ตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี

นายชุมพล โมฆรัตน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

UML COMPONENT DIAGRAM TO ACME COMPILER

Mr. Chumpol Mokarat

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 2013

หัวข้อวิทยานิพนธ์	ตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี
โดย	นายชุมพล โมฆรัตน์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

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

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.ทรงศักดิ์ รองวิริยะพานิช)

ชุมพล โมฆรัตน์ : ตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี. (UML COMPONENT DIAGRAM TO ACME COMPILER) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร.วิวัฒน์ วัฒนาวุฒิ, 105 หน้า.

วิทยานิพนธ์ฉบับนี้นำเสนอเครื่องมือที่เป็นอัตโนมัติ สำหรับการแปลงแผนภาพคอมโพเนนท์ยูเอ็มแอลแทนด้วยภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี ซึ่งเป็นภาษาเชิงรูปนัยสำหรับอธิบายโครงสร้างสถาปัตยกรรม งานวิจัยนี้จึงนำเสนอคำนิยามกฎสำหรับการตรวจสอบคำศัพท์ของแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชันและกฎสำหรับวิเคราะห์โครงสร้างของแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ ซึ่งประกอบด้วยแผนภาพคอมโพเนนท์ แผนภาพคอมโพเนนท์ย่อย คอมโพเนนท์ การเชื่อมต่อ พอร์ตและอินเตอร์เฟส นิยามกฎที่นำเสนอสำหรับตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีได้รับการพัฒนาด้วยเครื่องมือเฟล็กซ์และแย็กค์ ซึ่งตัวแปลภาษานี้ยังทำหน้าที่ตรวจสอบความเข้ากันได้ด้วยไวยากรณ์ไม่พึ่งบริบทเชิงรูปนัยของทั้งสองภาษา

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5470925821 : MAJOR SOFTWARE ENGINEERING
KEYWORDS : UML COMPILER / XMI / ACME / CONTEXT FREE GRAMMAR
CHUMPOL MOKARAT : UML COMPONENT DIAGRAM TO ACME COMPILER. ADVISOR : ASSOC. PROF. WIWAT VATANAWOOD, Ph.D., 105 pp.

This thesis proposes an automatic tool to translate the popular UML component diagram represented by XMI into one of the formal architectural description languages called ACME. This thesis proposes the definition of the rules to verify and analyze the lexical and syntax for the UML component diagram in a regular expression and EBNF metasyntax notations. The definition of such rules can verify notations which consist of the component diagram, sub-component diagram, components, connectors, ports, and interfaces. The definition of the proposed rules applied in a UML component diagram to ACME compiler is implemented using FLEX and YACC tools. In order to ensure the compiler serves the formal conformation of the context free grammars of the two languages.

Department : Computer Engineering Student's Signature .....
Field of Study : Software Engineering Advisor's Signature .....
Academic Year : 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดีจากคำแนะนำของท่าน รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้สละเวลาอันมีค่าเพื่อให้คำปรึกษาและแนวทางในการทำวิจัยที่ดี และคอยชี้แนะวิธีการแก้ปัญหาในแต่ละด้านเพื่อให้เกิดผลสัมฤทธิ์สูงสุดและเป็นประโยชน์ต่อการทำวิจัย และสามารถนำไปถ่ายทอดแก่ส่วนรวมได้โดยสมบูรณ์ ผู้วิจัยจึงขอกราบขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

ขอขอบพระคุณท่าน รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์และผู้ช่วยศาสตราจารย์ ดร.ทรงศักดิ์ รองวิริยะพานิช กรรมการสอบวิทยานิพนธ์ ที่ได้กรุณาเสียสละเวลาอันมีค่ายิ่ง เพื่อให้ข้อแนะนำและแนวทางการดำเนินการ ตลอดจนความรู้และแง่คิดที่เป็นประโยชน์ต่อการทำวิจัย ซึ่งทำให้วิทยานิพนธ์สำเร็จลุล่วงไปด้วยดี

ขอขอบพระคุณคณาจารย์ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่คอยอบรม สั่งสอนและประสิทธิ์ประสาทวิชาความรู้ ตลอดจนถ่ายทอดประสบการณ์อันมีค่า

ขอขอบคุณเพื่อนๆ ห้องปฏิบัติการวิศวกรรมซอฟต์แวร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้ความอนุเคราะห์ในทุกๆ ด้านตลอดจนเจ้าหน้าที่และบุคลากรภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่คอยนำแนะและให้ความช่วยเหลืออย่างดีเสมอมา

สุดท้ายขอกราบขอบพระคุณบิดา มารดาและครอบครัว ที่คอยให้กำลังใจ และแนวคิดสำหรับการดำรงชีวิตที่ดี และสนับสนุนทั้งเรื่องงานและการศึกษาด้วยดีเสมอมา

สารบัญ

หน้า

หน้า

สารบัญตาราง

สารบัญภาพ

หน้า

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

เนื่องจากการพัฒนาโครงการด้านวิศวกรรมซอฟต์แวร์ หนึ่งในขั้นตอนที่สำคัญต่อการพัฒนาโครงการคือขั้นตอนการวิเคราะห์ และออกแบบซอฟต์แวร์ ซึ่งถือว่าเป็นกิจกรรมที่สำคัญต่อวัฏจักรการพัฒนาซอฟต์แวร์ อีกทั้งนักวิเคราะห์และออกแบบซอฟต์แวร์ยังต้องใช้เครื่องมือสำหรับนำเสนอผลการดำเนินงานในขั้นตอนดังกล่าว ประกอบกับปัจจุบันมีเครื่องมือหลากหลาย ที่ซึ่งใช้สำหรับการอธิบายข้อมูลเชิงโครงสร้างและเชิงกายภาพด้วยภาษายูเอ็มแอล (Unified Modeling Language: UML) [1] จากเครื่องมือต่างๆ เช่น ArgoUML [2], Rational Rose [3], Visual Paradigm [4] เป็นต้น เพื่อนำเสนอลักษณะองค์ประกอบของซอฟต์แวร์แล้วนั้น ด้วยเครื่องมือที่หลากหลาย จึงมีการสร้างมาตรฐานสำหรับเป็นการแลกเปลี่ยนข้อมูล (Interchange Format) ระหว่างเครื่องมือดังกล่าวและภาษาเอกซ์เอ็มไอ (XML Metadata Interchange: XMI) [5] ถือได้ว่าเป็นมาตรฐานสำหรับการแลกเปลี่ยนข้อมูลระหว่างเครื่องมือทางภาษายูเอ็มแอล ซึ่งเป็นมาตรฐานที่กำหนดโดยองค์กรบริหารงานเชิงวัตถุ (Object Management Group: OMG) [5] โดยมุ่งเน้นไปที่ประสิทธิภาพของแผนภาพยูเอ็มแอลในรูปแบบการประมวลผลด้วยภาษาเอกซ์เอ็มไอ อีกทั้งยังเป็นการเพิ่มความสามารถของการทำงานร่วมกันระหว่างเครื่องมือที่อธิบายด้วยภาษายูเอ็มแอล

แต่ด้วยข้อจำกัดของภาษาเอกซ์เอ็มไอ ที่ต้องนิยามให้สอดคล้องกับรายละเอียดในเมทาดาต้า (Metadata) และทุกๆ องค์ประกอบเอกซ์เอ็มแอล (XML element) ที่ถูกกำหนดด้วยข้อกำหนดในเนมสเปซ (Namespace) [5] และจากการกำหนดนิยามดังกล่าวอาจมีความซับซ้อน ซึ่งอาจส่งผลให้ทำความเข้าใจได้ยากด้วยเหตุผลดังกล่าว David Galan, Robert Monroe และ David Wile [6] จึงนำเสนอแนวคิดเพื่อตอบสนองความต้องการที่หลากหลาย โดยพยายามเชื่อมโยงกลุ่มเป้าหมายและผู้มีส่วนเกี่ยวข้อง ด้วยการจัดเตรียมกรอบโครงสร้างทางสถาปัตยกรรม สำหรับอำนวยความสะดวกในการระบุข้อมูลหรือกำหนดรายละเอียดเพิ่มเติมให้กับภาษาเอดีแอล (Architecture Description Language: ADL) [7] เรียกว่า ภาษาแอคมี (ACME Language) [6] ซึ่งเป็นภาษาที่พัฒนาจากความพยายามร่วมกันของกลุ่มผู้วิจัยทางด้านสถาปัตยกรรมซอฟต์แวร์ เพื่อเป็นรูปแบบการแลกเปลี่ยนสำหรับเครื่องมือการออกแบบสถาปัตยกรรม ที่ซึ่งตรวจสอบความถูกต้องและความสอดคล้องกันทางไวยากรณ์ในกลุ่มภาษาเอดีแอลและสามารถอธิบายแผนภาพคอมโพเนนท์รวมถึงองค์ประกอบ (Element) พื้นฐาน อีกทั้งภาษาแอคมีมีเป้าหมายในการทำหน้าที่เป็นพาหนะสำหรับเป็นแบบตั้งต้นเกี่ยวกับมาตรฐานสำหรับข้อมูลทางด้านสถาปัตยกรรม ซึ่งมีกลไกคุณลักษณะ (Property) ที่สนับสนุน

เกี่ยวกับการอนุญาตให้ใช้เป็นกลุ่มภาษาย่อยสำหรับคุณลักษณะดังกล่าวด้วยประเภทและค่าของคุณลักษณะ [6] และศึกษาทำความเข้าใจได้ง่ายกว่า

ดังนั้น ผู้วิจัยจึงมีแนวคิดในการนำภาษาแอคมีเข้ามาช่วยอธิบายแผนภาพคอมโพเนนท์ ด้วยลักษณะของกลไกที่มีความยืดหยุ่นและสนับสนุนการทำงานร่วมกับเครื่องหมายที่สัมพันธ์กันและไม่ซับซ้อน ทั้งนี้ในปัจจุบันยังมีภาษาเอกซ์เอ็มแอล (Extensible Markup Language: XML) [8] ซึ่งเป็นอีกภาษาที่นิยมใช้สำหรับการแลกเปลี่ยนข้อมูล แต่ภาษาแอคมีก็สามารถเป็นอีกทางเลือกสำหรับใช้ในการศึกษาและวิจัย ที่ซึ่งนำเสนอเพื่อเป็นทางเลือกใหม่ในการอธิบายโครงสร้างแผนภาพคอมโพเนนท์แทนที่จะใช้เฉพาะภาษาเอกซ์เอ็มไออย่างเดียว ด้วยการผนวกวิธีการทำงานของตัวแปลภาษา (Compiler) [13] ซึ่งใช้วิธีการตรวจสอบความถูกต้องและความสอดคล้องกันของโครงสร้างไวยากรณ์ในภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี อีกทั้งเป็นสื่อกลางการแลกเปลี่ยนข้อมูลระหว่างภาษายูเอ็มแอลไปยังภาษาเอดีแอลด้วยเครื่องมือที่นำเสนอ ซึ่งเรียกว่า ตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี

1.2 วัตถุประสงค์ของการวิจัย

เพื่อออกแบบและพัฒนาตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี

1.3 ขอบเขตของงานวิจัย

1. ข้อมูลนำเข้าต้องเป็นภาษาเอกซ์เอ็มไอ รุ่นที่ 2.1 ที่ได้จากแผนภาพคอมโพเนนท์ยูเอ็มแอล รุ่นที่ 2.0
2. ผลลัพธ์ของแผนภาพคอมโพเนนท์ที่ได้จากตัวแปลภาษาอยู่ในรูปแบบภาษาแอคมี รุ่นที่ 2.0
3. การออกแบบและพัฒนาเครื่องมือตัวแปลภาษา ได้นำข้อกำหนด นิยามและเครื่องมือที่เกี่ยวข้องเพื่อพัฒนาเครื่องมือตัวแปลภาษา ดังนี้
    1) นำข้อกำหนดการสร้างแบบจำลองสถาปัตยกรรมซอฟต์แวร์ในภาษายูเอ็มแอล [24, 26] เป็นกฎตั้งต้นสำหรับตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ที่ซึ่งออกแบบไว้สำหรับแสดงสถาปัตยกรรมซอฟต์แวร์
    2) นำเครื่องมือโอเพ่นซอร์สเจเฟล็กซ์ (JFlex) ใช้งานในกระบวนการวิเคราะห์คำศัพท์ ซึ่งจะตรวจสอบและวิเคราะห์คำศัพท์ตามไวยากรณ์แผนภาพคอมโพเนนท์ในรูปแบบเรกูลาร์เอกซ์เพรสชัน (Regular Expression) ที่เป็นข้อมูลตั้งต้นสำหรับการวิเคราะห์คำศัพท์
    3) นำเครื่องมือแย็กค์ (Yacc) ใช้งานในกระบวนการวิเคราะห์ไวยากรณ์ กำหนดลักษณะทางไวยากรณ์ตามนิยามเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ (EBNF) ของแผนภาพคอมโพเนนท์เป็นข้อมูลนำเข้าตั้งต้น สำหรับการวิเคราะห์ไวยากรณ์

4) นำนิยามภาษาแอคมี รุ่นที่ 2.0 ซึ่งเป็นเมทาดาต้าวากยสัมพันธ์บีเอ็นเอฟ (BNF) สำหรับอามานิพาสเซอร์ (ArmaniParser) เป็นกฎตั้งต้นสำหรับขั้นตอนการปรับรวมโครงสร้างทางไวยากรณ์ของภาษาเอกซ์เอ็มไอไปเป็นภาษาแอคมี

4. เครื่องมือตัวแปลภาษาที่ออกแบบและพัฒนา มีคุณสมบัติ ดังนี้

1) สามารถตรวจสอบเครื่องหมาย (Notation) แผนภาพคอมโพเนนท์ยูเอ็มแอล ดังนี้ คอมโพเนนท์ (Component), การเชื่อมต่อ (Connector) ซึ่งประกอบด้วยการเชื่อมต่อ ดังนี้ แอสโซซิเอชัน (Association) เรียลไลเซชัน (Realization) ยูซเสจ (Usage) และดีเพนเดนซี (Dependency) รวมทั้งอินเตอร์เฟส (Interface) และพอร์ต (Port) ได้

2) สามารถตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ที่ซึ่งออกแบบไว้สำหรับแสดงสถาปัตยกรรม

3) สามารถตรวจสอบสถาปัตยกรรมย่อย (Sub Architecture) ได้หนึ่งระดับเป็นอย่างน้อย

4) สามารถตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ได้ออกแบบถูกต้องตามโครงสร้างไวยากรณ์ (Syntax) ยูเอ็มแอล รุ่นที่ 2.0 ได้เป็นอย่างน้อย

5. ใช้แผนภาพคอมโพเนนท์จำนวนสามระบบเป็นอย่างน้อย สำหรับการตรวจสอบและประเมินผลเครื่องมือตัวแปลภาษา และนำผลลัพธ์ไปแสดงผลบนเครื่องมือแอคมีสตูดิโอ (AcmeStudio)

1.4 ขั้นตอนและวิธีการดำเนินการวิจัย

วิทยานิพนธ์ฉบับนี้แบ่งเนื้อหาเป็นหกบท ดังนี้ บทที่ 1 บทนำ กล่าวถึงที่มาและความสำคัญของปัญหา รวมถึงวัตถุประสงค์ของการวิจัย ขอบเขตของงานวิจัยและบทความวิชาการที่ได้รับการตีพิมพ์ บทที่ 2 กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 นำเสนอการวิเคราะห์และนิยามกฎสำหรับตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล บทที่ 4 การออกแบบและการพัฒนาเครื่องมือตัวแปลภาษา บทที่ 5 กล่าวถึงการทดสอบเครื่องมือตัวแปลภาษา และบทที่ 6 บทสรุปผลการวิจัยและข้อเสนอแนะสำหรับการต่อยอดงานวิจัยในอนาคตและอภิปรายงานวิจัยเพิ่มเติม

1.5 บทความวิจัยที่ได้รับการตีพิมพ์

บทความวิจัยที่ได้รับการตีพิมพ์ เป็นบทความวิจัยระดับนานาชาติ เรื่อง “UML COMPONENT DIAGRAM TO ACME COMPILER” โดยมีผู้เขียนดังนี้ นายชุมพล โมฆรัตน์ (Chumpol Mokarat) และ รศ. ดร.วิวัฒน์ วัฒนาวุฒิ (Wiwat Vatanawood) ซึ่งได้นำเสนอและตีพิมพ์ในงาน "The 4th International Conference on Information Science and Applications (ICISA 2013)" ระหว่างวันที่ 24-26 มิถุนายน พ.ศ. 2556 ณ โรงแรมฮิลตันพัทยา (Hilton Pattaya Hotel) เมืองพัทยา ตำบลหนองปรือ อำเภอบางละมุง จังหวัดชลบุรี ประเทศไทย

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ในทฤษฎีที่เกี่ยวข้องกับงานวิจัยฉบับนี้ ประกอบด้วยแผนภาพคอมโพเนนท์ ภาษาเอกซ์เอ็มไอ ตัวแปลภาษา เครื่องมือเฟล็กซ์และแย็กค์ ภาษาแอคมีและเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ ซึ่งมีรายละเอียด ดังนี้

2.1.1 แผนภาพคอมโพเนนท์ (Component Diagrams) [23]

แผนภาพคอมโพเนนท์ยูเอ็มแอล แสดงคอมโพเนนท์ในระบบและเป็นส่วนหนึ่งของมุมมองการพัฒนา (Development View) กล่าวถึงส่วนของระบบที่จัดตั้งเป็นโมดูลและคอมโพเนนท์อย่างไร ซึ่งสามารถช่วยจัดการในเรื่องเลเยอร์สำหรับสถาปัตยกรรมระบบ โดยแผนภาพคอมโพเนนท์ เป็นส่วนหนึ่งของซอฟต์แวร์ที่มีลักษณะการห่อหุ้ม (Encapsulated) การนำกลับมาใช้ใหม่ (Reusable) และการแทนที่ (Replaceable) ซึ่งคอมโพเนนท์ถูกสร้างขึ้นจากบล็อก (Block) และรวมเป็นรูปแบบซอฟต์แวร์ เนื่องจากคอมโพเนนท์เป็นหนึ่งในส่วนประกอบหลักสำหรับการออกแบบซอฟต์แวร์ โดยเฉพาะในด้านความสัมพันธ์แบบหลวม (Loosely Coupling) ที่ซึ่งเมื่อมีการเปลี่ยนแปลงเกิดขึ้นที่คอมโพเนนท์ใดๆ แล้วจะไม่ส่งผลกระทบต่อระบบที่มีอยู่ และสำหรับการสนับสนุนด้านความสัมพันธ์แบบหลวมและการห่อหุ้ม โดยการเข้าถึงคอมโพเนนท์จึงต้องเข้าถึงผ่านอินเตอร์เฟส ซึ่งอินเตอร์เฟสจะแยกส่วนพฤติกรรมออกจากส่วนการนำเสนอเพื่อลดโอกาสที่จะเกิดผลกระทบกันทั้งระบบจากการเปลี่ยนแปลงในคอมโพเนนท์ใดๆ อธิบายรายละเอียด ดังนี้

1) คอมโพเนนท์ยูเอ็มแอลเบื้องต้น (A Basic Component in UML)

คอมโพเนนท์ สร้างจากรูปสี่เหลี่ยมกำกับสเทอริโอไทป์ <<component>> และไอคอนแท็บสี่เหลี่ยมอยู่มุมขวาด้านบน ดังรูปที่ 2.1 ซึ่งแสดงคอมโพเนนท์ชื่อ “ConversionManagement”

รูปที่ 2.1 ตัวอย่างสัญลักษณ์คอมโพเนนท์ยูเอ็มแอล [23]

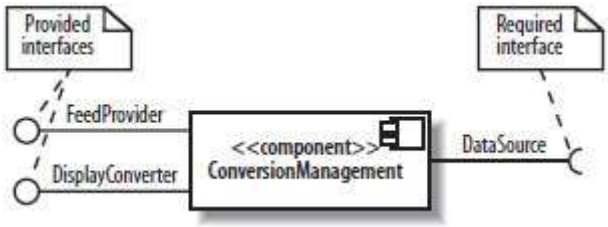
2) อินเตอร์เฟสโพรไวด์และรีไควร์ของคอมโพเนนท์ (Provided and Required Interfaces of a Component)

คอมโพเนนท์ จำเป็นที่จะมีความสัมพันธ์แบบหลวม ดังนั้นเมื่อมีการเปลี่ยนแปลงเกิดขึ้นที่คอมโพเนนท์ใดๆ แล้วจะต้องไม่ส่งผลกระทบต่อระบบอื่น จากสาเหตุดังกล่าวจึงมีการนำอินเตอร์เฟส

เข้ามาใช้งาน โดยคอมโพเนนท์จะมีปฏิสัมพันธ์กับคอมโพเนนท์อื่นผ่านทางอินเตอร์เฟสโพรไวด์และรีไควร์ในการควบคุมการขึ้นต่อกันระหว่างคอมโพเนนท์และการสร้างคอมโพเนนท์ให้สามารถถอดเปลี่ยนได้อธิบายรายละเอียด ดังนี้

- อินเตอร์เฟสโพรไวด์ของคอมโพเนนท์ เป็นอินเตอร์เฟสที่ซึ่งเตรียมไว้ให้บริการแก่คอมโพเนนท์หรือคลาสอื่น
- อินเตอร์เฟสรีไควร์ เป็นอินเตอร์เฟสที่ซึ่งต้องการไปใช้บริการตลอดจนฟังก์ชันจากคอมโพเนนท์หรือคลาสอื่นที่เตรียมไว้ให้บริการ

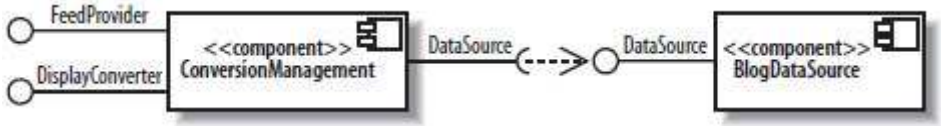
โดยในภาษายูเอ็มแอลอินเตอร์เฟสโพรไวด์และรีไควร์ มีสามรูปแบบที่เป็นมาตรฐานดังนี้ สัญลักษณ์บอล (Ball) และซ็อกเก็ต (Socket), สัญลักษณ์สเทอริโอไทป์และรายการข้อความ ซึ่งแสดงสัญลักษณ์บอลและซ็อกเก็ตสำหรับอินเตอร์เฟส โดยกำหนดให้คอมโพเนนท์ชื่อ “ConversionManagement” ประกอบอินเตอร์เฟสโพรไวด์ชื่อ “FeedProvider” และ “DisplayConverter” และอินเตอร์เฟสรีไควร์ชื่อ “DataSource” ดังรูปที่ 2.2



รูปที่ 2.2 ตัวอย่างสัญลักษณ์อินเตอร์เฟสโพรไวด์และรีไควร์ [23]

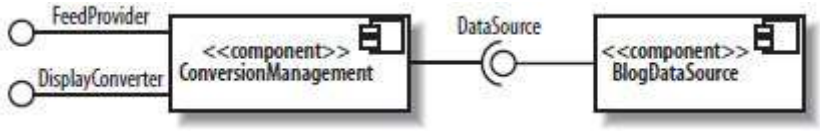
3) การทำงานร่วมกันของคอมโพเนนท์ (Components Working Together)

การทำงานร่วมกันระหว่างคอมโพเนนท์นั้น ถ้าคอมโพเนนท์ประกอบด้วยอินเตอร์เฟสรีไควร์เมื่อจำเป็นต้องใช้งานคอมโพเนนท์หรือคลาสอื่นในระบบที่ซึ่งมีอินเตอร์เฟสโพรไวด์ไว้ให้บริการ โดยแสดงให้เห็นว่าคอมโพเนนท์กับอินเตอร์เฟสรีไควร์ขึ้นอยู่กับคอมโพเนนท์ที่มีอินเตอร์เฟสโพรไวด์นั้น ซึ่งสามารถวาดลูกศรแบบขึ้นต่อกันจากสัญลักษณ์ซ็อกเก็ตของคอมโพเนนท์ที่ขึ้นอยู่กับไปยังสัญลักษณ์บอลของคอมโพเนนท์ที่ให้บริการดังกล่าว แสดงดังรูปที่ 2.3 โดยคอมโพเนนท์ชื่อ “ConversionManagement” ประกอบด้วยอินเตอร์เฟสรีไควร์ชื่อ “DataSource” และคอมโพเนนท์ชื่อ “BlogDataSource” ที่มีอินเตอร์เฟสโพรไวด์เตรียมไว้ให้บริการแก่คอมโพเนนท์ดังกล่าว



รูปที่ 2.3 ตัวอย่างการทำงานร่วมกันของคอมโพเนนท์ ผ่านทางสัญลักษณ์อินเตอร์เฟสโพรไวด์และรีไควร์ [23]

สำหรับการนำเสนอสัญลักษณ์จากรูปที่ 2.3 ในเครื่องมือการสร้างแบบจำลองภาษายูเอ็มแอล สามารถที่จะนำสัญลักษณ์บอลและซ็อกเก็ตเชื่อมกันได้โดยซ่อนลูกศรแบบขึ้นต่อกัน ซึ่งสามารถแทนสัญลักษณ์การเชื่อมต่อแบบแอสแซมบลีได้เช่นเดียวกัน ดังรูปที่ 2.4



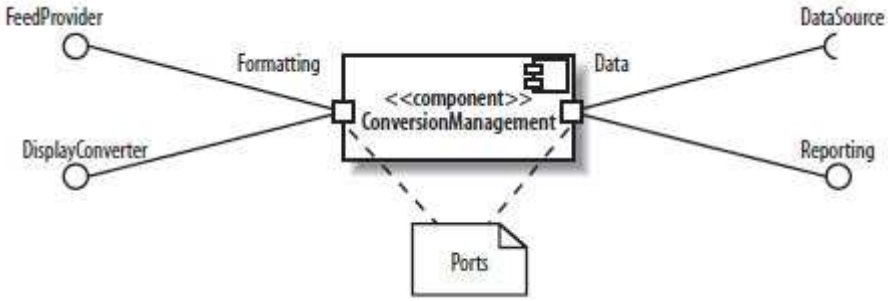
รูปที่ 2.4 ตัวอย่างการนำเสนอสัญลักษณ์บอลและซ็อกเก็ต ด้วยการเชื่อมต่อแบบแอสแซมบลี [23]

อย่างไรก็ตาม สามารถซ่อนอินเตอร์เฟสและเขียนความสัมพันธ์แบบขึ้นต่อกันโดยตรงระหว่างคอมโพเนนท์ได้เช่นเดียวกัน ดังรูปที่ 2.5

รูปที่ 2.5 ตัวอย่างการนำเสนอสัญลักษณ์ความสัมพันธ์แบบขึ้นต่อกันระหว่างคอมโพเนนท์ [23]

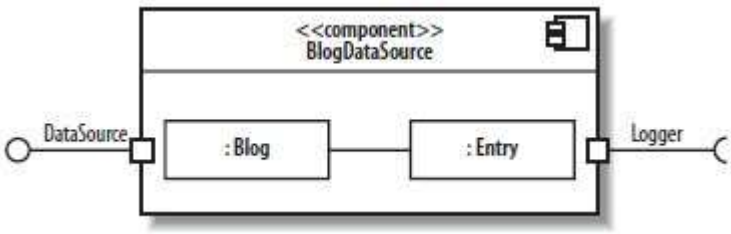
4) พอร์ตและโครงสร้างภายใน (Ports and Internal Structure)

คอมโพเนนท์ สามารถประกอบด้วยพอร์ตและโครงสร้างภายใน ซึ่งสามารถใช้พอร์ตในการจำลองวิธีการที่ต่างกันซึ่งคอมโพเนนท์สามารถใช้อินเตอร์เฟสที่เกี่ยวข้องกันแนบไปกับพอร์ต ดังรูปที่ 2.6 คอมโพเนนท์ชื่อ “ConversionManagement” ประกอบด้วยพอร์ตชื่อ “Formatting” กับ “Data” ตามลำดับโดยแต่ละพอร์ตก็มีความสัมพันธ์กับอินเตอร์เฟสที่ต่างกัน ดังรูปที่ 2.6



รูปที่ 2.6 ตัวอย่างสัญลักษณ์พอร์ตสำหรับคอมโพเนนท์รวมกลุ่มอินเตอร์เฟสที่ใช้งานเหมือนกัน [23]

ในคอมโพเนนท์ยังสามารถแสดงโครงสร้างภายในแบบจำลองซึ่งประกอบด้วยส่วน (Parts) คุณสมบัติ (Properties) และการเชื่อมต่อ (Connectors) ดังรูปที่ 2.7 แสดงโครงสร้างภายในคอมโพเนนท์ชื่อ “BlogDataSource” ที่ซึ่งประกอบด้วยคลาสชื่อ “Blog” และ “Entry” ตามลำดับ

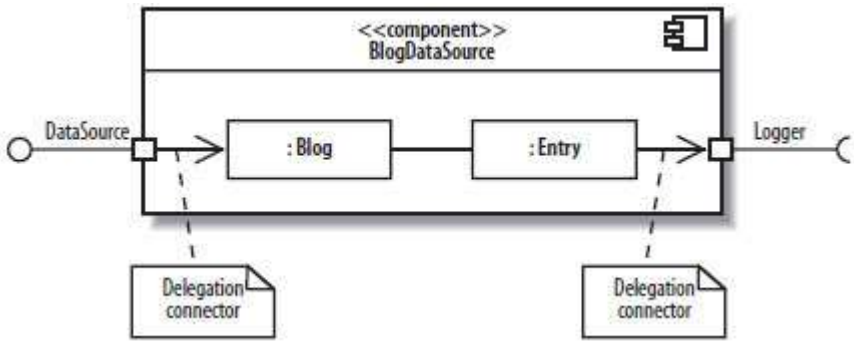


รูปที่ 2.7 โครงสร้างภายในคอมโพเนนท์ [23]

5) การเชื่อมต่อแบบดีลิเกชัน (Delegation Connectors)

สำหรับคอมโพเนนท์ ที่มีอินเตอร์เฟสโพรไวด์ที่สัมพันธ์กับส่วนภายใน (Internal Parts) โดยสิ่งเหมือนกันส่วนภายในก็มีคอมโพเนนท์ที่ต้องการติดต่อกับส่วนภายนอก ในกรณีเช่นนี้สามารถใช้การเชื่อมต่อแบบดีลิเกชันเพื่อแสดงให้เห็นว่ามีการเชื่อมต่อส่วนภายในจริงหรือมีการใช้อินเตอร์เฟสของคอมโพเนนท์

การเชื่อมต่อแบบดีลิเกทชันสามารถเขียนด้วยลูกศรแสดงทิศทางการเชื่อมต่อ โดยแนบไปกับพอร์ตเพื่อส่งต่อไปยังอินเตอร์เฟสกับส่วนภายใน ถ้าเป็นอินเตอร์เฟสโพรไวด์ทิศทางลูกศรจะส่งออกจากพอร์ตไปยังส่วนภายใน และถ้าเป็นอินเตอร์เฟสรีไควร์ทิศทางลูกศรจะส่งออกจากส่วนภายในไปยังพอร์ต ดังรูปที่ 2.8

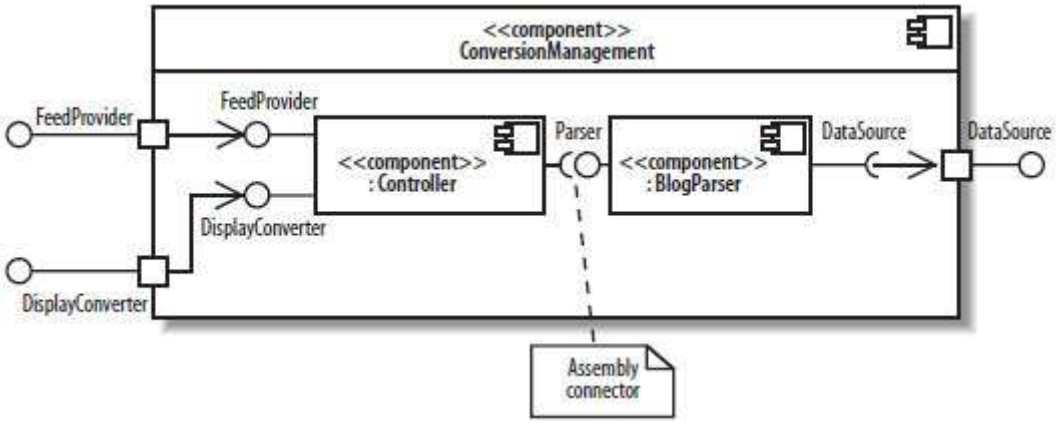


รูปที่ 2.8 ตัวอย่างการเชื่อมต่อแบบดีลิเกชัน [23]

จากรูปที่ 2.8 แสดงตัวอย่างการเชื่อมต่อแบบดีลิเกชันที่สอดคล้องกันกับส่วนของอินเตอร์เฟส ซึ่งคลาสชื่อ “Blog” เชื่อมต่อกับโพรไวด์อินเตอร์เฟสชื่อ “DataSource” และคลาส “Entry” เชื่อมต่อกับรำควร์อินเตอร์เฟสชื่อ “Logger” ตามลำดับ

6) การเชื่อมต่อแบบแอสแซมบลี (Assembly Connectors)

การเชื่อมต่อแบบแอสแซมบลี แสดงถึงคอมโพเนนท์ที่ต้องการเรียกใช้งานอินเตอร์เฟสซึ่งคอมโพเนนท์ให้บริการอินเตอร์เฟสดังกล่าว โดยการเชื่อมต่อแบบแอสแซมบลีจะใช้สัญลักษณ์บอลและ ซ็อกเก็ต เพื่อแทนด้วยอินเตอร์เฟสรีไควร์และโพรไวด์ตามลำดับ ดังรูปที่ 2.9



รูปที่ 2.9 ตัวอย่างการเชื่อมต่อแบบแอสแซมบลี [23]

จากรูปที่ 2.9 แสดงสัญลักษณ์การเชื่อมต่อแอสแซมบลีที่เชื่อมต่อไปยังคอมโพเนนท์ชื่อ “Controller” ไปยังคอมโพเนนท์ชื่อ “BlogParser” ตามลำดับ

2.1.2 ภาษาเอกซ์เอ็มไอ (XML Metadata Interchange: XMI) [5]

ภาษาเอกซ์เอ็มไอ เป็นมาตรฐานในการจัดเก็บแผนภาพและเป็นรูปแบบการแลกเปลี่ยนข้อมูลกันระหว่างเครื่องมือด้านการออกแบบสำหรับแบบจำลองยูเอ็มแอล มีภาษาเอกซ์เอ็มแอล ซึ่งเป็นภาษาตั้งต้นสำหรับการนิยามมาตรฐานของภาษาเอกซ์เอ็มไอ โดยองค์กรบริหารงานเชิงวัตถุเป็นผู้กำหนดมาตรฐานและแนวทางดังกล่าว สำหรับการกำหนดรูปแบบภาษาเอกซ์เอ็มไอ ดังนี้

- เอกสารเอกซ์เอ็มไอ (XMI Document) เป็นการกำหนดรูปแบบของเอกสารที่ถูกต้อง (Valid) และดี (Well-formed) ซึ่งนิยามด้วยภาษาเอกซ์เอ็มแอล และมีความสอดคล้องกับกฎการสร้างที่กำหนดในมาตรฐาน
- เอกซ์เอ็มไอสกีมา (XMI Schema) เป็นการสร้างให้เกิดความเทียบเท่ากับกฎการสร้างเอกซ์เอ็มไอสกีมาในเอกสารใดๆ ซึ่งหมายถึงเอกสารเอกซ์เอ็มไอต้องถูกต้องและสอดคล้องกันตามกฎการสร้างภาษาเอกซ์เอ็มไอ

โดยหลักการสำคัญของเอกซ์เอ็มไอสกีมาสำหรับกำหนดภาษาเอกซ์เอ็มไอ คือ เป็นการนิยามลักษณะขององค์ประกอบเอกซ์เอ็มไอ (XMI element) รวมทั้งประเภทและกลุ่มของลักษณะประจำ (Attribute) ซึ่งรวมอยู่ในชุดคำสั่งสำหรับใช้ในการตรวจสอบรูปแบบเอกซ์เอ็มแอล เพื่อให้สอดคล้องกับรายละเอียดในเมทาดาต้าและทุกๆ องค์ประกอบเอกซ์เอ็มแอลที่กำหนดด้วยข้อกำหนดในเนมสเปซ ตาม “http://schema.omg.org/spec/XMI/*version*” โดยที่ *version* เป็นรุ่นที่ใช้สำหรับกำหนดรายละเอียดของภาษาเอกซ์เอ็มไอและกลไกของเอกซ์เอ็มแอลเนมสเปซ (XML namespace) สามารถนำหลักการของเอ็มโอเอฟโมเดล (MOF model) [9] ซึ่งเป็นสถาปัตยกรรมสี่แบบสำหรับสร้างเมตาโมเดล (Metamodel) [10] ที่ซึ่งสามารถนำออกจากแพลตฟอร์มใดๆ ไปแสดงผลยังแพลตฟอร์มที่ต่างกันได้ นำไปสู่การแก้ปัญหาสำหรับนักออกแบบและผู้พัฒนาเพื่อการทำความเข้าใจ

ร่วมกันด้านสถาปัตยกรรม และช่วยขจัดความขัดแย้งระหว่างองค์ประกอบในภาษาเอกซ์เอ็มไอกับเอกซ์เอ็มแอล สามารถแสดงโครงสร้างของเอกซ์เอ็มไอสกีมาและเอกสารเอกซ์เอ็มไอ [5] โครงสร้างของข้อมูลในภาษาเอกซ์เอ็มไอ แสดงตัวอย่างรูปแบบเอกซ์เอ็มไอสกีมาข้อมูลภาษาเอกซ์เอ็มไอ ดังรูปที่ 2.10

```
<?xml version="1.0" encoding="UTF-8"?> <xsd:schema
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xp="http:///xp.ecore" targetNamespace=
"http:///xp.ecore">
  <xsd:annotation>
    <xsd:documentation>An architectural view
     that describes the concurrent
    aspect of the system: tasks (processes) and
    their interactions.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:import schemaLocation="XMI.xsd" namespace=
  "http://www.omg.org/XMI"/>
  <xsd:complexType name="UserStory">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="title" type="xsd:string"/>
    <xsd:attribute name="pointsDone" type="xsd:int"/>
  </xsd:complexType>
  <xsd:element name="UserStory" type="xp:UserStory"/>
</xsd:schema>
```

รูปที่ 2.10 เอกซ์เอ็มไอสกีมาของข้อมูลภาษาเอกซ์เอ็มไอ [11]

ส่วนโครงสร้างของเอกสารเอกซ์เอ็มไอที่เป็นการกำหนดกฎในการดำเนินการกับลำดับค่าอินสแตนซ์ (Instance) ของแบบจำลองในเอกสารเอกซ์เอ็มไอดังรูปที่ 2.11

```
<?xml version="1.0" encoding="ASCII"?> <xp:UserStory
xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xp="http:///xp.ecore" title="User Authentication"
pointsDone="6"/>
```

รูปที่ 2.11 เอกสารเอกซ์เอ็มไอของข้อมูลภาษาเอกซ์เอ็มไอ [11]

จากรูปที่ 2.11 ซึ่งแสดงให้เป็นว่าลักษณะของเอกสารในรูปแบบภาษาเอกซ์เอ็มไอไปเป็นภาษาเอกซ์เอ็มแอล ที่ซึ่งมีการกำหนดเวอร์ชันและลักษณะของการเข้ารหัสแต่ละภาษาและตามด้วยรากขององค์ประกอบเอกซ์เอ็มไอ คือ คลาส UserStory และมีการใช้เนมสเปซเพื่อแยกความแตกต่างสำหรับคลาสที่มีชื่อซ้ำซ้อนกัน คือ xmi ซึ่งกำหนดโดยองค์ประกอบและลักษณะประจำของเอกซ์เอ็มไอ โดยมีลักษณะประจำเป็นรุ่นที่ 2.0 มี namespace เป็น xp แทนองค์ประกอบที่ถูกกำหนดในแบบจำลองยูเอ็มแอลและลักษณะประจำ title กับ pointDone ของคลาส UserSroty จะถูกแทนด้วยลักษณะประจำในรูปแบบขององค์ประกอบเอกซ์เอ็มแอลเป็น xp:UserStory โดยรูปแบบภาษาเอกซ์เอ็มไอในกฎที่กำหนดขึ้นสำหรับการจัดลำดับความสัมพันธ์ที่ระบุไว้ในภาษาบีเอ็นเอฟ (Backus-Naur Form: BNF) [12] เป็นภาษาสำหรับกำหนดรูปแบบภาษาเอกซ์เอ็มไอโดยเฉพาะ

2.1.3 ตัวแปลภาษา (Compiler) [13]

ตัวแปลภาษา เป็นรูปแบบของการแปลภาษาต้นฉบับซึ่งเป็นภาษาระดับสูง ที่ซึ่งถูกเขียนเป็นภาษาโปรแกรมและภาษาวัตถุ (Object Language) ใดๆ และถูกดำเนินการแปลงเพื่อให้อยู่ในรูปแบบที่ใกล้เคียงกับภาษาเครื่อง (Machine Language) สำหรับการนำเสนอเครื่องมือตัวแปลภาษาในวิทยานิพนธ์ฉบับนี้จะนำเสนอด้วยการกำหนดกฎสำหรับภาษาโปรแกรม โดยแปลงการดำเนินงานตามที่ระบุไว้ในไวยากรณ์ข้อมูลจริงกับรูปแบบไวยากรณ์ไม่พึ่งบริบท ที่ซึ่งจะนำไปสร้างเป็นตัวแปลภาษา ซึ่งการแปลภาษานั้นเป็นแนวทางที่ต้องดำเนินการตามไวยากรณ์ของภาษาต้นฉบับหรืออาจกล่าวได้ว่าเป็นไวยากรณ์กำกับ (Syntax-directed) [13] โดยทั่วไปตัวแปลภาษาจะประกอบด้วยหลายขั้นตอนในแต่ละขั้นตอนจะดำเนินการส่งข้อมูลผลลัพธ์ไปยังขั้นตอนต่อไป ซึ่งประกอบด้วยห้าขั้นตอน ดังนี้

1) ขั้นตอนการวิเคราะห์คำศัพท์ (Lexical analyzer phase)

ขั้นตอนการวิเคราะห์คำศัพท์เป็นการจัดกลุ่มอักขระในหน่วยของคำศัพท์ (Lexical) หรือโทเค็น (Token) สำหรับเป็นข้อมูลนำเข้าในขั้นตอนการวิเคราะห์คำศัพท์ในรูปแบบของสายอักขระและผลลัพธ์ที่ได้อยู่ในรูปแบบที่เป็นสายอักขระของโทเค็นโดยใช้เรกูลาร์เอกซ์เพรสชัน [14] สำหรับการนิยามกฎเพื่อสืบค้นโทเค็นของข้อมูลนำเข้า เรียกว่า สแกนเนอร์ (Scanner) ที่ซึ่งนำเสนอในรูปแบบของเครื่องสถานะจำกัด (Finite State Machine) มีเครื่องมือเล็กซ์ (Lex) [15] และเฟล็กซ์ [14] เป็นเครื่องมือที่สนับสนุนการทำงานกับการสร้างสแกนเนอร์จำแนกแบบรูปของโทเค็นในสายอักขระ ซึ่งแสดงการทำงานดังรูปที่ 2.12

Lexeme	Token Type
foo	Variable
=	Assignment Operator
1	Number
-	Subtraction Operator
3	Number
**	Power Operator
2	Number

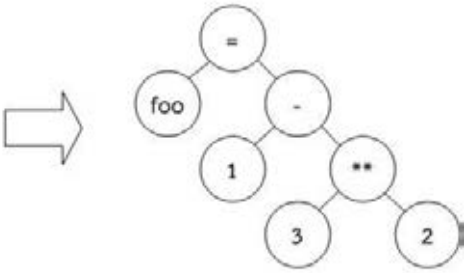
รูปที่ 2.12 การดำเนินการของสแกนเนอร์ [16]

2) ขั้นตอนตัวแจงส่วน (Parser phase)

ขั้นตอนตัวแจงส่วนเป็นการจัดกลุ่มของเครื่องหมาย โดยนำไปสู่การสร้างหน่วยของประโยคผลลัพธ์ ซึ่งในตัวแจงส่วน (Parser) แสดงลักษณะการวิเคราะห์คำในต้นไม้การแจงส่วน (Parse Tree) ที่มีไวยากรณ์ไม่พึ่งบริบท (Context-Free Grammar) สำหรับนิยามโครงสร้างโปรแกรม นำเสนอด้วย

ออโตมาตาแบบกดลง (Push-Down Automata) โดยมีเครื่องมือแย็กค์ [15], ไบสัน (Bison) [14] ที่ซึ่งทำหน้าที่จำแนกโครงสร้างทางไวยากรณ์โปรแกรม ดังรูปที่ 2.13

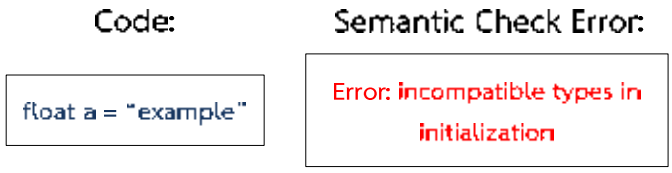
Lexeme	Token Type
foo	Variable
=	Assignment Operator
1	Number
-	Subtraction Operator
3	Number
**	Power Operator
2	Number



รูปที่ 2.13 การดำเนินการของตัวแจงส่วน [16]

3) ขั้นตอนการวิเคราะห์ความหมาย (Semantic analyzer phase)

ขั้นตอนการวิเคราะห์ความหมาย เป็นการวิเคราะห์คำในต้นไม้การแจงส่วนตามบริบทของข้อมูล เรียกว่า ค่าความหมายคงที่ (Static Semantic) ผลลัพธ์จากขั้นตอนนี้จัดเก็บข้อมูลในรูปแบบต้นไม้ โดยนำคุณสมบัติทางไวยากรณ์ไปใช้อธิบายลักษณะค่าความหมายของภาษาโปรแกรม โดยปกติจะถูกรวมเข้าไว้ในขั้นตอนการแจงส่วนในระหว่างการจำแนกข้อมูลที่เกี่ยวข้องกับตัวแปรหรือวัตถุที่เก็บไว้ในตารางสัญลักษณ์ (Symbol table) เพื่อใช้ในการดำเนินการตรวจสอบตามบริบท ดังรูปที่ 2.14 อธิบายรหัสโปรแกรม ที่ซึ่งประกาศตัวแปรเป็นประเภททศนิยมแต่กำหนดค่าเป็นประเภทข้อความ ในขั้นตอนการวิเคราะห์ความหมายจึงตรวจสอบพบว่าเป็นข้อผิดพลาด



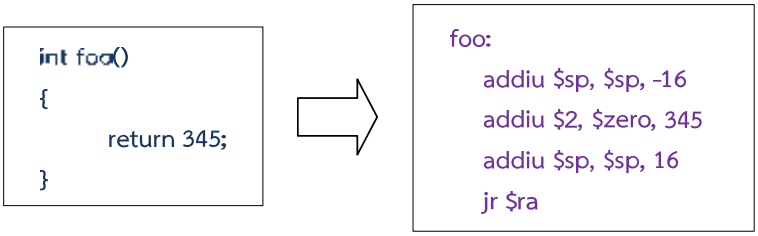
รูปที่ 2.14 การดำเนินการวิเคราะห์ความหมาย [16]

4) ขั้นตอนการปรับปรุง (Optimizer phase)

ขั้นตอนการปรับปรุง เป็นการจัดเก็บลักษณะทางความหมายจากการแปลงคำอธิบายที่ซึ่งเป็นส่วนประกอบของต้นไม้ เพื่อลดความซับซ้อนของโครงสร้างและจะอำนวยความสะดวกเพื่อนำไปสู่การสร้างรหัสที่มีประสิทธิภาพมากยิ่งขึ้น

5) ขั้นตอนการสร้างรหัส (Code generator phase)

ขั้นตอนการสร้างรหัส เป็นการแปลงข้อมูลในต้นไม้การแจงส่วนของรหัสออบเจ็กต์ (Object Code) โดยใช้กฎที่มีความหมายสอดคล้องกับภาษาต้นฉบับ ซึ่งบางครั้งอาจรวมขั้นตอนนี้ไว้ในขั้นตอนการแจงส่วน แสดงตัวอย่าง ดังรูปที่ 2.15 ซึ่งเป็นลักษณะการสร้างรหัสวัตถุด้วยภาษาต้นฉบับ



รูปที่ 2.15 การดำเนินการสร้างรหัส [16]

ตัวอย่างรูปแบบการนำเสนอตัวแปลภาษาในรูปแบบการนิยามภาษาโปรแกรมเบื้องต้น ดังรูปที่ 2.16

```
program ::= LET [ declarations ] IN command sequence END
declarations ::= INTEGER [ id seq ] IDENTIFIER .
id seq ::= id seq... IDENTIFIER ,
command sequence ::= command... command
command ::= SKIP ;
          | IDENTIFIER := expression ;
          | IF exp THEN command sequence ELSE command sequence FI ;
          | WHILE exp DO command sequence END ;
          | READ IDENTIFIER ;
          | WRITE expression ;
expression ::= NUMBER | IDENTIFIER | '(' expression ')'
          | expression + expression | expression - expression
          | expression * expression | expression / expression
          | expression ^ expression
          | expression = expression
          | expression < expression
          | expression > expression
```

รูปที่ 2.16 ตัวอย่างรูปแบบภาษาการนำเสนอตัวแปลภาษาเบื้องต้น [16]

จากรูปที่ 2.16 เป็นรูปแบบภาษาการนำเสนอตัวแปลภาษาเบื้องต้น ซึ่งมีไวยากรณ์ไม่พึ่งบริบท โดยอักขระตัวพิมพ์เล็กแทนสัญลักษณ์นอลเทอร์มินอล (Non-Terminal) ส่วนอักขระที่อยู่ภายในเครื่องหมาย “[ ]” หรือสัญลักษณ์จริงแทนสัญลักษณ์เทอร์มินอล (Terminal) แต่ก็พบว่าถ้าใช้สัญลักษณ์จริงแทนด้วยเทอร์มินอลจะมีความขัดแย้งกับสัญลักษณ์เมตา (Meta) ในเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ ดังนั้น เพื่อระบุว่าเป็นสัญลักษณ์เทอร์มินอลที่แทนด้วยสัญลักษณ์แท้จริง จะมีเครื่องหมาย ‘ ’ กำกับไว้ ตัวอย่าง ‘(’ และอักขระตัวพิมพ์ใหญ่แทนด้วยไวยากรณ์ของภาษา

2.1.4 เครื่องมือเฟล็กซ์ (Flex) [14] และแย็กค์ (Yacc) [15]

เฟล็กซ์และแย็กค์เป็นเครื่องมือสำหรับสร้างโปรแกรมการแปลงโครงสร้างข้อมูลนำเข้า (Input) ซึ่งวิเคราะห์และสืบค้นจากแบบรูปของข้อความ และยังเป็นเครื่องมือที่ออกแบบสำหรับสร้างตัวแปลภาษาที่นำไปใช้เป็นประโยชน์ได้กับงานด้านอื่นที่ต้องการนำเสนอในรูปแบบตัวแปลภาษา ซึ่ง

ตัวแปลภาษาเกิดขึ้นในปีคริสต์ศักราช (ค.ศ.) 1950 โดยใช้วิธีการทำงานแบบเฉพาะสำหรับการวิเคราะห์ไวยากรณ์ของรหัสโปรแกรม ต่อมาในปี ค.ศ. 1960 นักวิชาการได้หันมาให้ความสนใจกันมากขึ้นจึงส่งผลให้ปี ค.ศ. 1970 จึงเกิดการวิเคราะห์เพื่อการทำความเข้าใจร่วมกันสำหรับงานด้านตัวแปลภาษา

ลักษณะการทำงานของเครื่องมือดังกล่าวได้แยกการทำงานออกเป็นสองส่วนที่สำคัญ คือ การวิเคราะห์คำ (Lexical Analysis) หรือสแกนเนอร์และการวิเคราะห์ไวยากรณ์ (Syntax Analyzer) หรือตัวแจงส่วน โดยที่สแกนเนอร์จะแยกเป็นคำที่มีความหมาย เรียกว่า โทเค็นและตัวแจงส่วน จะดำเนินการแยกโทเค็นและดูความสัมพันธ์กับโทเค็นตัวอื่น ดังตัวอย่างรหัสภาษาซีอย่างสั้น

alpha = beta + gamma ;

สแกนเนอร์จะแยกเป็นโทเค็นเป็น alpha, "=", beta, "+", gamma และ ";" ตามลำดับ ดังนั้นการทำงานของพาสเซอร์กำหนดให้ beta + gamma เป็นหนึ่งนิพจน์ และหลังจากนั้นจะกำหนดค่านิพจน์ดังกล่าวให้กับโทเค็น alpha โดยส่วนใหญ่แล้วระบบปฏิบัติการลีนุกซ์จะมี Flex เป็นส่วนหนึ่งของระบบ แต่ถ้าไม่มีสามารถติดตั้งโปรแกรม Flex จาก http://flex.sourceforge.net/ ซึ่งเป็นโครงการซอร์สฟอร์จ (Sourceforge Project) และโปรแกรม Yacc สามารถดาวน์โหลดจาก http://invisible-island.net/byacc/byacc.html

สำหรับโครงสร้างโปรแกรม Flex และ Yacc จะประกอบด้วยสามส่วน คือ ส่วนกำหนดคำนิยาม (Definition) ส่วนกฎดำเนินการ (Rule) และส่วนการเรียกใช้งาน (User subroutines) ดังโปรแกรมต่อไปนี้

```
...definition section ...
%%
... rules section ...
%%
... user subroutines ...
```

จากโปรแกรม ในแต่ละส่วนจะแยกด้วยเครื่องหมาย "%%" โดยในส่วนกำหนดคำนิยามต้องระบุ ส่วนกำหนดกฎอาจจะไม่ระบุและส่วนการเรียกใช้งานจะดำเนินการหลังจากเครื่องหมาย "%%" สุดท้าย เนื่องจากในโปรแกรมเฟล็กซ์โดยตัวโครงสร้างภาษาถูกเขียนขึ้นมาด้วยโปรแกรมภาษาซี

ต่อมา Elliot Berk ณ มหาวิทยาลัยพรินท์ตัน (Princeton University) ได้พัฒนาต่อยอดจากโปรแกรมเฟล็กซ์เดิมเป็นโปรแกรมภาษาซีพลัสพลัส (C++) สำหรับเป็นเครื่องมือเฟล็กซ์และต่อมาได้พัฒนาโดยใช้ประโยชน์จากเครื่องมือเจเฟล็กซ์ ซึ่งเป็นเครื่องมือสำหรับสร้างการวิเคราะห์คำด้วยโปรแกรมภาษาจาวา (Java) เรียกว่า JFlex (Flast Lexical analyser generator for Java) [17] และในโปรแกรมแย็กค์ Scott E. Hudson จากสถาบันเทคโนโลยีจอร์เจีย (Georgia Institute of Technology) ได้พัฒนาเครื่องมือคัพ (CUP) [18] ซึ่งเป็นเครื่องมือสำหรับสร้างตัวแจงส่วนด้วยภาษาจาวาและยังสนับสนุนการทำงานร่วมกับเครื่องมือเจเฟล็กซ์

2.1.5 ภาษาแอคมี (Acme Language) [6]

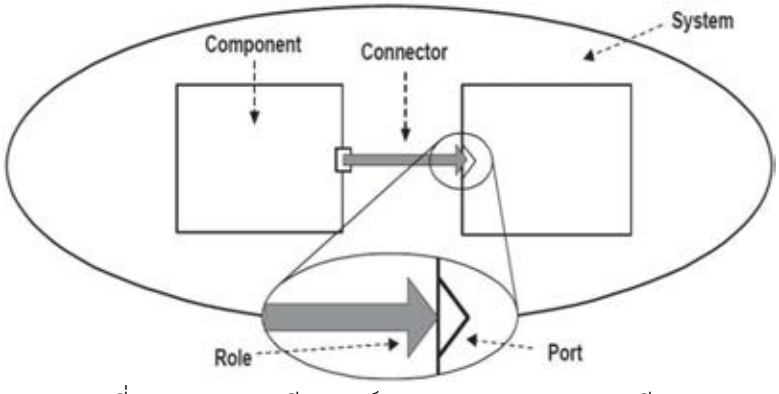
ภาษาแอคมี เป็นภาษาสำหรับการแลกเปลี่ยนคำอธิบายการออกแบบทางสถาปัตยกรรม โดยมีการกำหนดรูปแบบของคำศัพท์ การแสดงโครงสร้างทางสถาปัตยกรรม โดยภาษาแอคมีมีเป้าหมายและประโยชน์ คือ สามารถใช้ข้อมูลโครงสร้างหลักของภาษา เพื่อสนับสนุนการทำงานร่วมกันระหว่างเครื่องมือทางภาษาเอดีแอล ตัวอย่างเช่น ภาษา Aesop [6], C2 [6] และ Wright [6] เป็นต้น ซึ่งแต่ละภาษาที่กล่าวมานั้นมีความสามารถในเฉพาะด้านที่แสดงถึงคุณลักษณะทางด้านสถาปัตยกรรมที่ต่างกัน ในกลุ่มผู้วิจัยภาษาแอคมีพยายามร่วมมือกันพัฒนานวัตกรรมทางวิชาการ เพื่อเป็นรูปแบบการแลกเปลี่ยนสำหรับเครื่องมือออกแบบด้านสถาปัตยกรรม โดยมีคุณสมบัติที่สำคัญ ดังนี้

1) ออกแบบประเภทองค์ประกอบสถาปัตยกรรมแอคมี (Acme Architecture Design Element Type)

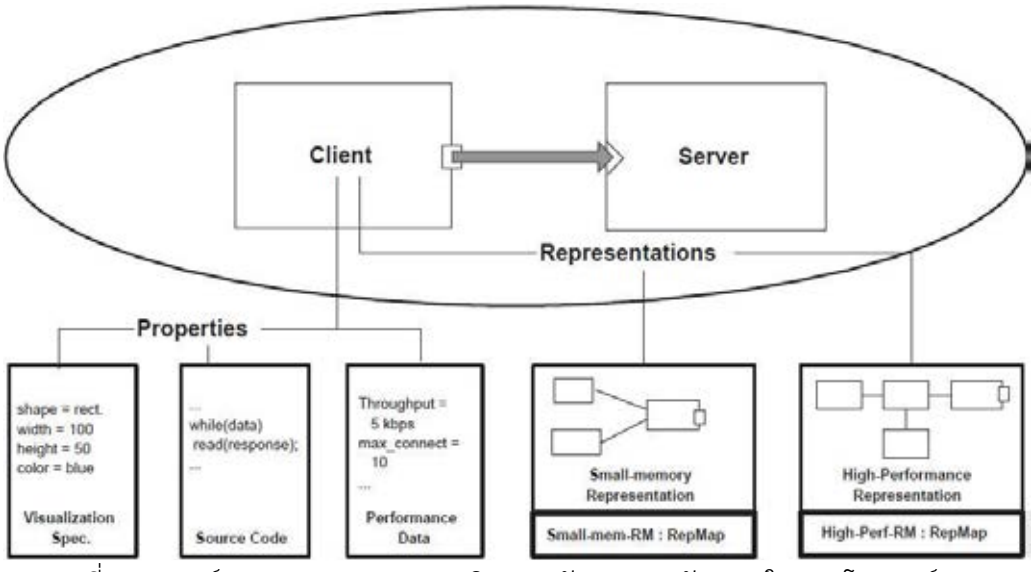
ภาษาแอคมี สร้างจากพื้นฐานความสำคัญของออนโทโลยี (Ontology) ของสิ่งที่สนใจเจ็ดประเภท ซึ่งใช้แสดงลักษณะของสถาปัตยกรรม ดังนี้ คอมโพเนนท์ (Component) การเชื่อมต่อ (Connector) ระบบ (System) พอร์ต (Port) บทบาท (Role) เรพพริเซนเทชัน (Representation) และเรพพริเซนเทชัน-แมพ (Representations-map) ส่วนใหญ่องค์ประกอบที่นำมาใช้อธิบายรายละเอียดทางสถาปัตยกรรมจะประกอบด้วยคอมโพเนนท์ การเชื่อมต่อและระบบ ซึ่งในวิทยานิพนธ์ฉบับนี้ขอยกตัวอย่างเกี่ยวกับรายละเอียดประเภทขององค์ประกอบเบื้องต้นสามองค์ประกอบ ดังนี้

- คอมโพเนนท์ ใช้เป็นสัญลักษณ์การแสดงถึงองค์ประกอบและที่เก็บข้อมูลของระบบเป็นสำคัญ ตัวอย่างเช่น เครื่องลูกข่าย (Client) เครื่องแม่ข่าย (Server) และฐานข้อมูล (Database) เป็นต้น
- การเชื่อมต่อ ใช้เป็นสัญลักษณ์การแสดงถึงการปฏิสัมพันธ์ระหว่างคอมโพเนนท์ เสมือนเป็นตัวเชื่อมของการออกแบบทางด้านสถาปัตยกรรม ตัวอย่างเช่น ไปป์ (Pipe) โปรซีเจอร์คอล (Procedure Call) อีเว้นท์บอร์ดแคส (Event broadcast) เป็นต้น
- ระบบ ใช้เป็นสัญลักษณ์ของการกำหนดค่าในคอมโพเนนท์และการเชื่อมต่อ

ดังรูปที่ 2.17 แสดงเกี่ยวกับรายละเอียดองค์ประกอบในภาษาแอคมีและรูปที่ 2.18 แสดงองค์ประกอบประเภทเรพพริเซนเทชันและคุณลักษณะ (Properties) ในคอมโพเนนท์

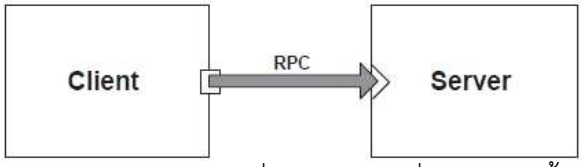


รูปที่ 2.17 รายละเอียดองค์ประกอบของภาษาแอคมี [6]



รูปที่ 2.18 องค์ประกอบประเภทเรพพริเซนเทชันและคุณลักษณะในคอมโพเนนท์ [6]

ตัวอย่างแผนภาพเครื่องลูกข่าย-เครื่องแม่ข่าย เบื้องต้นแสดงได้ดังรูปที่ 2.19 ซึ่งประกอบด้วยคอมโพเนนท์ คือ เครื่องลูกข่ายกับเครื่องแม่ข่ายที่เชื่อมต่อกันด้วยการเชื่อมต่อชื่ออาร์พีซี (RPC) และรูปที่ 2.20 แสดงตัวอย่างระบบเครื่องลูกข่าย-เครื่องแม่ข่าย ในรูปแบบภาษาแอคมีเบื้องต้น โดยพบว่าในคอมโพเนนท์เครื่องลูกข่ายมีการประกาศพอร์ตเพียงหนึ่งพอร์ต คือ send-request และในคอมโพเนนท์เครื่องแม่ข่ายมีการประกาศ พอร์ตเพียงหนึ่งพอร์ตเช่นเดียวกันคือ receive-request ส่วนในการเชื่อมต่อได้ออกแบบให้มีการทำงานเป็นสองบทบาทคือ caller และ callee โดยออนโทโลยีของระบบนี้ถูกประกาศโดยกลุ่มของรายการแอทแทชเมนท์ (Attachment)



รูปที่ 2.19 ตัวอย่างแผนภาพเครื่องลูกข่าย-เครื่องแม่ข่ายเบื้องต้น [6]

```
System simple_cs = {
  Component client = { Port send-requestt }
  Component server = { Port receive-request }
  Connector rpc  = { Roles {caller, callee} }
  Attachments : {
      client.send-request to rpc.caller ;
      server.receive-request to rpc.callee }
}
```

รูปที่ 2.20 ตัวอย่างระบบเครื่องลูกข่าย-เครื่องแม่ข่ายในรูปแบบภาษาแอคมีเบื้องต้น [6]

2) ลักษณะภาษาแอคมี (Acme Properties)

สำหรับลักษณะของภาษาแอคมี เป็นประโยชน์สำหรับการนำมาสร้างเครื่องมือเพื่อช่วยในการวิเคราะห์ การแปลงและการจัดการ ซึ่ง “type” ของลักษณะสามารถระบุความเป็น “sublanguage” ของลักษณะนั้นๆ ได้ โดยสนับสนุนการนิยามชนิดข้อมูล (Type) เบื้องต้น เช่น

จำนวนเต็ม (Integer) สายอักขระ (String) และค่าความจริง (Boolean) เป็นต้น สำหรับชนิดข้อมูลอื่นที่ภาษาแอคมีไม่ได้นิยามไว้ ในเครื่องมือใดๆ ควรต้องจัดการไว้ให้ ซึ่งถ้าพฤติกรรมที่เครื่องมือภาษาแอคมีไม่สามารถทำความเข้าใจกับลักษณะนั้นได้ก็จะไม่ตีความลักษณะนั้น แต่จะยังคงรักษาลักษณะนั้นไว้ให้กับเครื่องมืออื่นที่สนับสนุนการทำงานร่วมกับคุณลักษณะดังกล่าว

3) นิยามแม่แบบและสไตล์ในภาษาแอคมี (Acme Template and Style Definition: Acme Families)

จากคุณสมบัติและโครงสร้างทางภาษาแอคมีข้างต้นเพียงพอที่จะใช้นิยามอินสแตนซ์ทางสถาปัตยกรรมได้ แต่ความเป็นจริงรูปแบบพื้นฐานสำหรับของคุณสมบัติหลักของภาษาแอคมีที่มีทั้งเครื่องมือที่สามารถแยกและไม่สามารถแยกอินสแตนซ์ดังกล่าวได้ แต่อย่างไรก็ตามในการนิยามดังกล่าวถ้าระบบมีความซับซ้อนและต้องการทำงานกับรูปแบบของระบบเดิมซ้ำๆ เฉพาะบางส่วนที่ต้องการนั้น ซึ่งในการแก้ปัญหาดังกล่าวในภาษาแอคมีจึงผนวกรวมเป็นนิยามของแม่แบบ (Templates) ซึ่งเป็นประเภทของพารามิเตอร์ที่กำหนดไว้ใช้สำหรับการทำงานในรูปแบบวนซ้ำและในแม่แบบมีการนิยามโครงสร้างไวยากรณ์ที่สามารถนำไปขยายแทนได้สำหรับดำเนินการประกาศเพื่อใช้งานใหม่ ซึ่งค่อนข้างจะมีความยืดหยุ่นต่อการนิยามแอทแทชเมนท์ของต่อการระบุคอมโพเนนท์และการเชื่อมต่อ แม่แบบมีประโยชน์คือสามารถที่จะขยายได้ตลอดเมื่อถูกจัดให้อยู่ในกลุ่มสไตล์ในสถาปัตยกรรม ในภาษาแอคมีสไตล์เป็นการนิยามเซตของแม่แบบที่เกี่ยวข้องกันที่ซึ่งสามารถสร้างเป็นคำศัพท์ที่ใช้ได้ทั่วไปในแฟมิลี่ของระบบ โดยสไตล์ได้จัดเตรียมกลไกสำหรับการจับและการนำโครงสร้างที่ใช้บ่อยกลับมาใช้ใหม่ซึ่งถือได้ว่าเป็นสำนวนสำหรับการออกแบบทางสถาปัตยกรรม

โดยประโยชน์ของแม่แบบ คือ สามารถนำไปขยายเพื่อสร้างเป็นกลุ่มสไตล์ (Styles) ของสถาปัตยกรรมที่มีความสัมพันธ์กับแม่แบบที่สร้างขึ้น เพื่อใช้ทำงานร่วมกันในรูปแบบของแฟมิลี (Family) ของระบบ ซึ่งสไตล์มีกลไกการตรวจสอบและนำโครงสร้างแม่แบบพื้นฐานกลับมาใช้ซ้ำ เช่น การใช้เครื่องลูกข่าย-เครื่องแม่ข่ายในรูปแบบของแม่แบบและสไตล์ สามารถนำไปสร้างเป็นรูปแบบของแม่แบบและนิยามอินสแตนซ์ของระบบที่สนับสนุนการทำงานร่วมกับหลายเครื่องลูกข่ายและเครื่องแม่ข่าย เป็นต้น

สำหรับตัวอย่างการใช้เครื่องลูกข่าย-เครื่องแม่ข่ายในรูปแบบของแม่แบบและสไตล์แสดงดังรูปที่ 2.21 ประกอบด้วยเครื่องลูกข่าย เครื่องแม่ข่ายและแม่แบบอาร์พีซีซึ่งเป็นการขยายลักษณะการทำงานของเครื่องลูกข่าย-เครื่องแม่ข่ายแบบเฉพาะโดยนำไปสร้างเป็นรูปแบบของแม่แบบและนิยามอินสแตนซ์ของระบบที่สนับสนุนการทำงานร่วมกับหลายๆ เครื่องลูกข่ายและเครื่องแม่ข่าย

4) การเปิดกรอบความหมายในภาษาแอคมี (Acme's Open Semantic Framework)

สำหรับรูปแบบของภาษาแอคมีอาศัยลักษณะการทำงานของการเปิดกรอบความหมาย (Open Semantic Framework) ซึ่งมีการจัดเตรียมโครงสร้างทางความหมายเบื้องต้น ในการระบุถึงลักษณะภาษาเอดีแอลที่มีความสัมพันธ์กันทางด้านสถาปัตยกรรม โดยใช้โครงสร้างทางคุณลักษณะและการเปิดกรอบความหมาย จับคู่กันระหว่างโครงสร้างของภาษากับรูปแบบทางตรรกะภายใต้ความสัมพันธ์และข้อจำกัด ที่ซึ่งได้จากการทำนาย เรียกว่า (Prescription) ตัวอย่างเช่น พิจารณาจากระบบเครื่องลูกข่าย-เครื่องแม่ข่ายอย่างง่ายจากรูปที่ 2.19 และ 2.20 ซึ่งเครื่องลูกข่ายจะเชื่อมต่อไปยังเครื่องแม่ข่ายเพียงหนึ่งการเชื่อมต่อที่กำหนดการวางเงื่อนไขได้ โดยกำหนดแอทแทชเมนท์

ด้วยการทำนายลักษณะการนำกลไกมาตรฐาน โดยนำหลักทฤษฎีของตรรกะวิทยาระดับแรก (First-order Logical) ทำงานร่วมกัน เพื่อกำหนดรูปแบบเชิงรูปนัย เป็นต้น

ตัวอย่างภาษาแอคมี ซึ่งพิจารณาจากระบบเครื่องแม่ข่ายอย่างง่ายจากรูปที่ 2.19 และ 2.20 และสามารถกำหนดการวางเงื่อนไขได้ ดังรูปที่ 2.22

```
Style client-server = {
    Component Template client(rpc-call-ports : Ports) = {
        Ports rpc-call-ports;
        Properties {   Aesop-style : style-id = client-server;
                       Unicon-style : style-id = cs;
                       source-code : external = "CODE-LIB/client.c" }
    }
    Component Template server(rpc-receive-ports : Ports) = {
        Ports rpc-receive-ports;
        Properties {   Aesop-style : style-id = client-server;
                       Unicon-style : style-id = cs; ... }
    }
    Template rpc(caller_port, callee_port : Port) defining (conn : Connector) =
    {
        conn = Connector {
            Roles {caller, callee}
                Properties {  synchronous : boolean = true;
                              max-roles : integer = 2; }
                              protocol : Wright = "..." }}
            Attachments {  conn.caller to caller_port;
                              conn.callee to callee_port; }
    }
}
System complex_cs : client-server = {
    c1 = client(send-request); c2 = client(send-request);
    c3 = client(send-request); s1 = server(receive-request);
    s2 = server(receive-request);
    rpc(c1.send-request, s1.receive-request);
    rpc(c2.send-request, s1.receive-request);
    rpc(c3.send-request, s2.receive-request);
}
```

รูปที่ 2.21 ตัวอย่างการนิยามแม่แบบและสไตล์ในภาษาแอคมี [6]

```
exists client, server, rpc |
     component(client) ^
     component(server) ^
     connector(rpc) ^
     attached(client.send-request,rpc.caller) ^
     attached(server.receive-request,rpc.callee)
```

รูปที่ 2.22 ตัวอย่างการกำหนดการวางเงื่อนไขสำหรับการเปิดกรอบความหมายในภาษาแอคมี [6]

จากตัวอย่างภาษาแอคมีข้างต้น จะสังเกตได้ว่าแอทแทชเมนท์สามารถให้เหตุผลด้วยการนำมาตรฐานกลไกโดยหลักทฤษฎีของตรรกะวิทยาระดับแรกทำงานร่วมกันสำหรับกำหนดรูปแบบเชิงรูปนัยในการดำเนินงาน ซึ่งในกรณีนี้จำเป็นต้องมีหนึ่งอย่างที่สามารถเป็นสิ่งประดิษฐ์ที่มีความหมายแทนเครื่องลูกข่ายและเครื่องแม่ข่ายกรณีนี้ระบุเป็นคอมโพเนนท์ อาร์พีซีระบุเป็นการเชื่อมต่อและแอทแทชเมนท์ระบุด้วยสิ่งที่มีการกล่าวถึงไว้แล้วตามลำดับ

2.1.6 เมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ (Extended Backus-Naur Form: EBNF) [12]

Backus ได้กำหนดเครื่องหมาย ซึ่งมีลักษณะที่ง่าย แม่นยำ และมีประสิทธิภาพต่อการอธิบายหลักไวยากรณ์ภาษาโปรแกรมใดๆ ด้วยการนำเครื่องหมายเหล่านี้ช่วยให้นักพัฒนาโปรแกรมหรือตัวแปลภาษาสามารถตัดสินใจได้ว่าโปรแกรมสามารถเขียนถูกต้องตามหลักไวยากรณ์ของภาษา ทั้งในส่วนของกฎการสร้างไวยากรณ์และวรรคตอน โดย Peter Naur เป็นบรรณาธิการการรายงานของภาษาอัลกอล (ALGOL) ซึ่งเป็นเครื่องหมายที่ได้รับการนิยมในการใช้อธิบายในลักษณะไวยากรณ์ที่ถูกต้องของภาษาอัลกอล โดยทั่วไปเรียกว่า Backus-Naur Form (BNF) [14] แต่ได้ปรับปรุงและพัฒนากลายเป็น Extended Backus-Naur Form (EBNF) เพื่อขยายขีดความสามารถทางเครื่องหมายสำหรับสนับสนุนการทำงานร่วมกับลักษณะของไวยากรณ์ที่ซับซ้อน และครอบคลุมในรูปแบบของไวยากรณ์ภาษาเมตตา (Syntactic Meta-languages) มากขึ้น

กฎในเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ [18] ประกอบด้วยสามส่วนคือ ด้านซ้ายมือ (Left-Hand Side: LHS) ด้านขวามือ (Right-Hand Side: RHS) และอักขระ (Character) ซึ่งมีเครื่องหมาย “::=” เป็นส่วนแยกสองข้าง โดยด้านซ้ายมือจะบรรจุคำหนึ่งคำอาจจะคั่นด้วยเครื่องหมายยัติภังค์ ในลักษณะตัวอักษรพิมพ์เล็กเพื่อนิยามเป็นชื่อกฎ ส่วนด้านขวามือเป็นการนิยามหรือจัดหาข้อมูลที่เกี่ยวข้องกับชื่อรวมเอากฎการควบคุมเบื้องต้นตามมาตรฐานภาษาเมตตา (Metalanguage) สำหรับอีบีเอ็นเอฟเข้าไว้ด้วย

ซึ่งในกฎของอีบีเอ็นเอฟสามารถรวมเอาสัญลักษณ์ทั้งหกอักขระ ได้แก่ ::=, |, [,], {, และ } อธิบายทางความหมายของกฎและห้ามนำสัญลักษณ์เหล่านี้ไปใช้ในการตั้งชื่อกฎของอีบีเอ็นเอฟโดยสิ่งที่จะต้องปรากฏอยู่ในด้านขวามือจะประกอบด้วยตัวอักษร (Letter) ตัวเลข (Digit) วรรคตอน วงเล็บ และเครื่องหมายหรืออักขระใดๆ เท่านั้น ตัวอย่างการสร้างกฎอีบีเอ็นเอฟไปใช้อธิบายจำนวนเต็ม โดยด้านขวามือจะแสดงเกี่ยวกับรูปแบบการควบคุมที่ใช้เป็นประโยชน์ได้ในอีบีเอ็นเอฟ ดังรูปที่ 2.23

```
digit      ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
integer    ::= [+|-]digit{digit}
```

รูปที่ 2.23 กฎในรูปแบบเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ [19]

จากรูปที่ 2.23 แสดงให้เห็นว่าตัวเลขนิยามให้เป็นหนึ่งใน 10 ของทั้งหมดจาก 0 ถึง 9 และจำนวนเต็ม โดยกำหนดเป็นสามลำดับการทำงาน คือ เลือกเครื่องหมาย (+ หรือ -) ตามด้วยตัวเลขใดๆ และทำซ้ำกับตัวเลขใดๆ ซึ่งอาจจะไม่มีหรือมีมากกว่าหนึ่งตัว โดยขึ้นอยู่กับการเลือกตัวเลขในนิยามกฎชื่อ digit และด้านขวามือของกฎ integer ได้รวมเอารูปแบบการควบคุม ลำดับ (Sequence) ทางเลือก (Option) ตัวเลือก (Choice) และการวนซ้ำ (Repetition) สามารถอธิบายรูปแบบการควบคุม ดังตารางที่ 2.1

ตารางที่ 2.1 รูปแบบการควบคุมส่วนด้านขวามือสำหรับกฎเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ [12]

รูปแบบ	รายละเอียด
ลำดับ	รายการปรากฏจากซ้ายไปขวา (left-to-right) ซึ่งลำดับจะมีความสำคัญ
ทางเลือก	การเลือกรายการที่แยกด้วยเครื่องหมาย “|” (stroke) โดยหนึ่งในรายการจะถูกเลือก ซึ่งลำดับจะไม่มีความสำคัญ
การเลือก	รายการตัวเลือกจะอยู่ภายในเครื่องหมาย “[“ และ “]” (square brackets) โดยรายการสามารถเลือกหรือไม่เลือก
การทำซ้ำ	รายการตัวเลือกจะอยู่ภายในเครื่องหมาย “{“ และ “}” (curly braces) โดยรายการสามารถเป็นจำนวนศูนย์หรือมากกว่าหนึ่งรายการ

ในการพิสูจน์สัญลักษณ์ที่ระบุว่าเป็นจำนวนเต็ม ซึ่งจะต้องเข้าคู่ได้กับอักขระที่อยู่ในกฎ integer จากรูปที่ 2.23 ตามรูปแบบควบคุม ถ้าสามารถเข้าคู่กันได้จริงจะถูกจำแนกเป็น integer นอกนั้นจะแยกเป็นสัญลักษณ์ที่ผิดกฎตามรายละเอียดที่ได้นิยามไว้ในกฎ integer ดังกล่าว

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 งานวิจัย Acme: An Architecture Description Interchange Language [6]

เสนอโดย David Galan, Robert Monroe และ David Wile ในปี ค.ศ. 2010 ได้นำเสนอการเชื่อมโยงระหว่างภาษา และมุ่งเน้นที่จะพัฒนาเพื่อตอบสนองความต้องการของกลุ่มผู้ที่เกี่ยวข้อง ซึ่งสามารถออกแบบได้สมบูรณ์ในรูปแบบภาษาแอคมีรุ่นเริ่มต้น แต่สองปีถัดมาก็ได้รับกล่าวถึงจากหลายการประชุมของนักวิจัยและผู้ปฏิบัติงาน ที่ซึ่งมีบุคคลได้ให้ข้อเสนอแนะที่สำคัญสำหรับการเป็นผู้เริ่มต้นภาษา โดยนำเสนอภาษาแอคมีในสองรูปแบบ คือ สำรวจภาษาที่มีความสามารถในการศึกษาเกี่ยวกับสถาปัตยกรรมระบบ และศึกษาชุดเครื่องมือเพื่อใช้ในการทดสอบที่จะสนับสนุนการทำงานร่วมกับภาษาแอคมี ในการแลกเปลี่ยนข้อมูลการออกแบบด้านสถาปัตยกรรม ปัจจุบันมีเครื่องมือที่สามารถใช้งานร่วมกับภาษาแอคมี คือ AeSop กับ UniCon, Wright กับ Rapide และเครื่องมือที่จะสนับสนุนการแลกเปลี่ยนที่ซับซ้อนยิ่งขึ้นระหว่าง Rapide, Wright และเอสเอดีแอล (SADL) ซึ่งอยู่ในระหว่างการพัฒนา

ปัจจุบันภาษาแอคมียังเป็นศูนย์กลางสำหรับการขยายความสามารถเครื่องมือที่จัดเตรียมและวิเคราะห์คุณลักษณะภาษาแอคมีสำหรับกลุ่มของเครื่องมือทางสถาปัตยกรรมที่ซึ่งสามารถแปลงไปยังภาษาแอคมี ประกอบกับการพยายามที่จะทำให้เกิดความสอดคล้องกันของลักษณะประจำทั่วไปในกลุ่มผู้พัฒนา ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงนำเสนอในมุมมองของการแลกเปลี่ยนข้อมูลกันระหว่างแบบจำลองต่างประเภท เพื่อการวิจัยและขยายความสามารถของภาษาแอคมีร่วมกับภาษายูเอ็มแอล

2.2.2 งานวิจัย Towards Interoperability of UML Tools for Exchanging High-Fidelity Diagrams [9]

เสนอโดย Shihong Huang, Vashali Gohel และ Sam Hsu ในปี ค.ศ. 2007 นำเสนอกรอบการทำงานการรักษาข้อมูลการออกแบบด้วยภาษายูเอ็มแอลที่ถูกต้อง เพื่อเป็นการรายงานและประเมินการทำงานร่วมกันระหว่างเครื่องมือการสร้างแบบจำลองด้วยการเข้าถึงส่วนเอกสารเอกซ์เอ็ม

ไอที่แทนการออกแบบ อีกทั้งยังป้องกันการสูญเสียข้อมูลเอกสารการออกแบบที่ถูกต้องระหว่างการแลกเปลี่ยนข้อมูลภาษายูเอ็มแอลกับภาษาเอกซ์เอ็มไอการนำเข้า (Import) และการนำออก (Export) รวมถึงความเข้ากันได้ในภาษายูเอ็มแอลต่างรุ่นและสภาพแวดล้อมที่ต่างกัน ซึ่งปัญหาดังกล่าวทำให้ไม่สามารถนำข้อมูลกลับมาใช้งานซ้ำได้ ในงานวิจัยได้ใช้เครื่องมือทดสอบกรอบงานที่นำเสนอสองตัว คือ เครื่องมือ Rational Rose กับ ArgoUML โดยได้ประยุกต์การทำงานของตัวแจงส่วนเอกซ์เอ็มไอ (XMI Parser) ผนวกรวมตัวอ่านเอกซ์เอ็มไอ (XMI Reader) และตัวเขียนเอกซ์เอ็มไอ (XMI Writer) เพื่อให้สามารถระบุและปรับปรุงข้อมูลส่วนที่ไม่ถูกต้องในระหว่างการดำเนินงาน เช่น การระบุหรือปรับปรุงแท็ก (Tags) ที่ไม่สามารถทำงานร่วมกับเครื่องมือปลายทางสร้างเป็นการอธิบายในลักษณะข้อผิดพลาด เป็นต้น

จากงานวิจัยนำเสนอกรอบงาน ซึ่งจำกัดการทำงานร่วมกับเครื่องมือการสร้างแบบจำลองเพียงสองตัวดังกล่าว ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงนำเสนอการแลกเปลี่ยนข้อมูลระหว่างเครื่องมือการสร้างแบบจำลองต่างประเภท เช่น Visual Paradigm เป็นต้น และยังเป็นการศึกษาความสอดคล้องกันในภาษายูเอ็มแอลและภาษาเอกซ์เอ็มไอรุ่นใหม่กว่า

2.2.3 งานวิจัย Bridging the gap between Acme and UML 2.0 for CBD [20]

เสนอโดย Miguel Goulao และ Fernando Brito e Abreu ในปี ค.ศ. 2003 จากการทำงานร่วมกันระหว่างเครื่องมือต่างประเภทยังไม่มีความเสถียรประกอบกับข้อจำกัดในการนำเสนอ Miguel Goulao และ Fernando Brito e Abreu จึงนำเสนอความเป็นไปได้สำหรับการแสดงสถาปัตยกรรมที่ใช้ความสำคัญของภาษาเอดีแอลด้วยภาษาแอคมีกับภาษายูเอ็มแอล รุ่นที่ 2.0 ด้วยการแปลงภาษาเอดีแอลไปเป็นภาษายูเอ็มแอลกำหนดเป็นสองขั้นตอน ดังนี้ ขั้นตอนแรกแปลงภาษาเอดีแอลเป็นแอคมี และขั้นตอนที่สองดำเนินการแปลงภาษาแอคมีเป็นภาษายูเอ็มแอล โดยวิเคราะห์ถึงลักษณะความสอดคล้องและคุณสมบัติตามภาษายูเอ็มแอล ซึ่งวิธีที่นำเสนอได้สร้างจากภาษายูเอ็มแอล รุ่นที่ 2.0 สำหรับแนวคิดด้านสถาปัตยกรรม เมื่อมีการเปรียบเทียบกับรุ่นก่อนหน้าด้วยลักษณะขององค์ประกอบกับประเภทของพอร์ตที่จัดเตรียมไว้สำหรับส่วนต่อประสาน (Interface) ที่ต้องพิสูจน์เพื่อการส่งต่อเป็นสะพานที่ใช้เชื่อมต่อช่องว่างระหว่างสถาปัตยกรรม การออกแบบข้อมูล และเป็นการตรวจติดตามการอธิบายด้วยสถาปัตยกรรม ที่นำเสนอโดยใช้การออกแบบเป็นตัวกลางในการตรวจติดตามความสอดคล้องกันของระบบซอฟต์แวร์

จากงานวิจัยมุ่งเน้นไปยังมุมมองของโครงสร้าง โดยการแปลงข้อมูลดังกล่าวได้นำรหัสโอซีแอล (OCL Code) สำหรับอธิบายภาษาธรรมชาติทั้งโอซีแอลฟังก์ชันและกฎการตรวจสอบความสอดคล้องกันสำหรับการแปลงภาษาแอคมีกับยูเอ็มแอลแล้วนั้น ซึ่งทุกองค์ประกอบในภาษาแอคมีจะแปลงเป็นภาษายูเอ็มแอลส่วนคอมโพเนนท์โดยระบุสเทอริโอไทป์สำหรับแต่ละองค์ประกอบ ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงนำเสนอการแปลงภาษายูเอ็มแอลไปยังภาษาแอคมีโดยนิยามองค์ประกอบตั้งต้นจากงานวิจัย แต่นิยามองค์ประกอบเพิ่มเติม คือ แอทแทชเมนท์ เพื่อช่วยในการระบุปฏิสัมพันธ์ระหว่างคอมโพเนนท์และการเชื่อมต่อ โดยผ่านทางพอร์ตและบทบาทตามลำดับ

2.2.4 งานวิจัย Checking component assembly in Acme: An approach applied on UML 2.0 Components Model [21]

เสนอโดย Mourad Kmimech, Mohamed Tahar Bhiri และ Phillipe Aniorté ในปี ค.ศ. 2009 โดยเสนอแนวทางที่เป็นระเบียบแบบแผนสำหรับแบบจำลองคอมโพเนนท์ยูเอ็มแอล รุ่นที่ 2.0 เพื่อตรวจสอบความสอดคล้อง ซึ่งอธิบายด้วยเมตาโมเดล (Meta-model) เรียกว่า ระดับ เอ็ม2 (M2 Level) โดยใช้รูปแบบสถาปัตยกรรมของภาษาแอคมี ส่วนคอมโพเนนท์แอสแซมบลี (Assembly) ในยูเอ็มแอล รุ่นที่ 2.0 จะอธิบายแทนด้วยแนวคิดในระบบแอคมีในระดับ เอ็ม1 (M1 Level) ซึ่งกล่าวได้ว่า ระดับ เอ็ม1 ต้องมีความสอดคล้องกับระดับ เอ็ม2 และจากการอธิบายด้วยกฎดังกล่าวจะนำไปตรวจสอบโครงสร้างและคุณลักษณะของแบบจำลองคอมโพเนนท์ด้วยเครื่องมือแอคมีสตูดิโอ ที่ซึ่งใช้ตรวจสอบคุณสมบัติและตัวแปรของแบบจำลองภาษาแอคมี จากงานวิจัยยังไม่ได้นำเสนอการพัฒนาระบบที่ใช้กฎการแปลงคอมโพเนนท์ในระดับ เอ็ม2 จัดเตรียมไว้ในรูปแบบภาษาแอคมี เรียกว่า ซียูเอ็มแอลสไตล์ (CUML style) ที่มีความเป็นอัตโนมัติ ดังนั้นในวิทยานิพนธ์นี้จึงนำเสนอการนิยามกฎเพื่อตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ รวมทั้งกลไกตัวแปลภาษาสำหรับการแปลงคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี แต่ก็ได้นำนิยามอินเตอร์เฟสโพรไวด์และรีไควร์มาประยุกต์ใช้ในเครื่องมือตัวแปลภาษา

2.2.5 งานวิจัย UML-Compiler: A Framework for Syntactic and Semantic Verification of UML Diagrams [22]

เสนอโดย Jayeeta Chanda, Ananya Kanjinal และ Sabnam Sengupta ในปี ค.ศ. 2010 จากการนิยามและการกำหนดโครงสร้างที่ยังไม่กำหนดโดยละเอียด ทางกลุ่มวิจัยจึงได้นำเสนอเครื่องมือตัวแปลภาษาสำหรับภาษายูเอ็มแอล ซึ่งนิยามเป็นแบบจำลองในรูปแบบเชิงรูปนัยสำหรับแผนภาพคลาส (Class Diagram) และแผนภาพลำดับการทำงาน (Sequence Diagram) ในมุมมองเชิงโครงสร้างและพฤติกรรม นำเสนอในรูปแบบมาตรฐานของภาษายูเอ็มแอล รุ่นที่ 2.0 ซึ่งกำหนดกฎเกณฑ์การตรวจสอบที่ประกอบด้วยกฎสำหรับระบุความถูกต้องและความสอดคล้องกันในแต่ละไวยากรณ์ โดยในงานวิจัยดังกล่าวจะมุ่งเน้นไปที่การตรวจสอบหลักไวยากรณ์เป็นหลัก

แต่ในงานวิจัยไม่ได้เน้นส่วนการแปลงข้อมูลเอกซ์เอ็มไอเป็นข้อความกับหลักภาษา ซึ่งนิยามไว้ในงานวิจัยประกอบกับแบบจำลองที่ตรวจสอบยังไม่สามารถนิยามให้สนับสนุนการทำงานร่วมกับแผนภาพยูเอ็มแอลประเภทอื่น เช่น แผนภาพคอมโพเนนท์ เป็นต้น ซึ่งในงานวิจัยออกแบบเพื่อสนับสนุนแผนภาพคลาสกับแผนภาพลำดับการทำงานเท่านั้น ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงนำเสนอนิยามและนิยามเป็นแบบจำลองในรูปแบบเชิงรูปนัยสำหรับแผนภาพคอมโพเนนท์เพิ่มเติม โดยประยุกต์นิยามจากแผนภาพคลาสและแผนภาพลำดับการทำงานเป็นนิยามตั้งต้น

จากการศึกษาวรรณกรรมและงานวิจัยที่เกี่ยวข้องดังกล่าว ยังไม่พบเครื่องมือที่ช่วยอำนวยความสะดวกต่อนักออกแบบและพัฒนาซอฟต์แวร์ ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงนำเสนอการสร้างเครื่องมือที่เป็นอัตโนมัติ ในการแปลงแบบจำลองเชิงโครงสร้างไปยังมุมมองทางสถาปัตยกรรมด้วยรูปแบบของตัวแปลภาษาที่มีความเป็นอัตโนมัติ และเครื่องมือต้องมีคุณสมบัติตามขอบเขตของงานวิจัยเป็นอย่างน้อย

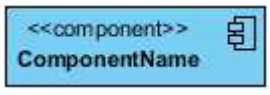
บทที่ 3
การวิเคราะห์และนิยามกฎสำหรับตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล

สำหรับการวิเคราะห์และออกแบบกฎสำหรับตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล อธิบายแยกเป็นสี่ส่วนโดยส่วนแรกกล่าวถึงการนิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ส่วนที่สองการตรวจสอบภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนที่สามกล่าวถึงการนิยามกฎตรวจสอบคำศพท์ภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนที่สี่กล่าวถึงการนิยามกฎสำหรับตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ และส่วนที่ห้ากล่าวถึงการนิยามกฎการแปลงข้อมูลแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี ซึ่งอธิบายรายละเอียด ดังนี้

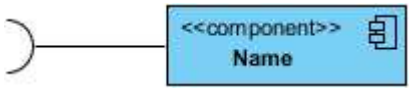
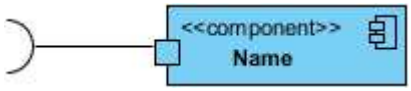
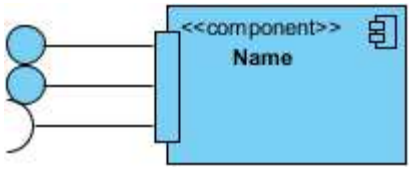
3.1 การนิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์

สำหรับการนิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ [24, 26] จากนิยามของคอมโพเนนท์หนึ่งคอมโพเนนท์และคอมโพเนนท์แพ็กเกจโดยคอมโพเนนท์แพ็กเกจหมายถึงการนำคอมโพเนนท์ใดๆ หนึ่งคอมโพเนนท์หลายๆ คอมโพเนนท์มาเชื่อมต่อกันเป็นแผนภาพคอมโพเนนท์ ซึ่งต้องมีการเชื่อมต่อกันได้หลายประเภทตั้งแต่การใช้ประเภทขึ้นต่อกัน (Dependency) หรือการต่อผ่านอินเตอร์เฟส ซึ่งผ่านทางการอินเตอร์เฟสโพรไวด์ (Provided) และอินเตอร์เฟสรีไควร์ (Required) แล้วนั้น ในวิทยานิพนธ์ฉบับนี้ได้นิยามสัญลักษณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ [26] ดังตารางที่ 3.1 และ 3.2

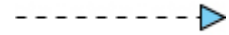
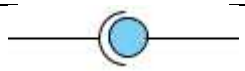
ตารางที่ 3.1 สัญลักษณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์รวมถึงโครงสร้างแผนภาพ [26]

ประเภทโหนด (Node Type)	สัญลักษณ์	คำอธิบาย
คอมโพเนนท์		คอมโพเนนท์ระบุสเทอริโอไทป์ <<component>> และมีชื่อคอมโพเนนท์ระบุไว้
คอมโพเนนท์อิมพลีเมนท์อินเตอร์เฟส (Component implements Interface)		คอมโพเนนท์กับอินเตอร์เฟสที่ซึ่งเตรียมไว้ให้บริการแก่คอมโพเนนท์หรือคลาสอื่น

ตารางที่ 3.1 สัญลักษณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์รวมถึงโครงสร้างแผนภาพ [26] (ต่อ)

ประเภทโหนด (Node Type)	**สัญลักษณ์**	**คำอธิบาย**
คอมโพเนนท์มีพอร์ตประเภทโพรไวด์ *(ระบุประเภทตามอินเตอร์เฟส)*		คอมโพเนนท์ที่ซึ่งมีพอร์ตประเภทโพรไวด์ ระบุตามประเภทอินเตอร์เฟสและคอมโพเนนท์สามารถใช้อินเตอร์เฟสที่แนบมากับพอร์ตได้
คอมโพเนนท์ยูสอินเตอร์เฟส (Component uses Interface)		คอมโพเนนท์กับอินเตอร์เฟสที่ซึ่งต้องการไปใช้บริการตลอดจนฟังก์ชันจากคอมโพเนนท์หรือคลาสอื่นที่เตรียมไว้ให้บริการ
คอมโพเนนท์มีพอร์ตประเภทรีไควร์ *(ระบุประเภทตามอินเตอร์เฟส)*		คอมโพเนนท์ที่ซึ่งมีพอร์ตประเภทรีไควร์ ระบุตามประเภทอินเตอร์เฟสและคอมโพเนนท์สามารถใช้อินเตอร์เฟสที่แนบมากับพอร์ตได้
คอมโพเนนท์มีพอร์ตมากกว่าหนึ่งพอร์ต *(ระบุอินเตอร์เฟสโพรไวด์และรีไควร์)*		คอมโพเนนท์ที่ซึ่งมีพอร์ตประเภทโพรไวด์และรีไควร์ระบุตามประเภทอินเตอร์เฟสและคอมโพเนนท์สามารถใช้อินเตอร์เฟสที่แนบมากับพอร์ตได้มากกว่าหนึ่งอินเตอร์เฟส

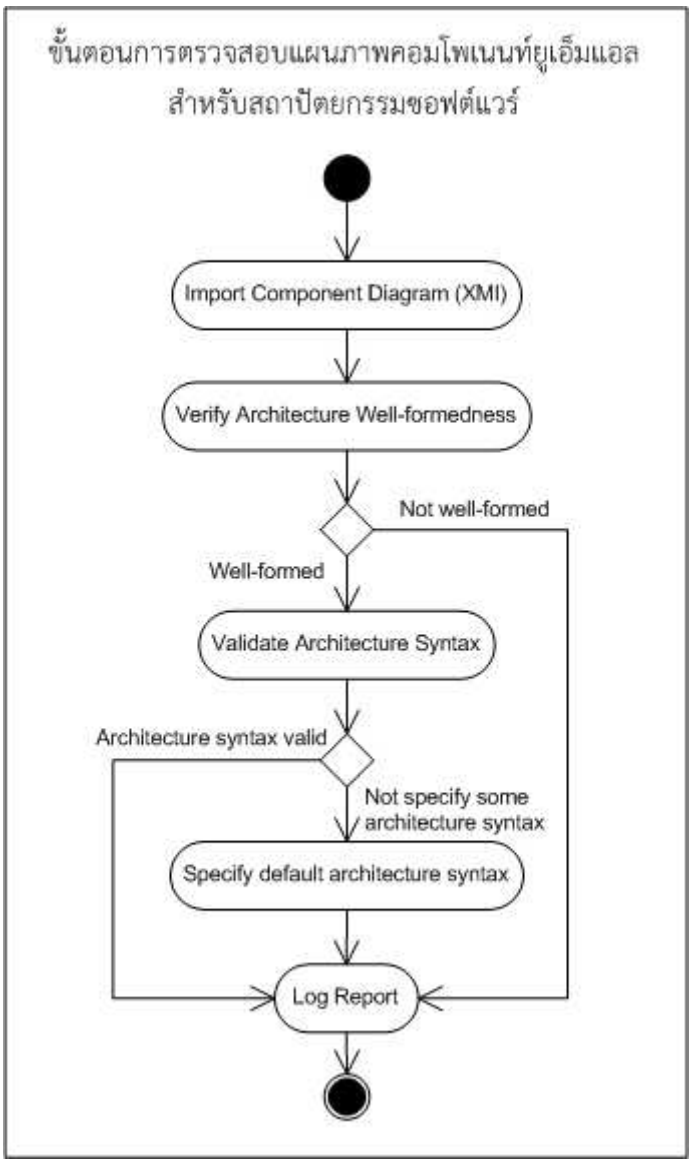
ตารางที่ 3.2 เส้นทาง (Path) แผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ รวมถึงโครงสร้างแผนภาพและโครงสร้างประกอบ (Composite) [26]

ประเภทเส้นทาง (Path Type)	**สัญลักษณ์**	**คำอธิบาย**
Component realization		เส้นทางประเภทเรียลไลเซชัน
Assembly connector		เส้นทางประเภทแอสเซมบลี
Delegate connector		เส้นทางประเภทดีลีเกท (Delegate)

เพื่อให้การตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์มีความสอดคล้องกับนิยามดังตารางที่ 3.1 และ 3.2 แล้วนั้น ในเครื่องมือแปลภาษาจึงวิเคราะห์และนิยามกฎตั้งต้น ดังนี้

1) **กฎข้อที่หนึ่ง** แบบจำลองสถาปัตยกรรมซอฟต์แวร์ ประกอบด้วยแผนภาพคอมโพเนนท์ยูเอ็มแอลหนึ่งแผนภาพเป็นอย่างน้อย
2) **กฎข้อที่สอง** แผนภาพคอมโพเนนท์ต้องประกอบด้วยคอมโพเนนท์ตั้งแต่สองคอมโพเนนท์ขึ้นไปและประกอบด้วยการเชื่อมต่อหนึ่งการเชื่อมต่อเป็นอย่างน้อย
3) **กฎข้อที่สาม** คอมโพเนนท์ใดๆ อาจจะประกอบด้วยคอมโพเนนท์ย่อยหนึ่งคอมโพเนนท์ย่อยเป็นอย่างน้อย
4) **กฎข้อที่สี่** คอมโพเนนท์ใดๆ อาจจะประกอบด้วยพอร์ตตั้งแต่หนึ่งพอร์ตขึ้นไป โดยระบุไปกับคอมโพเนนท์ ถ้าไม่ระบุพอร์ตใดๆ สำหรับคอมโพเนนท์จะระบุเป็นพอร์ตพื้นฐาน
5) **กฎข้อที่ห้า** การเชื่อมต่อประกอบด้วยการเชื่อมต่อหนึ่งการเชื่อมต่อเป็นอย่างน้อย สำหรับแผนภาพคอมโพเนนท์ใดๆ ที่ซึ่งแสดงปฏิสัมพันธ์ระหว่างคอมโพเนนท์ด้วยประเภทการเชื่อมต่อ ดังนี้ การขึ้นต่อกัน แอสโซซิเอชัน เรียลไลเซชันและยูซเสจ
6) **กฎข้อที่หก** อินเตอร์เฟสเป็นส่วนต่อประสานระหว่างการเชื่อมต่อประเภทเรียลไลเซชันและยูซเสจ เรียกว่า การเชื่อมต่อแบบแอสแซมบลี ซึ่งขึ้นกับ**กฎข้อที่ห้า**

สำหรับในเครื่องมือตัวแปลภาษานั้น แสดงแผนภาพกิจกรรมขั้นตอนการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ดังรูปที่ 3.1

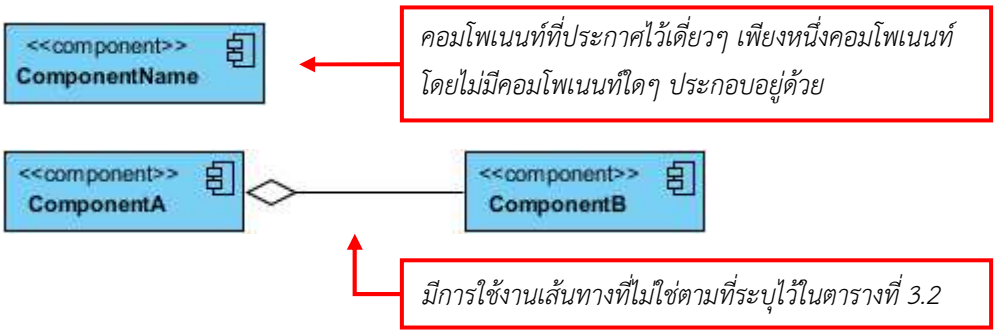


รูปที่ 3.1 แผนภาพกิจกรรมขั้นตอนการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล
สำหรับสถาปัตยกรรมซอฟต์แวร์

จากรูปที่ 3.1 แผนภาพกิจกรรมซึ่งระบุการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ที่ซึ่งอธิบายรายละเอียด ดังนี้

1) นำเข้าแผนภาพคอมโพเนนท์ในรูปแบบภาษาเอกซ์เอ็มไอ (Import Component Diagram; XMI) ผ่านเครื่องมือตัวแปลภาษา ที่ซึ่งนำออกจากเครื่องมือ Visual Paradigm
2) ตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ (Verify Architecture Well-formedness) ถ้าแผนภาพคอมโพเนนท์ยูเอ็มแอลดังกล่าวถูกตรวจสอบว่าเป็นแผนภาพที่ซึ่งออกแบบสำหรับสถาปัตยกรรมซอฟต์แวร์ จะ

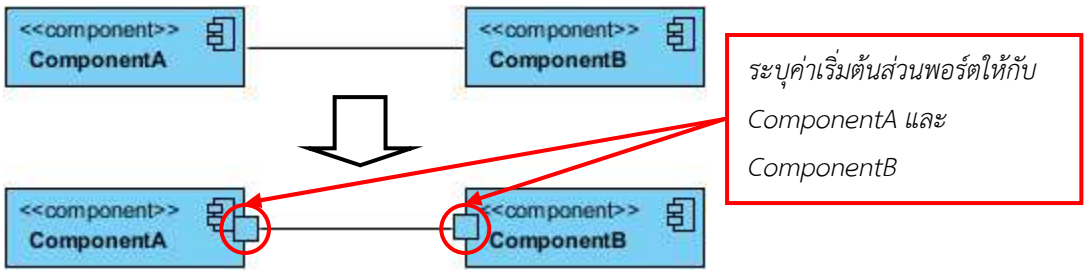
ดำเนินการต่อในข้อ 3) แต่ถ้าตรวจสอบพบว่าแผนภาพคอมโพเนนท์ยูเอ็มแอลดังกล่าวไม่ถูกออกแบบสำหรับสถาปัตยกรรมซอฟต์แวร์ ตัวอย่างเช่น ตรวจสอบพบว่าโครงสร้างไวยากรณ์มีข้อผิดพลาด (Syntax Error) ซึ่งข้อมูลนำเข้าเป็นแผนภาพประเภทอื่นที่ไม่ได้ระบุเป็นแผนภาพคอมโพเนนท์ยูเอ็มแอล เป็นคอมโพเนนท์ที่ประกาศไว้เดี่ยวๆ เพียงหนึ่งคอมโพเนนท์โดยไม่มีคอมโพเนนท์ใดๆ ประกอบอยู่ด้วย และเป็นคอมโพเนนท์ที่มีการใช้งานเส้นทางที่ไม่ใช่ตามที่ระบุไว้ในตารางที่ 3.6 เป็นต้น เครื่องมือตัวแปลภาษาจะรายงานผลการดำเนินการ (Log Report) และจบการทำงาน ดังรูปที่ 3.2



รูปที่ 3.2 โครงสร้างไวยากรณ์ที่มีข้อผิดพลาด ในแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์

3) ตรวจสอบโครงสร้างไวยากรณ์สำหรับสถาปัตยกรรมซอฟต์แวร์ (Validate Architecture Syntax) ซึ่งจะดำเนินการตรวจสอบโครงสร้างไวยากรณ์เขียนถูกต้องตามเครื่องหมายทางสถาปัตยกรรมซอฟต์แวร์ทั้งแผนภาพ ซึ่งเครื่องมือตัวแปลภาษาจะรายงานผลและจบการดำเนินงาน แต่ถ้าไม่ได้ระบุส่วนที่แทนด้วยสถาปัตยกรรมซอฟต์แวร์ทั้งแผนภาพดังอธิบายต่อไป

4) ระบุส่วนที่แทนสถาปัตยกรรมซอฟต์แวร์ด้วยค่าเริ่มต้น (Specify Default Architecture Syntax) ซึ่งในเครื่องมือตัวแปลภาษาสามารถระบุส่วนที่แทนการดำเนินการดังกล่าวเป็นค่าเริ่มต้น (Default) และแสดงข้อความเตือน (Warning Message) ที่แสดงถึงเส้นทางสำหรับในแต่ละคอมโพเนนท์ใดๆ ในกรณีที่ไม่ได้ระบุส่วนที่แทนด้วยสถาปัตยกรรมซอฟต์แวร์ทั้งแผนภาพและแผนภาพถูกออกแบบสำหรับสถาปัตยกรรมซอฟต์แวร์ ซึ่งเครื่องมือตัวแปลภาษาจะรายงานผลการดำเนินการและจบการดำเนินงาน ดังรูปที่ 3.3




```
WARNING:   The delegate connector between ComponentA and
           ComponentB doesn't fill port notation.
```

รูปที่ 3.3 การระบุค่าเริ่มต้นและข้อความเตือน ในในแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์

5) รายงานผลการดำเนินการ (Log Report) ในเครื่องมือตัวแปลภาษาจะรายงานผลการดำเนินการจากการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ตรวจสอบโครงสร้างไวยากรณ์สำหรับสถาปัตยกรรมซอฟต์แวร์และการแปลภาษาในรูปแบบเอกสารข้อความ

3.2 การตรวจสอบภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล

สำหรับข้อมูลนำเข้าภาษาเอกซ์เอ็มไอนั้น ในเครื่องมือตัวแปลภาษากำหนดให้สามารถตรวจสอบแท็ก (Tags) ที่เป็นส่วนประกอบของแผนภาพคอมโพเนนท์ยูเอ็มแอล ดังนี้ uml:Diagram, uml:Model, ownedMember ที่ซึ่งประกอบด้วยประเภท ดังนี้ uml:Component, uml:Interface, uml:Association, uml:Realization และ uml:Usage เป็นต้น แท็ก ownedEnd ที่ซึ่งประกอบด้วยประเภท uml:Property ที่เป็นส่วนระบุคุณลักษณะแท็ก ownedMember ประเภท uml:Association แท็ก ownedPort ที่ซึ่งประกอบด้วยประเภท uml:Port และแท็ก generalization ที่ซึ่งประกอบด้วยประเภท uml:Generalization ดังแสดงรายละเอียดเอกซ์เอ็มไอแท็ก (XMI Tags) ลักษณะประจำ (Attribute) และคำอธิบาย (Description) ดังตารางที่ 3.3

ตารางที่ 3.3 เอกซ์เอ็มไอแท็กสำหรับส่วนประกอบแผนภาพคอมโพเนนท์ยูเอ็มแอล [4]

เอกซ์เอ็มไอแท็ก	ลักษณะประจำ	คำอธิบาย
uml:Diagram	diagramType	ประเภทแผนภาพ
	documentation	คำอธิบายแผนภาพ
	name	ชื่อแผนภาพ
	toolName	เครื่องมือสำหรับนำออกแผนภาพ
	xmi:id	รหัสแผนภาพ

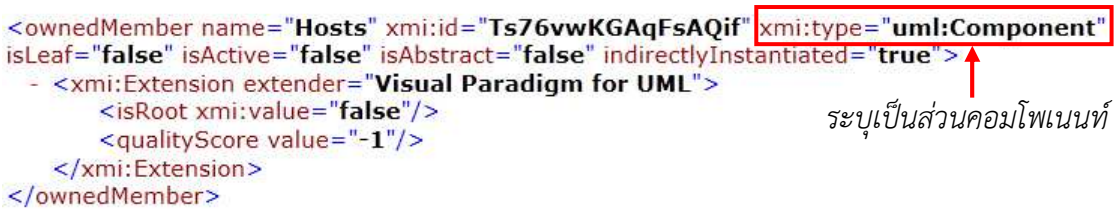
ตารางที่ 3.3 เอกซ์เอ็มไอแท็กสำหรับส่วนประกอบแผนภาพคอมโพเนนท์ยูเอ็มแอล [4] (ต่อ)

เอกซ์เอ็มไอแท็ก	ลักษณะประจำ	คำอธิบาย
uml:Model	name	ชื่อแบบจำลอง
	xmi:id	รหัสแบบจำลอง
ownedMember	indirectlyInstantiated [T1]	คุณลักษณะองค์ประกอบแบบจำลอง (Model Element Property) แบบ indirectlyInstantiated
	client [T4, T5, T6]	รหัสคอมโพเนนท์หรือพอร์ตที่ซึ่งทำหน้าที่เป็น Client
	realizingClassifier [T4]	รหัสคอมโพเนนท์ พอร์ตหรืออินเตอร์เฟส ที่ซึ่งทำหน้าที่เป็น realizingClassifier
	supplier [T4, T5, T6]	รหัสคอมโพเนนท์ พอร์ตหรืออินเตอร์เฟส ที่ซึ่งทำหน้าที่เป็น Supplier
	isAbstract [T1, T2, T3]	คุณลักษณะองค์ประกอบแบบจำลองแบบ isAbstract
	isDerived [T3]	คุณลักษณะองค์ประกอบแบบจำลองแบบ isDerived
	isActive [T1, T2]	คุณลักษณะองค์ประกอบแบบจำลองแบบ isActive
	isLeaf [T1, T2, T3]	คุณลักษณะองค์ประกอบแบบจำลองแบบ isLeaf
	visibility [T2]	ระบุค่า Visibility ดังนี้ Unspecified, private, protected, package และ public
	name [T1, T2, T3, T4, T5, T6]	ชื่อ [T1, T2, T3, T4, T6]
	xmi:id [T1, T2, T3, T4, T5, T6]	รหัส [T1, T2, T3, T4, T5, T6]
	xmi:type [T1, T2, T3, T4, T5, T6]	ประเภทเอกซ์เอ็มไอ ซึ่งประกอบด้วย T1 - uml:Component T2 - uml:Interface T3 - uml:Association T4 - uml:Realization T5 - uml:Usage T6 – uml:Dependency
ownedPort	aggregation	ระบุค่า Aggregation ดังนี้ none, shared และ composite

ตารางที่ 3.3 เอกซ์เอ็มไอแท็กสำหรับส่วนประกอบแผนภาพคอมโพเนนท์ยูเอ็มแอล [4] (ต่อ)

เอกซ์เอ็มไอแท็ก	ลักษณะประจำ	คำอธิบาย
ownedPort	isBehavior	คุณลักษณะองค์ประกอบแบบจำลองแบบ isBehavior
	isDerived	คุณลักษณะองค์ประกอบแบบจำลองแบบ isDerived
	isDerivedUnion	คุณลักษณะองค์ประกอบแบบจำลองแบบ isDerivedUnion
	isLeaf	คุณลักษณะองค์ประกอบแบบจำลองแบบ isLeaf
	isReadOnly	คุณลักษณะองค์ประกอบแบบจำลองแบบ isReadOnly
	isService	คุณลักษณะองค์ประกอบแบบจำลองแบบ isService
	isStatic	คุณลักษณะองค์ประกอบแบบจำลองแบบ isStatic
	name	ชื่อพอร์ต (Port)
	xmi:id	รหัสพอร์ต
	xmi:type	ระบุประเภทเอกซ์เอ็มไอเป็น uml:Port
ownedEnd	aggregation	ระบุค่า Aggregation ดังนี้ none, aggregate และ composite
	association	รหัสเส้นความสัมพันธ์แอสโซซิเอชัน (Association)
ownedEnd	isDerived	คุณลักษณะองค์ประกอบแบบจำลองแบบ isDerived
	isNavigable	คุณลักษณะองค์ประกอบแบบจำลองแบบ isNavigable
	type	รหัสคอมโพเนนท์หรือพอร์ตที่เป็นจุดเริ่มและสิ้นสุดสำหรับความสัมพันธ์แบบแอสโซซิเอชัน
	xmi:id	รหัสแท็ก
	xmi:type	ระบุประเภทเอกซ์เอ็มไอเป็น uml:Property
generalization	general	รหัสคอมโพเนนท์หรือพอร์ต ซึ่งมีความสัมพันธ์แบบเจนเนอรัลไลเซชัน (Generalization)
	xmi:id	รหัสเส้นความสัมพันธ์แบบเจนเนอรัลไลเซชัน
	xmi:type	ระบุประเภทเอกซ์เอ็มไอเป็น uml:Generalization

จากตารางที่ 3.3 สามารถแสดงตัวอย่างภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลส่วนคอมโพเนนท์ ดังรูปที่ 3.4



รูปที่ 3.4 ตัวอย่างภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล

จากรูปที่ 3.4 แสดงตัวอย่างภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล ซึ่งประกอบด้วยแท็ก ownedMember ซึ่งระบุเป็นส่วนคอมโพเนนท์ ประกอบด้วยคุณลักษณะองค์ประกอบแบบจำลองดังนี้ indirectlyInstantiated มีค่าเป็น “true” isAbstract มีค่าเป็น “false” isActive มีค่าเป็น “false” isLeaf มีค่าเป็น “false” รหัสคอมโพเนนท์มีค่าเป็น “Ts76vwKGAqFsAQif” ชื่อคอมโพเนนท์มีค่าเป็น “Hosts”

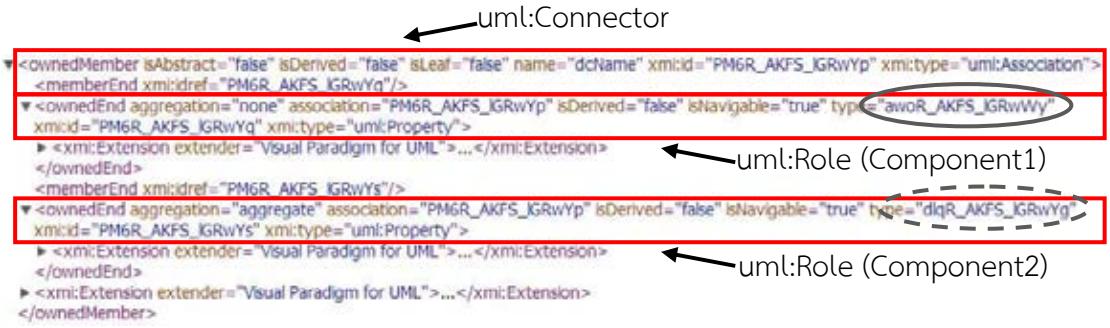
จากการตรวจสอบภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล สำหรับเครื่องมือตัวแปลภาษานั้นจะดำเนินการสร้างสายอักขระเพิ่มเติม ดังนี้ uml:Connector, uml:Role, uml:Attachment และ uml:Representation เพื่อระบุเป็นคุณสมบัติเฉพาะเพิ่มเติมและนำไปใช้ประกอบการประมวลผลในเครื่องมือตัวแปลภาษา สำหรับการนิยามกฎการตรวจสอบภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล อธิบายรายละเอียด ดังตารางที่ 3.4

ตารางที่ 3.4 สายอักขระเพิ่มเติมหลังการตรวจสอบภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล

สายอักขระ	คำอธิบาย
uml:Connector	แสดงส่วนประกอบประเภทการเชื่อมต่อ
uml:Role	แสดงส่วนประกอบบทบาทสำหรับเส้นเชื่อมต่อ
uml:Attachment	แสดงส่วนประกอบแอทแทชเมนท์ สำหรับคอมโพเนนท์ใดๆ ที่ส่งออกไปยังเส้นเชื่อมต่อใดๆ
uml:Representation	แสดงส่วนประกอบเรพพรีเซนเทชัน สำหรับคอมโพเนนท์ใดๆ ที่ประกอบด้วยคอมโพเนนท์ย่อย (Subcomponent)

จากตารางที่ 3.4 สามารถแสดงการระบุสายอักระเพิ่มเติมหลังการตรวจสอบภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนคอมโพเนนท์ที่มีความสัมพันธ์แบบแอสโซซิเอชัน ดังรูปที่ 3.5

(ก) ส่วนคอมโพเนนท์



(ข) ส่วนการเชื่อมต่อและบทบาท

รูปที่ 3.5 การระบุสายอักระเพิ่มเติมหลังการตรวจสอบภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล

จากรูปที่ 3.5 โครงสร้างเอกซ์เอ็มไอประกอบด้วยคอมโพเนนท์สองคอมโพเนนท์ ดังนี้ Component1 และ Component2 โดยในแต่ละคอมโพเนนท์จะประกอบด้วยพอร์ตที่เป็นเริ่มต้น (Default Port) ซึ่งเชื่อมต่อกันด้วยความสัมพันธ์แบบแอสโซซิเอชัน โดยใน Component1 มีบทบาทเป็นแอสโซซิเอชันประเภท “none” ส่วน Component2 มีบทบาทเป็นแอสโซซิเอชันประเภท “aggregate” ตามลำดับ

3.3 การนิยามกฎการตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล

ในส่วนนี้จะอธิบายเกี่ยวกับการสแกนข้อมูลนำเข้าภาษาเอกซ์เอ็มไอ ซึ่งจะถูกแปลงเป็นลำดับของเครื่องหมายที่แทนด้วยแผนภาพคอมโพเนนท์ยูเอ็มแอล ที่ซึ่งวิเคราะห์คำศัพท์จากแบบรูปตามมาตรฐานภาษายูเอ็มแอล รุ่นที่ 2.0 [1] สำหรับการตรวจสอบคำศัพท์นั้นจะนิยามด้วยรูปแบบเรกูลาร์เอกซ์เพรสชัน

การกำหนดหลักไวยากรณ์สำหรับกลุ่มของคำศัพท์จะนิยามตาม [22] ซึ่งแทนด้วย (S, N, T, P) โดยที่ S แทนเครื่องหมายเริ่ม N แทนนอลเทอร์มินอล T แทนคำศัพท์หรือเทอร์มินอลและ P แทนกฎไวยากรณ์ (Production Rules) สำหรับการสร้างกฎไวยากรณ์ P จะถูกสร้างจากเครื่องหมายอื่นที่เป็นไวยากรณ์อิสระ N ดังรูปที่ 3.6

T = { string, boolean, association, realization, usage, dependency, indirectlyInstantiated, isAbstract, isActive, isLeaf, isBehavior, isDerived, isDerivedUnion, isReadOnly, isService, isStatic, isNavigable, Unspecified, public, package, protected, private, none, aggregate, composite, shared, composited }

P: S → umldiagram

umldiagram → umlcomponent+ umlconnector+

umlcomponent → name component_property+ id umlport*

component_property → visibility model_element_property

visibility → **Unspecified | public | package | protected | private**

model_element_property → **indirectlyInstantiated | isAbstract | isActive | isLeaf**

umlport → name visibility port_aggregation model_element_property id port_connector_type*

port_aggregation → **None | shared | composited**

indirectlyInstantiated → **boolean**

isAbstract → **boolean**

isActive → **boolean**

isLeaf → **boolean**

isBehavior → **boolean**

isDerived → **boolean**

isDerivedUnion → **boolean**

isLeaf → **boolean**

isReadOnly → **boolean**

isService → **boolean**

isStatic → **boolean**

isNavigable → **boolean**

port_connector_type → **association | realization | usage**

umlconnector → **association | realization | usage | dependency**

รูปที่ 3.6 การตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลในรูปแบบเรกูลาร์เอกซ์เพรสชัน [22]

realization → client name realizing_classifier supplier id
umlinterface → model_element_property name visibility id
usage → client name supplier id
association → association_id name visibility model_element_property association_property*
association_property → association_aggregation association_id model_element_property property_type id
model_element_property → **isBehavior | isDerived | isDerivedUnion | isLeaf | isReadOnly | isService | isStatic | isNavigable**
association_aggregation → **None | aggregate | composite**
client → **string**
supplier → **string**
realizing_classifier → **string**
association_id → **string**
property_type → **string**
name → **string**
id → **string**
string → **[A-Za-z0-9][A-Za-z0-9.-_]+**
boolean → **true | false**

รูปที่ 3.6 การตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลในรูปแบบเรกูลาร์เอกซ์เพรสชัน [22] (ต่อ)

จากรูปที่ 3.6 สามารถอธิบายเกี่ยวกับกฎการตรวจสอบในรูปแบบเรกูลาร์เอกซ์เพรสชันสำหรับการตรวจสอบคำศัพท์หรือเครื่องหมาย การคืนค่าโทเค็นและคำอธิบาย ดังตารางที่ 3.5

ตารางที่ 3.5 การตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชัน

คำศัพท์/เครื่องหมาย	**การคืนค่าโทเค็น**	**คำอธิบาย**
name	NAME	ค้นพบคำศัพท์ “name” กฎการตรวจสอบจะคืนค่าโทเค็น NAME
connector	CONNECTOR	ค้นพบคำศัพท์ “connector” กฎการตรวจสอบจะคืนค่าโทเค็น CONNECTOR
xmi:id	ID	ค้นพบคำศัพท์ “xmi:id” กฎการตรวจสอบจะคืนค่าโทเค็น ID

ตารางที่ 3.5 การตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชัน (ต่อ)

คำศัพท์/เครื่องหมาย	การคืนค่าโทเค็น	คำอธิบาย
=	SEQ	ค้นพบเครื่องหมาย “ = ” กฎการตรวจสอบจะคืนค่าโทเค็น SEQ
“	QUOTE	ค้นพบเครื่องหมาย “ ” ” กฎการตรวจสอบจะคืนค่าโทเค็น QUOTE
true	BOOLEANT	ค้นพบคำศัพท์ “true” กฎการตรวจสอบจะคืนค่าโทเค็น BOOLEANT
false	BOOLEANF	ค้นพบคำศัพท์ “false” กฎการตรวจสอบจะคืนค่าโทเค็น BOOLEANF
indirectlyInstantiated	INDINS	ค้นพบคำศัพท์ “indirectlyInstantiated” กฎการตรวจสอบจะคืนค่าโทเค็น INDINS
general	GENERAL	ค้นพบคำศัพท์ “general” กฎการตรวจสอบจะคืนค่าโทเค็น GENERAL
isAbstract	ISAB	ค้นพบคำศัพท์ “isAbstract” กฎการตรวจสอบจะคืนค่าโทเค็น ISAB
isActive	ISAC	ค้นพบคำศัพท์ “isActive” กฎการตรวจสอบจะคืนค่าโทเค็น ISAC
isLeaf	ISLE	ค้นพบคำศัพท์ “isLeaf” กฎการตรวจสอบจะคืนค่าโทเค็น ISLE
isDerived	ISDE	ค้นพบคำศัพท์ “isDerived” กฎการตรวจสอบจะคืนค่าโทเค็น ISDE
realizingClassifier	REALCLASS	ค้นพบคำศัพท์ “realizingClassifier” กฎการตรวจสอบจะคืนค่าโทเค็น REALCLASS
supplier	SUPPLIER	ค้นพบคำศัพท์ “supplier” กฎการตรวจสอบจะคืนค่าโทเค็น SUPPLIER
client	CLIENT	ค้นพบคำศัพท์ “client” กฎการตรวจสอบจะคืนค่าโทเค็น CLIENT
Unspecified	VISUNS	ค้นพบคำศัพท์ “Unspecified” กฎการตรวจสอบจะคืนค่าโทเค็น VISUNS
public	VISPUB	ค้นพบคำศัพท์ “public” กฎการตรวจสอบจะคืนค่าโทเค็น VISPUB
package	VISPAC	ค้นพบคำศัพท์ “package” กฎการตรวจสอบจะคืนค่าโทเค็น VISPAC

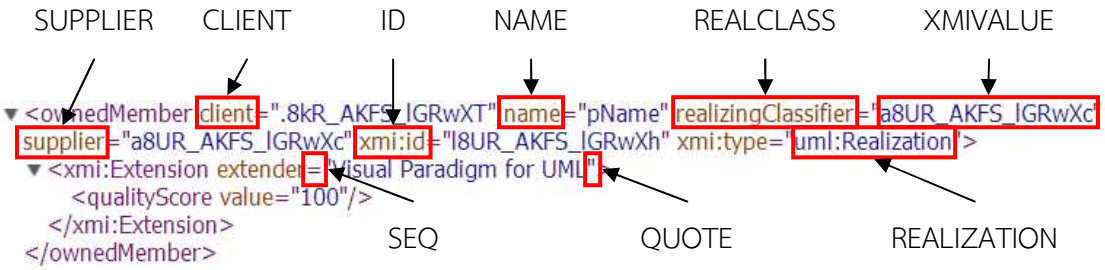
ตารางที่ 3.5 การตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอ สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล ด้วยเรกูลาร์เอกซ์เพรสชัน (ต่อ)

คำศัพท์/เครื่องหมาย	การคืนค่าโทเค็น	คำอธิบาย
protected	VISPRO	ค้นพบคำศัพท์ “protected” กฎการตรวจสอบจะคืนค่าโทเค็น VISPRO
private	VISPRI	ค้นพบคำศัพท์ “private” กฎการตรวจสอบจะคืนค่าโทเค็น VISPRI
visibility	VISIBILITY	ค้นพบคำศัพท์ “visibility” กฎการตรวจสอบจะคืนค่าโทเค็น VISIBILITY
isFrom	ISFROM	ค้นพบคำศัพท์ “isFrom” กฎการตรวจสอบจะคืนค่าโทเค็น ISFROM
isTo	ISTO	ค้นพบคำศัพท์ “isTo” กฎการตรวจสอบจะคืนค่าโทเค็น ISTO
portType	PORTTYPE	ค้นพบคำศัพท์ “portType” กฎการตรวจสอบจะคืนค่าโทเค็น PORTTYPE
isPort	ISPORT	ค้นพบคำศัพท์ “isPort” กฎการตรวจสอบจะคืนค่าโทเค็น ISPORT
umldiagram	UMLDIAGRAM	ค้นพบคำศัพท์ “uml:Diagram” กฎการตรวจสอบจะคืนค่าโทเค็น UMLDIAGRAM
umlcomponent	UMLCOMPONENT	ค้นพบคำศัพท์ “uml:Component” กฎการตรวจสอบจะคืนค่าโทเค็น UMLCOMPONENT
umlconnector	UMLCONNECTOR	ค้นพบคำศัพท์ “uml:Connector” กฎการตรวจสอบจะคืนค่าโทเค็น UMLCONNECTOR
umlrole	UMLROLE	ค้นพบคำศัพท์ “uml:Role” กฎการตรวจสอบจะคืนค่าโทเค็น UMLROLE
generalization	GENERALIZATION	ค้นพบคำศัพท์ “uml:Generalization” กฎการตรวจสอบจะคืนค่าโทเค็น GENERALIZATION
association	ASSOCIATION	ค้นพบคำศัพท์ “uml:Association” กฎการตรวจสอบจะคืนค่าโทเค็น ASSOCIATION
usage	USAGE	ค้นพบคำศัพท์ “uml:Usage” กฎการตรวจสอบจะคืนค่าโทเค็น USAGE

ตารางที่ 3.5 การตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอ สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชัน (ต่อ)

คำศัพท์/เครื่องหมาย	การคืนค่าโทเค็น	คำอธิบาย
realization	REALIZATION	ค้นพบคำศัพท์ “uml:Realization” กฎการตรวจสอบจะคืนค่าโทเค็น REALIZATION
umlinterface	UMLINTERFACE	ค้นพบคำศัพท์ “uml:Interface” กฎการตรวจสอบจะคืนค่าโทเค็น UMLINTERFACE
umlport	UMLPORT	ค้นพบคำศัพท์ “uml:Port” กฎการตรวจสอบจะคืนค่าโทเค็น UMLPORT
umlattachment	UMLATTACHMENT	ค้นพบคำศัพท์ “uml:Attachment” กฎการตรวจสอบจะคืนค่าโทเค็น UMLATTACHMENT
umlrepresentation	UMLREPRESENTATION	ค้นพบคำศัพท์ “uml:Representation” กฎการตรวจสอบจะคืนค่าโทเค็น UMLREPRESENTATION
string	XMIVALUE	ค้นพบคำศัพท์ “[A-Za-z0-9][A-Za-z0-9.-_]+” ที่ซึ่งประกอบด้วย ตัวอักษรภาษาอังกฤษ A-Z (ทั้งตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่), ตัวเลข 0-9 เครื่องหมาย “.” “-” “_” และ “ ” อย่างน้อยสองตัวขึ้นไป โดยกฎการตรวจสอบจะคืนค่าโทเค็น XMIVALUE
<<EOF>>	EOF	ค้นพบคำศัพท์ “<<EOF>>” กฎการตรวจสอบจะคืนค่าโทเค็น EOF

จากตารางที่ 3.5 สามารถแสดงรูปแบบการตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอ สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชัน ดังรูปที่ 3.7



รูปที่ 3.7 รูปแบบการตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอ สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชัน

จากรูปที่ 3.7 แสดงการตรวจสอบคำศัพท์ภาษาเอกซ์เอ็มไอ สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชัน ซึ่งตรวจสอบแท็ก ownedMember โดยพบคำศัพท์หรือเครื่องหมายและคืนค่าโทเค็น ดังนี้ client, name, realizingClassifier, xmivalue (ชื่อหรือรหัส), supplier, xmi:id, uml:Realization, “ = ” และ “ ” ” การตรวจสอบดังกล่าวคืนค่า CLIENT, ID, NAME, REALCLASS, XMIVALUE, SUPPILER, ID, REALIZATION, SEQ และ QUOTE ตามลำดับ

3.4 การนิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ

ในการนิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ [12] นั้น เพื่อระบุหรือจำแนกโครงสร้างของแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี ซึ่งการนิยามกฎอีบีเอ็นเอฟ เพื่อใช้ตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเครื่องมือตัวแปลภาษา แสดงชื่อกฎ การดำเนินการและคำอธิบาย ดังตารางที่ 3.6

ตารางที่ 3.6 นิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ

ชื่อกฎ	การดำเนินการ	คำอธิบาย
diagrams	components connectors attachments	ตรวจสอบสายอักขระ ซึ่งเป็นแผนภาพคอมโพเนนท์ ส่วนคอมโพเนนท์ย่อย
	diagram components connectors attachments	ตรวจสอบสายอักขระ ซึ่งเป็นแผนภาพคอมโพเนนท์ ส่วนคอมโพเนนท์หลัก
diagram	UMLDIAGRAM NAME SEQ QUOTE XMIVALUE QUOTE ID SEQ QUOTE XMIVALUE QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนแผนภาพคอมโพเนนท์
components	component ports representations	ตรวจสอบสายอักขระ ซึ่งเป็นคอมโพเนนท์และส่วนคอมโพเนนท์ย่อย
components	component ports representations { components }	ตรวจสอบสายอักขระ ซึ่งเป็นคอมโพเนนท์และส่วนคอมโพเนนท์ย่อยที่ซึ่งมีมากกว่าหนึ่งคอมโพเนนท์
	component ports	ตรวจสอบสายอักขระ ซึ่งเป็นคอมโพเนนท์
	component ports { components }	ตรวจสอบสายอักขระ ซึ่งเป็นคอมโพเนนท์ ที่ซึ่งมีมากกว่าหนึ่งคอมโพเนนท์

ตารางที่ 3.6 นิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ (ต่อ)

ชื่อกฎ	การดำเนินการ	คำอธิบาย
representations	representation diagrams	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนคอมโพเนนท์ย่อย
	representation diagrams { representations }	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนคอมโพเนนท์ย่อย ที่ซึ่งมีมากกว่าหนึ่งคอมโพเนนท์ย่อย
representation	UMLREPRESENTATION NAME SEQ QUOTE XMIVALUE QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนคอมโพเนนท์ย่อย
component	UMLCOMPONENT NAME SEQ QUOTE XMIVALUE QUOTE ID SEQ QUOTE XMIVALUE QUOTE INDINS SEQ boolean ISAB SEQ boolean ISAC SEQ boolean ISLE SEQ boolean	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนคอมโพเนนท์
ports	port	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนพอร์ต
	port { ports }	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนพอร์ตที่มีมากกว่าหนึ่งพอร์ต
port	UMLPORT NAME SEQ QUOTE XMIVALUE QUOTE ID SEQ QUOTE XMIVALUE QUOTE PORTTYPE SEQ QUOTE XMIVALUE QUOTE ISPORT SEQ QUOTE boolean QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนพอร์ต
connectors	connector roles	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนการเชื่อมต่อ
	connector roles { connectors }	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนการเชื่อมต่อที่มากกว่าหนึ่งการเชื่อมต่อ

ตารางที่ 3.6 นิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ (ต่อ)

ชื่อกฎ	การดำเนินการ	คำอธิบาย
connector	UMLCONNECTOR REALIZATION ID SEQ QUOTE XMIVALUE QUOTE CONNECTOR SEQ QUOTE XMIVALUE QUOTE NAME SEQ QUOTE XMIVALUE QUOTE REALCLASS SEQ QUOTE XMIVALUE QUOTE SUPPLIER SEQ QUOTE XMIVALUE QUOTE CLIENT SEQ QUOTE XMIVALUE QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนการเชื่อมต่อประเภทเรียลไลเซชัน (Realization)
	UMLCONNECTOR USAGE ID SEQ QUOTE XMIVALUE QUOTE CONNECTOR SEQ QUOTE XMIVALUE QUOTE NAME SEQ QUOTE XMIVALUE QUOTE SUPPLIER SEQ QUOTE XMIVALUE QUOTE CLIENT SEQ QUOTE XMIVALUE QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนการเชื่อมต่อประเภทยูซเสจ (Usage)
	UMLCONNECTOR DEPENDENCY ID SEQ QUOTE XMIVALUE QUOTE CONNECTOR SEQ QUOTE XMIVALUE QUOTE NAME SEQ QUOTE XMIVALUE QUOTE SUPPLIER SEQ QUOTE XMIVALUE QUOTE CLIENT SEQ QUOTE XMIVALUE QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนการเชื่อมต่อประเภทดีเพนเดนซี (Dependency)
	UMLCONNECTOR ASSOCIATION ID SEQ QUOTE XMIVALUE QUOTE CONNECTOR SEQ QUOTE XMIVALUE QUOTE NAME SEQ QUOTE XMIVALUE QUOTE ISAB SEQ boolean ISDE SEQ boolean ISLE SEQ boolean	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนการเชื่อมต่อประเภทแอสโซซิเอชัน (Association)

ตารางที่ 3.6 นิยามกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ (ต่อ)

ชื่อกฎ	การดำเนินการ	คำอธิบาย
connector	UMLCONNECTOR UMLINTERFACE ID SEQ QUOTE XMIVALUE QUOTE CONNECTOR SEQ QUOTE XMIVALUE QUOTE NAME SEQ QUOTE XMIVALUE QUOTE VISIBILITY SEQ visibilities ISAB SEQ boolean ISAC SEQ boolean ISLE SEQ boolean	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนการเชื่อมต่อประเภทอินเตอร์เฟส (Interface)
roles	role	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนบทบาท (Role)
	role { roles }	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนบทบาทที่มากกว่าหนึ่งบทบาท
role	UMLROLE ID SEQ QUOTE XMIVALUE QUOTE NAME SEQ QUOTE XMIVALUE QUOTE CLIENT SEQ QUOTE XMIVALUE QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนบทบาท
attachments	attachment	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนแอทแทชเมนท์
	attachment { attachments }	ตรวจสอบสายอักขระ ซึ่งเป็นส่วนแอทแทชเมนท์ ที่มากกว่าหนึ่ง แอทแทชเมนท์
attachment	ATTACHMENT ISFROM SEQ QUOTE XMIVALUE QUOTE ISTO SEQ QUOTE XMIVALUE QUOTE	ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนแอทแทชเมนท์
boolean	BOOLEANT | BOOLEANF	ตรวจสอบสายอักขระ ซึ่งเป็นค่าความจริง (Boolean)
visibilities	VISUNS | VISPUB | VISPAC | VISPRO | VISPRI	ตรวจสอบสายอักขระ ซึ่งเป็นค่า visibilities

จากตารางที่ 3.6 อักษรตัวพิมพ์ใหญ่ แทนสัญลักษณ์เทอมินัลและอักษรตัวพิมพ์เล็ก แทนสัญลักษณ์นอลเทอร์มินัล ตัวอย่างจากกฎชื่อ “diagrams” สามารถดำเนินการได้สองทางเลือก ดังนี้

1) ดำเนินการตรวจสอบสายอักขระซึ่งเป็นแผนภาพคอมโพเนนท์ ที่ซึ่งประกอบด้วยชื่อกฎ components, connectors, และ attachments โดยนำไปใช้สำหรับตรวจสอบโครงสร้างแผนภาพคอมโพเนนท์ส่วนแผนภาพคอมโพเนนท์ย่อย
2) ดำเนินการตรวจสอบสายอักขระซึ่งเป็นแผนภาพคอมโพเนนท์ ที่ซึ่งประกอบด้วยชื่อกฎ diagram, components, connectors, และ attachments โดยนำไปใช้สำหรับตรวจสอบโครงสร้างแผนภาพคอมโพเนนท์ส่วนแผนภาพคอมโพเนนท์หลัก

3.5 การนิยามกฎการแปลงข้อมูลแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี

การนิยามกฎสำหรับการแปลงไวยากรณ์ทางภาษาระหว่างภาษาเอกซ์เอ็มไอกับภาษาแอคมีด้วยกฎการตรวจสอบไวยากรณ์ของแผนภาพคอมโพเนนท์ ซึ่งการแปลงข้อมูลแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมีประกอบด้วยไวยากรณ์ ดังนี้ ระบบ คอมโพเนนท์ การเชื่อมต่อ พอร์ต บทบาท ลักษณะประจำ แอทแทชเมนท์ และระบบย่อย โดยอธิบายไวยากรณ์ภาษายูเอ็มแอลและภาษาแอคมี รวมทั้งคำอธิบายรายละเอียดดังตารางที่ 3.7

ตารางที่ 3.7 นิยามกฎการแปลงข้อมูลระหว่างแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี [25]

ไวยากรณ์ภาษายูเอ็มแอล (UML Syntax)	**ไวยากรณ์ภาษาแอคมี (Acme: BNF/EBNF Grammar)**	**คำอธิบาย**
Component Diagram	SystemDeclaration ::= {: "System" <IDENTIFIER> :} ({: "=" :} SystemBody ({: ";" :})? | {: ";" :})	แปลงข้อมูลส่วนแผนภาพคอมโพเนนท์ ที่ซึ่งประกอบด้วยไวยากรณ์ SystemBody และ SystemStructure
	SystemBody ::= ({: "{ }" :} | {: "{" :} (SystemStructure)+ {: "}" :})	แปลงข้อมูลส่วนแผนภาพคอมโพเนนท์ (โครงสร้างหลัก) ซึ่งประกอบด้วยไวยากรณ์ SystemStructure
	SystemStructure ::= ComponentDeclaration | ConnectorDeclaration | PortDeclaration | RoleDeclaration | PropertyDeclaration | AttachmentsDeclaration | RepresentationDeclaration	แปลงข้อมูลส่วนแผนภาพคอมโพเนนท์ (โครงสร้าง) ที่ซึ่งประกอบด้วยไวยากรณ์ Component, Connector, Port, Role, Property, Attachments, และ Representation

ตารางที่ 3.7 นิยามกฎการแปลงข้อมูลระหว่างแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี [25] (ต่อ)

ไวยากรณ์ภาษายูเอ็มแอล (UML Syntax)	**ไวยากรณ์ภาษาแอคมี (Acme: BNF/EBNF Grammar)**	**คำอธิบาย**
Component	ComponentDeclaration ::= <COMPONENT> <IDENTIFIER> ("=" parse_ComponentDescription ";" | ";")	แปลงข้อมูลคอมโพเนนท์และลักษณะประจำสำหรับคอมโพเนนท์ใดๆ
	parse_ComponentDescription ::= ("{" (PortDeclaration | PropertyDeclaration | RepresentationDeclaration)* "}") PropertyDeclaration | RepresentationDeclaration)* "}")	แปลงข้อมูลลักษณะประจำสำหรับคอมโพเนนท์ใดๆ ที่ซึ่งประกอบด้วยไวยากรณ์ Port, Property, และ Representation
Connector	ConnectorDeclaration ::= <CONNECTOR> <IDENTIFIER> ("=" parse_ConnectorDescription ";" | ";")	แปลงข้อมูลส่วนการเชื่อมต่อและลักษณะประจำ สำหรับการเชื่อมต่อใดๆ
	parse_ConnectorDescription ::= ("{" (RoleDeclaration | PropertyDeclaration | RepresentationDeclaration)* "}")	แปลงข้อมูลลักษณะประจำสำหรับการเชื่อมต่อใดๆ ที่ซึ่งประกอบด้วยไวยากรณ์ Role, Property, และ Representation
Port	PortDeclaration ::= <PORT> <IDENTIFIER> ("=" parse_PortDescription ";" | ";")	แปลงข้อมูลส่วนพอร์ตและลักษณะประจำ สำหรับพอร์ตใดๆ
	parse_PortDescription ::= ("{" (PropertyDeclaration | RepresentationDeclaration)* "}")	แปลงข้อมูลลักษณะประจำสำหรับพอร์ตใดๆ ที่ซึ่งประกอบด้วยไวยากรณ์ Property และ Representation

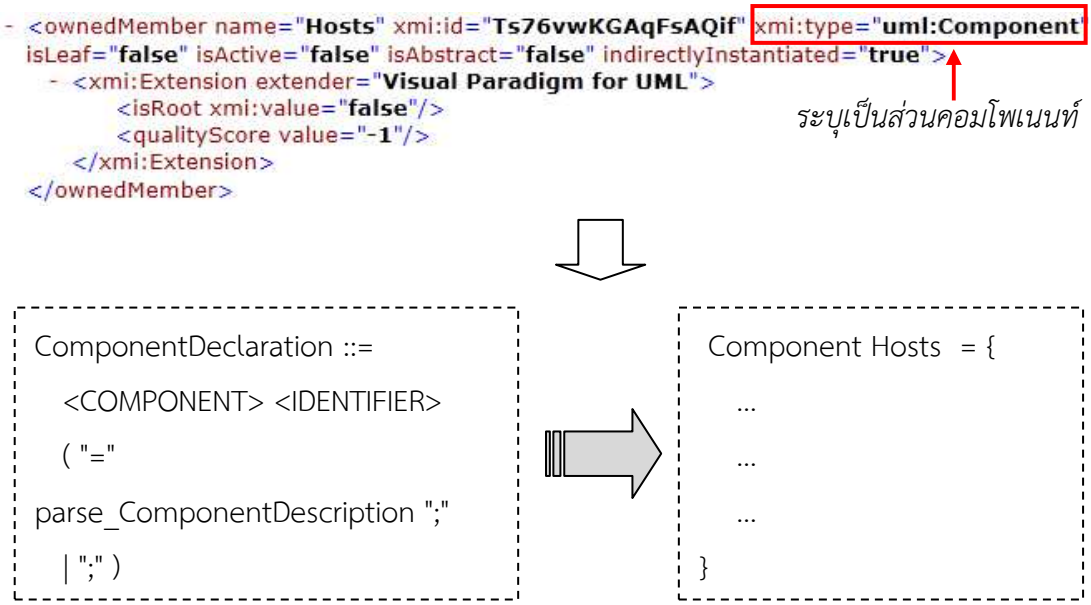
ตารางที่ 3.7 นิยามกฎการแปลงข้อมูลระหว่างแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี [25] (ต่อ)

ไวยากรณ์ภาษายูเอ็มแอล (UML Syntax)	**ไวยากรณ์ภาษาแอคมี (Acme: BNF/EBNF Grammar)**	**คำอธิบาย**
Role *(ระบุด้วยประเภทการเชื่อมต่อ)*	RoleDeclaration ::= <ROLE> <IDENTIFIER> ("=" parse_RoleDescription ";" | ";")	แปลงข้อมูลส่วนบทบาทและลักษณะประจำสำหรับบทบาทใดๆ
	parse_RoleDescription ::= ("{" (PropertyDeclaration | RepresentationDeclaration)* "}")	แปลงข้อมูลลักษณะประจำสำหรับบทบาทใดๆ ที่ซึ่งประกอบด้วยไวยากรณ์ Property และ Representation
Attachment *(ระบุด้วยคอมโพเนนท์และพอร์ต กับการเชื่อมต่อและประเภทการเชื่อมต่อ)*	AttachmentsDeclaration ::= (<ATTACHMENT> <IDENTIFIER> "." <IDENTIFIER> "to" <IDENTIFIER> "." <IDENTIFIER> ";")	แปลงข้อมูลส่วน attachment และลักษณะประจำ สำหรับ attachment ใดๆ ที่ซึ่งระบุถึงความสัมพันธ์ระหว่างคอมโพเนนท์และพอร์ต กับ การเชื่อมต่อและประเภทการเชื่อมต่อ
Attribute	PropertyDeclaration ::= <PROPERTY> parse_PropertyDescription ";"	แปลงข้อมูลลักษณะประจำใดๆ ที่ซึ่งประกอบด้วยไวยากรณ์รายการคำอธิบายลักษณะประจำ
	parse_PropertyDescription ::= <IDENTIFIER> (":" PropertyTypeDescription)? ("=" PropertyValueDeclaration)?	แปลงข้อมูลคำอธิบายลักษณะประจำใดๆ ที่ซึ่งประกอบด้วยไวยากรณ์ประเภทและค่าลักษณะประจำ
	PropertyTypeDescription ::= <ANY> | <STRING> | <BOOLEAN>	แปลงข้อมูลประเภทลักษณะประจำใดๆ
	PropertyValueDeclaration ::= <STRING_LITERAL> | <FALSE> | <TRUE> | <IDENTIFIER>	แปลงข้อมูลค่าลักษณะประจำใดๆ

ตารางที่ 3.7 นิยามกฎการแปลงข้อมูลระหว่างแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี [25] (ต่อ)

ไวยากรณ์ภาษายูเอ็มแอล (UML Syntax)	ไวยากรณ์ภาษาแอคมี (Acme: BNF/EBNF Grammar)	คำอธิบาย
Subsystem	RepresenatationDeclaration ::= <REPRESENTATION> (<IDENTIFIER> "=")? "{" SystemDeclaration "}" (";")?	แปลงข้อมูลส่วนคอมโพเนนท์ย่อย ที่ซึ่งประกอบด้วยไวยากรณ์ System ใดๆ

จากตารางที่ 3.7 สามารถแสดงตัวอย่างการแปลงข้อมูลระหว่างระหว่างแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี (จากคอมโพเนนท์ยูเอ็มแอลในรูปที่ 3.4) ดังรูปที่ 3.8



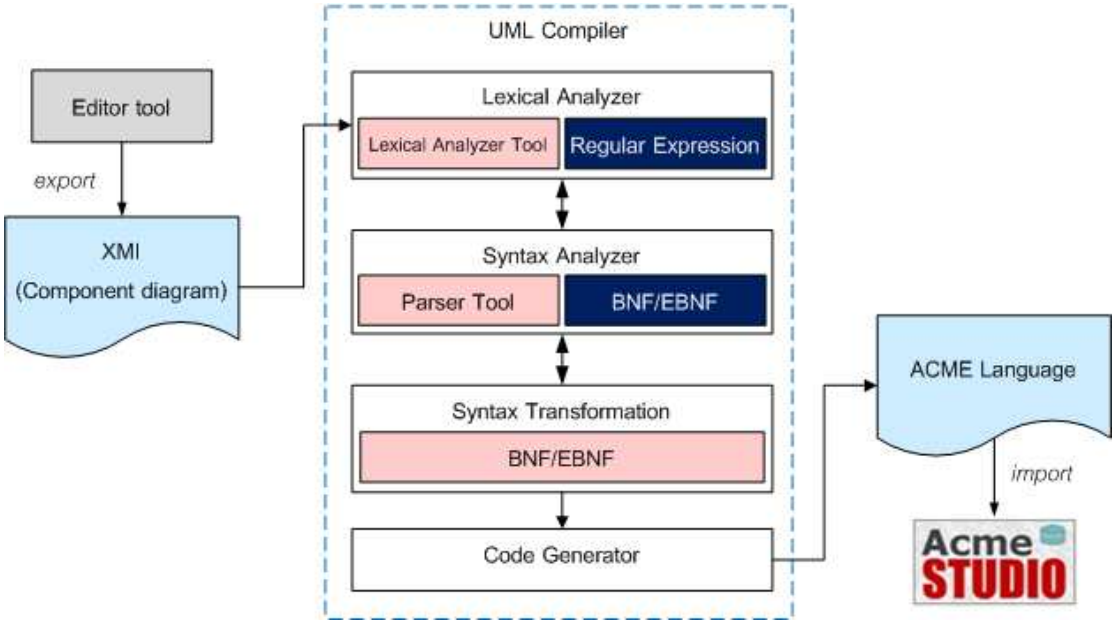
รูปที่ 3.8 การแปลงข้อมูลระหว่างระหว่างแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี

บทที่ 4
การออกแบบและพัฒนาเครื่องมือตัวแปลภาษา

สำหรับการออกแบบและพัฒนาเครื่องมือตัวแปลภาษา ซึ่งอธิบายแยกเป็นสองส่วนโดยส่วนแรกกล่าวถึงการออกแบบเครื่องมือ ซึ่งอธิบายภาพรวมการดำเนินงานเครื่องมือตัวแปลภาษา และขั้นตอนการดำเนินงาน โดยนำเสนอด้วยแผนภาพกิจกรรม แผนภาพยูสเคสและแผนภาพคลาสตามลำดับ และส่วนที่สองกล่าวถึงการพัฒนาเครื่องมือตัวแปลภาษา กล่าวถึงสภาพแวดล้อมสำหรับการพัฒนาเครื่องมือและส่วนต่อประสานกับผู้ใช้เครื่องมือ ซึ่งมีรายละเอียด ดังนี้

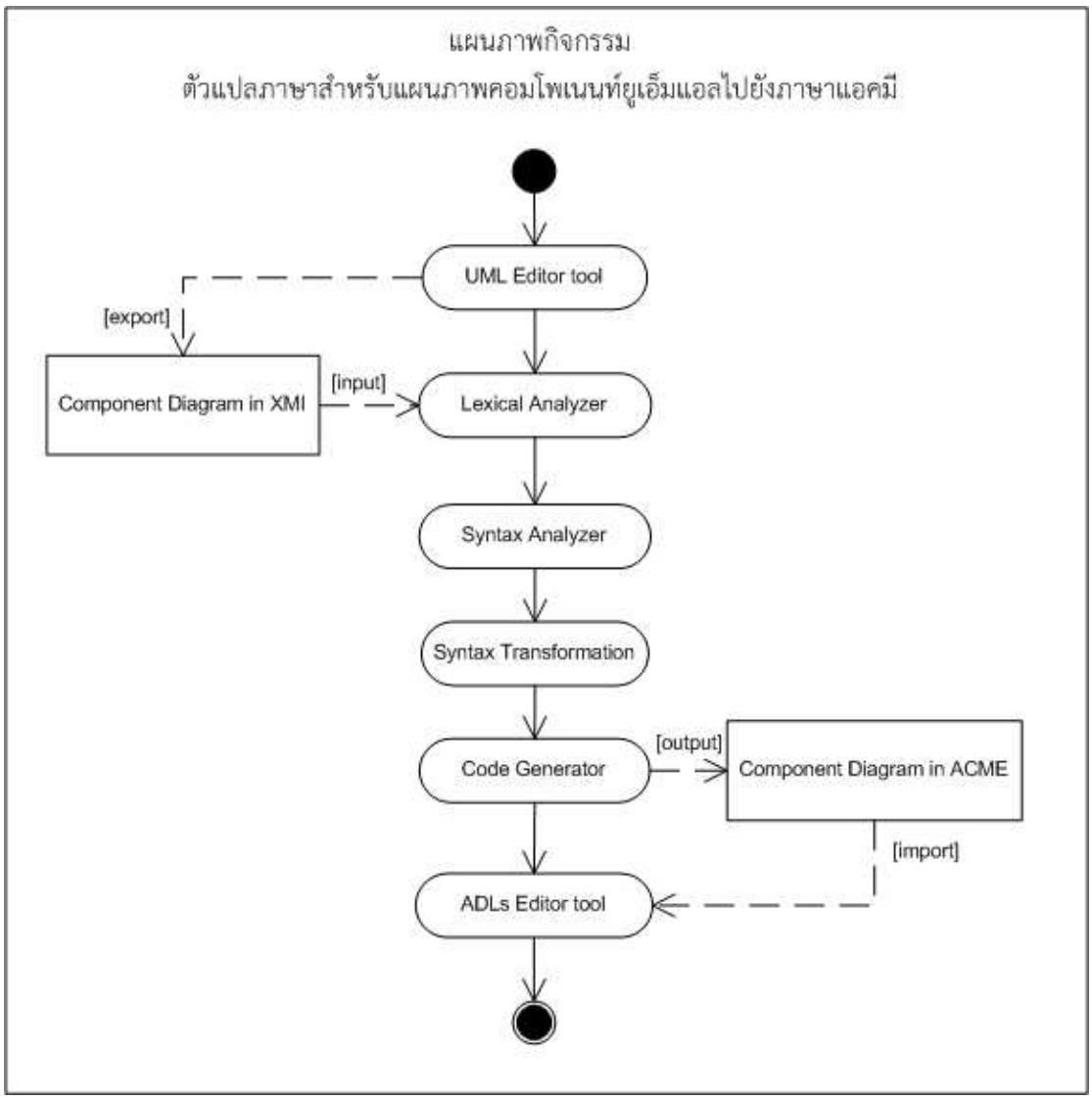
4.1 ภาพรวมการดำเนินงานเครื่องมือตัวแปลภาษา

ในภาพรวมการดำเนินงานเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล ซึ่งแผนภาพดังกล่าวถูกจัดเก็บเป็นภาษาเอกซ์เอ็มไอ โดยวิเคราะห์จากลักษณะโครงสร้างไวยากรณ์ของแผนภาพคอมโพเนนท์ เพื่อสร้างข้อมูลผลลัพธ์ภาษาแอคมีและนำข้อมูลผลลัพธ์ดังกล่าวไปอธิบายแบบจำลองด้านสถาปัตยกรรมด้วยเครื่องมือทางภาษาเอดีแอลได้ ตัวอย่างเช่น เครื่องมือแอคมีสตูดิโอ เป็นต้น โดยแสดงภาพรวมการดำเนินงาน ดังรูปที่ 4.1



รูปที่ 4.1 ภาพรวมการดำเนินงานตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล

จากรูปที่ 4.1 แสดงแผนภาพกิจกรรมภาพรวมการดำเนินงานตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี ดังรูปที่ 4.2

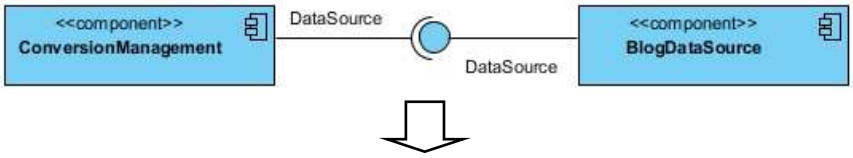


รูปที่ 4.2 แผนภาพกิจกรรมภาพรวมการดำเนินงานตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี

จากรูปที่ 4.2 สามารถแสดงรายละเอียดขั้นตอนการดำเนินดำเนินงานตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี ซึ่งอธิบายรายละเอียดการดำเนินงาน ดังนี้

4.1.1 การเตรียมข้อมูลแผนภาพคอมโพเนนท์ยูเอ็มแอลเป็นภาษาเอกซ์เอ็มไอ

โดยข้อมูลดังกล่าวได้นำออกมาจากเครื่องมือ Visual Paradigm รุ่นที่ 10.0 เพื่อใช้เป็นข้อมูลนำเข้าสำหรับเครื่องมือตัวแปลภาษา ดังตัวอย่างการเตรียมข้อมูลแผนภาพคอมโพเนนท์ยูเอ็มแอลเป็นภาษาเอกซ์เอ็มไอ ซึ่งประกอบด้วยคอมโพเนนท์สองคอมโพเนนท์ ที่ซึ่งมีความสัมพันธ์แบบแอสเซมบลี ดังรูปที่ 4.3




```
- <ownedMember name="ConversionManagement" xmi:id=".VWiHgKGAqFsAQV_"
  xmi:type="uml:Component" isLeaf="false" isActive="false" isAbstract="false"
  indirectlyInstantiated="true">
    - <xmi:Extension extender="Visual Paradigm for UML">
          <isRoot xmi:value="false"/>
          <qualityScore value="-1"/>
      </xmi:Extension>
  </ownedMember>
- <ownedMember name="BlogDataSource" xmi:id="m3WiHgKGAqFsAQWE"
  xmi:type="uml:Component" isLeaf="false" isActive="false" isAbstract="false"
  indirectlyInstantiated="true">
    - <xmi:Extension extender="Visual Paradigm for UML">
          <isRoot xmi:value="false"/>
          <qualityScore value="-1"/>
      </xmi:Extension>
  </ownedMember>
- <ownedMember name="Interface" xmi:id="dSTiHgKGAqFsAQWs" xmi:type="uml:Interface"
  isLeaf="false" isActive="false" isAbstract="false" visibility="public">
    - <xmi:Extension extender="Visual Paradigm for UML">
          <isRoot xmi:value="false"/>
          <modelType value="Class"/>
          <businessModel xmi:value="false"/>
          <qualityScore value="-1"/>
          <appliedStereotype xmi:value="Class_Interface_id"/>
      </xmi:Extension>
  </ownedMember>
- <ownedMember name="DataSource" xmi:id="MaTiHgKGAqFsAQWw"
  xmi:type="uml:Realization" supplier="dSTiHgKGAqFsAQWs"
  realizingClassifier="dSTiHgKGAqFsAQWs" client=".VWiHgKGAqFsAQV_">
    - <xmi:Extension extender="Visual Paradigm for UML">
          <qualityScore value="-1"/>
      </xmi:Extension>
  </ownedMember>
- <ownedMember name="DataSource" xmi:id="ugLiHgKGAqFsAQW7" xmi:type="uml:Usage"
  supplier="dSTiHgKGAqFsAQWs" client="m3WiHgKGAqFsAQWE">
    - <xmi:Extension extender="Visual Paradigm for UML">
          <qualityScore value="-1"/>
      </xmi:Extension>
  </ownedMember>
```

รูปที่ 4.3 ข้อมูลนำเข้าภาษาเอกซ์เอ็มไอแปลงจากแผนภาพคอมโพเนนท์ [23]

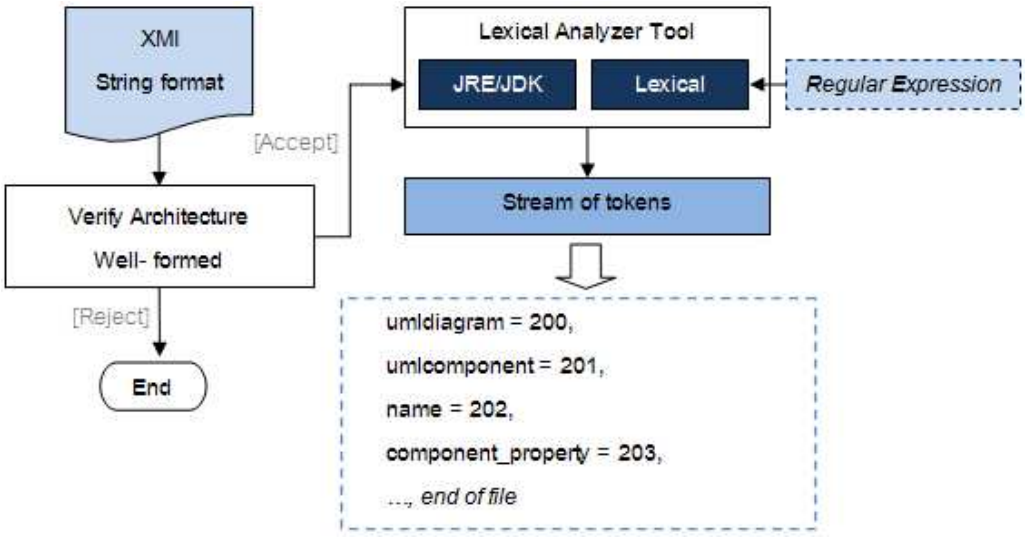
สำหรับข้อมูลนำเข้าภาษาเอกซ์เอ็มไอนั้น เครื่องมือตัวแปลภาษากำหนดให้สามารถตรวจสอบแท็ก ที่ซึ่งเป็นส่วนประกอบของแผนภาพคอมโพเนนท์ยูเอ็มแอล ดังตารางที่ 3.1

4.1.2 กระบวนการวิเคราะห์คำศัพท์แผนภาพคอมโพเนนท์ (Lexical Analyzer)

โดยแผนภาพคอมโพเนนท์จัดเก็บเป็นภาษาเอกซ์เอ็มไอ ต้องผ่านการตรวจสอบรูปแบบของภาษาเอกซ์เอ็มไอว่าเป็นรูปแบบของแผนภาพคอมโพเนนท์ที่ซึ่งแทนสถาปัตยกรรมระบบ และสอดคล้องกับข้อกำหนดการสร้างแบบจำลองสถาปัตยกรรมซอฟต์แวร์ในภาษายูเอ็มแอล [24, 26] ถ้าแผนภาพคอมโพเนนท์ดังกล่าวได้ตรวจสอบว่าถูกออกแบบสำหรับแทนสถาปัตยกรรมโดยแท้จริงแล้วนั้น จะระบุเป็นข้อมูลนำเข้าสำหรับกระบวนการวิเคราะห์คำศัพท์ ซึ่งดำเนินการด้วยเครื่องมือ

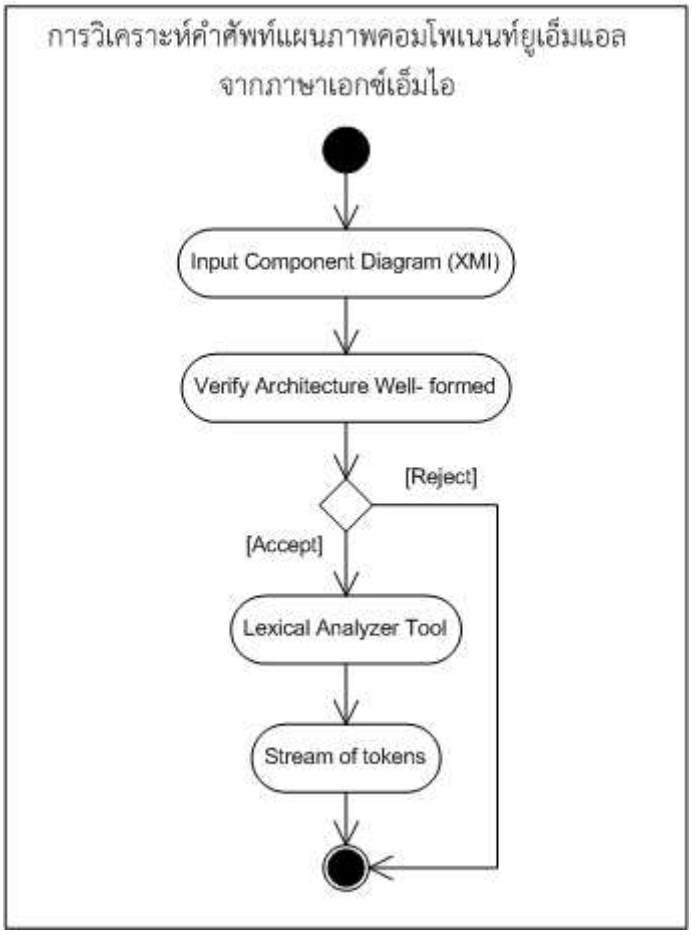
เจเฟล็กซ์ ที่ซึ่งเป็นเครื่องมือใช้สำหรับการวิเคราะห์คำศัพท์และนำเสนอด้วยภาษาจาวาและข้อมูลผลลัพธ์ถูกแปลงเป็นลำดับของเครื่องหมายที่แทนด้วยแผนภาพคอมโพเนนท์ในรูปแบบภาษาเอกซ์เอ็มไอที่วิเคราะห์คำศัพท์จากแบบรูป ซึ่งนิยามตามมาตรฐานภาษายูเอ็มแอล รุ่นที่ 2.0 สำหรับคำศัพท์ของแผนภาพคอมโพเนนท์นิยามด้วยรูปแบบเรกูลาร์เอกซ์เพรสชัน เพื่ออธิบายโครงสร้างของแผนภาพคอมโพเนนท์ แต่ถ้าแผนภาพคอมโพเนนท์ดังกล่าวได้ตรวจสอบแล้วว่าไม่ได้ถูกออกแบบสำหรับแทนสถาปัตยกรรมแล้วนั้นเครื่องมือตัวแปลภาษาจะสิ้นสุดการดำเนินการ

สำหรับการกำหนดหลักไวยากรณ์ในลักษณะกลุ่มของคำศัพท์ นิยามไว้ดังรูปที่ 3.3 ซึ่งจากกฎดังกล่าวจะนำเสนอด้วยเครื่องมือที่เป็นอัตโนมัติ ด้วยการวิเคราะห์คำศัพท์จากการนิยามกฎ (Rules) และผลการดำเนินการ (Action) และจากการนิยามกฎไวยากรณ์ทางภาษาดังกล่าว นำไปสู่กระบวนการวิเคราะห์ความหมายของแผนภาพคอมโพเนนท์และสร้างเป็นผลลัพธ์ในรูปแบบการไหลของเครื่องหมาย โดยใช้เป็นข้อมูลนำเข้าของกระบวนการวิเคราะห์ไวยากรณ์ต่อไป แสดงดังรูปที่ 4.4



รูปที่ 4.4 การวิเคราะห์คำศัพท์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากภาษาเอกซ์เอ็มไอ

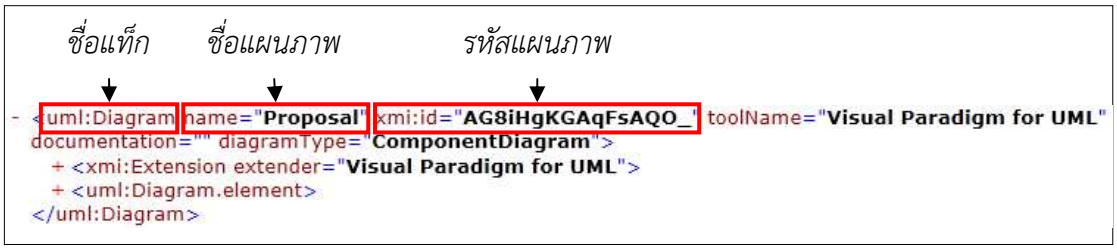
จากรูปที่ 4.4 สามารถแสดงแผนภาพกิจกรรมการวิเคราะห์คำศัพท์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากภาษาเอกซ์เอ็มไอ ดังรูปที่ 4.5



รูปที่ 4.5 แผนภาพกิจกรรมการวิเคราะห์คำศัพท์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากภาษาเอกซ์เอ็มไอ

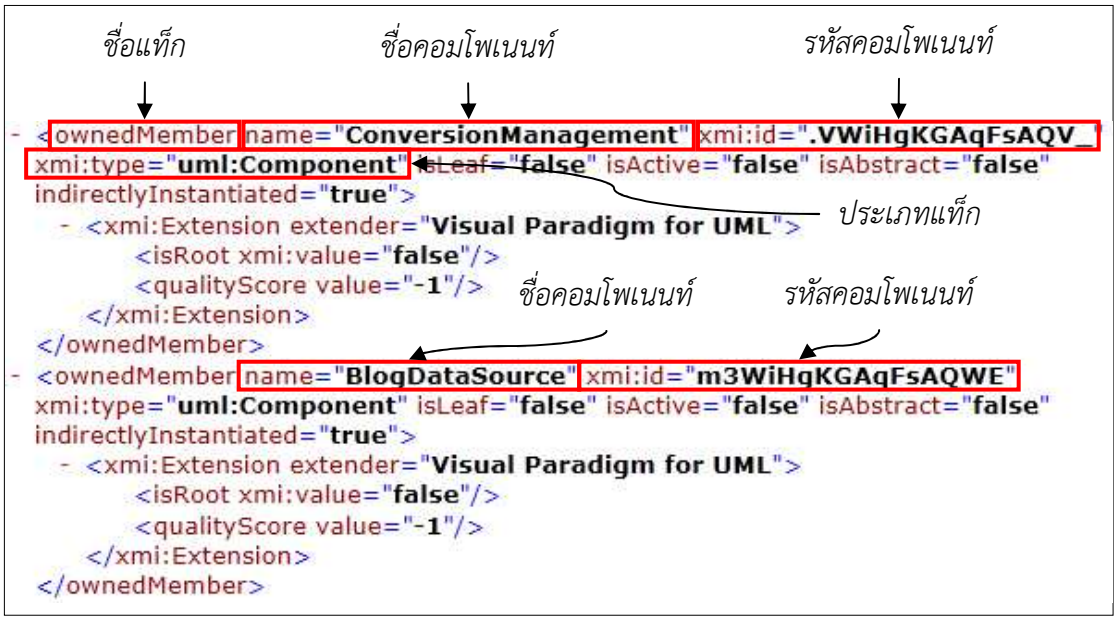
จากกระบวนการดังกล่าว สามารถแสดงตัวอย่างการวิเคราะห์คำศัพท์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเอกสารเอกซ์เอ็มไอ จากรูปที่ 4.3 ดังนี้

1) ส่วนแผนภาพคอมโพเนนท์ วิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนแผนภาพคอมโพเนนท์ ซึ่งวิเคราะห์แท็ก uml:Diagram ที่ซึ่งประกอบด้วยชื่อ “Proposal” และรหัสแผนภาพ “AG8iHgKGAqFsAQO_” ตามลำดับ แสดงดังรูปที่ 4.6



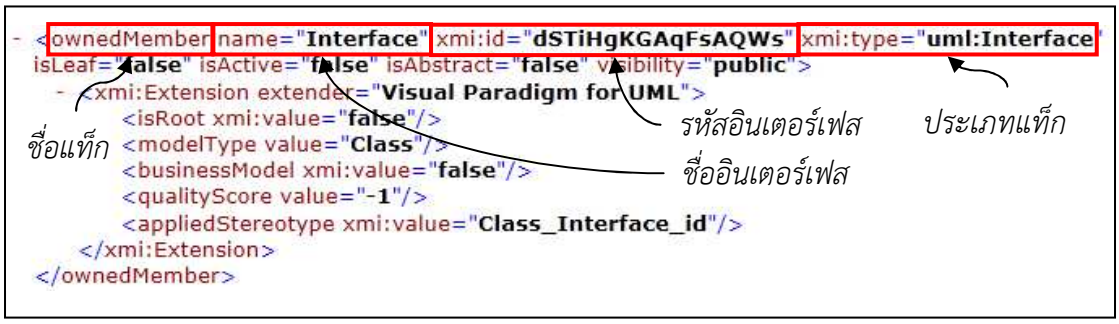
รูปที่ 4.6 การวิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนแผนภาพคอมโพเนนท์

2) ส่วนคอมโพเนนท์ วิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนคอมโพเนนท์ ซึ่งวิเคราะห์แท็ก ownedMember ด้วยลักษณะประจำประเภท “uml:Component” ที่ซึ่งประกอบด้วยชื่อ “ConversionManagement” และ “ BlogDataSource” และรหัส “ .VWiHgKGAqFsAQV_” และ “m3WiHgKGAqFsAQWE” ตามลำดับ แสดงดังรูปที่ 4.7



รูปที่ 4.7 การวิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนคอมโพเนนท์

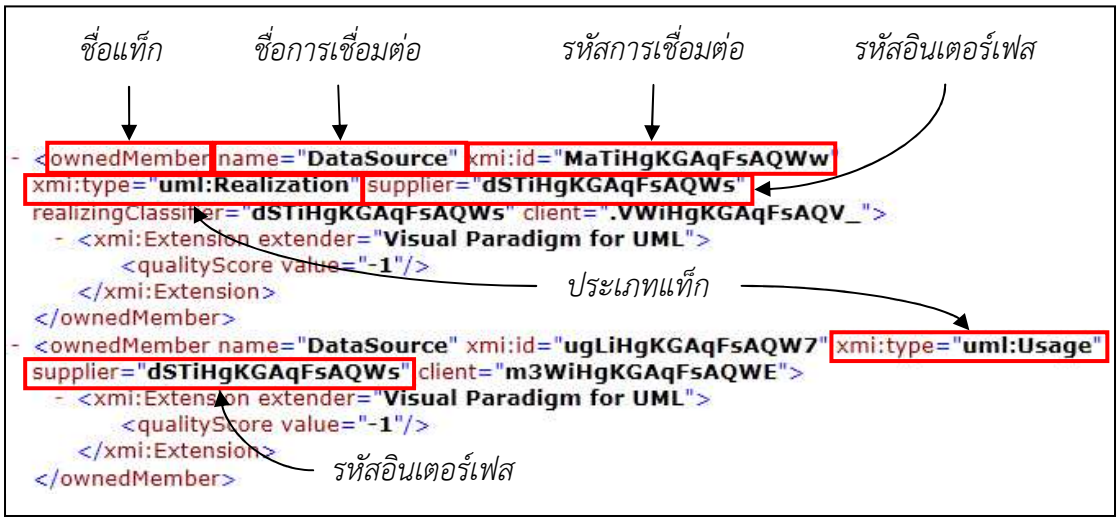
3) ส่วนอินเตอร์เฟส วิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนอินเตอร์เฟส ซึ่งวิเคราะห์แท็ก ownedMember ด้วยลักษณะประจำประเภท “uml:Interface” ที่ซึ่งประกอบด้วยชื่อ “Interface” และรหัส “dSTiHgKGAqFsAQWs” ตามลำดับ แสดงดังรูปที่ 4.8



รูปที่ 4.8 การวิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนอินเตอร์เฟส

4) ส่วนการเชื่อมต่อ วิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนการเชื่อมต่อ ซึ่งวิเคราะห์แท็ก ownedMember ด้วยลักษณะประจำประเภท “uml:Realization” และ

“uml:Usage” ที่ซึ่งประกอบด้วยชื่อ “DataSource” ทั้งสองการเชื่อมต่อและรหัส “MaTiHgKGAqFsAQWw” และ “ugLiHgKGAqFsAQW7” ตามลำดับโดยทั้งสองการเชื่อมต่อเรียกใช้งานอินเตอร์เฟสเดียวกัน คือ อินเตอร์เฟสรหัส “dSTiHgKGAqFsAQWs” (รูปที่ 4.8) แสดงดังรูปที่ 4.9



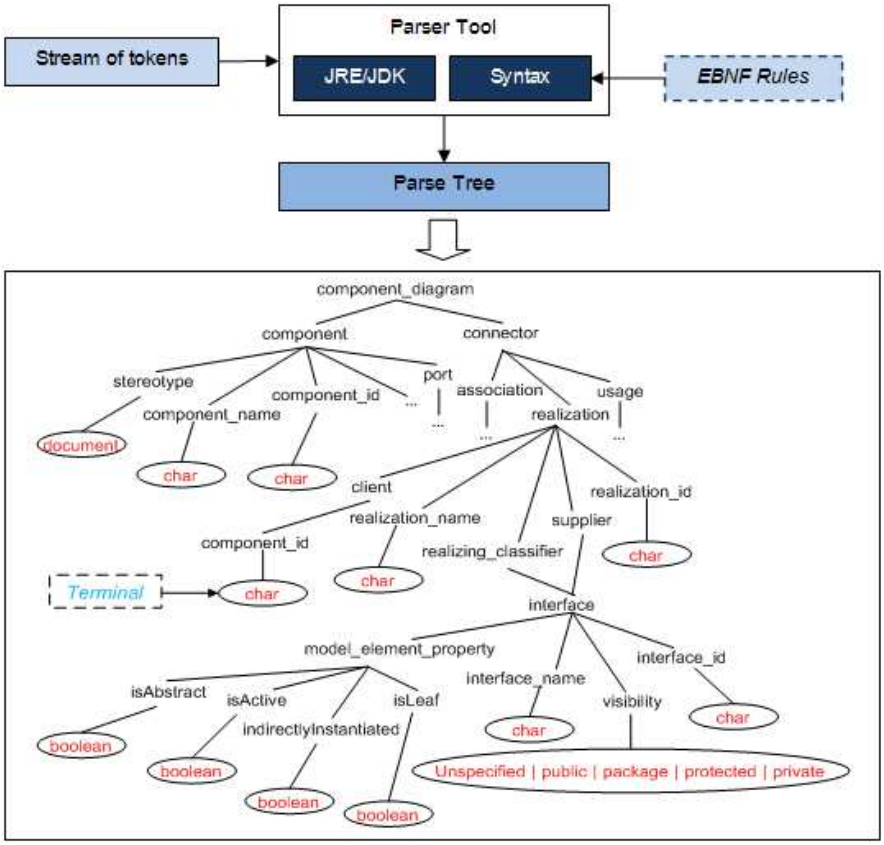
รูปที่ 4.9 การวิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วนการเชื่อมต่อ

5) ส่วน attachment วิเคราะห์คำศัพท์เอกสารเอกซ์เอ็มไอส่วน attachment ซึ่งระบุด้วยคอมโพเนนท์ชื่อ “ConversionManagement” และ “BlogDataSource” ที่ส่งออกไปยังเส้นเชื่อมต่อชื่อ “DataSource” โดยผ่านอินเตอร์เฟสชื่อ “Interface” ที่ซึ่งมีบทบาท (Role) เป็น “Realization” และ “Usage” ตามลำดับ

4.1.3 กระบวนการวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ (Syntax Analyzer)

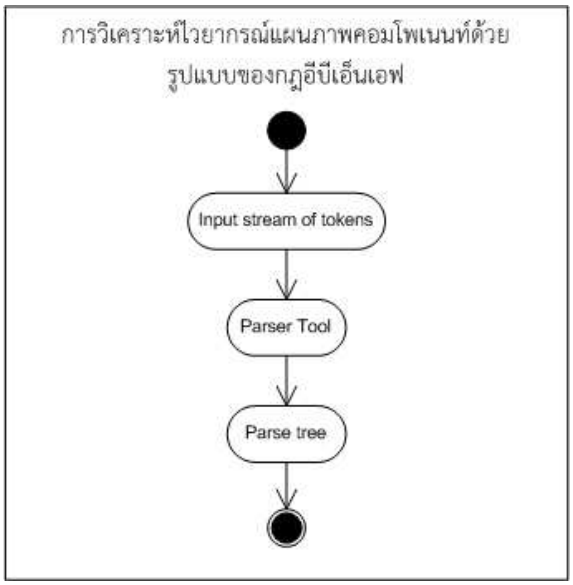
สำหรับกระบวนการวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์นำเสนอด้วยเครื่องมือแย็กค์ ซึ่งผลลัพธ์ของกระบวนการจะแปลงเป็นต้นไม้การแจงส่วน ตามหลักไวยากรณ์ภาษายูเอ็มแอล นำเข้าข้อมูลผลลัพธ์จากกระบวนการที่หนึ่ง ซึ่งการนิยามลักษณะไวยากรณ์จะนิยามด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ เพื่อจำแนกและจัดการลักษณะไวยากรณ์ของแผนภาพคอมโพเนนท์ยูเอ็มแอลที่ถูกต้อง

สำหรับการนิยามรูปแบบไวยากรณ์สำหรับแผนภาพคอมโพเนนท์ด้วยรูปแบบของกฎอีบีเอ็นเอฟ เพื่อเป็นกฎเบื้องต้นในการตรวจสอบแผนภาพคอมโพเนนท์ด้วยเครื่องมือตัวแปลภาษา แสดงดังตารางที่ 3.4 ซึ่งจากการนิยามลักษณะของไวยากรณ์ทางภาษาดังกล่าว สามารถนำไปใช้เป็นกฎตั้งต้นสำหรับกระบวนการวิเคราะห์ไวยากรณ์โครงสร้างของแผนภาพคอมโพเนนท์ ซึ่งการวิเคราะห์แผนภาพคอมโพเนนท์จะดำเนินการในกระบวนการวิเคราะห์ไวยากรณ์ ที่ซึ่งคืนค่าผลลัพธ์ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.10



รูปที่ 4.10 การวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ด้วยรูปแบบของกฎอีบีเอ็นเอฟ

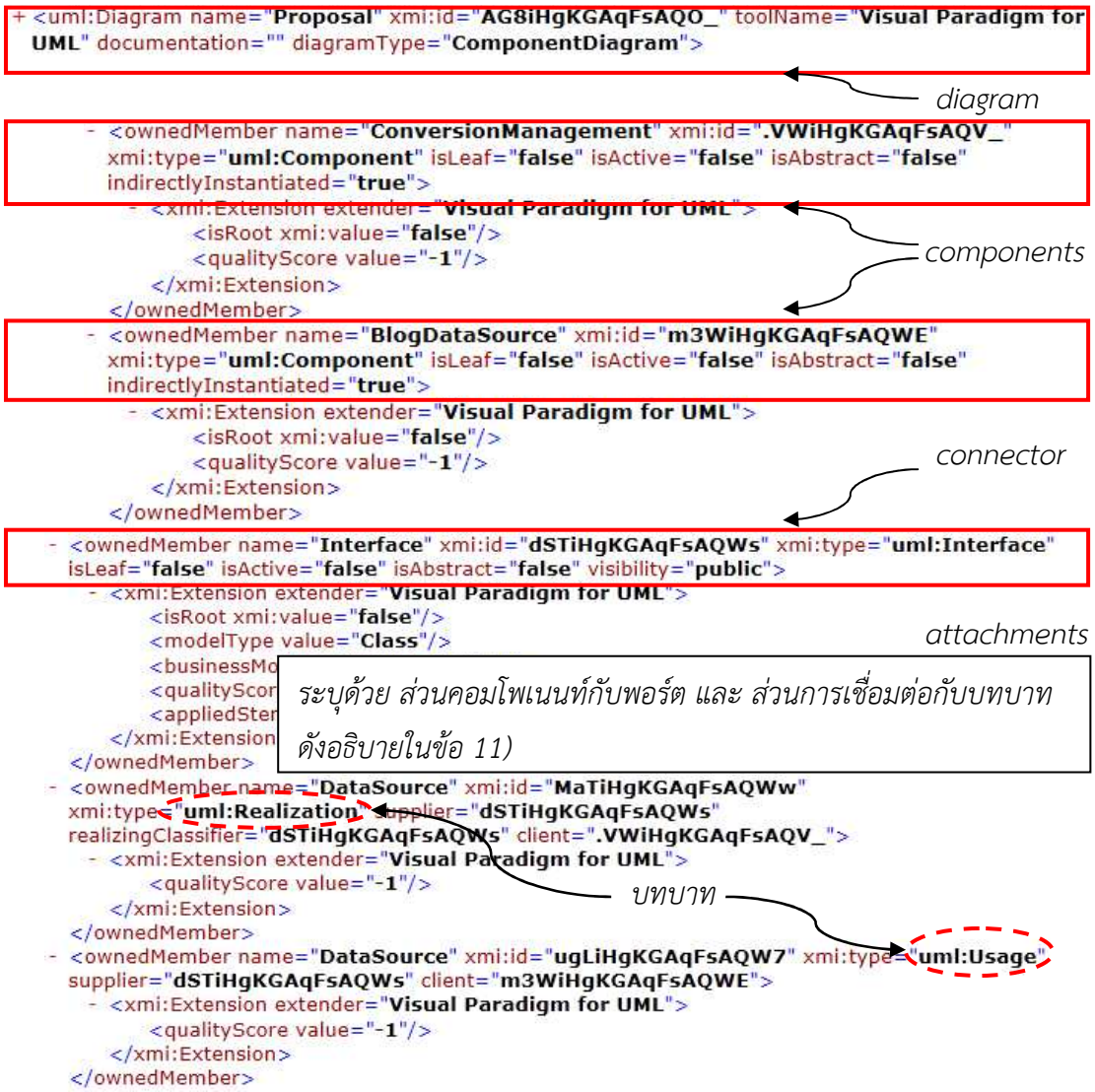
จากรูปที่ 4.10 สามารถแสดงแผนภาพกิจกรรมการวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ด้วยรูปแบบของกฎอีบีเอ็นเอฟ ดังรูปที่ 4.11



รูปที่ 4.11 แผนภาพกิจกรรมการวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ด้วยกฎอีบีเอ็นเอฟ

จากรูปที่ 4.11 เป็นกระบวนการวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ด้วยกฎอีบีเอ็นเอฟ ซึ่งเริ่มด้วยการนำเข้าข้อมูลสายอักขระโทเค็น ที่ซึ่งตรวจสอบคำศัพท์จากภาษาเอกซ์เอ็มไอ นำเข้ายังเครื่องมือตัวแจงส่วน เพื่อวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลและคืนค่าผลลัพธ์ในรูปแบบต้นไม้การแจงส่วนอธิบายรายละเอียด (จากเอกซ์เอ็มไอในรูปที่ 4.3) ดังนี้

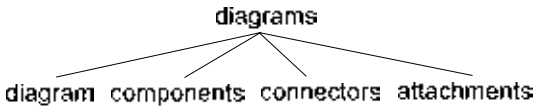
1) กฎ diagrams ตรวจสอบสายอักขระ ซึ่งเป็นแผนภาพคอมโพเนนท์ โดยส่วนคอมโพเนนท์หลัก ที่ซึ่งประกอบไปด้วยนอลเทอร์มินัล diagram components connectors และ attachments ตามลำดับ แสดงการตรวจสอบดังรูปที่ 4.12



รูปที่ 4.12 การตรวจสอบสายอักขระด้วยกฎ diagrams

จากรูปที่ 4.12 จำลองการนำเข้าสายอักขระโทเค็นที่ซึ่งอธิบายแทนด้วยเอกซ์เอ็มไอ โดยประกอบด้วย ส่วนแผนภาพคอมโพเนนท์ ตรวจสอบด้วยกฎ diagram ส่วนคอมโพเนนท์ ตรวจสอบ

ด้วยกฎ components และส่วนการเชื่อมและแอทแทชเมนท์ ตรวจสอบด้วยกฎ connectors และ attachments ตามลำดับ ซึ่งรายละเอียดของสายอักขระโทเค็นที่ตรวจสอบได้แสดงดังข้อต่อไป และแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ diagrams ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.13



รูปที่ 4.13 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ diagrams ในรูปแบบต้นไม้การแจงส่วน

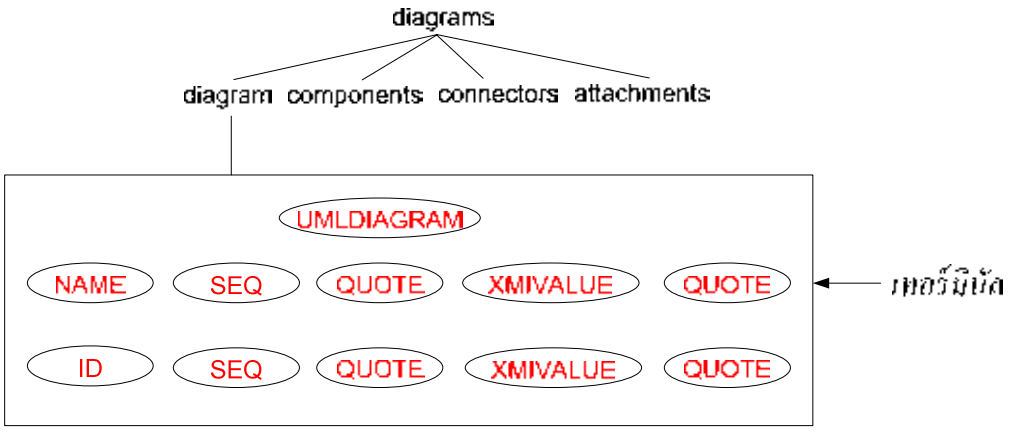
2) กฎ diagram ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ โดยส่วนแผนภาพคอมโพเนนท์ประกอบด้วยเทอร์มินัล แสดงการตรวจสอบดังรูปที่ 4.14

```
+ <uml:Diagram name="Proposal" xmi:id="AG8iHgKGAqFsAQO_" toolName="Visual Paradigm for
  UML" documentation="" diagramType="ComponentDiagram">
```

uml:Diagram	name	=	“	Proposal	”
UMLDIAGRAM	NAME	SEQ	QUOTE	XMIVALUE	QUOTE
	xmi:id	=	“	AG8iHgKGAqFsAQO_	”
	ID	SEQ	QUOTE	XMIVALUE	QUOTE

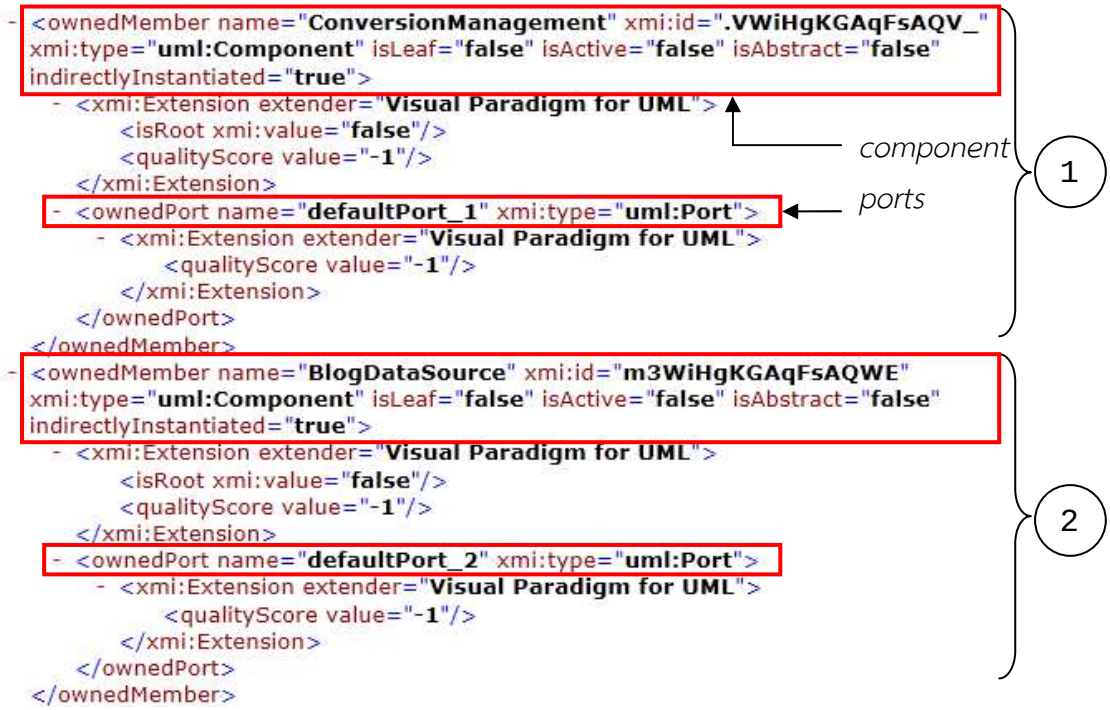
รูปที่ 4.14 การตรวจสอบสายอักขระด้วยกฎ diagram

จากรูปที่ 4.14 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ diagram ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.15



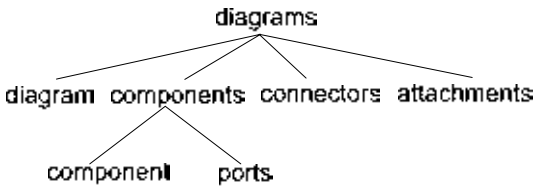
รูปที่ 4.15 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ diagram ในรูปแบบต้นไม้การแจงส่วน

3) กฎ components ตรวจสอบสายอักขระ ซึ่งเป็นคอมโพเนนท์ โดยมีมากกว่าหนึ่งคอมโพเนนท์ ที่ซึ่งประกอบไปด้วยนอลเทอร์มินัล component ports และ components ตามลำดับ แสดงการตรวจสอบดังรูปที่ 4.16



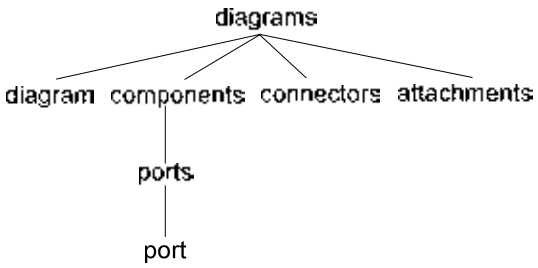
รูปที่ 4.16 การตรวจสอบสายอักขระด้วยกฎ components

จากรูปที่ 4.16 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ components ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.17



รูปที่ 4.17 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ components ในรูปแบบต้นไม้การแจงส่วน

4) กฎ ports ตรวจสอบสายอักขระ ซึ่งเป็นส่วนพอร์ต โดยประกอบด้วยนอลเทอร์มินัล port และ ports แสดงการตรวจสอบดังรูปที่ 4.16 และสามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ ports ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.18



รูปที่ 4.18 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ ports ในรูปแบบต้นไม้การแจงส่วน

5) กฎ port ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนพอร์ต โดยประกอบด้วยเทอร์มินัล ดังแสดงการตรวจสอบดังรูปที่ 4.19

```
- <ownedMember name="ConversionManagement" xmi:id=".VWiHgKGAqFsAQV_"
xmi:type="uml:Component" isLeaf="false" isActive="false" isAbstract="false"
indirectlyInstantiated="true">
  - <xmi:Extension extender="Visual Paradigm for UML">
      <isRoot xmi:value="false"/>
      <qualityScore value="-1"/>
    </xmi:Extension>
  - <ownedPort name="defaultPort_1" xmi:type="uml:Port">
      - <xmi:Extension extender="Visual Paradigm for UML">
```

1

uml:Port name=”defaultPort_1” xmi:id=”.VWiHgK..” xmi:type=”Usage” isPort=”true”

```
</ownedMember>
- <ownedMember name="BlogDataSource" xmi:id="m3WiHgKGAqFsAQWE"
xmi:type="uml:Component" isLeaf="false" isActive="false" isAbstract="false"
indirectlyInstantiated="true">
  - <xmi:Extension extender="Visual Paradigm for UML">
      <isRoot xmi:value="false"/>
      <qualityScore value="-1"/>
    </xmi:Extension>
  - <ownedPort name="defaultPort_2" xmi:type="uml:Port">
      - <xmi:Extension extender="Visual Paradigm for UML">
```

2

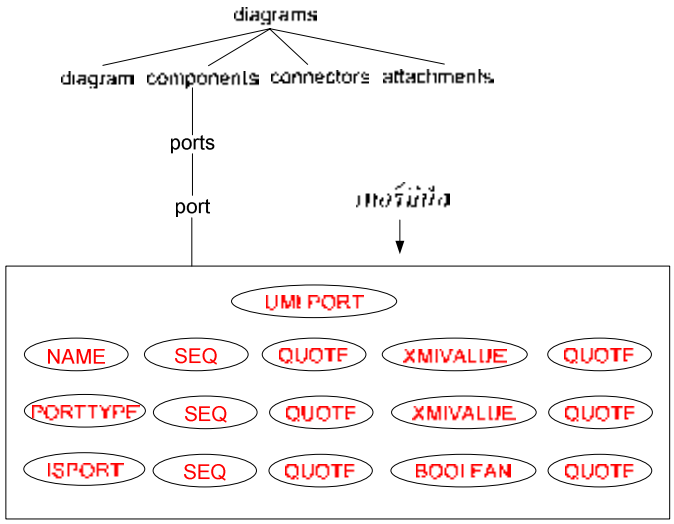
uml:Port name=”defaultPort_2” xmi:id=”m3WiHK..” xmi:type=”Realization” isPort=”true”

```
</ownedMember>
```

uml:Port	name	=	“	defaultPort_1/defaultPort_2	”
UMLPORT	NAME	SEQ	QUOTE	XMIVALUE	QUOTE
(ตรวจสอบด้วยแท็ก ownedPort และประเภท uml:Port)	xmi:type	=	“	Usage/Realization	”
	PORTTYPE	SEQ	QUOTE	XMIVALUE	QUOTE
	isPort	=	“	true	”
	ISPORT	SEQ	QUOTE	BOOLEAN	QUOTE

รูปที่ 4.19 การตรวจสอบสายอักขระด้วยกฎ port

จากรูปที่ 4.19 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ port ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.20



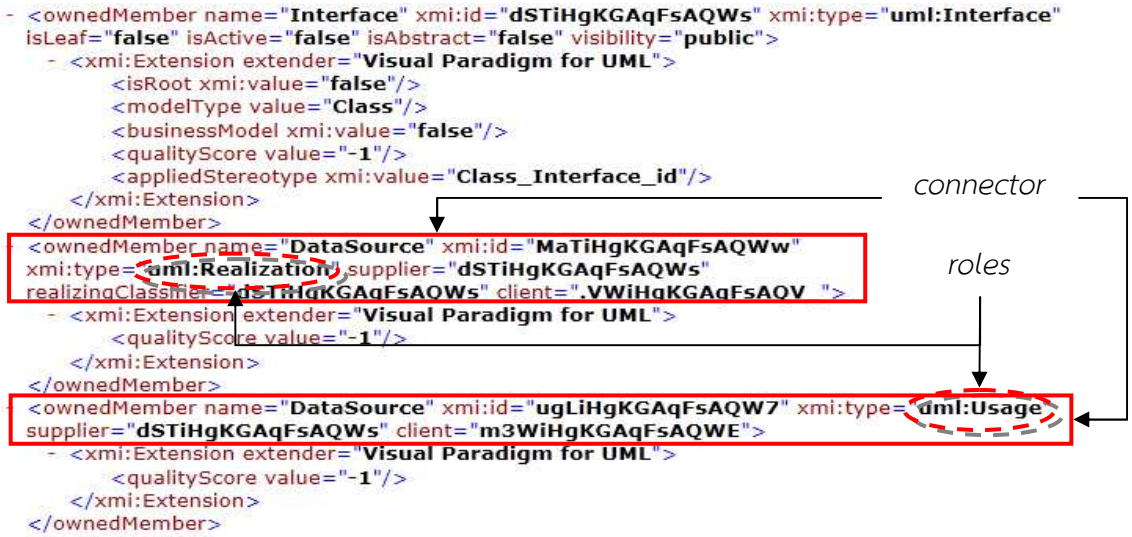
รูปที่ 4.20 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ port ในรูปแบบต้นไม้การแจงส่วน

6) กฎ component ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนคอมโพเนนท์ โดยประกอบด้วยเทอร์มินัล (การตรวจสอบไวยากรณ์จากรูปที่ 4.16) ดังแสดงผลการตรวจสอบสายอักระดังรูปที่ 4.21

uml:Component	name	=	"	ConversionManagment/ BlogDataSource	"
UMLCOMPONENT	NAME	SEQ	QUOTE	XMIVALUE	QUOTE
	xmi:id	=	"	.VWiHgKGAqFsAQV_/ m3WiHgKGAqFsAQWE	"
	ID	SEQ	QUOTE	XMIVALUE	QUOTE
	isLeaf	=	"	false	"
	ISLE	SEQ	QUOTE	BOOLEANF	QUOTE
	isActive	=	"	false	"
	ISAC	SEQ	QUOTE	BOOLEANF	QUOTE
	isAbstract	=	"	false	"
	ISAB	SEQ	QUOTE	BOOLEANF	QUOTE
	indirectlyIn stantiated	=	"	true	"
	INDINS	SEQ	QUOTE	BOOLEANT	QUOTE

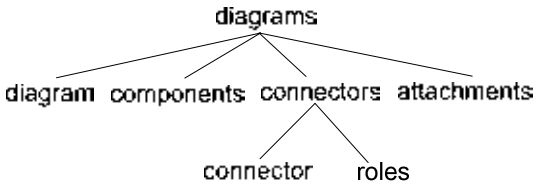
รูปที่ 4.21 การตรวจสอบสายอักขระด้วยกฎ component

7) กฎ connectors ตรวจสอบสายอักขระ ซึ่งเป็นส่วนการเชื่อมต่อ โดยประกอบด้วยนอลเทอร์มินัล connector และ roles แสดงการตรวจสอบดังรูปที่ 4.22



รูปที่ 4.22 การตรวจสอบสายอักขระด้วยกฎ connectors

จากรูปที่ 4.22 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ connectors ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.23



รูปที่ 4.23 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ connectors ในรูปแบบต้นไม้การแจงส่วน

8) กฎ connector ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วนการเชื่อมต่อประเภทเรียลไลเซชันและยูซเสจ โดยประกอบด้วยเทอร์มินัล (การตรวจสอบไวยากรณ์จากรูปที่ 4.22) แสดงผลการตรวจสอบสายอักระดังรูปที่ 4.24

uml:Connector	uml:Realization				
UMLCONNECTOR	REALIZATION				
(ตรวจสอบด้วยแท็ก ownedMember และประเภท uml:Realization)	xmi:id	=	"	MaTiHgKGAqFsAQWw	"
	ID	SEQ	QUOTE	XMIVALUE	QUOTE
	connector	=	"	assemblyConnector	"
	CONNECTOR	SEQ	QUOTE	XMIVALUE	QUOTE
	name	=	"	DataSource	"
	NAME	SEQ	QUOTE	XMIVALUE	QUOTE
	realizingClassifier	=	"	dSTiHgKGAqFsAQWs	"
	REALCLASS	SEQ	QUOTE	XMIVALUE	QUOTE
	supplier	=	"	dSTiHgKGAqFsAQWs	"
	SUPPLIER	SEQ	QUOTE	XMIVALUE	QUOTE
	client	=	"	.VWiHgKGAqFsAQV_	"
	CLIENT	SEQ	QUOTE	XMIVALUE	QUOTE

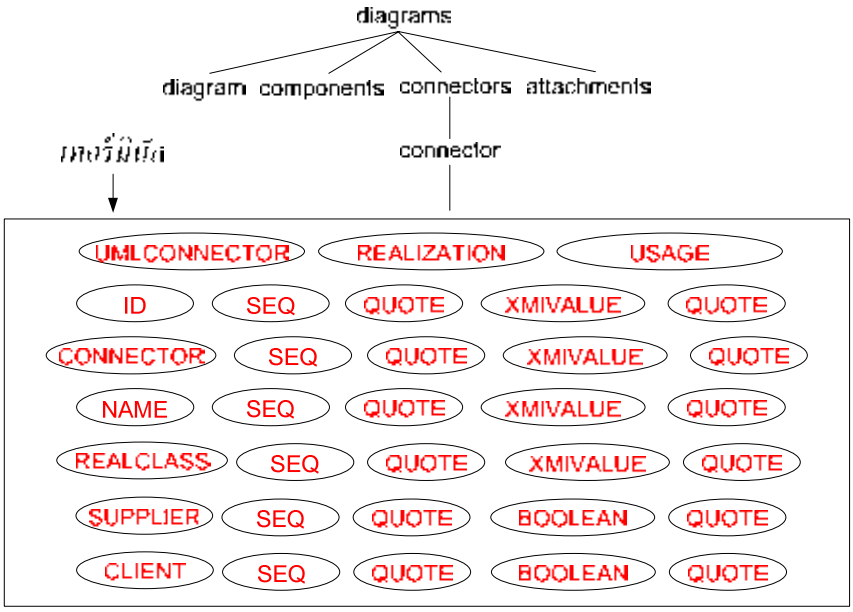
(ก) *การเชื่อมต่อประเภทเรียลไลเซชัน*

uml:Connector	uml:Usage				
UMLCONNECTOR	USAGE				
(ตรวจสอบด้วยแท็ก ownedMember และประเภท uml:Usage)	xmi:id	=	"	ugLiHgKGAqFsAQW7	"
	ID	SEQ	QUOTE	XMIVALUE	QUOTE
	connector	=	"	assemblyConnector	"
	CONNECTOR	SEQ	QUOTE	XMIVALUE	QUOTE
	name	=	"	DataSource	"
	NAME	SEQ	QUOTE	XMIVALUE	QUOTE
	supplier	=	"	dSTiHgKGAqFsAQWs	"
	SUPPLIER	SEQ	QUOTE	XMIVALUE	QUOTE
	client	=	"	m3WiHgKGAqFsAQWE	"
	CLIENT	SEQ	QUOTE	XMIVALUE	QUOTE

(ข) *การเชื่อมต่อประเภทยูซเสจ*

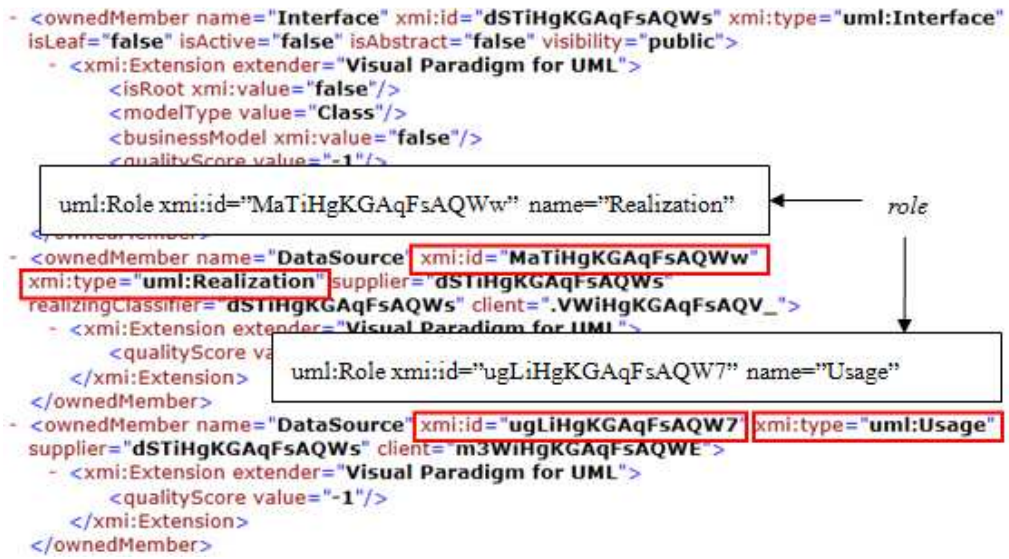
รูปที่ 4.24 การตรวจสอบสายอักขระด้วยกฎ connector

จากรูปที่ 4.24 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ connector ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.25



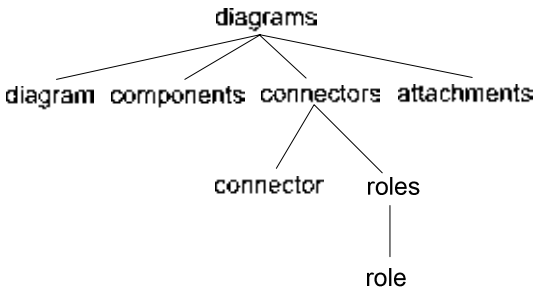
รูปที่ 4.25 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ connector ในรูปแบบต้นไม้การแจงส่วน

9) กฎ roles ตรวจสอบสายอักขระ ซึ่งเป็นส่วนบทบาท โดยประกอบด้วยนอลเทอร์มินัล role ดังแสดงการตรวจสอบดังรูปที่ 4.26



รูปที่ 4.26 การตรวจสอบสายอักขระด้วยกฎ roles

จากรูปที่ 4.26 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ roles ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.27



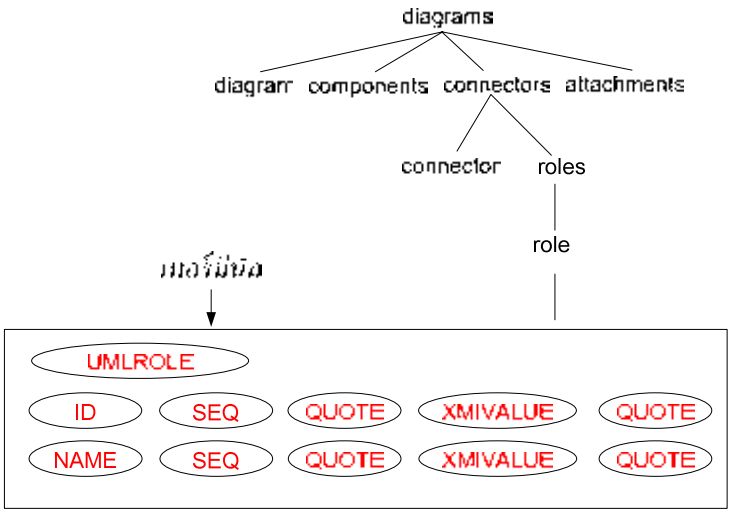
รูปที่ 4.27 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ roles ในรูปแบบต้นไม้การแจงส่วน

10) กฎ role ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำส่วนบทบาท โดยประกอบไปด้วยเทอร์มินัล (การตรวจสอบไวยากรณ์จากรูปที่ 4.26) แสดงผลการตรวจสอบสายอักระดังรูปที่ 4.28

uml:Role	xmi:id	=	“	MaTiHgKGAqFsAQWw/ ugLiHgKGAqFsAQW7	”
UMLROLE	ID	SEQ	QUOTE	XMIVALUE	QUOTE
	name	=	“	Realization/ Usage	”
	NAME	SEQ	QUOTE	XMIVALUE	QUOTE

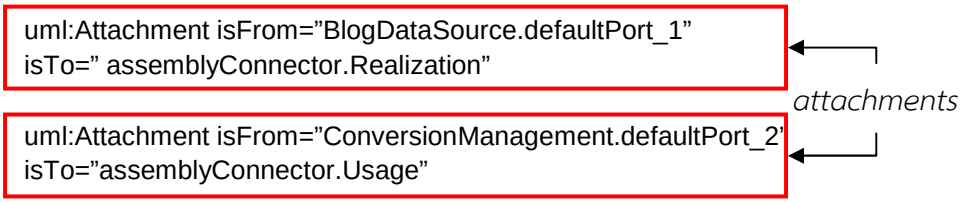
รูปที่ 4.28 การตรวจสอบสายอักขระด้วยกฎ role

จากรูปที่ 4.28 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ role ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.29



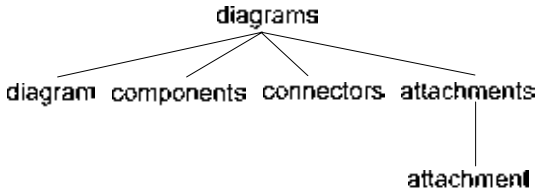
รูปที่ 4.29 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ role ในรูปแบบต้นไม้การแจงส่วน

11) กฎ attachments ตรวจสอบสายอักขระ ซึ่งเป็นส่วน attachment ที่ซึ่งประกอบไปด้วยนอลเทอร์มินัล attachment (ตรวจสอบไวยากรณ์จากรูปที่ 4.3) แสดงการตรวจสอบดังรูปที่ 4.30



รูปที่ 4.30 การตรวจสอบสายอักขระด้วยกฎ attachments

จากรูปที่ 4.30 สายอักระดังกล่าวเป็นการตรวจสอบจากส่วนคอมโพเนนท์ผ่านพอร์ตและส่งต่อไปยังส่วนการเชื่อมต่อ ที่ซึ่งทำงานผ่านบทบาทโดยสัมพันธ์กับส่วนการเชื่อมต่อ ดังแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ attachments ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.31



รูปที่ 4.31 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ attachments ในรูปแบบต้นไม้การแจงส่วน

12) กฎ attachment ตรวจสอบสายอักขระ ซึ่งเป็นลักษณะประจำ ส่วน attachment โดยประกอบด้วยเทอร์มินัล (การตรวจสอบไวยากรณ์จากรูปที่ 4.30) แสดงผลการตรวจสอบสายอักระดังรูปที่ 4.32

uml:Attachment	isFrom	=	"	BlogDataSource. defaultPort_1	"
UMLATTACHMENT	ISFROM	SEQ	QUOTE	XMIVALUE	QUOTE
	isTo	=	"	assemblyConnector. Realization	"
	ISTO	SEQ	QUOTE	XMIVALUE	QUOTE

(ก) *attachment สำหรับคอมโพเนนท์ "BlogDataSource"*

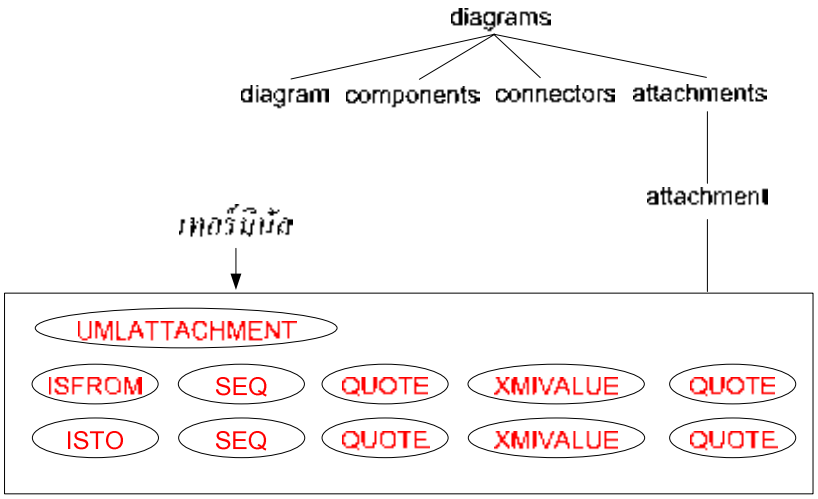
รูปที่ 4.32 (ก) ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ attachment

uml:Attachment	isFrom	=	“	ConversionManagement. defaultPort_2	”
UMLATTACHMENT	ISFROM	SEQ	QUOTE	XMIVALUE	QUOTE
	isTo	=	“	assemblyConnector. Usage	”
	ISTO	SEQ	QUOTE	XMIVALUE	QUOTE

(ข) *attachment สำหรับคอมโพเนนท์ “ConversionManagement”*

รูปที่ 4.32 (ข) ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ attachment

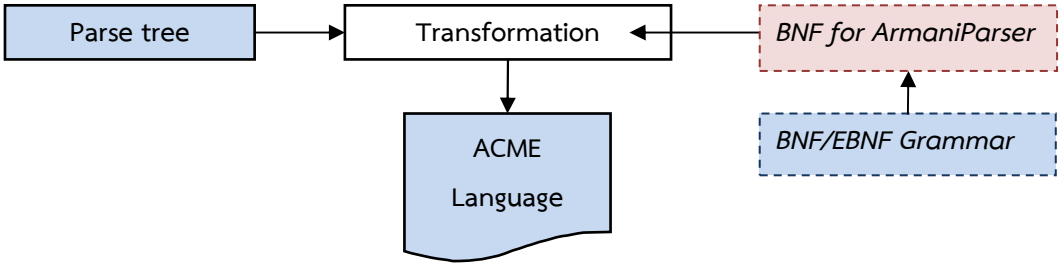
จากรูปที่ 4.32 สามารถแสดงผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ attachment ในรูปแบบต้นไม้การแจงส่วน ดังรูปที่ 4.33



รูปที่ 4.33 ผลลัพธ์การตรวจสอบสายอักขระด้วยกฎ attachment ในรูปแบบต้นไม้การแจงส่วน

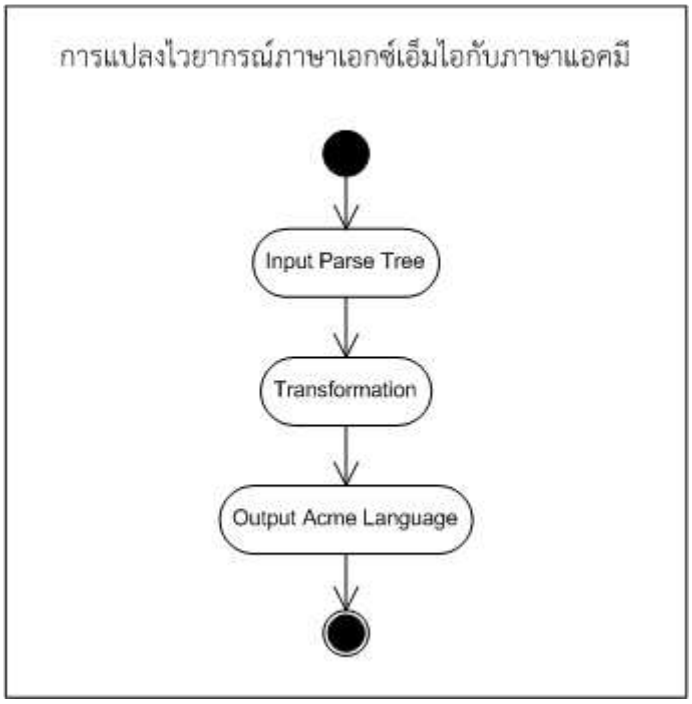
4.1.4 กระบวนการแปลงไวยากรณ์ภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี (Syntax Transformation)

จากข้อมูลผลลัพธ์กระบวนการที่ 4.1.3 จะใช้เป็นข้อมูลนำเข้าในกระบวนการนี้ เพื่อสร้างเป็นรหัสภาษาเป้าหมาย โดยนิยามกฎสำหรับการแปลงไวยากรณ์ทางภาษาระหว่างภาษาเอกซ์เอ็มไอกับภาษาแอคมี ด้วยกฎการตรวจสอบไวยากรณ์ของแผนภาพคอมโพเนนท์ เพื่อให้การแปลภาษามีความสอดคล้องตรงกันระหว่างสองภาษาแล้วนั้น จึงกำหนดกฎตั้งต้นดังอธิบายใน 3.4 แสดงกระบวนการแปลงไวยากรณ์ทางภาษาเอกซ์เอ็มไอกับภาษาแอคมี ดังรูปที่ 4.34



รูปที่ 4.34 การแปลงไวยากรณ์ภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี

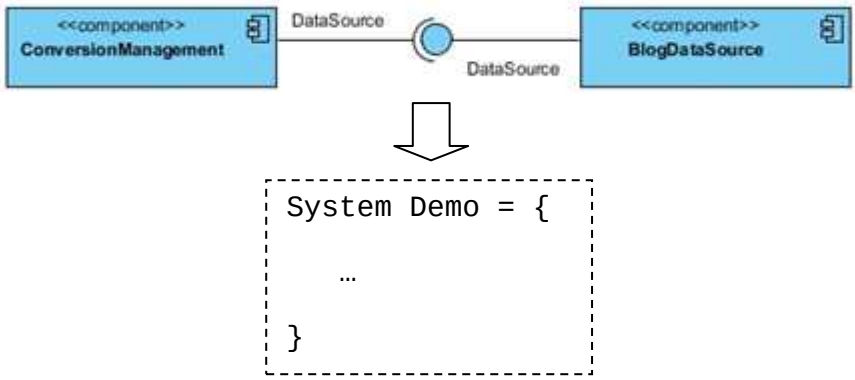
จากรูปที่ 4.34 สามารถแสดงแผนภาพกิจกรรมการแปลงไวยากรณ์ภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี ดังรูปที่ 4.35



รูปที่ 4.35 แผนภาพกิจกรรมการแปลงไวยากรณ์ภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี

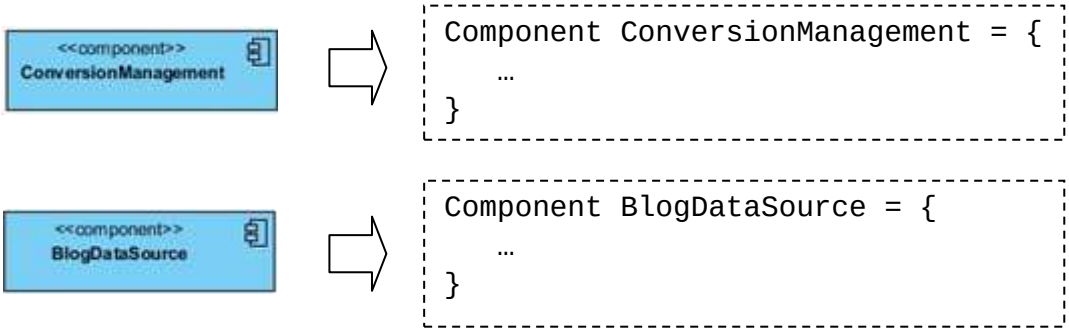
จากรูปที่ 4.35 แสดงกระบวนการแปลงไวยากรณ์ภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี ซึ่งนำเข้าข้อมูลในรูปแบบต้นไม้การแจงส่วน ที่ซึ่งผ่านการตรวจสอบคำศัพท์และการวิเคราะห์โครงสร้างไวยากรณ์ของแผนภาพคอมโพเนนท์ยูเอ็มแอล เพื่อนำเข้ากระบวนการแปลงไวยากรณ์ภาษาเอกซ์เอ็มไอไปยังภาษาแอคมี ซึ่งสามารถแสดงตัวอย่าง (ข้อมูลนำเข้าจากรูปที่ 4.3) ดังนี้

1) แผนภาพคอมโพเนนท์ การแปลงข้อมูลส่วนแผนภาพคอมโพเนนท์ไปยังภาษาแอคมี ที่ซึ่งประกอบด้วยข้อมูลโครงสร้างไวยากรณ์ภาษาแอคมี ดังรูปที่ 4.36



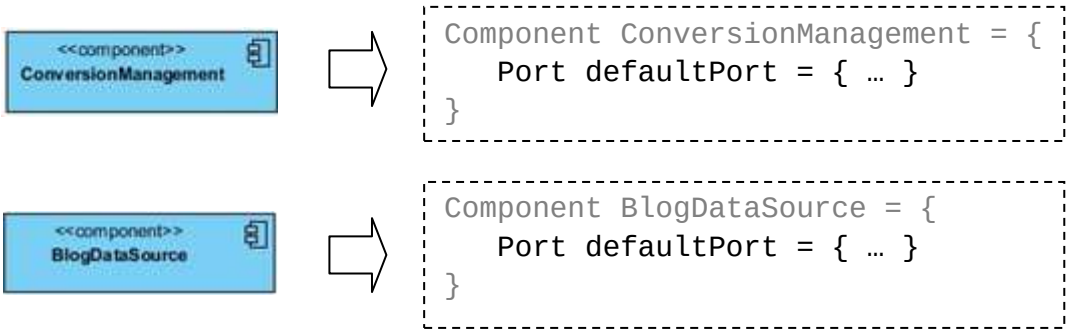
รูปที่ 4.36 การแปลงข้อมูลส่วนแผนภาพคอมโพเนนท์ไปยังภาษาแอคมี

2) คอมโพเนนท์ การแปลงข้อมูลส่วนคอมโพเนนท์ไปยังภาษาแอคมี ที่ซึ่งประกอบด้วยข้อมูลโครงสร้างไวยากรณ์ภาษาแอคมี ดังรูปที่ 4.37



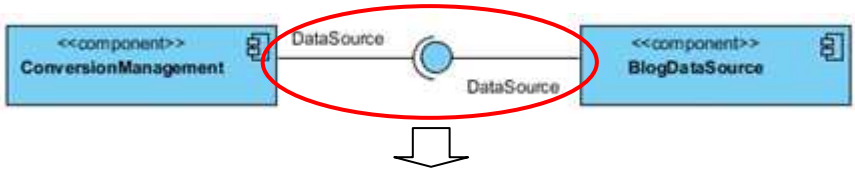
รูปที่ 4.37 การแปลงข้อมูลส่วนคอมโพเนนท์ไปยังภาษาแอคมี

3) พอร์ต การแปลงข้อมูลส่วนพอร์ตไปยังภาษาแอคมี ที่ซึ่งประกอบด้วยข้อมูลโครงสร้างไวยากรณ์ภาษาแอคมี เนื่องจากคอมโพเนนท์ดังกล่าวไม่ได้ระบุส่วนพอร์ต ดังนั้นเครื่องมือตัวแปลภาษาจึงได้ระบุพอร์ตในรูปแบบที่เป็นค่าพื้นฐานสำหรับคอมโพเนนท์ ดังรูปที่ 4.38



รูปที่ 4.38 การแปลงข้อมูลส่วนพอร์ตไปยังภาษาแอคมี

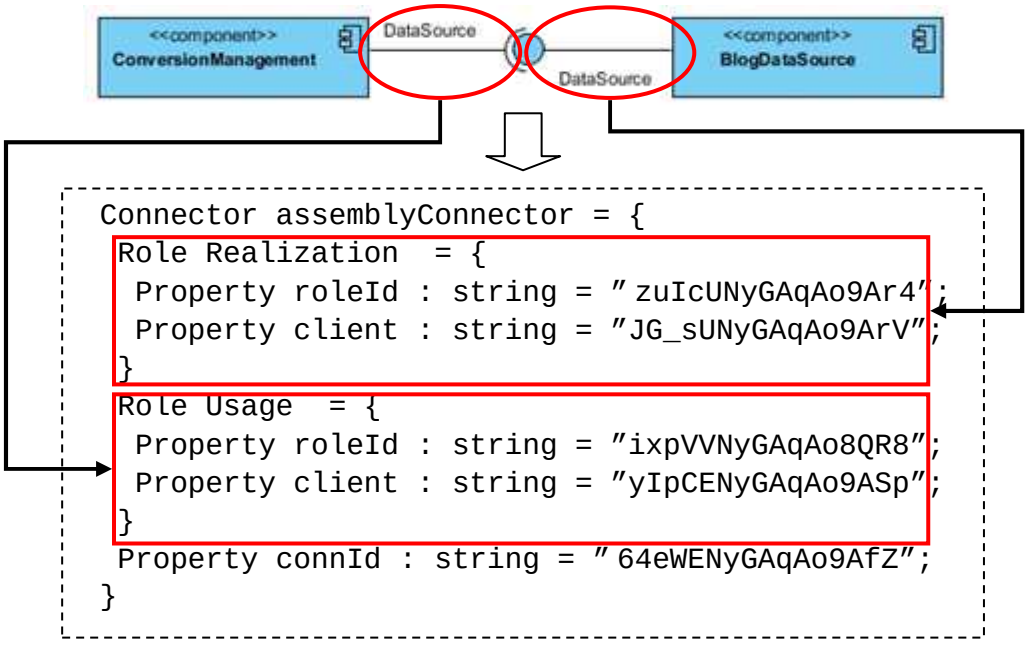
4) การเชื่อมต่อ การแปลงข้อมูลส่วนการเชื่อมต่อไปยังภาษาแอคมี ที่ซึ่งประกอบด้วยข้อมูลโครงสร้างไวยากรณ์ภาษาแอคมี ดังรูปที่ 4.39




```
Connector assemblyConnector = {
    Role Realization  = { }
    Role Usage  = { }
}
```

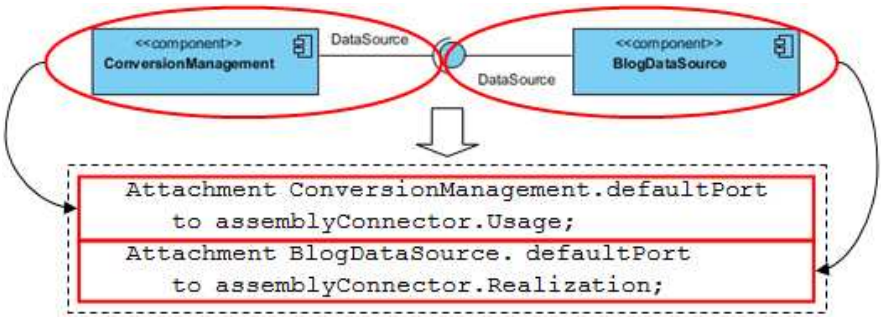
รูปที่ 4.39 การแปลงข้อมูลส่วนการเชื่อมต่อไปยังภาษาแอคมี

5) บทบาท การแปลงข้อมูลส่วนบทบาทไปยังภาษาแอคมี ที่ซึ่งประกอบด้วยข้อมูลโครงสร้างไวยากรณ์ภาษาแอคมี ดังรูปที่ 4.40



รูปที่ 4.40 การแปลงข้อมูลส่วนบทบาทไปยังภาษาแอคมี

6) แอทแทชเมนท์ การแปลงข้อมูลส่วนแอทแทชเมนท์และลักษณะประจำ สำหรับแอทแทชเมนท์ใดๆ ที่ซึ่งระบุถึงความสัมพันธ์ระหว่างคอมโพเนนท์และพอร์ต กับ การเชื่อมต่อและประเภทการเชื่อมต่อประกอบด้วยโครงสร้างไวยากรณ์ภาษาแอคมี ดังรูปที่ 4.41



รูปที่ 4.41 การแปลงข้อมูลส่วนแอทแทชเมนท์

4.1.5 กระบวนการสร้างข้อมูลผลลัพธ์ภาษาแอคมี (Code Generator)

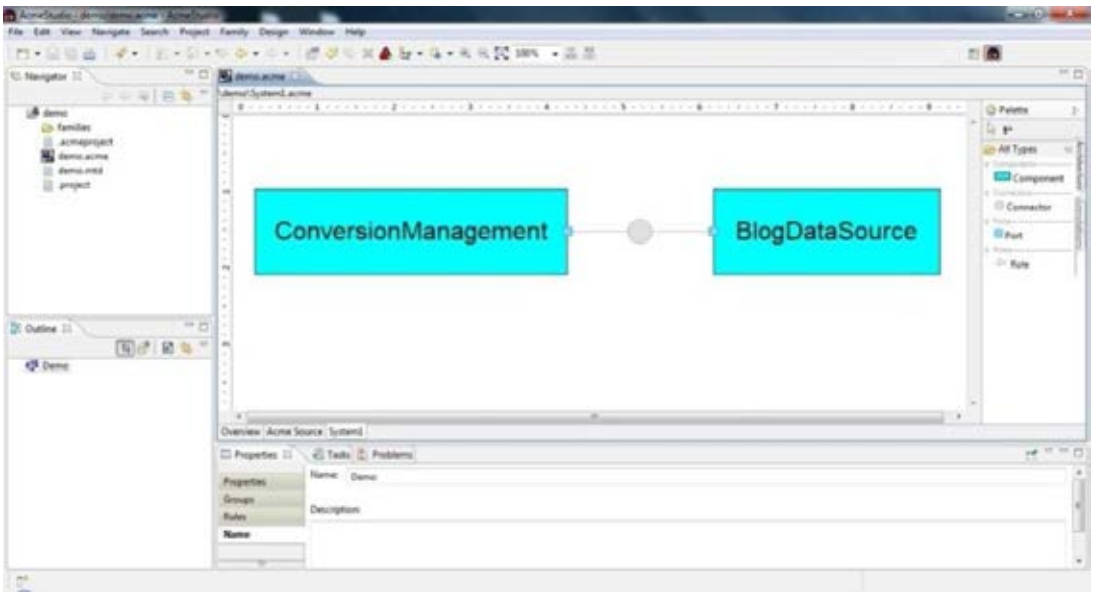
จากข้อมูลผลลัพธ์ในกระบวนการที่ 4.1.4 ที่ซึ่งแสดงเป็นแผนภาพคอมโพเนนท์ในรูปแบบของภาษาแอคมี ดังรูปที่ 4.42

```
System Demo = {
  Component ConversionManagement = {
    Property componentId : string = "yIpCENyGAqAo9ASp";
    Port defaultConversionManagement = {
        Property portId : string ="yIpCENyGAqAo9ASp";
        Property Usage : boolean = true; }
  }
  Component BlogDataSource = {
    Property componentId : string = "JG_sUNyGAqAo9ArV";
    Port defaultBlogDataSource = {
        Property portId : string ="JG_sUNyGAqAo9ArV";
        Property Realization : boolean = true; }
  }
  Connector assemblyConnector = {
    Property connectorId : string = "0gK.ENyGAqAo9AlM";
    Property connectorName : string = "DataSource";
    Role Usage = {
      Property roleId : string = "ixpVVNyGAqAo8QR8";
      Property client : string = "yIpCENyGAqAo9ASp"; }
    Role Realization = {
      Property roleId : string = "zuIcUNyGAqAo9Ar4";
      Property client : string = "JG_sUNyGAqAo9ArV"; }
  }
  Attachment BlogDataSource.defaultBlogDataSource to assemblyConnector.Realization;
  Attachment ConversionManagement.defaultConversionManagement to assemblyConnector.Usage;
}
```

รูปที่ 4.42 ข้อมูลผลลัพธ์แผนภาพคอมโพเนนท์ในรูปแบบของภาษาแอคมี

4.1.6 การนำข้อมูลผลลัพธ์ภาษาแอคมีไปแสดงผลบนเครื่องมือด้านสถาปัตยกรรม

การแสดงผลข้อมูลผลลัพธ์จากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ แสดงดังรูปที่ 4.43



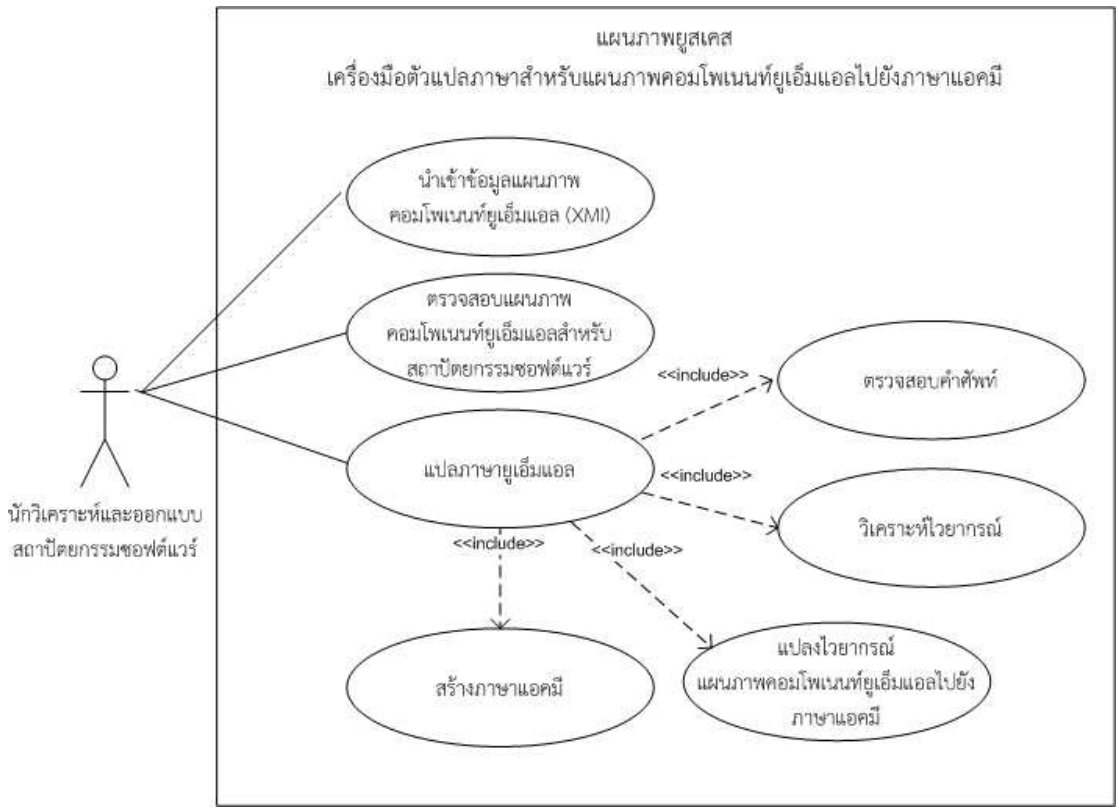
รูปที่ 4.43 ข้อมูลผลลัพธ์การแปลงภาษาเอกซ์เอ็มไอไปเป็นภาษาแอคมีด้วยเครื่องมือแอคมีสตูดิโอ

4.2 การออกแบบเครื่องมือตัวแปลภาษา

จากการวิเคราะห์และออกแบบกฎการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีดังกล่าวข้างต้น นำไปพัฒนาเครื่องมือตัวแปลภาษา สำหรับการออกแบบและพัฒนาเครื่องมือได้นำเสนอในแผนภาพยูสเคส แผนภาพคลาสและแผนภาพซีเควนซ์ อธิบายรายละเอียดดังนี้

4.2.1 แผนภาพยูสเคส

สำหรับแผนภาพยูสเคสอธิบายความสัมพันธ์ผู้ใช้งานกับระบบย่อย ที่ซึ่งอธิบายเครื่องมือตัวแปลภาษานั้น ประกอบด้วยหกยูสเคส ดังนี้ นำเข้าแผนภาพคอมโพเนนท์ยูเอ็มแอลในรูปแบบเอกซ์เอ็มไอ ตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ แปลภาษายูเอ็มแอล ตรวจสอบคำศัพท์ วิเคราะห์ไวยากรณ์ ปรับรวมโครงสร้างไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี และสร้างภาษาแอคมี อธิบายดังรูปที่ 4.44



รูปที่ 4.44 แผนภาพยูสเคสสำหรับเครื่องมือตัวแปลภาษา

จากรูปที่ 4.44 สามารถแสดงคำอธิบายยูสเคสดังกล่าว ดังตารางที่ 4.1-4.6

ตารางที่ 4.1 คำอธิบายยูสเคสนำเข้าแผนภาพคอมโพเนนท์ยูเอ็มแอลในรูปแบบเอกซ์เอ็มไอ

ชื่อยูสเคส	นำเข้าแผนภาพคอมโพเนนท์ยูเอ็มแอลในรูปแบบเอกซ์เอ็มไอ
ผู้กระทำ	นักวิเคราะห์และออกแบบสถาปัตยกรรมซอฟต์แวร์
รายละเอียด	นำเข้าเอกสารแผนภาพคอมโพเนนท์ยูเอ็มแอล ที่ซึ่งจัดเก็บในภาษาเอกซ์เอ็มไอ เพื่อนำไปใช้เป็นข้อมูลนำเข้าสำหรับกระบวนการตรวจสอบคำศัพท์
เงื่อนไขก่อนหน้า	เอกสารเอกซ์เอ็มไอดังกล่าวต้องนำออกจากเครื่องมือ Visual Paradigm รุ่นที่ 10.0 หรือใหม่กว่า
ขั้นตอน	1. ผู้ใช้เข้าสู่หน้าจอแรกในเครื่องมือตัวแปลภาษา 2. ผู้ใช้เลือกปุ่มนำเข้าเอกสาร 3. ผู้ใช้เลือกเอกสารเอกซ์เอ็มไอ 4. ผู้ใช้งานยืนยันการนำเข้าเอกสารเอกซ์เอ็มไอ
เงื่อนไขภายหลัง	ไม่ระบุ

ตารางที่ 4.2 คำอธิบายยูสเคสตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์

ชื่อยูสเคส	ตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับ สถาปัตยกรรมซอฟต์แวร์
ผู้กระทำ	นักวิเคราะห์และออกแบบสถาปัตยกรรมซอฟต์แวร์
รายละเอียด	นำเข้าข้อมูลคอมโพเนนท์ยูเอ็มแอลในรูปแบบสายอักขระอักษร ที่ซึ่งระบุเป็นองค์ประกอบหรือลักษณะประจำในแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ คืนค่าผลการตรวจสอบ
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าเป็นสายอักขระอักษรภาษาเอกซ์เอ็มไอ
ขั้นตอน	1. ตรวจสอบสายอักขระอักษรจากข้อมูลนำเข้า 2. ดำเนินการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ (ดังอธิบายในหัวข้อที่ 3.5) 3. คืนค่าผลการตรวจสอบ
เงื่อนไขภายหลัง	คืนค่าผลการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล สำหรับสถาปัตยกรรมซอฟต์แวร์

ตารางที่ 4.3 คำอธิบายยูสเคสแปลภาษายูเอ็มแอล

ชื่อยูสเคส	แปลภาษายูเอ็มแอล
ผู้กระทำ	นักวิเคราะห์และออกแบบสถาปัตยกรรมซอฟต์แวร์
รายละเอียด	นำเข้าข้อมูลคอมโพเนนท์ยูเอ็มแอลในรูปแบบสายอักขระอักษร ที่ซึ่งตรวจสอบด้วยการตรวจสอบคำศัพท์ การวิเคราะห์ไวยากรณ์และการปรับรวมโครงสร้าง คืนค่าผลการแปลงภาษาแอคมี
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าเป็นสายอักขระอักษร ซึ่งตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์
ขั้นตอน	1. นำเข้าข้อมูลแผนภาพคอมโพเนนท์ยูเอ็มแอลในรูปแบบสายอักขระอักษร 2. ดำเนินการแปลภาษายูเอ็มแอลจากแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี 3. คืนค่าผลภาษาแอคมี
เงื่อนไขภายหลัง	คืนค่าผลการแปลงแผนภาพคอมโพเนนท์ยูเอ็มแอลเป็นภาษาแอคมี

ตารางที่ 4.4 คำอธิบายยูสเคสตรวจสอบคำศัพท์

ชื่อยูสเคส	ตรวจสอบคำศัพท์
ผู้กระทำ	ไม่ระบุ
รายละเอียด	นำเข้าข้อมูลคอมโพเนนท์ยูเอ็มแอลที่จัดเก็บในภาษาเอกซ์เอ็มไอสำหรับกระบวนการตรวจสอบคำศัพท์ ที่ซึ่งระบุเป็นองค์ประกอบหรือลักษณะประจำในแผนภาพคอมโพเนนท์ยูเอ็มแอล คืนค่าสายอักขระโทคเค็น เพื่อนำไปใช้เป็นข้อมูลนำเข้าสำหรับกระบวนการวิเคราะห์ไวยากรณ์
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าผ่านการตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์
ขั้นตอน	1. ตรวจสอบคำศัพท์จากข้อมูลนำเข้า 2. ดำเนินการตรวจสอบคำศัพท์ สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเรกูลาร์เอกซ์เพรสชัน (ดังอธิบายในตารางที่ 3.3) 3. คืนค่าผลลัพธ์เป็นสายอักขระโทคเค็น
เงื่อนไขภายหลัง	คืนค่าผลลัพธ์สายอักขระโทคเค็น ที่ซึ่งแทนแผนภาพคอมโพเนนท์ยูเอ็มแอล

ตารางที่ 4.5 คำอธิบายยูสเคสวิเคราะห์ไวยากรณ์

ชื่อยูสเคส	วิเคราะห์ไวยากรณ์
ผู้กระทำ	ไม่ระบุ
รายละเอียด	นำเข้าข้อมูลคอมโพเนนท์ยูเอ็มแอลที่แทนด้วยสายอักขระโทคเค็น ซึ่งระบุองค์ประกอบหรือลักษณะประจำในแผนภาพคอมโพเนนท์ยูเอ็มแอล และวิเคราะห์ไวยากรณ์ แล้วคืนค่าไวยากรณ์ต้นไม้การแจงส่วน
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าผ่านการตรวจสอบคำศัพท์แผนภาพคอมโพเนนท์ยูเอ็มแอล
ขั้นตอน	1. วิเคราะห์ไวยากรณ์จากข้อมูลนำเข้า 2. ดำเนินการวิเคราะห์ไวยากรณ์ สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยกฎสำหรับตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ (ดังอธิบายในตารางที่ 3.4) 3. คืนค่าผลลัพธ์เป็นไวยากรณ์ต้นไม้การแจงส่วน
เงื่อนไขภายหลัง	คืนค่าผลลัพธ์ในรูปแบบไวยากรณ์ต้นไม้การแจงส่วน ที่ซึ่งแทนแผนภาพคอมโพเนนท์ยูเอ็มแอล

ตารางที่ 4.6 คำอธิบายยูสเคสแปลงไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี

ชื่อยูสเคส	แปลงไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี
ผู้กระทำ	ไม่ระบุ
รายละเอียด	นำเข้าข้อมูลคอมโพเนนท์ยูเอ็มแอลในรูปแบบไวยากรณ์ต้นไม้การแจงส่วน ที่ซึ่งระบุเป็นองค์ประกอบหรือลักษณะประจำสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอล ซึ่งคืนค่าผลลัพธ์โครงสร้างไวยากรณ์

ตารางที่ 4.6 คำอธิบายยูสเคสแปลงไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี (ต่อ)

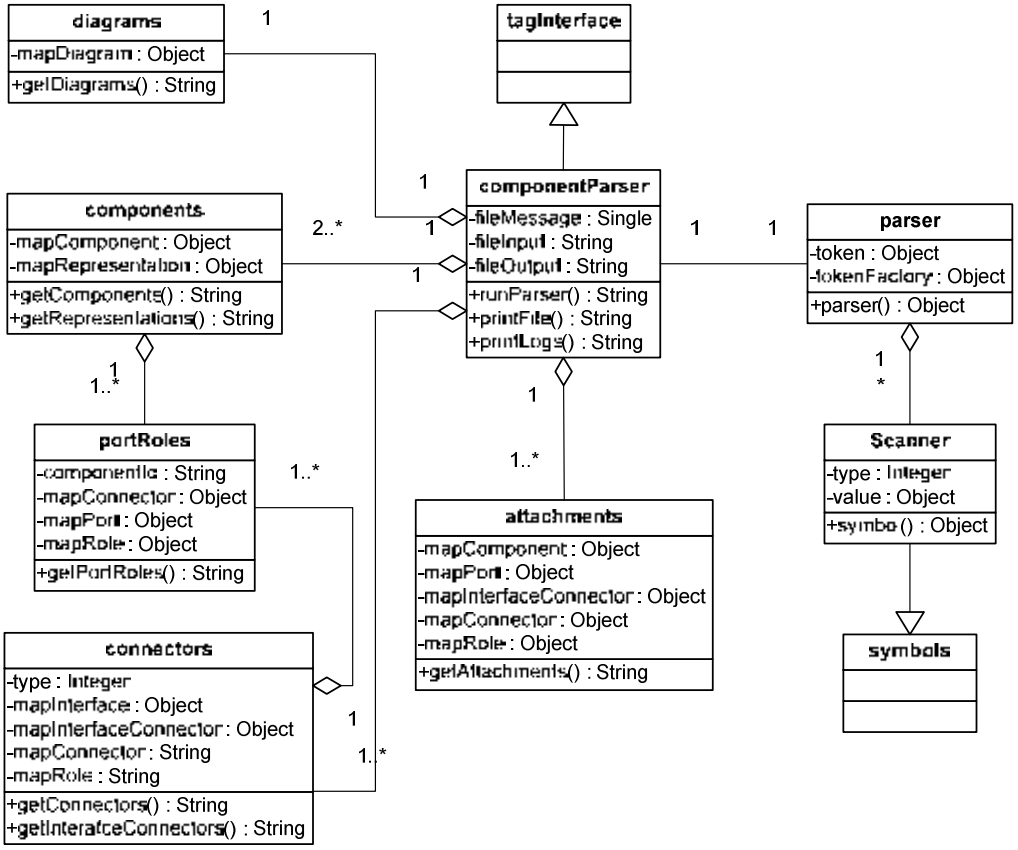
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าเป็นไวยากรณ์ต้นไม้การแจงส่วน ซึ่งวิเคราะห์ไวยากรณ์จากแผนภาพคอมโพเนนท์ยูเอ็มแอล
ขั้นตอน	1. ตรวจสอบไวยากรณ์ต้นไม้การแจงส่วนจากข้อมูลนำเข้า 2. ดำเนินการแปลงไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี (ดังอธิบายในตารางที่ 3.5) 3. คืนค่าโครงสร้างไวยากรณ์
เงื่อนไขภายหลัง	คืนค่าผลลัพธ์ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลเป็นภาษาแอคมี

ตารางที่ 4.7 คำอธิบายยูสเคสสร้างภาษาแอคมี

ชื่อยูสเคส	สร้างภาษาแอคมี
ผู้กระทำ	นักวิเคราะห์และออกแบบสถาปัตยกรรมซอฟต์แวร์
รายละเอียด	นำเข้าข้อมูลไวยากรณ์คอมโพเนนท์ยูเอ็มแอลในภาษาแอคมี ซึ่งคืนค่าผลลัพธ์คอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าเป็นไวยากรณ์ที่ปรับรวมโครงสร้าง จากแผนภาพคอมโพเนนท์ยูเอ็มแอลกับภาษาแอคมี
ขั้นตอน	1. นำเข้าข้อมูลไวยากรณ์คอมโพเนนท์ยูเอ็มแอลในภาษาแอคมี 2. คืนค่าผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลเป็นภาษาแอคมี
เงื่อนไขภายหลัง	แสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลเป็นภาษาแอคมี

4.2.2 แผนภาพคลาส

สำหรับแผนภาพคลาสอธิบายความสัมพันธ์ระหว่างคลาสภายในระบบรวมถึงโครงสร้างข้อมูลที่ซึ่งอธิบายเครื่องมือตัวแปลภาษานั้น ประกอบด้วย 10 คลาส ดังนี้ คลาส componentParser, คลาส diagrams, คลาส components, คลาส portRoles, คลาส connectors, คลาส attachments, คลาส tagInterface, คลาส parser, คลาส Scanner และคลาส symbols แสดงดังรูปที่ 4.45



รูปที่ 4.45 แผนภาพคลาสสำหรับเครื่องมือตัวแปลภาษา

จากรูปที่ 4.45 สามารถอธิบายความสัมพันธ์ระหว่างคลาส และรายละเอียดแต่ละคลาส ดังนี้

1) คลาส componentParser เป็นคลาสส่วนดำเนินการตรวจสอบเอกซ์เอ็มไอ เพื่อคัดกรองส่วนแท็กที่ไม่ได้ระบุเกี่ยวกับแผนภาพคอมโพเนนท์ยูเอ็มแอล และนำแท็กที่ระบุเกี่ยวกับแผนภาพคอมโพเนนท์ยูเอ็มแอล ที่ซึ่งเป็นข้อมูลนำเข้าในเครื่องมือตัวแปลภาษา
2) คลาส diagrams เป็นคลาสส่วนดำเนินการตรวจสอบเอกซ์เอ็มไอ เพื่อคัดกรองแท็กที่ระบุเกี่ยวกับแผนภาพคอมโพเนนท์ยูเอ็มแอล ที่ซึ่งเป็นข้อมูลนำเข้าในเครื่องมือตัวแปลภาษา
3) คลาส components เป็นคลาสส่วนดำเนินการตรวจสอบเอกซ์เอ็มไอ เพื่อคัดกรองแท็กที่ระบุเกี่ยวกับส่วนคอมโพเนนท์ ที่ซึ่งเป็นข้อมูลนำเข้าในเครื่องมือตัวแปลภาษา
4) คลาส portRoles เป็นคลาสส่วนดำเนินการตรวจสอบเอกซ์เอ็มไอ เพื่อคัดกรองแท็กที่ระบุเกี่ยวกับส่วนพอร์ตและบทบาท ที่ซึ่งเป็นข้อมูลนำเข้าในเครื่องมือตัวแปลภาษา
5) คลาส connectors เป็นคลาสส่วนดำเนินการตรวจสอบเอกซ์เอ็มไอ เพื่อคัดกรองแท็กที่ระบุเกี่ยวกับส่วนการเชื่อมต่อ ที่ซึ่งเป็นข้อมูลนำเข้าในเครื่องมือตัวแปลภาษา

6) คลาส attachments เป็นคลาสส่วนดำเนินการตรวจสอบเอกซ์เอ็มไอ เพื่อคัดกรองแท็กที่ระบุเกี่ยวกับส่วนแอชแทชเมนท์ ซึ่งระบุถึงความสัมพันธ์ระหว่างคอมโพเนนท์และพอร์ต กับ การเชื่อมต่อและประเภทการเชื่อมต่อ ที่ซึ่งเป็นข้อมูลนำเข้าในเครื่องมือตัวแปลภาษา
7) คลาส tagInterface เป็นอินเตอร์เฟสคลาสที่ระบุส่วนแท็กสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลที่ต้องตรวจสอบด้วยคลาส componentParser
8) คลาส parser เป็นคลาสที่ดำเนินการวิเคราะห์ไวยากรณ์สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลที่ดำเนินการผ่านเครื่องมือคัพ
9) คลาส Scanner เป็นคลาสที่ดำเนินการตรวจสอบคำศัพท์สำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลที่ดำเนินการผ่านเครื่องมือเจเฟล็กซ์
10) คลาส symbols เป็นอินเตอร์เฟสคลาสที่ระบุส่วนโทคเค็นสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลที่ต้องวิเคราะห์และตรวจสอบด้วยคลาส Scanner

4.3 การพัฒนาเครื่องมือตัวแปลภาษา

สำหรับการพัฒนาเครื่องมือตัวแปลภาษาอธิบายแยกเป็นสองส่วน โดยส่วนแรกกล่าวถึงสภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือและส่วนที่สองกล่าวถึงส่วนต่อประสานผู้ใช้งานเครื่องมือ ซึ่งมีรายละเอียดดังนี้

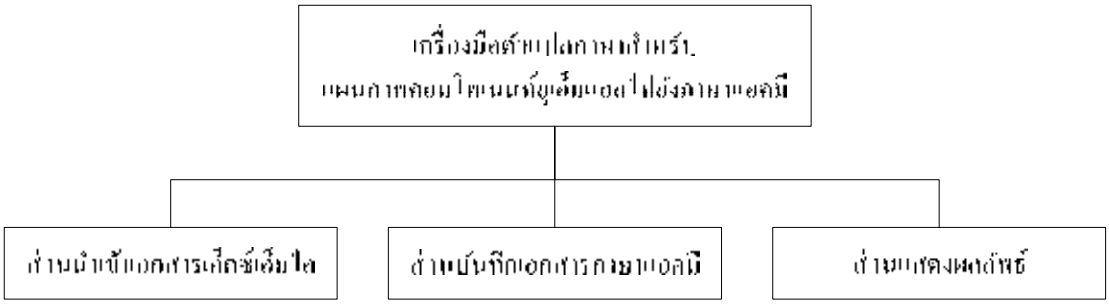
4.2.1 สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ

สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ มีรายละเอียดดังนี้

1) ฮาร์ดแวร์ (Hardware) เครื่องคอมพิวเตอร์โน้ตบุ๊ค (Notebook) ซึ่งมีคุณสมบัติดังนี้
   - หน่วยประมวลผลอินเทลคอร์ไอทรี ซึ่งมีความเร็วสัญญาณนาฬิกา 2.27 กิกะเฮิร์ท (Intel Core i3 CPU 2.27 GHz)
   - หน่วยความจำสำรอง (RAM) ขนาด 2.00 กิกะไบต์
   - ฮาร์ดดิสก์ (Hard Disk) ขนาด 500 กิกะไบต์
2) ซอฟต์แวร์ (Software)
   - ระบบปฏิบัติการวินโดว์เจ็ดชนิด 32 บิต (Windows 7 32-bit)
   - เครื่องมือเจเฟล็กซ์ รุ่น 1.4.3 และเครื่องมือคัพ รุ่น 11เอ (เบต้า)
   - เครื่องมือ Visual Paradigm รุ่น 10.0
   - เครื่องมือแอคมีสตูดิโอ รุ่น 3.5.4 (เบต้า)
   - ชุดเครื่องมือพัฒนาภาษาจาวา (JDK) รุ่น 1.7.0

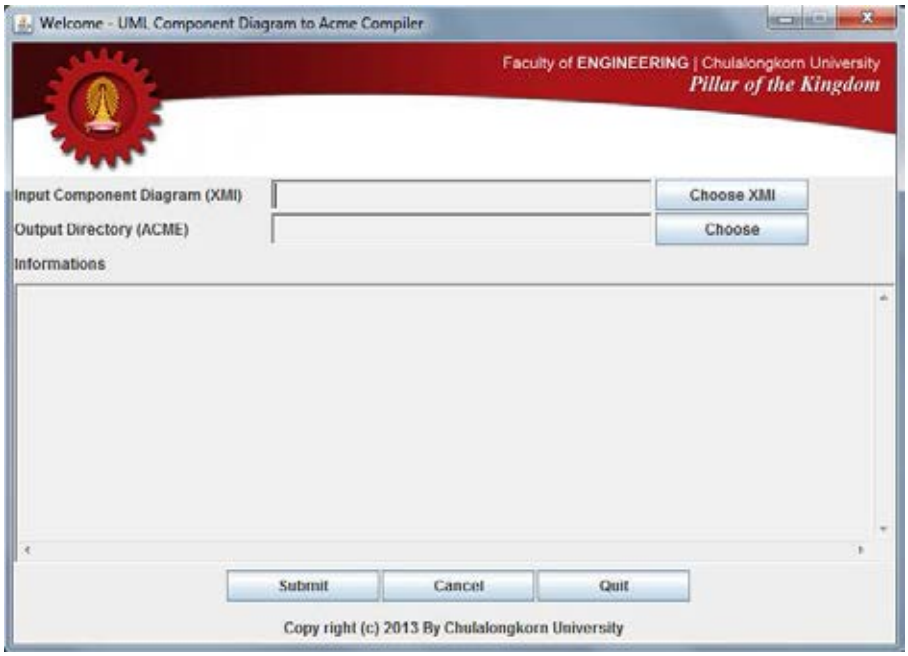
4.2.2 ส่วนต่อประสานกับผู้ใช้งานเครื่องมือ

ส่วนต่อประสานกับผู้ใช้งานเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีนั้น สามารถอธิบายโครงสร้างส่วนต่อประสานผู้ใช้งานเครื่องมือ ดังรูปที่ 4.46



รูปที่ 4.46 แผนภาพโครงสร้างส่วนต่อประสานผู้ใช้งานเครื่องมือ

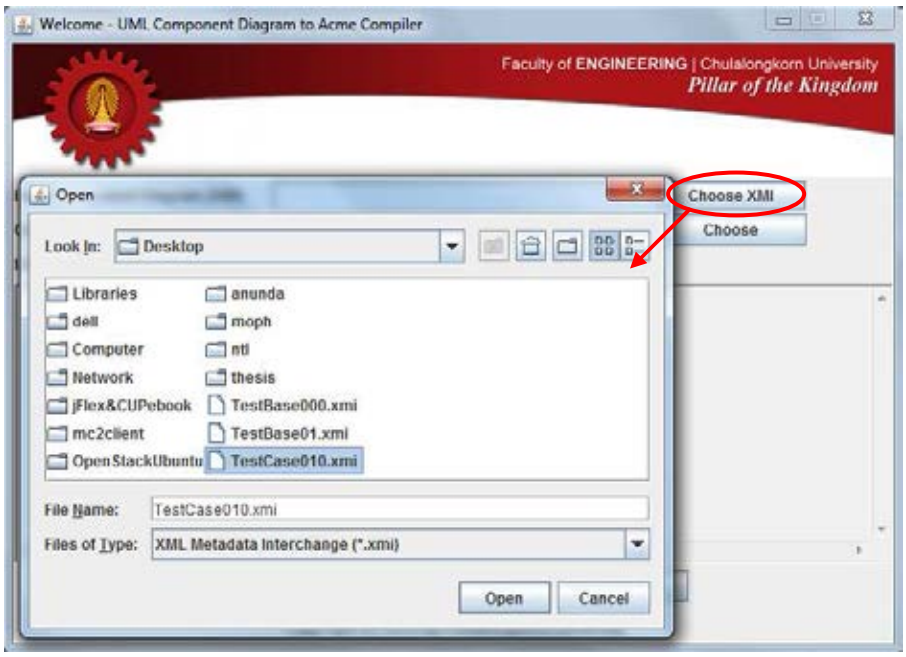
จากรูปที่ 4.46 โครงสร้างส่วนต่อประสานผู้ใช้งานเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีนั้นประกอบด้วย ส่วนประกอบ ดังรูปที่ 4.47



รูปที่ 4.47 หน้าจอส่วนต่อประสานผู้ใช้งานเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี

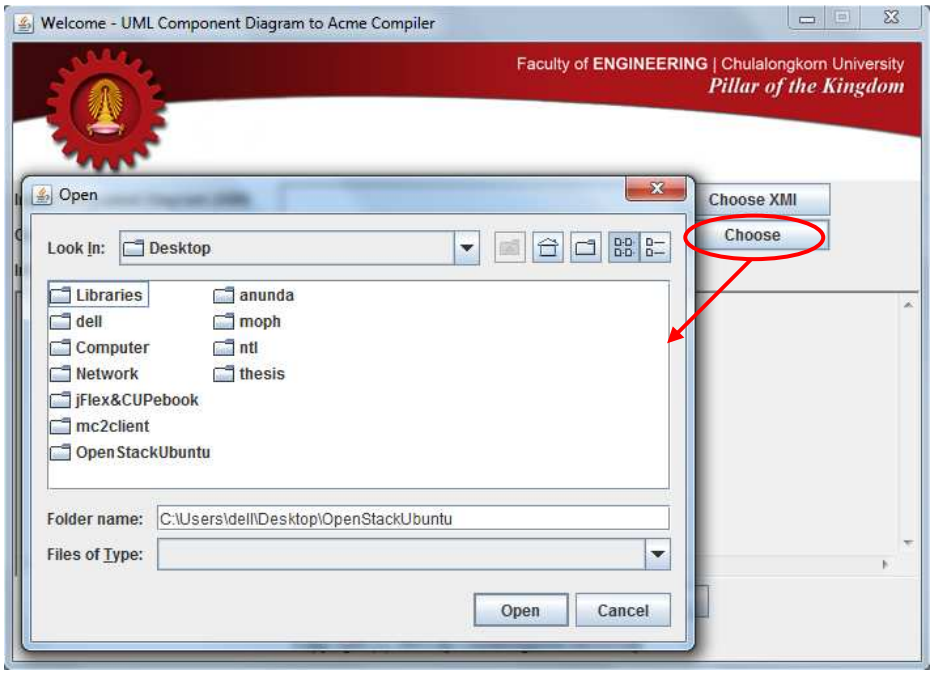
ซึ่งอธิบายรายละเอียดหน้าจอส่วนต่อประสานผู้ใช้งาน ดังนี้

1) ส่วนนำเข้าเอกสารเอกซ์เอ็มไอ เป็นส่วนที่เตรียมไว้สำหรับผู้ใช้เลือกนำเข้าเอกสารเอกซ์เอ็มไอ ด้วยการกดปุ่ม “Choose XMI” เพื่อเป็นข้อมูลนำเข้าสำหรับเครื่องมือดังกล่าวแสดงการใช้งาน ดังรูปที่ 4.48



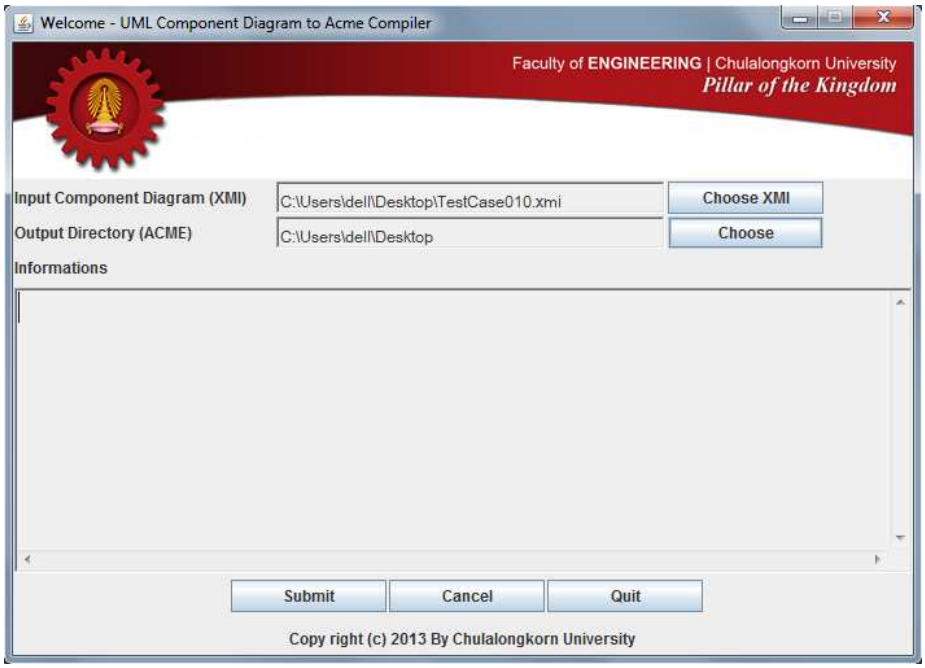
รูปที่ 4.48 หน้าจอส่วนต่อประสานผู้ใช้งานส่วนนำเข้าเอกสารเอกซ์เอ็มไอ

2) ส่วนบันทึกเอกสารภาษาแอคมี เป็นส่วนที่เตรียมไว้สำหรับผู้ใช้เลือกแฟ้มข้อมูลที่ต้องการจัดเก็บเอกสารภาษาแอคมี ด้วยการกดปุ่ม “Choose” เพื่อใช้เป็นส่วนการจัดเก็บข้อมูลผลลัพธ์ ซึ่งประมวลผลด้วยเครื่องมือดังกล่าว แสดงการใช้งานดังรูปที่ 4.49



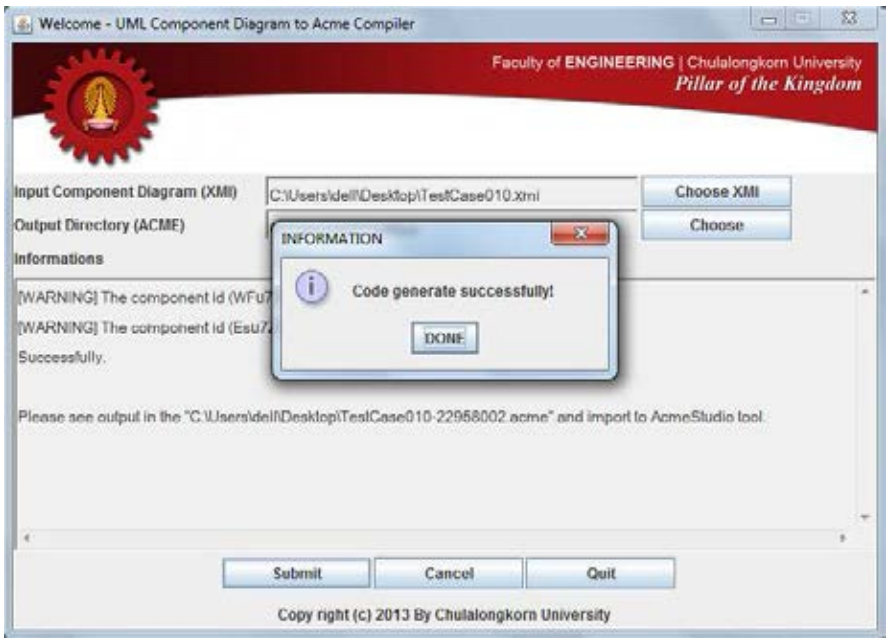
รูปที่ 4.49 หน้าจอส่วนต่อประสานผู้ใช้งานส่วนบันทึกเอกสารภาษาแอคมี

จากรูปที่ 4.48 และ 4.49 จะปรากฏข้อมูลเอกสารเอกซ์เอ็มไอและแฟ้มข้อมูลสำหรับการจัดเก็บผลลัพธ์ ดังรูปที่ 4.50

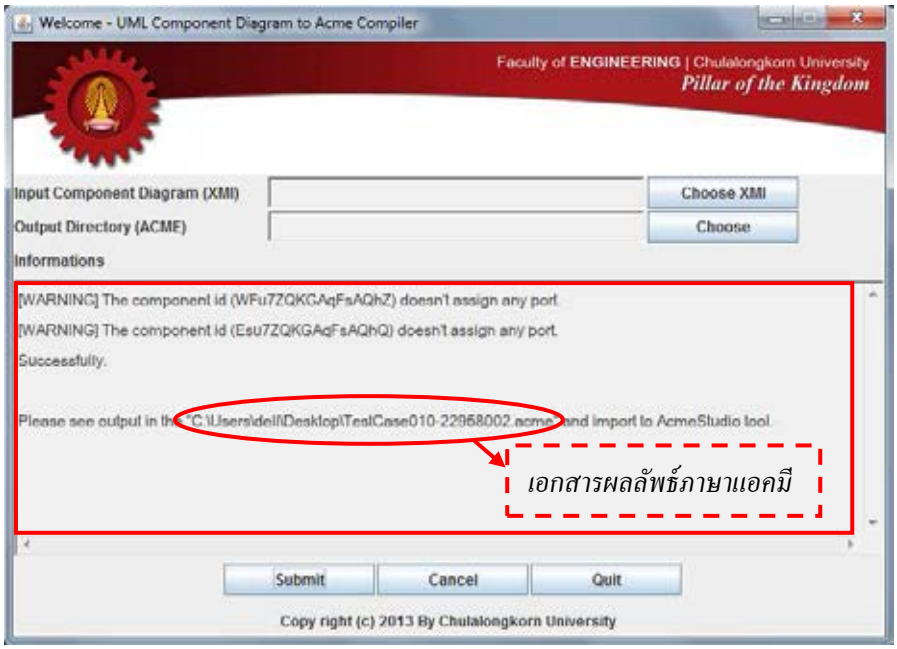


รูปที่ 4.50 หน้าจอส่วนต่อประสานผู้ใช้งานหลังจากระบุเอกสารเอกซ์เอ็มไอและแฟ้มข้อมูลสำหรับการจัดเก็บผลลัพธ์

3) ส่วนแสดงผลลัพธ์ เป็นส่วนที่แสดงผลข้อมูลสำหรับการประมวลผลเครื่องมือตัวแปลภาษา ด้วยการกดปุ่ม “Submit” ซึ่งแสดงผลการดำเนินงาน ข้อผิดพลาดและคำเตือนรวมทั้งข้อความที่เกิดจากการประมวลผลด้วยเครื่องมือดังกล่าวต่อผู้ใช้งาน แสดงการใช้งานดังรูปที่ 4.51 และ 4.52



รูปที่ 4.51 หน้าจอส่วนต่อประสานผู้ใช้งานส่วนแสดงผลการดำเนินการ



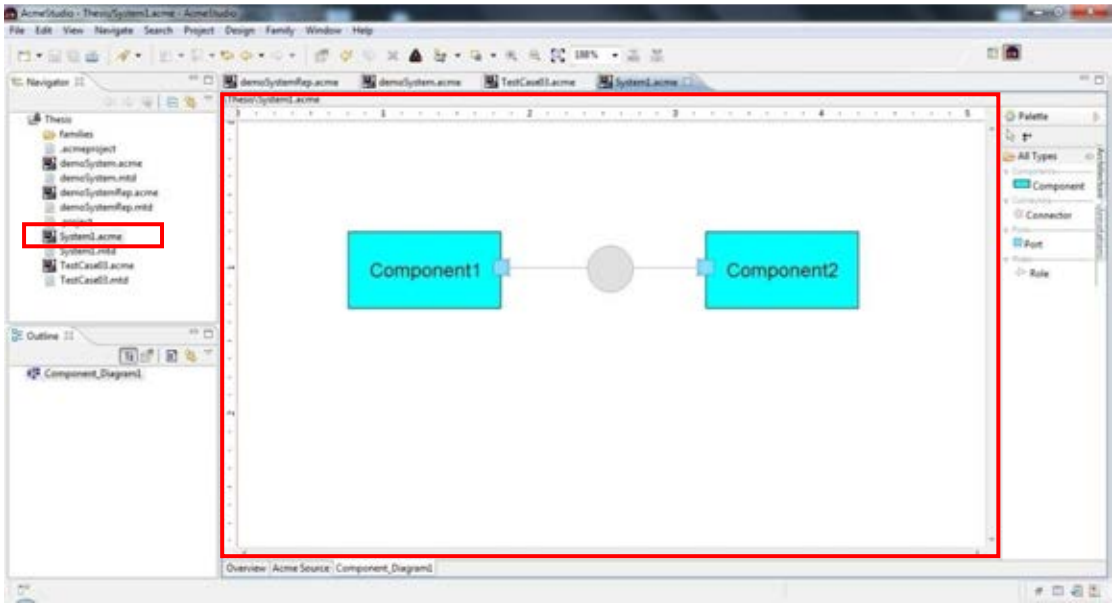
รูปที่ 4.52 หน้าจอส่วนต่อประสานผู้ใช้งานส่วนแสดงผลข้อผิดพลาดและคำเตือน

จากรูปที่ 4.52 แสดงคำเตือนต่อผู้ใช้งานว่าแผนภาพคอมโพเนนท์ยูเอ็มแอลไม่ได้ระบุพอร์ตสำหรับคอมโพเนนท์ใดๆ แต่เครื่องมือตัวแปลภาษาก็ดำเนินการดังกล่าวโดยถือว่าเป็นแผนภาพคอมโพเนนท์สำหรับสถาปัตยกรรมซอฟต์แวร์ เมื่อผู้ใช้ดูข้อมูลผลลัพธ์ดังเอกสาร “C:\Users\dell\Desktop\TestCase-010-22958002.acme” จะปรากฏข้อมูลภาษาแอคมีดังรูปที่ 4.53

```
System Component_Diagram1 = {
  Component Component2 = {
    Property componentId : string = "nTKQIYKGAqFsAQM6";
    Port defaultPort_1 = {
        Property portId : string ="MGtQIYKGAqFsAQNT";
        Property Usage : boolean = true;
    }
  }
  Component Component1 = {
    Property componentId : string = "S2yQIYKGAqFsAQMx";
    Port defaultPort_0 = {
        Property portId : string ="D31QIYKGAqFsAQNJ";
        Property Realization : boolean = true;
    }
  }
  Connector assemblyConnector = {
    Property connectorId : string = "rhLQIYKGAqFsAQNl";
    Property connectorName : string = "assemblyConnector";
    Role Usage_1 = {
      Property roleId : string ="rvPQIYKGAqFsAQN7";
      Property client : string ="MGtQIYKGAqFsAQNT";
    }
    Role Realization_0 = {
      Property roleId : string ="exLQIYKGAqFsAQNr";
      Property client : string ="D31QIYKGAqFsAQNJ";
    }
  }
  Attachment Component2.defaultPort_1 to assemblyConnector.Usage_1;
  Attachment Component1.defaultPort_0 to assemblyConnector.Realization_0;
}
```

รูปที่ 4.53 ข้อมูลผลลัพธ์เอกสารภาษาแอคมี

จากรูปที่ 4.53 เมื่อผู้ใช้นำข้อมูลผลลัพธ์เอกสารภาษาแอคมีไปแสดงผลบนเครื่องมือแอคมีสตูดิโอ สามารถแสดงผลดังรูปที่ 4.54



รูปที่ 4.54 ข้อมูลผลลัพธ์เอกสารภาษาแอคมีบนเครื่องมือแอคมีสตูดิโอ

บทที่ 5
การทดสอบเครื่องมือตัวแปลภาษา

สำหรับการทดสอบเครื่องมือตัวแปลภาษา เป็นการทดสอบการประมวลผลและตรวจสอบความถูกต้องเครื่องมือตัวแปลภาษาแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี โดยกล่าวถึงรายละเอียดการทดสอบและผลการทดสอบเครื่องมือ ดังนี้

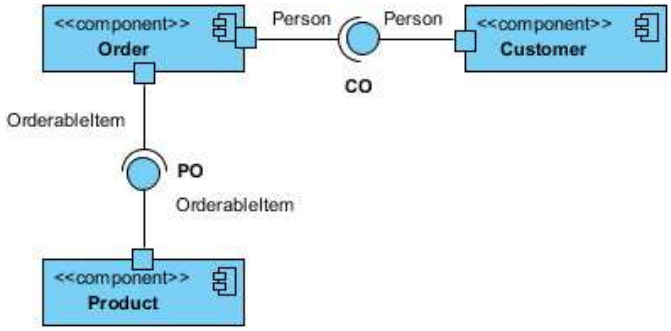
5.1 การทดสอบเครื่องมือตัวแปลภาษา

ในการทดสอบเครื่องมือดังกล่าว กำหนดให้ทดสอบด้วยกรณีทดสอบเจ็ดกรณีทดสอบ ซึ่งแยกตามลักษณะการทดสอบ ดังนี้

1) กรณีทดสอบที่หนึ่ง เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์จำนวนสามคอมโพเนนท์
2) กรณีทดสอบที่สอง เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์หลักและคอมโพเนนท์ย่อย รวมทั้งการเชื่อมต่อระหว่างคอมโพเนนท์หลักและระหว่างคอมโพเนนท์ย่อย
3) กรณีทดสอบที่สาม เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์จำนวน 20 คอมโพเนนท์
4) กรณีทดสอบที่สี่ เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์จำนวนหนึ่งคอมโพเนนท์และไม่ระบุการเชื่อมต่อ
5) กรณีทดสอบที่ห้า เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์จำนวนสามคอมโพเนนท์และมีหนึ่งการเชื่อมต่อ
6) กรณีทดสอบที่หก เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์จำนวนสองคอมโพเนนท์และไม่ระบุการเชื่อมต่อ
7) กรณีทดสอบที่เจ็ด เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์สามคอมโพเนนท์และระบุการเชื่อมต่อที่ถูกต้องและไม่ถูกต้องสำหรับสถาปัตยกรรมซอฟต์แวร์

5.1.1 การทดสอบกรณีทดสอบที่หนึ่ง

เป็นการทดสอบการวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพซึ่งประกอบด้วยคอมโพเนนท์จำนวนสามคอมโพเนนท์ โดยในกรณีทดสอบเป็นแผนภาพคอมโพเนนท์สำหรับสโตร์ (Store) ซึ่งมีส่วนของการเชื่อมต่อแบบแอสแซมบลี แสดงกรณีทดสอบในรูปแบบแผนภาพคอมโพเนนท์ ดังรูปที่ 5.1



รูปที่ 5.1 กรณีทดสอบที่หนึ่งสำหรับแผนภาพคอมโพเนนท์และการเชื่อมต่อแบบแอสแซมบลี [26]

จากรูปที่ 5.1 สามารถนำออกแผนภาพคอมโพเนนท์ ที่ซึ่งจัดเก็บเป็นภาษาเอกซ์เอ็มไอ เพื่อใช้เป็นข้อมูลนำเข้าสำหรับเครื่องมือตัวแปลภาษา ซึ่งเมื่อประมวลผลด้วยเครื่องมือตัวแปลภาษาแล้วจะได้ผลลัพธ์ดังรูปที่ 5.2

จากรูปที่ 5.1 และ 5.2 ที่ซึ่งวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเครื่องมือตัวแปลภาษา ซึ่งประกอบด้วยแผนภาพคอมโพเนนท์จำนวน 1 แผนภาพ คอมโพเนนท์จำนวน 3 คอมโพเนนท์ พอร์ตจำนวน 4 พอร์ตและการเชื่อมต่อแบบแอสแซมบลี 2 การเชื่อมต่อที่ซึ่งประกอบด้วยการเชื่อมต่อแบบเรียลไลเซชันกับยูซเสจ สรุปดังตารางที่ 5.1

ตารางที่ 5.1 สรุปการวิเคราะห์แผนภาพคอมโพเนนท์ยูเอ็มแอล สำหรับกรณีทดสอบที่หนึ่ง

ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอล	จำนวน	คำอธิบาย
Component Diagram	1	แผนภาพคอมโพเนนท์
Component	3	คอมโพเนนท์
Port	4	พอร์ต
Interface	2	อินเตอร์เฟส สำหรับระบุเป็นการเชื่อมต่อแบบแอสแซมบลี
Connector Usage	2	การเชื่อมต่อแบบยูซเสจ
Connector Realization	2	การเชื่อมต่อแบบเรียลไลเซชัน

```
System Store = {
    Component Order  = {
        Port defaultPort_1  = {
            Property portId : string = "VQQiXQKGAqFsAQOd";
            Property Usage : boolean = true;
        }
        Port defaultPort_0  = {
            Property portId : string = "VogiXQKGAqFsAQOS";
            Property Usage : boolean = true;
        }
        Property componentId : string = "wRNCXQKGAqFsAQNf";
    }
    Component Customer  = {
        Port defaultPort_2  = {
            Property portId : string = "PJwiXQKGAqFsAQOq";
            Property Realization : boolean = true;
        }
        Property componentId : string = "Rw9CXQKGAqFsAQNo";
    }
    Component Product  = {
        Port defaultPort_3  = {
            Property portId : string = "7_IiXQKGAqFsAQO3";
            Property Realization : boolean = true;
        }
        Property componentId : string = "8oTCXQKGAqFsAQN1";
    }
    Connector CO  = {
        Role Usage_3  = {
            Property roleId : string = "FPViXQKGAqFsAQSV";
            Property client : string = "VQQiXQKGAqFsAQOd";
        }
        Role Realization_1  = {
            Property roleId : string = "wOFiXQKGAqFsAQSG";
            Property client : string = "PJwiXQKGAqFsAQOq";
        }
        Property connectorId : string = "HmFiXQKGAqFsAQSC";
        Property connectorName : string = "CO";
    }
    Connector PO  = {
        Role Realization_0  = {
            Property roleId : string = "kAmiXQKGAqFsAQRB";
            Property client : string = "7_IiXQKGAqFsAQO3";
        }
        Role Usage_2  = {
            Property roleId : string = "anmiXQKGAqFsAQRM";
            Property client : string = "VogiXQKGAqFsAQOS";
        }
        Property connectorId : string = "9vGiXQKGAqFsAQQ9";
        Property connectorName : string = "PO";
    }
    Attachment Order.defaultPort_1 to CO.Usage_3;
    Attachment Order.defaultPort_0 to PO.Usage_2;
    Attachment Customer.defaultPort_2 to CO.Realization_1;
    Attachment Product.defaultPort_3 to PO.Realization_0;
}
```

รูปที่ 5.2 ผลลัพธ์เครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่หนึ่ง

สำหรับผลการทดสอบกรณีที่หนึ่ง แสดงผลการแปลงแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีด้วยเครื่องมือตัวแปลภาษา ซึ่งสามารถสรุปไวยากรณ์การแปลงแผนภาพคอมโพเนนท์ดังกล่าวดังอธิบายในตารางที่ 5.2

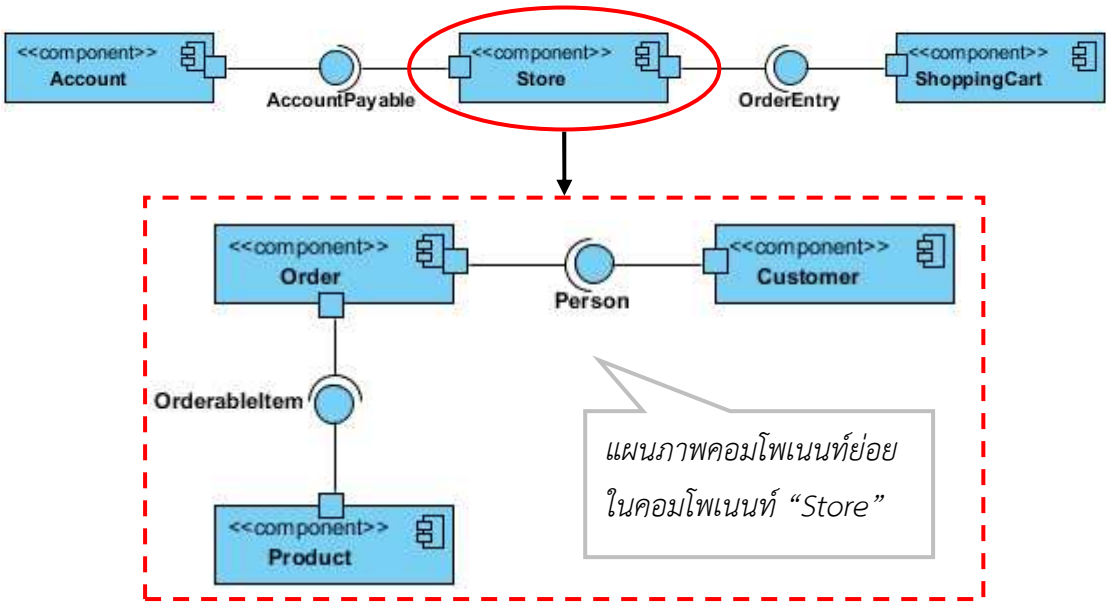
ตารางที่ 5.2 สรุปการแปลงไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่หนึ่ง

ไวยากรณ์		**คำอธิบาย**
ภาษาเอกซ์เอ็มไอ	**ภาษาแอคมี**	
Component Diagram	System	แผนภาพคอมโพเนนท์
Component	Component	คอมโพเนนท์
Port	Port	พอร์ต
Interface *(ระบุเป็นการเชื่อมต่อแอสแซมบลี)*	Connector	การเชื่อมต่อ ระบุด้วยอินเตอร์เฟสสำหรับการเชื่อมต่อแบบแอสแซมบลี
Connector Usage	Role	บทบาท ระบุด้วยการเชื่อมต่อ
Connector Realization	Role	บทบาท ระบุด้วยการเชื่อมต่อ

สามารถการแสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ ดังรูปที่ ก-1 (ภาคผนวก ก)

5.1.2 การทดสอบกรณีทดสอบที่สอง

กรณีทดสอบที่สอง เป็นการทดสอบการวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์หลักและคอมโพเนนท์ย่อย รวมทั้งการเชื่อมต่อระหว่างคอมโพเนนท์หลักและระหว่างคอมโพเนนท์ย่อย โดยในกรณีทดสอบเป็นแผนภาพคอมโพเนนท์สำหรับสโตร์ ที่ซึ่งมีส่วนของการเชื่อมต่อแบบแอสแซมบลีผ่านคอมโพเนนท์หลัก แสดงกรณีทดสอบในรูปแบบแผนภาพคอมโพเนนท์ดังรูปที่ 5.3



รูปที่ 5.3 กรณีทดสอบที่สองสำหรับแผนภาพคอมโพเนนท์ ซึ่งประกอบด้วยคอมโพเนนท์หลัก คอมโพเนนท์ย่อยและการเชื่อมต่อแบบแอสแซมบลี [26]

จากรูปที่ 5.3 สามารถนำออกแผนภาพคอมโพเนนท์ ที่ซึ่งจัดเก็บเป็นภาษาเอกซ์เอ็มไอ เพื่อใช้เป็นข้อมูลนำเข้าสำหรับเครื่องมือตัวแปลภาษา ซึ่งเมื่อประมวลผลด้วยเครื่องมือตัวแปลภาษาแล้วจะได้ผลลัพธ์ ดังรูปที่ 5.4

```
System TestCase002 = {
    Component Store  = {
        Port defaultPort_2  = {
            Property Usage : boolean = true;
            Property portId : string = "mC4qvwKGAqFsAQWn"; }
        Port defaultPort_1  = {
            Property Usage : boolean = true;
            Property portId : string = "H9wqvwKGAqFsAQWQ"; }
        Property componentId : string = "s2QqvwKGAqFsAQWK";

        Representation Store = {..
        }..
    }
    Component Account  = {
        Port defaultPort_0  = {
            Property Realization : boolean = true;
            Property portId : string = "IMvKvwKGAqFsAQVs"; }
        Property componentId : string = "jaXKvwKGAqFsAQVh";
    }
    Component ShoppingCart  = {
        Port defaultPort_11  = {
            Property Realization : boolean = true;
            Property portId : string = "tSsqvwKGAqFsAQXF"; }
        Property componentId : string = "XAUqvwKGAqFsAQW3";
    }
    Connector OrderEntry  = {
        Role Usage_5  = {
            Property roleId : string = "QdaqvwKGAqFsAQXg";
            Property client : string = "mC4qvwKGAqFsAQWn"; }
        Role Realization_1  = {
            Property roleId : string = "8S8qvwKGAqFsAQXV";
            Property client : string = "tSsqvwKGAqFsAQXF"; }
        Property connectorId : string = "Hi8qvwKGAqFsAQXR";
        Property connectorName : string = "OrderEntry";
    }
    Connector AccountPayable  = {
        Role Usage_4  = {
            Property roleId : string = "12oqvwKGAqFsAQWc";
            Property client : string = "H9wqvwKGAqFsAQWQ"; }
        Role Realization_0  = {
            Property roleId : string = "4aAqvwKGAqFsAQV_";
            Property client : string = "IMvKvwKGAqFsAQVs"; }
        Property connectorId : string = "aiAqvwKGAqFsAQV6";
        Property connectorName : string = "AccountPayable";
    }
    Attachment Store.defaultPort_2 to OrderEntry.Usage_5;
    Attachment Store.defaultPort_1 to AccountPayable.Usage_4;
    Attachment Account.defaultPort_0 to AccountPayable.Realization_0;
    Attachment ShoppingCart.defaultPort_11 to OrderEntry.Realization_1;
}
```

แผนภาพคอมโพเนนท์ย่อย ในคอมโพเนนท์ “Store”

รูปที่ 5.4 ผลลัพธ์เครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่สอง

จากรูปที่ 5.3 และ 5.4 ที่ซึ่งวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเครื่องมือตัวแปลภาษา ซึ่งประกอบด้วยแผนภาพคอมโพเนนท์จำนวน 1 แผนภาพ คอมโพเนนท์

จำนวน 3 คอมโพเนนท์ พอร์ตจำนวน 4 พอร์ตและการเชื่อมต่อแบบแอสแซมบลี 2 การเชื่อมต่อ และแผนภาพคอมโพเนนท์ย่อยจำนวน 1 แผนภาพ (ภายในคอมโพเนนท์หลัก “Store”) ซึ่งประกอบด้วยคอมโพเนนท์จำนวน 3 คอมโพเนนท์ พอร์ตจำนวน 4 พอร์ตและการเชื่อมต่อแบบแอสแซมบลี 2 การเชื่อมต่อ สรุปดังตารางที่ 5.3

ตารางที่ 5.3 สรุปการวิเคราะห์แผนภาพคอมโพเนนท์ยูเอ็มแอล สำหรับกรณีทดสอบที่สอง

ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอล	**จำนวน**	**คำอธิบาย**
Component Diagram	1	แผนภาพคอมโพเนนท์
Component	3	คอมโพเนนท์
Port	4	พอร์ต
Interface	2	อินเตอร์เฟส สำหรับระบุเป็นการเชื่อมต่อแบบแอสแซมบลี
Connector Usage	2	การเชื่อมต่อแบบยูซเสจ
Connector Realization	2	การเชื่อมต่อแบบเรียลไลเซชัน
Sub Component Diagram	**1**	**แผนภาพคอมโพเนนท์ย่อย**
Component	**3**	**คอมโพเนนท์ [*]**
Port	**4**	**พอร์ต [*]**
Interface	**2**	**อินเตอร์เฟส สำหรับระบุเป็นการเชื่อมต่อแบบแอสแซมบลี [*]**
Connector Usage	**2**	**การเชื่อมต่อแบบยูซเสจ [*]**
Connector Realization	**2**	**การเชื่อมต่อแบบเรียลไลเซชัน [*]**

หมายเหตุ * เป็นส่วนประกอบสำหรับแผนภาพคอมโพเนนท์ย่อย

สำหรับผลการทดสอบกรณีที่สอง แสดงผลการแปลงแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีด้วยเครื่องมือตัวแปลภาษา ซึ่งสามารถสรุปไวยากรณ์การแปลงแผนภาพคอมโพเนนท์ดังกล่าวดังอธิบายในตารางที่ 5.4

ตารางที่ 5.4 สรุปการแปลงไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่สอง

ไวยากรณ์		**คำอธิบาย**
ภาษาเอกซ์เอ็มไอ	**ภาษาแอคมี**	
Component Diagram	System	แผนภาพคอมโพเนนท์
Component	Component	คอมโพเนนท์
Port	Port	พอร์ต

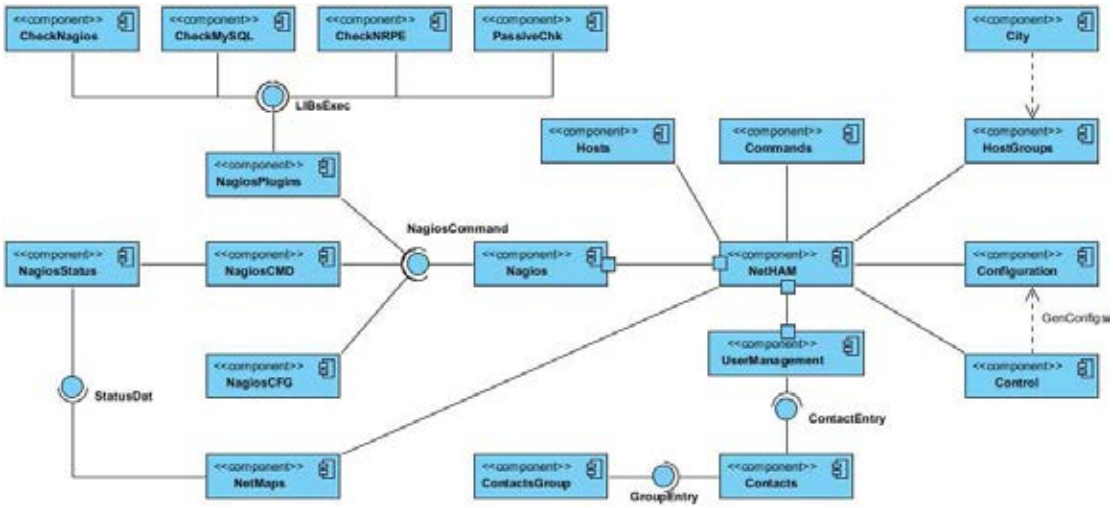
ตารางที่ 5.4 สรุปการแปลงไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่สอง (ต่อ)

ไวยากรณ์		คำอธิบาย
ภาษาเอกซ์เอ็มไอ	**ภาษาแอคมี**	
Interface *(ระบุเป็นการเชื่อมต่อแอสแซมบลี)*	Connector	การเชื่อมต่อ ระบุด้วยอินเตอร์เฟสสำหรับการเชื่อมต่อแบบแอสแซมบลี
Connector Usage	Role	บทบาท ระบุด้วยการเชื่อมต่อ
Connector Realization	Role	บทบาท ระบุด้วยการเชื่อมต่อ
Sub Component Diagram	**Representation**	**แผนภาพคอมโพเนนท์ย่อย**

สามารถการแสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ ดังรูปที่ ก-2

5.1.3 การทดสอบกรณีทดสอบที่สาม

กรณีทดสอบที่สาม เป็นการทดสอบการวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอล ส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์และการเชื่อมต่อระหว่างคอมโพเนนท์ 20 คอมโพเนนท์ โดยในกรณีทดสอบเป็นแผนภาพคอมโพเนนท์ตัวอย่างสำหรับระบบตรวจสอบสุขภาพเครือข่าย (Network Health and Monitoring System) แสดงกรณีทดสอบในรูปแบบแผนภาพคอมโพเนนท์ดังรูปที่ 5.5



รูปที่ 5.5 กรณีทดสอบที่สามสำหรับแผนภาพคอมโพเนนท์ที่ประกอบด้วยคอมโพเนนท์ 20 คอมโพเนนท์

จากรูปที่ 5.5 ที่ซึ่งวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเครื่องมือตัวแปลภาษา ซึ่งประกอบด้วยแผนภาพคอมโพเนนท์จำนวน 1 แผนภาพ คอมโพเนนท์จำนวน 20 คอมโพเนนท์ พอร์ตจำนวน 4 พอร์ตและพอร์ตพื้นฐาน 38 พอร์ต การเชื่อมต่อแบบแอสโซซิเอชัน 9 การเชื่อมต่อ การเชื่อมต่อแบบยูซเสจ 2 การเชื่อมต่อและการเชื่อมต่อแบบแอสแซมบลี 10 การเชื่อมต่อ สรุป ดังตารางที่ 5.5

ตารางที่ 5.5 สรุปการวิเคราะห์แผนภาพคอมโพเนนท์ยูเอ็มแอล สำหรับกรณีทดสอบที่สาม

ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอล	จำนวน	คำอธิบาย
Component Diagram	1	แผนภาพคอมโพเนนท์
Component	20	คอมโพเนนท์
Port (Default port)	4 (38)	พอร์ต (พอร์ตพื้นฐาน)
Interface	10	อินเตอร์เฟส สำหรับระบุเป็นการเชื่อมต่อแบบแอสแซมบลี
Connector Usage	12	การเชื่อมต่อแบบยูซเสจ
Connector Realization	10	การเชื่อมต่อแบบเรียลไลเซชัน
Connector Association	9	การเชื่อมต่อแบบแอสโซซิเอชัน

สำหรับผลการทดสอบกรณีที่สาม แสดงผลการแปลงแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีด้วยเครื่องมือตัวแปลภาษา ซึ่งสามารถสรุปไวยากรณ์การแปลงแผนภาพคอมโพเนนท์ดังกล่าวดังอธิบายในตารางที่ 5.2 สามารถการแสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ ดังรูปที่ ก-3

5.1.4 กรณีทดสอบที่สี่

กรณีทดสอบที่สี่ เป็นการทดสอบการวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์หนึ่งคอมโพเนนท์และไม่ระบุการเชื่อมต่อใดๆ โดยในกรณีทดสอบเป็นแผนภาพคอมโพเนนท์ที่ซึ่งตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ตามกฎตั้งต้น (นิยามไว้ในบทที่ 3) แสดงกรณีทดสอบในรูปแบบแผนภาพคอมโพเนนท์ ดังรูปที่ 5.6

รูปที่ 5.6 กรณีทดสอบที่สี่สำหรับแผนภาพคอมโพเนนท์ที่ประกอบด้วยคอมโพเนนท์หนึ่งคอมโพเนนท์

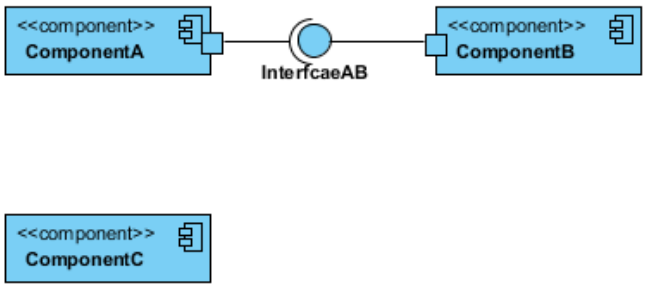
จากรูปที่ 5.6 กรณีทดสอบที่สี่ดำเนินการทดสอบเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี เพื่อตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับ

สถาปัตยกรรมซอฟต์แวร์ ซึ่งเมื่อประมวลผลด้วยเครื่องมือตัวแปลภาษาแล้วตรวจสอบพบว่าแผนภาพคอมโพเนนท์ดังกล่าวประกอบด้วยคอมโพเนนท์น้อยกว่าสองคอมโพเนนท์และไม่ได้ระบุการเชื่อมต่อใดๆ ดังนั้นเครื่องมือตัวแปลภาษารายงานเป็นข้อผิดพลาด ดังรูปที่ 5.7

รูปที่ 5.7 ผลการทดสอบเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่สี่

5.1.5 กรณีทดสอบที่ห้า

กรณีทดสอบที่ห้า เป็นการทดสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์จำนวนสามคอมโพเนนท์และมีหนึ่งการเชื่อมต่อ โดยในกรณีทดสอบเป็นแผนภาพคอมโพเนนท์ที่ซึ่งตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ตามกฎตั้งต้น แสดงกรณีทดสอบในรูปแบบแผนภาพคอมโพเนนท์ ดังรูปที่ 5.8



รูปที่ 5.8 กรณีทดสอบที่ห้าสำหรับแผนภาพคอมโพเนนท์ที่ประกอบด้วยคอมโพเนนท์สามคอมโพเนนท์และมีหนึ่งการเชื่อมต่อ

จากรูปที่ 5.8 กรณีทดสอบที่ห้าดำเนินการทดสอบเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี เพื่อตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ซึ่งเมื่อประมวลผลด้วยเครื่องมือตัวแปลภาษาแล้วตรวจสอบพบว่าแผนภาพคอมโพเนนท์ดังกล่าวประกอบด้วยคอมโพเนนท์อย่างน้อยสองคอมโพเนนท์และระบุการเชื่อมต่ออย่างน้อยหนึ่งการเชื่อมต่อ ดังนั้นเมื่อประมวลผลด้วยเครื่องมือตัวแปลภาษาแล้วจะได้ผลลัพธ์ ดังรูปที่ 5.9

```
System TestCase005 = {
  Component ComponentC = {
    Property componentId : string = "5kPcTIKGAqFsAQdA";
  }
  Component ComponentB = {
    Property componentId : string = "w6XcTIKGAqFsAQcz";
    Port defaultPort_1 = {
        Property portId : string ="af_cTIKGAqFsAQdf";
        Property Realization : boolean = true;
    }
  }
  Component ComponentA = {
    Property componentId : string = "LfHcTIKGAqFsAQcq";
    Port defaultPort_0 = {
        Property portId : string ="ObfcTIKGAqFsAQdV";
        Property Usage : boolean = true;
    }
  }
  Connector InterfcaeAB = {
    Property connectorId : string = "QXQ8TIKGAqFsAQdx";
    Property connectorName : string = "InterfcaeAB";
    Role Realization_0 = {
      Property roleId : string ="ePQ8TIKGAqFsAQd3";
      Property client : string ="af_cTIKGAqFsAQdf";
    }
    Role Usage_1 = {
      Property roleId : string ="bkk8TIKGAqFsAQeF";
      Property client : string ="ObfcTIKGAqFsAQdV";
    }
  }
  Attachment ComponentB.defaultPort_1 to InterfcaeAB.Realization_0;
  Attachment ComponentA.defaultPort_0 to InterfcaeAB.Usage_1;
}
```

รูปที่ 5.9 ผลลัพธ์เครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีสำหรับกรณีทดสอบที่ห้า

จากรูปที่ 5.9 สามารถวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลด้วยเครื่องมือตัวแปลภาษา ซึ่งประกอบด้วยแผนภาพคอมโพเนนท์จำนวน 1 แผนภาพ คอมโพเนนท์จำนวน 3 คอมโพเนนท์ พอร์ตจำนวน 2 พอร์ตและการเชื่อมต่อแบบแอสแซมบลี 1 การเชื่อมต่อสรุปดังตารางที่ 5.6

ตารางที่ 5.6 สรุปการวิเคราะห์แผนภาพคอมโพเนนท์ยูเอ็มแอล สำหรับกรณีทดสอบที่ห้า

ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอล	จำนวน	คำอธิบาย
Component Diagram	1	แผนภาพคอมโพเนนท์
Component	3	คอมโพเนนท์
Port	2	พอร์ต
Interface	1	อินเตอร์เฟส สำหรับระบุเป็นการเชื่อมต่อแบบแอสแซมบลี
Connector Usage	1	การเชื่อมต่อแบบยูซเสจ
Connector Realization	1	การเชื่อมต่อแบบเรียลไลเซชัน

สำหรับผลการทดสอบกรณีที่ห้า แสดงผลการแปลงแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมีด้วยเครื่องมือตัวแปลภาษา ซึ่งสามารถสรุปไวยากรณ์การแปลงแผนภาพคอมโพเนนท์ดังกล่าวดังอธิบายในตารางที่ 5.2 สามารถการแสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษาด้วยเครื่องมือแอคมีสตูดิโอ ดังรูปที่ ก-4

5.1.6 กรณีทดสอบที่หก

กรณีทดสอบที่หก เป็นการทดสอบการวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์สองคอมโพเนนท์และไม่ระบุการเชื่อมต่อใดๆ โดยในกรณีทดสอบเป็นแผนภาพคอมโพเนนท์ที่ซึ่งตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ตามกฎตั้งต้น แสดงกรณีทดสอบในรูปแบบแผนภาพคอมโพเนนท์ ดังรูปที่ 5.10

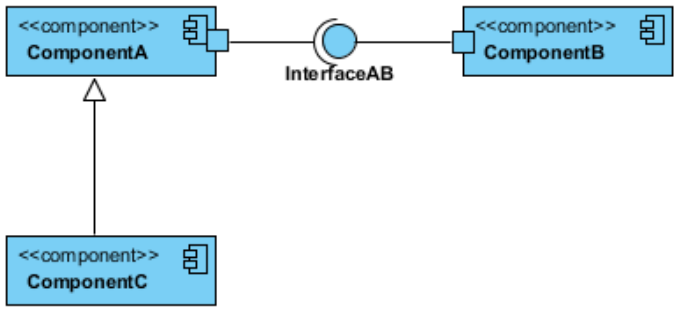
รูปที่ 5.10 กรณีทดสอบที่หกสำหรับแผนภาพคอมโพเนนท์ที่ประกอบด้วยคอมโพเนนท์สองคอมโพเนนท์และไม่ระบุการเชื่อมต่อ

จากรูปที่ 5.10 กรณีทดสอบที่หกดำเนินการทดสอบเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี เพื่อตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ซึ่งเมื่อประมวลผลด้วยเครื่องมือตัวแปลภาษาแล้วตรวจสอบพบว่าแผนภาพคอมโพเนนท์ดังกล่าวประกอบด้วยคอมโพเนนท์ตั้งแต่สองคอมโพเนนท์ขึ้นไป แต่ไม่ได้ระบุการเชื่อมต่อใดๆ ดังนั้นเครื่องมือตัวแปลภาษารายงานเป็นข้อผิดพลาด ดังรูปที่ 5.11

รูปที่ 5.11 ผลการทดสอบเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่หก

5.1.7 กรณีทดสอบที่เจ็ด

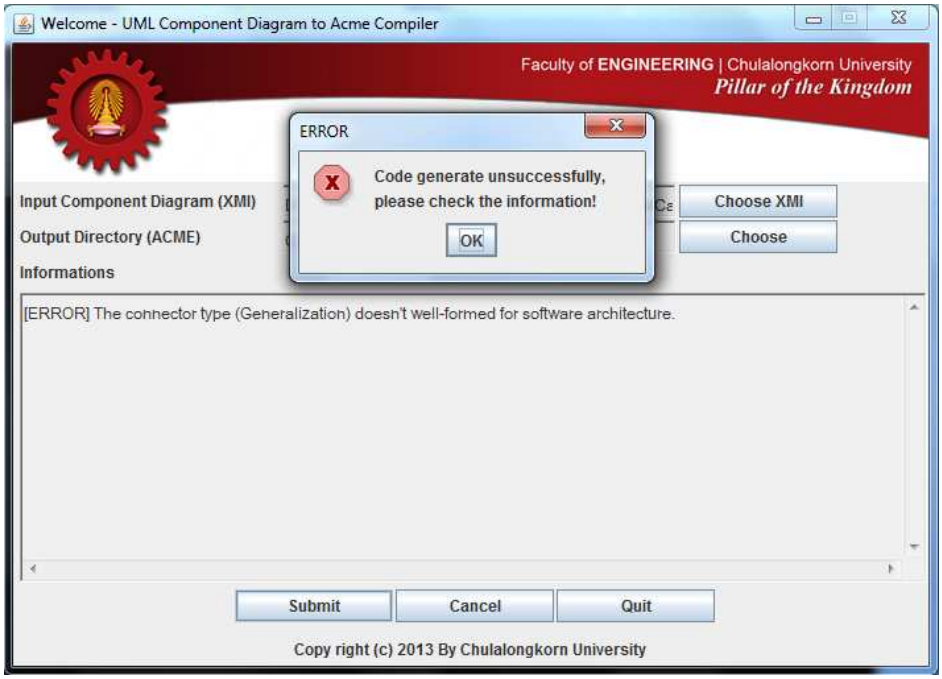
กรณีทดสอบที่เจ็ด เป็นการทดสอบการวิเคราะห์และตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลส่วนแผนภาพคอมโพเนนท์จำนวนหนึ่งแผนภาพ ซึ่งประกอบด้วยคอมโพเนนท์สามคอมโพเนนท์และระบุการเชื่อมต่อแบบแอสแซมบลีกับเจนเนอรัลไลเซชัน โดยในกรณีทดสอบเป็นแผนภาพคอมโพเนนท์ ที่ซึ่งตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ตามกฎตั้งต้น แสดงกรณีทดสอบในรูปแบบแผนภาพคอมโพเนนท์ ดังรูปที่ 5.12



รูปที่ 5.12 กรณีทดสอบที่เจ็ดสำหรับแผนภาพคอมโพเนนท์ที่ประกอบด้วยคอมโพเนนท์สามคอมโพเนนท์และระบุการเชื่อมต่อแบบแอสแซมบลีกับเจนเนอรัลไลเซชัน

จากรูปที่ 5.12 กรณีทดสอบที่เจ็ดดำเนินการทดสอบเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี เพื่อตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ซึ่งเมื่อประมวลผลด้วยเครื่องมือตัวแปลภาษาแล้วตรวจสอบพบว่าแผนภาพ

คอมโพเนนท์ดังกล่าวประกอบด้วยคอมโพเนนท์ตั้งแต่สองคอมโพเนนท์ขึ้นไปจริง แต่พบว่าได้ระบุการเชื่อมต่อแบบเจนเนอรัลไลเซชัน ที่ซึ่งไม่สอดคล้องกับแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์ ดังนั้นเครื่องมือตัวแปลภาษารายงานเป็นข้อผิดพลาด ดังรูปที่ 5.13



รูปที่ 5.13 ผลการทดสอบเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สำหรับกรณีทดสอบที่เจ็ด

บทที่ 6
สรุปผลการดำเนินงานวิจัย

จากการดำเนินการพัฒนาและวิจัยเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สามารถสรุปผลการดำเนินงานวิจัย ประโยชน์ที่ได้รับจากงานวิจัยและข้อจำกัดของเครื่องมือที่พัฒนา แนวทางในการพัฒนางานวิจัยต่อและสรุปและอภิปรายงานวิจัยเพิ่มเติม ดังนี้

6.1 สรุปผลการวิจัย

จากงานวิจัยได้นำเสนอเครื่องมือตัวแปลภาษาสำหรับแผนภาพคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี สามารถใช้เป็นทางเลือกสำหรับการแปลงสถาปัตยกรรมซอฟต์แวร์ โดยใช้รูปแบบไวยากรณ์ไม่พึ่งบริบทในขั้นตอนการตรวจสอบและวิเคราะห์แผนภาพคอมโพเนนท์ยูเอ็มแอล เพื่อให้มีความยืดหยุ่นและใช้งานง่ายในการแปลงไวยากรณ์ภาษาที่ไม่ซับซ้อนมากนัก โดยออกแบบกฎในรูปแบบเรกูลาร์เอกซ์เพรสชันสำหรับการตรวจสอบคำศัพท์ในเอกสารเอกซ์เอ็มไอ และออกแบบกฎในรูปแบบเมทาดาต้าวากยสัมพันธ์อีบีเอ็นเอฟ สำหรับการวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอลเพื่อแปลงเป็นภาษาแอคมี ซึ่งสามารถตรวจสอบแผนภาพคอมโพเนนท์ แผนภาพคอมโพเนนท์ย่อย คอมโพเนนท์ การเชื่อมต่อ พอร์ตและอินเตอร์เฟส โดยเฉพาะการจำลองส่วนการเชื่อมต่อโพรไวด์และรีไควร์ที่เรียกใช้โดยผ่านอินเตอร์เฟสในยูเอ็มแอลนั้น เมื่อแปลงไปยังภาษาแล้วจะระบุเป็นอินเตอร์เฟสเป็นส่วนการเชื่อมต่อ ที่ซึ่งประกอบด้วยการเชื่อมต่อโพรไวด์และรีไควร์ ซึ่งระบุเป็นบทบาทสำหรับการเชื่อมต่อในภาษาแอคมี

ทั้งนี้ผู้วิจัยได้ดำเนินการออกแบบและทดลองโดยนำเครื่องมือเฟล็กซ์และแย็กค์ ใช้ในเครื่องมือตัวแปลภาษาสำหรับขั้นตอนการตรวจสอบคำศัพท์และการวิเคราะห์ไวยากรณ์แผนภาพคอมโพเนนท์ยูเอ็มแอล และในรุ่นปัจจุบันเครื่องมือตัวแปลภาษาสามารถทำงานร่วมกับสัญลักษณ์ยูเอ็มแอล รุ่น 2.0 เอกซ์เอ็มไอ รุ่น 2.1 และภาษาแอคมี รุ่น 2.1 ซึ่งเครื่องมือดังกล่าวจะสามารถใช้เป็นทางเลือกในทางปฏิบัติ สำหรับการเชื่อมโยงการสร้างแบบจำลองระหว่างภาษายูเอ็มแอลและภาษาเอดีแอล

6.2 ประโยชน์ที่ได้รับจากงานวิจัย

1) เครื่องมือที่นำเสนอสามารถสร้างผลลัพธ์ จากแผนภาพคอมโพเนนท์ที่จัดเก็บในรูปแบบภาษาเอกซ์เอ็มไอไปเป็นภาษาแอคมีได้ถูกต้อง ตามไวยากรณ์ภาษายูเอ็มแอลกับภาษาเอดีแอล

2) เครื่องมือที่นำเสนอสามารถสร้างแบบจำลองด้านสถาปัตยกรรม และแลกเปลี่ยนข้อมูลระหว่างเครื่องมือการสร้างแบบจำลองต่างประเภท เช่น เครื่องมือ Visual Paradigm ซึ่งเป็นเครื่องมือการสร้างแบบจำลองภาษายูเอ็มแอลกับเครื่องมือแอคมีสตูดิโอ ซึ่งเป็นเครื่องมือการสร้างแบบจำลองทางด้านสถาปัตยกรรม เป็นต้น

3) เครื่องมือที่นำเสนอสามารถใช้เป็นทางเลือก สำหรับนักออกแบบกับนักพัฒนาซอฟต์แวร์ โดยแปลงแผนภาพยูเอ็มแอลไปเป็นแผนภาพทางด้านสถาปัตยกรรมในรูปแบบภาษาแอคมี เพื่อใช้ในการศึกษาและวิจัย

6.3 ข้อจำกัดของเครื่องมือ

สำหรับเครื่องมือที่พัฒนามีข้อจำกัดการใช้งาน ซึ่งจำกัดด้วยเครื่องมือที่ส่งออกเอกสารเอกซ์เอ็มไอ ที่ซึ่งเป็นข้อมูลนำเข้าและรูปแบบของแผนภาพคอมโพเนนท์ยูเอ็มแอล รวมทั้งผลลัพธ์ภาษาแอคมี โดยมีข้อจำกัดดังนี้

1) เครื่องมือที่ใช้สำหรับนำออกแผนภาพคอมโพเนนท์ยูเอ็มแอลในรูปแบบเอกสารเอกซ์เอ็มไอนั้น ต้องนำออกด้วยเครื่องมือ Visual Paradigm รุ่น 10.0

2) ข้อมูลภาษาเอกซ์เอ็มไอ ต้องเป็นเอกสารเอกซ์เอ็มไอที่ถูกต้องและดีแล้วเท่านั้น

3) ข้อมูลผลลัพธ์ภาษาแอคมี เมื่อนำไปแสดงผลบนเครื่องแอคมีสตูดิโอแล้วต้องจัดเรียงสัญลักษณ์ใหม่ เพื่อความสมบูรณ์ในการแสดงผล

6.4 แนวทางในการพัฒนางานวิจัยต่อ

สำหรับแนวทางในการพัฒนางานวิจัยต่อนั้น ผู้วิจัยมีข้อแนะนำสำหรับการนำเครื่องมือตัวแปลภาษาไปต่อยอด เพื่อการศึกษาและวิจัยดังนี้

1) พัฒนาขั้นตอนการวิเคราะห์และตรวจสอบส่วนคอมโพเนนท์หลักกับคอมโพเนนท์ย่อยที่ซึ่งส่งผ่านการเชื่อมต่อแบบดีลีเกชัน

2) ข้อมูลผลลัพธ์ควรวิเคราะห์และตรวจสอบในส่วนคุณสมบัติเพิ่มเติม และพัฒนาต่อยอดเครื่องมือตัวแปลภาษาเป็นปลั๊กอิน (Plug-in) สำหรับเครื่องมือแอคมีสตูดิโอ ซึ่งเป็นการเพิ่มความสะดวกและง่ายต่อการใช้งานยิ่งขึ้น

3) ส่วนรายงานผลการแปลงแผนภาพคอมโพเนนท์ อาจจะพัฒนาต่อยอดเป็นอีกหนึ่งขั้นตอนเพื่อนำออกเป็นข้อมูลการแปลงแผนภาพคอมโพเนนท์สำหรับผู้ใช้งาน

6.5 สรุปและอภิปรายงานวิจัยเพิ่มเติม

1) จากงานวิจัย Bridging the gap between Acme and UML 2.0 for CBD [20] ผู้วิจัยได้นำเสนอการจับคู่ภาษาแอคมีไปยังภาษายูเอ็มแอล โดยนำรหัสโอซีแอล (OCL Code) ใช้สำหรับการอธิบายภาษาธรรมชาติของทั้งโอซีแอลฟังก์ชันและกฎการตรวจสอบความสอดคล้องกันสำหรับการจับคู่ทางภาษาดังกล่าว ซึ่งเริ่มด้วยการนิยามฟังก์ชัน IsAcmeComponent(), IsAcmeConnector(), IsAcmePort(), IsAcmeRole(), IsAcmeProperty(), และ IsAcmeSystem() ซึ่งในฟังก์ชันดังกล่าวเป็นฟังก์ชันที่นิยามไว้ตรวจสอบองค์ประกอบคอมโพเนนท์ การเชื่อมต่อ พอร์ต บทบาท คุณลักษณะและระบบตามลำดับ ซึ่งผู้วิจัยได้มองว่าทุกองค์ประกอบในภาษาแอคมีนั้นอธิบายได้ซับซ้อนกว่าภาษายูเอ็มแอล เช่น การเชื่อมต่อในภาษาแอคมีแทนปฏิสัมพันธ์ระหว่างคอมโพเนนท์ เนื่องจากการเชื่อมต่อถือได้ว่าเป็นการเตรียมส่วนที่ใช้ต่อประสานสำหรับการออกแบบสถาปัตยกรรม

ซึ่งมีหลายทางเลือกสำหรับการพิจารณาเพื่อแทนยูเอ็มแอล โดยทางเลือกแรกสามารถแทนการเชื่อมต่อดังกล่าวด้วยการเชื่อมต่อแบบแอสแซมบลี การเชื่อมต่อในแอคมีอาจจะมีความซับซ้อนกว่าการเป็นอินเตอร์เฟสแบบธรรมดา แต่สามารถเตรียมการเชื่อมต่อเพื่อเป็นตัวเชื่อมระหว่างคอมโพเนนท์และเพื่อความชัดเจนในการออกแบบ จึงออกแบบให้ใช้คอมโพเนนท์ที่มีสเทอริโอไทป์เป็น <<AcmeConnector>> เป็นต้น โดยทุกองค์ประกอบในภาษาแอคมีนั้นจะแทนด้วยคอมโพเนนท์ ที่ซึ่งระบุด้วยสเทอริโอไทป์ที่ต่างกัน ดังนี้ <<AcmeComponent>>, <<AcmePort>>, <<AcmeConnector>>, <<AcmeRole>>, <<AcmeProperty>> และ <<AcmeSystem>>

โดยในแต่ละคอมโพเนนท์จะอธิบายด้วยคุณลักษณะเป็นการแทนข้อมูลทางความหมายเกี่ยวกับระบบและองค์ประกอบในสถาปัตยกรรม ซึ่งในยูเอ็มแอล 2.0 คอมโพเนนท์สามารถได้ทั้งมุมมองภายนอกและมุมมองภายใน โดยในงานวิจัยกล่าวว่าสามารถสร้างคุณลักษณะดังกล่าวไว้ภายนอกคอมโพเนนท์ได้แต่ต้องอยู่ในขอบเขตของคอมโพเนนท์

2) จากงานวิจัย Checking component assembly in Acme: An approach applied on UML 2.0 Components Model [21] ผู้วิจัยได้นำเสนอแนวทางเพื่อตรวจสอบความสอดคล้องกันระหว่างยูเอ็มแอล 2.0 กับภาษาแอคมี ซึ่งอธิบายด้วยเมต้าโมเดล เรียกว่า ระดับเอ็ม2 โดยใช้รูปแบบสถาปัตยกรรมของภาษาแอคมี ส่วนคอมโพเนนท์แอสแซมบลีในยูเอ็มแอล 2.0 จะอธิบายแทนด้วยแนวคิดในระบบแอคมีในระดับเอ็ม1 ซึ่งกล่าวได้ว่า ระดับเอ็ม1 ต้องมีความสอดคล้องกับระดับเอ็ม2 ในงานวิจัยมองว่าภาษาแอคมีมีลักษณะทางความหมายที่ละเอียดกว่ายูเอ็มแอล 2.0/โอซีแอล 2.0 โดยเมต้าโมเดลของคอมโพเนนท์ยูเอ็มแอล 2.0 ได้จำลองมาจากสไตล์ในภาษาแอคมีโดยใช้แนวคิดแฟมิลี่ (Family) ซึ่งมีความเกี่ยวข้องกับแบบจำลองคอมโพเนนท์ยูเอ็มแอล 2.0 สำหรับสัญลักษณ์เฉาพะการดำเนินงาน, อินเตอร์เฟสโพรไวด์, อินเตอร์เฟสรีไควร์, คอมโพเนนท์และการเชื่อมต่อแบบแอสแซมบลี ในงานวิจัยได้ใช้ประโยชน์จากประเภทที่นำเสนอด้วยภาษาแอคมี ดังนี้ สัญลักษณ์เฉพาะทางคุณสมบัติ, พอร์ต, คอมโพเนนท์, การเชื่อมต่อ, บทบาท, เซต (Set), ระเบียน (Record) และ ลำดับ (Sequence)

โดยกฎที่สร้างขึ้นอนุญาตให้ใช้เป็นแบบแผนสำหรับการสร้างยูเอ็มแอล 2.0 ในรูปแบบภาษาแอคมี อธิบายรายละเอียดกฎ ดังนี้

- R1: บริการที่ประกาศไว้ภายในอินเตอร์เฟส กำหนดแบบแผนในภาษาแอคมีนั้นจะกำหนดด้วยกลุ่มของสัญลักษณ์เฉพาะ
- R2: อินเตอร์เฟสโพรไวด์และอินเตอร์เฟสรีไควร์ กำหนดแบบแผนในภาษาแอคมีนั้นจะกำหนดด้วยประเภทพอร์ตที่ระบุ *ProvidedInterface* และ *RequiredInterface* ตามลำดับ ซึ่งการระบุอินเตอร์เฟสจะระบุเป็นการเชื่อมต่อประเภท R4 ภายใต้สองบทบาท คือ เซิร์ฟเวอร์และไคลเอนต์

- R3: คอมโพเนนท์ในยูเอ็มแอล 2.0 กำหนดแบบแผนในภาษาแอคมีที่เป็นประเภทคอมโพเนนท์ เรียกว่า *ComponentUML* ซึ่งดำเนินการตรวจสอบตามข้อจำกัดใน Armani และแสดงบทบาทที่พบในคอมโพเนนท์ยูเอ็มแอล 2.0
- R4: การเชื่อมต่อแบบแอสแซมบลีในยูเอ็มแอล 2.0 กำหนดแบบแผนในภาษาแอคมีที่เป็นกระเภทการเชื่อมต่อ เรียกว่า *AssemblyUML* ซึ่งประกอบด้วยสองบทบาทและดำเนินการตรวจสอบตามข้อจำกัดในอาร์มานิ (Armani)

ในงานวิจัยได้มุ่งเน้นการนิยามส่วนการเชื่อมต่อประเภทแอสแซมบลี ที่ซึ่งแทนด้วยคอมโพเนนท์ยูแอสแซมบลีสำหรับเอ็มแอล 2.0 ในภาษาแอคมีเท่านั้น

ดังนั้น จากการศึกษาและค้นคว้างานวิจัยที่เกี่ยวข้องดังกล่าวในวิทยานิพนธ์ฉบับนี้มีส่วนที่คล้ายกันในส่วนของการกำหนดองค์ประกอบการแปลงจากภาษายูเอ็มแอลไปยังภาษาแอคมี ที่ประกอบด้วยระบบ คอมโพเนนท์ พอร์ต การเชื่อมต่อ บทบาทและเรพพรีเซนเทชัน ซึ่งอธิบายการแทนข้อมูลทางความหมายเกี่ยวกับระบบและองค์ประกอบในสถาปัตยกรรมด้วยคุณลักษณะที่อยู่ภายในคอมโพเนนท์ ส่วนที่นำเสนอเพิ่มเติมในวิทยานิพนธ์ฉบับนี้ คือ การนิยามกฎเพื่อตรวจสอบแผนภาพคอมโพเนนท์ยูเอ็มแอลสำหรับสถาปัตยกรรมซอฟต์แวร์และองค์ประกอบส่วนแอชแทชเมนท์เพื่อช่วยในการระบุปฏิสัมพันธ์ระหว่างคอมโพเนนท์และการเชื่อมต่อ โดยผ่านทางพอร์ตและบทบาทตามลำดับ รวมทั้งการนำเสนอกลไกตัวแปลภาษาสำหรับการแปลงคอมโพเนนท์ยูเอ็มแอลไปยังภาษาแอคมี

รายการอ้างอิง

[1] Object Management Group (OMG) – UML (Online), Available from: http://www.uml.org/ [June 27, 2012]

[2] CollabNet, ArgoUML - Open Source Software Engineering Tools (Online), Available from: http://argouml.tigris.org/ [2009]

[3] IBM - Rational Rose Enterprise – Software (Online), Available from: http://www-01.ibm.com/software/awdtools/developer/rose/enterprise/index.html

[4] Visual Paradigm for UML (Online), Available from: http://www.visual-paradigm.com/

[5] Object Management Group (OMG) - XML Metadata Interchange (XMI) Mapping Specification version 2.1 (Online), Available from: http://www.omg.org/spec/XMI/2.1/ [September 2005]

[6] David Galan, Robert Monroe and David Wile, Acme: An Architecture Description Interchange Language. CASCON First Decade High Impact Papers (CASCON'10), November 2010.

[7] Prabhat Mishra and Nikil Dutt, Architecture Description Languages, Customizable and Configurable Embedded Processors, 2006.

[8] W3C Recommendation, Extensible Markup Language 1.0 (Fifth Edition) (Online), Available from: http://www.w3.org/TR/xml/ [November 26, 2008]

[9] Shihong Huang, Vashali Gohel and Sam Hsu, Towards Interoperability of UML Tools for Exchanging High-Fidelity Diagrams, Proceedings of the 25th annual ACM international conference on Design of communication (SIGDOC '07), New York, 2007, Pages 134-141.

[10] Andrey Naumenko and Alain Wegmann, A Metamodel for the Unified Modeling Language, UML 2002, LNCS 2460, 2002, Pages 2-17.

[11] G.Caoncas, M. Marchesi, A. Cau, S. Pinna, K. Mannaro and N. Serra, XMI for XP Project Data Interchange, Proceedings of the 2004 workshop on Quantitative techniques for software agile process (QUTE-SWAP'04), New York, November 5, 2004, Pages 53 – 58.

[12] International Organization for Standardization (ISO), ISO/IEC 14977:1996(E)-Information technology-Syntactic metalanguage-Extended BNF(Online), Available from: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf

[13] Anthony A. Aaby, Compiler Construction using Flex and Bison, Walla Walla College, February 25, 2004.

[14] John R. Levine, flex & bison, O'Reilly Media Publications, 1005 Gravenstein Highway North, Sebastopol, CA 95472, August 2009.

[15] John R. Levine, Tony Mason and Doug Brown, lex & yacc, O'Reilly & Associates, Inc. 103 Morris Street, Suite A Sebastopol, CA 95472, Second Edition, October 1992.

[16] Aaron Myles Landwehr, Compiler Design (Flex/Bison Tutorial), the Computer Architecture and Parallel Systems Laboratory (CAPSL), 2012.

[17] JFlex-The Fast Scanner Generator for Java (Online), Available from: http://jflex.de/ [2009]

[18] CUP Parser Generator for Java (Online), Available from: http://www2.cs.tum.edu/projects/cup/

[19] Richard E. Pattis, EBNF: A Notation to Describe Syntax (Online), Available from: http://www.cs.cmu.edu/~pattis/misc/ebnf.pdf [October 15, 2005]

[20] Miguel Goulao and Fernando Brito e Abreu, Bridging the gap between Acme and UML 2.0 for CBD, Specification and Verification of Component-Based Systems (SAVCBS'03), 2003.

[21] Mourad Kmimech, Mohamed Tahar Bhiri and Philipe Aniorte, Checking component assembly in Acme: An approach applied on UML 2.0 Components Model, The Fourth International Conference on Software Engineering Advances (ICSEA'09), Portugal, 2009, Pages 494-499.

[22] Jayeeta Chanda, Ananya Kanjinal and Sabnam Sengupta, UML-Compiler: A Framework for Syntactic and Semantic Verification of UML Diagrams, The 6th International Conference on Distributed Computing and Internet Technologies (ICDCIT'10), Berlin, 2010, Pages 194-205.

[23] Russ Miles and Kim Hamilton, Learning UML 2.0, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, First Edition, April 2006.

[24] Nenad Medvidovic, David S. Rosenblum, David F. Redmiles and Jason E. Robbins, Modeling Software Architectures in the Unified Modeling Language, ACM Transactions on Software Engineering and Methodology, Vol. 11, No. 1, January 2002, Pages 2–57.

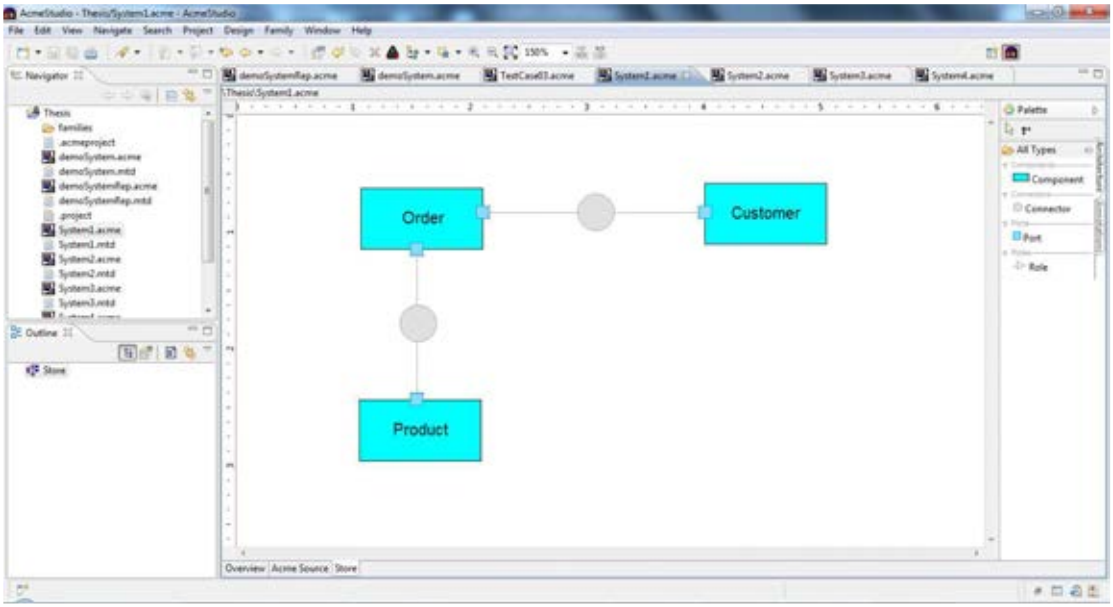
[25] Acme, BNF for armaniparser.jj (Online), Available from: http://www.cs.cmu.edu/~acme/html/ArmaniParser.html [January 4, 2011

[26] OMG Unified Modeling LanguageTM (OMG UML), Superstructure Version 2.3 (Online), Available from: http://www.omg.org/spec/UML/2.3/Superstructure/PDF/

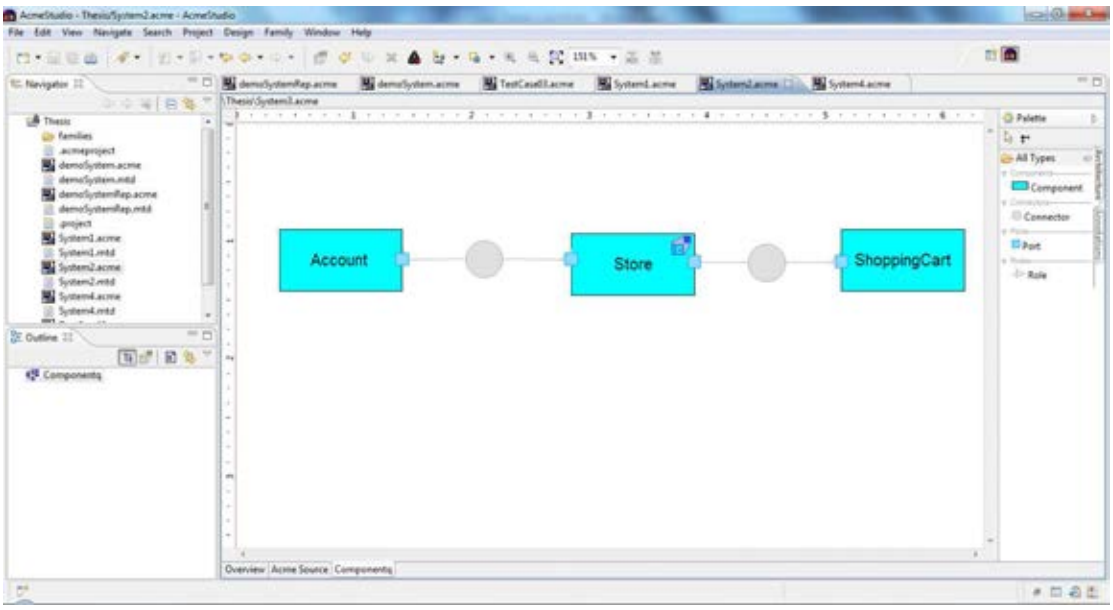
ภาคผนวก

ภาคผนวก ก
แสดงผลลัพธ์จากกรณีทดสอบด้วยเครื่องมือแอคมีสตูดิโอ

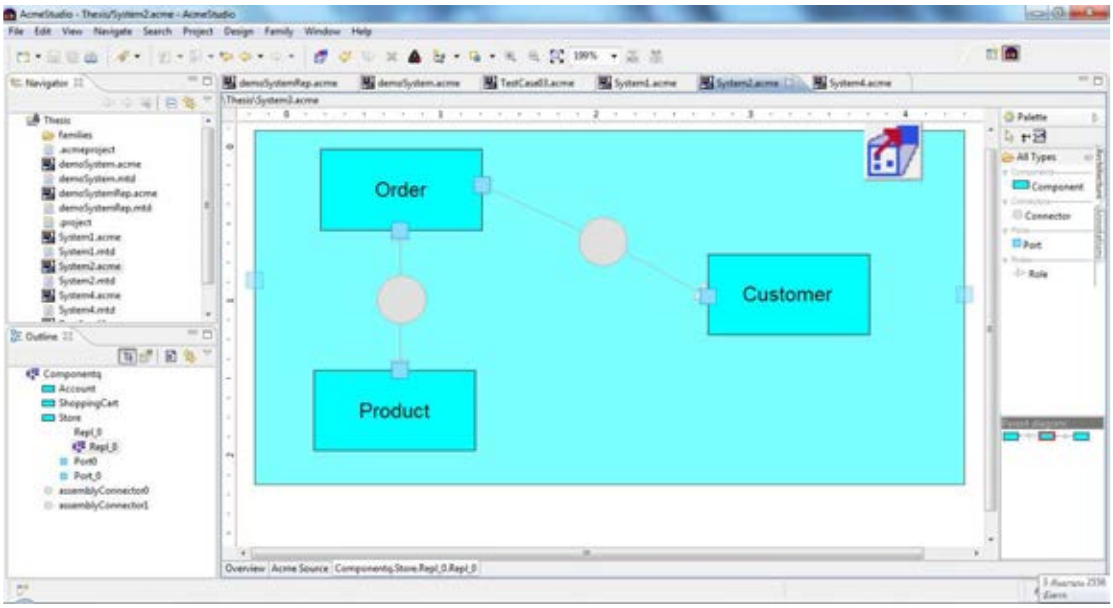
สำหรับการแสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ ซึ่งเป็นแผนภาพที่แทนด้วยแบบจำลองสถาปัตยกรรมซอฟต์แวร์ จากกรณีทดสอบที่หนึ่ง สอง สามและห้าตามลำดับ (แสดงในบทที่ 5) สามารถแสดงผลดังรูปที่ ก-1 ก-2 ก-3 และ ก-4



รูปที่ ก-1 แสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ สำหรับกรณีทดสอบที่หนึ่ง

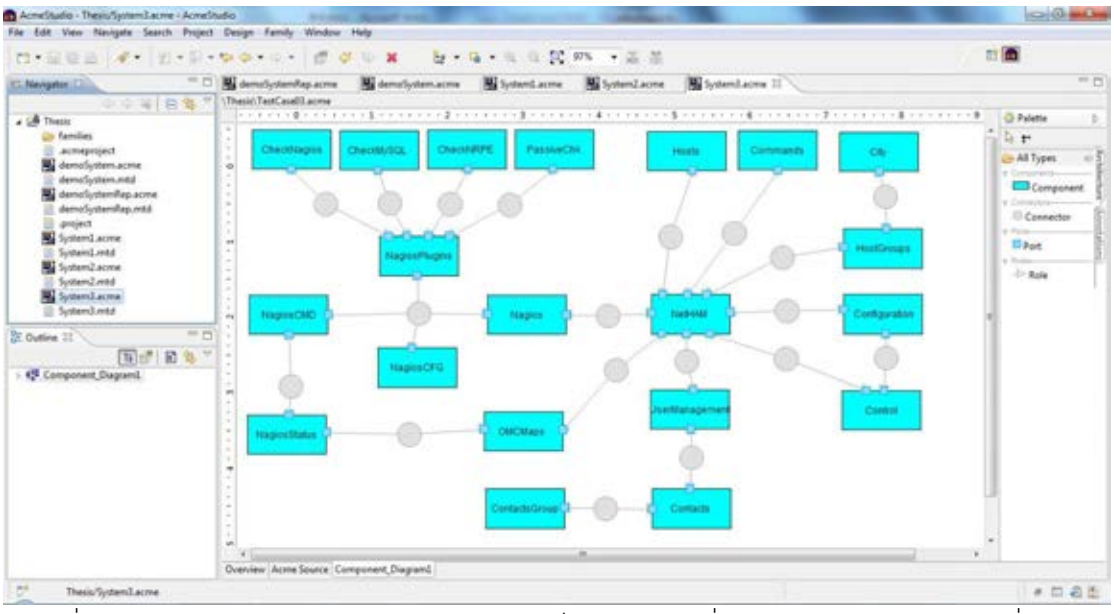


ก) แสดงระบบหลัก

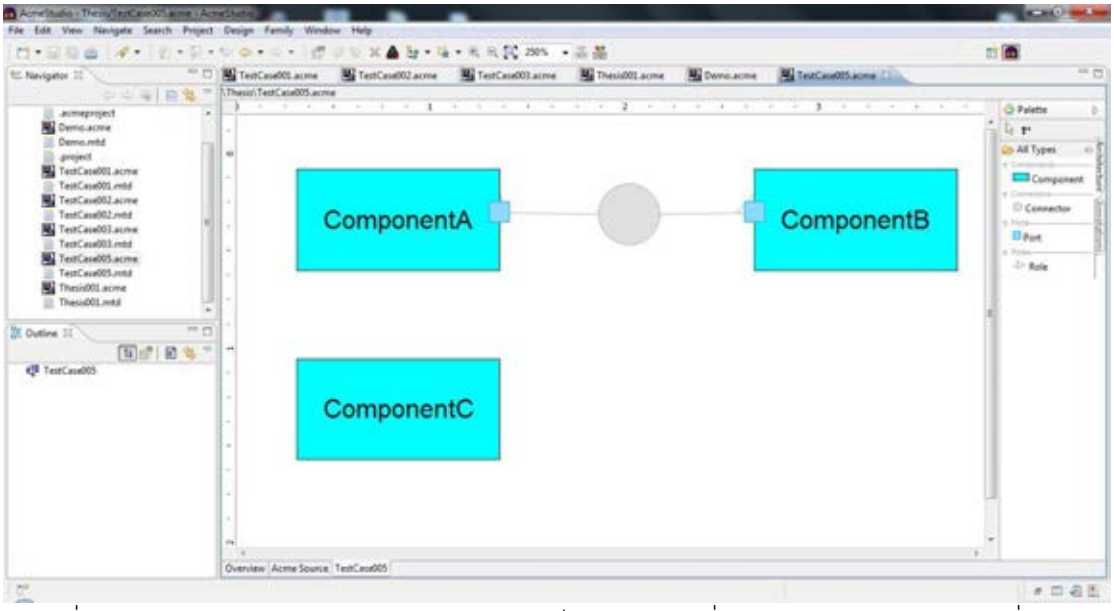


ข) แสดงระบบย่อย

รูปที่ ก-2 แสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ สำหรับกรณีทดสอบที่สอง



รูปที่ ก-3 แสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ สำหรับกรณีทดสอบที่สาม



รูปที่ ก-4 แสดงผลลัพธ์แผนภาพคอมโพเนนท์ยูเอ็มแอลจากเครื่องมือตัวแปลภาษา ด้วยเครื่องมือแอคมีสตูดิโอ สำหรับกรณีทดสอบที่ห้า

ประวัติผู้เขียนวิทยานิพนธ์

นายชุมพล โมฆรัตน์ เกิดเมื่อวัน อังคาร ที่ 31 เดือน สิงหาคม พุทธศักราช 2525 ที่จังหวัดกาฬสินธุ์ สำเร็จการศึกษาระดับปริญญาบัณฑิต ในหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ จากคณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา วิทยาเขตบางแสน จังหวัดชลบุรี เมื่อปีการศึกษา 2548 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2556