1/1/1980
- สิทธิ์ (Permissions) เช่น รายละเอียดทางบัญชีสามารถเข้าถึงได้เฉพาะสมาชิกประเภททองเท่านั้น (Gold members)
- การประเมินผล (Evaluation) เช่น ถ้ามียอดรวมการสั่งซื้อสินค้ามากกว่า 4,000 ยูโร ลูกค้าจะได้รับส่วนลด 4 เปอร์เซ็นต์
- กฎการดำเนินการ (Process rule) เช่น ถ้ามีการล็อคประตู สัญญาณแสดงการใช้งานต้องถูกสั่งให้ทำงาน

2.1.2 เอกซ์เอ็มแอล

เอกซ์เอ็มแอล [11, 12] เป็นภาษามาร์กอัพเช่นเดียวกับเอชทีเอ็มแอล (HTML: Hypertext Markup Language) ถูกพัฒนามาจากเอสจีเอ็มแอล (SGML: Standard Generalized Markup Language) โดยดัดแปลงให้มีความซับซ้อนน้อยลง ภายใต้ความดูแลของดับเบิลยูสามซี (W3C: World Wide Web Consortium) ซึ่งมีจุดประสงค์เพื่อใช้ในการสื่อสาร แลกเปลี่ยนและเก็บข้อมูล โดยให้ความสนใจว่าข้อมูลนั้นคืออะไร แตกต่างจากเอชทีเอ็มแอลที่ถูกออกแบบมาเพื่อใช้ในการแสดงผลข้อมูล และสนใจว่าจะแสดงข้อมูลนั้นอย่างไร

เอกซ์เอ็มแอลสามารถอธิบายตัวเองได้ และยังเป็นภาษาที่ไม่มีรูปแบบโครงสร้างกำหนดไว้ล่วงหน้า ทำให้สามารถกำหนดชื่อแท็กหรืออิลิเมนต์ (Element) และชื่อแอตทริบิวต์ (Attribute) ให้เหมาะสมตามความต้องการในการใช้งานได้ เอกสารเอกซ์เอ็มแอลมีโครงสร้างทางตรรกะ (Logical structure) และโครงสร้างทางกายภาพ (Physical structure) โดยโครงสร้างทางกายภาพประกอบด้วยหน่วยหลายๆ หน่วย ที่เรียกว่าเอนทรีส์ (Entries) ซึ่งแต่ละเอนทรี (Entry) อาจจะอ้างถึงเอนทรีอื่นๆ ที่อยู่ในเอกสารได้อีก ส่วนโครงสร้างทางตรรกะประกอบด้วยการประกาศ (Declaration) อิลิเมนต์ (Element) คำอธิบาย (Comment) อักขระอ้างอิง (Character references) และคำสั่งประมวลผล (Processing instruction)

กฎไวยากรณ์ของเอกซ์เอ็มแอล (XML Syntax Rules) มีดังนี้

- แต่ละเอกซ์เอ็มแอลอิลิเมนต์ต้องซ้อนกันอย่างมีลำดับ
- เอกสารเอกซ์เอ็มแอลต้องมีอิลิเมนต์ราก (Root element) คือมีเพียงอิลิเมนต์เดียวที่ครอบคลุมอิลิเมนต์อื่นๆ ทั้งหมด
- ค่าของเอกซ์เอ็มแอลแอตทริบิวต์ต้องอยู่ในอัญประกาศ หรือเครื่องหมายคำพูด
- การเขียนคำอธิบายในเอกซ์เอ็มแอล เป็นดังนี้ <!--This is a comment -->
- การเว้นวรรคจะถูกคงไว้ในเอกซ์เอ็มแอล ซึ่งต่างจากเอชทีเอ็มแอลที่จะตัดทอนให้เหลือเพียงเว้นวรรคเดียว

- อักขระพิเศษในเอกซ์เอ็มแอลที่กำหนดไว้ล่วงหน้า ถ้าหากต้องการใช้ในเอกซ์เอ็มแอลอิลิเมนต์ มี 5 ตัว ดังนี้

<	แทน	< (less than)
>	แทน	> (greater than)
&	แทน	& (ampersand)
'	แทน	' (apostrophe)
"	แทน	" (quotation mark)

เอกสารเอกซ์เอ็มแอลที่มีความถูกต้องตามกฎไวยากรณ์ เรียกว่าเอกสารเอกซ์เอ็มแอลที่มีรูปแบบถูกต้อง (Well-formed) ส่วนเอกสารเอกซ์เอ็มแอลที่มีความถูกต้อง (Valid) คือ เอกสารเอกซ์เอ็มแอลที่เป็นไปตามนิยามโครงสร้างการประกาศข้อมูล และข้อจำกัดที่กำหนดไว้ ภาพที่ 2.1 เป็นตัวอย่างเอกสารเอกซ์เอ็มแอล

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

ภาพที่ 2.1 ตัวอย่างเอกสารเอกซ์เอ็มแอล

2.1.3 เอกซ์เอสดี

เอกซ์เอสดี (XSD: XML Schema Definition) [13, 14] ถูกเผยแพร่โดยดับเบิลยู สามซี เป็นหนึ่งในหลายๆ ภาษาของเอกซ์เอ็มแอลสคีมา ใช้ในการอธิบายโครงสร้างของเอกสารเอกซ์เอ็มแอล ดังนี้

- กำหนดอิลิเมนต์ที่ปรากฏอยู่ในเอกสารได้
- กำหนดแอตทริบิวต์ที่ปรากฏอยู่ในเอกสารได้
- กำหนดว่าอิลิเมนต์ใดที่มีอิลิเมนต์ลูก
- กำหนดลำดับของอิลิเมนต์ลูก
- กำหนดจำนวนของอิลิเมนต์ลูก
- กำหนดว่าอิลิเมนต์นั้นเป็นอิลิเมนต์ว่างหรือสามารถใส่ข้อความได้
- กำหนดชนิดข้อมูลของอิลิเมนต์และแอตทริบิวต์
- กำหนดค่าให้อัตโนมัติ และค่าบังคับ ให้กับอิลิเมนต์และแอตทริบิวต์

นอกจากนี้เอกซ์เอสดี ยังสามารถกำหนดข้อบังคับ (restriction) และรูปแบบ (pattern) ของข้อมูลได้อีกด้วย ข้อบังคับประเภทข้อมูลแสดงในตารางที่ 2.1

ตารางที่ 2.1 ข้อบังคับประเภทข้อมูล [14]

ข้อบังคับ	**คำอธิบาย**
enumeration	การกำหนดรายการข้อมูลที่เป็นไปได้
fractionDigits	การกำหนดค่าสูงสุดของจำนวนทศนิยม ต้องเท่ากับหรือมากกว่า 0
length	การกำหนดจำนวนที่แน่นอนของอักขระหรือรายการข้อมูล ต้องเท่ากับหรือมากกว่า 0
maxExclusive	การกำหนดขอบบนของค่าตัวเลข (ค่าข้อมูลต้องน้อยกว่าค่าที่กำหนด)
maxInclusive	การกำหนดขอบบนของค่าตัวเลข (ค่าข้อมูลต้องน้อยกว่าหรือเท่ากับค่าที่กำหนด)
maxLength	การกำหนดจำนวนสูงสุดของอักขระหรือรายการข้อมูล ต้องเท่ากับหรือมากกว่า 0
minExclusive	การกำหนดขอบล่างของค่าตัวเลข (ค่าข้อมูลต้องมากกว่าค่าที่กำหนด)
minInclusive	การกำหนดขอบล่างของค่าตัวเลข (ค่าข้อมูลต้องมากกว่าหรือเท่ากับค่าที่กำหนด)
minLength	การกำหนดจำนวนต่ำสุดของอักขระหรือรายการข้อมูล ต้องเท่ากับหรือมากกว่า 0
pattern	การกำหนดรูปแบบการจัดเรียงของอักขระ
totalDigits	การกำหนดจำนวนที่แน่นอนของจุดทศนิยม ต้องมากกว่า 0
whiteSpace	การกำหนดวิธีการจัดการกับ white space (line feeds, tabs spaces, carriage return) “preserve” คือ คงค่า white space ไว้, “replace” คือ แทนค่าด้วย space, “collapse” คือ ตัด white space ออก และแทนค่าด้วย 1 space

ภาพที่ 2.2 แสดงตัวอย่างเอกซ์เอสดี ที่ใช้ในการตรวจสอบความถูกต้องของเอกสารเอกซ์เอ็มแอลในภาพที่ 2.1 ซึ่งเอกซ์เอสดีมีอิลิเมนต์ 2 ประเภท คือ ชนิดข้อมูลเชิงซ้อน (Complex type) และชนิดข้อมูลอย่างง่าย (Simple type) โดยชนิดข้อมูลเชิงซ้อน เป็นอิลิเมนต์ที่

สามารถมีอิลิเมนต์อื่นๆ และ/หรือแอตทริบิวต์ ได้หลายๆ ค่า ส่วนชนิดข้อมูลอย่างง่ายเป็นอิลิเมนต์ที่ประกอบด้วยข้อความอย่างเดียวเท่านั้น

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<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>
```

ภาพที่ 2.2 ตัวอย่างเอกซ์เอสดี

จากภาพที่ 2.2 อิลิเมนต์ note เป็นชนิดข้อมูลเชิงซ้อน และอิลิเมนต์ to, from, heading และ body เป็นชนิดข้อมูลอย่างง่าย โดยอิลิเมนต์ note ประกอบด้วยอิลิเมนต์ย่อยๆ ทั้ง 4 เรียงตามลำดับ

2.1.4 เอกซ์พาธ

เอกซ์พาธ (XPath: XML Path Language) [15, 16] เป็นภาษาที่ใช้ในการดึงข้อมูลจากเอกสารเอกซ์เอ็มแอล โดยเข้าถึงอิลิเมนต์ และแอตทริบิวต์ ด้วยการกำหนดนิพจน์ของพาธ ซึ่งคล้ายกับพาธของไฟล์ที่อยู่ในเครื่องคอมพิวเตอร์ ตัวอย่างเอกซ์พาธแสดงในตารางที่ 2.2

ตารางที่ 2.2 ตัวอย่างเอกซ์พาธ [16]

เอกซ์พาธ	**ผลลัพธ์**
/bookstore/book[1]	เลือก book อิลิเมนต์แรก ของ bookstore อิลิเมนต์ หมายเหตุ: ตั้งแต่ IE5 ขึ้นไป ใช้ [0] เป็นอิลิเมนต์แรก แต่ที่สอดคล้องตามมาตรฐานของดับเบิลยูสามซี คือ [1]
/bookstore/book[last()]	เลือก book อิลิเมนต์สุดท้าย ของ bookstore อิลิเมนต์
/bookstore/book[position()<3]	เลือก book 2 อิลิเมนต์แรก ของ bookstore อิลิเมนต์
//title[@lang]	เลือกทุก title อิลิเมนต์ ที่มีแอตทริบิวต์ lang

ตารางที่ 2.2 ตัวอย่างเอกซ์พาธ (ต่อ)

เอกซ์พาธ	**ผลลัพธ์**
//title[@lang='eng']	เลือกทุก title อิลิเมนต์ ที่มีค่าของแอตทริบิวต์ lang เท่ากับ eng
/bookstore/book[price>35.00]	เลือกทุก book อิลิเมนต์ ของ bookstore อิลิเมนต์ ที่มีค่าของ price อิลิเมนต์มากกว่า 35.00
/bookstore/book[price>35.00]/title	เลือกทุก title อิลิเมนต์ ของ book อิลิเมนต์ ของ bookstore อิลิเมนต์ ที่มีค่าของ price อิลิเมนต์ มากกว่า 35.00

2.1.5 เอกซ์เอสแอลที

เอกซ์เอสแอลที (XSLT: Extensible Stylesheet Language Transformations) [17, 18] เป็นภาษาที่ใช้ในการแปลงเอกสารเอกซ์เอ็มแอล ไปเป็นเอกสารเอกซ์เอ็มแอลในรูปแบบอื่น หรือแปลงเป็นเอกสารชนิดอื่นที่เบราว์เซอร์รู้จัก เช่น เอชทีเอ็มแอล

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

ภาพที่ 2.3 ตัวอย่างเอกซ์เอสแอลที

จากภาพที่ 2.3 เป็นการแปลงเอกสารเอกซ์เอ็มแอลไปเป็นเอกซ์เอชทีเอ็มแอล (XHTML) ซึ่งอิลิเมนต์รากคือ <xsl:stylesheet> หรือ <xsl:transform> โดยเอกซ์เอสแอลทีอิลิเมนต์ที่ใช้บ่อย แสดงในตารางที่ 2.3

ตารางที่ 2.3 เอกซ์เอสแอลทีอิลิเมนต์ที่ใช้บ่อย

อิลิเมนต์	**คำอธิบาย**
template	ใช้กำหนดรูปแบบของข้อมูลเอกสารเอาท์พุต จากข้อมูลในเอกสารเอกซ์เอ็มแอลที่กำหนดในแอตทริบิวต์ match ใช้เอกซ์พาธในการกำหนดค่า
value-of	ดึงค่าข้อมูลจากเอกซ์เอ็มแอลอิลิเมนต์
for-each	สำหรับแต่ละอิลิเมนต์ ที่กำหนดในแอตทริบิวต์ select (ใช้เอกซ์พาธ)
sort	เรียงลำดับข้อมูลเอาท์พุต โดยใส่ไว้ใน for-each
if	ตรวจสอบเงื่อนไข
choose	ใช้ร่วมกับ when และ otherwise ในการตรวจสอบหลายเงื่อนไข
apply-templates	กำหนดให้ใช้รูปแบบกับอิลิเมนต์ปัจจุบันหรืออิลิเมนต์ลูก

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 งานวิจัยเรื่อง “Rule Engine Research and Implementation in Financial System” [19]

จากการพัฒนาที่รวดเร็วของเศรษฐกิจ ส่งผลให้ต้องทำการเปลี่ยนแปลงตรรกะทางธุรกิจอยู่บ่อยครั้ง แต่เนื่องจากตรรกะทางธุรกิจรวมอยู่ในโปรแกรม ทำให้เกิดค่าใช้จ่ายสูง งานวิจัยนี้ จึงได้นำเสนอเครื่องประมวลผลกฎที่ใช้กับระบบทางการเงิน เพื่อช่วยลดค่าใช้จ่าย และทำให้ระบบมีความยืดหยุ่นมากขึ้น ชื่อว่าเอฟอาร์เอ็ม (FRM: Financial Rules Manager) โดยกำหนดภาษาที่ใช้ในการเขียนกฎขึ้นมาเอง (*.rb files) หลังจากนั้นทำการแปลงไปเป็นไฟล์จาวาคลาสไบนารี (Java Class binary files) แล้วเก็บลงฐานข้อมูล การใช้งานกฎเหล่านี้สามารถทำได้ผ่านทางเอพีไอ (API)

ไวยากรณ์สำหรับเขียนกฎประกอบด้วย 2 ส่วนคือ เพรดิเคต (predicate) และแอ็คชัน (action) โดยใช้ IF และ THEN-ELSE ในการอธิบายกฎ คำสำคัญที่ใช้ในการเขียนกฎมีดังนี้

- OBJECT ใช้สำหรับการประกาศตัวแปร อยู่ตอนต้นของกฎ และอิมพลีเมนต์อินเทอร์เฟซ IRules เพื่อให้กฎใช้งานทุกเมท็อดพื้นฐานได้

- การนิยามกฎต้องขึ้นต้นด้วย RULE แทนกฎพื้นฐาน (basic rule) หรือ RULECP แทนกฎผสม (compound rule) กฎผสมสามารถเรียกใช้กฎพื้นฐานได้

- ชื่อกฎที่ขึ้นต้นด้วย 'FR' เป็นกฎที่สามารถตั้งค่าได้ว่าต้องการประมวลผลหรือไม่

- VALUEPROPERTY เป็นการกำหนดค่าข้อมูลส่งกลับของกฎ ถ้านิพจน์ใน IF เป็นจริง จะมีค่าเป็น true แต่ถ้าไม่จะมีค่าเป็น false

จากตัวอย่างกฎในงานวิจัยนี้ จะเห็นว่ากฎที่เขียนนั้นยังไม่เป็นอิสระ แยกจากโปรแกรมอย่างแท้จริง สังเกตได้จากการประกาศตัวแปรที่ต้องสอดคล้องกับอ็อบเจกต์ในโปรแกรมที่เรียกใช้ และถ้าหากต้องการแก้ไขกฎ ต้องทำการคอมไพล์กฎไปเป็นไฟล์จาวาคลาสไบนารี เพื่อเก็บลงสู่ฐานข้อมูลใหม่ ซึ่งในงานวิจัยนี้ไม่ได้กล่าวถึงความสามารถในการจัดการและแยกแยะกฎที่ทำการแก้ไขว่าจะเก็บเข้าสู่ฐานข้อมูลอย่างไร เป็นรายการใหม่หรืออัพเดทที่รายการเดิม

2.2.2 งานวิจัยเรื่อง "Research on Rule Definition and Engine for General Text Processing" [20]

จากความต้องการในการประมวลผลข้อความ ที่เกิดขึ้นบ่อยครั้ง และมีความหลากหลาย ถ้าหากต้องพัฒนาโปรแกรมเพื่อตอบสนองต่อความต้องการที่แตกต่างกัน โปรแกรมก็จะมีความซับซ้อน และยากต่อการนำกลับมาใช้ใหม่ งานวิจัยนี้จึงกำหนดกฎในการอธิบายตรรกะของการประมวลผลข้อความ และออกแบบเครื่องมือเพื่อใช้ในการประมวลผลกฎที่ได้นำเสนอ ซึ่งช่วยลดความซับซ้อนของการประมวลผลข้อความออกจากโปรแกรมได้

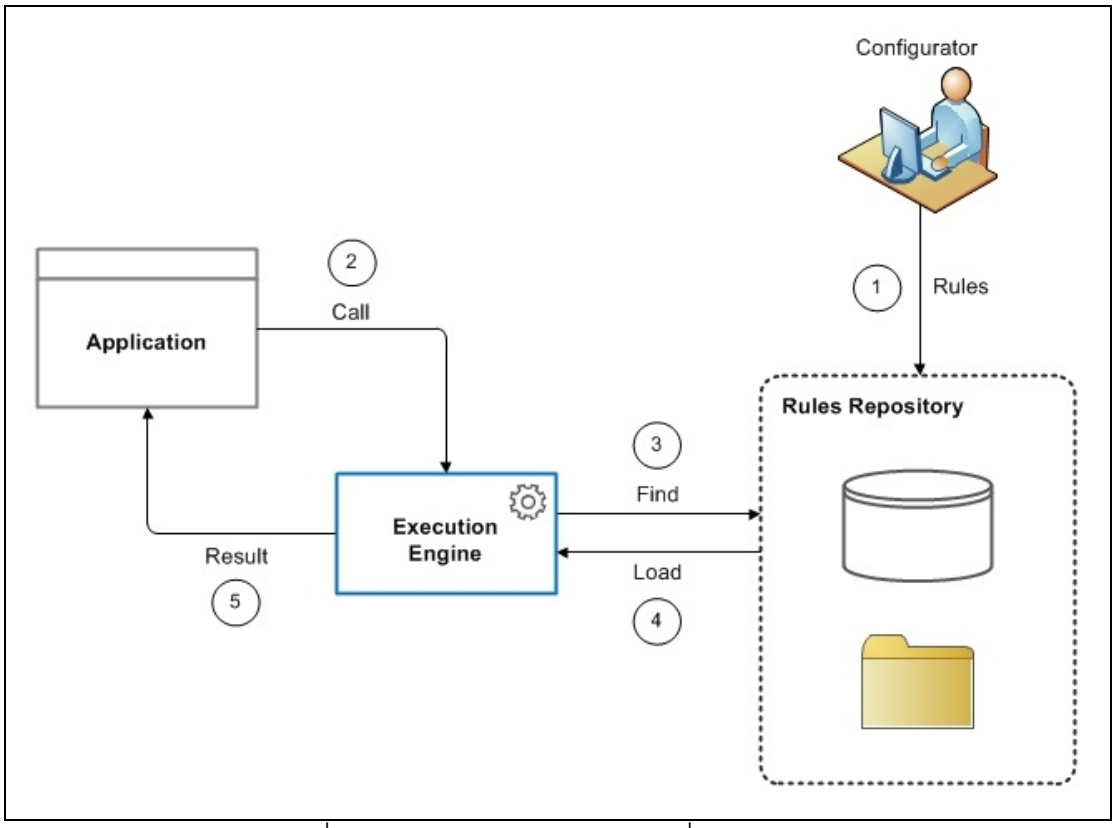
งานวิจัยนี้เป็นการนำเอาแนวคิดของเครื่องประมวลผลกฎ มาประยุกต์ใช้ในการพัฒนาเครื่องมือเพื่อประมวลผลข้อความ ซึ่งเครื่องประมวลผลข้อความนี้ ใช้นิพจน์ปรกติ (Regular Expression) ในการค้นหาข้อความจากข้อมูลนำเข้า เพื่อนำมาปรับเปลี่ยนให้เป็นไปตามกฎที่เขียนขึ้น ข้อดีของงานวิจัยชิ้นนี้ คือการแยกความต้องการในการประมวลผลข้อความออกมาจากโปรแกรม หากแต่ยังไม่ได้กล่าวถึงวิธีการนำเอาเครื่องมือไปใช้ในการทำงานร่วมกับโปรแกรมอื่น

บทที่ 3
การออกแบบและวิธีการดำเนินงาน

3.1 การทำงานของเครื่องประมวลผล

เครื่องประมวลผล ถูกออกแบบให้ทำงานอยู่ภายใต้โปรแกรมอื่นที่มีความต้องการในการประมวลผลกฎ โดยสามารถนำไปใช้งานได้ทั้งกับโปรแกรมที่ทำงานอยู่บนเครือข่าย และแบบสแตนด์อโลน กล่าวคือสามารถนำไปใช้งานกับ เว็บแอพพลิเคชัน (Web Application) เว็บเซอร์วิส (Web Service) และวินโดวส์แอพพลิเคชัน (Windows Application) ได้

เครื่องประมวลผลจะทำงานเมื่อมีการเรียกใช้ ด้วยการระบุชื่อกฎที่ต้องการประมวลผล และข้อมูลนำเข้าที่ต้องใช้ในกฎนั้น โดยกฎต้องเขียนอยู่ในรูปแบบที่กำหนด ซึ่งผู้วิจัยได้แนวคิดมาจากหลายเครื่องประมวลผลกฎ เช่น รูส์พาเล็ต (RulesPalatte) [21] เอ็นเอกซ์บีอาร์อี (NxBRE) เฟล็กซ์รูล (FlexRule) และซิมเพิลรูลเอนจิน (Simple Rule Engine)



ภาพที่ 3.1 ภาพรวมการทำงานของเครื่องประมวลผล

การทำงานของเครื่องประมวลผลจากในภาพที่ 3.1 ประกอบด้วยลำดับการทำงาน 5 ขั้นตอน ดังนี้

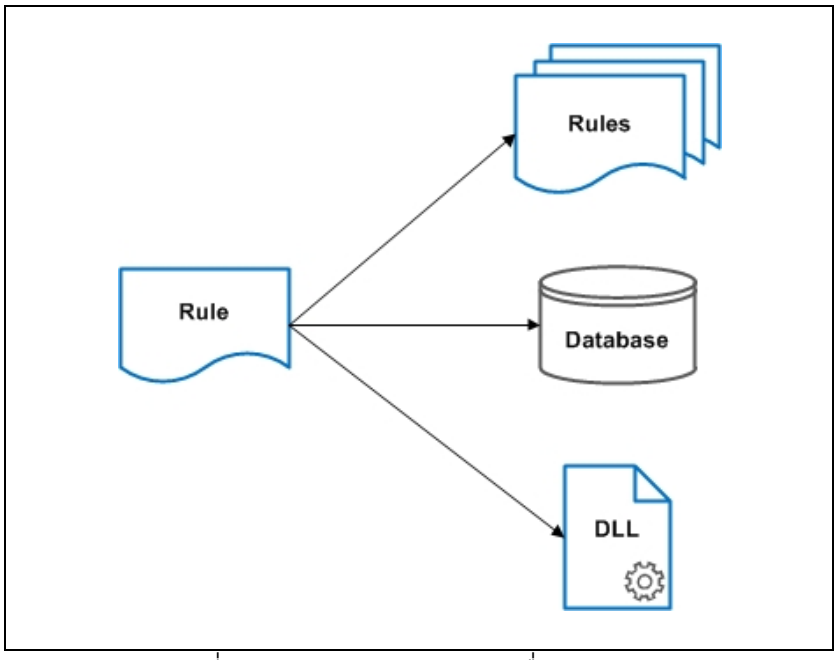
1) นำกฎที่เขียนถูกต้องตามข้อกำหนดและผ่านการทดสอบแล้ว เข้าสู่คลังเก็บ (Repository)

2) โปรแกรมที่ต้องการประมวลผลกฎจากข้อที่ 1 ทำการเรียกใช้ผ่านเครื่องประมวลผล โดยระบุชื่อกฎและข้อมูลนำเข้า

3) เครื่องประมวลผลทำการค้นหากฎตามที่โปรแกรมในข้อ 2 ระบุไว้ จากคลังเก็บ ซึ่งอาจเป็นฐานข้อมูล หรือไฟล์เอกซ์เอ็มแอลที่อยู่ภายใต้โฟลเดอร์ชื่อเดียวกันกับกฎ

4) หลังจากค้นหากฎที่ต้องการจากคลังเก็บพบแล้ว เครื่องประมวลผลทำการโหลดกฎเข้าหน่วยความจำ และดำเนินการประมวลผลชุดคำสั่งภายในกฎ

5) เครื่องประมวลผล ส่งผลลัพธ์ให้กับโปรแกรมที่เรียกใช้งาน



ภาพที่ 3.2 ความสามารถในการเชื่อมต่อของกฎ

กฎมีความสามารถในการเชื่อมต่อกับภายนอกได้ ดังภาพที่ 3.2 คือ

- สามารถสั่งประมวลผลกฎอื่นๆ ที่อยู่ในคลังเก็บเดียวกัน แล้วนำผลลัพธ์จากกฎที่เรียกใช้นั้นกลับมาใช้งานต่อได้
- สามารถติดต่อฐานข้อมูล เพื่อดึงข้อมูลที่ต้องการมาทำงานภายในกฎได้

- สามารถสั่งประมวลผลไฟล์ดีแอลแอล ที่อิมพลีเมนต์อินเทอร์เฟซตามที่กำหนดและอยู่ภายในโปรแกรมเดียวกันกับที่ใช้งานเครื่องประมวลผลได้

3.2 ข้อกำหนดในการเขียนกฎ

รูปแบบโครงสร้างของภาษาเอกซ์เอ็มแอลที่ใช้ในการเขียนกฎเป็นดังภาพที่ 3.3 และมีประเภทของข้อมูลที่สามารถใช้ได้ กำหนดไว้ในตารางที่ 3.1

```
<Function NAME="..." RETURN="..." DATATYPE="...">
  <Parameters>
    <Parameter NAME="..." TYPE="..." DATATYPE="..."/>
  </Parameters>
  <Declaration>
    <Define NAME="..." DATATYPE="...">value</Define>
  </Declaration>
  <Logic>
    <Variable NAME="..." TYPE="..." FUNCTIONNAME="..." DATASOURCE="..."
NAMESPACE="..." OBJECT="...">
      <Parameters> ... </Parameters>
    </Variable>
    <IF CONDITION="...">
      <Variable> ... </Variable>
      <IF> ... </IF>
      <LOOP> ... </LOOP>
    </IF>
    <LOOP CONDITION="...">
      <Variable> ... </Variable>
      <IF> ... </IF>
      <LOOP> ... </LOOP>
    </LOOP>
  </Logic>
</Function>
```

ภาพที่ 3.3 โครงสร้างในการเขียนกฎ

ตารางที่ 3.1 ประเภทของข้อมูลในการเขียนกฎ

ประเภทข้อมูล	**คำอธิบาย**
STRING	ข้อมูลสายอักขระ
INTERGER	จำนวนเต็ม
DECIMAL	เลขทศนิยม
BOOLEAN	ตรรกะแบบบูล ค่าที่เป็นไปได้ คือ True หรือ False
DATE	วันที่
XML	ข้อมูลที่อยู่ในรูปแบบเอกซ์เอ็มแอล

แต่ละอิลิเมนต์ภายในโครงสร้างจากภาพที่ 3.3 สามารถอธิบาย ได้ดังนี้

1) อิลิเมนต์ Function เป็นอิลิเมนต์ราก ซึ่งกฎทุกกฎจะต้องเริ่มต้นและจบด้วยอิลิเมนต์นี้เท่านั้น ภายในประกอบด้วย 3 อิลิเมนต์ย่อย คือ อิลิเมนต์ Parameters อิลิเมนต์ Declaration และอิลิเมนต์ Logic โดยแอตทริบิวต์ของอิลิเมนต์ Function เป็นดังตารางที่ 3.2

ตารางที่ 3.2 แอตทริบิวต์ของอิลิเมนต์ Function

แอตทริบิวต์	**คำอธิบาย**
NAME	ชื่อของฟังก์ชันหรือกฎ
RETURN	ชื่อตัวแปรที่ต้องการส่งค่ากลับ เมื่อประมวลผลกฎเสร็จสิ้น
DATATYPE	ประเภทข้อมูลของค่าที่ต้องการส่งกลับ

2) อิลิเมนต์ Parameters เป็นอิลิเมนต์ที่ใช้กำหนดพารามิเตอร์ของกฎ ว่าต้องการใช้ข้อมูลนำเข้า และข้อมูลนำออกอะไรบ้าง ในการทำงานระหว่างผู้เรียกใช้กับเครื่องประมวลผลกฎ ภายในประกอบด้วย อิลิเมนต์ Parameter ซึ่งมีแอตทริบิวต์ดังตารางที่ 3.3

ตารางที่ 3.3 แอตทริบิวต์ของอิลิเมนต์ Parameter

แอตทริบิวต์	**คำอธิบาย**
NAME	ชื่อของพารามิเตอร์
TYPE	ประเภทของพารามิเตอร์ ว่าเป็นข้อมูลนำเข้า หรือข้อมูลนำออก ค่าที่เป็นไปได้ INPUT หรือ OUTPUT
DATATYPE	ประเภทข้อมูลของพารามิเตอร์

3) อิลิเมนต์ Declaration เป็นอิลิเมนต์ที่ใช้ในการประกาศตัวแปรที่ใช้ในกฎ ภายในประกอบด้วย อิลิเมนต์ Define ซึ่งมีแอตทริบิวต์ดังตารางที่ 3.4

ตารางที่ 3.4 แอตทริบิวต์ของอิลิเมนต์ Define

แอตทริบิวต์	**คำอธิบาย**
NAME	ชื่อตัวแปร
DATATYPE	ประเภทข้อมูลของตัวแปร

4) อิลิเมนต์ Logic เป็นอิลิเมนต์ที่ใช้ในการกำหนดตรรกะการทำงานของกฎภายในประกอบด้วย 3 อิลิเมนต์ย่อยที่เป็นไปได้ คือ อิลิเมนต์ Variable อิลิเมนต์ IF และอิลิเมนต์ LOOP

4.1) อิลิเมนต์ Variable ใช้สำหรับกำหนดการดำเนินการให้กับตัวแปร มีแอตทริบิวต์ตามตารางที่ 3.5 และประเภทการดำเนินการแสดงในตารางที่ 3.6 ซึ่งแต่ละการดำเนินการต้องการแอตทริบิวต์ไม่เหมือนกัน ดังตารางที่ 3.7 โดยถ้าหากกำหนดว่าเป็นการดำเนินการเพื่อเรียกใช้ฟังก์ชันอื่น จะมีอิลิเมนต์ Parameters เป็นอิลิเมนต์ย่อย

ตารางที่ 3.5 แอตทริบิวต์ของอิลิเมนต์ Variable

แอตทริบิวต์	**คำอธิบาย**
NAME	ชื่อตัวแปรที่ต้องการใช้ดำเนินการ
TYPE	ประเภทการดำเนินการ
FUNCTIONNAME	ชื่อฟังก์ชันหรือกฎที่ต้องการเรียกใช้
DATASOURCE	ข้อมูลการเข้าถึงฐานข้อมูล
NAMESPACE	ชื่อเนมสเปซของคลาสที่ต้องการเข้าถึง เพื่อใช้งาน
OBJECT	ชื่อคลาสที่ต้องการเรียกใช้ หรือตัวแปรอื่นที่ต้องการอ้างถึง

ตารางที่ 3.6 ประเภทการดำเนินการของอิลิเมนต์ Variable

การดำเนินการ	**คำอธิบาย**
VALUE	ใช้สำหรับกำหนดค่าข้อมูล
EXPRESSION	ใช้ในการคำนวณ หรือเรียกใช้คำสั่งของ .Net ที่อยู่ในเนมสเปซ Microsoft.VisualBasic
FUNCTIONCALL	เรียกใช้ฟังก์ชันหรือกฎอื่น
FUNCTIONCALLDOTNET	เรียกใช้กฎจากไฟล์ดีแอลแอล
SQL	ใช้คำสั่งเอสคิวแอลในการดึงข้อมูลจากฐานข้อมูล
XPATH	ใช้เอกซ์พาธเพื่อเข้าถึงข้อมูลภายในตัวแปรที่เป็นเอกซ์เอ็มแอล

ตารางที่ 3.7 ความสัมพันธ์ระหว่างประเภทการดำเนินการและการกำหนดแอตทริบิวต์ในอิลิเมนต์ Variable

		แอตทริบิวต์					
		NAME	TYPE	DATASOURCE	FUNCTIONNAME	NAMESPACE	OBJECT
การดำเนินการ	VALUE	✓	✓				
	EXPRESSION	✓	✓				
	FUNCTIONCALL	✓	✓		✓		
	FUNCTIONCALLDOTNET	✓	✓			✓	✓
	SQL	✓	✓	✓			
	XPATH	✓	✓				✓

4.2) อิลิเมนต์ IF และอิลิเมนต์ LOOP มีแอตทริบิวต์เดียวคือ CONDITION เพื่อใช้ในการตรวจสอบเงื่อนไข ก่อนทำงานกับอิลิเมนต์ที่อยู่ภายในต่อไป ซึ่งอิลิเมนต์ย่อยสามารถเป็นได้ทั้ง อิลิเมนต์ Variable อิลิเมนต์ IF และอิลิเมนต์ LOOP โดยการใช้ตัวดำเนินการ (Operator) ในแอตทริบิวต์ CONDITION แสดงในตารางที่ 3.8

ตารางที่ 3.8 ตัวดำเนินการที่ใช้ในกฎ

ประเภท	**ตัวดำเนินการใน VB.NET**	**ตัวดำเนินการในกฎ**
การเปรียบเทียบ (Comparison)	= เท่ากับ (Equality)	=
	<> ไม่เท่ากับ (Inequality)	<>
	< น้อยกว่า (Less than)	&lt;
	<= น้อยกว่าเท่ากับ (Less than or equal to)	<=
	> มากกว่า (Greater than)	&gt;
	>= มากกว่าเท่ากับ (Greater than or equal to)	>=
ตรรกะ (Logical)	And ประพจน์เชื่อม (Conjunction)	And
	Or ประพจน์เลือก (Disjunction)	Or
	Not นิเสธ (Negation)	Not

ตารางที่ 3.8 ตัวดำเนินการที่ใช้ในกฎ (ต่อ)

ประเภท	**ตัวดำเนินการใน** VB.NET	**ตัวดำเนินการในกฎ**
การคำนวณ (Arithmetic)	+ บวก (Addition)	+
	- ลบ (Subtraction)	-
	* คูณ (Multiplication)	*
	/ หาร (Division)	/
	Mod หารเอาเศษ (Modulus)	%
	^ ยกกำลัง (Exponentiation)	^

การอ้างอิงถึงตัวแปรใดๆ ภายในกฎใช้ สัญลักษณ์ [$ ตามด้วยชื่อตัวแปร และปิดท้ายด้วย] เช่น หากต้องการอ้างถึงตัวแปรชื่อ mVariable สามารถเขียนได้ว่า [$mVariable] ดังในตัวอย่างภาพที่ 3.4 ที่เป็นตัวอย่างกฎสำหรับการหาประเภทสามเหลี่ยม และตารางที่ 3.9 เป็นการอธิบายรายละเอียดของกฎจากภาพที่ 3.4

```
<Function NAME="Function-TriangleType" RETURN="vTriangle" DATATYPE="BOOLEAN">
  <Parameters>
    <Parameter NAME="iX" TYPE="INPUT" DATATYPE="INTEGER"></Parameter>
    <Parameter NAME="iY" TYPE="INPUT" DATATYPE="INTEGER"></Parameter>
    <Parameter NAME="iZ" TYPE="INPUT" DATATYPE="INTEGER"></Parameter>
    <Parameter NAME="oType" TYPE="OUTPUT" DATATYPE="STRING">[$vType]</Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="X" DATATYPE="INTEGER">[$iX]</Define>
    <Define NAME="Y" DATATYPE="INTEGER">[$iY]</Define>
    <Define NAME="Z" DATATYPE="INTEGER">[$iZ]</Define>
    <Define NAME="vTriangle" DATATYPE="BOOLEAN">FALSE</Define>
    <Define NAME="vType" DATATYPE="STRING">Not a Triangle</Define>
  </Declaration>
  <Logic>
    <IF CONDITION="[$X] &gt; 0 And [$Y] &gt; 0 And [$Z] &gt;0 And
[$X] &lt; ([$Y] + [$Z]) And [$Y] &lt; ([$X] + [$Z]) And [$Z] &lt; ([$X] + [$Y])">
      <Variable NAME="vTriangle" TYPE="VALUE">TRUE</Variable>
      <IF CONDITION="[$X] = [$Y] And [$Y] = [$Z]">
        <Variable NAME="vType" TYPE="VALUE">Equilateral</Variable>
      </IF>
      <IF CONDITION="[$X] &lt;&gt; [$Y] And [$X] &lt;&gt; [$Z] And
[$Y] &lt;&gt; [$Z]">
        <Variable NAME="vType" TYPE="VALUE">Scalene</Variable>
      </IF>
      <IF CONDITION="([$X] = [$Y] And [$X] &lt;&gt; [$Z]) Or
([$X] = [$Z] And [$X] &lt;&gt; [$Y]) Or ([$Y] = [$Z] And [$X] &lt;&gt; [$Z])">
        <Variable NAME="vType" TYPE="VALUE">Isosceles</Variable>
      </IF>
    </IF
  </Logic>
</Function>
```

ภาพที่ 3.4 ตัวอย่างการเขียนกฎสำหรับตรวจสอบประเภทสามเหลี่ยม

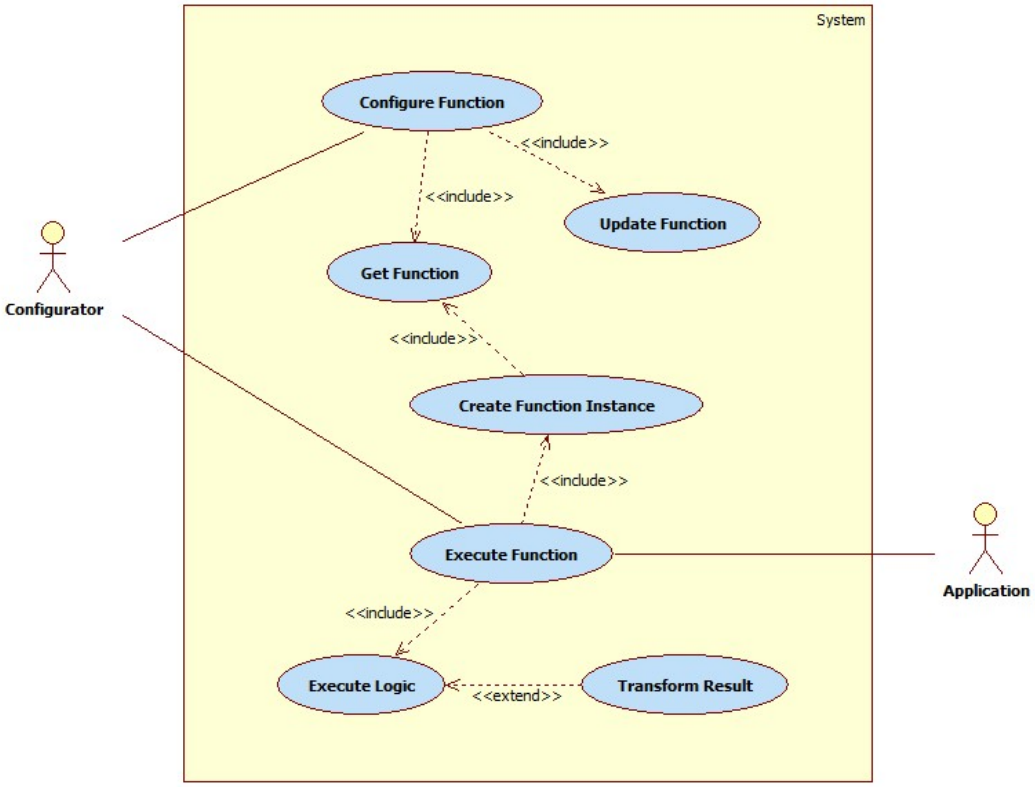
ตารางที่ 3.9 รายละเอียดของกฎสำหรับตรวจสอบประเภทสามเหลี่ยม

Rule Name	Function-TriangleType		
	Name	**Datatype**	**Value**
Return	vTriangle	BOOLEAN	refer to vTriangle
Input Parameter	iX	INTEGER	
	iY	INTEGER	
	iZ	INTEGER	
Output Parameter	oType	STRING	refer to vType
Declaration	X	INTEGER	refer to iX
	Y	INTEGER	refer to iY
	Z	INTEGER	refer to iZ
	vTriangle	BOOLEAN	FALSE
	vType	STRING	Not a Triangle
	Condition		**Result**
Logic	X > 0 and Y > 0 and Z > 0 and X < (Y + Z) and Y < (X + Z) and Z < (X + Y)		vTriangle = True
	X = Y and Y = Z		vType = Equilateral
	X <> Y and X <> Z and Y <> Z		vType = Scalene
	(X = Y and X <> Z) Or (X = Z and X <> Y) Or (Y = Z and X <> Z)		vType = Isosceles

3.3 การวิเคราะห์และออกแบบเครื่องมือ

3.3.1 ฟังก์ชันการทำงานของเครื่องมือ

แผนภาพยูสเคสแสดงฟังก์ชันการทำงานของเครื่องประมวลผล ในมุมมองของผู้ใช้ เป็นดังภาพที่ 3.5 และรายละเอียดของยูสเคสแสดงในตารางที่ 3.10 ถึงตารางที่ 3.16



ภาพที่ 3.5 แผนภาพยูสเคสของเครื่องประมวลผล

จากภาพที่ 3.5 สามารถอธิบายได้ดังนี้ ผู้ใช้นำกฎเข้าสู่ระบบ และจัดการกฎที่อยู่ในระบบ (Configure Function) โดยบันทึกกฎลงสู่คลังเก็บ (Update Function) และอ่านกฎจากคลังเก็บ (Get Function) เมื่อผู้ใช้หรือโปรแกรมอื่นต้องการประมวลผลกฎ (Execute Function) เครื่องประมวลผลจะทำการสร้างอินสแตนซ์ของกฎ (Create Function Instance) โดยการดึงกฎจากคลังเก็บด้วยชื่อกฎที่ระบุ แล้วทำการประมวลผลตรรกะภายในกฎ (Execute Logic) ซึ่งถ้าหากกฎนั้นมีการกำหนดเอกซ์เอสแอลทีเอาไว้ ก็จะทำการแปลงค่าผลลัพธ์ (Transform Result) ก่อนส่งกลับสู่ผู้ใช้หรือโปรแกรมที่เรียกใช้

ตารางที่ 3.10 รายละเอียดยูสเคสการจัดการกฎ

Use case:	Configure Function
Actors:	Configurator
Goal:	นำกฎเข้าสู่ระบบ และจัดการกฎที่อยู่ในระบบ
Related use cases:	Include: Update Function, Get Function
Preconditions:	กำหนดคลังเก็บไว้ถูกต้อง
Steps:	1. ผู้ใช้เรียกใช้เครื่องมือผ่านทางเว็บเบราว์เซอร์ 2. เข้าเมนูการจัดการ เลือกสร้างกฎใหม่ หรือเลือกกฎที่ต้องการแก้ไข 3. ระบบเข้าสู่หน้าการกำหนดกฎ 4. ผู้ใช้ทำการเพิ่มกฎ หรือแก้ไขกฎตามที่ต้องการ และกดปุ่มบันทึก
Postconditions:	ระบบกลับเข้าสู่หน้าจัดการ

ตารางที่ 3.11 รายละเอียดยูสเคสการบันทึกกฎ

Use case:	Update Function
Actors:	-
Goal:	บันทึกกฎลงสู่คลังเก็บ
Related use cases:	-
Preconditions:	กฎมีโครงสร้างที่ถูกต้องตามข้อกำหนดในการเขียนกฎ
Steps:	1. ระบบทำการตรวจสอบไวยากรณ์ 2. ระบบบันทึกกฎลงสู่คลังเก็บ
Postconditions:	กฎถูกบันทึกไว้ในคลังเก็บตามที่กำหนด

ตารางที่ 3.12 รายละเอียดยูสเคสการดึงข้อมูลกฎ

Use case:	Get Function
Actors:	-
Goal:	อ่านกฎจากคลังเก็บ
Related use cases:	-
Preconditions:	มีกฎอยู่ในคลังเก็บ

ตารางที่ 3.12 รายละเอียดยูสเคสการดึงข้อมูลกฎ (ต่อ)

Steps:	ระบบดึงข้อมูลของกฎจากคลังเก็บตามชื่อกฎที่ระบุ
Postconditions:	-

ตารางที่ 3.13 รายละเอียดยูสเคสการประมวลผลกฎ

Use case:	Execute Function
Actors:	Configurator, Application
Goal:	บันทึกกฎลงสู่คลังเก็บ
Related use cases:	Include: Create Function Instance, Execute Logic
Preconditions:	มีกฎอยู่ในคลังเก็บ
Steps:	1. ผู้เรียกใช้ ระบุชื่อกฎและข้อมูลนำเข้า 2. ระบบทำการสร้างอินสแตนสซ์ของกฎ 3. ผู้เรียกใช้สั่งประมวลผลกฎจากอินสแตนสซ์ที่สร้างขึ้น
Postconditions:	เครื่องประมวลผลส่งผลลัพธ์กลับสู่ผู้เรียกใช้

ตารางที่ 3.14 รายละเอียดยูสเคสการสร้างอินสแตนซส์ของกฎ

Use case:	Create Function Instance
Actors:	-
Goal:	สร้างอินสแตนซส์ของกฎเพื่อทำการประมวลผล
Related use cases:	Include: Get Function
Preconditions:	ต้องระบุชื่อกฎและข้อมูลนำเข้า
Steps:	1. ระบบอ่านกฎจากคลังเก็บ 2. ระบบสร้างอินสแตนซส์ของกฎ 3. ระบบทำการกำหนดค่าข้อมูลนำเข้าให้กับอินสแตนซส์ที่สร้างขึ้น
Postconditions:	อินสแตนซส์ของกฎพร้อมในการประมวลผล

ตารางที่ 3.15 รายละเอียดยูสเคสการประมวลผลตรรกะภายในกฎ

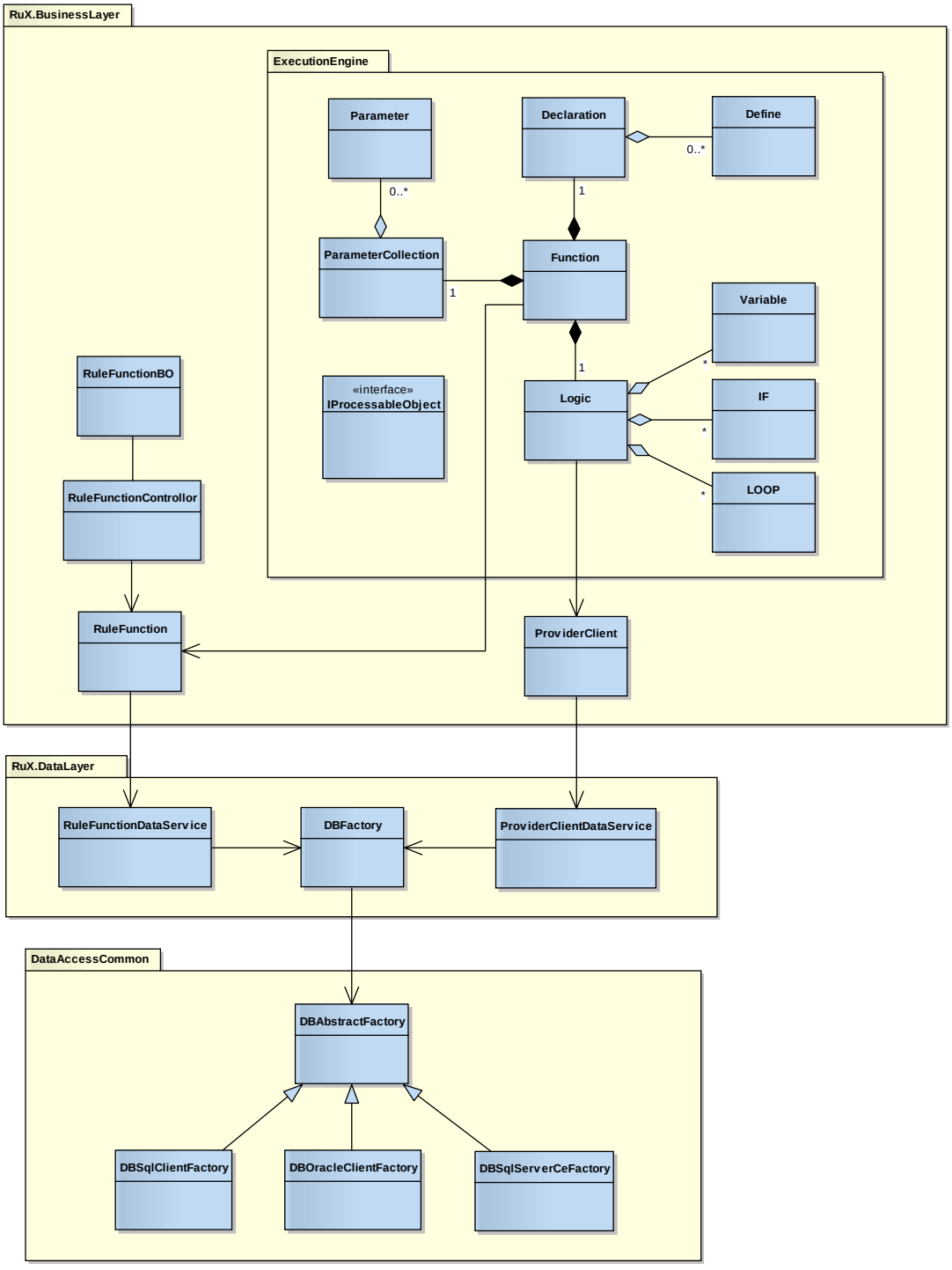
Use case:	Execute Logic
Actors:	-
Goal:	ทำการประมวลผลตรรกะของกฎ
Related use cases:	Exclude: Transform Result
Preconditions:	มีอินสแตนสซ์ของกฎแล้ว
Steps:	1. ระบบทำการอ่านคำสั่งภายในกฎตามลำดับ 2. ระบบทำการประมวลผลชุดคำสั่ง 3. ระบบแปลงค่าผลลัพธ์ ในกรณีที่มีการกำหนดเอกซ์เอสแอลทีไว้
Postconditions:	ผลลัพธ์ของการประมวลผลจากข้อมูลนำเข้า ถูกส่งค่ากลับ

ตารางที่ 3.16 รายละเอียดยูสเคสการแปลงค่าผลลัพธ์

Use case:	Transform Result
Actors:	-
Goal:	แปลงค่าผลลัพธ์ให้อยู่ในรูปแบบที่ต้องการ
Related use cases:	-
Preconditions:	เอกซ์เอสแอลทีถูกกำหนดไว้ สำหรับกฎที่สั่งประมวลผล
Steps:	ระบบแปลงค่าผลลัพธ์ตามเอกซ์เอสแอลทีที่กำหนดไว้
Postconditions:	ผลลัพธ์ถูกแปลงค่า

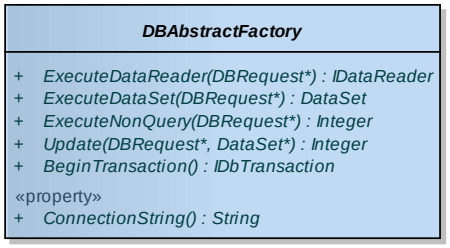
3.3.2 โครงสร้างส่วนประกอบของเครื่องมือ

แผนภาพคลาสของเครื่องประมวลผล แสดงโครงสร้างความสัมพันธ์ของคลาส แต่ละคลาสในระบบ เป็นดังภาพที่ 3.6 และมีรายละเอียดของคลาสต่างๆ ดังต่อไปนี้



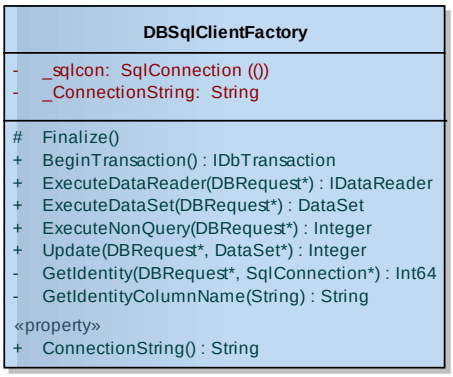
ภาพที่ 3.6 แผนภาพคลาสของเครื่องประมวลผล

1) คลาส DBAbstractFactory เป็นคลาสนามธรรม ที่เอาไว้ให้คลาสที่มีการติดต่อกับฐานข้อมูล มาสืบทอดไปใช้งาน รายละเอียดของคลาสเป็นดังภาพที่ 3.7



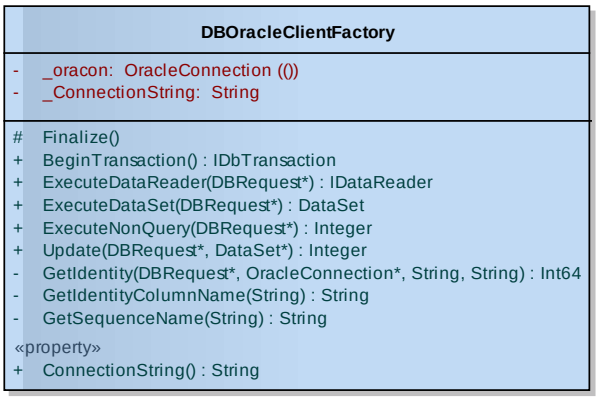
ภาพที่ 3.7 คลาส DBAbstractFactory

2) คลาส DBSqlClientFactory สืบทอดมาจากคลาส DBAbstractFactory เป็นคลาสที่ทำหน้าที่ในการติดต่อกับฐานข้อมูลซีควลเซิร์ฟเวอร์ รายละเอียดของคลาสเป็นดังภาพที่ 3.8



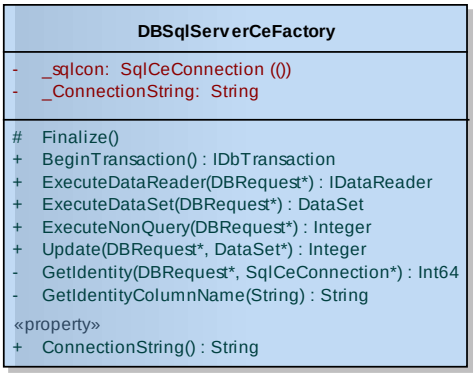
ภาพที่ 3.8 คลาส DBSqlClientFactory

3) คลาส DBOracleClientFactory สืบทอดมาจากคลาส DBAbstractFactory เป็นคลาสที่ทำหน้าที่ในการติดต่อกับฐานข้อมูลออราเคิล รายละเอียดของคลาสเป็นดังภาพที่ 3.9



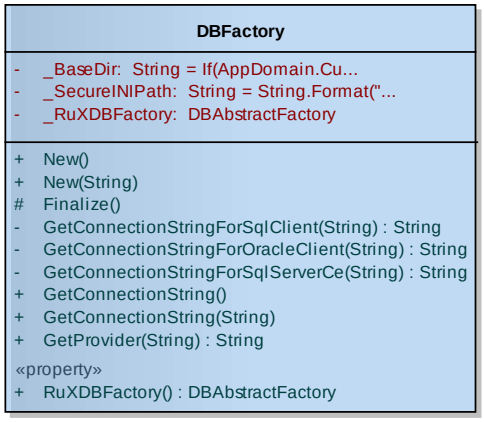
ภาพที่ 3.9 คลาส DBOracleClientFactory

4) คลาส DBSqlServerCeFactory สืบทอดมาจากคลาส DBAbstractFactory เป็นคลาสที่ทำหน้าที่ในการติดต่อกับฐานข้อมูลเอสคิวแอลซีอี รายละเอียดของคลาสเป็นดังภาพที่ 3.10



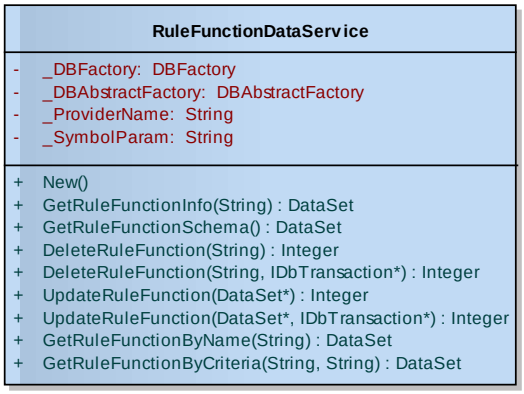
ภาพที่ 3.10 คลาส DBSqlServerCeFactory

5) คลาส DBFactory เป็นคลาสที่ทำหน้าที่ในการกำหนดข้อมูล ที่ใช้สำหรับติดต่อกับฐานข้อมูล ที่เป็นคลังเก็บของกฎ รายละเอียดของคลาสเป็นดังภาพที่ 3.11



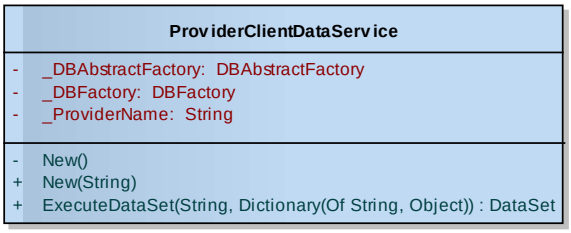
ภาพที่ 3.11 คลาส DBFactory

6) คลาส RuleFunctionDataService เป็นคลาสที่ทำหน้าที่ในการอ่านกฎ และจัดการกฎในฐานข้อมูล รายละเอียดของคลาสเป็นดังภาพที่ 3.12



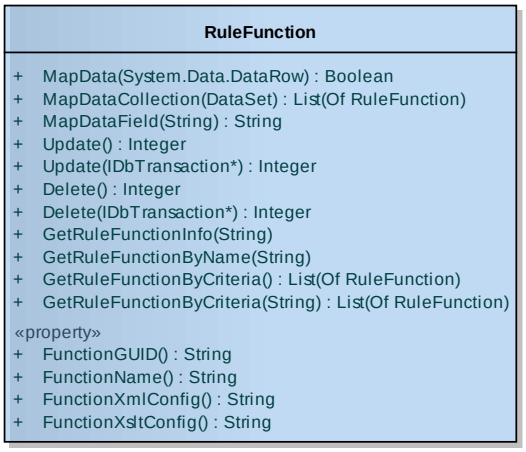
ภาพที่ 3.12 คลาส RuleFunctionDataService

7) คลาส ProviderClientDataService เป็นคลาสที่ทำหน้าที่ในการดึงข้อมูลจากฐานข้อมูล ตามที่กำหนดในชุดคำสั่งของกฎ รายละเอียดของคลาสเป็นดังภาพที่ 3.13



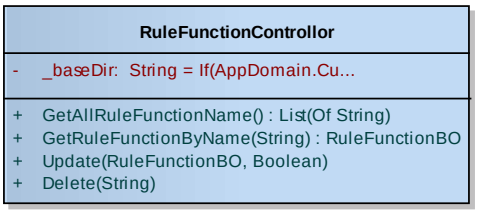
ภาพที่ 3.13 คลาส ProviderClientDataService

8) คลาส RuleFunction เป็นคลาสที่ทำหน้าที่ในการแปลงกฎ ระหว่างรูปแบบข้อมูลจากฐานข้อมูลและอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.14



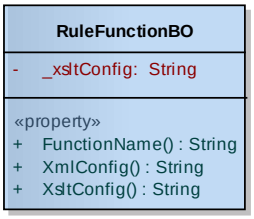
ภาพที่ 3.14 คลาส RuleFunction

9) คลาส RuleFunctionControllor เป็นคลาสที่ทำหน้าที่ในการอ่านกฎ และจัดการกฎในคลังเก็บ ซึ่งอาจเป็นฐานข้อมูล หรือไฟล์เอกซ์เอ็มแอล รายละเอียดของคลาสเป็นดังภาพที่ 3.15



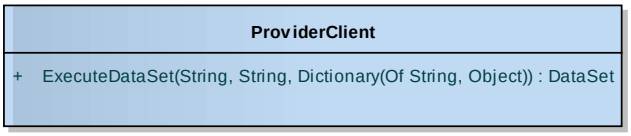
ภาพที่ 3.15 คลาส RuleFunctionControllor

10) คลาส RuleFunctionBO เป็นคลาสข้อมูลของกฎที่ได้จากฐานข้อมูล ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.16



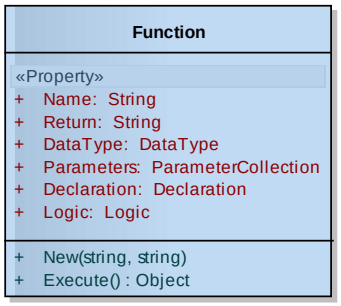
ภาพที่ 3.16 คลาส RuleFunctionBO

11) คลาส ProviderClient เป็นคลาสที่ทำหน้าที่ในการกำหนดข้อมูล ที่ใช้สำหรับติดต่อกับฐานข้อมูล ตามที่กำหนดในชุดคำสั่งของกฎ รายละเอียดของคลาสเป็นดังภาพที่ 3.17



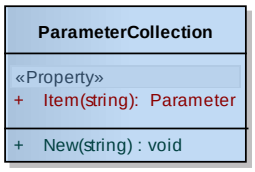
ภาพที่ 3.17 คลาส ProviderClient

12) คลาส Function เป็นคลาสตัวแทนของกฎ หรืออิลิเมนต์ Function ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.18



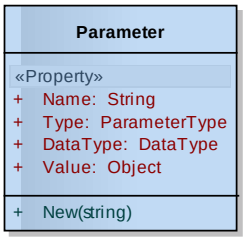
ภาพที่ 3.18 คลาส Function

13) คลาส ParameterCollection เป็นคลาสตัวแทนของอิลิเมนต์ Parameters ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.19



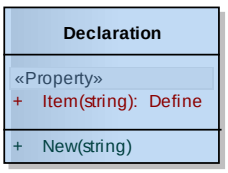
ภาพที่ 3.19 คลาส ParameterCollection

14) คลาส Parameter เป็นคลาสตัวแทนของอิลิเมนต์ Parameter ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.20



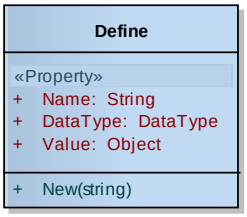
ภาพที่ 3.20 คลาส Parameter

15) คลาส Declaration เป็นคลาสตัวแทนของอิลิเมนต์ Declaration ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.21



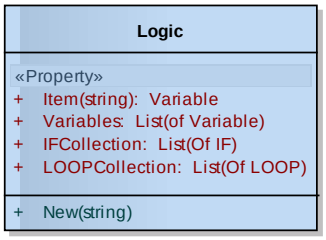
ภาพที่ 3.21 คลาส Declaration

16) คลาส Define เป็นคลาสตัวแทนของตัวแปรที่ประกาศในกฎ หรืออิลิเมนต์ Define ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.22



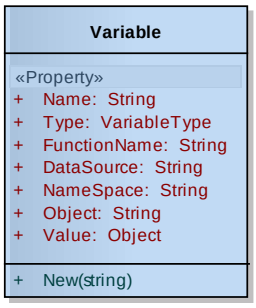
ภาพที่ 3.22 คลาส Define

17) คลาส Logic เป็นคลาสตัวแทนของตรรกะภายในกฎ หรืออิลิเมนต์ Logic ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.23



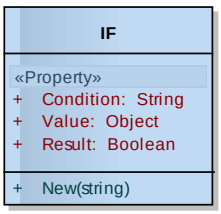
ภาพที่ 3.23 คลาส Logic

18) คลาส Variable เป็นคลาสตัวแทนของชุดคำสั่งการดำเนินการตัวแปร หรืออิลิเมนต์ Variable ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.24



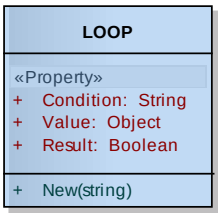
ภาพที่ 3.24 คลาส Variable

19) คลาส IF เป็นคลาสตัวแทนของชุดคำสั่งแบบมีเงื่อนไข หรืออิลิเมนต์ IF ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.25



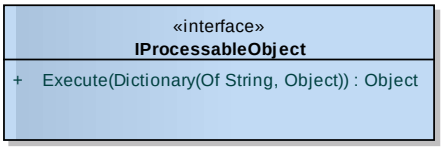
ภาพที่ 3.25 คลาส IF

20) คลาส LOOP เป็นคลาสตัวแทนของชุดคำสั่งแบบวนซ้ำ หรืออิลิเมนต์ LOOP ที่อยู่ในรูปแบบอ็อบเจกต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.26



ภาพที่ 3.26 คลาส LOOP

21) คลาส IProcessableObject เป็นคลาสอินเทอร์เฟซ ที่เอาไว้ให้คลาสที่ต้องการให้กฎสามารถสั่งประมวลผลได้ นำไปอิมพลีเมนต์ รายละเอียดของคลาสเป็นดังภาพที่ 3.27



ภาพที่ 3.27 คลาส IProcessableObject

3.3.3 ขั้นตอนการทำงานของเครื่องมือ

ข้อมูลนำเข้าที่ส่งให้เครื่องประมวลผล พร้อมกับชื่อกฎที่ต้องการเรียกใช้ แสดงดังภาพที่ 3.28 ซึ่งเป็นข้อมูลนำเข้าของกฎในภาพที่ 3.4

```
<Parameters>
  <Parameter NAME="iX">10</Parameter>
  <Parameter NAME="iY">5</Parameter>
  <Parameter NAME="iZ">10</Parameter>
</Parameters>
```

ภาพที่ 3.28 ตัวอย่างข้อมูลนำเข้า

1) การประมวลผลกฎ ดังภาพที่ 3.29 มีขั้นตอนการทำงานดังนี้

1.1) ผู้เรียกใช้ระบุชื่อกฎ และข้อมูลนำเข้า ให้กับเครื่องประมวลผล

1.2) เครื่องประมวลผลทำการค้นหากฎตามชื่อที่ระบุ จากในคลังเก็บ และทำการสร้างอินสแตนซส์ของกฎ

1.3) กำหนดค่าข้อมูลให้กับอินพุตพารามิเตอร์ จากข้อมูลนำเข้า

1.4) กำหนดค่าเริ่มต้น หรือค่าข้อมูลให้กับตัวแปร จากอินพุตพารามิเตอร์ และการอ้างอิงถึงตัวแปรอื่น ที่ได้มีการประกาศเอาไว้ก่อนหน้า ตามลำดับ

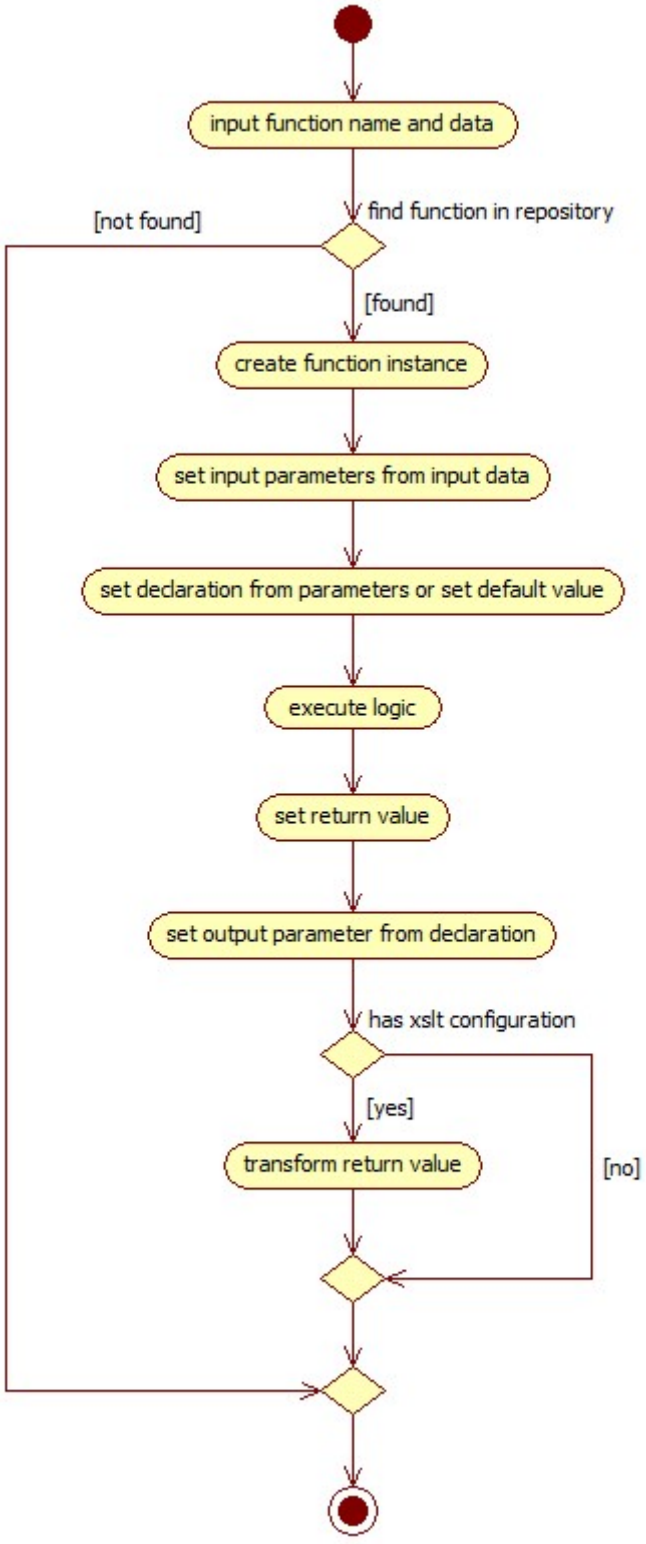
1.5) ประมวลผลชุดคำสั่งที่กำหนดไว้ ตามแต่ละชนิดการดำเนินงาน ซึ่งแยกได้เป็น

- ชุดคำสั่งการดำเนินการตัวแปร (Variable) ทำงานตามการดำเนินการที่กำหนด
- ชุดคำสั่งแบบมีเงื่อนไข (IF) ตรวจสอบเงื่อนไข ถ้าเป็นจริง จึงทำงานคำสั่งที่อยู่ภายใน
- ชุดคำสั่งแบบวนซ้ำ (LOOP) ทำงานคำสั่งที่อยู่ภายใน จนกว่าเงื่อนไขจะเป็นเท็จ

1.6) กำหนดค่าข้อมูลส่งกลับของฟังก์ชัน จากตัวแปรที่อ้างถึง

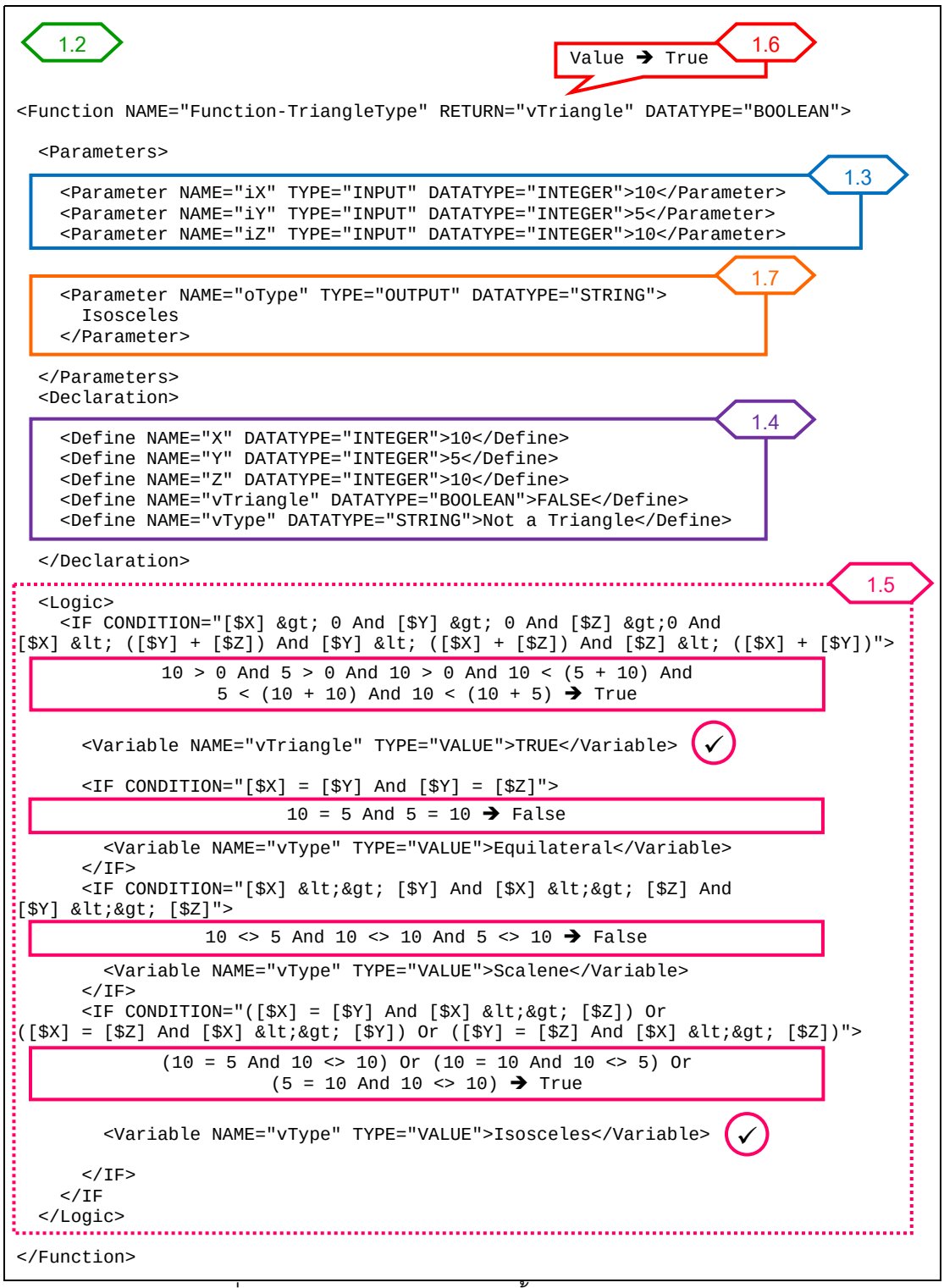
1.7) กำหนดค่าข้อมูลให้กับเอาท์พุตพารามิเตอร์ ตามที่ได้อ้างอิงตัวแปรเอาไว้

1.8) ทำการแปลงค่าข้อมูลส่งกลับด้วยเอกซ์เอสแอลที ถ้าหากมีการกำหนดไว้



ภาพที่ 3.29 แผนภาพกิจกรรมการประมวลผลกฎ

การประมวลผลกฎในแต่ละขั้นตอน จากกฎในภาพที่ 3.4 และข้อมูลนำเข้าในภาพที่ 3.28 เป็นดังภาพที่ 3.30



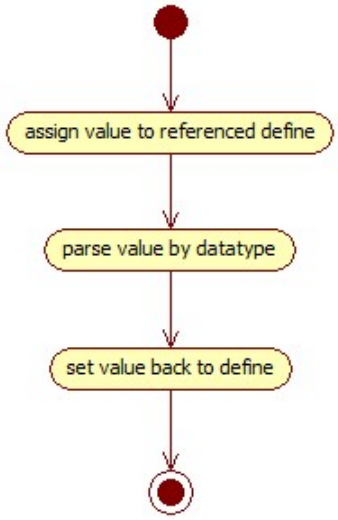
ภาพที่ 3.30 ตัวอย่างการทำงานในขั้นตอนการประมวลผลกฎ

2) การประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท VALUE ดังภาพที่ 3.31 มีขั้นตอนการทำงานดังนี้

2.1) กำหนดค่าข้อมูลแทนที่การอ้างอิงตัวแปรในชุดคำสั่ง

2.2) ตรวจสอบค่าข้อมูล ตามประเภทข้อมูลที่กำหนดของตัวแปร

2.3) กำหนดค่าข้อมูลคืนให้กับตัวแปรที่กำหนดในชุดคำสั่ง



ภาพที่ 3.31 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งการดำเนินการตัวแปรประเภท VALUE

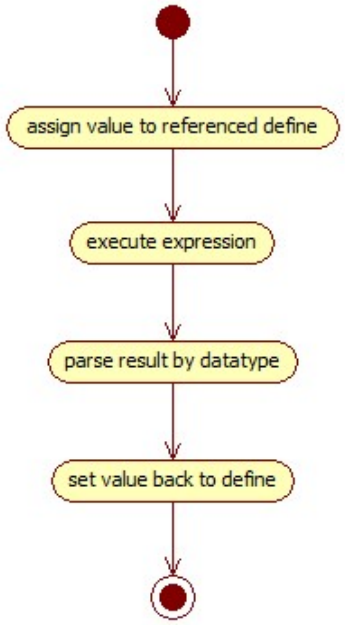
3) การประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท EXPRESSION ดังภาพที่ 3.32 มีขั้นตอนการทำงานดังนี้

3.1) กำหนดค่าข้อมูลแทนที่การอ้างอิงตัวแปรในชุดคำสั่ง

3.2) ประมวลผลนิพจน์

3.3) ตรวจสอบค่าข้อมูล ตามประเภทข้อมูลที่กำหนดของตัวแปร

3.4) กำหนดค่าข้อมูลคืนให้กับตัวแปรที่กำหนดในชุดคำสั่ง



ภาพที่ 3.32 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งการดำเนินการตัวแปร
ประเภท EXPRESSION

4) การประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท FUNCTIONCALL ดังภาพที่ 3.33 มีขั้นตอนการทำงานดังนี้

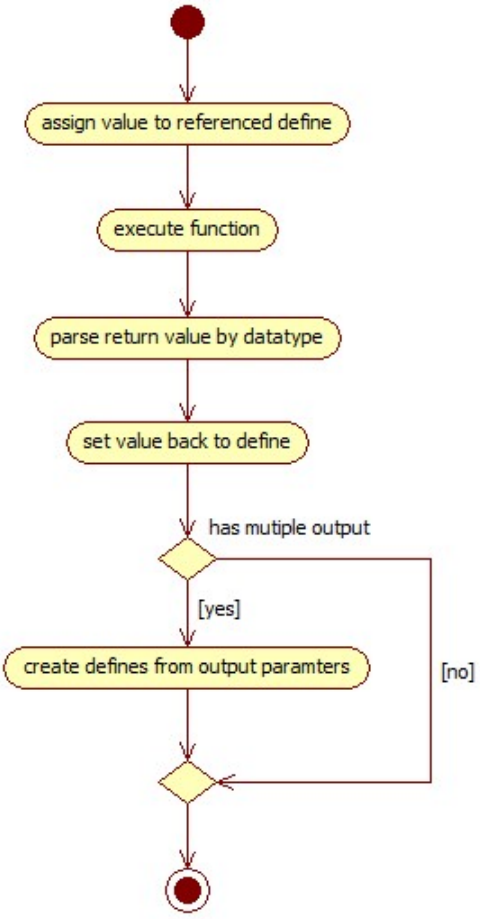
4.1) กำหนดค่าข้อมูลแทนที่การอ้างอิงตัวแปรในชุดคำสั่ง

4.2) สั่งประมวลผลกฎที่ระบุในชุดคำสั่ง

4.3) ตรวจสอบค่าผลลัพธ์ ตามประเภทข้อมูลที่กำหนดของตัวแปร

4.4) กำหนดค่าข้อมูลคืนให้กับตัวแปรที่กำหนดในชุดคำสั่ง

4.5) ถ้าหากกฎที่เรียกใช้งานมีการส่งค่าข้อมูลกลับแบบหลายค่า (มีหลาย Output Parameters) เครื่องประมวลผลจะทำการสร้างตัวแปรเพิ่ม เพื่อเก็บค่าข้อมูลเหล่านี้



ภาพที่ 3.33 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท FUNCTIONCALL

5) การประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท FUNCTIONCALLDOTNET ดังภาพที่ 3.34 มีขั้นตอนการทำงานดังนี้

5.1) กำหนดค่าข้อมูลแทนที่การอ้างอิงตัวแปรในชุดคำสั่ง

5.2) สร้างพารามิเตอร์เพื่อส่งให้ฟังก์ชันที่จะเรียกใช้

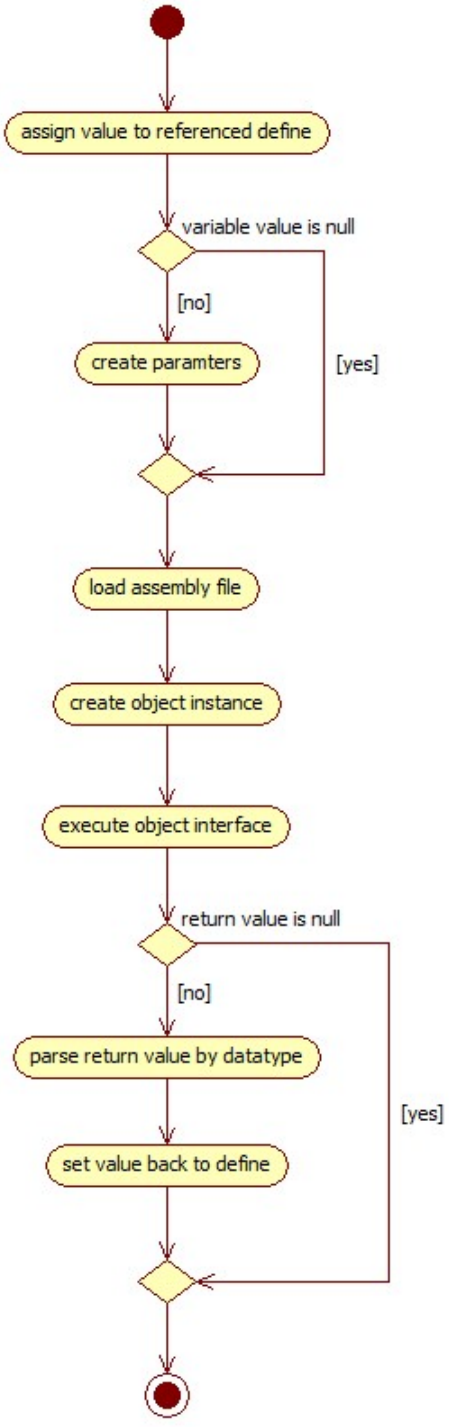
5.3) โหลดไฟล์แอสเซมบลี (Assembly file)

5.4) สร้างอินสแตนซส์อ็อบเจกต์ของฟังก์ชัน

5.5) สั่งประมวลผลฟังก์ชันผ่านอินเทอร์เฟซเมท็อด

5.6) ตรวจสอบค่าผลลัพธ์ ตามประเภทข้อมูลที่กำหนดของตัวแปร

5.7) กำหนดค่าข้อมูลคืนให้กับตัวแปรที่กำหนดในชุดคำสั่ง



ภาพที่ 3.34 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งการดำเนินการตัวแปรประเภท FUNCTIONCALLDOTNET

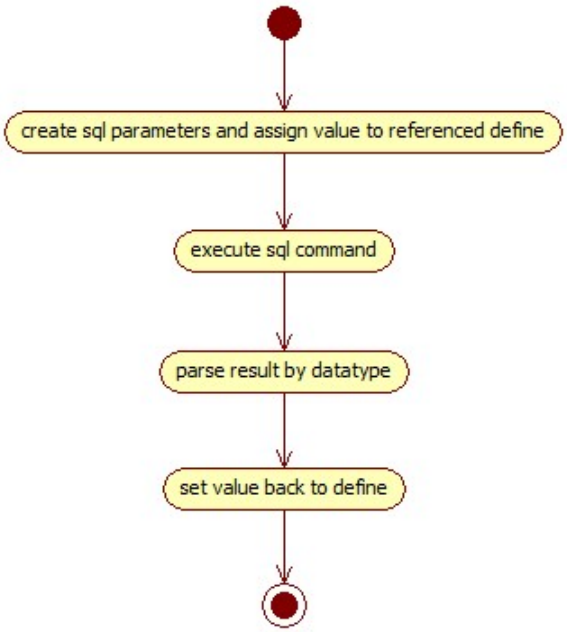
6) การประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท SQL ดังภาพที่ 3.35 มีขั้นตอนการทำงานดังนี้

6.1) สร้างเอสคิวแอลพารามิเตอร์ และกำหนดค่าข้อมูลแทนที่การอ้างอิงตัวแปรในชุดคำสั่ง

6.2) ประมวลผลคำสั่งเอสคิวแอล

6.3) ตรวจสอบค่าผลลัพธ์ ตามประเภทข้อมูลที่กำหนดของตัวแปร

6.4) กำหนดค่าข้อมูลคืนให้กับตัวแปรที่กำหนดในชุดคำสั่ง



ภาพที่ 3.35 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งการดำเนินการตัวแปรประเภท SQL

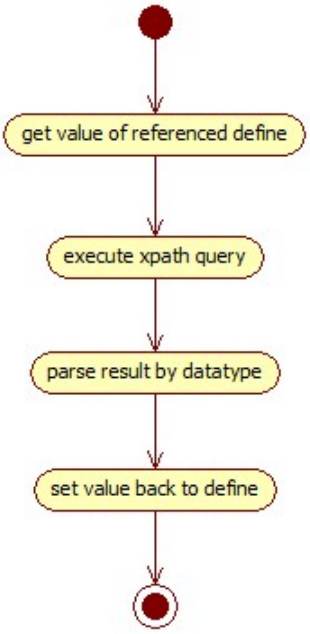
7) การประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท XPATH ดังภาพที่ 3.36 มีขั้นตอนการทำงานดังนี้

7.1) อ่านค่าตัวแปรที่ระบุในชุดคำสั่ง

7.2) ประมวลผลคำสั่งเอกซ์พาธ

7.3) ตรวจสอบค่าผลลัพธ์ ตามประเภทข้อมูลที่กำหนดของตัวแปร

7.4) กำหนดค่าข้อมูลคืนให้กับตัวแปรที่กำหนดในชุดคำสั่ง



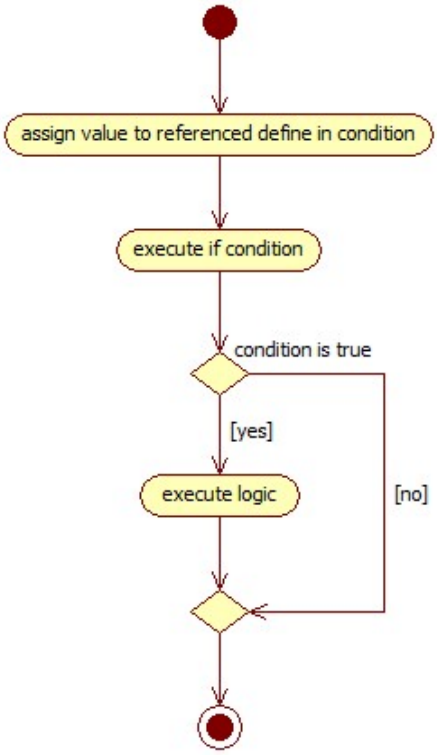
ภาพที่ 3.36 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งการดำเนินการตัวแปรประเภท XPATH

8) การประมวลผลชุดคำสั่งแบบมีเงื่อนไข ดังภาพที่ 3.37 มีขั้นตอนการทำงานดังนี้

8.1) กำหนดค่าข้อมูลแทนที่การอ้างอิงตัวแปรในนิพจน์เงื่อนไข

8.2) ประมวลผลเงื่อนไข

8.3) ถ้าเงื่อนไขเป็นจริง ให้ประมวลผลชุดคำสั่งที่อยู่ภายในต่อไป



ภาพที่ 3.37 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งแบบมีเงื่อนไข

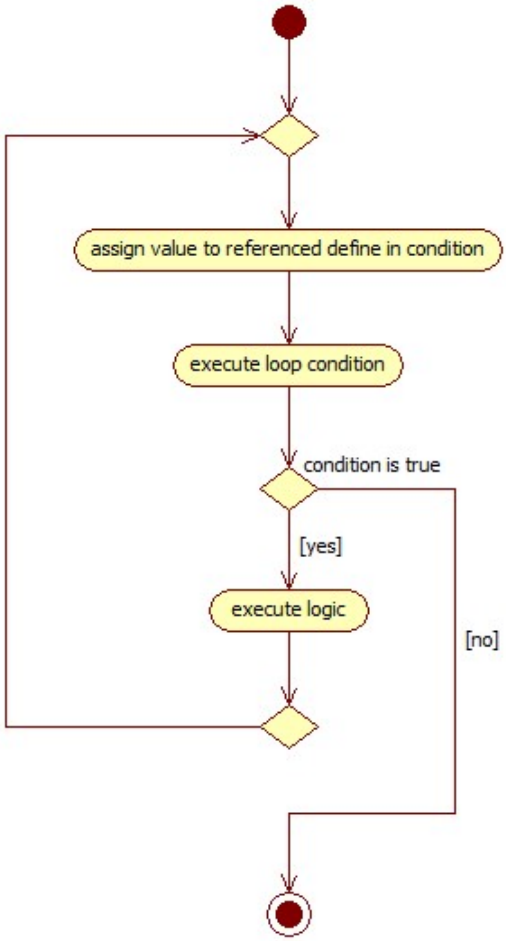
9) การประมวลผลชุดคำสั่งแบบวนซ้ำ ดังภาพที่ 3.38 มีขั้นตอนการทำงานดังนี้

9.1) กำหนดค่าข้อมูลแทนที่การอ้างอิงตัวแปรในนิพจน์เงื่อนไข

9.2) ประมวลผลเงื่อนไขการวนซ้ำ

9.3) ถ้าเงื่อนไขเป็นจริง ให้ประมวลผลชุดคำสั่งที่อยู่ภายใน

9.4) วนกลับไปทำข้อ 9.1 จนกว่าเงื่อนไขจะเป็นเท็จ



ภาพที่ 3.38 แผนภาพกิจกรรมการประมวลผลชุดคำสั่งแบบวนซ้ำ

บทที่ 4
การพัฒนาเครื่องมือ

4.1 สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ

1) ฮาร์ดแวร์ (Hardware)

1.1) เครื่องคอมพิวเตอร์โน๊ตบุค (Notebook) หน่วยประมวลผลอินเทลคอร์ทูดูโอ พี8700 2.53 กิกะเฮิรตซ์ (Intel® Core™ 2 Duo CPU P8700 2.53GHz)

1.2) หน่วยความจำสำรอง (RAM) 2.00 กิกะไบต์ (2.00 GB)

1.3) ฮาร์ดดิสก์ (Hard disk) 500 กิกะไบต์ (500 GB)

2) ซอฟต์แวร์ (Software)

2.1) ระบบปฏิบัติการ (Operating System) ไมโครซอฟท์วินโดวส์ 7 โฮมพรีเมี่ยม (Windows 7 Home Premium)

2.2) เครื่องมือที่ใช้พัฒนา วิชวลสตูดิโอดอทเน็ต 2010 อัลติเมท เซอร์วิสแพ็ค 1 (Visual Studio 2010 Ultimate Service Pack 1)

2.3) เฟรมเวิร์กที่ใช้พัฒนา ดอทเน็ตเฟรมเวิร์กสี่ (.NET Framework 4)

2.4) ฐานข้อมูล

- ซีควลเซิร์ฟเวอร์ 2008 เอกซ์เพรส (SQL Server 2008 Express Edition)
- ออราเคิล 11 จี เอกซ์เพรส (Oracle Database 11g Express Edition)
- ซีควลเซิร์ฟเวอร์คอมแพค หรือเอสคิวแอลซีอี 4.0 (SQL Server Compact 4.0)

2.5) เครื่องมือในการทดสอบ เอ็นยูนิต 2.5.10 (NUnit 2.5.10)

2.6) เว็บเซิร์ฟเวอร์ (Web Server) ไอไอเอส 7 (Internet Information Services – IIS 7)

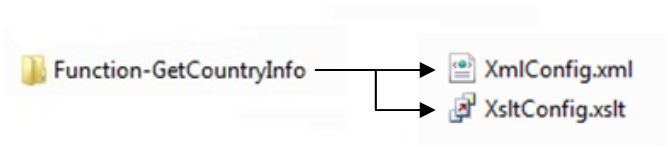
2.7) เว็บเบราว์เซอร์ (Web Browser) อินเทอร์เน็ตเอกซ์พลอเรอร์ 8 (Internet Explorer – IE 8)

4.2 การเก็บข้อมูลของเครื่องมือ

การจัดเก็บข้อมูลกฎของเครื่องประมวลผล สามารถทำได้ 2 แบบ คือ

1) ไฟล์เอกสาร

ชื่อกฎจะถูกใช้ในการตั้งเป็นชื่อแฟ้มเอกสาร ส่วนกฎที่สร้างขึ้นจะอยู่ภายใต้แฟ้มเอกสารนี้ เป็นไฟล์เอกสารเอกซ์เอ็มแอลที่ชื่อว่า “XmlConfig.xml” และถ้าหากกฎนี้มีการแปลงข้อมูลผลลัพธ์ด้วย ก็จะมีไฟล์เอกสารเอกซ์เอสแอลทีที่ชื่อว่า “XsltConfig.xslt” อยู่ภายใต้แฟ้มเอกสารเดียวกัน ดังตัวอย่างในภาพที่ 4.1 กฎมีชื่อว่า “Function-GetCountryInfo”



ภาพที่ 4.1 ตัวอย่างโครงสร้างกฎในรูปแบบไฟล์เอกสาร

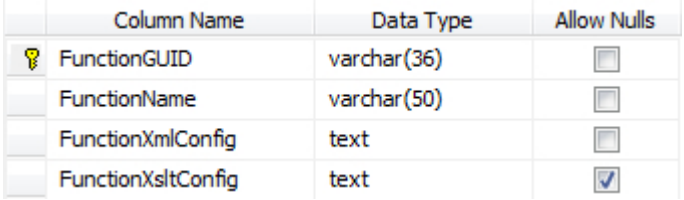
2) ฐานข้อมูล

กฎที่เก็บในฐานข้อมูล ต้องมีการตั้งชื่อคอลัมน์ (Column) ดังตารางที่ 4.1

ตารางที่ 4.1 ชื่อคอลัมน์สำหรับตารางเก็บข้อมูลกฎในฐานข้อมูล

ชื่อคอลัมน์	**คำอธิบาย**
FunctionGUID	เก็บข้อมูล Unique value ใช้เป็น Primary key
FunctionName	ชื่อกฎ
FunctionXmlConfig	ข้อมูลกฎ ซึ่งอยู่ในรูปแบบเอกซ์เอ็มแอล ตามข้อกำหนดในการเขียนกฎ
FunctionXsltConfig	ข้อมูลเอกซ์เอสแอลที สำหรับแปลงค่าผลลัพธ์

2.1) ซีควลเซิร์ฟเวอร์ มีโครงสร้างการเก็บข้อมูลกฎ ดังภาพที่ 4.2



ภาพที่ 4.2 โครงสร้างการเก็บข้อมูลกฎในซีควลเซิร์ฟเวอร์

2.2) ออราเคิล มีโครงสร้างการเก็บข้อมูลกฎ ดังภาพที่ 4.3

	COLUMN_NAME	DATA_TYPE	NULLABLE
1	FUNCTIONGUID	VARCHAR2(36 BYTE)	No
2	FUNCTIONNAME	VARCHAR2(50 BYTE)	No
3	FUNCTIONXMLCONFIG	CLOB	No
4	FUNCTIONXSLTCONFIG	CLOB	Yes

ภาพที่ 4.3 โครงสร้างการเก็บข้อมูลกฎในออราเคิล

2.3) เอสคิวแอลซีอี มีโครงสร้างการเก็บข้อมูลกฎ ดังภาพที่ 4.4

Column Name	Data Type	Length	Allow Nulls
FunctionGUID	nvarchar	36	No
FunctionName	nvarchar	50	No
FunctionXmlConfig	ntext	16	No
FunctionXsltConfig	ntext	16	Yes

ภาพที่ 4.4 โครงสร้างการเก็บข้อมูลกฎในเอสคิวแอลซีอี

4.3 ข้อกำหนดการใช้งานเครื่องมือ

การนำเครื่องประมวลผลไปใช้งาน ต้องมีการกำหนดคลังเก็บของกฎ ให้กับโปรแกรมที่ต้องการประมวลผลกฎ โดยมีรายละเอียดตามตารางที่ 4.2

ตารางที่ 4.2 การกำหนดข้อมูลคลังเก็บ

ชื่อคีย์ (Key)	**คำอธิบาย**
RuX_Connection	แหล่งข้อมูลที่ใช้สร้างการติดต่อกับฐานข้อมูล จากไฟล์ secure.ini
RuX_ConfigTable	ชื่อตารางเก็บข้อมูลกฎ ในกรณีที่คลังเก็บเป็นฐานข้อมูล
Plugin_Path	พาธภายในแอพพลิเคชัน ซึ่งเป็นที่อยู่ของไฟล์ดีแอลแอล (DLL) สำหรับการเรียกประมวลผลกฎภายนอก
App_Mode	โหมดของแอพพลิเคชัน ค่าที่เป็นไปได้ คือ - online กำหนดคลังเก็บเป็นฐานข้อมูล - offline กำหนดคลังเก็บในรูปแบบไฟล์เอกสาร
Source_Path	พาธภายในแอพพลิเคชัน ที่ใช้เป็นคลังเก็บ

1) วินโดวส์แอพพลิเคชัน (Windows Application) กำหนดข้อมูลคลังเก็บในไฟล์ app.config ดังตัวอย่างในภาพที่ 4.5

```
<configuration>
  <appSettings>
    <add key="RuX_Connection" value="DBAccess-3"/>
    <add key="RuX_ConfigTable" value="InternalFunctions"/>
    <add key="Plugin_Path" value="\"/>
    <add key="App_Mode" value="offline"/>
    <add key="Source_Path" value="\..\..\App_Data\"/>
  </appSettings>
     .
     .
     .
</configuration>
```

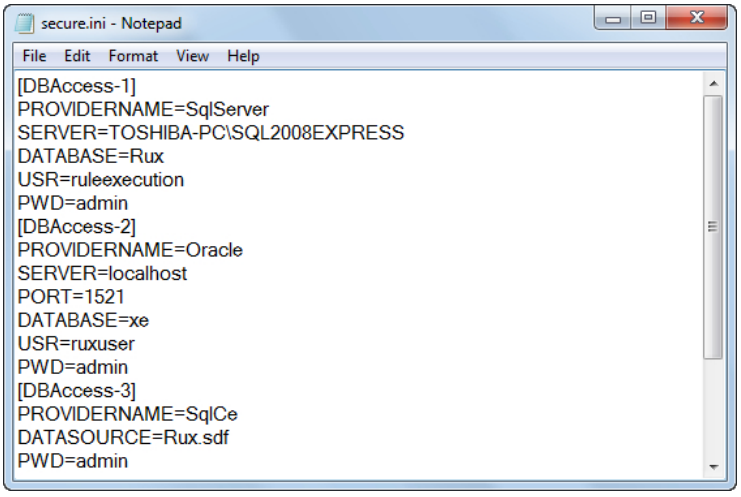
ภาพที่ 4.5 ตัวอย่างการกำหนดข้อมูลคลังเก็บของวินโดวส์แอพพลิเคชัน

2) เว็บแอพพลิเคชัน (Web Application) และเว็บเซอร์วิส (Web Service) กำหนดข้อมูลคลังเก็บในไฟล์ web.config ดังตัวอย่างในภาพที่ 4.6

```
<configuration>
  <appSettings>
    <add key="RuX_Connection" value="DBAccess-3"/>
    <add key="RuX_ConfigTable" value="InternalFunctions"/>
    <add key="Plugin_Path" value=" bin\"/>
    <add key="App_Mode" value="online"/>
    <add key="Source_Path" value="App_Data"/>
  </appSettings>
     .
     .
     .
</configuration>
```

ภาพที่ 4.6 ตัวอย่างการกำหนดข้อมูลคลังเก็บของเว็บแอพพลิเคชันและเว็บเซอร์วิส

การกำหนดข้อมูล เพื่อใช้ในการสร้างการติดต่อกับฐานข้อมูล จะถูกเก็บอยู่ในไฟล์ที่ชื่อว่า secure.ini มีลักษณะข้อมูลดังภาพที่ 4.7



ภาพที่ 4.7 ตัวอย่างไฟล์ secure.ini

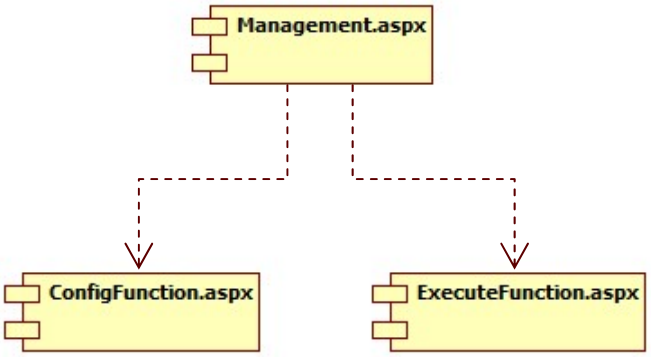
จากภาพที่ 4.7 มีการกำหนดข้อมูล เพื่อนำไปใช้สร้างการติดต่อฐานข้อมูล (Connection String) 3 ฐานข้อมูล ซึ่งระบุในส่วน PROVIDERNAME ค่าที่เป็นไปได้ คือ SqlServer Oracle หรือ SqlCe โดยแต่ละฐานข้อมูลมีการกำหนดข้อมูลที่แตกต่างกัน แสดงในตารางที่ 4.3

ตารางที่ 4.3 ความสัมพันธ์ระหว่างประเภทฐานข้อมูลและการกำหนดข้อมูลในการติดต่อ

		การกำหนดข้อมูล						
		PROVIDERNAME	SERVER	PORT	DATABASE	DATASOURCE	USR	PWD
ประเภทฐานข้อมูล	SqlServer	✓	✓		✓		✓	✓
	Oracle	✓	✓	✓	✓		✓	✓
	SqlCe	✓	✓			✓		✓

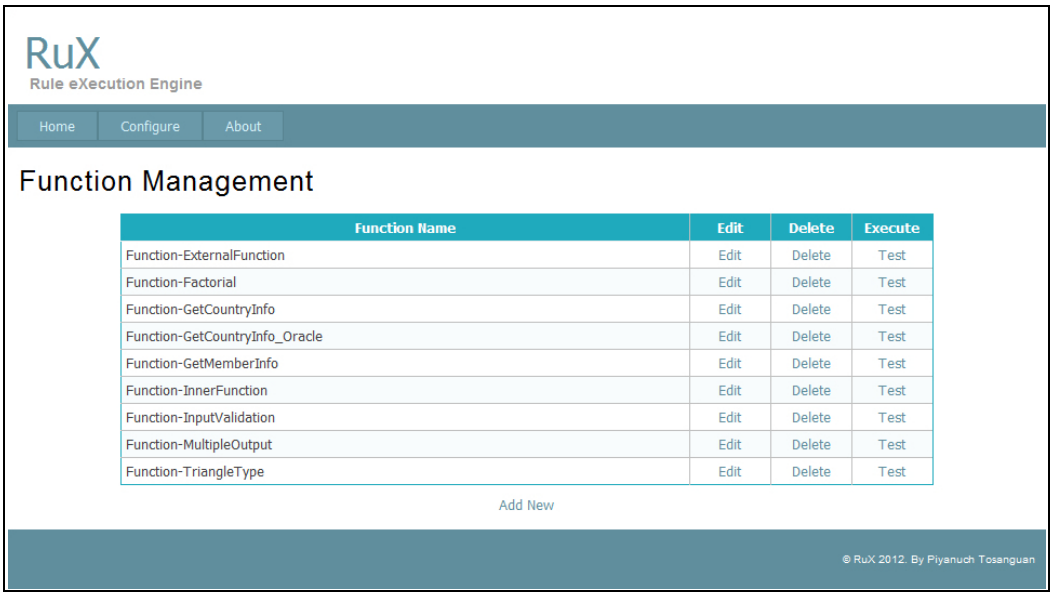
4.4 ส่วนต่อประสานกับผู้ใช้

โครงสร้างการจัดการกฎภายในคลังเก็บ อธิบายโดยใช้แผนภาพองค์ประกอบ (Component Diagram) เพื่อแสดงโครงสร้างและความสัมพันธ์ระหว่างหน้าจอ เป็นดังภาพที่ 4.8



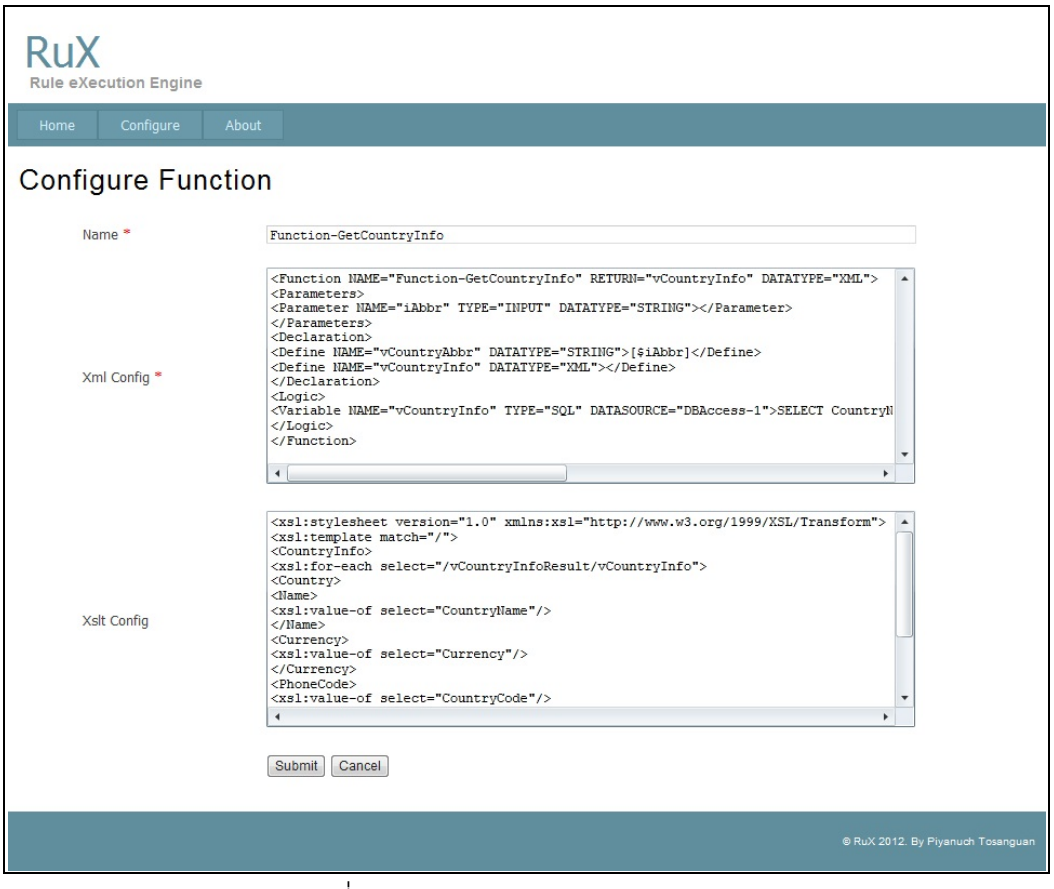
ภาพที่ 4.8 แผนภาพองค์ประกอบการจัดการกฎ

1) หน้าจอ Management.aspx เป็นส่วนการจัดการกฎ เมื่อต้องการเพิ่มเติม แก้ไข หรือลบกฎ ที่อยู่ภายในคลังเก็บที่กำหนด และสามารถเลือกทดสอบกฎได้จากหน้าจอนี้เช่นเดียวกัน ลักษณะหน้าจอเป็นดังภาพที่ 4.9



ภาพที่ 4.9 หน้าจอ Management.aspx

2) หน้าจอ ConfigFunction.aspx ใช้ในการเพิ่มกฎ หรือแก้ไขกฎ มีหน้าจอเป็นดังภาพที่ 4.10



ภาพที่ 4.10 หน้าจอ ConfigFunction.aspx

3) หน้าจอ ExecuteFunction.aspx สำหรับใช้ในการทดสอบกฎ มีหน้าจอเป็นดังภาพที่ 4.11 เมื่อใส่ข้อมูลนำเข้าในช่อง Input และกดปุ่ม Execute ผลลัพธ์จะแสดงในช่อง Output และพารามิเตอร์ทุกตัวทั้งประเภทอินพุต และเอาท์พุต แสดงในตารางช่อง Parameters

ภาพที่ 4.11 หน้าจอ ExecuteFunction.aspx

บทที่ 5
การทดสอบ

วัตถุประสงค์ของการทดสอบเครื่องประมวลผลในบทนี้ ต้องการตรวจสอบว่าเครื่องประมวลผลที่สร้างขึ้นมานั้น สามารถทำงานได้ถูกต้องตามขั้นตอนที่ได้ออกแบบไว้

5.1 สภาพแวดล้อมที่ใช้ในการทดสอบ

การทดสอบเครื่องประมวลผลใช้สภาพแวดล้อมเดียวกัน กับสภาพแวดล้อมที่ใช้ในการพัฒนา ดังที่ได้กล่าวไว้ในบทที่ 4

5.2 แนวทางการทดสอบ

1) ใช้เครื่องมือช่วยทดสอบเอ็นยูนิต (NUnit) เป็นตัวแทนในการทดสอบการทำงานของเครื่องประมวลผล กับวินโดวส์แอพพลิเคชัน

2) สร้างเว็บไซต์สำหรับจัดการกฎในคลังเก็บ เพื่อทดสอบการทำงานของเครื่องประมวลผล กับเว็บแอพพลิเคชัน

3) สร้างตัวอย่างเว็บเซอร์วิสที่มีบริการ ที่ต้องเรียกใช้การประมวลผลกฎ เพื่อทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บเซอร์วิส

4) ทดสอบการเรียกใช้งานกฎที่อยู่ในรูปแบบไฟล์เอกสาร และฐานข้อมูลทั้ง 3 ประเภท

5) ทำการทดสอบในส่วนการประมวลผลชุดคำสั่ง คือ ชุดคำสั่งการดำเนินการตัวแปรในแต่ละประเภท (Variable) ชุดคำสั่งแบบมีเงื่อนไข (IF) และชุดคำสั่งแบบวนซ้ำ (LOOP)

6) ตรวจสอบผลลัพธ์ที่ได้จากการประมวลผลชุดคำสั่งในแต่ละแบบ ว่าเป็นตามที่คาดหวังหรือไม่

7) ทำการทดสอบกฎที่มีการนำชุดคำสั่งต่างๆ มารวมเข้าไว้ด้วยกัน

8) ตรวจสอบผลลัพธ์ของการประมวลผลกฎที่เขียนขึ้น ว่าสามารถทำงานได้ถูกต้องหรือไม่

5.3 กรณีทดสอบ

5.3.1 ชุดคำสั่งการดำเนินการตัวแปรประเภท VALUE และ EXPRESSION

การทดสอบการประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท VALUE และ EXPRESSION ใช้กฎ Function-InputValidation ในภาพที่ 5.1 โดยมีข้อมูลนำเข้าดังภาพที่ 5.2 และผลลัพธ์ที่คาดหวังดังตารางที่ 5.1

```
<Function NAME="Function-InputValidation" RETURN="vString" DATATYPE="STRING">
  <Parameters>
    <Parameter NAME="iStringParam" TYPE="INPUT" DATATYPE="STRING"></Parameter>
    <Parameter NAME="iIntegerParam" TYPE="INPUT" DATATYPE="INTEGER"></Parameter>
    <Parameter NAME="iDecimalParam" TYPE="INPUT" DATATYPE="DECIMAL"></Parameter>
    <Parameter NAME="iBooleanParam" TYPE="INPUT" DATATYPE="BOOLEAN"></Parameter>
    <Parameter NAME="iDateParam" TYPE="INPUT" DATATYPE="DATE"></Parameter>
    <Parameter NAME="iXmlParam" TYPE="INPUT" DATATYPE="XML"></Parameter>
    <Parameter NAME="oStringParam" TYPE="OUTPUT" DATATYPE="STRING">[$vString]</Parameter>
    <Parameter NAME="oIntegerParam" TYPE="OUTPUT" DATATYPE="INTEGER">[$vInteger]</Parameter>
    <Parameter NAME="oDecimalParam" TYPE="OUTPUT" DATATYPE="DECIMAL">[$vDecimal]</Parameter>
    <Parameter NAME="oBooleanParam" TYPE="OUTPUT" DATATYPE="BOOLEAN">[$vBoolean]</Parameter>
    <Parameter NAME="oDateParam" TYPE="OUTPUT" DATATYPE="DATE">[$vDate]</Parameter>
    <Parameter NAME="oXmlParam" TYPE="OUTPUT" DATATYPE="XML">[$vXml]</Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="vStringParam" DATATYPE="STRING">[$iStringParam]</Define>
    <Define NAME="vIntegerParam" DATATYPE="INTEGER">[$iIntegerParam]</Define>
    <Define NAME="vDecimalParam" DATATYPE="DECIMAL">[$iDecimalParam]</Define>
    <Define NAME="vBooleanParam" DATATYPE="BOOLEAN">[$iBooleanParam]</Define>
    <Define NAME="vDateParam" DATATYPE="DATE">[$iDateParam]</Define>
    <Define NAME="vXmlParam" DATATYPE="XML">[$iXmlParam]</Define>
    <Define NAME="vString" DATATYPE="STRING"></Define>
    <Define NAME="vInteger" DATATYPE="INTEGER"></Define>
    <Define NAME="vDecimal" DATATYPE="DECIMAL"></Define>
    <Define NAME="vBoolean" DATATYPE="BOOLEAN"></Define>
    <Define NAME="vDate" DATATYPE="DATE"></Define>
    <Define NAME="vXml" DATATYPE="XML"></Define>
  </Declaration>
  <Logic>
    <Variable NAME="vString" TYPE="VALUE">[$vStringParam]</Variable>
    <Variable NAME="vInteger" TYPE="EXPRESSION">[$vIntegerParam] * 10</Variable>
    <Variable NAME="vDecimal" TYPE="EXPRESSION">[$vDecimalParam] * 10</Variable>
    <Variable NAME="vBoolean" TYPE="EXPRESSION">not [$vBooleanParam]</Variable>
    <Variable NAME="vDate" TYPE="VALUE">[$vDateParam]</Variable>
    <Variable NAME="vXml" TYPE="VALUE">[$vXmlParam]</Variable>
  </Logic>
</Function>
```

ภาพที่ 5.1 กฎ Function-InputValidation

```
<Parameters>
  <Parameter NAME='iStringParam'>AGENCY</Parameter>
  <Parameter NAME='iIntegerParam'>100</Parameter>
  <Parameter NAME='iDecimalParam'>500.75</Parameter>
  <Parameter NAME='iBooleanParam'>False</Parameter>
  <Parameter NAME='iDateParam'>2013-01-12</Parameter>
  <Parameter NAME='iXmlParam'>
    <![CDATA[<Parameters>
      <Parameter>A</Parameter><Parameter></Parameter>
    </Parameters>]]>
  </Parameter>
</Parameters>
```

ภาพที่ 5.2 ข้อมูลนำเข้าในการทดสอบกฎ Function-InputValidation

ตารางที่ 5.1 ผลลัพธ์ที่คาดหวังในการทดสอบกฎ Function-InputValidation

Rule Name	Function-InputValidation	
	Name	**Expected value**
Return	vString	AGENCY
Output Parameter	oStringParam	AGENCY
	oIntegerParam	1000
	oDecimalParam	5007.5
	oBooleanParam	True
	oDateParam	2013-01-12
	oXmlParam	<Parameters> <Parameter>A</Parameter> <Parameter></Parameter> </Parameters>

5.3.2 ชุดคำสั่งการดำเนินการตัวแปรประเภท SQL

การทดสอบการประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท SQL ใช้กฎ Function-GetCountryInfo ในภาพที่ 5.3 และมีการแปลงข้อมูลด้วยเอกซ์เอสแอลที ดังในภาพที่ 5.4 โดยข้อมูลนำเข้าสำหรับการทดสอบนี้ เป็นดังภาพที่ 5.5 และผลลัพธ์ที่คาดหวังดังตารางที่ 5.2

```
<Function NAME="Function-GetCountryInfo" RETURN="vCountryInfo" DATATYPE="XML">
  <Parameters>
    <Parameter NAME="iAbbr" TYPE="INPUT" DATATYPE="STRING"></Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="vCountryAbbr" DATATYPE="STRING">[$iAbbr]</Define>
    <Define NAME="vCountryInfo" DATATYPE="XML"></Define>
  </Declaration>
  <Logic>
    <Variable NAME="vCountryInfo" TYPE="SQL" DATASOURCE="DBAccess-1">
      SELECT CountryName, Currency, CountryCode FROM Countrys WHERE Abbreviation = [$vCountryAbbr]
    </Variable>
  </Logic>
</Function>
```

ภาพที่ 5.3 กฎ Function-GetCountryInfo

```xml
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <CountryInfo>
      <xsl:for-each select="/vCountryInfoResult/vCountryInfo">
        <Country>
          <Name>
            <xsl:value-of select="CountryName"/>
          </Name>
          <Currency>
            <xsl:value-of select="Currency"/>
          </Currency>
          <PhoneCode>
            <xsl:value-of select="CountryCode"/>
          </PhoneCode>
        </Country>
      </xsl:for-each>
    </CountryInfo>
  </xsl:template>
</xsl:stylesheet>
```

ภาพที่ 5.4 เอกซ์เอสแอลทีของกฎ Function-GetCountryInfo

```xml
<Parameters>
  <Parameter NAME='iAbbr'>UK</Parameter>
</Parameters>
```

ภาพที่ 5.5 ข้อมูลนำเข้าในการทดสอบกฎ Function-GetCountryInfo

ตารางที่ 5.2 ผลลัพธ์ที่คาดหวังในการทดสอบกฎ Function-GetCountryInfo

Rule Name	Function-GetCountryInfo	
	Name	Expected value

Return	vCountryInfo	`<vCountryInfoResult>`
		`<vCountryInfo>`
		`<CountryName>United Kingdom</CountryName>`
		`<Currency>GBP<Currency>`
		`<CountryCode>44</CountryCode>`
		`</vCountryInfo>`
		`</vCountryInfoResult>`

After Transform		`<?xml version="1.0" encoding="utf-16"?>`
		`<CountryInfo>`
		`<Country>`
		`<Name>United Kingdom</Name>`
		`<Currency>GBP</Currency>`
		`<PhoneCode>44</PhoneCode>`
		`</Country>`
		`</CountryInfo>`

5.3.3 ชุดคำสั่งการดำเนินการตัวแปรประเภท FUNCTIONCALL และ XPATH

การทดสอบการประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท FUNCTIONCALL และ XPATH ใช้กฎ Function-InnerFunction ในภาพที่ 5.6 ซึ่งภายในมีการเรียกใช้กฎ Function-GetCountryInfo จากข้อ 5.3.2 โดยข้อมูลนำเข้าสำหรับการทดสอบนี้ เป็นดังภาพที่ 5.7 และผลลัพธ์ที่คาดหวังดังตารางที่ 5.3

```
<Function NAME="Function-InnerFunction" RETURN="vName" DATATYPE="STRING">
  <Parameters>
    <Parameter NAME="iAbbr" TYPE="INPUT" DATATYPE="STRING"></Parameter>
    <Parameter NAME="oCurrency" TYPE="OUTPUT" DATATYPE="STRING">[$vCurrency]</Parameter>
    <Parameter NAME="oPhoneCode" TYPE="OUTPUT" DATATYPE="STRING">[$vPhoneCode]</Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="vCountryAbbr" DATATYPE="STRING">[$iAbbr]</Define>
    <Define NAME="vCountryInfo" DATATYPE="XML"></Define>
    <Define NAME="vName" DATATYPE="STRING"></Define>
    <Define NAME="vCurrency" DATATYPE="STRING"></Define>
    <Define NAME="vPhoneCode" DATATYPE="STRING"></Define>
  </Declaration>
  <Logic>
    <Variable NAME="vCountryInfo" TYPE="FUNCTIONCALL" FUNCTIONNAME="Function-GetCountryInfo">
      <Parameters>
        <Parameter NAME="iAbbr">[$vCountryAbbr]</Parameter>
      </Parameters>
    </Variable>
    <Variable NAME="vName" TYPE="XPATH" OBJECT="vCountryInfo">/CountryInfo/Country/Name[1]</Variable>
    <Variable NAME="vCurrency" TYPE="XPATH" OBJECT="vCountryInfo">/CountryInfo/Country/Currency[1]</Variable>
    <Variable NAME="vPhoneCode" TYPE="XPATH" OBJECT="vCountryInfo">/CountryInfo/Country/PhoneCode[1]</Variable>
  </Logic>
</Function>
```

ภาพที่ 5.6 กฎ Function-InnerFunction

```
<Parameters>
  <Parameter NAME='iAbbr'>TH</Parameter>
</Parameters>
```

ภาพที่ 5.7 ข้อมูลนำเข้าในการทดสอบกฎ Function-InnerFunction

ตารางที่ 5.3 ผลลัพธ์ที่คาดหวังในการทดสอบกฎ Function-InnerFunction

Rule Name	Function-InnerFunction	
	Name	**Expected value**
Return	vName	Thailand
Output Parameter	oCurrency	THB
	oPhoneCode	66

5.3.4 ชุดคำสั่งการดำเนินการตัวแปรประเภท FUNCTIONCALLDOTNET

การทดสอบการประมวลผลชุดคำสั่งการดำเนินการตัวแปร ประเภท FUNCTIONCALLDOTNET ใช้กฎ Function-ExternalFunction ในภาพที่ 5.8 ภายในมีการเรียกใช้ดีแอลแอล ExternalLib เพื่อคำนวนแฟกทอเรียล ด้วยการเรียกประมวลผลคลาส Factorial ซึ่งรับอินพุตพารามิเตอร์ที่ชื่อว่า iNumber โดยข้อมูลนำเข้าสำหรับการทดสอบนี้ เป็นดังภาพที่ 5.9 และผลลัพธ์ที่คาดหวังดังตารางที่ 5.4

```
<Function NAME="Function-ExternalFunction" RETURN="vResult" DATATYPE="INTEGER">
  <Parameters>
    <Parameter NAME="iNumber" TYPE="INPUT" DATATYPE="INTEGER"></Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="vNumber" DATATYPE="INTEGER">[$iNumber]</Define>
    <Define NAME="vResult" DATATYPE="INTEGER"></Define>
  </Declaration>
  <Logic>
    <Variable NAME="vResult" TYPE="FUNCTIONCALLDOTNET" NAMESPACE="ExternalLib" OBJECT="Factorial">
      <Parameters>
        <Parameter NAME="iNumber">[$vNumber]</Parameter>
      </Parameters>
    </Variable>
  </Logic>
</Function>
```

ภาพที่ 5.8 กฎ Function-ExternalFunction

```
<Parameters>
  <Parameter NAME='iNumber'>5</Parameter>
</Parameters>
```

ภาพที่ 5.9 ข้อมูลนำเข้าในการทดสอบกฎ Function-ExternalFunction

ตารางที่ 5.4 ผลลัพธ์ที่คาดหวังในการทดสอบกฎ Function-ExternalFunction

Rule Name	Function-ExternalFunction	
	Name	**Expected value**
Return	vResult	120

5.3.5 ชุดคำสั่งแบบมีเงื่อนไขและแบบวนซ้ำ

การทดสอบการประมวลผลชุดคำสั่งแบบมีเงื่อนไข และแบบวนซ้ำ ใช้กฎ Function-Factorial ในภาพที่ 5.10 โดยมีข้อมูลนำเข้าดังภาพที่ 5.11 และผลลัพธ์ที่คาดหวังดังตารางที่ 5.5

```
<Function  NAME="Function-Factorial" RETURN="vResult" DATATYPE="INTEGER">
  <Parameters>
    <Parameter NAME="iNumber" TYPE="INPUT" DATATYPE="INTEGER"></Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="vNumber" DATATYPE="INTEGER">[$iNumber]</Define>
    <Define NAME="vResult" DATATYPE="INTEGER">1</Define>
    <Define NAME="vIndex" DATATYPE="INTEGER">1</Define>
  </Declaration>
  <Logic>
    <IF CONDITION="[$vNumber] &gt; 1">
      <LOOP CONDITION="[$vIndex] &lt;= [$vNumber]">
        <Variable NAME="vResult" TYPE="EXPRESSION">[$vIndex] * [$vResult]</Variable>
        <Variable NAME="vIndex" TYPE="EXPRESSION">[$vIndex] + 1</Variable>
      </LOOP>
    </IF>
  </Logic>
</Function>
```

ภาพที่ 5.10 กฎ Function-Factorial

```
<Parameters>
  <Parameter NAME='iNumber'>5</Parameter>
</Parameters>
```

ภาพที่ 5.11 ข้อมูลนำเข้าในการทดสอบกฎ Function-Factorial

ตารางที่ 5.5 ผลลัพธ์ที่คาดหวังในการทดสอบกฎ Function-Factorial

Rule Name	Function-Factorial	
	Name	**Expected value**
Return	vResult	120

5.3.6 การรวมกันของชุดคำสั่งแบบต่างๆ

การทดสอบกฎที่มีการใช้ชุดคำสั่งที่หลากหลาย ใช้กฎ Function-GetDiscount ในภาพที่ 5.12 ซึ่งเป็นตัวอย่างกฎธุรกิจของร้านขายของ ที่มีเงื่อนไขในการให้ส่วนลดแก่ลูกค้า โดยลูกค้าที่เป็นสมาชิกจะได้รับส่วนลด 10 เปอร์เซ็นต์ แต่ถ้าเป็นวันพุธลูกค้าทุกคนจะได้รับส่วนลด 20 เปอร์เซ็นต์ ทั้งที่เป็นสมาชิก และไม่ได้เป็นสมาชิกกับทางร้าน โดยภายกฎนี้ในมีการเรียกประมวลผลกฎ Function-AccumulatePoint ในภาพที่ 5.13 เพื่อหาคะแนนสะสมของมาชิก ข้อมูลนำเข้าสำหรับการทดสอบนี้ เป็นดังภาพที่ 5.14 และผลลัพธ์ที่คาดหวังดังตารางที่ 5.6

```
<Function NAME="Function-GetDiscount" RETURN="vDiscount" DATATYPE="DECIMAL">
  <Parameters>
    <Parameter NAME="iPhoneNo" TYPE="INPUT" DATATYPE="STRING"></Parameter>
    <Parameter NAME="oPhoneNo" TYPE="OUTPUT" DATATYPE="STRING">[$vPhoneFormat]</Parameter>
    <Parameter NAME="oAccumulatePoint" TYPE="OUTPUT" DATATYPE="INTEGER">[$vPoint]</Parameter>
    <Parameter NAME="oMonth" TYPE="OUTPUT" DATATYPE="INTEGER">[$vMonth]</Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="vPhoneNo" DATATYPE="STRING">[$iPhoneNo]</Define>
    <Define NAME="vPhoneFormat" DATATYPE="STRING"></Define>
    <Define NAME="vMemberData" DATATYPE="XML"></Define>
    <Define NAME="vMember" DATATYPE="INTEGER">0</Define>
    <Define NAME="vDiscount" DATATYPE="DECIMAL">0</Define>
    <Define NAME="vPoint" DATATYPE="INTEGER">0</Define>
    <Define NAME="vMonth" DATATYPE="INTEGER">0</Define>
    <Define NAME="vWeekDay" DATATYPE="INTEGER">0</Define>
  </Declaration>
  <Logic>
    <Variable NAME="vPhoneFormat" TYPE="EXPRESSION">string.format("66{0}",right("[$vPhoneNo]",9))</Variable>
    <Variable NAME="vMemberData" TYPE="SQL" DATASOURCE="DBAccess-1">
      select count(*) as Rec from members where mobileno = [$vPhoneFormat]
    </Variable>
    <Variable NAME="vMember" TYPE="XPATH" OBJECT="vMemberData">/vMemberDataResult/vMemberData/Rec[1]</Variable>
    <IF CONDITION="[$vMember] &gt; 0">
      <Variable NAME="vDiscount" TYPE="VALUE">0.10</Variable>
      <Variable NAME="vPoint" TYPE="FUNCTIONCALL" FUNCTIONNAME="Function-AccumulatePoint">
        <Parameters>
          <Parameter NAME="iPhoneNo">[$vPhoneNo]</Parameter>
        </Parameters>
      </Variable>
      <Variable NAME="vMonth" TYPE="VALUE">[$vPoint.oMonth]</Variable>
    </IF>
    <Variable NAME="vWeekDay" TYPE="EXPRESSION">WeekDay(Now())</Variable>
    <IF CONDITION="[$vWeekDay] = 4">
      <Variable NAME="vDiscount" TYPE="VALUE">0.20</Variable>
    </IF>
  </Logic>
</Function>
```

ภาพที่ 5.12 กฎ Function-GetDiscount

```
<Function NAME="Function-AccumulatePoint" RETURN="vPoint" DATATYPE="INTEGER">
  <Parameters>
    <Parameter NAME="iPhoneNo" TYPE="INPUT" DATATYPE="STRING"></Parameter>
    <Parameter NAME="oMonth" TYPE="OUTPUT" DATATYPE="INTEGER">[$vNum]</Parameter>
  </Parameters>
  <Declaration>
    <Define NAME="vPhoneNo" DATATYPE="STRING">[$iPhoneNo]</Define>
    <Define NAME="vPhoneFormat" DATATYPE="STRING"></Define>
    <Define NAME="vPointData" DATATYPE="XML"></Define>
    <Define NAME="vNumData" DATATYPE="XML"></Define>
    <Define NAME="vNum" DATATYPE="INTEGER">0</Define>
    <Define NAME="vSelectPoint" DATATYPE="INTEGER">0</Define>
    <Define NAME="vIndex" DATATYPE="INTEGER">1</Define>
    <Define NAME="vPoint" DATATYPE="INTEGER">0</Define>
  </Declaration>
  <Logic>
    <Variable NAME="vPhoneFormat" TYPE="EXPRESSION">string.format("66{0}",right("[$vPhoneNo]",9))</Variable>
    <Variable NAME="vPointData" TYPE="SQL" DATASOURCE="DBAccess-1">
      select * from points where mobileno = [$vPhoneFormat]
    </Variable>
    <Variable NAME="vNumData" TYPE="SQL" DATASOURCE="DBAccess-1">
      select count(*) as Rec from points where mobileno = [$vPhoneFormat]
    </Variable>
    <Variable NAME="vNum" TYPE="XPATH" OBJECT="vNumData">/vNumDataResult/vNumData/Rec[1]</Variable>
    <LOOP CONDITION="[$vIndex] &lt;= [$vNum]">
      <Variable NAME="vSelectPoint" TYPE="XPATH" OBJECT="vPointData">
        /vPointDataResult/vPointData[[$vIndex]]/Point[1]
      </Variable>
      <Variable NAME="vPoint" TYPE="EXPRESSION">[$vPoint] + [$vSelectPoint]</Variable>
      <Variable NAME="vIndex" TYPE="EXPRESSION">[$vIndex] + 1</Variable>
    </LOOP>
  </Logic>
</Function>
```

ภาพที่ 5.13 กฎ Function-AccumulatePoint

```
<Parameters>
  <Parameter NAME='iPhoneNo'>0815554848</Parameter>
</Parameters>
```

ภาพที่ 5.14 ข้อมูลนำเข้าในการทดสอบกฎ Function-GetDiscount

ตารางที่ 5.6 ผลลัพธ์ที่คาดหวังในการทดสอบกฎ Function-GetDiscount

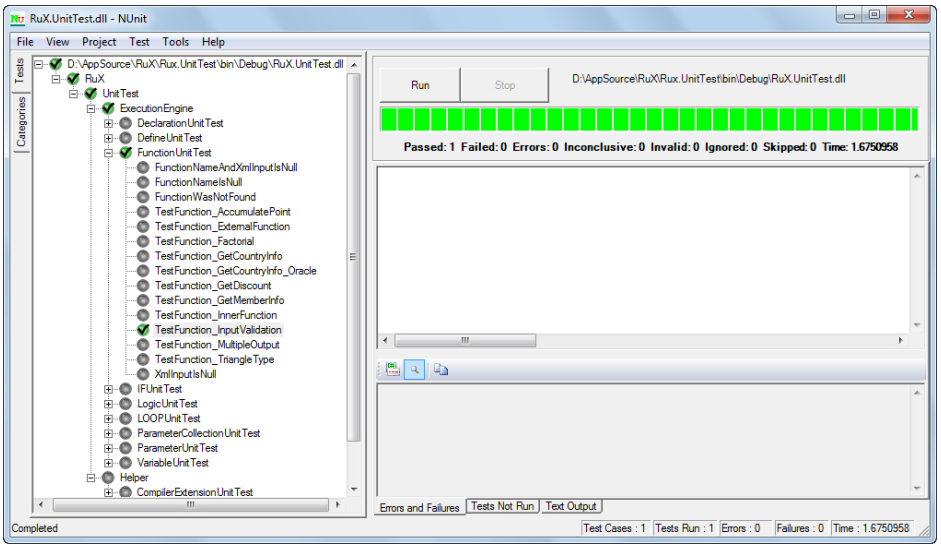
Rule Name	Function-GetDiscount	
	Name	Expected value
Return	vDiscount	If Thursday then 0.20 else 0.10
Output Parameter	oPhoneNo	66815554848
	oAccumulatePoint	26
	oMonth	4

5.4 ผลการทดสอบ

5.4.1 กฎเก็บอยู่ในรูปแบบไฟล์เอกสาร

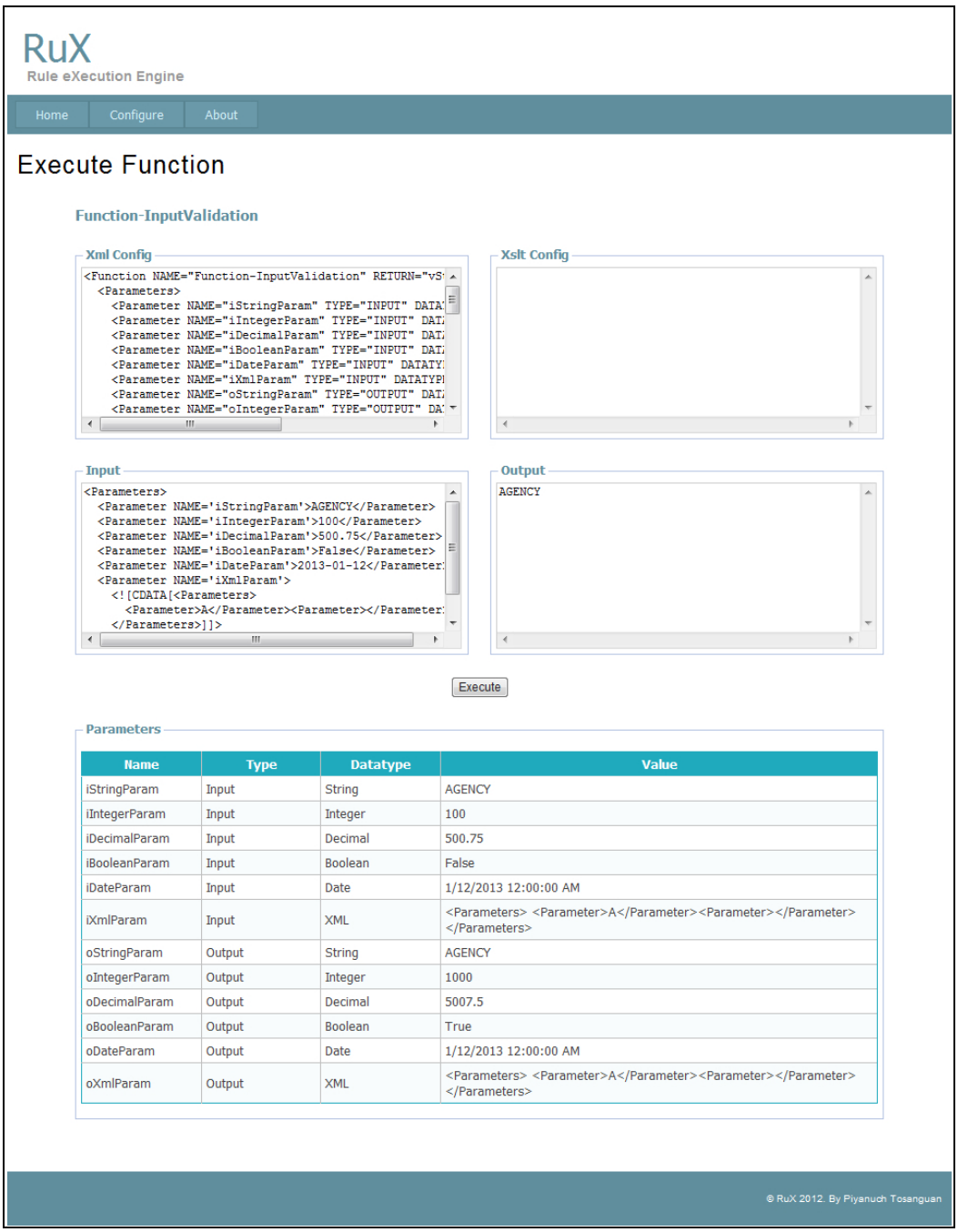
1) Function-InputValidation

1.1) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับวินโดวส์แอพพลิเคชัน ในการประมวลผลกฎ Function-InputValidation พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.15



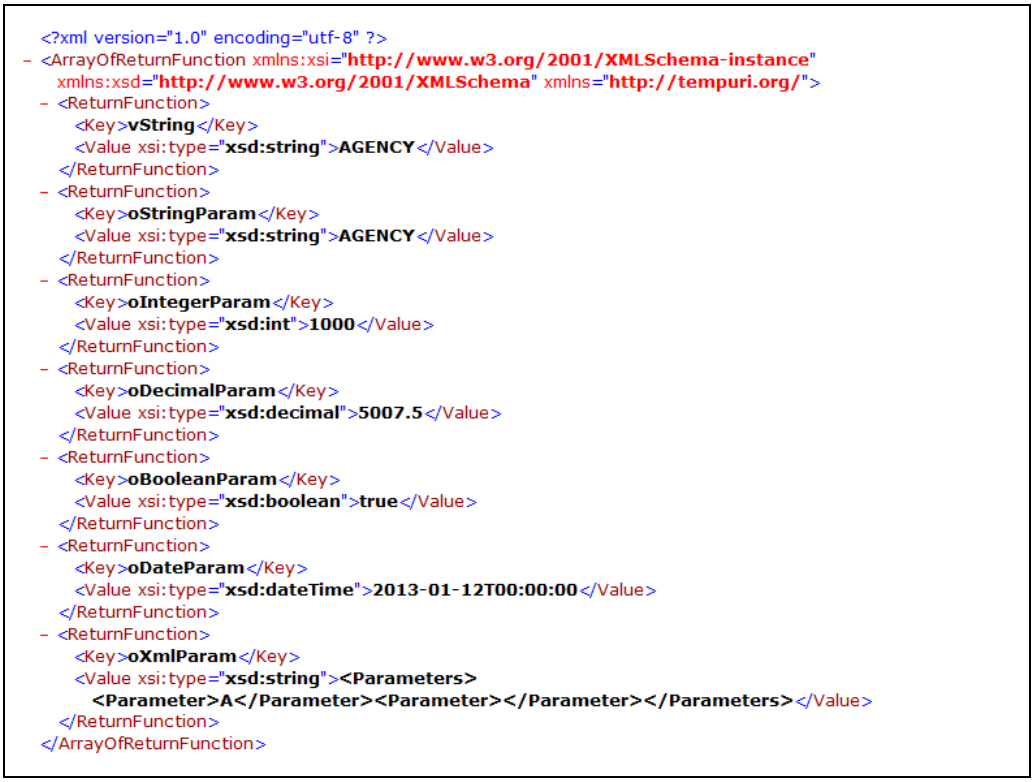
ภาพที่ 5.15 การประมวลผลกฎ Function-InputValidation จากวินโดวส์แอพพลิเคชัน

1.2) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บแอพพลิเคชัน ในการประมวลผลกฎ Function-InputValidation พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.16



ภาพที่ 5.16 การประมวลผลกฎ Function-InputValidation จากเว็บแอพพลิเคชัน

1.3) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บเซอร์วิส ในการประมวลผลกฎ Function-InputValidation พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.17



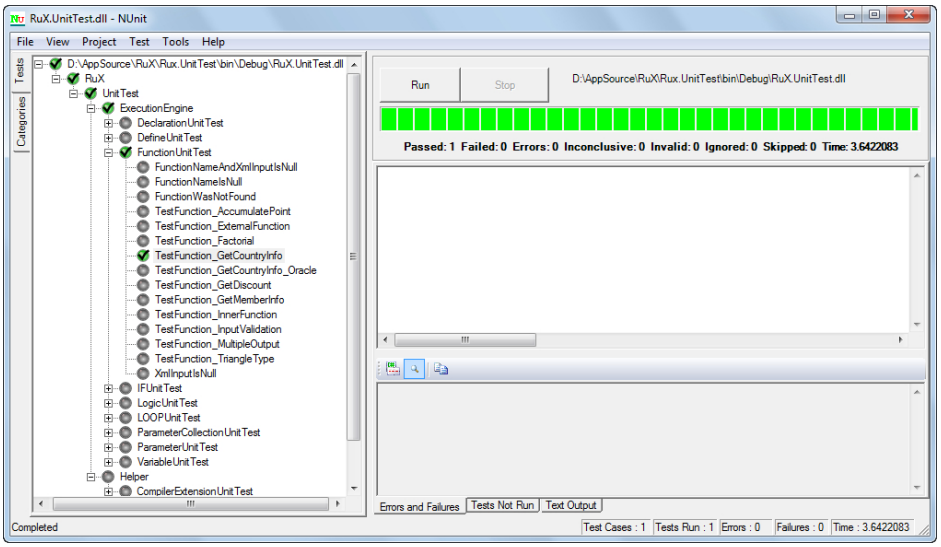
```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfReturnFunction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org/">
  - <ReturnFunction>
      <Key>vString</Key>
      <Value xsi:type="xsd:string">AGENCY</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oStringParam</Key>
      <Value xsi:type="xsd:string">AGENCY</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oIntegerParam</Key>
      <Value xsi:type="xsd:int">1000</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oDecimalParam</Key>
      <Value xsi:type="xsd:decimal">5007.5</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oBooleanParam</Key>
      <Value xsi:type="xsd:boolean">true</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oDateParam</Key>
      <Value xsi:type="xsd:dateTime">2013-01-12T00:00:00</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oXmlParam</Key>
      <Value xsi:type="xsd:string"><Parameters>
        <Parameter>A</Parameter><Parameter></Parameter></Parameters></Value>
    </ReturnFunction>
</ArrayOfReturnFunction>
```

ภาพที่ 5.17 การประมวลผลกฎ Function-InputValidation จากเว็บเซอร์วิส

2) Function-GetCountryInfo

2.1) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับวินโดวส์แอพพลิเคชัน ในการประมวลผลกฎ Function-GetCountryInfo พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.18

2.2) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บแอพพลิเคชัน ในการประมวลผลกฎ Function-GetCountryInfo พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.19



ภาพที่ 5.18 การประมวลผลกฎ Function-GetCountryInfo จากวินโดวส์แอพพลิเคชัน

ภาพที่ 5.19 การประมวลผลกฎ Function-GetCountryInfo จากเว็บแอพพลิเคชัน

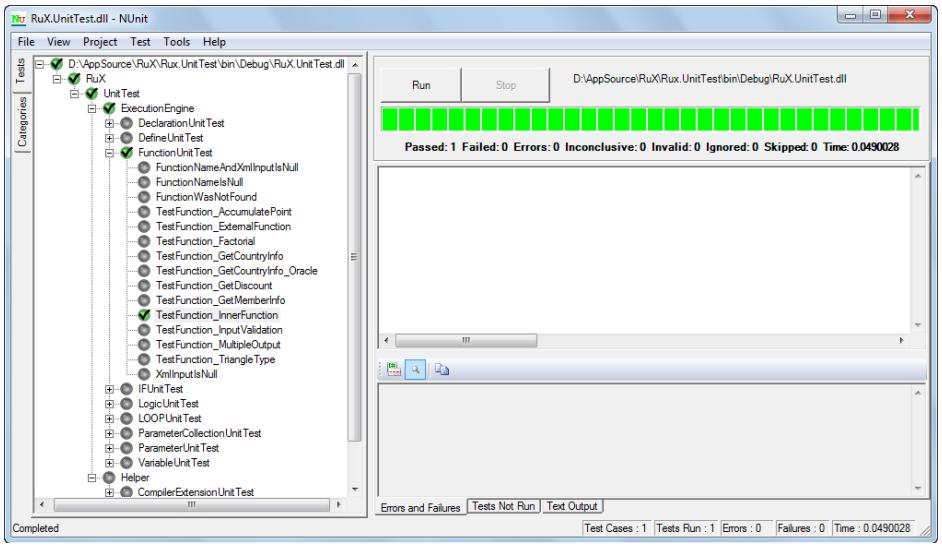
2.3) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บเซอร์วิส ในการประมวลผลกฎ Function-GetCountryInfo พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.20


```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfReturnFunction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org/">
  - <ReturnFunction>
    <Key>vCountryInfo</Key>
    <Value xsi:type="xsd:string"><?xml version="1.0" encoding="utf-16"?
      ><CountryInfo><Country><Name>United
      Kingdom</Name><Currency>GBP</Currency><PhoneCode>44</PhoneCode></Country></CountryInfo></Value>
  </ReturnFunction>
</ArrayOfReturnFunction>
```


ภาพที่ 5.20 การประมวลผลกฎ Function-GetCountryInfo จากเว็บเซอร์วิส

3) Function-InnerFunction

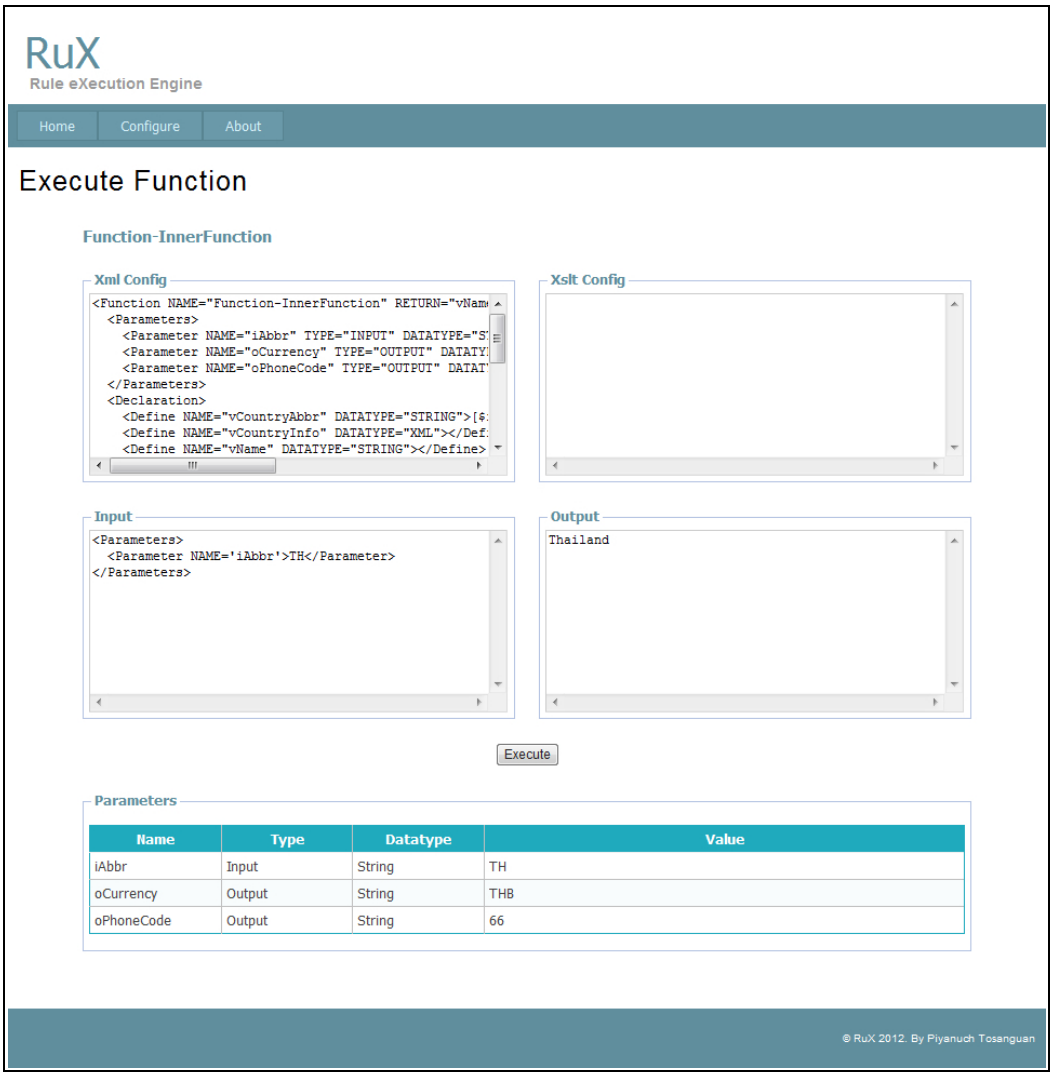
3.1) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับวินโดวส์แอพพลิเคชัน ในการประมวลผลกฎ Function-InnerFunction พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.21



ภาพที่ 5.21 การประมวลผลกฎ Function-InnerFunction จากวินโดวส์แอพพลิเคชัน

3.2) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บแอพพลิเคชัน ในการประมวลผลกฎ Function-InnerFunction พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.22

3.3) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บเซอร์วิส ในการประมวลผลกฎ Function-InnerFunction พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.23

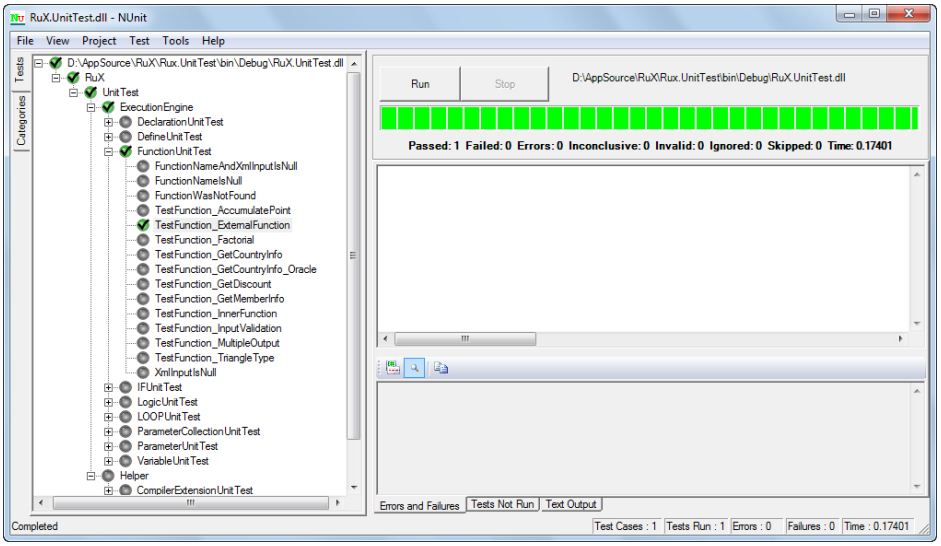


ภาพที่ 5.22 การประมวลผลกฎ Function-InnerFunction จากเว็บแอพพลิเคชัน

ภาพที่ 5.23 การประมวลผลกฎ Function-InnerFunction จากเว็บเซอร์วิส

4) Function-ExternalFunction

4.1) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับวินโดวส์แอพพลิเคชัน ในการประมวลผลกฎ Function-ExternalFunction พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.24



ภาพที่ 5.24 การประมวลผลกฎ Function-ExternalFunction จากวินโดวส์แอพพลิเคชัน

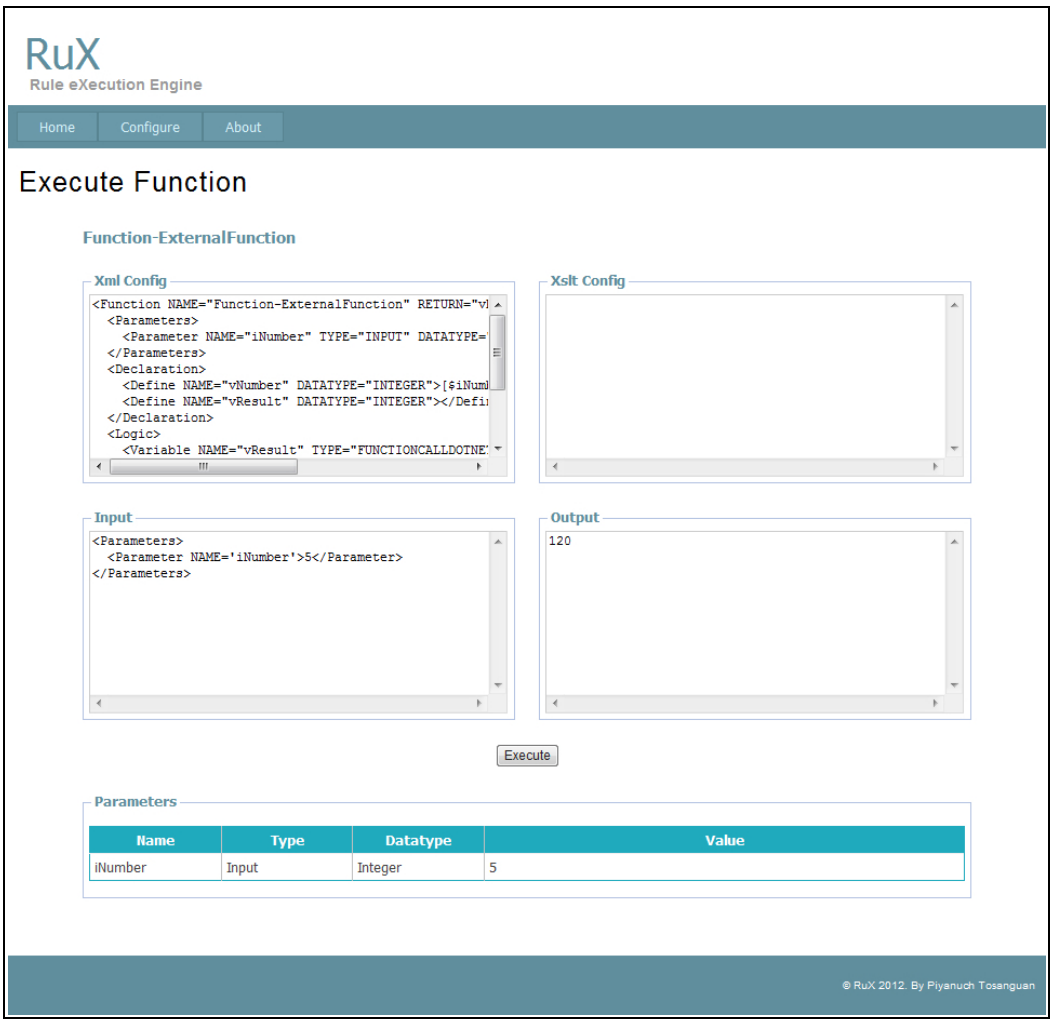
4.2) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บแอพพลิเคชัน ในการประมวลผลกฎ Function-ExternalFunction พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.25

4.3) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บเซอร์วิส ในการประมวลผลกฎ Function-ExternalFunction พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.26

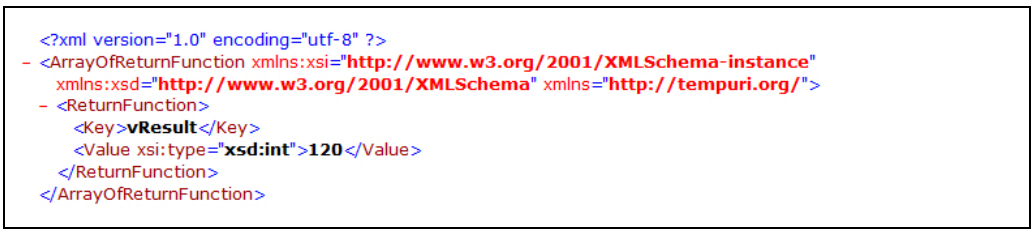
5) Function-Factorial

5.1) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับวินโดวส์แอพพลิเคชัน ในการประมวลผลกฎ Function-Factorial พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.27

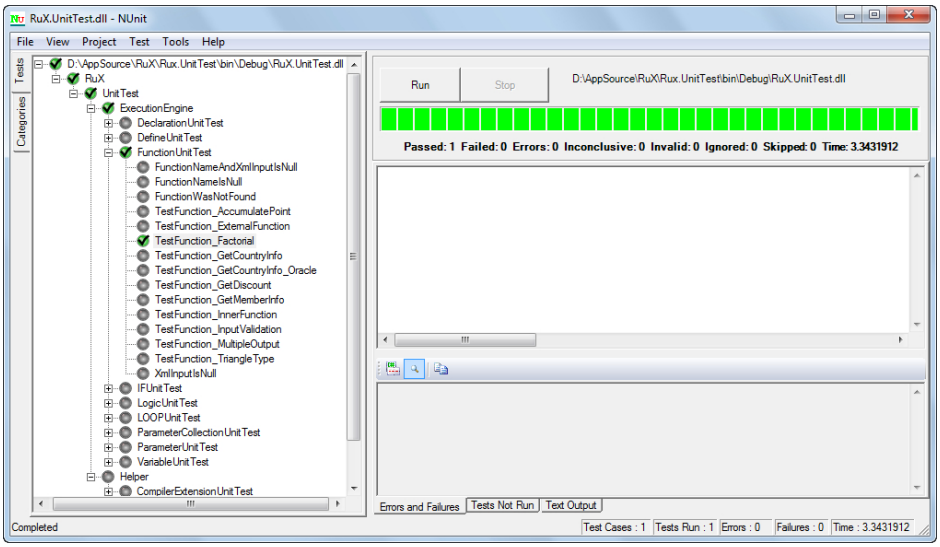
5.2) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บแอพพลิเคชัน ในการประมวลผลกฎ Function-Factorial พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.28



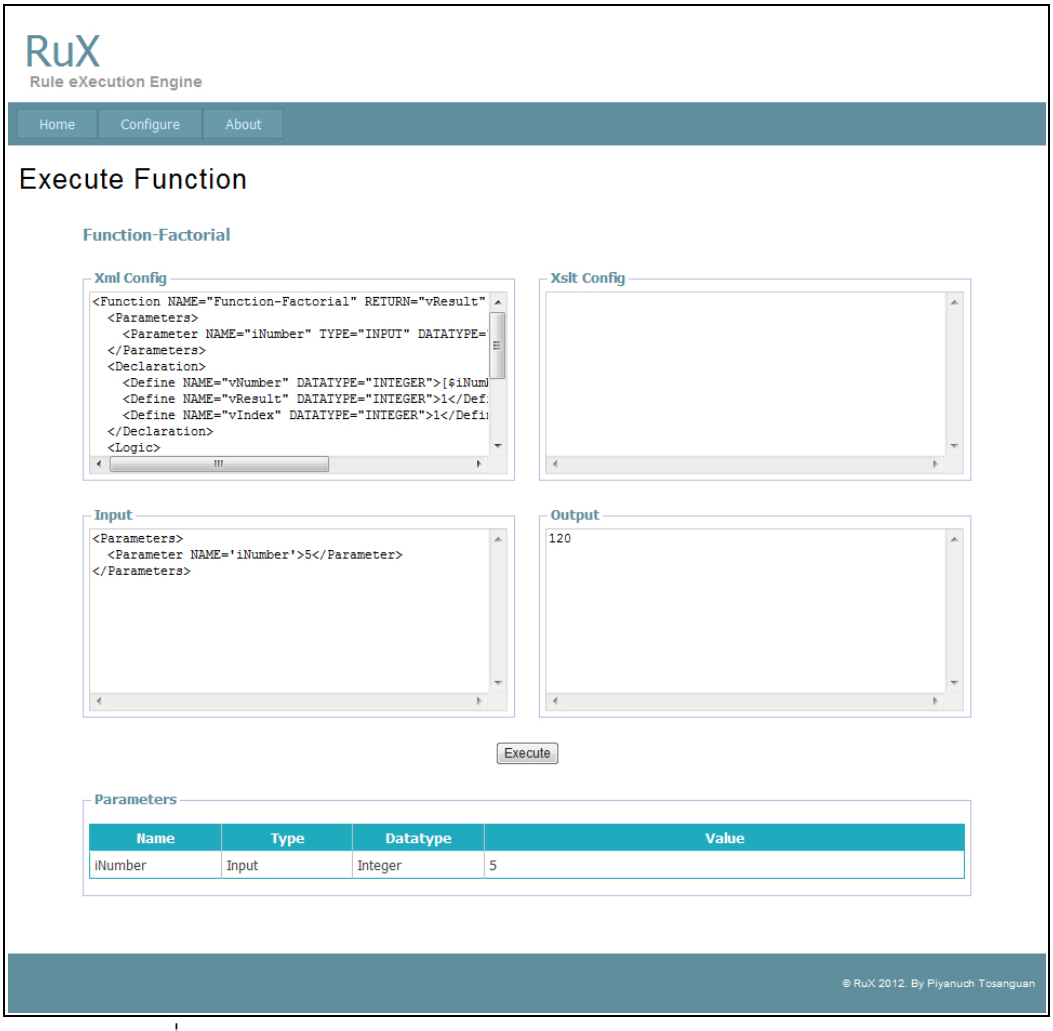
ภาพที่ 5.25 การประมวลผลกฎ Function-ExternalFunction จากเว็บแอพพลิเคชัน



ภาพที่ 5.26 การประมวลผลกฎ Function-ExternalFunction จากเว็บเซอร์วิส



ภาพที่ 5.27 การประมวลผลกฎ Function-Factorial จากวินโดวส์แอพพลิเคชัน



ภาพที่ 5.28 การประมวลผลกฎ Function-Factorial จากเว็บแอพพลิเคชัน

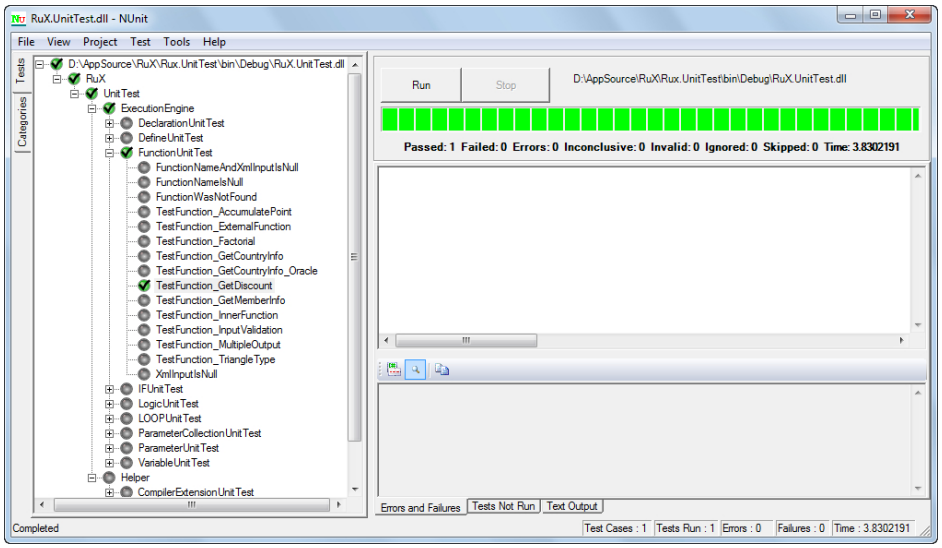
5.3) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บเซอร์วิส ในการประมวลผลกฎ Function-Factorial พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.29

```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfReturnFunction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org/">
  - <ReturnFunction>
      <Key>vResult</Key>
      <Value xsi:type="xsd:int">120</Value>
    </ReturnFunction>
  </ArrayOfReturnFunction>
```

ภาพที่ 5.29 การประมวลผลกฎ Function-Factorial จากเว็บเซอร์วิส

6) Function-GetDiscount

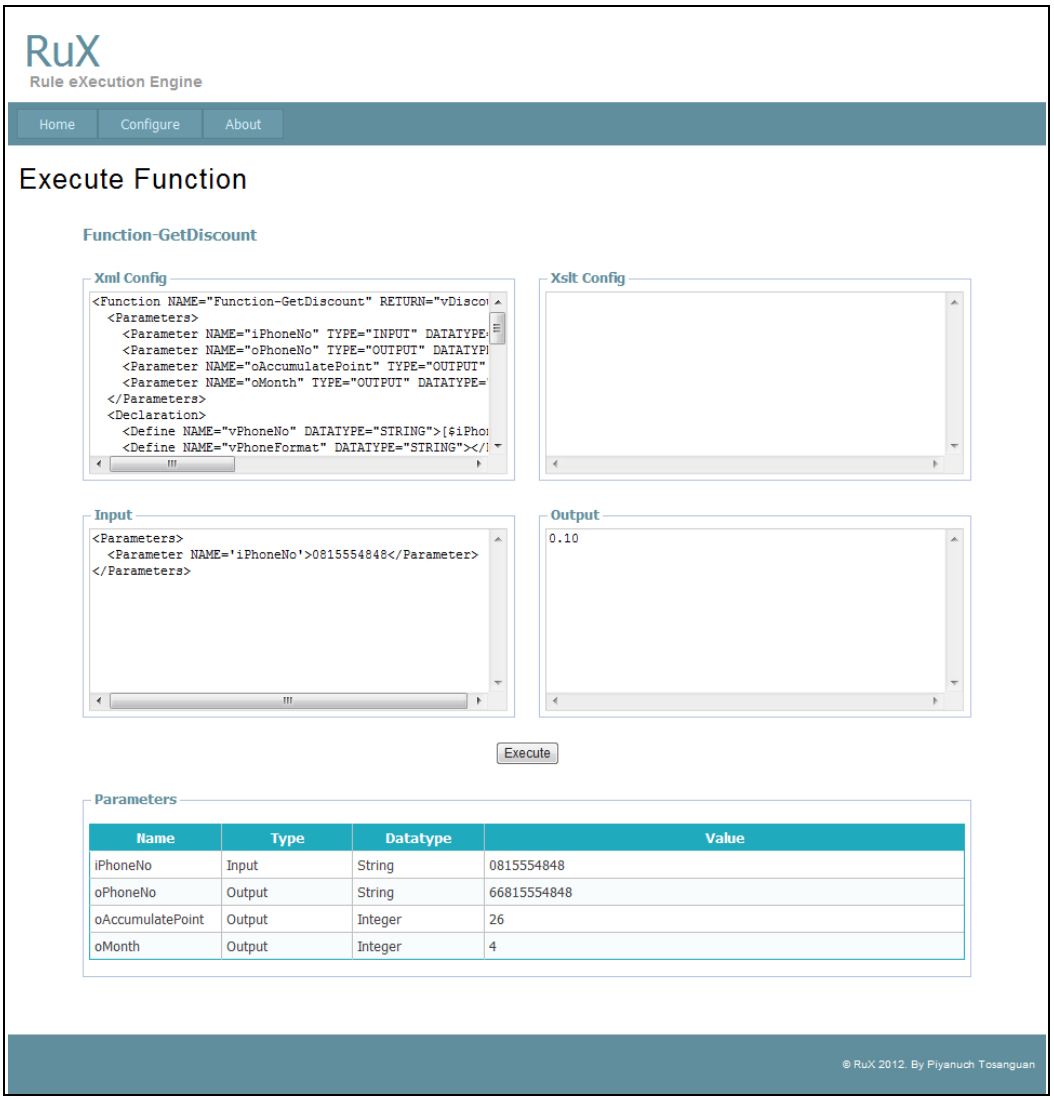
6.1) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับวินโดวส์แอพพลิเคชัน ในการประมวลผลกฎ Function-GetDiscount พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.30



ภาพที่ 5.30 การประมวลผลกฎ Function-GetDiscount จากวินโดวส์แอพพลิเคชัน

6.2) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บแอพพลิเคชัน ในการประมวลผลกฎ Function-GetDiscount พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.31

6.3) ผลการทดสอบการทำงานร่วมกันระหว่างเครื่องประมวลผล กับเว็บเซอร์วิส ในการประมวลผลกฎ Function-GetDiscount พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้อง ได้ผลลัพธ์เป็นไปตามที่คาดหวัง ดังภาพที่ 5.32



ภาพที่ 5.31 การประมวลผลกฎ Function-GetDiscount จากเว็บแอพพลิเคชัน

```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfReturnFunction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org/">
  - <ReturnFunction>
      <Key>vDiscount</Key>
      <Value xsi:type="xsd:decimal">0.10</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oPhoneNo</Key>
      <Value xsi:type="xsd:string">66815554848</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oAccumulatePoint</Key>
      <Value xsi:type="xsd:int">26</Value>
    </ReturnFunction>
  - <ReturnFunction>
      <Key>oMonth</Key>
      <Value xsi:type="xsd:int">4</Value>
    </ReturnFunction>
</ArrayOfReturnFunction>
```

ภาพที่ 5.32 การประมวลผลกฎ Function-GetDiscount จากเว็บเซอร์วิส

5.4.2 กฎเก็บอยู่ในฐานข้อมูลซีควลเซิร์ฟเวอร์

ได้ผลการทดสอบเช่นเดียว กับกฎที่เก็บอยู่ในรูปแบบไฟล์เอกสาร

5.4.3 กฎเก็บอยู่ในฐานข้อมูลออราเคิล

ได้ผลการทดสอบเช่นเดียว กับกฎที่เก็บอยู่ในรูปแบบไฟล์เอกสาร

5.4.4 กฎเก็บอยู่ในฐานข้อมูลเอสคิวแอลซีอี

ได้ผลการทดสอบเช่นเดียว กับกฎที่เก็บอยู่ในรูปแบบไฟล์เอกสาร

5.5 สรุปผลการทดสอบ

จากการทดสอบการทำงานร่วมกันของเครื่องประมวลผล กับโปรแกรมอื่น ในการเรียกใช้งานกฎที่อยู่ในคลังเก็บ ด้วยกรณีทดสอบที่สร้างขึ้นมา พบว่าเครื่องประมวลผลสามารถทำงานได้อย่างถูกต้องตามที่ได้ออกแบบไว้ และกฎที่เรียกใช้งาน สามารถเก็บอยู่ในรูปแบบไฟล์เอกสาร หรือฐานข้อมูล สำหรับผลการทดสอบคุณลักษณะของเครื่องประมวลผล สรุปดังตารางที่ 5.7

ตารางที่ 5.7 สรุปผลการทดสอบคุณลักษณะของเครื่องประมวลผล

คุณลักษณะ	**คำอธิบายเพิ่มเติม**
สถาปัตยกรรมในการใช้งาน	ทำงานร่วมกับโปรแกรม ต่อไปนี้ได้ - โปรแกรมประเภทวินโดวส์แอพพลิเคชัน - โปรแกรมประเภทเว็บแอพพลิเคชัน - โปรแกรมประเภทเว็บเซอร์วิส
การติดต่อฐานข้อมูล	เข้าถึงฐานข้อมูล ต่อไปนี้ได้ - ฐานข้อมูลซีควลเซิร์ฟเวอร์ - ฐานข้อมูลออราเคิล - ฐานข้อมูลเอสคิวแอลซีอี
คำสั่งภาษาโปรแกรม	ใช้งานคำสั่งที่อยู่ภายใต้เนมสเปซ Microsoft.VisualBasic ได้
ใช้งานกฎอื่น	เรียกประมวลผลกฎที่อยู่ภายในคลังเก็บเดียวกันได้

ตารางที่ 5.7 สรุปผลการทดสอบคุณลักษณะของเครื่องประมวลผล (ต่อ)

คุณลักษณะ	**คำอธิบายเพิ่มเติม**
ใช้งานโปรแกรมภายนอก	สั่งประมวลผลไฟล์ดีแอลแอล ที่มีการอิมพลีเมนต์อินเทอร์เฟซตามที่กำหนดได้
ภาษาเอกซ์เอ็มแอล	ใช้งานคำสั่งที่เป็นความสามารถของภาษาเอกซ์เอ็มแอลดังต่อไปนี้ได้ - เอกซ์พาธ ในการเข้าถึงข้อมูลเอกซ์เอ็มแอล - เอกซ์เอสแอลที ในการแปลงข้อมูลเอกซ์เอ็มแอล

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้นำเสนอรูปแบบในการเขียนกฎ ที่มีลักษณะเหมือนกับฟังก์ชันในการเขียนโปรแกรม และทำการพัฒนาเครื่องประมวลผล สำหรับใช้ในการประมวลผลกฎที่เขียนอยู่ในรูปแบบที่นำเสนอ ซึ่งตรรกะของกฎจะถูกเขียนอธิบายด้วยภาษาเอกซ์เอ็มแอล และไวยากรณ์ในการเขียนกฎมีลักษณะที่ใกล้เคียงกับการเขียนโปรแกรม ทำให้ผู้พัฒนาโดยทั่วไปใช้ระยะเวลาในการเรียนรู้ไม่นาน และทำความเข้าใจได้โดยไม่ยากนัก

เครื่องประมวลผลที่พัฒนาขึ้นมานั้น สามารถนำไปใช้ร่วมงานกับโปรแกรมอื่นได้โดยไม่ผูกติดกับสถาปัตยกรรม กล่าวคือ สามารถใช้งานได้กับทั้งโปรแกรมประเภทวินโดวส์แอพพลิเคชัน เว็บแอพพลิเคชัน และเว็บเซอร์วิส นอกจากนี้ยังสามารถเลือกคลังเก็บ สำหรับใช้ในการเก็บข้อมูลกฎ ให้เหมาะสมกับสถาปัตยกรรมของโปรแกรมที่มีความต้องการในการประมวลผลกฎเหล่านี้ได้อีกด้วย เนื่องจากเครื่องประมวลผล สามารถเรียกใช้งานกฎที่อยู่ในรูปแบบของไฟล์เอกสาร และฐานข้อมูลได้ ซึ่งฐานข้อมูลที่สามารถติดต่อได้ คือ ซีควลเซิร์ฟเวอร์ ออราเคิล และเอสคิวแอลซีอี

สำหรับในกรณีที่กฎ มีความซับซ้อนมากเกินกว่าจะสามารถเขียนอธิบายได้ด้วยรูปแบบที่กำหนด หรือในกรณีที่มีโปรแกรมในส่วนการทำงานเดิมอยู่แล้ว และไม่ต้องการแปลงตรรกะทางโปรแกรมเหล่านั้นให้มาอยู่ในรูปแบบที่กำหนด ก็สามารถเลือกใช้วิธีการเขียนโปรแกรมแบบเดิม แล้วทำการอิมพลีเมนต์อินเทอร์เฟซที่เครื่องประมวลผลกำหนด คอมไพล์โปรแกรมให้เป็นไฟล์ดีแอลแอล และเรียกใช้งานผ่านเครื่องประมวลผลได้เช่นเดียวกัน โดยการเขียนกฎเพื่อสั่งประมวลผลไฟล์ดีแอลแอลนั้นแทน

จากลักษณะของเครื่องประมวลผลที่ฝังตัวอยู่ในโปรแกรมอื่น ทำให้โปรแกรมหลายๆ โปรแกรม สามารถเรียกใช้งานกฎเดียวกันได้ ซึ่งเป็นการทำให้เกิดการนำกลับมาใช้ใหม่ และเป็นการเพิ่มความน่าเชื่อถือ เนื่องจากไม่ต้องเขียนตรรกะของกฎไว้ในแต่ละโปรแกรมเอง ซึ่งอาจทำให้เกิดความแตกต่างกันได้

เมื่อสามารถแยกกฎที่อาจมีการเปลี่ยนแปลงบ่อยครั้ง ออกจากตรรกะของโปรแกรมได้ การเปลี่ยนแปลงตรรกะภายในกฎก็สามารถทำได้อย่างอิสระ ไม่กระทบต่อโปรแกรมที่เรียกใช้งาน ดังนั้นจึงทำให้สามารถแก้ไขได้อย่างรวดเร็ว และช่วยลดค่าใช้จ่ายได้

6.2 ข้อจำกัดของงานวิจัย

1) เครื่องประมวลผลสามารถติดต่อกับฐานข้อมูลได้ 3 ประเภทเท่านั้น คือ ซีควลเซิร์ฟเวอร์ ออราเคิล และเอสคิวแอลซีอี

2) คำสั่งภาษาโปรแกรมที่สามารถใช้งานภายในกฎได้ เป็นคำสั่งที่อยู่ภายใต้เนมสเปซ Microsoft.VisualBasic เท่านั้น

3) โปรแกรมที่ต้องการใช้งานเครื่องประมวลผล ต้องพัฒนาด้วยดอทเน็ตเฟรมเวิร์กสี่ขึ้นไปเท่านั้น (ไมโครซอฟท์เทคโนโลยี)

6.3 ข้อเสนอแนะและแนวทางการดำเนินงานต่อ

1) เพิ่มความสามารถในการติดต่อกับฐานข้อมูลอื่นๆ ให้กับเครื่องประมวลผล เช่น ซีควลไลท์ (SQLite) ไมโครซอฟท์แอคเซส (Microsoft Access) เป็นต้น

2) พัฒนาเครื่องประมวลผล ให้รองรับการใช้คำสั่งภาษาโปรแกรมอื่นนอกเหนือจากภาษาวิชวลเบสิก เช่น วิชวลซีชาร์ป (Visual C#) วิชวลซีพลัสพลัส (Visual C++) เป็นต้น

รายการอ้างอิง

[1] Wikipedia, the free encyclopedia. Business rule management system [Online]. Available from : http://en.wikipedia.org/wiki/BRMS [2011,August 20]

[2] Red Hat, Inc. JBoss - Global Leader in Open Source Middleware Software [Online]. Available from : http://www.jboss.com/products/platforms/brms/ [2011,September 24]

[3] IBM. WebSphere ILOG JRules business rule management system (BRMS) [Online]. Available from: http://www-01.ibm.com/software/integration/business-rule-management/jrules-family/ [2011,September 24]

[4] NxBRE [Online]. Available from : http://sourceforge.net/apps/trac/nxbre/wiki [2011,September 3]

[5] Simple Rule Engine - SRE [Online]. Available from : http://simpleruleengine.tripod.com/ [2011,August 14]

[6] ACTL Business Rules Engine [Online]. Available from : http://www.bizruleengine.com/ [2011,August 14]

[7] FLEXRULE [Online]. Available from : http://www.flexrule.com/ [2011,August 14]

[8] Chisholm, M. How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering. 1st ed. Morgan Kaufmann, 2003.

[9] Business Rules Group. Defining Business Rules ~ What Are They Really? [Online]. Available from : http://www.businessrulesgroup.org/first_paper/br01c1.htm#s1e [2011,August 20]

[10] Business Rule Definition - All About Requirements [Online]. Available from : http://www.allaboutrequirements.com/2011/12/business-rule-definition.html [2012,December 30]

[11] W3C Recommendation. Extensible Markup Language (XML) 1.0 (Fifth Edition) [Online]. Available from : http://www.w3.org/TR/xml/ [2011,August 24]

[12] W3Schools. XML Tutorial [Online]. Available from : http://www.w3schools.com/xml/default.asp [2011,August 24]

[13] W3C Recommendation. XML Schema Part 0: Primer Second Edition [Online]. Available from : http://www.w3.org/TR/xmlschema-0/ [2011,August 24]

[14] W3Schools. XML Schema Tutorial [Online]. Available from : http://www.w3schools.com/schema/default.asp [2011,August 241]

[15] W3C Recommendation. XML Path Language (XPath) [Online]. Available from : http://www.w3.org/TR/xpath/ [2011,August 24]

[16] W3Schools. XPath Tutorial [Online]. Available from : http://www.w3schools.com/xpath/ [2011,August 24]

[17] W3C Recommendation. XSL Transformations (XSLT) [Online]. Available from : http://www.w3.org/TR/xslt [2011,August 24]

[18] W3Schools. XSLT Tutorial [Online]. Available from : http://www.w3schools.com/xsl/default.asp [2011,August 24]

[19] Zhang, H., and Yang, X. Rule Engine Research and Implementation in Financial System. In 2009 Fifth International Joint Conference on INC, IMS and IDC, pp. 1114-1117. Seoul, South Korea, 2009.

[20] You, S., Lu, H., and Shixing, L. Research on Rule Definition and Engine for General Text Processing. In 2009 4th International Conference on Computer Science & Education (ICCSE '09), pp. 1095-1100. Nanning, China, 2009.

[21] Oracle. XML Configuration Guide Release 9.4.0.0 [Online]. Available from : http://download.oracle.com/docs/cd/E18894_01/xml_guide/V9_XML_Configuration.htm [2011,September 3]

ภาคผนวก

ภาคผนวก ก
เอกซ์เอสดีของกฎ

เอกซ์เอสดีที่ใช้ในการตรวจสอบโครงสร้างของกฎ เป็นดังภาพ ก.1

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Function" type="Function_Type" />
  <xs:complexType name="Function_Type">
    <xs:sequence>
      <xs:element ref="Parameters" />
      <xs:element ref="Declaration" />
      <xs:element ref="Logic" />
    </xs:sequence>
    <xs:attribute name="NAME" use="required" type="xs:string" />
    <xs:attribute name="RETURN" use="required" type="xs:string" />
    <xs:attribute name="DATATYPE" use="required" type="DataType" />
  </xs:complexType>
  <xs:element name="Parameters" type="Parameters_Type" />
  <xs:complexType name="Parameters_Type">
    <xs:sequence>
      <xs:element ref="Parameter" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Parameter" type="Parameter_Type" />
  <xs:complexType name="Parameter_Type">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="NAME" use="required" type="xs:string" />
        <xs:attribute name="TYPE" type="ParameterType" />
        <xs:attribute name="DATATYPE" type="DataType" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="Declaration" type="Declaration_Type" />
  <xs:complexType name ="Declaration_Type">
    <xs:sequence>
      <xs:element ref="Define" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Define" type="Define_Type" />
  <xs:complexType name="Define_Type">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="NAME" use="required" type="xs:string" />
        <xs:attribute name="DATATYPE" use="required" type="DataType" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="Logic" type="Logic_Type" />
  <xs:complexType name="Logic_Type">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Variable" />
      <xs:element ref="IF" />
      <xs:element ref="LOOP" />
    </xs:choice>
  </xs:complexType>
  <xs:element name="Variable" type="Variable_Type" />
  <xs:complexType name="Variable_Type" mixed="true">
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:element ref="Parameters" />
    </xs:choice>
    <xs:attribute name="NAME" use="required" type="xs:string" />
    <xs:attribute name="TYPE" use="required" type="VariableType" />
    <xs:attribute name="FUNCTIONNAME" type="xs:string" />
    <xs:attribute name="DATASOURCE" type="xs:string" />
    <xs:attribute name="NAMESPACE" type="xs:string" />
    <xs:attribute name="OBJECT" type="xs:string" />
  </xs:complexType>
```

ภาพที่ ก.1 เอกสารเอกซ์เอสดีสำหรับตรวจสอบโครงสร้างกฎ

```
  <xs:element name="IF" type="IF_Type" />
  <xs:complexType name="IF_Type">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Variable" />
      <xs:element ref="IF" />
      <xs:element ref="LOOP" />
    </xs:choice>
    <xs:attribute name="CONDITION" use="required" type="xs:string" />
  </xs:complexType>
  <xs:element name="LOOP" type="LOOP_Type" />
  <xs:complexType name="LOOP_Type">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Variable" />
      <xs:element ref="IF" />
      <xs:element ref="LOOP" />
    </xs:choice>
    <xs:attribute name="CONDITION" use="required" type="xs:string" />
  </xs:complexType>
  <xs:simpleType name="ParameterType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="INPUT" />
      <xs:enumeration value="OUTPUT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="VariableType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="VALUE" />
      <xs:enumeration value="EXPRESSION" />
      <xs:enumeration value="FUNCTIONCALL" />
      <xs:enumeration value="FUNCTIONCALLDOTNET" />
      <xs:enumeration value="SQL" />
      <xs:enumeration value="XPATH" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DataType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="STRING" />
      <xs:enumeration value="INTEGER" />
      <xs:enumeration value="DECIMAL" />
      <xs:enumeration value="BOOLEAN" />
      <xs:enumeration value="DATE" />
      <xs:enumeration value="XML" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

ภาพที่ ก.1 เอกสารเอกซ์เอสดีสำหรับตรวจสอบโครงสร้างกฎ (ต่อ)

ภาคผนวก ข
ข้อมูลที่ใช้ในกรณีทดสอบ

ข.1 โครงสร้างตารางที่ใช้ในกรณีทดสอบ

1) ตาราง Countrys มีโครงสร้างการเก็บข้อมูล ดังภาพที่ ข.1

	Column Name	Data Type	Allow Nulls
🔑	Abbreviation	varchar(5)	☐
	CountryName	varchar(50)	☑
	Currency	varchar(5)	☑
	CountryCode	varchar(5)	☑

ภาพที่ ข.1 โครงสร้างการเก็บข้อมูลของตาราง Countrys

2) ตาราง Members มีโครงสร้างการเก็บข้อมูล ดังภาพที่ ข.2

	Column Name	Data Type	Allow Nulls
🔑	MemberGUID	varchar(36)	☐
	Title	varchar(10)	☑
	FirstName	varchar(50)	☑
	LastName	varchar(50)	☑
	Gender	char(1)	☑
	MobileNo	varchar(15)	☑

ภาพที่ ข.2 โครงสร้างการเก็บข้อมูลของตาราง Members

3) ตาราง Points มีโครงสร้างการเก็บข้อมูล ดังภาพที่ ข.3

	Column Name	Data Type	Allow Nulls
🔑	MobileNo	varchar(15)	☐
🔑	Month	smallint	☐
	Point	int	☐

ภาพที่ ข.3 โครงสร้างการเก็บข้อมูลของตาราง Points

ข.2 ข้อมูลที่ใช้ในกรณีทดสอบ

1) ข้อมูลในตาราง Countrys เป็นดังภาพที่ ข.4

	Abbreviation	CountryName	Currency	CountryCode
1	AE	United Arab Emirates	AED	971
2	AU	Australia	AUD	61
3	BH	Bahrain	BHD	973
4	BN	Brunei	BND	673
5	CA	Canada	CAD	1
6	CH	Switzerland	CHF	41
7	CN	China	CHY	86
8	DK	Denmark	DKK	45
9	HK	Hong Kong	HKD	852
10	ID	Indonesia	IDR	62
11	IN	India	INR	91
12	JP	Japan	JPY	81
13	KR	Korea	KRW	82
14	KW	Kuwait	KWD	965
15	MO	Macau	MOP	853
16	MY	Malaysia	MYR	60
17	NO	Norway	NOK	47
18	NZ	New Zealand	NZD	64
19	OM	Oman	OMR	968
20	PH	Philippines	PHP	63
21	QA	Qatar	QAR	974
22	RU	Russia	RUB	7
23	SA	Saudi Arabia	SAR	966
24	SE	Sweden	SEK	46
25	SG	Singapore	SGD	65
26	TH	Thailand	THB	66
27	TW	Taiwan	TWD	886
28	UK	United Kingdom	GBP	44
29	US	United States	USD	1
30	VN	Vietnam	VND	84
31	ZA	South Africa	ZAR	27

ภาพที่ ข.4 ข้อมูลในตาราง Countrys

2) ข้อมูลในตาราง Members เป็นดังภาพที่ ข.5

	MemberGUID	Title	FirstName	LastName	Gender	MobileNo
1	240D9CB2-499E-43FD-A360-8DD62F9AE1CC	Ms.	Janet	Leverling	F	66835553412
2	37D30E0E-992F-49F9-A37F-EA73FA389CCD	Ms.	Laura	Callahan	F	66865551189
3	3BE5B51C-3624-4217-8C11-E4070237E718	Mrs.	Margaret	Peacock	F	66865558122
4	4A725A25-A37B-44DE-A964-6A351B558504	Mr.	Andrew	Fuller	M	66895559482
5	4F40B54F-0C0B-4798-B056-F1F2BB0E81EE	Mr.	Robert	King	M	66835555598
6	770F9071-ADA7-4110-BE33-783C3967330F	Mr.	Michael	Suyama	M	66895557773
7	AAE021AC-9A43-40EE-BF32-06FF3B95BFFF	Ms.	Anne	Dodsworth	F	66815554444
8	AF4456A5-2D68-4AD5-A213-5E6FEA24EC5A	Mr.	Steven	Buchanan	M	66815554848
9	C87ED2A9-673E-4FBC-8390-B668E861189C	Ms.	Nancy	Davolio	F	66815559857

ภาพที่ ข.5 ข้อมูลในตาราง Members

3) ข้อมูลในตาราง Points เป็นดังภาพที่ ข.6

	MobileNo	Month	Point
1	66815554444	1	8
2	66815554444	2	19
3	66815554444	3	17
4	66815554444	4	8
5	66815554848	1	8
6	66815554848	2	5
7	66815554848	3	7
8	66815554848	4	6
9	66815559857	1	6
10	66815559857	2	20
11	66815559857	3	17
12	66815559857	4	9
13	66835553412	1	5
14	66835553412	2	11
15	66835553412	3	18
16	66835553412	4	2
17	66835555598	1	9
18	66835555598	2	15
19	66835555598	3	17
20	66835555598	4	4
21	66865551189	1	15
22	66865551189	2	8
23	66865551189	3	12
24	66865551189	4	9
25	66865558122	1	10
26	66865558122	2	13
27	66865558122	3	14
28	66865558122	4	0
29	66895557773	1	7
30	66895557773	2	13
31	66895557773	3	16
32	66895557773	4	6
33	66895559482	1	19
34	66895559482	2	6
35	66895559482	3	18
36	66895559482	4	5

ภาพที่ ข.6 ข้อมูลในตาราง Points

ภาคผนวก ค
ตัวอย่างผลลัพธ์การประมวลผล

ค.1 ผลลัพธ์การประมวลผลชุดคำสั่งการดำเนินการตัวแปรประเภท SQL

ผลลัพธ์จากการประมวลผลชุดคำสั่งการดำเนินการตัวแปรประเภท SQL จะอยู่ในรูปแบบของเอกซ์เอ็มแอล ตัวอย่างเช่น ถ้าหากตัวแปรสำหรับการดำเนินการชื่อ ITEM ผลลัพธ์ที่ได้จะมีโครงสร้างดังภาพที่ ค.1

```
<ITEMResult>
  <ITEM>
    <Column1>...</Column1>
    <Column2>...</Column2>
    <Column3>...</Column3>
  </ITEM>
  <ITEM>
    <Column1>...</Column1>
    <Column2>...</Column2>
    <Column3>...</Column3>
  </ITEM>
   .
   .
   .
  <ITEM>
    <Column1>...</Column1>
    <Column2>...</Column2>
    <Column3>...</Column3>
  </ITEM>
</ITEMResult>
```

ภาพที่ ค.1 ตัวอย่างโครงสร้างผลลัพธ์การประมวลผลชุดคำสั่ง
การดำเนินการตัวแปรประเภท SQL

ค.2 ผลลัพธ์การประมวลผลชุดคำสั่งในกฎ Function-AccumulatePoint

การประมวลผลชุดคำสั่งการดำเนินการตัวแปรประเภท SQL ของกฎ Function-AccumulatePoint จากภาพที่ 5.13 (หน้า 59) และมีข้อมูลนำเข้าเป็นดังภาพที่ 5.14 (หน้า 60) ได้ผลลัพธ์ดังนี้

1) การดำเนินการตัวแปร vPointData มีรายละเอียดชุดคำสั่งดังภาพที่ ค.2 และได้ผลลัพธ์ดังภาพที่ ค.3

```
<Variable NAME="vPointData" TYPE="SQL" DATASOURCE="DBAccess-1">
  Select * from points where mobileno = [$vPhoneFormat]
</Variable>
```

ภาพที่ ค.2 ชุดคำสั่งตัวแปร vPointData ของกฎ Function-AccumulatePoint

```
<vPointDataResult>
  <vPointData>
    <MobileNo>66815554848</MobileNo>
    <Month>1</Month>
    <Point>8</Point>
  </vPointData>
  <vPointData>
    <MobileNo>66815554848</MobileNo>
    <Month>2</Month>
    <Point>5</Point>
  </vPointData>
  <vPointData>
    <MobileNo>66815554848</MobileNo>
    <Month>3</Month>
    <Point>7</Point>
  </vPointData>
  <vPointData>
    <MobileNo>66815554848</MobileNo>
    <Month>4</Month>
    <Point>6</Point>
  </vPointData>
</vPointDataResult >
```

ภาพที่ ค.3 ผลลัพธ์การประมวลผลชุดคำสั่งตัวแปร vPointData ของ

กฎ Function-AccumulatePoint

2) การดำเนินการตัวแปร vNumData มีรายละเอียดชุดคำสั่งดังภาพที่ ค.4 และได้ผลลัพธ์ดังภาพที่ ค.5

```
<Variable NAME="vNumData" TYPE="SQL" DATASOURCE="DBAccess-1">
  Select count(*) as Rec from points where mobileno = [$vPhoneFormat]
</Variable>
```

ภาพที่ ค.4 ชุดคำสั่งตัวแปร vNumData ของกฎ Function-AccumulatePoint

```
<vNumDataResult>
  <vNumData>
    <Rec>4</Rec>
  </vNumData>
</vNumDataResult >
```

ภาพที่ ค.5 ผลลัพธ์การประมวลผลชุดคำสั่งตัวแปร vNumData ของ

กฎ Function-AccumulatePoint

ค.3 ผลลัพธ์การประมวลผลกฎ Function-AccumulatePoint

ผลลัพธ์จากการประมวลผลกฎในกรณีที่มีเอาท์พุตพารามิเตอร์ เช่นกฎ Function-AccumulatePoint จะสามารถอ้างอิงถึงตัวแปรเอาท์พุตพารามิเตอร์ได้ ดังการเรียกใช้ในกฎ Function-GetDiscount ภาพที่ 5.12 (หน้า 59) ในการดำเนินการตัวแปร vMonth มีรายละเอียดชุดคำสั่งดังภาพที่ ค.6

```
<Variable NAME="vPoint" TYPE="FUNCTIONCALL"
 FUNCTIONNAME="Function-AccumulatePoint">
  <Parameters>
    <Parameter NAME="iPhoneNo">[$vPhoneNo]</Paramter>
  </Parameters>
</Variable>
<Variable NAME="vMonth" TYPE="VALUE">[$vPoint.oMonth]</Variable>
```

ภาพที่ ค.6 ชุดคำสั่งตัวแปร vMonth ของกฎ Function-GetDiscount

หลังจากเรียกประมวลผลกฎ Function-AccumulatePoint ซึ่งมีเอาท์พุตพารามิเตอร์ชื่อ oMonth สามารถอ้างอิงถึงชื่อผลลัพธ์ที่เป็นเอาท์พุตพารามิเตอร์ได้ด้วย “ชื่อตัวแปรที่เรียกประมวลผลกฎ” + “.” + “ชื่อเอาท์พุตพารามิเตอร์” ในที่นี้คือ vPoint.oMonth

ภาคผนวก ง
การใช้งานเครื่องประมวลผล

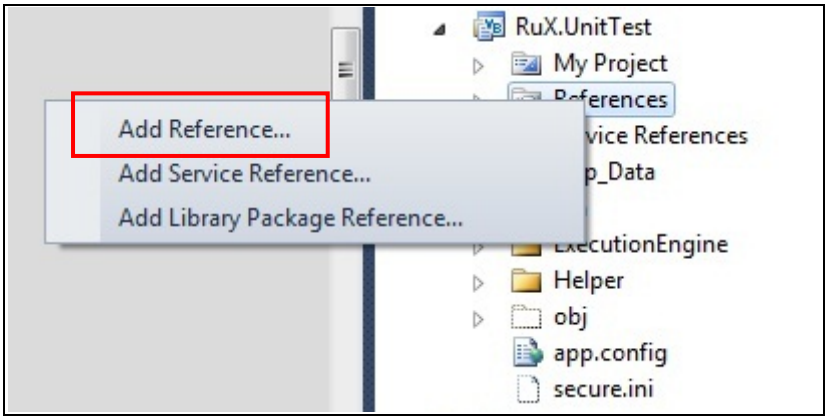
ง.1 ตัวอย่างการนำเครื่องประมวลผลไปใช้งาน

หลังจากที่กำหนดข้อมูลคลังเก็บในไฟล์ config และกำหนดข้อมูลสำหรับใช้ในการติดต่อฐานข้อมูลในไฟล์ secure.ini ตามข้อกำหนดในการใช้งาน (หน้า 47) ขั้นตอนการนำเครื่องประมวลผลไปใช้งานในแอพพลิเคชัน มีดังต่อไปนี้

1) คัดลอกไฟล์ตามรายการด้านล่าง ไปไว้ที่พาธของแอพพลิเคชัน ในตัวอย่างคือ D:\AppSource\RuX\RuX.UnitTest

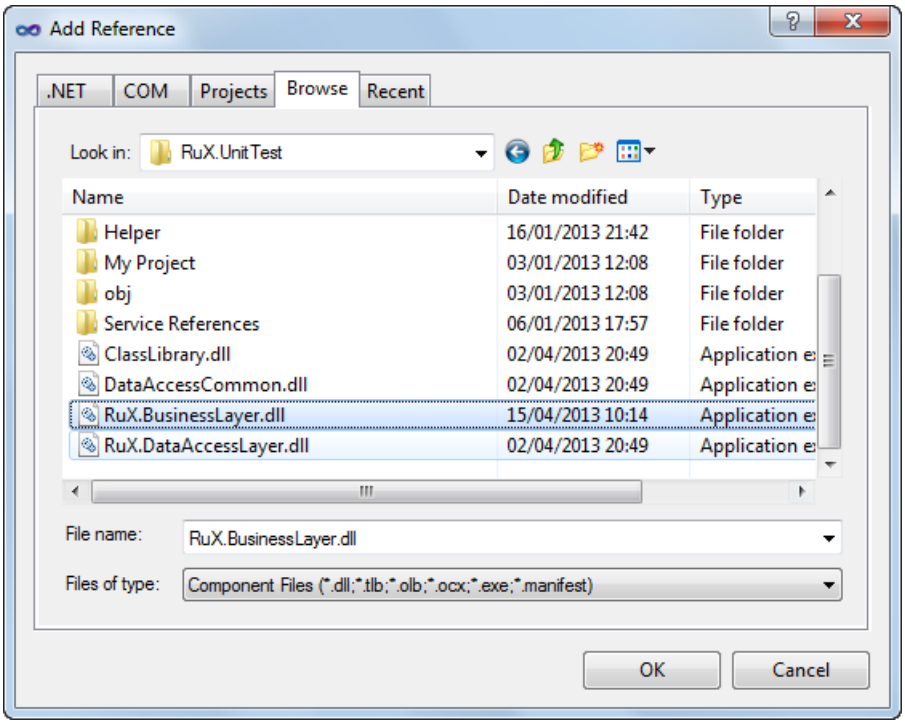
- ClassLibrary.dll
- DataAccessCommon.dll
- RuX.DataAccessLayer.dll
- RuX.BusinessLayer.dll

2) ทำการ Add Reference ที่โปรเจกต์ของแอพพลิเคชัน ดังภาพ ง.1



ภาพที่ ง.1 การ Add Reference

3) เลือก Browse ไปที่พาธในข้อ 1 และเลือกไฟล์ RuX.BusinessLayer.dll ดังภาพที่ ง.2



ภาพที่ ง.2 การ Referenceไฟล์ RuX.BusinessLayer.dll

ง.2 ตัวอย่างการเขียนโปรแกรมในการทำงานร่วมกับเครื่องประมวลผล

ตัวอย่างการเขียนโปรแกรม สำหรับทดสอบกฎ Function-GetDiscount เป็นดังภาพที่ ง.3 อธิบายได้ดังนี้

บรรทัด 2 ประกาศตัวแปรสำหรับข้อมูลนำเข้า และกำหนดค่าข้อมูลนำเข้า ซึ่งอยู่ในรูปแบบของเอกซ์เอ็มแอล

บรรทัด 3 สร้างอินสแตนซ์ของกฎ Function-GetDiscount

บรรทัด 4 สั่งประมวลผลกฎ โดยผลลัพธ์จะถูกเก็บไว้ที่ตัวแปรชื่อ objResult

บรรทัด 5-8 การเข้าถึงแอตทริบิวต์ต่างๆ ของอิลิเมนต์ Function เพื่อตรวจสอบค่าข้อมูล

บรรทัด 9-10 การเข้าถึงอิลิเมนต์ Parameters เพื่อตรวจสอบจำนวน Parameter

บรรทัด 11-26 การเข้าถึงแอตทริบิวต์ต่างๆ ของอิลิเมนต์ Parameter เรียงตามลำดับ เพื่อตรวจสอบค่าข้อมูล

บรรทัด 27-31 ตรวจสอบผลลัพธ์ที่ได้จากการประมวลผลกฎ

```
1  Public Sub TestFunction_GetDiscount()
2    Dim strXmlInput As String = String.Empty
     strXmlInput &= "<Parameters>"
     strXmlInput &= "<Parameter NAME='iPhoneNo'>0815554848</Parameter>"
     strXmlInput &= "</Parameters>"

3    Dim objFunction As New [Function]("Function-GetDiscount", strXmlInput)
4    Dim objResult As Object = objFunction.Execute()

     ' Function
5    Assert.IsNotNull(objFunction)
6    Assert.AreEqual("Function-GetDiscount", objFunction.Name)
7    Assert.AreEqual("vDiscount", objFunction.Return)
8    Assert.AreEqual(DataType.Decimal, objFunction.DataType)
     ' Parameters
9    Assert.IsNotNull(objFunction.Parameters)
10   Assert.AreEqual(4, objFunction.Parameters.Count)
     ' input#1
11   Assert.AreEqual("iPhoneNo", objFunction.Parameters.Item(0).Name)
12   Assert.AreEqual(ParameterType.Input, objFunction.Parameters.Item(0).Type)
13   Assert.AreEqual(DataType.String, objFunction.Parameters.Item(0).DataType)
14   Assert.AreEqual("0815554848", objFunction.Parameters.Item(0).Value)
     ' output#1
15   Assert.AreEqual("oPhoneNo", objFunction.Parameters.Item(1).Name)
16   Assert.AreEqual(ParameterType.Output, objFunction.Parameters.Item(1).Type)
17   Assert.AreEqual(DataType.String, objFunction.Parameters.Item(1).DataType)
18   Assert.AreEqual("66815554848", objFunction.Parameters.Item(1).Value)
     ' output#2
19   Assert.AreEqual("oAccumulatePoint", objFunction.Parameters.Item(2).Name)
20   Assert.AreEqual(ParameterType.Output, objFunction.Parameters.Item(2).Type)
21   Assert.AreEqual(DataType.Integer, objFunction.Parameters.Item(2).DataType)
22   Assert.AreEqual(26, objFunction.Parameters.Item(2).Value)
     ' output#3
23   Assert.AreEqual("oMonth", objFunction.Parameters.Item(3).Name)
24   Assert.AreEqual(ParameterType.Output, objFunction.Parameters.Item(3).Type)
25   Assert.AreEqual(DataType.Integer, objFunction.Parameters.Item(3).DataType)
26   Assert.AreEqual(4, objFunction.Parameters.Item(3).Value)

     ' Return
27   If Weekday(Now) = 4 Then
28     Assert.AreEqual(0.2, objResult)
29   Else
30     Assert.AreEqual(0.1, objResult)
31   End If

32 End Sub
```

ภาพที่ ง.3 ตัวอย่างการเขียนโปรแกรมสำหรับทดสอบกฎ Function-GetDiscount

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวปิยนุช โตสงวน เกิดเมื่อวันที่ 27 มีนาคม พ.ศ. 2527 สำเร็จการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ (เกียรตินิยมอันดับ 2) ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร วิทยาเขตพระราชวังสนามจันทร์ จังหวัดนครปฐม เมื่อปีการศึกษา 2548 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552 ที่อยู่ปัจจุบันที่สามารถติดต่อได้คือ 31/111 หมู่ 4 ต.นาดี อ.เมือง จ.สมุทรสาคร 74000 อีเมล my.pampero@gmail.com

ขณะศึกษาผู้วิจัย ได้เขียนบทความวิชาการในหัวข้อเรื่อง “An Approach for Defining Rules as Functions in Rule-Based Software Development” ซึ่งได้รับการคัดเลือกเพื่อนำเสนอและตีพิมพ์ในงานประชุมวิชาการ “Seventh International Conference on Digital Information Management (ICDIM 2012)” หน้า 30-34 ระหว่างวันที่ 22-24 สิงหาคม พ.ศ. 2555 ณ เขตปกครองพิเศษมาเก๊า สาธารณรัฐประชาชนจีน