การแปลงโมเดิร์นฟอร์แทรนสำหรับยูเอ็มแอลซีเควนซ์ไดอะแกรม

Modern Fortran Transformation for UML Sequence Diagrams

อนวัช เล่ห์ทองคำ

Anawat Leatongkam

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree the Master of Science in Information Technology
Prince of Songkla University
2561

การแปลงโมเดิร์นฟอร์แทรนสำหรับยูเอ็มแอลซีเควนซ์ไดอะแกรม

Modern Fortran Transformation for UML Sequence Diagrams

อนวัช เล่ห์ทองคำ

Anawat Leatongkam

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree the Master of Science in Information Technology
Prince of Songkla University
2561

ชื่อวิทยานิพนธ์ การแปลงโมเดิร์นฟอร์แทรนสำหรับยูเอ็มแอลซีเควนซ์ไดอะแกรม
ผู้เขียน นายอนวัช เล่ห์ทองคำ
สาขาวิชา เทคโนโลยีสารสนเทศ

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....
(ผู้ช่วยศาสตราจารย์ ดร.อซีส นันทอมรพงศ์)

คณะกรรมการสอบ

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.รัตนา เวทย์ประสิทธิ์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อซีส นันทอมรพงศ์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ทรงศรี ตั้งศรีไพโรจน์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

.....
(ศาสตราจารย์ ดร.ดำรงศักดิ์ ฟ้ารุ่งสาง)
คณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ.....
(ผู้ช่วยศาสตราจารย์ ดร.อชีส นันทอมรพงศ์)
อาจารย์ที่ปรึกษาวิทยานิพนธ์

ลงชื่อ.....
(นายอนวัช เล่ห์ทองคำ)
นักศึกษา

ข้าพเจ้าขอรับรองว่า ผลงานวิจัยนี้ไม่เคยเป็นส่วนหนึ่งในการอนุมัติปริญญาในระดับใดมาก่อน และไม่ได้ถูกใช้ในการยื่นขออนุมัติปริญญาในขณะนี้

ลงชื่อ.....

(นายอนวัช เล่ห์ทองคำ)

นักศึกษา

ชื่อวิทยานิพนธ์ การแปลงโมเดิร์นฟอร์แทรนสำหรับยูเอ็มแอลซีเควนซ์ไดอะแกรม
ผู้เขียน นายอนวัช เล่ห์ทองคำ
สาขาวิชา เทคโนโลยีสารสนเทศ
ปีการศึกษา 2560

บทคัดย่อ

ปัจจุบันวิศวกรรมย้อนกลับได้เป็นที่รู้จักกันอย่างแพร่หลาย โดยเป็นกระบวนการสกัดแนวคิดของระบบ และข้อมูลการออกแบบออกจากระบบ งานวิจัยนี้ให้ความสำคัญกับเครื่องมือด้านวิศวกรรมย้อนกลับที่มีชื่อว่า ForUML ซึ่งสามารถสร้างยูเอ็มแอลไดอะแกรมขึ้นมาจากซอร์สโค้ดภาษาฟอร์แทรน โดยทั่วไปภาษาฟอร์แทรนเป็นภาษาที่นิยมใช้สำหรับพัฒนาซอฟต์แวร์ในด้านที่เกี่ยวข้องกับวิทยาศาสตร์ และวิศวกรรมศาสตร์ เช่น การพยากรณ์อากาศ ดาราศาสตร์ และการออกแบบทางด้านวิศวกรรม ในปัจจุบันยังขาดเครื่องมือที่ใช้ในการแสดงภาพรวมการทำงานของระบบที่พัฒนาด้วยภาษาฟอร์แทรน โดยยูเอ็มแอลไดอะแกรมที่ได้จากเครื่องมือ ForUML จะช่วยให้นักวิทยาศาสตร์และวิศวกรที่พัฒนาซอฟต์แวร์เข้าใจโครงสร้างและพฤติกรรมของซอฟต์แวร์ในระดับภาพรวม นอกจากนี้ยูเอ็มแอลไดอะแกรมยังช่วยให้สมาชิกในทีมและนักพัฒนาภายนอกที่สนใจ สามารถเข้าใจและสื่อสารกันได้ดี

ในเวอร์ชันแรกของ ForUML สามารถแสดงได้เฉพาะคลาสไดอะแกรมเท่านั้น คลาสไดอะแกรมนั้นจะแสดงให้เห็นถึงโครงสร้าง และความสัมพันธ์ระหว่างคลาสภายในระบบ อย่างไรก็ตามคลาสไดอะแกรมก็ยังไม่เพียงพอต่อการวิเคราะห์และทำความเข้าใจระบบ โดยเฉพาะพฤติกรรมและปฏิสัมพันธ์ระหว่างคลาสในระบบ ซึ่งคุณลักษณะเหล่านี้สามารถถูกแสดงได้โดยยูเอ็มแอลซีเควนซ์ไดอะแกรม

ดังนั้นผู้วิจัยจึงมีแนวคิดที่จะเพิ่มศักยภาพความสามารถของเครื่องมือ ForUML โดยเฉพาะอย่างยิ่งความสามารถในการสร้างยูเอ็มแอลซีเควนซ์ไดอะแกรม และทำการเผยแพร่ซอฟต์แวร์ในรูปแบบโอเพนซอร์ส ซึ่งจะช่วยให้นักพัฒนาสามารถทำความเข้าใจระบบที่พัฒนาด้วยฟอร์แทรนได้ดีขึ้น รวมถึงยังช่วยให้นักพัฒนามีการตัดสินใจที่ดีขึ้นในกระบวนการพัฒนาซอฟต์แวร์ และการบำรุงรักษา

ซอฟต์แวร์ นอกจากนี้ความสามารถที่เพิ่มขึ้นมาจะช่วยเอื้ออำนวยให้การสื่อสารในชุมชนนักพัฒนาซอฟต์แวร์ทางด้านวิทยาศาสตร์ และวิศวกรรมศาสตร์ดียิ่งขึ้น

คำสำคัญ: วิศวกรรมย้อนกลับ ฟอร์แทรน แผนภาพยูเอ็มแอลซีเควนซ์ วิศวกรรมซอฟต์แวร์

Thesis Title Modern Fortran Transformation for UML Sequence Diagrams

Author Anawat Leatongkam

Major Program Information Technology

Academic Year 2017

ABSTRACT

Recently, reverse engineering has become widely recognized as a valuable process for extracting system abstractions and design information from existing software. The proposed research will focus on ForUML, a reverse engineering tool developed to extract UML diagrams from modern, object-oriented Fortran programs. Generally, Fortran is used to implement scientific and engineering software in various domains, such as weather forecasting, astrophysics, and engineering design. Methods for visualizing the existing design of object-oriented Fortran software, however, are lacking. UML diagrams of the Fortran software would help scientists and engineers communicate about the structure and behavior of their programs at a higher level of abstraction than the source code itself. UML diagrams can enhance discussions within development teams and with the broader scientific community.

The first version of ForUML produces only UML class diagrams. Class diagrams provide a useful window into the static structure of a program, including the make-up of each class and the relationships between classes. However, class diagrams lack temporal needed to understand class behavior and interactions between classes. UML sequence diagrams provide such important algorithmic information.

Therefore, the researcher proposes to extend ForUML to extract UML sequence diagrams from Fortran code and to offer this capability via a widely used open-source platform**.** This research argues that the proposed capability will enable the

visualization of object-oriented Fortran software behavior and algorithmic structure and thereby enhance the development, maintenance practices, decision processes, and communications in the scientific and engineering software community worldwide.

Keywords: Reverse Engineering, Fortran, UML Sequence Diagram, Software Engineering

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความอนุเคราะห์และความกรุณาอย่างยิ่งจากผู้มีพระคุณหลายท่าน โดยเฉพาะอย่างยิ่งอาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.อชีส นันทอมรพงศ์ ที่กรุณาให้ความรู้ที่มีคุณค่า ให้คำปรึกษา ชี้แนะแนวทาง และให้กำลังใจ ซึ่งเป็นการสร้างพลังให้ผู้วิจัยมีความอดทนและพยายามทำการศึกษาอย่างเต็มความสามารถ ผู้วิจัยรู้สึกซาบซึ้งในความเมตตา กรุณา และเสียสละของอาจารย์ในการประสิทธิ์ประสาทวิชา จึงขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.รัตนา เวทย์ประสิทธิ์ ผู้เป็นประธานกรรมการสอบวิทยานิพนธ์ และ ผู้ช่วยศาสตราจารย์ ดร.ทรงศรี ตั้งศรีไพโรจน์ ผู้เป็นคณะกรรมการสอบวิทยานิพนธ์ รวมถึงคณาจารย์ทุกท่าน ที่สละเวลาเพื่อช่วยให้กำลังใจ ให้ความรู้ ให้คำแนะนำและแนวทางที่มีประโยชน์ในการปรับปรุงวิทยานิพนธ์

ขอขอบคุณวิทยาลัยการคอมพิวเตอร์ และบัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ ที่ได้มอบโอกาสในการศึกษาและทุนสนับสนุนการวิจัยในครั้งนี้ รวมถึงบุคลากรทุกท่านที่คอยให้การสนับสนุนและความช่วยเหลือด้วยดีตลอดมา นอกจากนี้ผู้วิจัยยังได้รับกำลังใจจากบิดา มารดา และครอบครัว ที่คอยให้การสนับสนุน ส่งเสริมและความห่วงใยตลอดมา ตลอดจนกัลยาณมิตรทุกท่านที่ให้ความช่วยเหลือเกื้อกูลในทุกด้าน ประคับประคอง รวมทั้งให้กำลังใจที่มีคุณค่าอย่างยิ่ง ผู้วิจัยรู้สึกซาบซึ้งในความกรุณาและความปรารถนาดีของทุกท่านเป็นอย่างยิ่ง ผู้วิจัยจึงขอขอบพระคุณมา ณ โอกาสนี้

อนวัช เล่ห์ทองคำ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

รายการตาราง

รายการภาพประกอบ

รายการภาพประกอบ (ต่อ)

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของการวิจัย

ในปัจจุบันกระบวนการวิศวกรรมย้อนกลับ (Reverse Engineering) เป็นที่รู้จักกันอย่างแพร่หลาย โดยเฉพาะอย่างยิ่งในกลุ่มของนักพัฒนาซอฟต์แวร์ วิศวกรรมย้อนกลับเป็นกระบวนการค้นหาโครงสร้างและฟังก์ชันการทำงานของระบบหรือโปรแกรมหนึ่ง ๆ โดยจะมีการค้นหาส่วนประกอบของระบบ เช่น การค้นหาแนวคิดของระบบ (System Abstractions) ซึ่งเป็นขั้นตอนในการวิเคราะห์ซอร์สโค้ดเพื่อทำการค้นหาโครงสร้างโดยรวมของระบบ และการกู้คืนเอกสารการออกแบบ (Design Information) หรือเอกสารแสดงรายละเอียดการทำงานของระบบ โดยเอกสารเหล่านี้สามารถนำมาใช้ในการวิเคราะห์โครงสร้างการทำงานและทำความเข้าใจการทำงานในแต่ละส่วนของระบบ จากนั้นจึงนำส่วนประกอบที่ได้มาสร้างระบบหรือโปรแกรมขึ้นมาใหม่ ซึ่งมีความคล้ายคลึงกัน แต่อาจจะไม่เหมือนกับโครงสร้างการทำงานจากระบบเดิม (Müller, et al., 1993)

การทำวิศวกรรมย้อนกลับในด้านวิศวกรรมซอฟต์แวร์นั้นจะเกี่ยวข้องกับการอ่านและทำความเข้าใจซอร์สโค้ด เพื่อนำมาใช้ในการทำความเข้าใจระบบซอฟต์แวร์ ซึ่งในกระบวนการพัฒนาซอฟต์แวร์ถ้าหากระบบซอฟต์แวร์มีขนาดใหญ่ หรือมีจำนวนบรรทัดของโค้ด (Line of Code) มาก อาจจะก่อให้เกิดปัญหาเรื่องความซับซ้อนของระบบ จึงทำให้ยากต่อการอ่านและทำความเข้าใจซอร์สโค้ด ดังนั้นกระบวนการวิศวกรรมย้อนกลับจึงได้มีส่วนช่วยให้นักพัฒนาสามารถมองเห็นถึงภาพรวมของระบบได้ง่ายเพื่อที่จะได้ทำการบำรุงรักษา และปรับปรุงซอฟต์แวร์ อย่างไรก็ตามกระบวนการวิศวกรรมย้อนกลับของระบบซอฟต์แวร์ที่มีขนาดใหญ่หรือซับซ้อนนั้นเป็นเรื่องที่ยากและมีความท้าทายเป็นอย่างยิ่ง (Lanza and Ducasse, 2003) หนึ่งในความท้าทายของกระบวนการวิศวกรรมย้อนกลับ คือ การสร้างมุมมองที่สื่อ

ความหมายของสิ่งที่เป็นนามธรรมหรือสิ่งที่ไม่มีรูปร่าง เช่น ซอร์สโค้ด เพื่อให้อยู่ในรูปแบบที่สามารถอ่านและทำความเข้าใจความซับซ้อนได้ (Systa, 1999) เช่น ยูเอ็มแอล (Unified Modeling Language: UML)

ปัญหาที่พบเจอโดยทั่วไปทางด้านของวิศกรรมซอฟต์แวร์จะเกี่ยวข้องกับโค้ดที่มีการพัฒนาสืบทอดต่อกันมา ซึ่งโค้ดเหล่านี้แต่เดิมจะเป็นโค้ดที่เคยถูกพัฒนาขึ้นมาจากเวอร์ชันเก่า และมีการพัฒนาต่อกันมาจนถึงเวอร์ชันปัจจุบัน (Feathers, 2004) ดังนั้นจึงเป็นเรื่องยากที่จะแก้ไขหรือปรับเปลี่ยนโค้ด หากนักพัฒนาไม่มีความรู้ความเข้าใจเกี่ยวกับระบบเดิม โดยทั่วไปในกระบวนการพัฒนาซอฟต์แวร์นั้นนักพัฒนาจะทำการพัฒนาระบบซอฟต์แวร์ตามเอกสารการออกแบบ เพื่อให้สามารถพัฒนาซอฟต์แวร์ได้ตรงตามเป้าหมายที่ได้ระบุไว้ เมื่อเวลาผ่านไปนักพัฒนามีการแก้ไขหรือมีการปรับเปลี่ยนซอร์สโค้ดอยู่ตลอดเวลาทำให้โครงสร้างหรือฟังก์ชันการทำงานเปลี่ยนไปจากขอบเขตเดิมที่ได้กำหนดไว้ แต่ในขณะที่เอกสารการออกแบบ ไม่ได้มีการแก้ไขหรือปรับเปลี่ยนตามไปด้วย จึงทำให้นักพัฒนาไม่สามารถนำเอกสารเหล่านั้นไปใช้งานจริงในการบำรุงรักษาซอฟต์แวร์ได้ รวมถึงในปัจจุบันซอร์สโค้ดมีความซับซ้อนมากขึ้นทำให้เป็นเรื่องยากที่จะทำความเข้าใจเกี่ยวกับระบบ และทำให้ยากต่อการที่จะนำซอฟต์แวร์กลับมาใช้ใหม่ (Ning, et al., 1994) อีกทั้งถ้าหากนักพัฒนาไม่สามารถทำความเข้าใจระบบได้ก็จะทำให้ไม่สามารถปรับปรุงโครงสร้างการทำงานหรือเพิ่มคุณสมบัติใหม่ ๆ ในระบบได้

ปัจจุบันเครื่องมือในด้านวิศวกรรมย้อนกลับนั้นสามารถสนับสนุนวิศวกรซอฟต์แวร์ ในกระบวนการวิเคราะห์ และทำความเข้าใจระบบซอฟต์แวร์ที่มีความซับซ้อน ซึ่งความสามารถในการทำงานของเครื่องมือจะแตกต่างกันไปตามแต่ละภาษาที่ใช้พัฒนาโปรแกรม รวมทั้งเครื่องมือเหล่านี้จะช่วยสนับสนุนในกระบวนการบำรุงรักษาซอฟต์แวร์ โดยจะลดเวลาในการวิเคราะห์ และทำความเข้าใจซอร์สโค้ด จากงานวิจัยก่อนหน้านี้ได้มีการพัฒนาเครื่องมือที่มีชื่อว่า ForUML (Nanthaamornphong, et al., 2015) โดยเครื่องมือนี้จะแปลงโค้ดภาษาฟอร์แทรน (Fortran) (Decyk, et al., 2007) ไปเป็นยูเอ็มแอล โดยยูเอ็มแอลจะเป็นภาษาที่ใช้สำหรับอธิบาย แสดงความหมายและความสัมพันธ์ของระบบหรือโปรแกรมโดยอาศัยยูเอ็มแอลไดอะแกรม ที่เป็นที่รู้จักและมีการใช้งานกันอย่างแพร่หลายในงานด้านวิศวกรรมซอฟต์แวร์ สำหรับภาษาฟอร์แทรนนั้นจะเป็นภาษาที่นิยมใช้สำหรับพัฒนาซอฟต์แวร์ในด้านที่เกี่ยวข้องกับวิทยาศาสตร์ และวิศวกรรมศาสตร์ (Scientific Software) เช่น การพยากรณ์อากาศ ดาราศาสตร์ และการแพทย์ ซึ่งการพัฒนาซอฟต์แวร์ทางด้านนี้ยังขาดเครื่องมือทางด้านวิศวกรรมซอฟต์แวร์ที่ดีในการพัฒนาซอฟต์แวร์ควบคู่กันไป (Carver, et al., 2007) อีกทั้งการพัฒนาซอฟต์แวร์ในด้านนี้ส่วนใหญ่

จะพัฒนาขึ้นด้วยการลองผิดลองถูก และการศึกษาค้นคว้าด้วยตัวเอง เนื่องจากนักพัฒนาซอฟต์แวร์ในด้านนี้เป็นนักวิทยาศาสตร์ และวิศวกร ที่มีความรู้เพียงแค่พื้นฐานทั่วไปเกี่ยวกับการเขียนโค้ด จึงทำให้เกิดข้อจำกัดในการเขียนซอร์สโค้ดในเชิงลึก (Carver, 2009) ด้วยเหตุนี้หากมีข้อบกพร่องหรือผิดพลาดบางอย่างเกิดขึ้นในกระบวนการพัฒนาซอฟต์แวร์ จะทำให้ส่งผลกระทบอย่างมากต่อผู้คนในสังคม (Carver, 2012) โดยทั่วไปการพัฒนาซอฟต์แวร์ทางด้านนี้จะมีความท้าทายต่อนักพัฒนาที่เป็นนักวิทยาศาสตร์และวิศวกร (Nanthaamornphong, et al., 2015) ดังนี้

(1) นักพัฒนาไม่ได้ผ่านการฝึกอบรมในด้านวิศวกรรมซอฟต์แวร์อย่างเพียงพอ

(2) เครื่องมือบางอย่างในด้านวิศวกรรมซอฟต์แวร์นั้นมีความยากต่อการใช้งาน เนื่องจากนักพัฒนาไม่มีความรู้ความเชี่ยวชาญ

(3) ซอฟต์แวร์ทางด้านวิทยาศาสตร์ และวิศวกรรมศาสตร์โดยส่วนใหญ่จะขาดเอกสารสำหรับนักพัฒนา เช่น เอกสารการออกแบบ

ในปัจจุบัน ForUML เป็นซอฟต์แวร์เดียวที่สามารถแปลงโค้ดภาษาฟอร์แทรนซึ่งปัจจุบันได้ถูกพัฒนาเป็นภาษาโปรแกรมเชิงวัตถุ (Object-Oriented) คล้ายกับภาษาจาวา (JAVA) หรือภาษาซีพลัสพลัส (C++) ให้เป็นยูเอ็มแอลคลาสไดอะแกรม (UML Class Diagram) โดยไดอะแกรมจะแสดงให้เห็นถึงโครงสร้างการทำงานของคลาส (Class) ซึ่งจะบอกว่าแต่ละคลาสมีหน้าที่การทำงานที่สัมพันธ์ (Relationship) กับคลาสอื่นอย่างไร เพื่อง่ายต่อการทำความเข้าใจระบบ นอกจากนี้เครื่องมือ ForUML ยังถูกนำไปใช้ในการเรียนการสอนเกี่ยวกับการออกแบบซอฟต์แวร์ที่พัฒนาด้วยภาษาฟอร์แทรน ซึ่งจะช่วยให้ผู้เรียนสามารถเข้าใจถึงโครงสร้างของระบบที่พัฒนาได้ดีกว่าการอ่านจากซอร์สโค้ดโดยตรง

อย่างไรก็ตามเครื่องมือ ForUML ที่ถูกพัฒนาขึ้นในเวอร์ชันแรกนั้นยังมีขีดความสามารถที่จำกัด และยังไม่ได้รับการประเมินอย่างเพียงพอ เนื่องจากเครื่องมือนี้สามารถแสดงได้เฉพาะคลาสไดอะแกรมเพียงเท่านั้น โดยแผนภาพนี้จะเป็นแบบจำลองในมุมมองเชิงโครงสร้าง (Structure View) ซึ่งจะแสดงโครงสร้างของระบบที่ไม่มีการเคลื่อนไหว และไม่มีการระบุขั้นตอนการดำเนินงาน หรือลำดับการทำงานก่อนหลัง ดังนั้นจึงไม่เพียงพอต่อการวิเคราะห์ และทำความเข้าใจระบบ นอกจากนี้ผู้ใช้งานเครื่องมือได้ให้ข้อเสนอแนะสำหรับการเพิ่มคุณสมบัติหรือความสามารถใหม่ ๆ เช่น ยูเอ็มแอลซีเควนซ์ไดอะแกรม (UML Sequence Diagram) ซึ่งเป็นแบบจำลองในมุมมองเชิงพฤติกรรม (Behavior View) ที่แสดงถึงลำดับการทำงานภายในระบบ เพื่อให้สามารถมองเห็นถึงภาพรวมของระบบที่พัฒนาขึ้นมาจาก

ภาษาฟอร์แทรน รวมถึงสามารถช่วยในการตัดสินใจเกี่ยวกับกระบวนการพัฒนาซอฟต์แวร์และการบำรุงรักษาซอฟต์แวร์ได้ดียิ่งขึ้น

ในงานวิจัยนี้ทางผู้วิจัยจึงมีแนวคิดที่จะพัฒนาเครื่องมือ โดยมีการเพิ่มขีดความสามารถของเครื่องมือ ForUML ให้ดียิ่งขึ้น ได้แก่ การสร้างซีเควนซ์ไดอะแกรม ซึ่งเป็นแผนภาพที่แสดงให้เห็นถึงการปฏิสัมพันธ์กันระหว่างอ็อบเจกต์ (Objects) ของคลาส โดยจะมีการส่งข้อมูลระหว่างอ็อบเจกต์ตามลำดับของเวลา ซึ่งงานวิจัยนี้จะเป็นประโยชน์ต่อนักพัฒนาซอฟต์แวร์ที่ใช้ภาษาฟอร์แทรน ในการพัฒนาผู้วิจัยเชื่อว่าการมีเอกสารการออกแบบในหลายมุมมอง โดยเฉพาะยูเอ็มแอลไดอะแกรม จะช่วยให้นักพัฒนาสามารถวิเคราะห์ และทำความเข้าใจในระบบซอฟต์แวร์ได้ดียิ่งขึ้น อีกทั้งยังมีส่วนช่วยในกระบวนการพัฒนาซอฟต์แวร์และการบำรุงรักษาซอฟต์แวร์ (Dobing and Parsons, 2006) เนื่องจากการวิเคราะห์และทำความเข้าใจในระบบซอฟต์แวร์ที่มีขนาดใหญ่นั้น เพียงแค่ยูเอ็มแอลคลาสไดอะแกรมอย่างเดียวอาจจะไม่เพียงพอ ดังนั้นทางผู้วิจัยจึงมีแนวคิดที่จะเพิ่มความสามารถดังที่กล่าวมาข้างต้นเพื่อให้นักพัฒนาซอฟต์แวร์มีการตัดสินใจที่ดีขึ้นในกระบวนการพัฒนาซอฟต์แวร์

1.2 วัตถุประสงค์

1.2.1 พัฒนาศักยภาพของซอฟต์แวร์ ForUML

1.2.2 เพื่อช่วยเหลือนักพัฒนาที่ใช้ภาษาฟอร์แทรนในการพัฒนาสามารถทำความเข้าใจระบบซอฟต์แวร์ที่พัฒนาขึ้นมาจากภาษาฟอร์แทรนได้

1.2.3 พัฒนาองค์ความรู้ทางด้านวิศวกรรมย้อนกลับ

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้ผู้วิจัยจะเพิ่มความสามารถให้กับเครื่องมือ ForUML โดยจะเพิ่มคุณสมบัติที่จะสามารถแปลงซอร์สโค้ดจากภาษาฟอร์แทรนไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยใช้เอกสารเอกซ์เอ็มไอตามข้อกำหนดของยูเอ็มแอลเวอร์ชัน 2.1 หลังจากนั้นจะมีการประเมินเครื่องมือเพื่อตรวจสอบความถูกต้องของการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งจะพิจารณาจากการนำซีเควนซ์ไดอะแกรมที่ได้จากเครื่องมือมาเปรียบเทียบกับซอร์สโค้ดของระบบแต่ละส่วนตามที่ผู้วิจัยได้มีการกำหนดไว้ จากนั้นจะให้ผู้เชี่ยวชาญในด้านของภาษาฟอร์แทรนทำการตรวจสอบความถูกต้องของข้อมูลอีกครั้งเพื่อยืนยันว่าข้อมูลนั้นมีความถูกต้องจริง รวมถึงมีการเผยแพร่ซอฟต์แวร์ในรูปแบบของซอฟต์แวร์โอเพนซอร์ส เพื่อให้นักพัฒนาที่สนใจพัฒนาซอฟต์แวร์ด้วยภาษาฟอร์แทรนสามารถนำไปใช้ประโยชน์ แก้ไขปรับเปลี่ยน หรือพัฒนาเพิ่มเติมจากเดิมได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 เพื่อให้นักพัฒนาซอฟต์แวร์ และนักออกแบบซอฟต์แวร์ที่ใช้ภาษาฟอร์แทรน มีเครื่องมือที่ช่วยให้ทำความเข้าใจในการออกแบบ และการบำรุงรักษาซอฟต์แวร์ได้ดีขึ้น

1.4.2 เพื่อให้นักวิทยาศาสตร์ และวิศวกรที่พัฒนาซอฟต์แวร์ด้วยภาษาฟอร์แทรนสามารถประยุกต์ใช้ความรู้ทางด้านวิศวกรรมซอฟต์แวร์เพื่อพัฒนา และปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ได้ดียิ่งขึ้น

1.4.3 เพื่อเพิ่มความรู้ทางด้านวิศวกรรมย้อนกลับ และการทำความเข้าใจโปรแกรม (Program Comprehension) ให้แก่นักวิจัยและนักพัฒนา รวมถึงสามารถนำความรู้ไปใช้ในการเรียนการสอนเกี่ยวกับการพัฒนาซอฟต์แวร์

บทที่ 2

เทคโนโลยีและวรรณกรรมที่เกี่ยวข้อง

ในบทนี้จะเป็นการอธิบายเทคโนโลยีและวรรณกรรมที่เกี่ยวข้อง โดยจะเริ่มจากเทคโนโลยีที่เกี่ยวข้อง และหลังจากนั้นจะเป็นการอธิบายวรรณกรรมที่เกี่ยวข้องกับการดำเนินงานวิจัยนี้

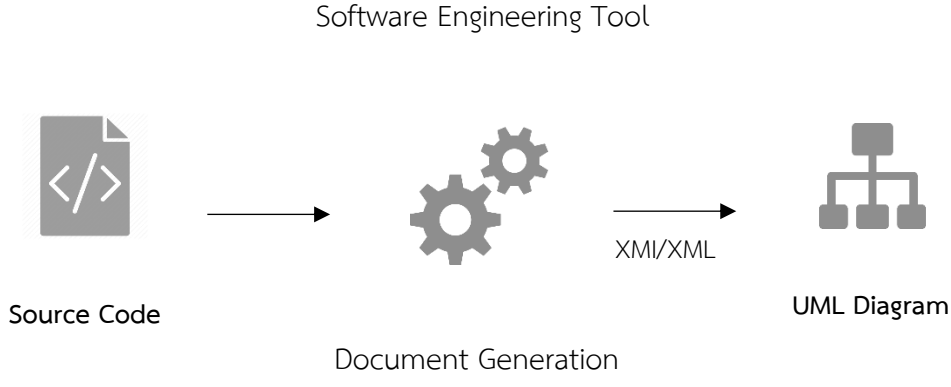
2.1 เทคโนโลยีที่เกี่ยวข้อง

เทคโนโลยีที่เกี่ยวข้องในงานวิจัย ได้แก่ วิศวกรรมย้อนกลับ (Reverse Engineering) ยูเอ็มแอลเมต้าโมเดล (UML Meta Model) เอกซ์เอ็มไอ (XML Metadata Interchange: XMI) ภาษาฟอร์แทรน (Fortran) เครื่องมือ ForUML และยูเอ็มแอลซีเควนซ์ไดอะแกรม (UML Sequence Diagram)

2.1.1 วิศวกรรมย้อนกลับ (Reverse Engineering)

วิศวกรรมย้อนกลับเป็นกระบวนการวิเคราะห์ระบบ เพื่อจะระบุส่วนประกอบของระบบและความสัมพันธ์ระหว่างส่วนประกอบของระบบ โดยจะทำการแสดงแบบจำลองของระบบขึ้นมาจากสิ่งที่เป็นนามธรรมหรือสิ่งที่ไม่มีรูปร่าง เพื่อให้สามารถนำมาใช้ในการวิเคราะห์โครงสร้างการทำงานและทำความเข้าใจการทำงานในแต่ละส่วนของระบบได้ โดยที่วิศวกรรมย้อนกลับนั้นไม่ได้ทำการเปลี่ยนแปลงการทำงานของระบบ หรือทำให้เกิดผลลัพธ์ขึ้นมาใหม่ (Abran, et al., 2001)

การทำวิศวกรรมย้อนกลับสำหรับซอฟต์แวร์โดยส่วนใหญ่จะเกี่ยวข้องกับซอร์สโค้ด บางครั้งจะเรียกว่า วิศวกรรมรหัสย้อนกลับ (Reverse Code Engineering) ซึ่งจะเป็นกระบวนการวิเคราะห์ส่วนต่าง ๆ ของซอร์สโค้ด เพื่อทำความเข้าใจซอร์สโค้ดในแต่ละส่วน โดยวิศวกรรมรหัสย้อนกลับมักจะใช้สำหรับการวิเคราะห์รหัสไบนารี่ (Binary Code) (Willems and Freiling, 2012) ตัวอย่างเช่น ซอฟต์แวร์หรือโปรแกรมที่ใช้งานในปัจจุบันนั้น อยู่ในรูปของรหัสไบนารี่หรืออยู่ในรูปแบบของภาษาเครื่องที่มนุษย์ไม่สามารถอ่านหรือทำความเข้าใจได้ สำหรับตัวอย่างของซอฟต์แวร์ที่สามารถทำการวิศวกรรมย้อนกลับจากรหัสไบนารี่กลับมาอยู่ในรูปของซอร์สโค้ด (Decompile) นั้น ได้แก่ Jad (Rukin, 2017) ซึ่งเป็นซอฟต์แวร์ที่สามารถแปลงรหัสไบนารี่ของภาษาจาวา เช่น ไฟล์ที่มีนามสกุล .class เพื่อให้กลับมาอยู่ในรูปของซอร์สโค้ดที่นักพัฒนาซอฟต์แวร์สามารถอ่านและทำความเข้าใจได้ นอกจากนี้ซอฟต์แวร์ที่เกี่ยวข้องกับวิศวกรรมย้อนกลับในปัจจุบันมีความสามารถในการแปลงข้อมูลในรูปแบบต่าง ๆ ที่หลากหลาย เช่น ArgoUML (Bogdan, et al., 2018), Modelio (Modeliosoft, 2017), UML Designer (Obeo, 2017) และ Umbrello (Team, 2017) ซึ่งซอฟต์แวร์ดังกล่าวจะมีความสามารถหลัก คือ การแปลงซอร์สโค้ดให้กลับมาอยู่ในรูปของยูเอ็มแอลไดอะแกรม เช่น ยูเอ็มแอลคลาสไดอะแกรม และยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งกระบวนการทำงานดังกล่าวแสดงดังรูปที่ 2.1



รูปที่ 2.1 ตัวอย่างของกระบวนการแปลงจากซอร์สโค้ดไปเป็นยูเอ็มแอลไดอะแกรม

จากรูปที่ 2.1 เป็นการแสดงตัวอย่างของกระบวนการแปลงซอร์สโค้ดไปเป็นยูเอ็มแอลไดอะแกรม ซึ่งเป็นหนึ่งในวิธีการที่พบได้โดยทั่วไปในเครื่องมือทางด้านวิศวกรรมย้อนกลับ โดย

กระบวนการวิศวกรรมย้อนกลับจะเริ่มตั้งแต่การนำซอร์สโค้ดเข้าสู่เครื่องมือวิศวกรรมย้อนกลับ ซึ่งภาษาของซอร์สโค้ดนั้นจะขึ้นอยู่กับความสามารถของเครื่องมือ เช่น ภาษาจาวา ภาษาฟอร์แทรน ภาษาซีพลัสพลัส หรือภาษาอื่น ๆ หลังจากนั้นเครื่องมือจะทำการวิเคราะห์ซอร์สโค้ดเพื่อค้นหาโครงสร้างและความสัมพันธ์ของระบบ โดยจะเก็บอยู่ในรูปแบบของ XML Metadata Interchange (XMI) ซึ่งเป็นรูปแบบที่นิยมใช้ในการแลกเปลี่ยนข้อมูลเพื่อแสดงยูเอ็มแอลไดอะแกรม (Linzhang, et al., 2017; Mythily, et al., 2018; Nikulchev and Deryugina, 2016) ในขั้นตอนสุดท้ายเครื่องมือการวิศวกรรมย้อนกลับจะทำการสร้างยูเอ็มแอลไดอะแกรม เช่น ยูเอ็มแอลคลาสไดอะแกรม และยูเอ็มแอลซีเควนซ์ไดอะแกรม

Andritsos และ Miller (Andritsos and Miller, 2001) ได้กล่าวถึงระบบซอฟต์แวร์โดยส่วนใหญ่เมื่อเริ่มมีอายุมากขึ้น การที่จะทำความเข้าใจ และบำรุงรักษานั้นเป็นเรื่องยาก บางครั้งอาจทำให้ระบบไม่มีประสิทธิภาพ และมีค่าใช้จ่ายเพิ่มสูงขึ้น ดังนั้นชุมชนด้านวิศวกรรมซอฟต์แวร์จึงให้ความสำคัญกับการสร้างเครื่องมือที่ช่วยให้นักวิเคราะห์ระบบสามารถมองเห็นถึงโครงสร้างของระบบดังกล่าว ซึ่งเครื่องมือทางด้านวิศวกรรมย้อนมีบทบาทที่สำคัญในวิศวกรรมซอฟต์แวร์ดังนี้

(1) การวิเคราะห์โปรแกรม (Program Analysis) จะเป็นการวิเคราะห์ซอร์สโค้ด และการสกัดข้อมูลที่เกี่ยวข้อง เช่น คลาส หรือเมท็อด (Methods)

(2) การรับรู้แผน (Plan Recognition) จะเป็นการระบุรูปแบบ (Pattern) โดยรูปแบบนั้นอาจจะเป็นรูปแบบของพฤติกรรมหรือโครงสร้างของระบบ

(3) การกำหนดแนวคิด (Concept Assignment) จะเป็นการค้นหารูปแบบภายในระบบ โดยจะรวมไปถึงการค้นหาโครงสร้างของซอร์สโค้ดและความสัมพันธ์ระหว่างองค์ประกอบของระบบ

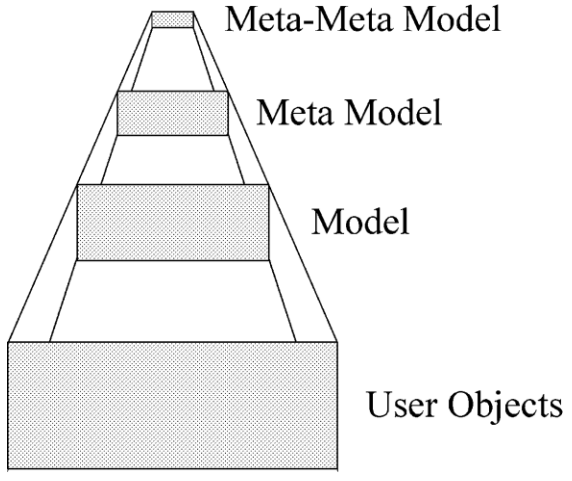
(4) การสร้างเอกสารขึ้นใหม่ (Re-documentation) จะเป็นการสร้างเอกสารสำหรับระบบที่ไม่มีเอกสาร หรือระบบเดิมที่เอกสารไม่ได้มีการปรับปรุง

วิศวกรรมย้อนกลับจะช่วยให้นักพัฒนาเข้าใจโครงสร้างของระบบซอฟต์แวร์ที่มีขนาดใหญ่ โดยเฉพาะอย่างยิ่งระบบดั้งเดิมที่มีการพัฒนาต่อเนื่องกันมา โดยที่ระบบเหล่านี้จะเป็นระบบที่มีความซับซ้อน และอาจจะไม่มีเอกสารเอกสารการออกแบบ หรือเอกสารการออกแบบเกิดสูญหาย หรือเอกสารการออกแบบไม่ได้รับการปรับปรุงให้ตรงตามความเป็นจริง ดังนั้นวิศวกรรมย้อนกลับจึงสามารถ

ช่วยกู้คืนเอกสารการออกแบบ ทำให้นักพัฒนาสามารถนำไปเปรียบเทียบกับโครงสร้างการทำงานของระบบ เพื่อใช้ในการวิเคราะห์ ทำความเข้าใจ และปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ได้ดีขึ้น

2.1.2 ยูเอ็มแอลเมต้าโมเดล (UML Meta Model)

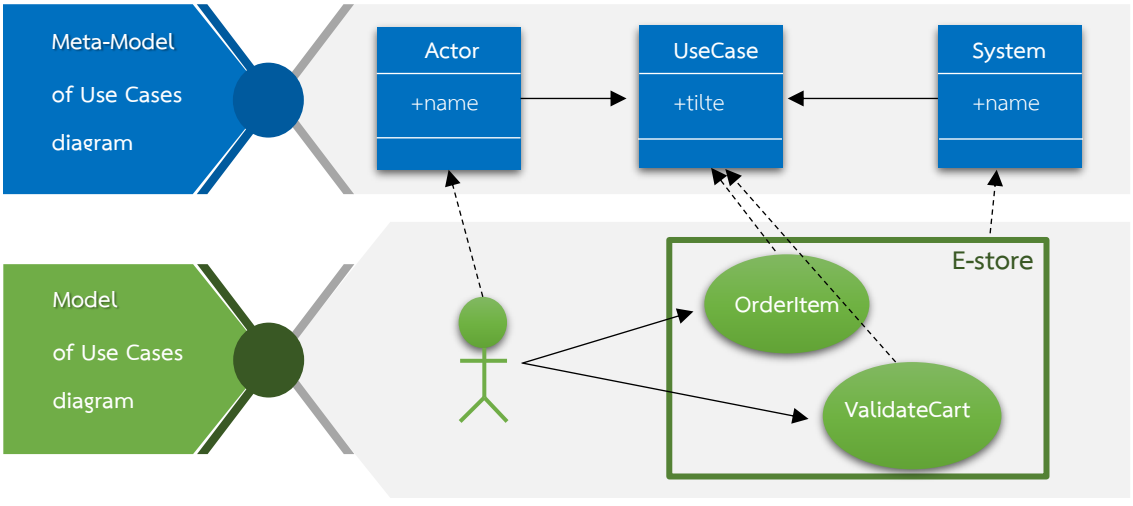
ยูเอ็มแอลเป็นภาษาที่ใช้สำหรับอธิบายโมเดลต่าง ๆ ซึ่งเป็นมาตรฐานในการสร้างโมเดลเชิงวัตถุ หรือเป็นมาตรฐานสำหรับสร้างแบบพิมพ์เขียวให้กับระบบซอฟต์แวร์ โดยยูเอ็มแอลถูกนำไปใช้ในการสร้างมุมมองของระบบ กำหนดรายละเอียดของระบบ และช่วยพัฒนาระบบ สำหรับเมต้าโมเดลจะเป็นโมเดลที่ใช้สำหรับอธิบายโมเดลอื่น ๆ ยูเอ็มแอลเมต้าโมเดลมีวัตถุประสงค์เพื่อกำหนดคำจำกัดความของวากยสัมพันธ์ (Syntax) และความหมายสำหรับโครงสร้าง หรือองค์ประกอบในยูเอ็มแอลโมเดล ซึ่งเมต้าโมเดลจะช่วยให้นักพัฒนาเข้าใจถึงความหมายของโมเดลที่ต้องการจะสื่อได้ถูกต้องตรงกัน และสามารถสร้างแบบจำลองต่าง ๆ ได้ตรงตามมาตรฐานของยูเอ็มแอล



รูปที่ 2.2 สถาปัตยกรรมเมต้าโมเดล 4 ระดับของยูเอ็มแอล (Medvidovic, et al., 2002)

จากรูปที่ 2.2 เป็นการแสดงให้เห็นถึงสถาปัตยกรรมเมต้าโมเดลของยูเอ็มแอลทั้ง 4 ระดับ โดยที่ระดับล่างสุด User Objects จะแสดงถึงรายละเอียดของวัตถุ เช่น Chair, Desk ในระดับถัดมา จะเป็นโมเดลที่ไว้สำหรับอธิบายถึงวัตถุนั้น เช่น Product, Unit, Price, Sale, Detail ในระดับถัดมา Meta Model จะเป็นโมเดลที่ไว้สำหรับอธิบายโมเดลนั้นอีกที เช่น Class, Attribute, Operation ใน

ระดับสุดท้าย Meta-Meta Model จะเป็นโมเดลไว้สำหรับอธิบายเมต้าโมเดลข้างต้น เช่น MetaClass, MetaAtribute, MetaOperation



รูปที่ 2.3 ตัวอย่างเมต้าโมเดลของระบบสั่งซื้อสินค้า

จากรูปที่ 2.3 เป็นการแสดงตัวอย่างเมต้าโมเดลของระบบสั่งซื้อสินค้า ซึ่งจะเป็นเมต้าโมเดลที่อธิบายถึงยูสเคสไดอะแกรม (Use Case Diagram) ของระบบสั่งซื้อสินค้า โดยเมต้าโมเดลนี้จะแสดงอยู่ในรูปของคลาสไดอะแกรมที่ใช้สำหรับอธิบายถึงโมเดลของยูสเคสไดอะแกรม โดยที่โมเดลของยูสเคสไดอะแกรมจะเป็นการอธิบายถึงฟังก์ชันการทำงานของระบบ E-store ซึ่งประกอบด้วย 2 ฟังก์ชัน คือ ผู้ใช้งานทำการสั่งซื้อสินค้า (OrderItem) และผู้ใช้งานทำการตรวจสอบตะกร้าสินค้า (ValidateCart) ซึ่งเมื่อนำโมเดลยูสเคสไดอะแกรมมาสร้างเป็นเมต้าโมเดลยูสเคสไดอะแกรมของระบบสั่งซื้อสินค้าจะได้คลาสทั้งหมด 3 คลาส คือ Actor แสดงถึง ผู้ใช้งานโดยมีการกำหนดรายละเอียดของชื่อผู้ใช้ UseCase แสดงถึงฟังก์ชันการทำงานของระบบโดยมีการกำหนดรายละเอียดของชื่อฟังก์ชัน และ System แสดงถึงระบบโดยมีการกำหนดรายละเอียดของชื่อระบบ โดยทั้ง 3 คลาสจะมีความสัมพันธ์กันเพื่อแสดงรายละเอียดการทำงานของยูสเคสไดอะแกรมข้างต้น

2.1.3 เอกซ์เอ็มไอ (XML Metadata Interchange: XMI)

เอกซ์เอ็มไอเป็นรูปแบบมาตรฐานของ OMG (Object Management Group) (OMG, 2017) ที่แสดงให้เห็นถึงรายละเอียดในการแลกเปลี่ยนข้อมูลของยูเอ็มแอลโมเดล ซึ่งจะอยู่ในรูปแบบของเอกสารเอกซ์เอ็มแอล (XML) ที่สามารถเก็บข้อมูลลงในไฟล์ และสามารถนำข้อมูลที่บันทึกไว้มาแสดง

ผลได้ผ่านซอฟต์แวร์ที่รองรับ โดยเอกซ์เอ็มแอลเป็นภาษา Markup ชนิดหนึ่งที่ถูกพัฒนาโดย W3C (World Wide Web Consortium) (Bray, et al., 2008) โดยมีแนวคิดหลัก คือ เป็นข้อมูลมาตรฐานสำหรับแลกเปลี่ยนข้อมูลในแพลตฟอร์มที่แตกต่างกัน ดังนั้นภาษาเอกซ์เอ็มแอลจึงเป็นภาษาที่ออกแบบมาเพื่อเป็นภาษากลางในการแลกเปลี่ยนข้อมูลของภาษาที่มีมาตรฐานที่แตกต่างกันในปัจจุบันเช่น การแลกเปลี่ยนข้อมูลระหว่างซอร์สโค้ดภาษาจาวากับยูเอ็มแอลไดอะแกรม สำหรับเอกสารเอกซ์เอ็มไอนั้นมีลักษณะเป็นเอกสารเอกซ์เอ็มแอลโดยมีองค์ประกอบ (Elements) และแอตทริบิวท์ (Attributes) เป็นไปตามมาตรฐานของ OMG ซึ่งองค์ประกอบที่สำคัญของเอกซ์เอ็มไอแสดงรายละเอียดดังต่อไปนี้

(1) เอกสารเอกซ์เอ็มไอมีการกำหนดเวอร์ชันของเอกซ์เอ็มแอลและการเข้ารหัสของการสารยกตัวอย่างเช่น <? XML version="1.0" ENCODING="UTF-8" ?>

(2) เอกสารเอกซ์เอ็มไอมีการกำหนดเวอร์ชันของยูเอ็มแอลไดอะแกรมและเอกซ์เอ็มไอยกตัวอย่างเช่น <uml:Model xmlns:uml="http://schema.omg.org/spec/UML/2.1.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmi:version="2.1" >

(3) เอกสารเอกซ์เอ็มไอมีการกำหนดรายละเอียดแยกตามประเภทของยูเอ็มแอลไดอะแกรมยกตัวอย่างเช่น <packagedElement xmi:type="uml:Interaction" xmi:id="xxx" >

```
<xmi:XMI xmlns:UML="http://schema.omg.org/spec/UML/1.4"
                xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">
    <UML:Package xmi:id="ppp" xmi:label="p1">
            <ownedElement xmi:type="UML:Class" xmi:id="ccc" xmi:label="c1">
                    <feature xmi:type="UML:Attribute" xmi:label="a1"/>
                    <feature xmi:type="UML:Attribute" xmi:label="a2"/>
            </ownedElement>
    </UML:Package>
</xmi:XMI>
```

รูปที่ 2.4 ตัวอย่างของเอกสารเอกซ์เอ็มไอ

จากรูปที่ 2.4 เป็นการแสดงตัวอย่างของเอกสารเอกซ์เอ็มไอ ซึ่งเอกสารเอกซ์เอ็มไอดังกล่าวจะเป็นเอกสารที่ใช้สำหรับแสดงข้อมูลของยูเอ็มแอลคลาสไดอะแกรม โดยจะมีข้อกำหนดตามรูปแบบของ OMG ที่ระบุถึงเวอร์ชันของยูเอ็มแอลไดอะแกรม คือ xmlns:UML="http://schema.omg.org/spec/UML/1.4" เวอร์ชันของเอกซ์เอ็มไอ คือ xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" โดยองค์ประกอบชื่อ ownedElement จะมีการกำหนดประเภทของยูเอ็มแอลไดอะแกรมเป็นคลาสไดอะแกรมนั้น คือ xmi:type="UML:Class" โดยจะมีหมายเลขกำกับของคลาส คือ

xmi:id="ccc" และชื่อของคลาส คือ xmi:label="c1" ซึ่งภายใต้องค์ประกอบ ownedElement จะมีการกำหนดแอตทริบิวท์ของคลาสหรือรายละเอียดของคลาส คือ xmi:type="UML:Attribute" โดยมีชื่อของแอตทริบิวท์ ได้แก่ xmi:label="a1" และ xmi:label="a2" ตามลำดับ

2.1.4 ยูเอ็มแอลซีเควนซ์ไดอะแกรม (UML Sequence Diagram)

ยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นแผนภาพที่แสดงให้เห็นถึงเหตุการณ์ (Scenarios) ที่มีความเกี่ยวข้องกันของแต่ละยูสเคส (Use Case) และแสดงถึงการปฏิสัมพันธ์กันระหว่างวัตถุไปตามลำดับของเวลา โดยจะแสดงถึงลำดับการส่งสาร (Messages) กันระหว่างวัตถุ (Bell, 2004) ยูเอ็มแอลซีเควนซ์ไดอะแกรมประกอบด้วยกัน 2 แกน คือ แกนแนวตั้งที่แสดงถึงลำดับเวลาของการส่งสาร และแกนแนวนอนที่แสดงถึงการส่งสารของวัตถุไปยังไลฟ์ไลน์ (Lifeline) โดยการปฏิสัมพันธ์กันระหว่างวัตถุมีองค์ประกอบดังนี้

(1) ชื่อของวัตถุ (Object Name) เป็นการระบุชื่อของวัตถุบนไลฟ์ไลน์ โดยวัตถุจะมีการทำงานเรียงจากซ้ายไปขวา

(2) ไลฟ์ไลน์ เป็นเส้นประที่ลากตามแนวตั้งจากวัตถุ

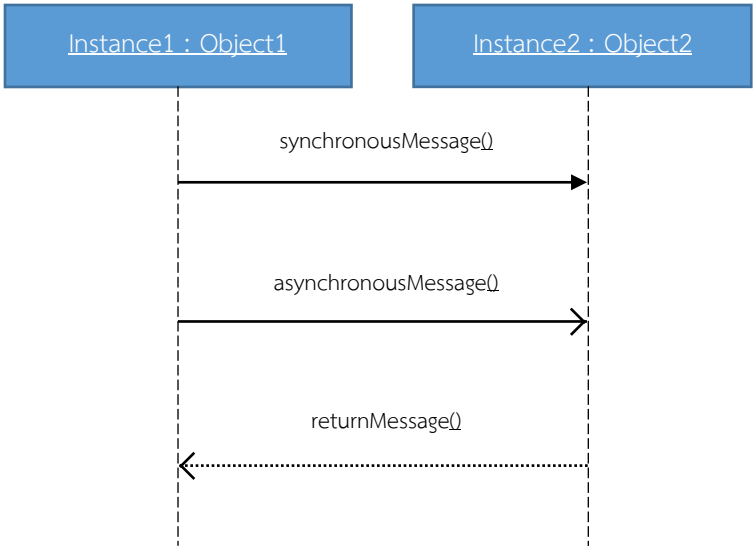
(3) การดำเนินงาน (Activation) เป็นการแสดงถึงกล่องสี่เหลี่ยมที่อยู่บนเส้นของไลฟ์ไลน์ โดยจะใช้แทนช่วงเวลาการทำงานที่เกิดขึ้นบนไลฟ์ไลน์

สำหรับสารที่ส่งระหว่างวัตถุนั้น แสดงดังรูปที่ 2.5 ซึ่งจะมีลักษณะดังนี้

(1) สารสมวาร (Synchronous Message) เป็นการส่งสารแบบรอคอยคำตอบหรือการตอบกลับก่อนที่จะทำงานอื่น ๆ ต่อไป

(2) สารอสมวาร (Asynchronous Message) เป็นการส่งสารแบบไม่รอคอยคำตอบ ซึ่งจะไม่มีการหยุดทำงานของผู้ส่ง ทำให้ผู้ส่งสามารถทำงานต่อเนื่องได้

(3) สารย้อนกลับ (Return Message) เป็นการส่งสารในกรณีที่ต้นทางเริ่มการติดต่อแล้วปลายทางต้องมีการติดต่อกลับ



รูปที่ 2.5 การส่งสารระหว่างวัตถุทั้ง 3 แบบ

สารดังกล่าวจะบ่งบอกความสัมพันธ์ระหว่างวัตถุบนไลฟ์ไลน์ โดยจะมีลูกศรเพื่อบ่งบอกทิศทางของสารต่าง ๆ ที่ถูกส่งระหว่างวัตถุ และมีกล่องสี่เหลี่ยมที่บ่งบอกจุดต้นทางและปลายทางของสารที่ไปยังเมท็อด ในการตอบสนองของสารแต่ละสารจะถูกแสดงโดยลูกศรระหว่างไลฟ์ไลน์ของวัตถุสองวัตถุ โดยชื่อของสารที่แสดงบนลูกศรจะประกอบไปด้วยชื่อของฟังก์ชันการทำงาน และชื่อตัวแปรของอาร์กิวเมนต์ที่มีการเรียกใช้ตอนส่งค่าไป สำหรับสารที่ส่งมาที่ตัวเองจะแสดงโดยให้ลูกศรวนกลับมาที่ไลฟ์ไลน์เดิม

2.1.5 ภาษาฟอร์แทรน (Fortran)

ภาษาฟอร์แทรนเป็นภาษาที่ออกแบบเพื่อประยุกต์ใช้งานในด้านที่เกี่ยวข้องกับตัวเลข (Metcalf, 2011; Reid, 2003; Reid, 2008) เช่น งานด้านวิทยาศาสตร์และวิศวกรรมศาสตร์ คำว่า “FORTRAN” เป็นคำย่อมาจาก FORmula TRANslation โดยภาษาฟอร์แทรนเป็นภาษาโปรแกรมระดับสูงภาษาแรกของโลก ซึ่งเริ่มต้นพัฒนาขึ้นในปลายปี ค.ศ. 1953 โดยบริษัท ไอบีเอ็ม (IBM) นำทีมโดย จอห์น แบคคัส (John Backus) และได้เผยแพร่ให้กับผู้ใช้งานเมื่อปี ค.ศ. 1957 โดยมีการพัฒนาเรื่อยมาตั้งแต่ Fortran I, Fortran Ii, Fortran Iv, Fortran 66, Fortran 77, Fortran 90, Fortran 95, Fortran 2003 และ Fortran 2008 ซึ่งมีการเผยแพร่ในปี ค.ศ. 2010 และได้กลายเป็นมาตรฐานที่มีการใช้งานอยู่ในปัจจุบัน

ปัจจุบันภาษาฟอร์แทรนได้ถูกพัฒนาให้มีลักษณะเชิงวัตถุ (Object-Oriented) เช่นเดียวกับภาษาจาวา หรือ ซีพลัสพลัส หรือเรียกอีกอย่างว่า Modern Fortran (Clerman and Spector, 2011) เพื่อให้รองรับการพัฒนาซอฟต์แวร์ที่มีความซับซ้อน และเป็นการนำหลักการทางด้านวิศวกรรมซอฟต์แวร์มาประยุกต์ใช้ให้ซอฟต์แวร์มีประสิทธิภาพมากยิ่งขึ้นโดยเฉพาะซอฟต์แวร์ทางด้านวิทยาศาสตร์และวิศวกรรม จะเห็นได้ว่ามีนักวิทยาศาสตร์และนักวิจัยทางด้านวิศวกรรมซอฟต์แวร์จำนวนมากให้ความสนใจและมีการนำ Modern Fortran มาใช้ในการพัฒนาซอฟต์แวร์ (Barbieri, et al., 2011; Budiardja, et al., 2012; Morris, et al., 2012; Rouson, et al., 2010) นอกจากนั้นในปัจจุบันผู้ผลิตคอมไพเลอร์สำหรับภาษาฟอร์แทรนจำนวนมากได้พัฒนาให้คอมไพเลอร์มีความสามารถรองรับ Modern Fortran เช่น Numerical Algorithm Group (NAG) (Brian, 2016), GNU Fortran (Paul, et al., 2016), IBM XL Fortran (IBM, 2016), Cary (Cray, 2016), และ Intel Fortran (Intel, 2016)

ตารางที่ 2.1 การเปรียบเทียบโครงสร้างของภาษาเชิงวัตถุระหว่างภาษาฟอร์แทรน และภาษาจาวา (Rouson, et al., 2010)

Object-Oriented Equivalent	Fortran	Java
Abstract Data Type (ADT)	Derived Type	Class
Attribute	Component	Property
Method	Type-bound Procedure	Method
Parent Class	Parent Type	Base Class
Child Class	Extend Type	Subclass
Package	Module	Package
Static Polymorphism	Generic Interface	Overloading
Abstract Method	Deferred Procedure Binding	Abstract
Primitive Type	Intrinsic Type	Primitive Type

จากตารางที่ 2.1 แสดงให้เห็นถึงตัวอย่างการเปรียบเทียบโครงสร้างของภาษาเชิงวัตถุระหว่างภาษาฟอร์แทรนและภาษาจาวา โดยที่ภาษาจาวานั้นเป็นภาษาที่นิยมใช้ในการเขียนโปรแกรมเชิงวัตถุในปัจจุบัน (IEEE, 2017) ขณะที่ภาษาฟอร์แทรนเองก็มีคุณสมบัติที่สำคัญของภาษาเชิงวัตถุ เช่น การสืบทอดคุณสมบัติ (Inheritance) การพ้องรูป (Polymorphism) การจัดสรรแบบพลวัต (Dynamic Type Allocation) และการกำหนดขอบเขตการทำงาน (Type-bound Procedures) ซึ่งเมื่อนำภาษาจาวาและภาษาฟอร์แทรนมาเปรียบเทียบโครงสร้างเชิงวัตถุสามารถระบุส่วนที่สำคัญที่มีความคล้ายคลึงกันคือ ในภาษาจาวาจะมีการสร้างคลาสขึ้นมาเพื่อระบุรายละเอียดการทำงานต่าง ๆ ของระบบ ซึ่งในภาษาฟอร์แทรนจะมีการสร้างคลาสขึ้นมาและมีหน้าที่การทำงานที่เหมือนกันแต่จะมีชื่อเรียกที่ต่างกันนั้น คือ Type แสดงดังตารางที่ 2.2 ในภาษาจาวาจะมีการกำหนดแพ็คเกจของคลาสขึ้นเพื่อระบุการทำงานของคลาสว่าอยู่ภายใต้แพ็คเกจใด ซึ่งในภาษาฟอร์แทรนจะมีการกำหนดแพ็กเกจของคลาสขึ้นมาเหมือนกันและมีหน้าที่การทำงานที่เหมือนกันแต่จะมีชื่อเรียกที่ต่างกันนั้น คือ Module แสดงดังตารางที่ 2.3 ในภาษาจาวาจะมีการสร้างเมท็อดขึ้นมาเพื่อกำหนดการทำงานของคลาส ซึ่งในภาษาฟอร์แทรนจะมีการสร้างเมท็อดขึ้นมาเหมือนกัน แต่เมท็อดของภาษาฟอร์แทรนนั้นจะมีการแบ่งส่วนการทำงานออกเป็น 2 เมท็อด คือ เมท็อดที่เป็นฟังก์ชัน (Function) และเมท็อดที่เป็นซับรูทีน (Subroutine) โดยที่เมท็อดของภาษาจาวานั้นจะไม่มีการแบ่งส่วนการทำงานเกิดขึ้นแสดงดังตารางที่ 2.4

ตารางที่ 2.2 ตัวอย่างซอร์สโค้ดในส่วนคลาสของภาษาฟอร์แทรน

ซอร์สโค้ดภาษาฟอร์แทรน	คำอธิบาย
type A **real ::** x **end type A**	มีการสร้างคลาสที่มีชื่อว่า A โดยภายในคลาสมีการสร้างตัวแปรที่มีชนิดเป็นจำนวนจริงชื่อว่า x

ตารางที่ 2.3 ตัวอย่างซอร์สโค้ดในส่วนแพ็กเกจของภาษาฟอร์แทรน

ซอร์สโค้ดภาษาฟอร์แทรน	คำอธิบาย

module class_B
  type B
  integer :: y
  end type B
end module class_B
	มีการสร้างแพ็กเกจที่มีชื่อว่า class_B โดยภายใต้แพ็กเกจมีการสร้างคลาสที่มีชื่อว่า B โดยภายในคลาสมีการสร้างตัวแปรที่มีชนิดเป็นตัวเลขชื่อว่า y

ตารางที่ 2.4 ตัวอย่างซอร์สโค้ดในส่วนเมท็อดของภาษาฟอร์แทรน

ซอร์สโค้ดภาษาฟอร์แทรน	คำอธิบาย

subroutine method_1 (this)
  type(A), intent(in) :: this
  real :: x
  x = method_2 (this)
end **subroutine method_1**
 	
function method_2 (this) result (y)
  type(A), **intent**(out) **::** y
end function method_2
	มีการสร้างเมท็อดที่เป็นซับรูทีนชื่อว่า method_1 โดยภายในซับรูทีนมีการประกาศคลาสชื่อว่า A และกำหนดเมท็อดนี้ให้ไม่มีการส่งค่ากลับ มีการสร้างตัวแปรที่มีชนิดเป็นจำนวนจริงชื่อว่า x มีการเรียกใช้งานแบบฟังก์ชันไปยัง method_2 โดยเก็บค่าอยู่ในตัวแปร x มีการสร้างเมท็อดที่เป็นฟังก์ชันชื่อว่า method_2 โดยภายในฟังก์ชันมีการประกาศคลาสชื่อว่า A และกำหนดเมท็อดนี้ให้มีการส่งค่ากลับผ่านตัวแปร y

โดยทั่วไปภาษาเชิงวัตถุ เช่น ภาษาจาวา และภาษาซีพลัสพลัส จะมีการกำหนดให้คลาสที่ประกอบด้วยข้อมูลและเมท็อดมีการแบ่งสร้างอินสแตนซ์ (Instance) ของคลาส เพื่อเรียกการทำงานและกำหนดข้อมูลของคลาสนั้น แต่ในภาษาฟอร์แทรนจะมีลักษณะเป็นมอดูล (Modules) ที่ประกอบด้วยข้อมูลต่าง ๆ ซึ่งจะไม่มีแนวคิดในการแบ่งสร้างอินสแตนซ์ของมอดูล แต่จะมีการจัดสรรชนิดของตัวแปรส่งผ่านทางเมท็อดที่อยู่ในมอดูลนั้น ซึ่งเมท็อดที่กล่าวถึงในที่นี้ คือ ฟังก์ชัน และซับรูทีน ดังตัวอย่างรูปที่ 2.6

```
module class_Circle
  implicit none
  private
  public :: Circle, circle_area, circle_print

  real :: pi = 3.1415926535897931d0 ! Class-wide private constant

  type Circle
     real :: radius
  end type Circle
contains
  function circle_area(this) result(area)
    type(Circle), intent(in) :: this
    real :: area
    area = pi * this%radius**2
  end function circle_area

  subroutine circle_print(this)
    type(Circle), intent(in) :: this
    real :: area
    area = circle_area(this)  ! Call the circle_area function
    print *, 'Circle: r = ', this%radius, ' area = ', area
  end subroutine circle_print
end module class_Circle

program circle_test
  use class_Circle
  implicit none

  type(Circle) :: c     ! Declare a variable of type Circle.
  c = Circle(1.5)       ! Use the implicit constructor, radius = 1.5.
  call circle_print(c)  ! Call a class subroutine
end program circle_test
```

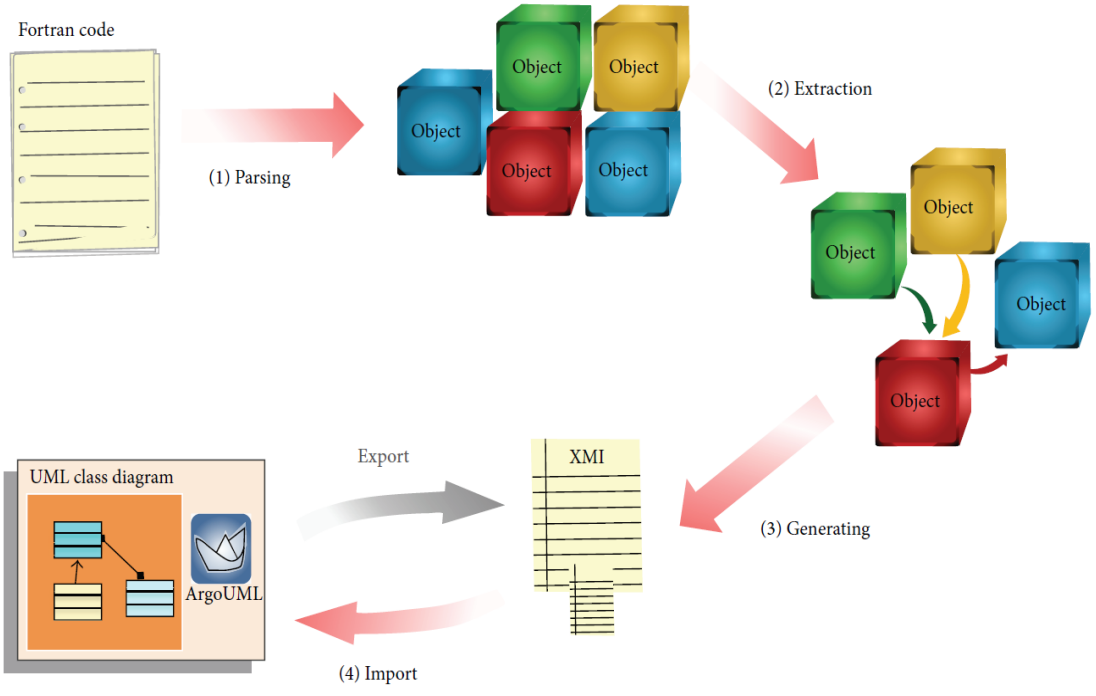
รูปที่ 2.6 ตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรน (Akin, 2003)

จากรูปที่ 2.6 เป็นการแสดงตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรน ประกอบด้วย 2 คลาส คือ circle_test และ Circle โดยคลาส circle_test จะทำหน้าที่เป็นโปรแกรมหลักเพื่อเรียกการทำงานของคลาส Circle ที่อยู่ภายใต้มอดูล class_Circle โดยจะมีการเรียกการทำงานที่เป็นซับรูทีน คือ call circle_print(c) ซึ่งจะมีการกำหนดการเรียกใช้งานไปยังคลาส Circle ผ่านตัวแปร c จากนั้นภายในซับรูทีนชื่อ circle_print ของคลาส Circle จะมีการเรียกการทำงานที่เป็นฟังก์ชัน คือ area = circle_area(this) ซึ่งจะมีการเรียกใช้งานฟังก์ชันภายในคลาสเดียวกัน

เนื่องจากภาษาฟอร์แทรนยังค่อนข้างเป็นเรื่องใหม่ในโลกของการพัฒนาซอฟต์แวร์เชิงวัตถุ (Object-Oriented Programming) จึงทำให้เครื่องมือที่เกี่ยวข้องยังมีน้อย และยังไม่ได้นำหลักการทางด้านวิศวกรรมซอฟต์แวร์เข้ามาช่วยมากนัก หากเปรียบเทียบกับภาษาเชิงวัตถุอื่น ๆ นั้น เครื่องมือทางด้านวิศวกรรมซอฟต์แวร์ใน Modern Fortran นั้นยังมีอยู่น้อยมาก โดยเฉพาะเครื่องมือในกลุ่มของการทำความเข้าใจโปรแกรม ซึ่งเป็นเครื่องมือที่จะช่วยให้นักพัฒนา และนักออกแบบซอฟต์แวร์สามารถทำความเข้าใจโค้ด หรือระบบซอฟต์แวร์ได้ง่ายขึ้น

2.1.6 เครื่องมือ ForUML

ForUML (Nanthaamornphong, et al., 2015) เป็นเครื่องมือที่สามารถทำการวิศวกรรมย้อนกลับจากซอร์สโค้ดที่พัฒนาขึ้นด้วยภาษาฟอร์แทรนเป็นยูเอ็มแอลไดอะแกรม นักวิจัยได้พัฒนาวิธีการการแปลงซอร์สโค้ดไปเป็นยูเอ็มแอลไดอะแกรมโดยมีรากฐานแนวคิดมาจาก โครงสร้างจำลอง (Meta-Model) ของภาษาเชิงวัตถุที่ถูกพัฒนาขึ้นมาโดย Lethbridge และคณะ (Lethbridge, et al., 2004) ซึ่งทีมของ Lethbridge ได้ทำการออกแบบโครงสร้างที่เรียกว่า The Dagstuhl Middle Metamodel (DMM) ซึ่งเป็นโครงสร้างจำลองที่ได้รับการพิสูจน์ในทางปฏิบัติแล้วว่าเป็นประโยชน์ต่อกระบวนการวิศวกรรมย้อนกลับซอฟต์แวร์ โดย ForUML จะประยุกต์ใช้โครงสร้างดังกล่าวกับซอร์สโค้ดของภาษาฟอร์แทรน สำหรับกระบวนการแปลงนั้นมีอยู่ด้วยกัน 4 ขั้นตอน ดังรูปที่ 2.7 ซึ่งมีรายละเอียดแต่ละขั้นตอนดังนี้



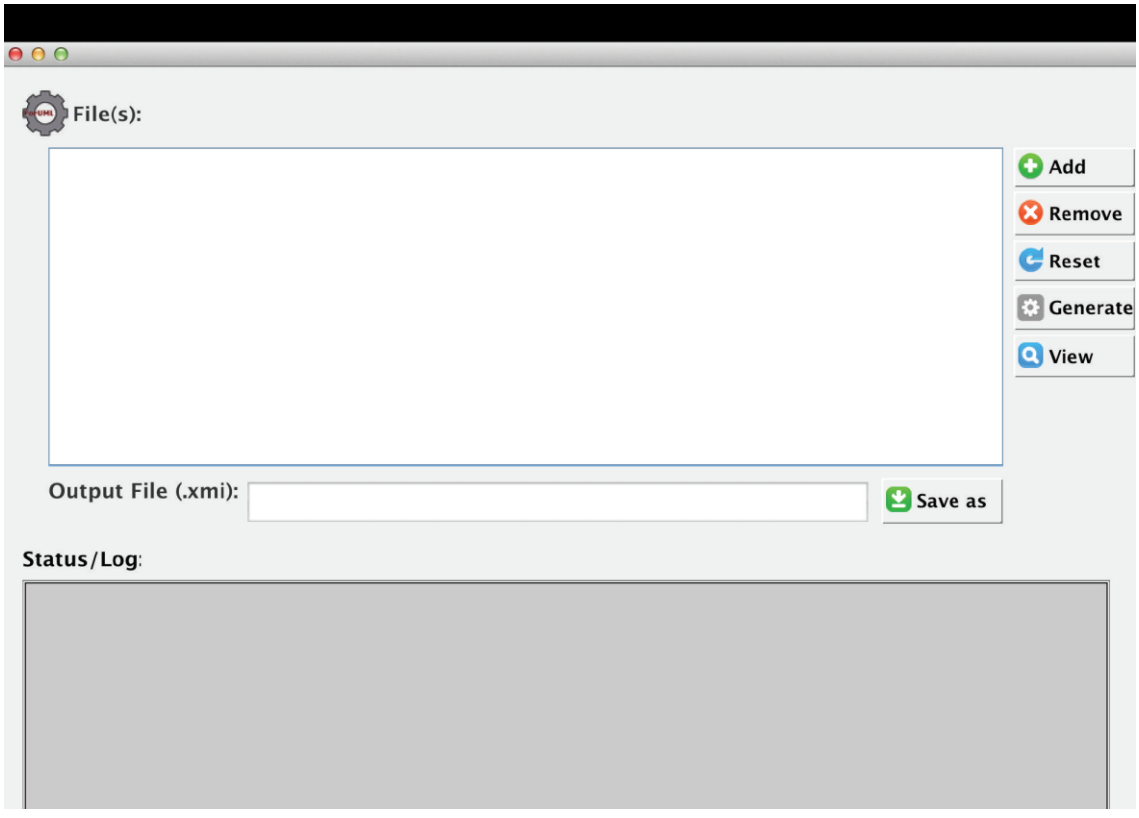
รูปที่ 2.7 กระบวนการแปลงซอร์สโค้ดไปเป็นยูเอ็มแอลไดอะแกรมของเครื่องมือ ForUML

(1) Parsing เป็นกระบวนการวจีวิภาคโค้ดออกเป็นส่วนย่อย ๆ (Element) โดยใช้ไลบรารี Open Fortran Parser (OFP) (Craig, et al., 2018) ซึ่งในกระบวนการตัดคำนี้จะอาศัยไฟล์ไวย์กรณ์ (Grammar) และวากยสัมพันธ์ ของภาษาฟอร์แทรนที่ได้ถูกพัฒนาไว้แล้วในไลบรารี OFP โดยกระบวนการนี้จะเป็นการตรวจสอบความถูกต้องของโค้ดที่ผู้ใช้ป้อนให้กับระบบ เพื่อป้องกันไม่ให้เกิดข้อผิดพลาดในขั้นตอนถัดไป

(2) Extraction เป็นการค้นหาความสัมพันธ์ระหว่างส่วนย่อย ๆ ที่ได้มาจากขั้นตอนแรก

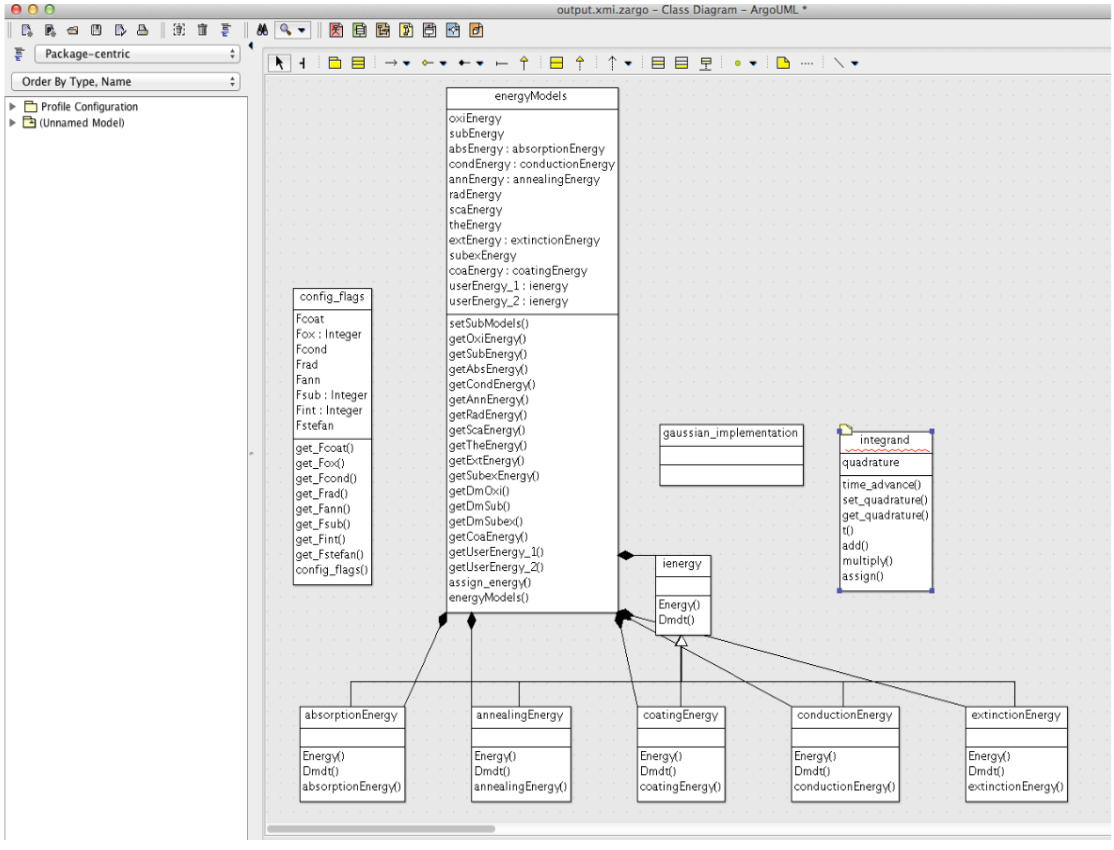
(3) Generating เป็นการรวบรวมเอาส่วนย่อย ๆ และความสัมพันธ์ที่ค้นหาได้จากขั้นตอนที่ 1 และ 2 มาสร้างเป็นเอกสารที่อยู่ในรูปแบบเอกซ์เอ็มไอ ซึ่งเอกสาร เอกซ์เอ็มไอนี้จะเก็บข้อมูลที่จำเป็นเพื่อใช้ในการแสดงยูเอ็มแอลคลาสไดอะแกรม

(4) Importing ในขั้นตอนนี้ ForUML จะนำเอาเอกสารเอกซ์เอ็มไอ ที่ได้ในข้อ 3 มาแปลงเข้าไปใน ArgoUML (Bogdan, et al., 2018) เพื่อแสดงยูเอ็มแอลคลาสไดอะแกรม



รูปที่ 2.8 ส่วนต่อประสานกราฟิกกับผู้ใช้ (Graphical User Interface) ของเครื่องมือ ForUML

จากรูปที่ 2.8 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ของเครื่องมือ ForUML ซึ่งผู้ใช้งานสามารถเลือกไฟล์ซอร์สโค้ดภาษาฟอร์แทรนที่ต้องการได้โดยกดปุ่ม Add และกดปุ่ม Remove ถ้าหากผู้ใช้งานต้องการลบไฟล์ที่เลือกไว้ที่ละไฟล์ หรือกดปุ่ม Reset ถ้าหากผู้ใช้งานต้องการลบไฟล์ทั้งหมด หลังจากได้เลือกไฟล์ตามที่ต้องการแล้วผู้ใช้งานสามารถกดปุ่ม Generate เพื่อสร้างไฟล์เอกซ์เอ็มไอขึ้นมา ซึ่งเป็นเอกสารที่ใช้สำหรับแสดงแผนภาพยูเอ็มแอลคลาสไดอะแกรม โดยระหว่างขั้นตอนนี้ผู้ใช้งานสามารถตรวจสอบสถานะและข้อผิดพลาดที่เกิดขึ้นได้ จากนั้นผู้ใช้งานสามารถกดปุ่ม View เพื่อแสดงยูเอ็มแอลคลาสไดอะแกรมขึ้นมาดังรูปที่ 2.9



รูปที่ 2.9 ตัวอย่างยูเอ็มแอลคลาสไดอะแกรมที่ได้จากเครื่องมือ ForUML

2.2 วรรณกรรมที่เกี่ยวข้อง

วรรณกรรมที่เกี่ยวข้องสำหรับการวิศวกรรมย้อนกลับจากซอร์สโค้ดไปเป็นยูเอ็มแอลไดอะแกรมมีดังต่อไปนี้

ไบรอันและคณะ (Briand, et al., 2003) ได้นำเสนอวิธีการและเครื่องมือสำหรับการวิศวกรรมย้อนกลับจากร่องรอย (Traces) ในการทำงานบางอย่างที่เกิดขึ้นในซอร์สโค้ด เช่น การเรียกใช้งานกันระหว่างคลาส ไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมโดยจะมีการแทรกร่องรอยในซอร์สโค้ดขณะโปรแกรมกำลังทำงาน (Run-time) เพื่อทำการสร้างโครงสร้างโมเดลของร่องรอย (Metamodel of Traces) ขึ้นมา ซึ่งจะใช้เอกซ์เอ็มไอเป็นตัวกลางในการแลกเปลี่ยนข้อมูล หลังจากนั้นจึงนำเอกซ์เอ็มไอที่

ได้ไปสร้างยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยงานวิจัยนี้ได้อธิบายถึงกรณีศึกษาของระบบตู้ ATM โดยมีการเปรียบเทียบยูเอ็มแอลซีเควนซ์ไดอะแกรมของผลลัพธ์ที่ได้จากเครื่องมือที่พัฒนาขึ้นมา กับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบจริงที่ได้มาจากการเอกสารการออกแบบ ซึ่งผลที่ได้แสดงให้เห็นว่า ยูเอ็มแอลซีเควนซ์ไดอะแกรมที่สร้างจากเครื่องมือที่พัฒนาขึ้นมานั้นมีความคล้ายคลึงกับ ยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบจริง แต่จะมีรายละเอียดบางอย่างเพิ่มเติมขึ้นมา ซึ่งนักออกแบบระบบนั้นไม่ได้มีการอธิบายไว้ในตอนออกแบบยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบจริง

จากงานวิจัยที่กล่าวมาข้างต้นมีความเกี่ยวข้องกับงานวิจัยที่กำลังทำการศึกษา คือ งานวิจัยนี้เป็นการนำเสนอวิธีการสำหรับวิศวกรรมย้อนกลับจากซอร์สโค้ดที่เป็นภาษาซีพลัสพลัส ไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ผู้วิจัยจึงเล็งเห็นถึงความเป็นไปได้ในการทำวิศวกรรมย้อนกลับจากซอร์สโค้ดภาษาฟอร์แทรน ไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งทั้ง 2 ภาษานั้นต่างเป็นการเขียนโปรแกรมเชิงวัตถุเหมือนกัน

จากงานวิจัยของไบรอันและคณะ ได้ให้ผลลัพธ์ที่คล้ายคลึงกับงานวิจัยของมานาร์และคณะ (Alalfi, et al., 2009) ซึ่งจะนำเสนอ วิธีการและเครื่องมือสำหรับการทำวิศวกรรมย้อนกลับจากเว็บแอปพลิเคชัน ไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยเครื่องมือดังกล่าวมีชื่อว่า PHP2XMI ซึ่งเป็นเครื่องมือที่มีวัตถุประสงค์เพื่อสร้างยูเอ็มแอลซีเควนซ์ไดอะแกรมจากเว็บแอปพลิเคชันที่พัฒนามาจากภาษา PHP โดยมีการใช้เอกซ์เอ็มไอเป็นตัวกลางในการแลกเปลี่ยนข้อมูล ซึ่งสามารถนำเอกซ์เอ็มไอที่ได้ไปใช้งานกับเครื่องมือตัวอื่นที่รองรับการแสดงยูเอ็มแอลไดอะแกรม เช่น Modelio และ UML Designer เพื่อแสดงยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมา

จากงานวิจัยที่กล่าวมาข้างต้นมีความเกี่ยวข้องกับงานวิจัยที่กำลังทำการศึกษา คือ งานวิจัยนี้ได้มีการนำเอกซ์เอ็มไอมาใช้สำหรับการแลกเปลี่ยนข้อมูลระหว่างซอร์สโค้ดและยูเอ็มแอลไดอะแกรม ซึ่งผู้วิจัยได้มีแนวคิดที่จะนำเอกซ์เอ็มไอมาใช้ในการแลกเปลี่ยนข้อมูลระหว่างซอร์สโค้ดภาษาฟอร์แทรน และยูเอ็มแอลซีเควนซ์ไดอะแกรม

จากงานวิจัยของคอร์ชูโนวาและคณะ (Korshunova, et al., 2006) ได้นำเสนอเครื่องมือที่มีชื่อว่า CPP2XML โดยเครื่องมือนี้สามารถช่วยแปลงซอร์สโค้ดที่เป็นภาษาซีพลัสพลัส ให้กลับมาเป็นยูเอ็มแอลคลาสไดอะแกรม ยูเอ็มแอลซีเควนซ์ไดอะแกรม และยูเอ็มแอลแอคทิวิตี้ไดอะแกรม ซึ่งจะแสดงในรูปแบบของเอกซ์เอ็มไอ งานวิจัยนี้ได้นำเสนอเทคโนโลยีที่นำมาใช้ในกระบวนการการพัฒนาที่ชื่อว่า Columbus ซึ่งเป็นเทคโนโลยีที่สามารถจีวิภาคโค้ดออกเป็นส่วนย่อย ๆ แล้วนำมาสร้างเป็น

โครงสร้าง AST (Abstract Syntax Tree) ซึ่งจะแสดงในรูปของเอกซ์เอ็มไอ และ CPPML (C++ Markup Language) จากนั้นจึงนำผลลัพธ์ที่ได้มาผ่านการวิเคราะห์เพื่อหาความสัมพันธ์แล้วนำมาสร้างเป็นยูเอ็มแอลไดอะแกรมที่อยู่ในรูปแบบของเอกซ์เอ็มไออีกครั้ง

จากงานวิจัยที่กล่าวมาข้างต้นมีความเกี่ยวข้องกับงานวิจัยที่กำลังทำการศึกษา คือ งานวิจัยนี้ได้มีการนำเสนอเทคโนโลยีที่น่าสนใจนั้น คือ Columbus โดยผู้วิจัยมีความเห็นว่าเทคโนโลยีดังกล่าวสามารถเป็นแนวทางในการค้นหาเทคโนโลยีที่มีความคล้ายคลึงกันเพื่อนำมาปรับใช้กับภาษาฟอร์แทรน ซึ่งจะเป็นผลดีต่อการพิจารณาเลือกใช้เทคโนโลยีที่มีความเหมาะสมกับงานวิจัย

จากงานวิจัยของอาบิลิโอและคณะ (Parada, et al., 2011) ได้นำเสนอวิธีการสำหรับสร้างซอร์สโค้ดภาษาจาวาขึ้นมาจากยูเอ็มแอลไดอะแกรม โดยส่วนที่เป็นโครงสร้าง (Structural) ของซอร์สโค้ดนั้นถูกสร้างขึ้นมาจากยูเอ็มแอลคลาสไดอะแกรม ซึ่งอธิบายถึง แอตทริบิวท์ เมท็อด และคอนสตรักเตอร์ (Constructors) อีกทั้งยังมีการพิจารณาถึงความสัมพันธ์ระหว่างคลาส (Classes) หรืออินเตอร์เฟซ (Interface) และส่วนที่เป็นพฤติกรรม (Behavioral) ของซอร์สโค้ดนั้นถูกสร้างขึ้นมาจากยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งจะอธิบายถึงการเรียกใช้งานของเมท็อด ซึ่งประกอบด้วย อาร์กิวเมนต์และการส่งค่ากลับ (Return) รวมไปถึงการวนซ้ำ (Loops) และเงื่อนไข (Conditions) งานวิจัยนี้แสดงให้เห็นถึงการสร้างซอร์สโค้ดขึ้นมาจากยูเอ็มแอลคลาสไดอะแกรม และยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยมีเอกซ์เอ็มไอเป็นตัวกลางในการแลกเปลี่ยนข้อมูล

งานวิจัยนี้มีความเกี่ยวข้องกับงานวิจัยที่กำลังทำการศึกษา คือ งานวิจัยนี้แสดงให้เห็นถึงความเป็นไปได้ในการแปลงยูเอ็มแอลคลาสไดอะแกรม และยูเอ็มแอลซีเควนซ์ไดอะแกรมให้กลายเป็นซอร์สโค้ดภาษาจาวา ซึ่งผู้วิจัยสามารถนำมาใช้เป็นแนวทางในการศึกษาการวิศวกรรมย้อนกลับจากซอร์สโค้ดภาษาฟอร์แทรนมาเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมได้

จากงานวิจัยของพจนาและคณะ (Sawprakhon, et al., 2014) ได้นำเสนอวิธีการสำหรับสร้างยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมาจากยูสเคสไดอะแกรม และคลาสไดอะแกรม โดยได้มีการใช้ ATL (ATLAS Transformation Language) ซึ่งเป็นโมเดลที่ไว้สำหรับการแปลงรูปแบบของโมเดลหนึ่งไปเป็นอีกโมเดลหนึ่ง งานวิจัยนี้จะนำรายละเอียดของยูสเคส (Use Case Description) และคลาสไดอะแกรมที่สร้างมาจากเครื่องมือ Visual Paradigm มาสร้างเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมซึ่งจะแสดงอยู่ในรูปแบบของเอกสารเอกซ์เอ็มไอ โดยมีการสร้างกฎการแปลงขึ้นมาเพื่อใช้สำหรับเชื่อมโยง

ระหว่างรายละเอียดของยูสเคสและคลาสไดอะแกรม ซึ่งผลลัพธ์ที่ได้จะออกมาในรูปแบบของเอกสารเอกซ์เอ็มไอ

งานวิจัยนี้มีความเกี่ยวข้องกับงานวิจัยที่กำลังทำการศึกษา คือ งานวิจัยนี้ได้นำเสนอกฎที่ใช้ในการแปลงรายละเอียดของยูสเคส และคลาสไดอะแกรม ไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งผู้วิจัยสามารถนำมาประยุกต์ใช้สำหรับการแปลงซอร์สโค้ดภาษาฟอร์แทรนไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมได้

จากงานวิจัยของพจนาและคณะ ได้ให้ผลลัพธ์ที่คล้ายคลึงกับงานวิจัยของแอลคาร์แมลและคณะ (Merah, et al., 2014) ซึ่งได้นำเสนอวิธีการในการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมไปเป็นเพรททิเน็ต (Petri Nets) ซึ่งเป็นเครื่องมือสร้างสมมุติฐานทางคณิตศาสตร์ ใช้ในการวิเคราะห์และจำลองระบบ งานวิจัยได้มีการสร้างกฎการแปลงขึ้นมา โดยจะใช้ ATL เป็นเครื่องมือในการแปลงรูปแบบของยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นเพรททิเน็ต

งานวิจัยนี้มีความเกี่ยวข้องกับงานวิจัยที่กำลังทำการศึกษา คือ งานวิจัยนี้ได้นำเสนอกฎที่ใช้สำหรับการแปลงขึ้นมาเช่นเดียวกับงานวิจัยของของพจนาและคณะ แต่จะทำการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมไปเป็นเพรททิเน็ต ซึ่งผู้วิจัยสามารถนำมาประยุกต์ใช้สำหรับการแปลงซอร์สโค้ดภาษาฟอร์แทรนไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมได้

ตารางที่ 2.5 เปรียบเทียบงานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้อง	ออกแบบกฎการแปลง	เทคนิคการแปลงโมเดล (ATLAS)	ออกแบบกระบวนการแปลง	ประเภทของไดอะแกรมสำหรับการแปลงที่เกี่ยวข้อง					ประเภทของภาษาโปรแกรมสำหรับการแปลงที่เกี่ยวข้อง				การแลกเปลี่ยนข้อมูลโดย XMI
				Class diagram	Sequence diagram	Usecase diagram	Activity diagram	Petri Nets diagram	Fortran	Java	C++	PHP	
งานวิจัยที่นำเสนอ	✓	✓	✓	-	✓	-		-	✓	-	-	-	✓
Merah, E., et al. (2014)	✓	✓	-	-	✓	-	-	✓	-	-	-	-	-
Sawprakhon, P., and Limpiyakorn, Y., (2014)	✓	✓	-	✓	✓	✓	-	-	-	-	-	-	✓
Briand, L. C., et al. (2003)	-	-	✓		✓	-	-	-	-	-	✓	-	✓
Alalfi, M. H., et al. (2009)	-	-	✓	-	✓	-	-	-	-	-	-	✓	✓
Korshunova, E., et al. (2006)	-	-	✓	✓	✓	-	✓	-	-	-	✓	-	✓
Parada, A. G., et al. (2011)	-	-	✓	✓	✓	-	-	-	-	✓	-	-	✓

จากตารางที่ 2.5 แสดงการเปรียบเทียบงานวิจัยระหว่างงานวิจัยที่ผู้วิจัยได้นำเสนอและงานวิจัยที่เกี่ยวข้อง โดยผู้วิจัยได้ทำการแบ่งกลุ่มสำหรับการเปรียบเทียบไว้เป็น 2 กลุ่มดังนี้

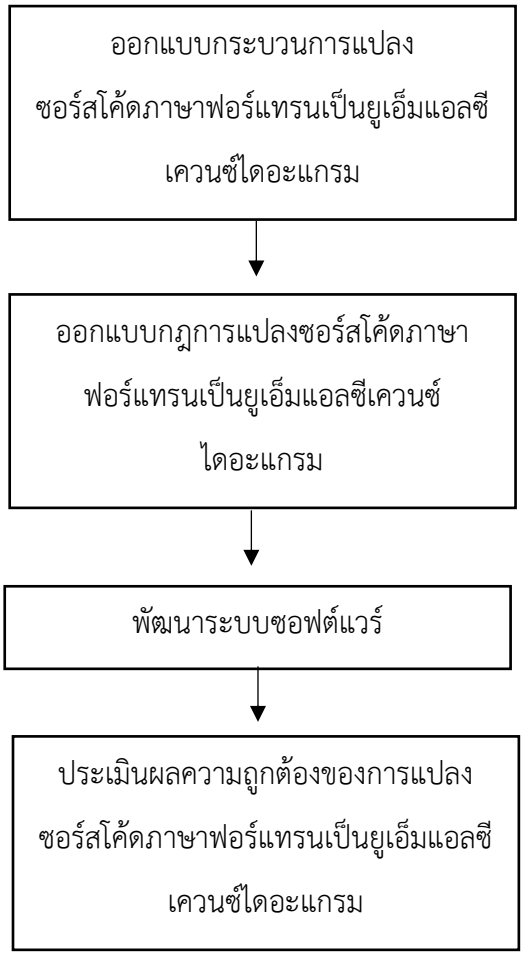
(1) งานวิจัยที่เกี่ยวข้องสำหรับการออกแบบกฎการแปลง โดยงานวิจัยของ (Merah, et al., 2014) และ (Sawprakhon, et al., 2014) เป็นการออกแบบกฎการแปลงขึ้นมา ซึ่งจะเกี่ยวข้องกับการแปลงระหว่างซอร์สโค้ดและไดอะแกรม เมื่อพิจารณางานวิจัยที่กล่าวมข้างต้นเปรียบเทียบกับงานวิจัยที่ผู้วิจัยได้นำเสนอ พบว่างานวิจัยที่กล่าวมาข้างต้นมีการใช้เทคนิคการแปลงโมเดลด้วย ATLAS ซึ่งผู้วิจัยได้พิจารณานำเทคนิคดังกล่าวมาทำการสร้างกฎการแปลงของซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมา

(2) งานวิจัยที่เกี่ยวข้องสำหรับการออกแบบกระบวนการแปลง โดยงานวิจัยของ (Briand, et al., 2003), (Alalfi, et al., 2009), (Korshunova, et al., 2006) และ (Parada, et al., 2011) เป็นการออกแบบกระบวนการแปลงขึ้นมา ซึ่งจะเกี่ยวข้องกับการแปลงจากซอร์สโค้ดไปเป็นไดอะแกรม หรือจากไดอะแกรมไปเป็นซอร์สโค้ด โดยมีการใช้เอกซ์เอ็มไอเป็นตัวกลางในการแลกเปลี่ยนข้อมูลระหว่างกัน เมื่อพิจารณางานวิจัยที่กล่าวมาข้างต้นเปรียบเทียบกับงานวิจัยที่ผู้วิจัยได้นำเสนอ พบว่างานวิจัยที่กล่าวมาข้างต้นมีความเกี่ยวข้องกับภาษาเชิงวัตถุ ซึ่งงานของผู้วิจัยจะมีการแปลงภาษาฟอร์แทรนที่เป็นภาษาเชิงวัตถุไปเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ดังนั้นจึงสามารถนำงานวิจัยดังกล่าวมาประยุกต์ใช้เพื่อสร้างกระบวนการแปลงขึ้นมา

บทที่ 3

วิธีดำเนินการวิจัย

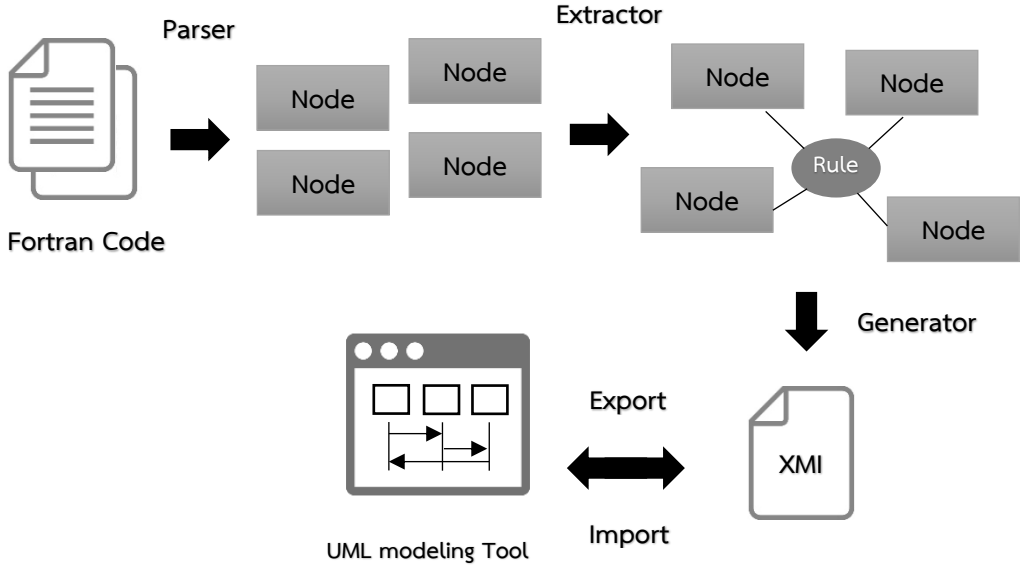
ในบทนี้จะกล่าวถึงขั้นตอนการพัฒนาศักยภาพของซอฟต์แวร์ ForUML สำหรับการดำเนินงานมีขั้นตอน ดังรูปที่ 3.1 โดยมีรายละเอียดการดำเนินงานแต่ละขั้นตอนดังนี้



รูปที่ 3.1 ขั้นตอนการดำเนินงานวิจัย

3.1 ออกแบบกระบวนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม

ในส่วนนี้จะเป็นการอธิบายถึงกระบวนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมซึ่งมีขั้นตอนทั้งหมด 4 ขั้นตอน ดังรูปที่ 3.2



รูปที่ 3.2 กระบวนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม

จากรูปที่ 3.2 แสดงขั้นตอนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมประกอบด้วย 4 ขั้นตอน ดังนี้

(1) ขั้นตอนของการวจีวิภาค (Parser) ซอร์สโค้ดออกเป็นส่วนย่อย ๆ โดยจะมีการใช้ไลบรารี OFP (Open Fortran Parser) สำหรับการวจีวิภาคซอร์สโค้ด ซึ่งในกระบวนการนี้จะอาศัยไฟล์ไวย์กรณ์ และวากยสัมพันธ์ของภาษาฟอร์แทรน ที่ได้ถูกพัฒนาไว้แล้วในไลบรารี OFP โดยจะนำไปใช้กับเครื่องมือ ANTLR (Terence, 2017) ซึ่งสามารถวจีวิภาคซอร์สโค้ดให้อยู่ในรูปของซอร์สโค้ดที่เป็นส่วนย่อย ๆ เพื่อนำไปใช้หาความสัมพันธ์ในขั้นตอนถัดไป

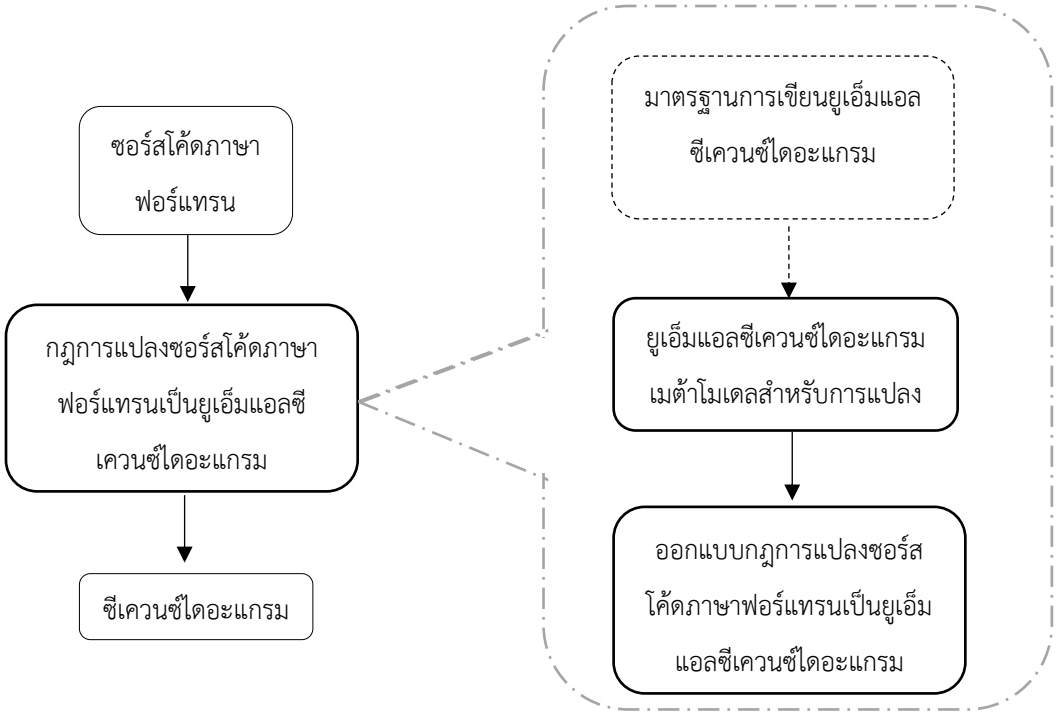
(2) ขั้นตอนของการหาความสัมพันธ์ (Extractor) ระหว่างส่วนย่อย ๆ โดยจะอาศัยกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้ออกแบบไว้ และหลังจากนั้นจึงหาความสัมพันธ์ของซอร์สโค้ดที่ได้แบ่งออกเป็นส่วนย่อย ๆ ดังที่แสดงในขั้นตอนที่ 1 เป็นไปตามกฎแต่ละข้อ

(3) ขั้นตอนของการนำความสัมพันธ์ที่ได้มาสร้าง (Generator) เอกสารเอกซ์เอ็มไอ โดยเอกสารเอกซ์เอ็มไอดังกล่าวจะถูกออกแบบให้อยู่ในรูปของ Document Type Definition (DTD) ซึ่งเป็นโครงสร้างข้อมูล (Schema) ที่นิยมใช้ในการสร้างเอกสารเอกซ์เอ็มไอ โดยจะมีการกำหนดรูปแบบการแปลงระหว่างความสัมพันธ์ที่ได้ และเอกสารเอกซ์เอ็มไอเพื่อใช้ในการแสดงยูเอ็มแอลซีเควนซ์ไดอะแกรม

(4) ขั้นตอนของการนำเข้า (Import) เอกสารเอกซ์เอ็มไอสำหรับซอฟต์แวร์มอเดลลิโอ (Modelio) ซึ่งเป็นเครื่องมือที่ใช้ในการสร้างยูเอ็มแอลไดอะแกรม ซอฟต์แวร์นี้จะมีการเปิดเผยซอร์สโค้ด ดังนั้นนักพัฒนาซอฟต์แวร์จึงสามารถแก้ไขหรือดัดแปลง โดยปัจจุบันซอฟต์แวร์มอเดลลิโอ เป็นซอฟต์แวร์ที่นักพัฒนาให้ความสนใจ และมีการพัฒนาอยู่ต่อเนื่อง จึงทำให้มีแหล่งศึกษาหาข้อมูลเพื่อปรับแต่งการทำงานของซอฟต์แวร์ให้เป็นไปตามที่ผู้วิจัยต้องการ และถ้าหากผู้ใช้งานต้องการแสดงยูเอ็มแอลไดอะแกรมผ่านเครื่องมือชนิดอื่น ผู้ใช้งานสามารถนำออก (Export) เอกสารเอกซ์เอ็มไอที่ได้จากเครื่องมือ ForUML ไปนำเข้าเครื่องมือชนิดนั้นได้

3.2 ออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม

ในส่วนนี้จะเป็นการอธิบายถึงแนวคิดของการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งเป็นแนวคิดที่ได้มาจากการนำมาตรฐานของยูเอ็มแอลซีเควนซ์ไดอะแกรมตามเอกสารรายละเอียดของยูเอ็มแอล (UML Specification) (OMG, 2017) และกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมที่มาจากวรรณกรรมที่เกี่ยวข้อง(Li, et al., 2014; Merah, 2014; Sawprakhon and Limpiyakorn, 2014) มาประยุกต์ใช้เพื่อสร้างกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมาใหม่ หลังจากนั้นจะเป็นการอธิบายถึงกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม



รูปที่ 3.3 แนวคิดการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม

จากรูปที่ 3.3 แสดงแนวคิดการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยจะเริ่มต้นจากการออกแบบยูเอ็มแอลซีเควนซ์ไดอะแกรมเมต้าโมเดลของกฎแต่ละข้อ ซึ่งเมต้าโมเดลนี้จะแสดงอยู่ในรูปของคลาสไดอะแกรม โดยจะเป็นโมเดลที่มีไว้สำหรับแสดงรูปแบบของซอร์สโค้ดภาษาฟอร์แทรน และยูเอ็มแอลซีเควนซ์ไดอะแกรม เพื่อใช้สำหรับแสดงความหมายของกฎต่าง ๆ ในการแปลงเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม และเพื่อใช้เป็นมาตรฐานกลางในการแลกเปลี่ยนข้อมูลระหว่างกัน

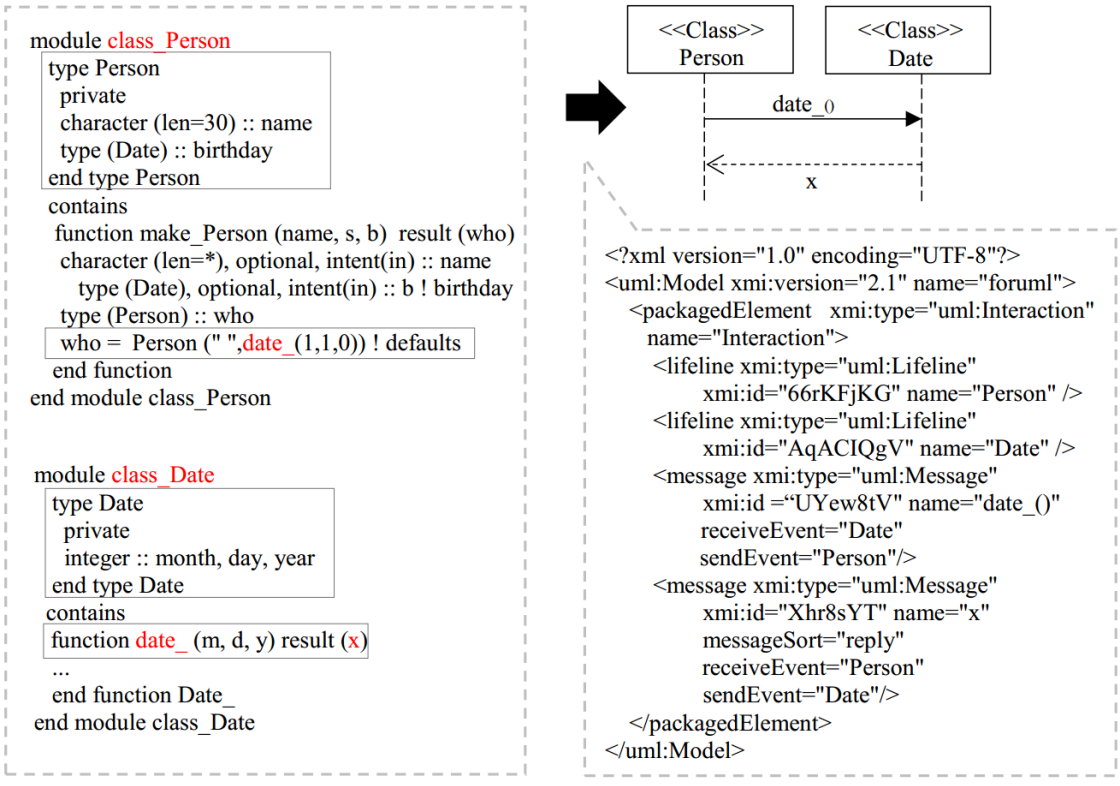
การออกแบบยูเอ็มแอลซีเควนซ์ไดอะแกรมเมต้าโมเดลจะพิจารณาจากความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและเอกสารเอกซ์เอ็มไอในส่วนของการสร้างคลาสไดอะแกรมซึ่งได้มาจากซอฟต์แวร์ ForUML ในงานวิจัยก่อนหน้านี้ซึ่งผู้วิจัยสามารถนำข้อมูลบางส่วนที่มีอยู่แล้วมาประยุกต์ใช้งานได้ ยกตัวอย่างเช่น การนำข้อมูลชื่อของคลาสแต่ละคลาสมาสร้างเป็นเส้นแนวตั้งหรือที่เรียกว่าไลฟ์ไลน์ จากนั้นจึงนำไปปรับใช้กับบางส่วนของซอร์สโค้ดภาษาฟอร์แทรนที่แสดงถึงการเรียกใช้งานกันระหว่างคลาส เพื่อนำมาสร้างสารที่ใช้ส่งระหว่างไลฟ์ไลน์ โดยที่ซอร์สโค้ดเหล่านี้จะเป็นข้อมูลที่เกี่ยวข้องกับพฤติกรรม เช่น การส่งค่าไป การส่งค่ากลับ การวนซ้ำ และเงื่อนไข

ตารางที่ 3.1 กฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลคลาสไดอะแกรมซึ่งอยู่ในรูปแบบของเอกซ์เอ็มไอ (Rouson, et al., 2010)

Fortran	XMI Elements
Derived Type	UML: Class
Type-bound Procedure	UML: Operation
Dummy Argument	UML: Parameter
Component	UML: Attribute
Intrinsic type	UML: DataType
Parent Type	UML: Generalization.parent
Extended Type	UML: Generalization.child
Composite	UML: Association (The aggregation property as 'composite')

สำหรับเอกสารเอกซ์เอ็มไอในส่วนของการสร้างคลาสไดอะแกรมที่ได้มาจากซอฟต์แวร์ ForUML จะได้มาจากกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนแสดงดังตารางที่ 3.1 ซึ่งผู้วิจัยได้เลือกในส่วนของ Derived Type, Type-bound Procedure, Dummy Argument และ Component มาประยุกต์ใช้ในการสร้างกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมา

ในการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมจะมีการศึกษารายละเอียดของเอกสารเอกซ์เอ็มไอในส่วนของการสร้างซีเควนซ์ไดอะแกรม ซึ่งเป็นรูปแบบมาตรฐานของ Object Management Group (OMG) ที่แสดงให้เห็นถึงรายละเอียดข้อมูลของยูเอ็มแอลโมเดล โดยจะเป็นรูปแบบเพื่อใช้สำหรับอธิบายความหมายของกฎต่าง ๆ ในการแปลงเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม และเพื่อใช้เป็นมาตรฐานกลางในการแลกเปลี่ยนข้อมูลระหว่างกัน สำหรับตัวอย่างเอกสารเอกซ์เอ็มไอที่มีข้อมูลของยูเอ็มแอลซีเควนซ์ไดอะแกรมซึ่งจะแสดงดังรูปที่ 3.4 ซึ่งประกอบด้วย

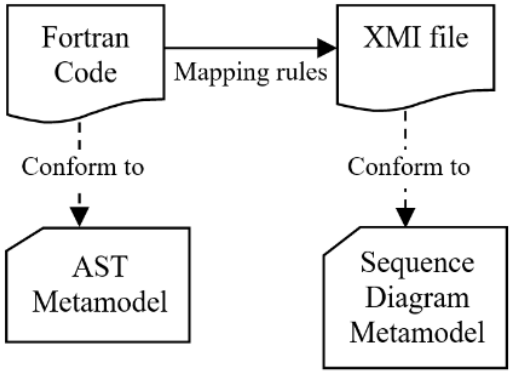


รูปที่ 3.4 ตัวอย่างของเอกสารเอกซ์เอ็มไอสำหรับ ยูเอ็มแอลซีเควนซ์ไดอะแกรมของโปรแกรมในภาษาฟอร์แทรน

(1) xmi:type="uml:Lifeline" จะเป็นรูปแบบที่ไว้กำหนดรายละเอียดของไลฟ์ไลน์ โดยมี xmi:id="66rKFjKG" ที่เป็นหมายเลขกำกับของไลฟ์ไลน์ และ name="Person" ที่แสดงชื่อของไลฟ์ไลน์

(2) xmi:type="uml:Message" จะเป็นรูปแบบที่ไว้กำหนดรายละเอียดของสาร โดยมี xmi:id="Xhr8sYT" เป็นหมายเลขกำกับของสาร messageSort="reply" แสดงประเภทของสาร name="Person" แสดงชื่อของสาร receiveEvent="Person" แสดงไลฟ์ไลน์ที่รับสาร และ sendEvent= "Date" แสดงไลฟ์ไลน์ที่ส่งสาร

สำหรับการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมนั้นจะมีขั้นตอนหลัก คือ การหาความสัมพันธ์ระหว่าง AST เมต้าโมเดลของภาษาฟอร์แทรนและเอกสารเอกซ์เอ็มไอ โดยเมต้าโมเดลของทั้งสองจะเป็นโมเดลที่ใช้เป็นตัวแทนของโมเดลหลักแสดงดังรูปที่ 3.5



รูปที่ 3.5 กระบวนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม

จากรูปที่ 3.5 เป็นกระบวนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งจะประกอบด้วย AST Metamodel ซึ่งเป็นเมต้าโมเดลที่มีความสอดคล้องกับซอร์สโค้ดภาษาฟอร์แทรน และ Sequence Diagram Metamodel ซึ่งเป็นเมต้าโมเดลที่มีความสอดคล้องกับเอกสารเอกซ์เอ็มไอ โดยที่ข้อมูลของ AST Metamodel จะได้มาจากการแยกส่วนประกอบต่าง ๆ ของซอร์สโค้ดผ่านไลบรารี OFP จากนั้นจะนำส่วนประกอบต่าง ๆ ของซอร์สโค้ดมากำหนดความสัมพันธ์ขึ้นมาโดยจะมีการสร้างกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมเพื่อกำหนดมาตรฐานในการสร้างเอกสารเอกซ์เอ็มไอ โดยที่ข้อมูลของ Sequence Diagram Metamodel จะเป็นไปตามข้อกำหนดของยูเอ็มแอลเวอร์ชัน 2.1 ซึ่งเป็นมาตรฐานที่ถูกกำหนดโดย OMG

หลังจากการศึกษามาตรฐานของยูเอ็มแอลซีเควนซ์ไดอะแกรมตามเอกสารรายละเอียดของยูเอ็มแอล (OMG, 2017) และกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้มาจากการศึกษาวรรณกรรมที่เกี่ยวข้อง (Li, et al., 2014; Merah, 2014; Sawprakhon and Limpiyakorn, 2014) ผู้วิจัยได้กำหนดกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมไว้ดังนี้

1) กฎสำหรับการสร้างไลฟ์ไลน์
2) กฎสำหรับการสร้างสารระหว่างไลฟ์ไลน์
3) กฎสำหรับกำหนดการรับและส่งของสารที่เกิดขึ้น
4) กฎสำหรับกำหนดการเริ่มต้นและสิ้นสุดการดำเนินงาน
5) กฎสำหรับกำหนดการดำเนินงานของสารบนไลฟ์ไลน์
6) กฎสำหรับการสร้างกรอบของแผนภาพ

3.3 พัฒนาระบบซอฟต์แวร์

หลังจากทำการออกแบบกระบวนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม และทำการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมจะทำให้ทราบถึงกระบวนการทำงานของระบบ โดยจะมีการพัฒนาในรูปแบบของเดสก์ท็อปแอปพลิเคชัน (Desktop Application) ซึ่งจะใช้ภาษาจาวาในการพัฒนา เนื่องจากมีไลบรารีที่รองรับสำหรับการพัฒนาแอปพลิเคชัน เช่น Open Fortran Parser (OFP) และสามารถเปิดใช้งานแอปพลิเคชันที่สร้างขึ้นมาได้ทุกแพลตฟอร์ม สำหรับเครื่องมือที่ใช้ในการพัฒนา ได้แก่ Netbeans ซึ่งเป็นเครื่องมือที่ช่วยในการเขียนโปรแกรมภาษาจาวา และมอเดลลิโอซึ่งเป็นเครื่องมือที่จะใช้สำหรับแสดงยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมาจากเอกสารเอกซ์เอ็มไอ โดยมีรายละเอียดของฮาร์ดแวร์ และซอฟต์แวร์ที่ใช้ในการพัฒนาระบบดังนี้

รายละเอียดของฮาร์ดแวร์ที่ใช้ในการพัฒนาระบบ

(1) CPU: Intel Core i7-6700HQ

(2) Graphics: Intel HD Graphics 530 + Nvidia GeForce GTX 950M

(3) RAM: 8GB DDR3L SDRAM (2 x 4GB)

(4) Storage: 1TB 5400 rpm SATA

รายละเอียดของระบบปฏิบัติการ และซอฟต์แวร์ที่ใช้ในการพัฒนาระบบ

(1) Microsoft Windows 10 Pro Professional 64 bit

(2) NetBeans IDE เวอร์ชัน 8.2

(3) Java Development Kit เวอร์ชัน 1.8

(4) Modelio เวอร์ชัน 3.7

โดยหลังจากที่พัฒนาระบบเสร็จแล้ว ผู้วิจัยจะนำระบบที่พัฒนาขึ้นไปทำการประเมินผลความถูกต้องซึ่งจะได้อธิบายในลำดับถัดไป

3.4 ประเมินผลความถูกต้องของการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม

การประเมินผลความถูกต้องของผลลัพธ์ที่ได้มาจากการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม จะประเมินจากการเปรียบเทียบผลลัพธ์ที่ได้ระหว่างเครื่องมือ ForUML และซอร์สโค้ดของระบบซึ่งจะพิจารณาด้วยผู้วิจัยเอง โดยขั้นตอนในการพิจารณาจะเริ่มต้นจากการตรวจสอบจำนวนส่วนประกอบของซอร์สโค้ดภายในระบบทั้งหมดที่ผู้วิจัยกำหนดไว้ จากนั้นจะทำการตรวจสอบจำนวนข้อมูลในซีเควนซ์ไดอะแกรมที่มีความสอดคล้องกับซอร์สโค้ดดังกล่าว หลังจากนั้นจะทำการเปรียบเทียบจำนวนส่วนประกอบของซอร์สโค้ดที่กำหนดไว้และจำนวนข้อมูลในยูเอ็มแอลซีเควนซ์ไดอะแกรมที่มีความสอดคล้องกัน ซึ่งผู้วิจัยได้มีการเขียนโปรแกรมขึ้นมานับจำนวนส่วนประกอบของซอร์สโค้ดเพื่อลดความผิดพลาดที่อาจเกิดขึ้นจากการนับด้วยตัวเอง โดยจำนวนส่วนประกอบของซอร์สโค้ดภายในระบบที่ได้กำหนดไว้มีดังนี้

(1) จำนวนคลาสทั้งหมดที่มีการเรียกใช้งานภายในระบบ ซึ่งจะต้องเป็นคลาสที่สร้างขึ้นอยู่ภายใต้แพ็กเกจซึ่งในภาษาฟอร์แทรนจะใช้คำว่ามอดูล (Module) หรือเป็นโปรแกรมหลัก (Program Main)

(2) จำนวนเมท็อดทั้งหมดที่มีการเรียกใช้งานจากคลาสอื่นเกิดขึ้นภายใน ซึ่งจะครอบคลุมไปถึงฟังก์ชันและซับรูทีน

(3) จำนวนการเรียกใช้งานของฟังก์ชันทั้งหมดที่เกิดขึ้นภายในระบบ

(4) จำนวนการเรียกใช้งานของซับรูทีนทั้งหมดที่เกิดขึ้นภายในระบบ

(5) จำนวน Statements ทั้งหมดที่มีการเรียกใช้งานฟังก์ชันหรือซับรูทีน ซึ่งเป็น Statements ในส่วนของเงื่อนไข ทางเลือกที่มีหลายเงื่อนไข และการทำซ้ำที่มีการเรียกใช้งานเกิดขึ้นภายใน

จากจำนวนส่วนประกอบของซอร์สโค้ดภายในระบบที่นำมาทดสอบข้างต้นนั้นเป็นข้อมูลในส่วนของซอร์สโค้ดที่สำคัญที่เกี่ยวข้องกับการสร้างยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งจะสอดคล้องกับกฎที่ได้ทำการออกแบบไว้ คือ 1) จำนวนคลาสทั้งหมดจะสอดคล้องกับกฎสำหรับการสร้างไลฟ์ไลน์ และกฎสำหรับกำหนดการดำเนินงานของสารบนไลฟ์ไลน์ 2) จำนวนเมท็อดทั้งหมดจะสอดคล้องกับกฎสำหรับการสร้างสารระหว่างไลฟ์ไลน์ 3) จำนวนการเรียกใช้งานของฟังก์ชันทั้งหมด และจำนวนการเรียกใช้งาน

ของซับรูทีนทั้งหมดจะสอดคล้องกับกฎสำหรับกำหนดการรับและส่งของสารที่เกิดขึ้น และกฎสำหรับกำหนดการเริ่มต้นและสิ้นสุดการดำเนินงาน 4) จำนวน Statements ทั้งหมดจะสอดคล้องกับกฎสำหรับการสร้างกรอบของแผนภาพ ซึ่งถ้าหากจำนวนข้อมูลจากส่วนประกอบของซอร์สโค้ดภายในระบบที่นำมาทดสอบตรงกับจำนวนข้อมูลของยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้มาจากเครื่องมือ จึงพิจารณาได้ว่าซอฟต์แวร์มีความถูกต้อง สำหรับซอฟต์แวร์ที่นำมาใช้ในการทดสอบมีดังนี้

(1) ForTrilinos (ForTrilinos, 2017) เป็นซอฟต์แวร์โอเพนซอร์สที่มีอินเตอร์เฟซเขียนด้วยภาษาฟอร์แทรนเพื่อเรียกใช้ Trilinos ที่เป็นซอฟต์แวร์หลักซึ่งประกอบด้วยชุดของไลบรารีที่ใช้ในการแก้ปัญหาของแอพพลิเคชันในด้านวิทยาศาสตร์และวิศวกรรมศาสตร์

(2) PSBLAS (PSBLAS, 2017) เป็นซอฟต์แวร์โอเพนซอร์สที่ใช้ในการแก้ไขปัญหา Parallel sparse matrix ซึ่งพัฒนาโดยใช้ภาษาฟอร์แทรนเวอร์ชัน 2003

(3) MLD2P4 (MLD2P4, 2017) เป็นซอฟต์แวร์โอเพนซอร์สที่ใช้ในการแก้ไขปัญหาระบบเชิงเส้น (Linear System) ซึ่งพัฒนาโดยใช้ภาษาฟอร์แทรนเวอร์ชัน 2003

สำหรับเหตุผลในการพิจารณาเลือกซอฟต์แวร์มาทดสอบนั้นจะพิจารณาจากซอฟต์แวร์ที่มีการพัฒนาขึ้นมาจากภาษาฟอร์แทรนที่เป็นภาษาเชิงวัตถุ และมีการพัฒนาขึ้นมาใช้งานในด้านของวิทยาศาสตร์และวิศวกรรมศาสตร์ และเป็นซอฟต์แวร์ที่มีการใช้งานอยู่จริง

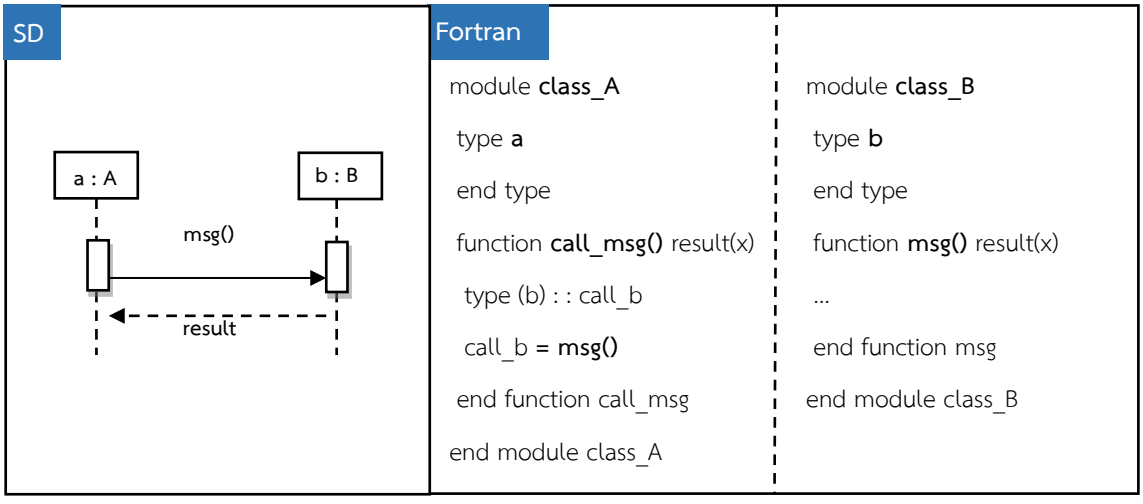
บทที่ 4

การออกแบบและพัฒนาระบบ

ในบทนี้จะเป็นการอธิบายผลการดำเนินงานวิจัย ซึ่งผู้วิจัยสามารถแบ่งผลการดำเนินงานวิจัยออกเป็น 2 ส่วน คือ 1) ส่วนของการออกแบบระบบ และ 2) ส่วนของการพัฒนาระบบ โดยมีรายละเอียดจากผลการดำเนินงานวิจัยดังนี้

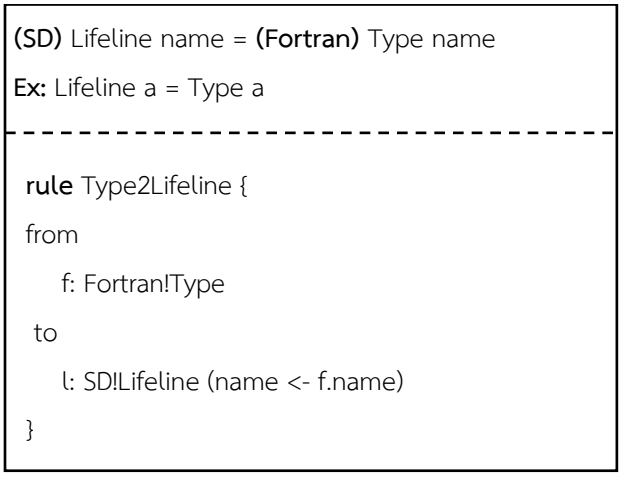
4.1 การออกแบบระบบ

การออกแบบระบบผู้วิจัยได้ทำการออกแบบกระบวนการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมา โดยส่วนของขั้นตอนการหาความสัมพันธ์ในกระบวนการแปลงนั้นผู้วิจัยได้ทำการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม เพื่อให้ง่ายต่อการทำความเข้าใจ ผู้วิจัยจะใช้ รูปที่ 4.1 เป็นการอธิบายกฎการแปลงที่สร้างขึ้นมา โดยการสร้างกฎการแปลงนั้นผู้วิจัยได้เลือกใช้ภาษา Atlas Transformation Language (ATL) ซึ่งเป็นภาษาที่นิยมใช้สำหรับการแปลงแบบจำลอง โดยผู้วิจัยได้นำส่วนที่สำคัญของกฎการแปลงที่ได้ทำการออกแบบมาแสดงดังนี้



รูปที่ 4.1 ตัวอย่างใช้อธิบายกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม

(1) กฎสำหรับการสร้างไลฟ์ไลน์ เป็นกฎที่ใช้ในการเชื่อมโยงระหว่างไลฟ์ไลน์ของยูเอ็มแอลซีเควนซ์ไดอะแกรมและชื่อคลาสของซอร์สโค้ดภาษาฟอร์แทรน (แสดงดังรูปที่ 4.2)



รูปที่ 4.2 กฎสำหรับการสร้างไลฟ์ไลน์

จากรูปที่ 4.2 เป็นการแปลงคลาส Type ของเมต้าโมเดล AST จากซอร์สโค้ดภาษาฟอร์แทรน ไปสู่ Lifeline Node ของเมต้าโมเดลซีเควนซ์ไดอะแกรม โดยมีแอตทริบิวท์ name เป็นตัวกำหนดชื่อของไลฟ์ไลน์ให้ตรงกับชื่อของคลาสในซอร์สโค้ดภาษาฟอร์แทรน

(2) กฎสำหรับการสร้างสารระหว่างไลฟ์ไลน์ เป็นกฎที่ใช้ในการเชื่อมโยงระหว่างสารของยูเอ็มแอลซีเควนซ์ไดอะแกรมและชื่อเมท็อดของภาษาฟอร์แทรน (แสดงดังรูปที่ 4.3)

```
(SD) Message name = (Fortran) Function/ Subroutine name
Ex: Message msg = Function msg
-----------------------------------------------------
 rule Procedures2Message {
 from
     f: Fortran!Procedures
 to
     m1: SD!SynchronousMessage(name <- f.FunctionName,
               messageSort  <- f. SynchronousMsg),
     m2: SD!SynchronousMessage(name <- f.SubroutineName,
               messageSort  <- f. SynchronousMsg),
     m3: SD!AsynchronousMessage(name <- f.SubroutineName,
               messageSort  <- f. AsynchronousMsg),
     m4: SD!CreateMessage(name <- f.ObjectName,
               messageSort  <- f. CreateMsg),
     m5: SD!ReplyMessage(name <- f. FunctionReplyName,
               messageSort  <- f. ReplyMsg),
     m6: SD!ReplyMessage(name <- f. SubroutineReplyName,
               messageSort  <- f. ReplyMsg)
 }
```

รูปที่ 4.3 กฎสำหรับการสร้างสารระหว่างไลฟ์ไลน์ของสารแบบ Synchronous Asynchronous Create และ Reply

จากรูปที่ 4.3 เป็นการแปลงคลาส Procedures ของเมต้าโมเดล AST จากซอร์สโค้ดภาษาฟอร์แทรน ไปสู่ Message Node ของเมต้าโมเดลซีเควนซ์ไดอะแกรม โดยสามารถแบ่งสารออกเป็น 4 แบบ คือ 1) Synchronous 2) Asynchronous 3) Create และ 4) Reply ซึ่งจะมีแอตทริบิวท์ name

เป็นตัวกำหนดชื่อของสารให้ตรงกับชื่อของเมท็อดในซอร์สโค้ดภาษาฟอร์แทรน และจะมีแอตทริบิวท์ messageSort เป็นตัวกำหนดชนิดของสารแต่ละชนิด

(3) กฎสำหรับกำหนดการรับและส่งของสารที่เกิดขึ้น เป็นกฎที่ใช้ในการกำหนดการรับสารของไลฟ์ไลน์ และการส่งสารของไลฟ์ไลน์ (แสดงดังรูปที่ 4.4)

(4) กฎสำหรับกำหนดการเริ่มต้นและสิ้นสุดการดำเนินงาน ของสารบนไลฟ์ไลน์ที่เกิดขึ้น เป็นกฎที่ใช้ในการกำหนดจุดเริ่มต้นของสารที่ส่งไปยังไลฟ์ไลน์ และจุดสิ้นสุดของสารบนไลฟ์ไลน์ (แสดงดังรูปที่ 4.4)

(5) กฎสำหรับกำหนดการดำเนินงานของสารบนไลฟ์ไลน์ เป็นกฎที่ใช้ในการกำหนดการดำเนินงานที่มีการเกิดขึ้นบนไลฟ์ไลน์นั้น (แสดงในรูปที่ 4.4)

```
Lifeline name {[Maping name] !/?(send/receive) [Message name];}
Ex: Lifeline a {ab_msg!msg; ab_result?result;}
      Lifeline b {ab_msg?msg; ab_result!result;}
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 rule Interaction {
 from
     f: Fortran!Operation
 to
     s: SD!SendOperationEven( id<- f.MessageSendOrder+1,
        covered<- f.TypeName, message<-f.ProceduresName),
      r: SD!ReceiveOperationEven(id<- f.MessageReceiveOrder+1,
        covered <- f.TypeName, message<-f.ProceduresName),
    m: SD!MessageOccurrence (name <- f.ProceduresName,
       send <- s, receive <- r),
     e: SD!ExecuteOccurrence (id<- f.LifelineExecuteOrder+1,
         covered <- f.TypeName),
   se: SD!SendExecuteSpecification(id<- f.SendExecuteOrder+1,
         covered <- f.TypeName, start <- s, finish <- e),
   re: SD!ReceiveExecuteSpecification(
        id<- f. ReceiveExecuteOrder+1,
        covered <- f.TypeName, start <- r, finish <- e)
 }
```

รูปที่ 4.4 กฎสำหรับการสื่อสารกันระหว่างสารและไลฟ์ไลน์

จากรูปที่ 4.4 เป็นการแปลงคลาส Operation ของเมต้าโมเดล AST จากซอร์สโค้ดภาษาฟอร์แทรนไปสู่ Message Occurrence Node, Execution Occurrence Node และ Execution Specification Node ของเมต้าโมเดลซีเควนซ์ไดอะแกรม โดยในส่วนของ Message Occurrence Node จะเป็นการกำหนดการรับและส่งของสารที่เกิดขึ้นบนไลฟ์ไลน์ โดยจะมีแอตทริบิวท์ name เป็นตัวกำหนดชื่อของสาร แอตทริบิวท์ send เป็นตัวอ้างอิงถึง SendOperationEven ที่เป็นการกำหนดการส่งของสารบนไลฟ์ไลน์ และแอตทริบิวท์ receive เป็นตัวอ้างอิงถึง ReceiveOperationEven ที่เป็นการกำหนดการรับของสารบนไลฟ์ไลน์ สำหรับในส่วนของ Execution Occurrence Node จะเป็นการกำหนดการดำเนินงานของสารบนไลฟ์ไลน์ โดยจะมีแอตทริบิวท์ covered เป็นตัวกำหนดชื่อของไลฟ์ไลน์ที่มีการดำเนินงาน สำหรับในส่วนของ Execution Specification Node จะเป็นการกำหนดการเริ่มต้นและสิ้นสุดของสารบนไลฟ์ไลน์นั้น โดยแบ่งออกเป็น 2 ส่วน คือ ส่วนที่กำหนดการรับของสาร และส่วนที่กำหนดการส่งของสาร โดยจะมีแอตทริบิวท์ start เป็นตัวอ้างอิงถึง SendOperationEven, ReceiveOperationEven และแอตทริบิวท์ finish เป็นตัวอ้างอิงถึง ExecuteOccurrence เพื่อกำหนดการดำเนินงานของสารบนไลฟ์ไลน์นั้น

(6) กฎสำหรับการสร้างกรอบของแผนภาพ เพื่อแสดงเงื่อนไข ทางเลือกที่มีหลายเงื่อนไข และการทำซ้ำ เป็นกฎที่ใช้ในการกำหนดกรอบของแผนภาพที่เกิดขึ้น (แสดงดังรูปที่ 4.5)

```
rule Alt {
from
    f: Fortran!Statement(f.Isif_else)
 to
    a: SD!CombinedFragment( interactionOperator
    <- 'Alt')
}
```

รูปที่ 4.5 กฎสำหรับการสร้างกรอบของแผนภาพในส่วนที่แสดงเงื่อนไข

จากรูปที่ 4.5 เป็นการแปลงคลาส Statement ของเมต้าโมเดล AST จากซอร์สโค้ดภาษาฟอร์แทรน ไปสู่ Combined Fragment Node ของเมต้าโมเดลซีเควนซ์ไดอะแกรม โดยในส่วนของ Statement นั้นจะมีการตรวจสอบว่าเป็น Combined Fragment ลักษณะใด เช่น เงื่อนไข หรือการทำซ้ำ จากนั้นจึงทำการกำหนดชื่อของ Combined Fragment นั้น ๆ

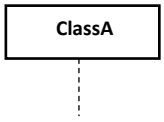
จากการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมข้างต้นผู้วิจัยได้พิจารณาข้อมูลเปรียบเทียบกับภาษาเชิงวัตถุภาษาอื่น ได้แก่ การเปรียบเทียบวากยสัมพันธ์ของภาษาเชิงวัตถุ เพื่อหาความสัมพันธ์ที่มีความคล้ายคลึงกันในการแสดงสัญลักษณ์แต่ละสัญลักษณ์ของยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งผู้วิจัยได้ทำการเปรียบเทียบกับภาษาจาวา ซึ่งเป็นภาษาโปรแกรมเชิงวัตถุที่นิยมในปัจจุบัน แสดงดังตารางที่ 4.1 โดยแบ่งออกเป็น 2 ส่วน ตามลักษณะการทำงานของสัญลักษณ์ซึ่งประกอบด้วย

(1) Lifeline แสดงถึง ตัวแทนของคลาส ประกอบด้วยชื่อของอินสแตนซ์ (Instance) และชื่อของคลาส

(2) Messages แสดงถึง สารที่ส่งระหว่างไลฟ์ไลน์ประกอบด้วย สารที่ใช้สร้างวัตถุ สารที่ใช้ตอบกลับ สารแบบซินโครนัส และสารแบบอะซิงโครนัส

ตารางที่ 4.1 เปรียบเทียบการแปลงซอร์สโค้ดเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมระหว่างภาษาจาวาและภาษาฟอร์แทรน

Rule	Java Syntax	Modern Fortran Syntax	Notations

Lifeline	package class_A;
	public class **ClassA** {
	...statement
	}
		module class_A
		type **ClassA**
		...
		end type ClassA
		... statement
		end module class_A
			

Messages			

Create Message	public class ClassB {
	ClassA a = new **ClassA**();
	}
		type ClassB
		type(**ClassA**) :: **a**
		end type ClassB
			

Reply Message	public class ClassC {
	public int getId(x) {
	...
	return id;
	}
	}
		type ClassC
		end type ClassC
		function getId(x) **result(id)**
		...
		end function getId	

Synchronous Message	public class ClassA {
	public void setA(..) {
	}
	}
	
	public class ClassB {
	public void callA(..) {
	a.setA(..);
	}
	}
		type ClassA
		end type ClassA
		subroutine **setA** (..)
		end subroutine setA
		
		type ClassB
		end type ClassB
		subroutine callA (..)
		call setA(..)
		end subroutine callA
			

Asynchronous Message	public class AsyncClassA {
	public void setA(..) {
	}
	}
	
	public class ClassB {
	public void callA(..) {
	a.setA(..);
	}
	}
		type AsyncClassA
		asynchronous :: a
		end type ClassA
		subroutine **setA** (..)
		end subroutine setA
		
		type ClassB
		end type ClassB
		subroutine callA (..)
		call setA(..)
		end subroutine callA	

สำหรับในส่วน Interaction Fragment ซึ่งเป็นส่วนที่สื่อสารกันระหว่างสารและไลฟ์ไลน์ ประกอบด้วย การรับและส่งของสารที่เกิดขึ้น การเริ่มต้นและสิ้นสุดการดำเนินงานของสารบนไลฟ์ไลน์ที่เกิดขึ้น ข้อกำหนดการดำเนินงานของสารบนไลฟ์ไลน์ และกรอบของแผนภาพ สามารถแสดงสัญลักษณ์ที่เหมือนกับในภาษาจาวาได้ ดังตารางที่ 4.2

ตารางที่ 4.2 สัญลักษณ์ของยูเอ็มแอลในส่วน Interaction Fragment

Rule	Fortran Semantics	Notations
Interaction Fragment		
Message Occurrence	Send and Receive Occurrence	
Execution Specification	Start and Finish Occurrence	
Execution Occurrence	Activation	
Combined Fragment	Loops, Branches, and Other Alternatives	

เพื่อเป็นการตรวจสอบว่ากฎการแปลงที่ได้สร้างขึ้นสามารถนำไปประยุกต์ใช้กับภาษาฟอร์แทรนได้จริง ผู้วิจัยได้ทำการยกตัวอย่างการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมา โดยประยุกต์ใช้กฎการแปลงที่ได้ทำการออกแบบไว้กับซอร์สโค้ดในภาษาฟอร์แทรนที่นำมาจากงานของ Akin (Akin, 2003) เพื่อสร้างยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้จะสอดคล้องกับสัญลักษณ์ของภาษายูเอ็มแอล ดังที่แสดงในตารางที่ 4.1 และตารางที่ 4.2 โดยซอร์สโค้ดภาษาฟอร์แทรนจะถูกแสดงดังรูปที่ 4.6 4.7 4.8 และ 4.9 และยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้มาจากซอร์สโค้ดดังกล่าวจะถูกแสดงดังรูปที่ 4.10

```
include 'class_Date.f90'
include 'class_Person.f90'
include 'class_Student.f90' ! see previous figure
program main ! create or correct a student
    use class_Student ! inherits class_Person, class_Date also
    type (Person) :: p
    type (Student) :: x
    ! Method 1
    p = make_Person ("Ann Jones","",0) ! optional person constructor
    x = Student_(p, "219360061", Date_(8,29,1955), 9, 3.1) ! public
    call set_DOB (p, 5, 13, 1977) ! add birth to person data
    call print_Name (p) ! list name
    print *, "Born :"; call print_DOB (p) ! list dob
    print *, "Sex :"; call print_Sex (p) ! list sex
    print *, "Matriculated:"; call print_DOM (x) ! list dom
    call print_GPA (x) ! list gpa
end program main
```

รูปที่ 4.6 ตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของโปรแกรมหลัก (Program Main)

จากรูปที่ 4.6 แสดงตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของโปรแกรมหลัก ซึ่งจะมีการเรียกใช้งานไฟล์ class_Date.f90 class_Person.f90 และ class_Student.f90 โดยทั้ง 3 ไฟล์นั้น จะเป็นมอดูลย่อยไว้สำหรับให้โปรแกรมหลักกำหนดการทำงาน ซึ่งโปรแกรมหลักจะประกอบด้วยส่วนที่เรียกใช้งานฟังก์ชัน คือ p = make_Person และ x = Student_ ส่วนที่เรียกใช้งานซับรูทีน คือ call set_DOB call print_Name call print_DOB call print_Sex call print_DOM และ call print_GPA

```
module class_Date ! filename: class_Date.f90
    public :: Date ! and everything not "private"
    type Date
        private
        integer :: month, day, year
    end type Date
    contains ! encapsulated functionality
    function Date_ (m, d, y) result (x) ! public constructor
        integer, intent(in) :: m, d, y ! month, day, year
        type (Date) :: x ! from intrinsic constructor
        if ( m < 1 .or. d < 1 ) stop 'Invalid components, Date_'
        x = Date (m, d, y) ; end function Date_
    subroutine print_Date (x) ! check and pretty print a date
        type (Date), intent(out) :: x
        character (len=*),parameter :: month_Name(12) = &
        (/ "January ", "February ", "March ", "April ",&
        "May ", "June ", "July ", "August ",&
        "September", "October ", "November ", "December "/)
        if ( x%month < 1 .or. x%month > 12 ) print *, "Invalid month"
        if ( x%day < 1 .or. x%day > 31 ) print *, "Invalid day "
        print *, trim(month_Name(x%month)),' ', x%day, ", ", x%year;
     end subroutine print_Date
    subroutine read_Date (x) ! read month, day, and year
        type (Date), intent(out) :: x ! into intrinsic constructor
        read *, x ; end subroutine read_Date
    function set_Date (m, d, y) result (x) ! manual constructor
        integer, optional, intent(in) :: m, d, y ! month, day, year
        type (Date) :: x
        x = Date (1,1,1997) ! default, (or use current date)
        if ( present(m) ) x%month = m ; if ( present(d) ) x%day = d
        if ( present(y) ) x%year = y ; end function set_Date
end module class_Date
```

รูปที่ 4.7 ตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของ class_Date

จากรูปที่ 4.7 แสดงตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของ class_Date ซึ่งจะมีการกำหนดชื่อของคลาส คือ Date และมีส่วนที่เป็นฟังก์ชัน คือ Date_ และ set_Date ส่วนที่เป็นซับรูทีน คือ print_Date และ read_Date

```
module class_Student ! filename class_Student.f90
        use class_Person ! inherits class_Date
        public :: Student, set_DOM, print_DOM
        type Student
            private
            type (Person) :: who ; character (len=9) :: id; type (Date) :: dom; integer :: credits;  real :: gpa
        end type Student
        contains ! coupled functionality
        function get_person (s) result (p)
            type (Student), intent(in) :: s ; type (Person) :: p ! name and sex
            p = s % who ; end function get_person
        function make_Student (w, n, d, c, g) result (x)  ! Optional Constructor for a Student type
            type (Person), intent(in) :: w ! who
            character (len=*), optional, intent(in) :: n ! ssn
            type (Date), optional, intent(in) :: d ! matriculation
            integer, optional, intent(in) :: c ! credits
            real, optional, intent(in) :: g ! grade point ave
            type (Student) :: x ! new student
            x = Student_(w, " ", Date_(1,1,1), 0, 0.) ! defaults
            if ( present(n) ) x % id = n ;  if ( present(d) ) x % dom = d ! optional values
            if ( present(c) ) x % credits = c;   if ( present(g) ) x % gpa = g; end function make_Student
       subroutine print_DOM (who)
            type (Student), intent(in) :: who
            call print_Date(who%dom) ; end subroutine print_DOM
        subroutine print_GPA (x)
            type (Student), intent(in) :: x
            print *,"My name is "; call print_Name (x % who)
            print *,", and my G.P.A. is ", x % gpa, "."; end subroutine
        subroutine set_DOM (who, m, d, y)
            type (Student), intent(inout) :: who ; integer, intent(in) :: m, d, y
            who % dom = Date_( m, d, y) ; end subroutine set_DOM
        function Student_ (w, n, d, c, g) result (x)   ! Public Constructor for a Student type
            type (Person), intent(in) :: w ;   character (len=*), intent(in) :: n ; type (Date), intent(in) :: d
            integer, intent(in) :: c; real, intent(in) :: g ;    type (Student) :: x ! new student
            x = Student (w, n, d, c, g) ; end function Student_
 end module class_Student
```

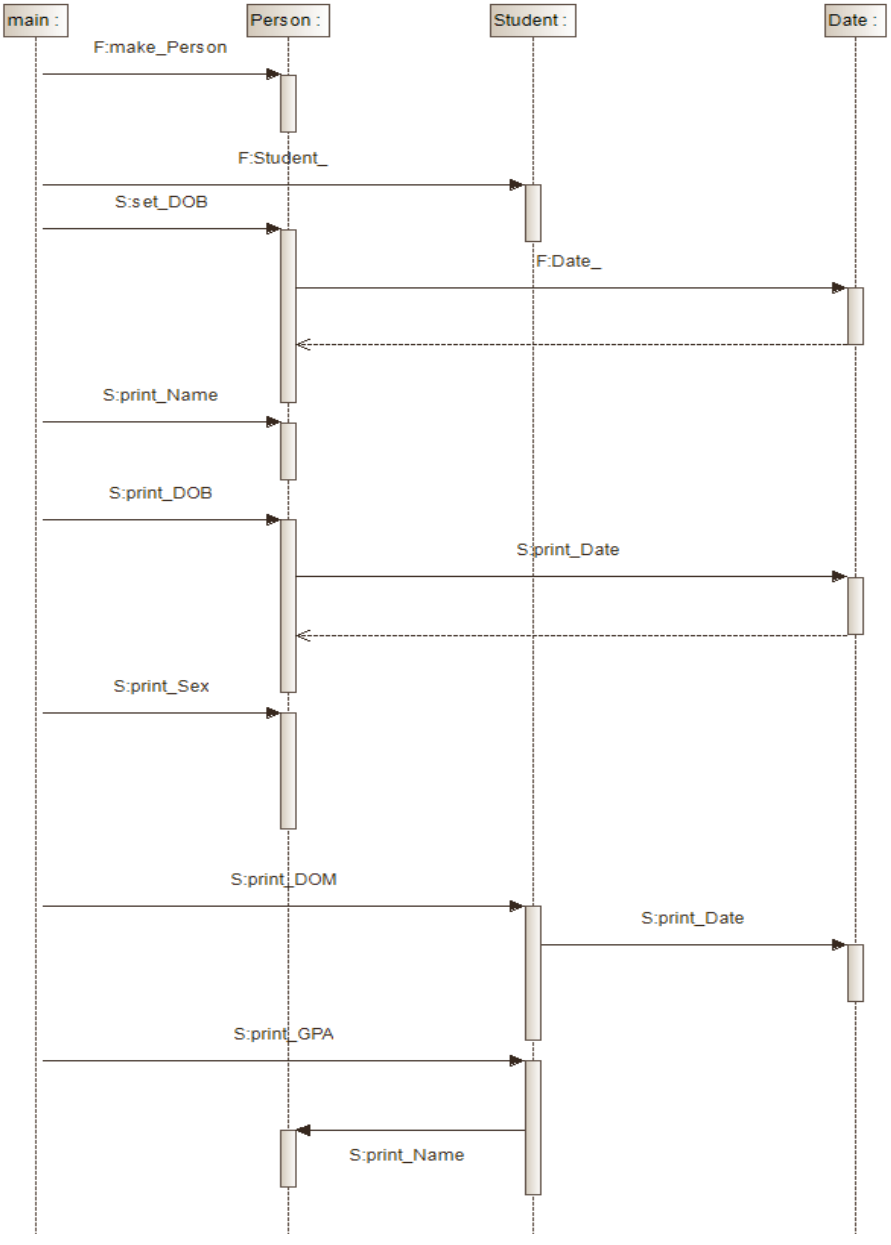
รูปที่ 4.8 ตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของ class_Student

จากรูปที่ 4.8 แสดงตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของ class_Student ซึ่งจะมีการกำหนดชื่อของคลาส คือ Student และมีส่วนที่เป็นฟังก์ชัน คือ get_person make_Student และ Student_ ส่วนที่เป็นซับรูทีน คือ print_DOM print_GPA และ set_DOM โดยในส่วนของซับรูทีน print_DOM จะมีการเรียกใช้งานซับรูทีน print_Date ในคลาส Date และในส่วนของซับรูทีน print_GPA จะมีการเรียกใช้งานซับรูทีน print_Name ในคลาส Person

```
module class_Person ! filename: class_Person.f90
    use class_Date
    public :: Person
    type Person
        private
        character (len=20) :: name; character (len=20) :: nationality; integer :: sex
    type (Date) :: dob, dod ! birth, death
    end type Person
    contains
    function make_Person (nam, nation, s, b, d) result (who) ! Optional Constructor for a Person type
        character (len=*), optional, intent(in) :: nam, nation
        integer, optional, intent(in) :: s ! sex
        type (Date), optional, intent(in) :: b, d ! birth, death
        type (Person) :: who
        who = Person (" ","USA",1,Date_(1,1,0),Date_(1,1,0))! defaults
        if ( present(nam) ) who % name = nam; if ( present(nation) ) who % nationality = nation
        if ( present(s) ) who % sex = s; if ( present(b) ) who % dob = b; if ( present(d) ) who % dod = d
    end function make_Person
    function Person_ (nam, nation, s, b, d) result (who) ! Public Constructor for a Person type
        character (len=*), intent(in) :: nam, nation; integer, intent(in) :: s ! sex
        type (Date), intent(in) :: b, d ! birth, death
        type (Person) :: who
        who = Person (nam, nation, s, b, d) ; end function Person_
    subroutine print_DOB (who)
        type (Person), intent(out) :: who
        call print_Date (who % dob) ; end subroutine print_DOB
    subroutine print_DOD (who)
        type (Person), intent(inout) :: who
        call print_Date (who % dod) ; end subroutine print_DOD
    subroutine print_Name (who)
        type (Person), intent(in) :: who
        print *, who % name ; end subroutine print_Name
    subroutine print_Nationality (who)
        type (Person), intent(in) :: who
        print *, who % nationality ; end subroutine print_Nationality
    subroutine print_Sex (who)
        type (Person), intent(in) :: who
        if ( who % sex == 1 ) then ; print *, "male"
        else ; print *, "female" ; end if ; end subroutine print_Sex
    subroutine set_DOB (who, m, d, y)
        type (Person), intent(inout) :: who
        integer, intent(in) :: m, d, y ! month, day, year
        who % dob = Date_ (m, d, y) ; end subroutine set_DOB
    subroutine set_DOD(who, m, d, y)
        type (Person), intent(inout) :: who
        integer, intent(in) :: m, d, y ! month, day, year
        who % dod = Date_ (m, d, y) ; end subroutine set_DOD
end module class_Person
```

รูปที่ 4.9 ตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของ class_Person

จากรูปที่ 4.9 แสดงตัวอย่างของซอร์สโค้ดภาษาฟอร์แทรนในส่วนของ class_Person ซึ่งจะมีการกำหนดชื่อของคลาส คือ Person และมีส่วนที่เป็นฟังก์ชัน คือ make_Person และ Person_ ส่วนที่เป็นซับรูทีน คือ print_DOB print_DOD print_Name print_Nationality print_Sex set_DOB และ set_DOD โดยในส่วนของฟังก์ชัน make_Person จะมีการเรียกใช้งานฟังก์ชัน Date_ ในคลาส Date และในส่วนของซับรูทีน print_DOB จะมีการเรียกใช้งานซับรูทีน print_Date ในคลาส Date



รูปที่ 4.10 ยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้มาจากซอร์สโค้ดตัวอย่าง

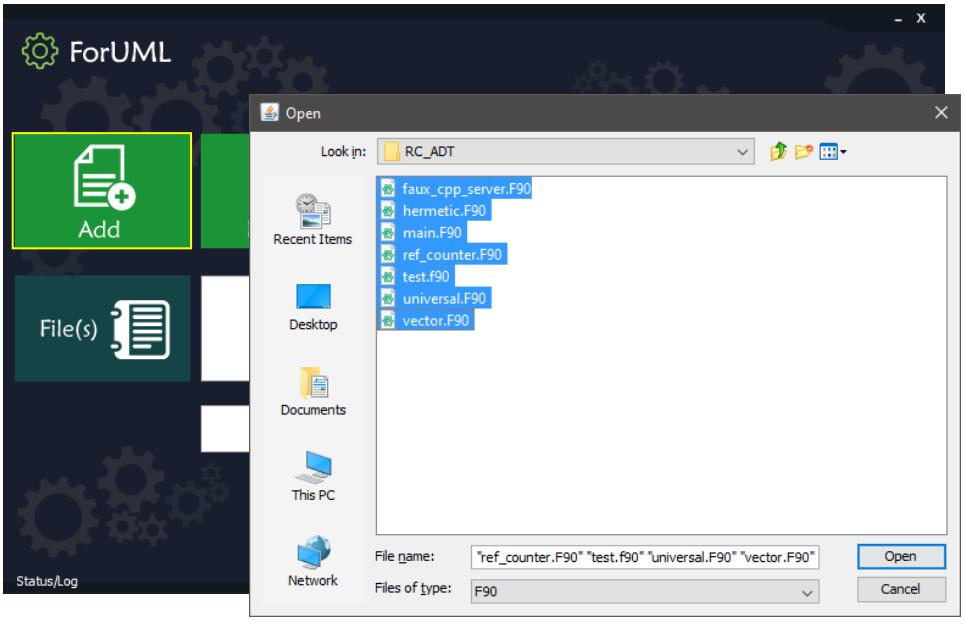
จากรูปที่ 4.10 แสดงยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้มาจากซอร์สโค้ดตัวอย่างมีไลฟ์ไลน์ทั้งหมด 4 ไลฟ์ไลน์ คือ main Person Student และ Date โดยแต่ละไลฟ์ไลน์จะแสดงถึงคลาสภายในซอร์สโค้ดตัวอย่างของแต่ละไฟล์ ซึ่งจะเริ่มต้นการทำงานจากคลาส main เสมอ สำหรับการกำหนดการรับและส่งของสารระหว่างไลฟ์ไลน์นั้น จะกำหนดจากการเรียกใช้งานฟังก์ชันหรือซับรูทีนระหว่างคลาส เช่น คลาส main มีการเรียกใช้งานฟังก์ชัน make_Person ของคลาส Person คลาส main มีการเรียกใช้งานฟังก์ชัน Student_ ของคลาส Student คลาส main มีการเรียกใช้งานซับรูทีน set_DOB ของคลาส Person เป็นต้น ซึ่งผู้วิจัยได้มีการกำหนดชนิดของสารโดยแบ่งตามชนิดของเมท็อด คือ เมท็อดที่เป็นฟังก์ชันจะมีการกำหนดสัญลักษณ์ F ไว้ด้านหน้า และเมท็อดที่เป็นซับรูทีนจะมีการกำหนดสัญลักษณ์ S ไว้ด้านหน้า

4.2 การพัฒนาระบบ

งานวิจัยนี้ได้ออกแบบและพัฒนาซอฟต์แวร์ที่ใช้สำหรับการแปลงซอร์สโค้ดภาษาฟอร์แทรนให้อยู่ในรูปของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยระบบที่ได้พัฒนาขึ้นมาสามารถแบ่งการทำงานออกเป็น 4 ส่วนหลักดังนี้ 1) ส่วนสำหรับการจัดการเอกสารซอร์สโค้ดภาษาฟอร์แทรน 2) ส่วนสำหรับการหาความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม 3) ส่วนสำหรับการสร้างเอกสารเอกซ์เอ็มไอ 4) ส่วนสำหรับแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมจากเอกสารเอกซ์เอ็มไอ ซึ่งทั้ง 4 ส่วนได้ถูกพัฒนาขึ้นในซอฟต์แวร์ชื่อ ForUML โดยมีรายละเอียดดังนี้

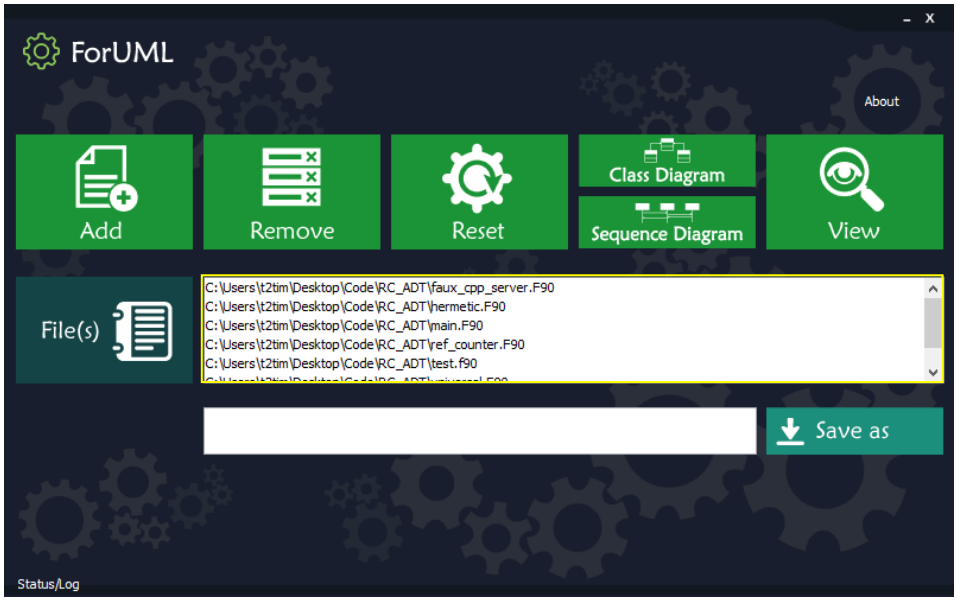
4.2.1 ส่วนสำหรับการจัดการเอกสารซอร์สโค้ดภาษาฟอร์แทรน

การจัดการเอกสารซอร์สโค้ดภาษาฟอร์แทรนเป็นขั้นตอนในการจัดการเกี่ยวกับรายละเอียดของเอกสารซอร์สโค้ดภาษาฟอร์แทรน โดยขั้นตอนการจัดการเอกสารเริ่มต้นจากการเพิ่มเอกสาร ซึ่งสามารถเพิ่มได้จากการกดปุ่ม Add หรือทำลากเอกสารที่ต้องการลงในช่องที่กำหนดไว้ และจะมีการกำหนดรายละเอียดของเอกสารที่สามารถเพิ่มได้เฉพาะเอกสารที่มีนามสกุลเป็น .F90 เท่านั้น โดยเอกสารที่มีนามสกุลดังกล่าวจะต้องเป็นเอกสารที่มีการใช้ภาษาเชิงวัตถุในการเขียนขึ้นมา แสดงดังรูปที่ 4.11

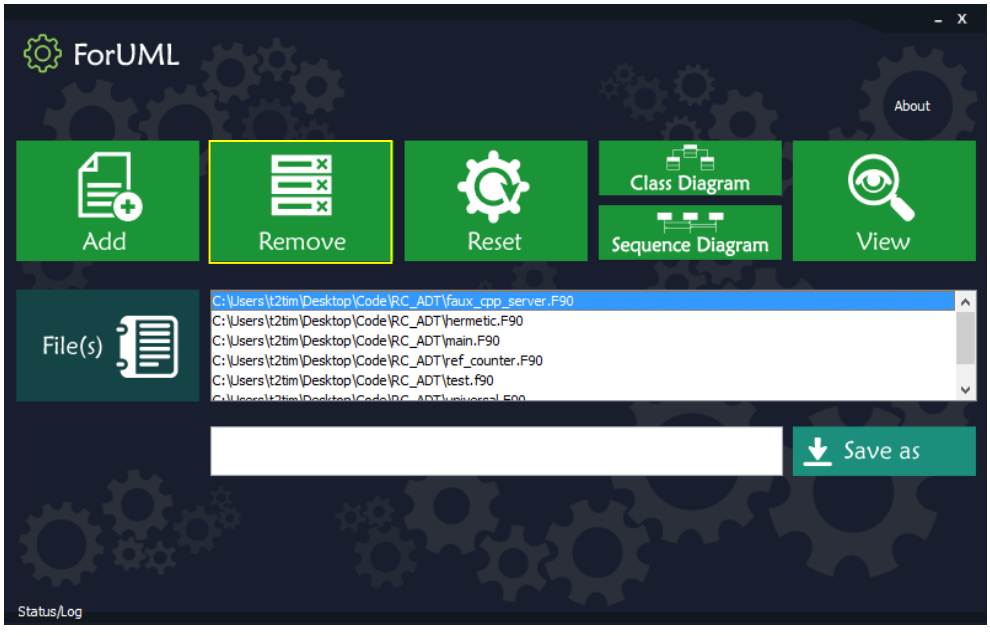


รูปที่ 4.11 ขั้นตอนการเพิ่มเอกสาร

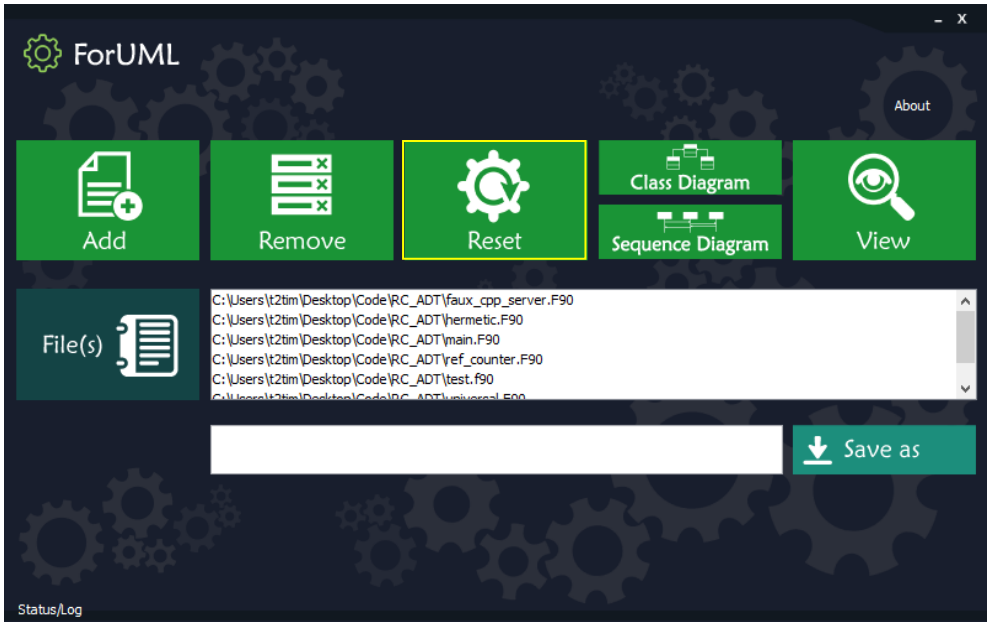
หลังจากทำการเพิ่มเอกสารเรียบร้อยแล้วรายละเอียดของเอกสารที่เพิ่มเข้ามาจะแสดงในหน้าจอของระบบ ดังรูปที่ 4.12 ซึ่งเอกสารดังกล่าวสามารถลบโดยการเลือกเอกสารที่ต้องการจะลบแล้วกดปุ่ม Remove ดังรูปที่ 4.13 หรือหากต้องการรีเซตระบบทั้งหมดสามารถกดปุ่ม Reset ดังรูปที่ 4.14



รูปที่ 4.12 รายละเอียดของเอกสาร

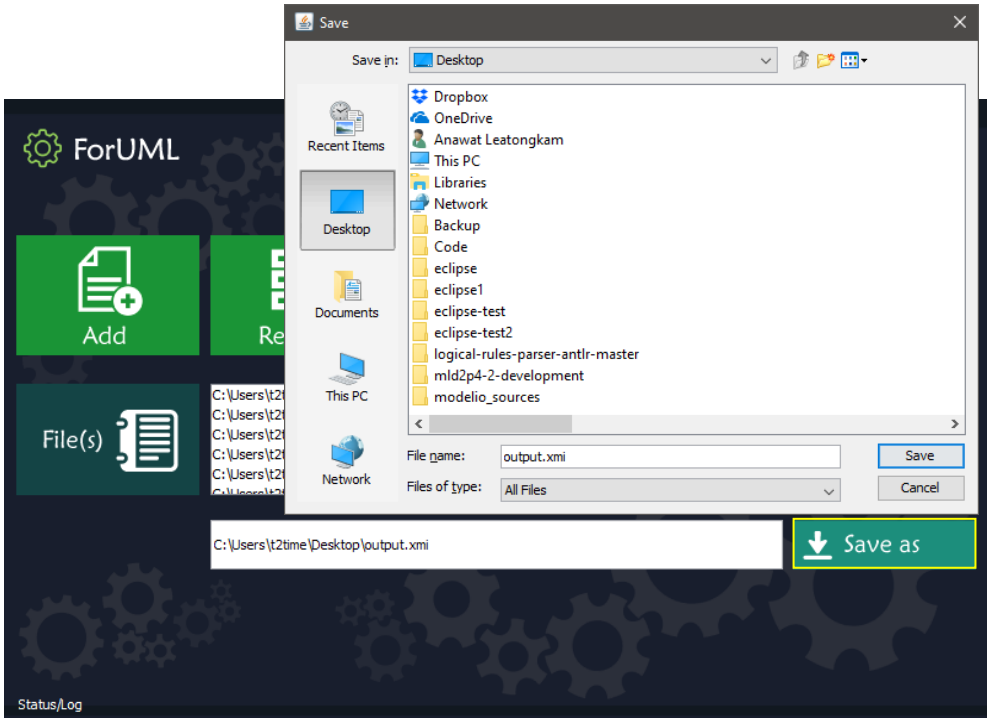


รูปที่ 4.13 ขั้นตอนการลบเอกสาร



รูปที่ 4.14 ขั้นตอนการรีเซตระบบ

สำหรับขั้นตอนการบันทึกเอกสารนั้นจะมีการกำหนดให้เลือกสถานที่บันทึกเอกสารก่อนทำการสร้างเอกสารเอกซ์เอ็มไอขึ้นมา แสดงดังรูปที่ 4.15



รูปที่ 4.15 ขั้นตอนการบันทึกเอกสาร

4.2.2 ส่วนสำหรับการหาความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม

การหาความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นขั้นตอนหลังจากนำเข้าเอกสารซอร์สโค้ดภาษาฟอร์แทรนเข้าสู่ระบบ จากนั้นมีการแยกส่วนประกอบต่าง ๆ ของซอร์สโค้ดอยู่ในรูปของโครงสร้าง AST โดยจะมีการใช้ไลบรารี OFP ในการแยกส่วนประกอบของซอร์สโค้ด จากนั้นจึงนำส่วนประกอบที่ได้มาหาความสัมพันธ์กันเพื่อสร้างเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งการสร้างเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมจำเป็นต้องมีการกำหนดรายละเอียดที่สำคัญ 5 ส่วนด้วยกันตามมาตรฐานของ OMG ได้แก่ 1) Lifeline 2) Message 3) Message Occurrence Specification 4) Execution Occurrence Specification และ 5) Behavior Execution Specification โดยผู้วิจัยได้พัฒนาไลบรารีเพื่อหาความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและแผนภาพไดอะแกรม ซึ่งมีโครงสร้างและรายละเอียดของคลาส แสดงดังรูปที่ 4.16 ซึ่งแต่ละคลาสมีหน้าที่การทำงานดังนี้

(1) Type เป็นข้อมูลชื่อของคลาสที่ได้มาจากซอร์สโค้ดภาษาฟอร์แทรน ซึ่งเป็นข้อมูลที่ใช้สำหรับกำหนดรายละเอียดที่เกี่ยวข้องกับไลฟ์ไลน์

(2) Function เป็นข้อมูลชื่อของเมท็อดที่ได้มาจากซอร์สโค้ดภาษาฟอร์แทรน ซึ่งเป็นข้อมูลที่ใช้สำหรับกำหนดรายละเอียดที่เกี่ยวข้องกับสารที่เป็นฟังก์ชัน

(3) Subroutine เป็นข้อมูลชื่อของเมท็อดที่ได้มาจากซอร์สโค้ดภาษาฟอร์แทรน ซึ่งเป็นข้อมูลที่ใช้สำหรับกำหนดรายละเอียดที่เกี่ยวข้องกับสารที่เป็นซับรูทีน

(4) SD_ Lifeline ทำหน้าที่ในการรับข้อมูลชื่อของ Type จากเอกสารซอร์สโค้ดภาษาฟอร์แทรนเพื่อใช้ในการกำหนดรายละเอียดของไลฟ์ไลน์ขึ้นมา มีการกำหนดการทำงานที่เกิดขึ้นครอบคลุมบนไลฟ์ไลน์นั้น ๆ และมีการเชื่อมโยงกับคลาส SD_EOS เพื่อกำหนดการทำงานที่มีการสิ้นสุดบนไลฟ์ไลน์นั้น

(5) SD_Function ทำหน้าที่ในการรับชื่อของ Function จากเอกสารซอร์สโค้ดภาษาฟอร์แทรนเพื่อใช้ในการกำหนดรายละเอียดของสารขึ้นมา มีการกำหนดรายละเอียดการเรียกใช้งานของ Function เพื่อกำหนดการรับและส่งของสารที่เกิดขึ้น และมีการเชื่อมโยงกับคลาส SD_MOS เพื่อกำหนดว่าสารดังกล่าวอยู่บนไลฟ์ไลน์ใด

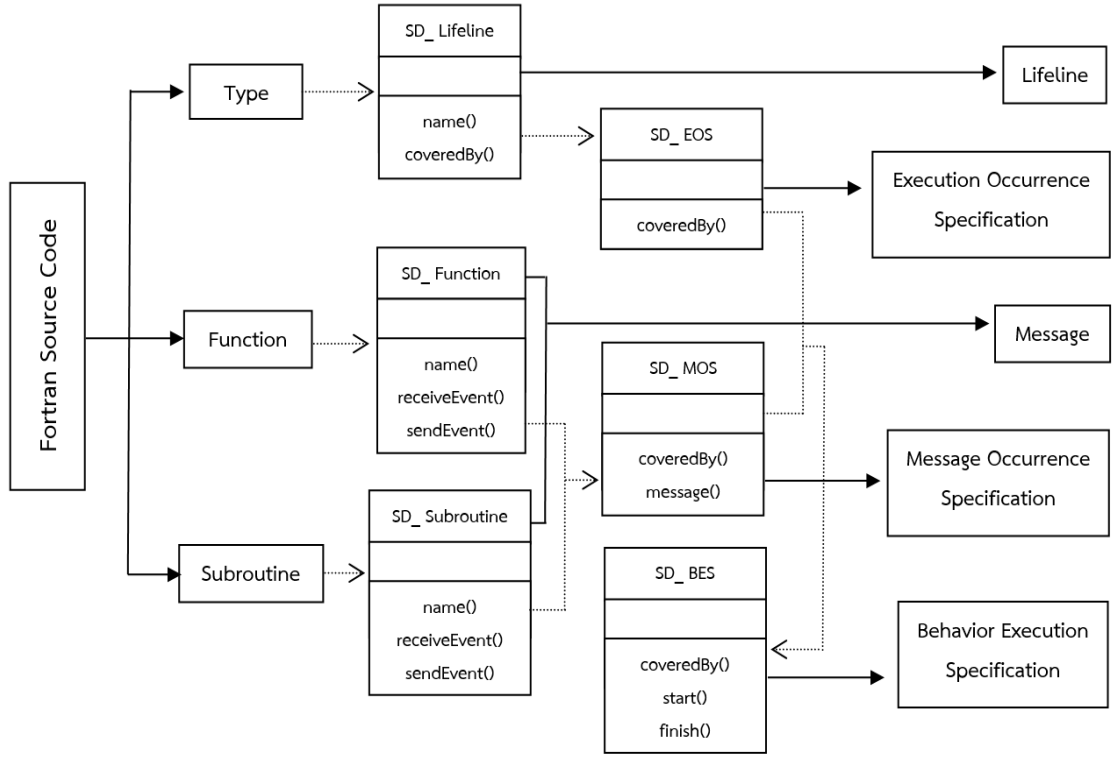
(6) SD_Subroutine ทำหน้าที่ในการรับชื่อของ Subroutine จากเอกสารซอร์สโค้ดภาษาฟอร์แทรนเพื่อใช้ในการกำหนดรายละเอียดของสารขึ้นมา มีการกำหนดรายละเอียดการเรียกใช้งานของ Subroutine เพื่อกำหนดการรับและส่งของสารที่เกิดขึ้น และมีการเชื่อมโยงกับคลาส SD_MOS เพื่อกำหนดว่าสารดังกล่าวอยู่บนไลฟ์ไลน์ใด

(7) SD_EOS ทำหน้าที่กำหนดจุดสิ้นสุดการทำงานเกิดขึ้นตรงไลฟ์ไลน์ใด กำหนดโดยการตรวจสอบจากการเรียกใช้งานของ Function หรือ Subroutine ถ้าหาก Function หรือ Subroutine ไม่มีการเรียกใช้งานไปยัง Type อื่นแสดงว่า Function หรือ Subroutine นั้นมีการสิ้นสุดการทำงาน และมีการเชื่อมโยงกับคลาส SD_BES เพื่อกำหนดพฤติกรรมที่เกิดขึ้นของสารบนไลฟ์ไลน์

(8) SD_MOS ทำหน้าที่กำหนดการทำงานของสารว่ามีการเกิดขึ้นตรงไลฟ์ไลน์ใด กำหนดโดยการตรวจสอบ Function หรือ Subroutine มีการทำงานเกิดขึ้นตรง Type ใดในเอกสารเอกสารซอร์สโค้ดภาษาฟอร์แทรน และมีการเชื่อมโยงกับคลาส SD_BES เพื่อกำหนดพฤติกรรมที่เกิดขึ้นของสารบนไลฟ์ไลน์

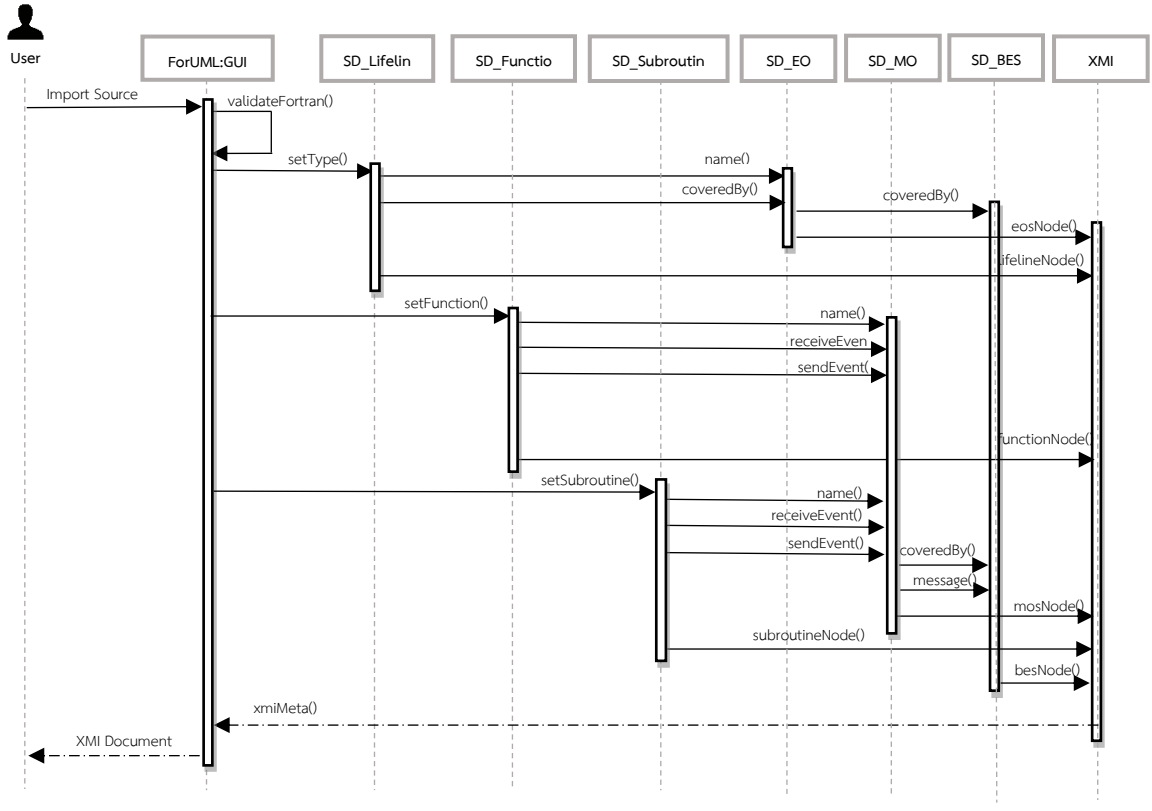
(9) SD_BES ทำหน้าที่กำหนดพฤติกรรมการทำงานระหว่างสารและไลฟ์ไลน์ กำหนดโดยการตรวจสอบการเรียกใช้งานของ Function หรือ Subroutine ระหว่าง Type ถ้าหากมีการเรียกใช้งานเกิดขึ้นจะทำการกำหนดจุดเริ่มต้นของไลฟ์ไลน์ที่มีการเรียกใช้งานและจุดสิ้นสุดของไลฟ์ไลน์ที่

ถูกใช้งานโดยรับข้อมูลมาจากคลาส SD_MOS และถ้าหาก Function หรือ Subroutine ไม่มีการเรียกใช้งานต่อจะทำการกำหนดจุดสิ้นสุดการทำงานโดยรับข้อมูลมาจากคลาส SD_EOS



รูปที่ 4.16 ความสัมพันธ์ของคลาสในส่วนของการแปลงเป็นเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม

จากโครงสร้างความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมข้างต้น เพื่อง่ายต่อการทำความเข้าใจผู้วิจัยได้สร้างแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของโครงสร้างดังกล่าวขึ้นมา แสดงดังรูปที่ 4.17 ซึ่งจะเริ่มตั้งแต่ผู้ใช้งาน (User) นำเข้าซอร์สโค้ดภาษาฟอร์แทรนเข้าสู่ระบบ จากนั้นระบบจะทำการตรวจสอบความถูกต้องของซอร์สโค้ด ถ้าหากซอร์สโค้ดมีข้อผิดพลาดเกิดขึ้นระบบจะมีการแจ้งเตือนผ่านกล่องข้อความ status/log หลังจากนั้นระบบจะทำการส่งข้อมูลที่ผ่านการแยกส่วนประกอบแล้วมาหาความสัมพันธ์ ซึ่งข้อมูลที่ทำการส่งจะประกอบด้วย Type Function และ Subroutine โดยจะมีคลาส SD_Lifeline SD_Function และ SD_Subroutine สำหรับเก็บข้อมูลดังกล่าว จากนั้นจึงนำข้อมูลที่เก็บไว้มาทำการกำหนดความสัมพันธ์ ซึ่งจะมีคลาสสำหรับกำหนดความสัมพันธ์ คือ SD_EOS SD_MOS และSD_BES สุดท้ายเมื่อได้ความสัมพันธ์แล้วจึงนำมาสร้างเป็นเอกสารเอกซ์เอ็มไอ



รูปที่ 4.17 แผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของโครงสร้างความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม

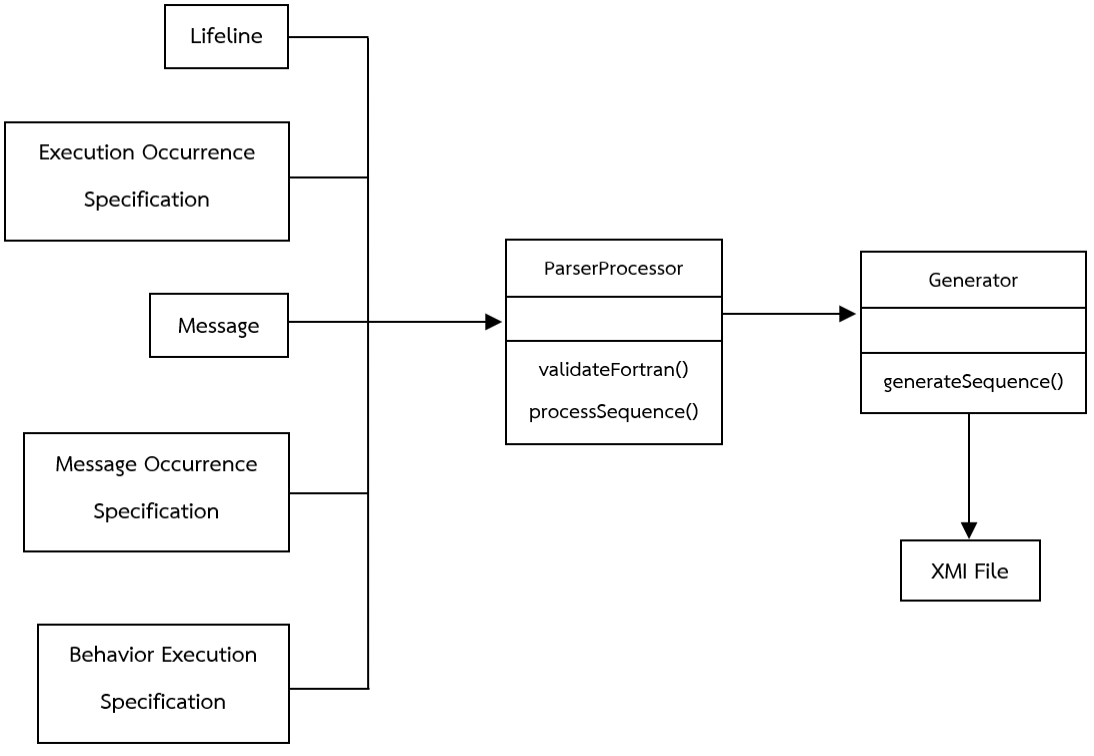
4.2.3 ส่วนสำหรับการสร้างเอกสารเอกซ์เอ็มไอ

การสร้างเอกสารเอกซ์เอ็มไอเป็นขั้นตอนหลังจากการหาความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยหลังจากได้ความสัมพันธ์ที่เป็นไปตามข้อกำหนดในการสร้างเอกสารเอกซ์เอ็มไอแล้วจะนำความสัมพันธ์ที่ได้มาสร้างเอกสารเอกซ์เอ็มไอ ผู้วิจัยได้พัฒนาไลบรารีเพื่อสร้างเอกสารเอกซ์เอ็มไอที่มีโครงสร้าง แสดงดังรูปที่ 4.18 โดยมีรายละเอียดดังนี้

(1) ParserProcessor ทำหน้าที่ในการประมาลผลการสร้างเอกสารเอกซ์เอ็มไอ โดยมีการตรวจสอบเอกสารซอร์สโค้ดภาษาฟอร์แทรนที่รับเข้ามาเป็นไฟล์ชนิด .F90 มีส่วนประกอบของ Module และไม่มีส่วนของซอร์สโค้ดที่ผิดพลาด จากนั้นจะส่งเอกสารที่ผ่านการตรวจสอบแล้วไปแยกส่วนประกอบต่าง ๆ ของซอร์สโค้ด โดยมีไลบรารี OFP ในการแยกส่วนประกอบ เมื่อแยกส่วนประกอบเสร็จแล้วจะทำการหาความสัมพันธ์ดังที่อธิบายไปในขั้นตอนการหาความสัมพันธ์

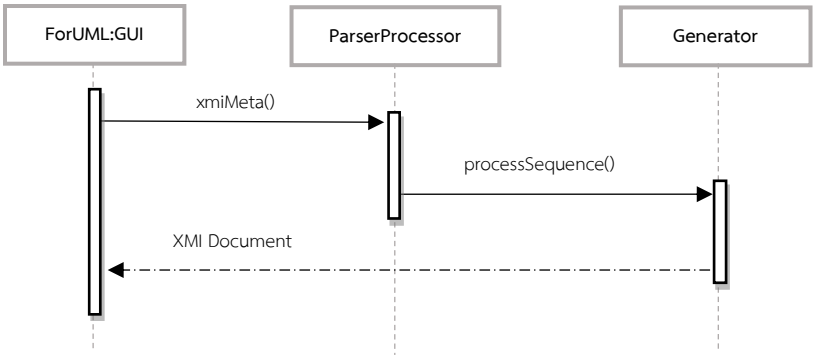
เมื่อได้ความสัมพันธ์แล้วจะทำการส่งรายละเอียดที่ได้กลับมาเพื่อทำการสร้างเอกสารเอกซ์เอ็มไอโดยมีการเชื่อมโยงกับคลาส Generator

(2) Generator ทำหน้าที่ในการสร้างเอกสารเอกซ์เอ็มไอ โดยมีข้อกำหนดของเอกสารเอกซ์เอ็มไอตามมาตรฐานของ OMG เพื่อแสดงแผนภาพแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม เวอร์ชัน 2.0



รูปที่ 4.18 ความสัมพันธ์ของคลาสในส่วนของการสร้างเอกสารเอกซ์เอ็มไอ

จากโครงสร้างของการสร้างเอกสารเอกซ์เอ็มไอข้างต้น เพื่อง่ายต่อการทำความเข้าใจ ผู้วิจัยได้สร้างแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของโครงสร้างดังกล่าวขึ้นมา แสดงดังรูปที่ 4.19 ซึ่งจะเริ่มต้นจากระบบทำการส่งข้อมูลสำหรับสร้างเอกสารเอกซ์เอ็มไอไปยังคลาส ParserProcessor เพื่อประมวลผล โดยข้อมูลดังกล่าวจะได้มาหลังจากขั้นตอนการหาความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม ซึ่งได้ผ่านการตรวจสอบความถูกต้องของซอร์สโค้ดแล้ว หลังจากนั้นจึงนำข้อมูลที่ผ่านการประมวลผลส่งไปยังคลาส Generator เพื่อทำการสร้างเอกสารเอกซ์เอ็มไอต่อไป

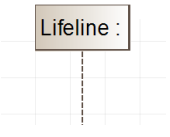
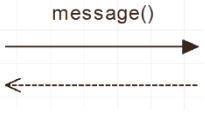


รูปที่ 4.19 แผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของโครงสร้างความสัมพันธ์ระหว่างซอร์สโค้ดภาษาฟอร์แทรนและเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม

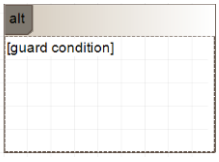
4.2.4 ส่วนสำหรับแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมจากเอกสารเอกซ์เอ็มไอ

การแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมจากเอกสารเอกซ์เอ็มไอนั้น จะแสดงผ่านซอฟต์แวร์มอเดลลิโอ ซึ่งเป็นซอฟต์แวร์ที่ใช้สำหรับสร้างหรือแสดงแผนภาพยูเอ็มแอล มอเดลลิโอเป็นซอฟต์แวร์ประเภทโอเพนซอร์สและอยู่ภายใต้สัญญาของ GPL โดยมีเวอร์ชัน 3.7 เป็นเวอร์ชันล่าสุด และรองรับมาตรฐานของยูเอ็มแอลเวอร์ชัน 2.0 ซึ่งผู้วิจัยได้เลือกใช้เวอร์ชัน 3.7 มาประยุกต์ในการแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยมีสัญลักษณ์ที่สำคัญสำหรับแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม ดังตารางที่ 4.3

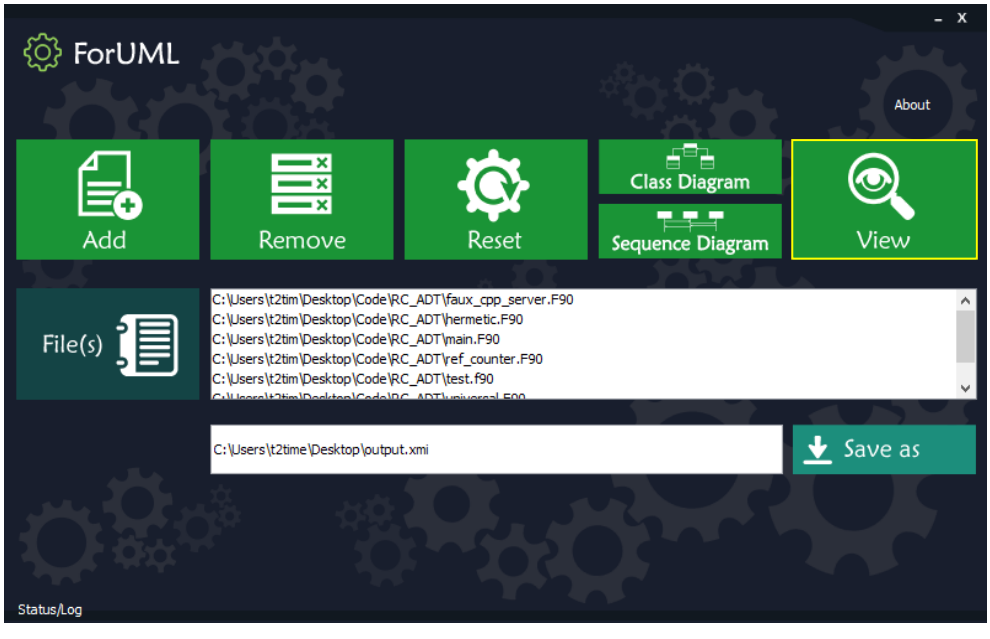
ตารางที่ 4.3 สัญลักษณ์สำหรับแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม

ชื่อ	สัญลักษณ์	ความหมาย
Lifeline		แสดงถึง อ็อบเจกต์หรือคลาส
Message		แสดงถึง การเรียกใช้งานของอ็อบเจกต์หนึ่งไปยังอีกอ็อบเจกต์หนึ่ง

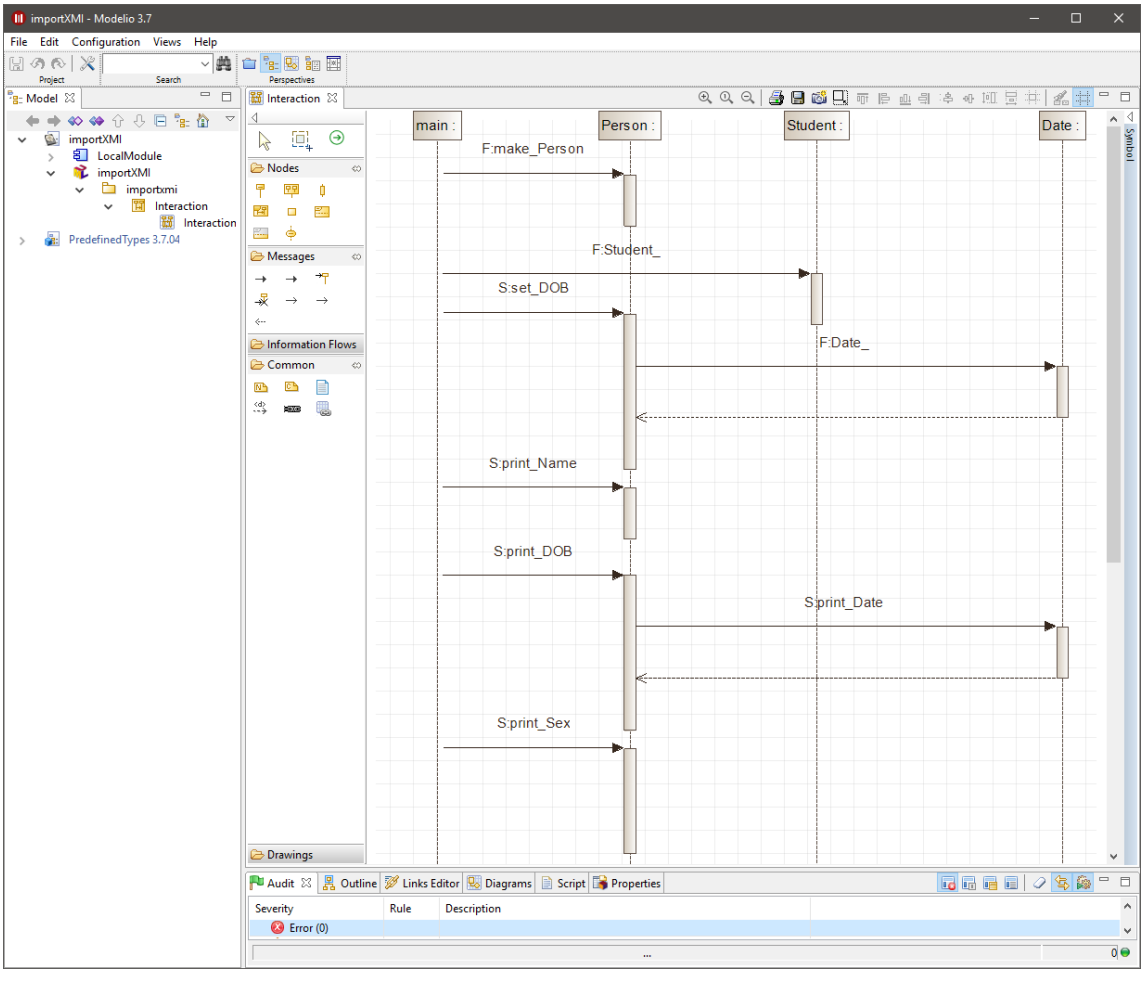
ตารางที่ 4.3 สัญลักษณ์สำหรับแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม (ต่อ)

ชื่อ	สัญลักษณ์	ความหมาย
Activation		แสดงถึง จุดเริ่มต้น จุดสิ้นสุด หรือช่วงเวลาการทำงานที่เกิดขึ้นบนไลฟ์ไลน์
Combined fragment		แสดงถึง เงื่อนไขการทำงานที่เกิดขึ้น

ในการแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมสามารถแสดงได้จากการกดปุ่มมอง (View) แสดงดังรูปที่ 4.20 ซึ่งผู้วิจัยได้ประยุกต์ใช้ซอฟต์แวร์มอเดลลิโอเข้ามาในระบบ โดยผู้ใช้งานไม่จำเป็นต้องติดตั้งซอฟต์แวร์มอเดลลิโอลงบนเครื่องก่อนใช้งาน สำหรับตัวอย่างของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่แสดงผ่านซอฟต์แวร์มอเดลลิโอ แสดงดังรูปที่ 4.21



รูปที่ 4.20 ขั้นตอนการแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม



รูปที่ 4.21 ตัวอย่างของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมแสดงผ่านซอฟต์แวร์มอเดลลิโอ

บทที่ 5

การประเมินผล

ในบทนี้จะเป็นการอธิบายการประเมินผลงานวิจัยโดยใช้การเปรียบเทียบผลลัพธ์ที่ได้มาจากซอฟต์แวร์ ForUML ซึ่งจะอยู่ในรูปของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมกับซอร์สโค้ดภาษาฟอร์แทรนซึ่งเป็นข้อมูลจริงของซอฟต์แวร์แพ็คเกจที่ถูกนำมาทดสอบ ประกอบด้วย 1) จำนวน Type แสดงถึง คลาสทั้งหมดที่ถูกเรียกใช้งาน 2) จำนวน Procedure แสดงถึง เมท็อดทั้งหมดที่มีการเรียกใช้งานเกิดขึ้นภายใน 3) จำนวน Function calls แสดงถึง การเรียกใช้งานของฟังก์ชันทั้งหมด 4) จำนวน Subroutine calls แสดงถึง การเรียกใช้งานของซับรูทีนทั้งหมด 5) จำนวน Statements แสดงถึง เงื่อนไข ทางเลือกที่มีหลายเงื่อนไข และการทำซ้ำ ที่มีการเรียกใช้งานเมท็อดเกิดขึ้นภายใน โดยจะใช้ซอฟต์แวร์ในการทดสอบจำนวน 3 ตัวอย่างด้วยกัน ซึ่งการประเมินการทดสอบของตัวอย่างจะมีการแสดงรายละเอียด 2 ส่วน ดังนี้

(1) ซอร์สโค้ดภาษาฟอร์แทรน เป็นข้อมูลนำเข้าจะแสดงถึงจำนวนส่วนประกอบของซอร์สโค้ดจากระบบทั้ง 5 ส่วน ดังที่กำหนดไว้ข้างต้น

(2) ยูเอ็มแอลซีเควนซ์ไดอะแกรม เป็นข้อมูลผลลัพธ์ที่ได้จากการนำเข้าซอร์สโค้ดภาษาฟอร์แทรน ซึ่งจะแสดงผ่านซอฟต์แวร์มอเดลลิโอ

ในการทดสอบผู้วิจัยได้ทำการประเมินผลการเปรียบเทียบโดยพิจารณาจากการตรวจสอบจำนวนของรายละเอียดของซอร์สโค้ดตามที่ผู้วิจัยกำหนดขึ้นมาด้วยตัวเองก่อน จากนั้นจึงให้ผู้เชี่ยวชาญตรวจสอบความถูกต้องของข้อมูลในภายหลัง ซึ่งผู้วิจัยได้มีการนับจำนวนของซอฟต์แวร์แพ็คเกจที่ถูกนำมาทดสอบแยกตามแต่ละคลาสโดยเริ่มต้นจากคลาสที่เป็นโปรแกรมหลัก จากนั้นจึงเริ่มนับคลาสที่มีความเกี่ยวข้องกับโปรแกรมหลักนั้น คือ คลาสที่มีการเรียกใช้งานจากโปรแกรมหลักโดยเรียงลำดับการเรียกใช้งานภายในซอร์สโค้ดจากบนลงล่าง สำหรับในส่วนของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมผู้วิจัยทำการนับแยกตามแต่ละไลฟ์ไลน์โดยมีความสอดคล้องกับซอร์สโค้ด ดังนี้

(1) จำนวนไลฟ์ไลน์จะสอดคล้องกับจำนวนของ Type ที่แสดงถึงคลาสทั้งหมดที่ถูกเรียกใช้งาน

(2) จำนวนการดำเนินงานของสารบนไลฟ์ไลน์ที่มีการส่งสารไปยังไลฟ์ไลน์อื่นจะสอดคล้องกับจำนวนของ Procedure ที่แสดงถึงเมท็อดทั้งหมดที่มีการเรียกใช้งานเกิดขึ้นภายใน

(3) จำนวนสารบนไลฟ์ไลน์ที่มีการส่งแบบฟังก์ชันจะสอดคล้องกับจำนวนของ Function calls ที่แสดงถึงการเรียกใช้งานของฟังก์ชันทั้งหมด

(4) จำนวนสารบนไลฟ์ไลน์ที่มีการส่งแบบซับรูทีนจะสอดคล้องกับจำนวนของ Subroutine calls ที่แสดงถึงการเรียกใช้งานของซับรูทีนทั้งหมด

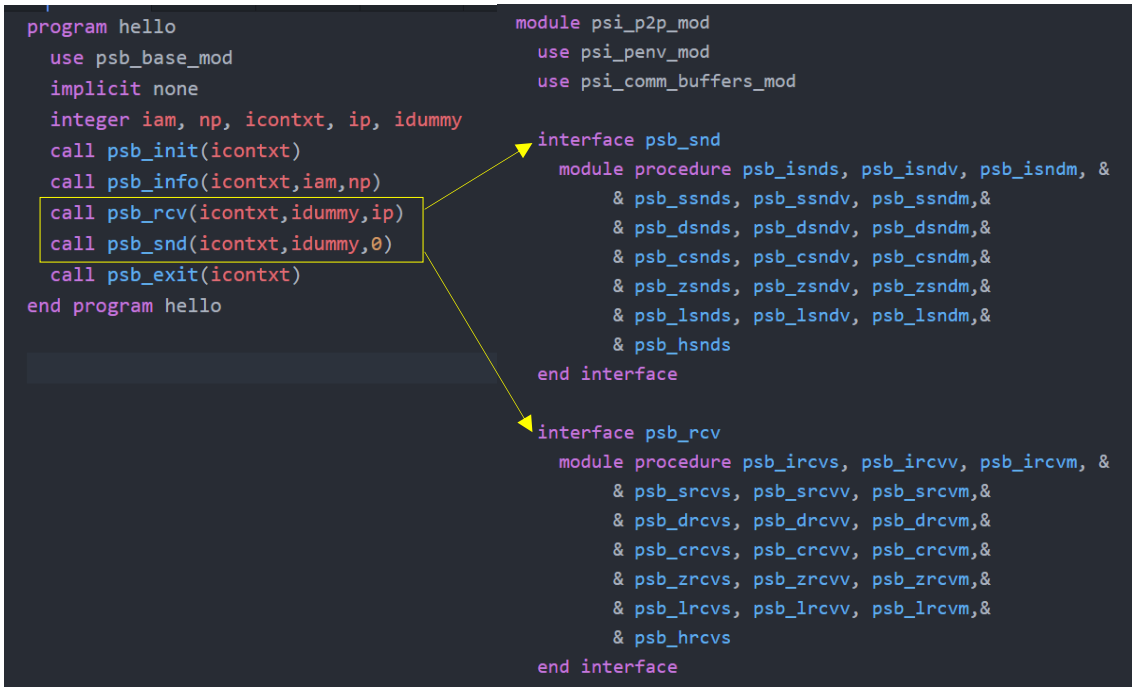
(5) จำนวนของกรอบของแผนภาพบนไลฟ์ไลน์จะสอดคล้องกับจำนวนของ Statements ที่แสดงถึงเงื่อนไข ทางเลือกที่มีหลายเงื่อนไข และการทำซ้ำ ที่มีการเรียกใช้งานเมท็อดเกิดขึ้นภายใน

สำหรับการสรุปผลการทดสอบนั้นจะพิจารณาจากการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยจะมีการคำนวณเป็นจำนวนร้อยละความถูกต้อง

5.1 ซอฟต์แวร์แพ็คเกจทดสอบที่ 1

5.1.1 ซอร์สโค้ดภาษาฟอร์แทรนของระบบ PSBLAS

ในซอฟต์แวร์แพ็คเกจทดสอบที่ 1 นั้น จะมีซอร์สโค้ดซึ่งจะประกอบด้วยคลาสทั้งหมด 8 คลาส ที่มีการเรียกใช้งาน โดยผู้วิจัยได้มีการสร้างคลาส hello ที่เป็นโปรแกรมหลักขึ้นมาเพื่อเรียกการทำงานของซอฟต์แวร์แพ็คเกจดังกล่าว ซึ่งภายในคลาส hello จะมีการเรียกใช้งานซับรูทีนทั้งหมด 5 ซับรูทีน ได้แก่ psb_init psb_info psb_rcv psb_snd และ psb_exit โดยที่ซับรูทีนแต่ละซับรูทีนนั้นจะแสดงถึงฟังก์ชันการทำงานของซอฟต์แวร์แพ็คเกจแต่ละฟังก์ชันการทำงาน กล่าวคือ ซับรูทีนในซอฟต์แวร์แพ็กเกจจะทำหน้าที่เรียกใช้งานอินเตอร์เฟซ จากนั้นภายในอินเตอร์เฟซจะมีการเรียกใช้งานซับรูทีนอีกครั้ง แสดงดังรูปที่ 5.1



รูปที่ 5.1 การเรียกใช้งานอินเตอร์เฟซของโปรแกรมหลัก

จากรูปที่ 5.1 แสดงการเรียกใช้งานอินเตอร์เฟซของโปรแกรมหลัก โดยภายใต้อินเตอร์เฟซของ psb_snd และ psb_rcv จะมีการเรียกหลายเมท็อด ซึ่งเมื่อทำการนับข้อมูลจะทำให้จำนวนการส่งสารของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมมีมากกว่าการเรียกใช้งานของซอร์สโค้ดภาษาฟอร์แทรน เนื่องจากโปรแกรมหลักจะมีการเรียกใช้งานไปยังอินเตอร์เฟซก่อน จากนั้นอินเตอร์เฟซจะเรียกใช้งานเมท็อดที่อยู่ภายใต้อีกครั้งหนึ่ง ซึ่งในส่วนของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมนั้น จะแสดงผลลัพธ์โดยใช้เมท็อดภายใต้อินเตอร์เฟซมาแทนที่ในส่วนของการเรียกใช้งานเดิมทำให้ได้ข้อมูลที่มากกว่า ดังผลลัพธ์ที่แสดงในตารางที่ 5.1 ของคลาส hello โดยจะมีจำนวนของ Subroutine call ของซอร์สโค้ดภาษาฟอร์แทรนเพียงแค่ 5 การเรียกใช้งาน แต่ในส่วนของการแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมจะมีการส่งสารที่เป็นซับรูทีนทั้งหมด 100 สาร ดังนั้นผู้วิจัยจึงมีการกำหนดเมื่อมีการส่งสารแบบอินเตอร์เฟซจะทำการตรวจสอบจากส่วนของซอร์สโค้ดภาษาฟอร์แทรนเป็นหลัก โดยถ้าส่วนของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมมีค่ามากกว่าหรือเท่ากับส่วนของซอร์สโค้ดแสดงว่ามีความครอบคลุมที่ 100%

ตารางที่ 5.1 ผลการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ PSBLAS

Type	Procedure	Function call	Subroutine call	Statements
hello	0/0	0/0	100/5	0/0
psb_init	1/1	0/0	3/3	1/1
psb_info	0/0	0/0	0/0	0/0
psb_rcv	28/28	0/0	28/28	0/0
psb_snd	0/0	0/0	0/0	0/0
psb_errstack	0/0	0/0	0/0	0/0
psb_exit	1/1	0/0	2/2	1/1
psb_buffer_node	0/0	0/0	0/0	0/0
Overall	30/30 100%	0/0 100%	133/38 100%	2/2 100%

จากตารางที่ 5.1 แสดงผลการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ PSBLAS โดยข้อมูลในตารางจะแสดงจำนวนส่วนของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่นับได้ต่อจำนวนส่วนของซอร์สโค้ดภาษาฟอร์แทรนที่นับได้ ซึ่งจะประกอบด้วยคลาสทั้งหมด 8 คลาส ได้แก่

(1) hello เป็นโปรแกรมหลัก ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีน จำนวน 5 ซับรูทีน ในซอร์สโค้ดภาษาฟอร์แทรนแต่มีการแสดงผลในยูเอ็มแอลซีเควนซ์ไดอะแกรม จำนวน 100 ซับรูทีน เนื่องจากเป็นการเรียกใช้งานแบบอินเตอร์เฟซ และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(2) psb_init เป็นอินเตอร์เฟซที่ถูกเรียกใช้ มีเมท็อดจำนวน 1 เมท็อด ที่ถูกเรียกใช้งานและมีการเรียกใช้งานเมท็อดของคลาสอื่นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีน จำนวน 3 ซับรูทีน และมีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข จำนวน 1 เงื่อนไข

(3) psb_info เป็นอินเตอร์เฟซที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(4) psb_rcv เป็นอินเตอร์เฟซที่ถูกเรียกใช้ มีเมท็อดจำนวน 28 เมท็อด ที่ถูกเรียกใช้งานและมีการเรียกใช้งานเมท็อดของคลาสอื่นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีน จำนวน 28 ซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(5) psb_snd เป็นอินเตอร์เฟซที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

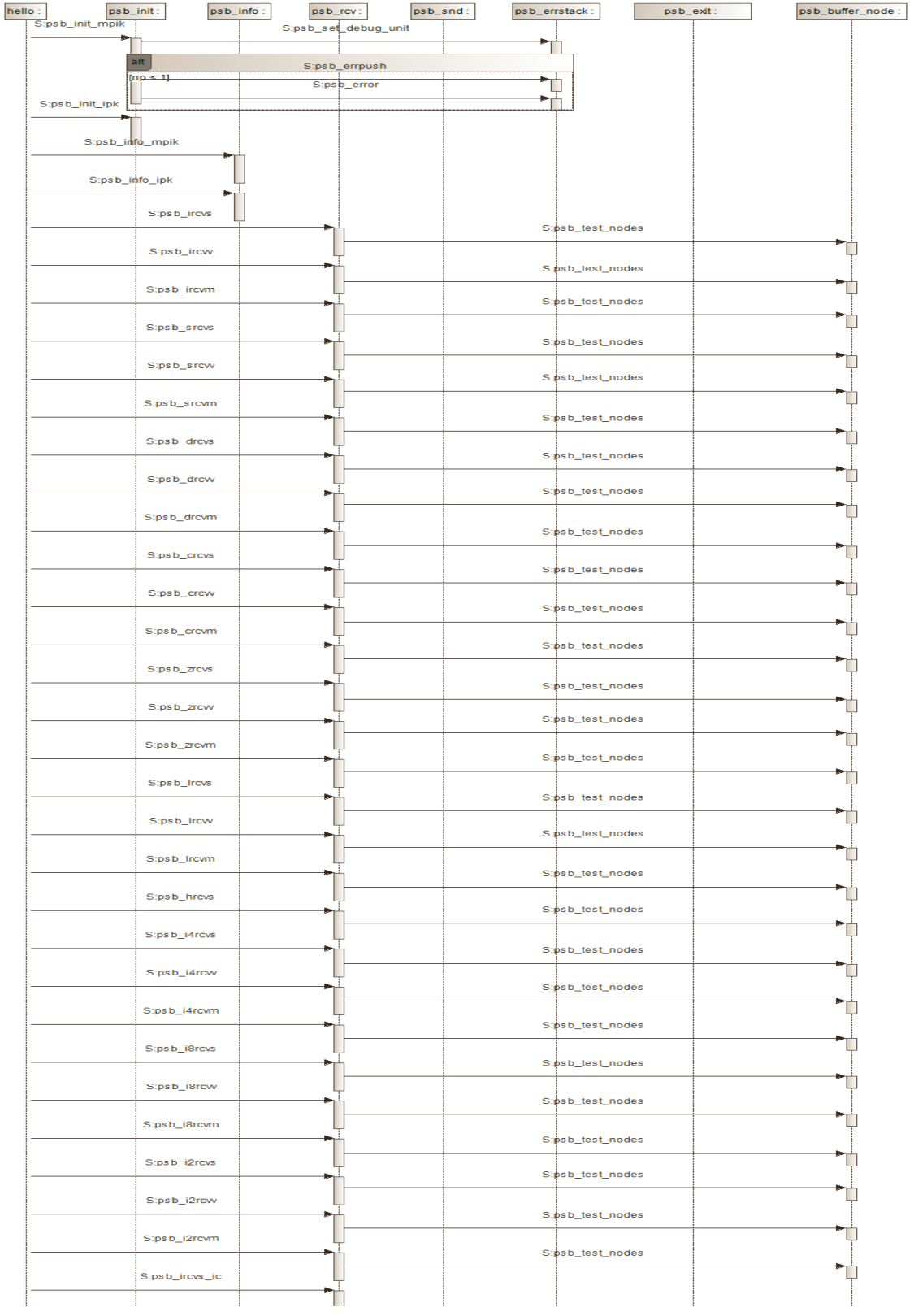
(6) psb_errstack เป็นคลาสที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(7) psb_exit เป็นอินเตอร์เฟซที่ถูกเรียกใช้ มีเมท็อดจำนวน 1 เมท็อด ที่ถูกเรียกใช้งานและมีการเรียกใช้งานเมท็อดของคลาสอื่นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีน จำนวน 2 ซับรูทีน และมีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข จำนวน 1 เงื่อนไข

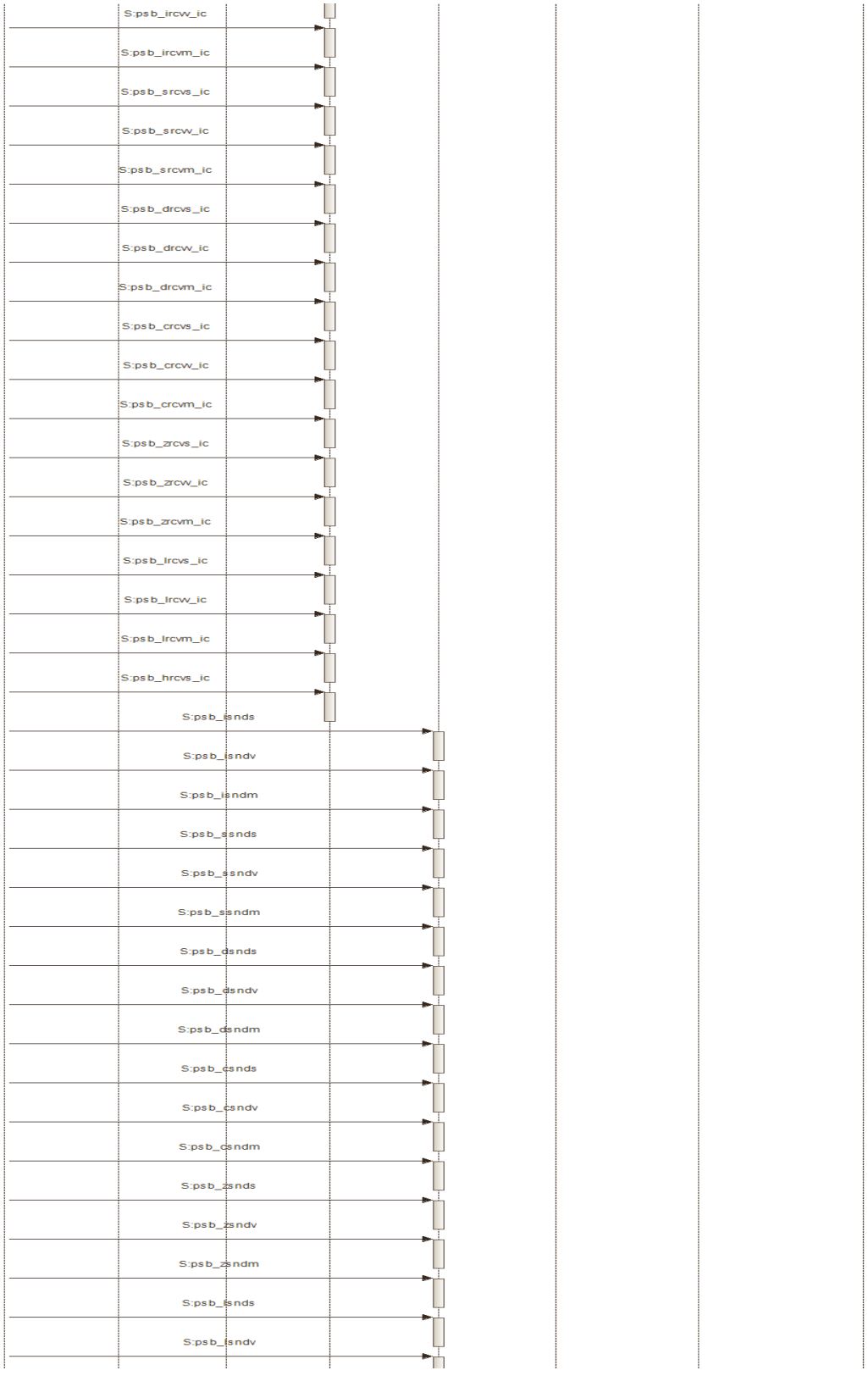
(8) psb_buffer_node เป็นคลาสที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

5.1.2 ยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ PSBLAS

ในยูเอ็มแอลซีเควนซ์ไดอะแกรมของซอฟต์แวร์แพ็คเกจทดสอบที่ 1 จะประกอบด้วยไลฟ์ไลน์ทั้งหมด 8 ไลฟ์ไลน์ แสดงดังรูปที่ 5.2 ซึ่งในส่วนของไลฟ์ไลน์ hello จะมีการส่งสารที่เป็นซับรูทีนจำนวน 100 สารไปยังไลฟ์ไลน์อื่นเพื่อเรียกใช้งาน ในส่วนของไลฟ์ไลน์ psb_init จะมีจำนวนเมท็อด 1 เมท็อด ที่ถูกเรียกใช้งานและมีการส่งสารที่เป็นซับรูทีน จำนวน 3 สารไปยังไลฟ์ไลน์อื่น ในส่วนของไลฟ์ไลน์ psb_rcv จะมี จำนวนเมท็อด 28 เมท็อด ที่ถูกเรียกใช้งานและมีการส่งสารที่เป็นซับรูทีน จำนวน 28 สารไปยังไลฟ์ไลน์อื่น ในส่วนของไลฟ์ไลน์ psb_exit จะมีจำนวนเมท็อด 1 เมท็อด ที่ถูกเรียกใช้งานและมีการส่งสารที่เป็นซับรูทีน จำนวน 2 สารไปยังไลฟ์ไลน์อื่น สำหรับในส่วนของไลฟ์ไลน์ psb_info psb_snd psb_errstack และ psb_buffer_node จะไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้น ดังนั้นจึงไม่มีการส่งสารไปยังไลฟ์ไลน์อื่น



รูปที่ 5.2 ผลลัพธ์แผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ PSBLAS



รูปที่ 5.2 ผลลัพธ์แผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ PSBLAS (ต่อ)

รูปที่ 5.2 ผลลัพธ์แผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ PSBLAS (ต่อ)

5.1.3 สรุปผล

ผลลัพธ์ที่ได้จากการนำซอร์สโค้ดภาษาฟอร์แทรนและยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบมาเปรียบเทียบกัน คือ ซอร์สโค้ดภาษาฟอร์แทรนของระบบ PSBLAS สามารถแปลงเป็นข้อมูลในรูปของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมได้ครบถ้วน กล่าวคือ สามารถตรวจพบสัญลักษณ์ของยูเอ็มแอลซีเควนซ์ไดอะแกรมและข้อมูลของซอร์สโค้ดภาษาฟอร์แทรนที่ตรงกัน แต่ในส่วนของคลาส hello จะมีข้อมูลเพิ่มขึ้นมาเนื่องจากเป็นการเรียกใช้งานแบบอินเตอร์เฟซ

5.2 ซอฟต์แวร์แพ็คเกจทดสอบที่ 2

5.2.1 ซอร์สโค้ดภาษาฟอร์แทรนของระบบ MLD2P4

ในซอฟต์แวร์แพ็คเกจทดสอบที่ 2 นั้น จะมีซอร์สโค้ดซึ่งจะประกอบด้วยคลาสทั้งหมด 10 คลาส ที่มีการเรียกใช้งาน โดยผู้วิจัยได้มีการสร้างคลาส mld_dexample_1lev ที่เป็นโปรแกรมหลักขึ้นมาเพื่อเรียกการทำงานของซอฟต์แวร์แพ็คเกจดังกล่าว ซึ่งภายในคลาส mld_dexample_1lev จะมีการเรียกใช้งานซับรูทีนทั้งหมดจำนวน 7 ซับรูทีน ได้แก่ psb_init psb_info psb_barrier psb_amx psb_exit เป็นซับรูทีนที่เรียกใช้งานอินเตอร์เฟซ และ psb_set_errverbosity psb_errpush เป็นซับรูทีนที่เรียกใช้งานเมท็อด และจะมีการเรียกใช้งานฟังก์ชันทั้งหมดจำนวน 2 ฟังก์ชัน ได้แก่ psb_genrm2 และ psb_geamax ซึ่งเป็นฟังก์ชันที่เรียกใช้งานอินเตอร์เฟซ

ตารางที่ 5.2 ผลการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ MLD2P4

Type	Procedure	Function call	Subroutine call	Statements
mld_dexample_1lev	0/0	6/2	29/7	1/1
psb_init	1/1	0/0	3/3	1/1
psb_info	0/0	0/0	0/0	0/0
psb_barrier	0/0	0/0	0/0	0/0
psb_errstack	0/0	0/0	0/0	0/0
psb_amx	0/0	0/0	0/0	0/0

ตารางที่ 5.2 ผลการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ MLD2P4 (ต่อ)

Type	Procedure	Function call	Subroutine call	Statements
psb_exit	1/1	0/0	2/2	1/1
psb_buffer_node	0/0	0/0	0/0	0/0
psb_genrm2	0/0	0/0	0/0	0/0
psb_geamax	0/0	0/0	0/0	0/0
Overall	2/2 100%	6/2 100%	34/12 100%	3/3 100%

จากตารางที่ 5.2 แสดงผลการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ MLD2P4 โดยข้อมูลในตารางจะแสดงจำนวนส่วนของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่นับได้ต่อจำนวนส่วนของซอร์สโค้ดภาษาฟอร์แทรนที่นับได้ ซึ่งจะประกอบด้วยคลาสทั้งหมด 10 คลาส ได้แก่

(1) mld_dexample_1lev เป็นโปรแกรมหลัก ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน มีการเรียกใช้งานแบบฟังก์ชัน จำนวน 2 ฟังก์ชัน ในซอร์สโค้ดภาษาฟอร์แทรนแต่มีการแสดงผลในยูเอ็มแอลซีเควนซ์ไดอะแกรม จำนวน 6 ฟังก์ชัน เนื่องจากเป็นการเรียกใช้งานแบบอินเตอร์เฟซ มีการเรียกใช้งานแบบซับรูทีน จำนวน 7 ซับรูทีน ในซอร์สโค้ดภาษาฟอร์แทรนแต่มีการแสดงผลในยูเอ็มแอลซีเควนซ์ไดอะแกรม จำนวน 29 ซับรูทีน เนื่องจากมีการเรียกใช้งานแบบอินเตอร์เฟซและมีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข จำนวน 1 เงื่อนไข

(2) psb_init เป็นอินเตอร์เฟซที่ถูกเรียกใช้ มีเมท็อดจำนวน 1 เมท็อด ที่ถูกเรียกใช้งานและมีการเรียกใช้งานเมท็อดของคลาสอื่นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีน จำนวน 3 ซับรูทีน และมีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข จำนวน 1 เงื่อนไข

(3) psb_info เป็นอินเตอร์เฟซที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(4) psb_barrier เป็นอินเตอร์เฟซที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(5) psb_errstack เป็นคลาสที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(6) psb_amx เป็นอินเตอร์เฟซที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(7) psb_exit เป็นอินเตอร์เฟซที่ถูกเรียกใช้ มีเมท็อดจำนวน 1 เมท็อด ที่ถูกเรียกใช้งานและมีการเรียกใช้งานคลาสอื่นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีน จำนวน 2 ซับรูทีน และมีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข จำนวน 1 เงื่อนไข

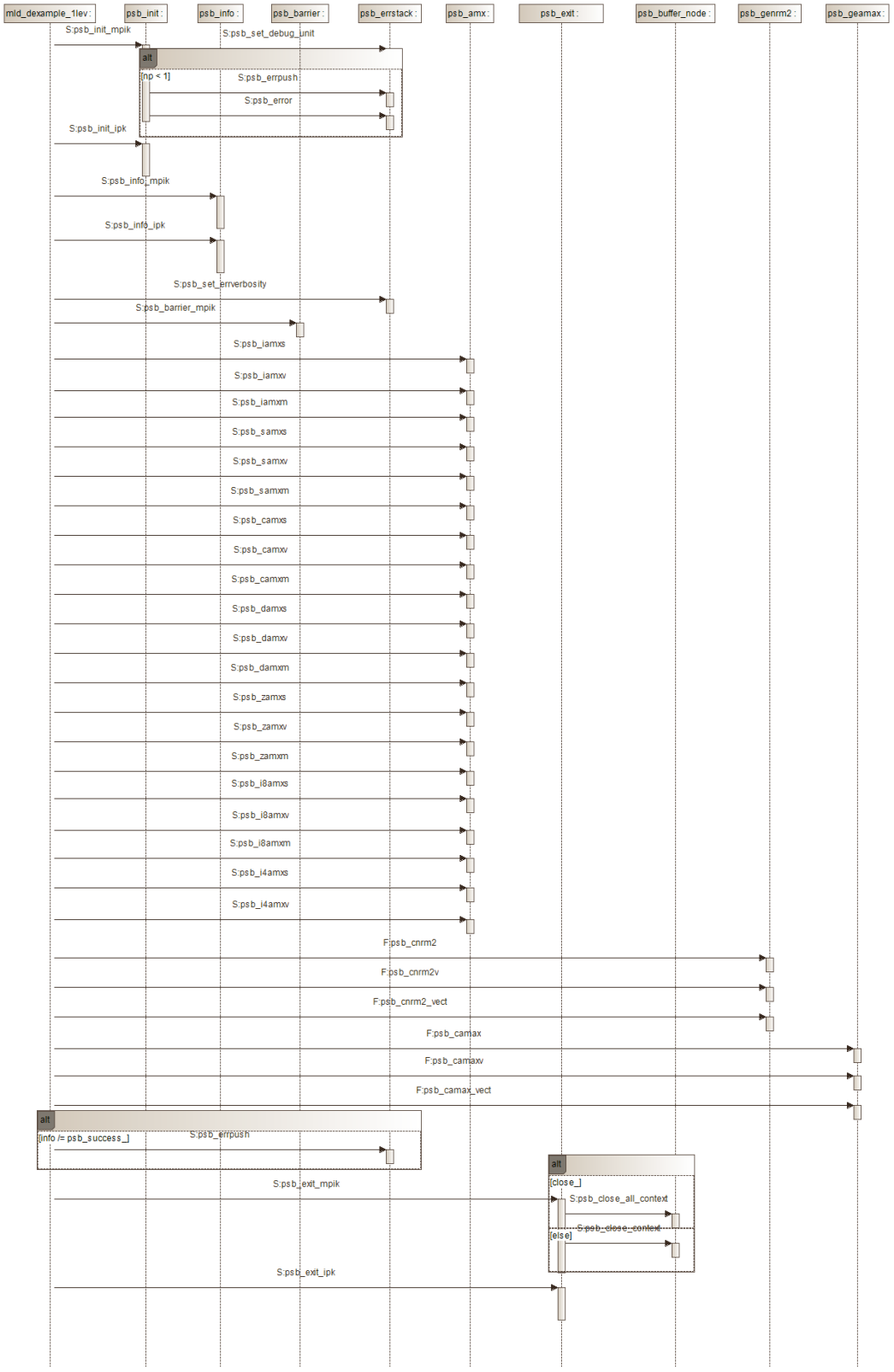
(8) psb_buffer_node เป็นคลาสที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(9) เป็นอินเตอร์เฟซที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(10) เป็นอินเตอร์เฟซที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

5.2.2 ยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ MLD2P4

ในยูเอ็มแอลซีเควนซ์ไดอะแกรมของซอฟต์แวร์แพ็คเกจทดสอบที่ 2 ประกอบด้วยไลฟ์ไลน์ทั้งหมด 10 ไลฟ์ไลน์ แสดงดังรูปที่ 5.3 ซึ่งในส่วนของไลฟ์ไลน์ mld_dexample_1lev จะมีการส่งสารที่เป็นฟังก์ชัน จำนวน 6 สารไปยังไลฟ์ไลน์อื่น เพื่อเรียกใช้งานและมีการส่งสารที่เป็นซับรูทีน จำนวน 29 สารไปยังไลฟ์ไลน์อื่นเพื่อเรียกใช้งาน ในส่วนของไลฟ์ไลน์ psb_init จะมีจำนวนเมท็อด 1 เมท็อด ที่ถูกเรียกใช้งานและมีการส่งสารที่เป็นซับรูทีน จำนวน 3 สารไปยังไลฟ์ไลน์อื่น ในส่วนของไลฟ์ไลน์ psb_exit จะมีจำนวนเมท็อด 1 เมท็อด ที่ถูกเรียกใช้งานและมีการส่งสารที่เป็นซับรูทีน จำนวน 2 สารไปยังไลฟ์ไลน์อื่น สำหรับในส่วนของไลฟ์ไลน์ psb_info psb_barrier psb_errstack psb_amx psb_buffer_node psb_genrm2 และ psb_geamax จะไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้น ดังนั้นจึงไม่มีการส่งสารไปยังไลฟ์ไลน์อื่น



รูปที่ 5.3 ผลลัพธ์แผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ MLD2P4

5.2.3 สรุปผล

ผลลัพธ์ที่ได้จากการนำซอร์สโค้ดภาษาฟอร์แทรนและยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบมาเปรียบเทียบกัน คือ ซอร์สโค้ดภาษาฟอร์แทรนของระบบ MLD2P4 สามารถแปลงเป็นข้อมูลในรูปของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมได้ครบถ้วน กล่าวคือ สามารถตรวจพบสัญลักษณ์ของยูเอ็มแอลซีเควนซ์ไดอะแกรมและข้อมูลของซอร์สโค้ดภาษาฟอร์แทรนที่ตรงกัน แต่ในส่วนของคลาส mld_dexample_1lev จะมีข้อมูลเพิ่มขึ้นมาเนื่องจากเป็นการเรียกใช้งานแบบอินเตอร์เฟซ

5.3 ซอฟต์แวร์แพ็คเกจทดสอบที่ 3

5.3.1 ซอร์สโค้ดภาษาฟอร์แทรนของระบบ ForTrilinos

ในซอฟต์แวร์แพ็คเกจทดสอบที่ 3 นั้น จะมีซอร์สโค้ดซึ่งจะประกอบด้วยคลาสทั้งหมด 3 คลาส ที่มีการเรียกใช้งาน โดยผู้วิจัยได้มีการสร้างคลาส test_TpetraCrsGraph ที่เป็นโปรแกรมหลักขึ้นมาเพื่อเรียกการทำงานของซอฟต์แวร์แพ็คเกจดังกล่าว ซึ่งภายในคลาส test_TpetraCrsGraph จะมีการเรียกใช้งานซับรูทีนทั้งหมด จำนวน 3 ซับรูทีน ได้แก่ createMPI create และ release จะเป็นตัวแปรที่ใช้อ้างอิงถึงซับรูทีนของคลาสที่ถูกเรียกใช้งาน ซึ่งสามารถมีได้มากกว่า 1 ซับรูทีน

ตารางที่ 5.3 ผลการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ ForTrilinos

Type	Procedure	Function call	Subroutine call	Statements
test_TpetraCrsGraph	0/0	0/0	5/3	1/1
TeuchosComm	5/5	4/4	1/1	0/0
C_PTR	0/0	0/0	0/0	0/0
Overall	5/5 100%	4/4 100%	6/4 100%	1/1 100%

จากตารางที่ 5.3 แสดงผลการเปรียบเทียบความแตกต่างระหว่างซอร์สโค้ดภาษาฟอร์แทรนกับยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ ForTrilinos โดยข้อมูลในตารางจะแสดงจำนวนส่วนของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่นับได้ต่อจำนวนส่วนของซอร์สโค้ดภาษาฟอร์แทรนที่นับได้ ซึ่งจะประกอบด้วยคลาสทั้งหมด 3 คลาส ได้แก่

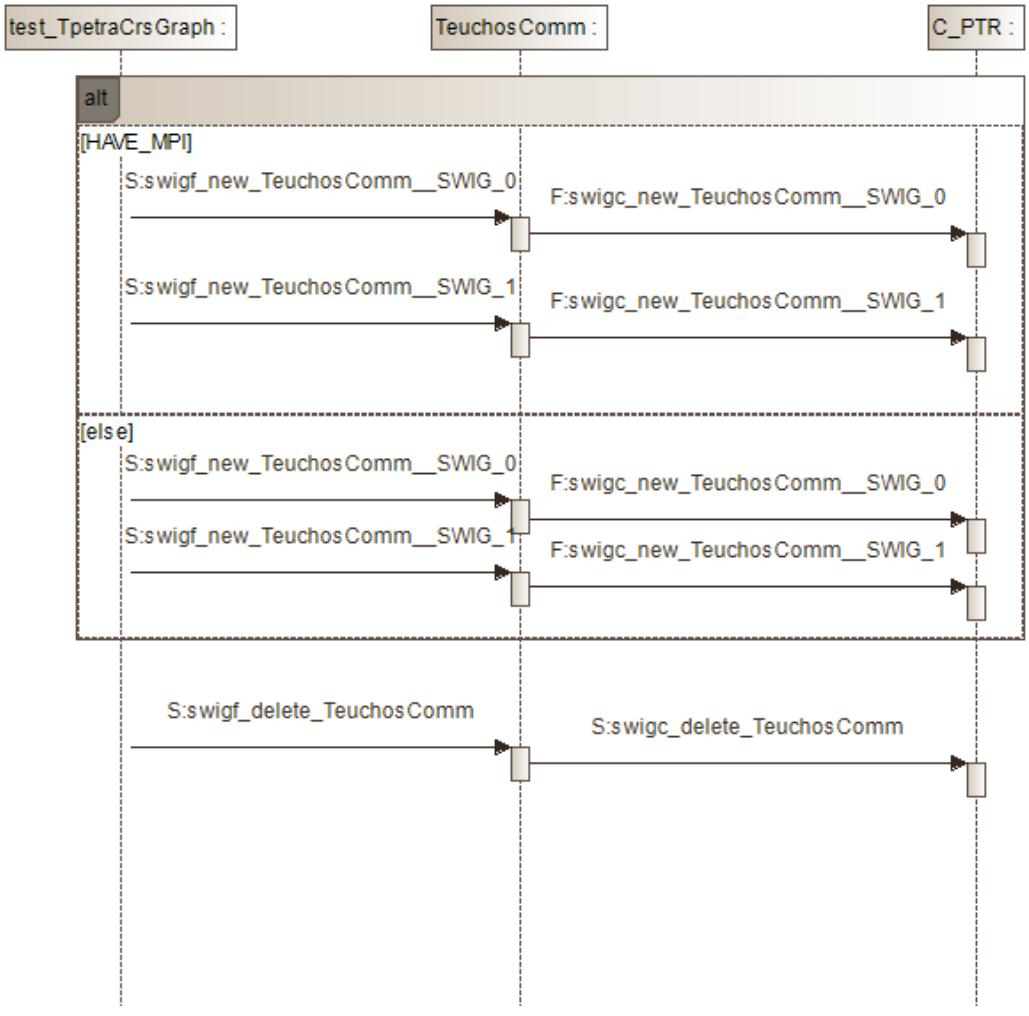
(1) test_TpetraCrsGraph เป็นโปรแกรมหลัก ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีน จำนวน 3 ซับรูทีน ในซอร์สโค้ดภาษาฟอร์แทรนแต่มีการแสดงผลในยูเอ็มแอลซีเควนซ์ไดอะแกรม จำนวน 5 ซับรูทีน เนื่องจากมีการเรียกใช้ตัวแปรที่ใช้อ้างอิงถึงซับรูทีน และมีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข จำนวน 1 เงื่อนไข

(2) TeuchosComm เป็นคลาสที่ถูกเรียกใช้งานเมท็อดผ่านตัวแปล มีเมท็อดจำนวน 5 เมท็อด ที่ถูกเรียกใช้งาน และมีการเรียกใช้งานเมท็อดของคลาสอื่นภายใน มีการเรียกใช้งานแบบฟังก์ชันจำนวน 4 ฟังก์ชัน มีการเรียกใช้งานแบบซับรูทีนจำนวน 1 ซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

(3) C_PTR เป็นคลาสที่ถูกเรียกใช้ ไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้นภายใน ไม่มีการเรียกใช้งานแบบฟังก์ชัน ไม่มีการเรียกใช้งานแบบซับรูทีน และไม่มีการเรียกใช้งานเกิดขึ้นภายใต้เงื่อนไข

5.3.2 ยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ ForTrilinos

ในยูเอ็มแอลซีเควนซ์ไดอะแกรมของซอฟต์แวร์แพ็คเกจทดสอบที่ 3 จะประกอบด้วยไลฟ์ไลน์ทั้งหมด 3 ไลฟ์ไลน์ แสดงดังรูปที่ 5.4 ซึ่งในส่วนของไลฟ์ไลน์ test_TpetraCrsGraph จะมีการส่งสารที่เป็นซับรูทีน จำนวน 5 สารไปยังไลฟ์ไลน์อื่นเพื่อเรียกใช้งาน ในส่วนของไลฟ์ไลน์ TeuchosComm จะมีจำนวนเมท็อด 5 เมท็อดที่ถูกเรียกใช้งาน มีการส่งสารที่เป็นฟังก์ชัน จำนวน 4 สารไปยังไลฟ์ไลน์อื่นเพื่อเรียกใช้งาน และมีการส่งสารที่เป็นซับรูทีน จำนวน 1 สารไปยังไลฟ์ไลน์อื่นเพื่อเรียกใช้งาน สำหรับในส่วนของไลฟ์ไลน์ C_PTR จะไม่มีเมท็อดที่มีการเรียกใช้งานเกิดขึ้น ดังนั้นจึงไม่มีการส่งสารไปยังไลฟ์ไลน์อื่น



รูปที่ 5.4 ผลลัพธ์แผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบ ForTrilinos

5.3.3 สรุปผล

ผลลัพธ์ที่ได้จากการนำซอร์สโค้ดภาษาฟอร์แทรนและยูเอ็มแอลซีเควนซ์ไดอะแกรมของระบบมาเปรียบเทียบกัน คือ ซอร์สโค้ดภาษาฟอร์แทรนของระบบ ForTrilinos สามารถแปลงเป็นข้อมูลในรูปของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมได้ครบถ้วน กล่าวคือ สามารถตรวจพบสัญลักษณ์ของยูเอ็มแอลซีเควนซ์ไดอะแกรมและข้อมูลของซอร์สโค้ดภาษาฟอร์แทรนที่ตรงกัน แต่ในส่วนของคลาส test_TpetraCrsGraph จะมีข้อมูลเพิ่มขึ้นมาเนื่องจากเป็นการเรียกใช้งานตัวแปรที่ใช้อ้างอิงถึงซับรูทีนของคลาส

บทที่ 6

อภิปรายและสรุปผลการวิจัย

6.1 อภิปรายผลการวิจัย

การอภิปรายผลของงานวิจัยนี้มาจากการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมและการประเมินผลของเครื่องมือที่ผู้วิจัยได้พัฒนาขึ้น

จากการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรม ผลลัพธ์ที่ได้แสดงให้เห็นว่า กฎการแปลงมีความครอบคลุมในทุกส่วนของการแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยที่ภาษาฟอร์แทรนนั้นจะเป็นภาษาเชิงวัตถุที่แตกต่างจากภาษาเชิงวัตถุโดยทั่วไป เช่น การจัดการหน่วยความจำ (Garbage Collection) เพื่อจัดการกับอ็อบเจกต์ของภาษาฟอร์แทรนจะต้องมีการเขียนโค้ดขึ้นมาเพื่อจัดสรรการทำงานเอง แต่ในภาษาเชิงวัตถุภาษาอื่นนั้น เช่น ภาษาจาวาจะมีการจัดสรรโดยอัตโนมัติ ในเรื่องของเมท็อดในภาษาฟอร์แทรนจะมีการแบ่งการทำงานโดยแยกเป็นฟังก์ชันและซับรูทีน แต่ในภาษาจาวาจะไม่มีการแบ่งการทำงานของเมท็อด ซึ่งจากข้อแตกต่างในเรื่องของภาษาเชิงวัตถุนี้ผู้วิจัยจึงทำการออกแบบกฎการแปลงให้เข้ากับภาษาฟอร์แทรน เช่น ในส่วนของการแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมผู้วิจัยได้มีการออกแบบชื่อของสารโดยถ้าสารมีการเรียกใช้งานแบบฟังก์ชันจะแสดงสัญลักษณ์ F ไว้หน้าชื่อและถ้าหากสารมีการเรียกใช้งานแบบซับรูทีนจะแสดงสัญลักษณ์ S ไว้หน้าชื่อ สำหรับการสร้างกฎการแปลงขึ้นมาด้วยภาษา ATL นั้นจะทำให้กฎที่ได้มีความน่าเชื่อถือ เนื่องจากภาษา ATL เป็นภาษาที่นิยมใช้ในการแปลงแบบจำลอง (Li, et al., 2014; Rhazali, et al., 2016; Srai, et al., 2017) ซึ่งถ้าหากปรับเปลี่ยนไปใช้วิธีการอื่นในการสร้างกฎการแปลงขึ้นมา กฎการแปลงหรือผลลัพธ์ที่ได้จะไม่มีความแตกต่างไปจากเดิม เนื่องจากภาษาที่ใช้ในการแปลงนั้น

จะเป็นเพียงส่วนที่ใช้จับคู่แบบจำลองระหว่างกันเพื่อตรวจสอบความถูกต้องของกฎการแปลง ซึ่งสุดท้ายจะต้องมีผู้เชี่ยวชาญเป็นคนตรวจสอบความถูกต้องของกฎการแปลงอีกในครั้งภายหลัง

จากการทดสอบซอฟต์แวร์แพ็คเกจทั้ง 3 ซอฟต์แวร์แพ็คเกจ ผลลัพธ์ที่ได้จากการทดสอบแสดงให้เห็นถึงการเรียกใช้งานของคลาสที่แตกต่างไปจากการเรียกใช้งานแบบฟังก์ชันหรือซับรูทีน นั้นคือการเรียกใช้งานแบบอินเตอร์เฟซ ซึ่งก่อนการเริ่มต้นการพัฒนาเครื่องมือขึ้นมานั้นผู้วิจัยได้ออกแบบระบบให้รองรับการเรียกใช้งานกันระหว่างคลาสเท่านั้น แต่จากการทดสอบระบบพบว่าซอฟต์แวร์แพ็คเกจโดยส่วนใหญ่นั้นจะมีการสร้างอินเตอร์เฟซขึ้นมาเป็นตัวกำหนดฟังก์ชันการทำงานต่าง ๆ เพื่อความสะดวกต่อผู้ใช้ในการเรียกใช้งาน ซึ่งโดยส่วนใหญ่การเรียกใช้งานแบบอินเตอร์เฟซจะเกิดขึ้นในส่วนของการเรียกใช้งานจากโปรแกรมหลัก ดังนั้นผู้วิจัยได้พัฒนาส่วนที่เกี่ยวข้องกับการเรียกใช้งานแบบอินเตอร์เฟซขึ้นมาทำให้ผลลัพธ์ของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่แสดงขึ้นมานั้นมีความครบถ้วนสมบูรณ์มากยิ่งขึ้น ในการประเมินผลของเครื่องมือนั้นผู้วิจัยไม่ได้เปรียบเทียบฟังก์ชันการทำงานของเครื่องมือที่พัฒนาขึ้นมากับเครื่องมือชนิดอื่นเนื่องจากเครื่องมือที่พัฒนาขึ้นมานั้นเป็นเครื่องมือเดียวที่สามารถใช้สำหรับแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมได้ สำหรับในส่วนของการแสดงผลของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมาผ่านซอฟต์แวร์มอเดลลิโอ นั้นจะทำให้ผู้ใช้งานทำความเข้าใจและวิเคราะห์ระบบได้ง่ายเนื่องจากยูเอ็มแอลไดอะแกรมที่ได้นั้นเป็นไปตามมาตรฐานของยูเอ็มแอลเวอร์ชัน 2.1 (Felderer and Herrmann, 2018; Jiang, et al., 2017; Salah, et al., 2016) อีกทั้งผู้ใช้งานสามารถแก้ไขปรับเปลี่ยนไดอะแกรมได้ตามต้องการ ซึ่งจะเป็นประโยชน์ต่อการวิเคราะห์และปรับปรุงระบบในอนาคต

จากการประเมินผลความถูกต้องของเครื่องมือซึ่งผู้วิจัยได้ทำการประเมินผลความถูกต้องขึ้นมาด้วยตัวเองก่อน จากนั้นจึงให้ผู้เชี่ยวชาญตรวจสอบความถูกต้องของข้อมูลในภายหลัง พบว่าผู้เชี่ยวชาญมีความคิดเห็นที่สอดคล้องกับผลลัพธ์จากการประเมินที่ผู้วิจัยได้นำเสนอ โดยผู้เชี่ยวชาญได้มีข้อเสนอแนะในเรื่องของการแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่เป็นส่วนของการเรียกใช้งานแบบอินเตอร์เฟซขึ้นมา สืบเนื่องจากเริ่มต้นผู้วิจัยไม่ได้มีการออกแบบระบบให้รองรับการเรียกใช้งานแบบอินเตอร์เฟซ ดังนั้นผู้วิจัยจึงได้เพิ่มส่วนของการเรียกใช้งานแบบอินเตอร์เฟซขึ้นมา และมีข้อเสนอแนะเกี่ยวกับการแยกประเภทของสารที่แสดงในแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยผู้วิจัยได้มีการปรับการแสดงผลของแผนภาพคือ สารที่เป็นฟังก์ชันจะแสดงสัญลักษณ์ F ไว้หน้าชื่อ และสารที่เป็นซับรูทีนจะแสดงสัญลักษณ์ S ไว้หน้าชื่อ

6.2 ข้อจำกัดและข้อเสนอแนะของงานวิจัย

ในการศึกษางานวิจัยนี้ผู้วิจัยมีข้อจำกัดและข้อเสนอแนะของงานวิจัยแก่นักวิจัยหรือผู้ที่สนใจในด้านของวิศวกรรมซอฟต์แวร์ สำหรับข้อจำกัดที่เกิดขึ้นจากการวิจัยมีดังนี้

(1) ระบบรองรับการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมโดยใช้เอกสารเอกซ์เอ็มไอตามข้อกำหนดของยูเอ็มแอลเวอร์ชัน 2.1 ซึ่งระบบไม่สามารถเปิดใช้กับเอกสารเอกซ์เอ็มไอเวอร์ชันต่ำกว่า 2.1 แต่สามารถเปิดใช้กับเอกสารเอกซ์เอ็มไอเวอร์ชันสูงกว่า 2.1 ได้

(2) ระบบรองรับซอร์สโค้ดภาษาฟอร์แทรนที่เป็นภาษาเชิงวัตถุตั้งแต่เวอร์ชัน 2003 ขึ้นไป ซึ่งระบบไม่รองรับซอร์สโค้ดภาษาฟอร์แทรนที่เป็นเวอร์ชันต่ำกว่า 2003 โดยถ้าหากมีการนำเข้าซอร์สโค้ดภาษาฟอร์แทรนที่เป็นเวอร์ชันต่ำกว่า 2003 ระบบจะทำการแจ้งเตือนข้อผิดพลาดขึ้นภายในหน้าต่าง status/log และระบบจะไม่มีการประมวลผลซอร์สโค้ดดังกล่าว

(3) การแสดงผลของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมาผ่านซอฟต์แวร์มอเดลลิโอนั้นจะมีความซับซ้อนมากขึ้นถ้าหากระบบมีขนาดใหญ่ และจะทำให้เกิดความล่าช้าในการแสดงผลแผนภาพ อย่างไรก็ตามแผนภาพที่ได้จะแสดงผลได้ถูกต้องตรงกับระบบที่นำเข้า โดยถ้าหากแผนภาพมีขนาดเกินขอบเขตของหน้าจอผู้ใช้งานสามารถเลื่อนแผนภาพเพื่อแสดงส่วนที่เกินขอบเขตนั้นได้ ซึ่งจะช่วยให้ผู้ใช้งานมีความสะดวกรวดเร็วในการวิเคราะห์

(4) ในงานวิจัยนี้ผู้วิจัยได้ทำการออกแบบกฎการแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมา ซึ่งในการออกแบบนั้นจะต้องคำนึงถึงเมต้าโมเดลของซอร์สโค้ดภาษาฟอร์แทรนและเมต้าโมเดลของยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นหลัก เนื่องจากเมต้าโมเดลของทั้งคู่เป็นส่วนสำคัญที่ใช้สำหรับการแปลงแบบจำลอง ดังนั้นผู้วิจัยจึงทำการศึกษาเครื่องมือที่สามารถออกแบบและตรวจสอบความถูกต้องของกฎการแปลง โดยผู้วิจัยได้เลือกใช้ภาษา ATL ซึ่งเป็นภาษาที่ใช้ในการแปลงแบบจำลอง ซึ่งถือเป็นเครื่องมือที่ช่วยตรวจสอบถึงความถูกต้องของกฎการแปลงได้ ทำให้กฎการแปลงที่ได้มีการออกแบบเป็นไปตามมาตรฐานหรือข้อกำหนดของการแปลงแบบจำลอง

(5) เนื่องจากงานวิจัยนี้ผู้วิจัยมีการประเมินผลของเครื่องมือโดยทำการเก็บข้อมูลจำนวนรายละเอียดของซอร์สโค้ดที่ผู้วิจัยได้กำหนดไว้ด้วยวิธีการตรวจสอบด้วยตัวผู้วิจัยเองก่อนที่จะนำไป

เปรียบเทียบกับจำนวนข้อมูลของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม อย่างไรก็ตามข้อมูลที่ได้จากวิธีการตรวจสอบด้วยตัวผู้วิจัยเองนั้นอาจก่อให้เกิดข้อผิดพลาดได้ ดังนั้นผู้วิจัยจึงพยายามลดข้อผิดพลาดโดยการเขียนโปรแกรมขึ้นมาเพื่อใช้สำหรับวัดความถูกต้องของข้อมูลจากวิธีการตรวจสอบของผู้วิจัย โดยโปรแกรมจะทำการเก็บข้อมูลรายละเอียดของซอร์สโค้ดที่ผู้วิจัยได้กำหนดไว้ และแสดงออกมาเป็นข้อมูลสรุป ซึ่งจะช่วยลดข้อผิดพลาดจากการตรวจสอบจากตัวผู้วิจัยเองทำให้ข้อมูลที่ได้มีความถูกต้องและน่าเชื่อถือ สำหรับในส่วนการตรวจสอบจำนวนข้อมูลของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมนั้น เพื่อลดข้อผิดพลาดผู้วิจัยได้ตรวจสอบข้อมูลจำนวนหลายครั้งเพื่อให้มั่นใจว่าข้อมูลเหล่านั้นตรงกับแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมจริง และผู้วิจัยได้ให้ผู้เชี่ยวชาญตรวจสอบข้อมูลรายละเอียดของซอร์สโค้ดและแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมทั้งหมดอีกครั้งเพื่อตรวจสอบว่าข้อมูลนั้นมีความถูกต้องจริง

โดยจากการวิจัยครั้งนี้ผู้วิจัยมีความคิดเห็นว่าเครื่องมือจะมีประสิทธิภาพ และสามารถนำไปใช้ประโยชน์มากขึ้นถ้าหากมีการพัฒนาและวิจัยในส่วนต่าง ๆ ดังต่อไปนี้

(1) ระบบสามารถเพิ่มความสามารถในการรองรับแผนภาพยูเอ็มแอลชนิดอื่นนอกเหนือไปจากคลาสไดอะแกรมและซีเควนซ์ไดอะแกรม เพื่อเป็นประโยชน์ต่อนักพัฒนาในการวิเคราะห์ระบบได้ดียิ่งขึ้น

(2) ระบบสามารถเพิ่มการแปลงจากแผนภาพยูเอ็มแอลกลับมาเป็นซอร์สโค้ดภาษาฟอร์แทรน เพื่อความสะดวกรวดเร็วของนักพัฒนาในการเขียนโปรแกรม

(3) ระบบสามารถเพิ่มส่วนของการแสดงรายงานของระบบที่นำเข้า เช่น จำนวนคลาสทั้งหมด จำนวนเมท็อดทั้งหมด จำนวนบรรทัดทั้งหมดของคลาส ซึ่งอาจจะอยู่ในรูปของเอกสาร หรือกราฟฟิก 2D เพื่อใช้สำหรับวิเคราะห์และแสดงให้เห็นถึงภาพรวมของระบบทั้งหมด ซึ่งจะช่วยให้นักพัฒนาสามารถมองเห็นถึงภาพรวมของระบบ รวมถึงสามารถเก็บเป็นหลักฐานการเปลี่ยนแปลงของระบบแต่ละเวอร์ชันได้

(4) เนื่องจากผู้วิจัยไม่ได้ทำการออกแบบกฎการแปลงซอร์สโค้ดไว้สำหรับรองรับภาษาเชิงวัตถุอื่น เพราะภาษาฟอร์แทรนเป็นภาษาเชิงวัตถุที่แตกต่างไปจากภาษาเชิงวัตถุทั่วไป จึงทำให้กฎการแปลงที่ได้ไม่ครอบคลุมถึงภาษาเชิงวัตถุอื่น อย่างไรก็ตามผู้วิจัยเห็นว่า ผลจากการศึกษาจะเป็นแนวทางสำหรับการออกแบบกฎการแปลงของภาษาเชิงวัตถุอื่นที่มีความคล้ายคลึงกัน

(5) จากผลลัพธ์ของการพัฒนาเครื่องมือสามารถแปลงซอร์สโค้ดภาษาฟอร์แทรนเป็นยูเอ็มแอลคลาสไดอะแกรมและยูเอ็มแอลซีเควนซ์ไดอะแกรมได้เพียงเท่านั้น ซึ่งยังไม่ครอบคลุมถึงภาษา

เชิงวัตถุและยูเอ็มแอลไดอะแกรมชนิดอื่น อย่างไรก็ตามผู้วิจัยเห็นว่าผลจากการศึกษาจะเป็นแนวทางสำหรับผู้สนใจในการแปลงภาษาเชิงวัตถุและยูเอ็มแอลไดอะแกรมชนิดอื่นที่มีความคล้ายคลึงกัน

6.3 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอเครื่องมือสำหรับแปลงซอร์สโค้ดภาษาฟอร์แทรนให้อยู่ในรูปของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม เพื่อใช้สำหรับวิเคราะห์และทำความเข้าใจระบบที่พัฒนาขึ้นด้วยภาษาฟอร์แทรน โดยเมื่อผู้ใช้งานนำซอร์สโค้ดภาษาฟอร์แทรนเข้าสู่ระบบ โดยระบบจะทำการสร้างเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมา และสามารถแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมผ่านซอฟต์แวร์มอเดลลิโอ ซึ่งข้อสรุปจากงานวิจัย คือ เครื่องมือที่ได้นำเสนอนั้นสามารถทำงานได้ถูกต้อง โดยในการทดสอบนั้นถึงแม้จะมีการเรียกใช้งานที่แตกต่างไปจากการเรียกใช้งานแบบฟังก์ชันหรือซับรูทีน แต่ผู้วิจัยได้มีการปรับปรุงเครื่องมือให้สามารถแสดงแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่มีความครบถ้วนสมบูรณ์ได้ ซึ่งผลจากการพัฒนาเครื่องมือจะช่วยให้นักพัฒนามีการตัดสินใจที่ดีขึ้นในการพัฒนาซอฟต์แวร์ อีกทั้งยังมีส่วนช่วยในกระบวนการบำรุงรักษาระบบ

เอกสารอ้างอิง

Abran, A., Bourque, P., Dupuis, R., and Moore, J. W. (2001). *Guide to the software engineering body of knowledge (SWEBOK)*, IEEE Computer Society Press.

Akin, E. (2003). *Object-oriented programming via Fortran 90/95*, Cambridge University Press.

Alalfi, M. H., Cordy, J. R., and Dean, T. R. (2009). "Automated reverse engineering of UML sequence diagrams for dynamic web applications.", *Proceedings of The 9th IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2009)*, Denver, CO, USA: 1-4 April, 2009.

Andritsos, P., and Miller, R. J. (2001). "Reverse engineering meets data analysis.", *Proceedings of The 9th International Workshop on Program Comprehension (IWPC 2001)*, Toronto, Ontario, Canada: 12-13 May, 2001.

Barbieri, D., Cardellini, V., Filippone, S., and Rouson, D. (2011). "Design patterns for scientific computations on sparse matrices.", *Proceedings of European Conference on Parallel (Euro-Par 2011)*, Bordeaux, France: 29 August – 2 September, 2011.

Bell, D. (2004). "UML basics: The sequence diagram." *IBM Corporation*, 1–24.

Bogdan, P., Bob, T., and Harald, B. (2018). "ArgoUML." *(Online) Available on http://argouml.tigris.org/ (20 Jan 2018)*.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2008). "Extensible markup language (XML) 1.0." *(Online) Available on https://www.w3.org/TR/2008/REC-xml-20081126/*

Brian, F. (2016). "NAG." *(Online) Available on http://www.nag.com/ (19 Dec 2017)*.

Briand, L. C., Labiche, Y., and Miao, Y. (2003). "Towards the Reverse Engineering of UML Sequence Diagrams.", *Proceedings of The 10th Working Conference on Reverse Engineering (WCRE 2003)*, Victoria, British Columbia, Canada: 13-16 November, 2003.

Budiardja, R., Cardall, C., Endeve, E., and Mezzacappa, A. (2012). "Poster: GenASiS: General Astrophysics Simulation System-Object-Oriented Approach to High Performance Multiphysics Code with Fortran 2003.", *Proceedings of High Performance*

Computing, Networking, Storage and Analysis (SCC), Salt Lake City, UT, USA: 10-16 November, 2012.

Carver, J. C. (2009). "Report: the second international workshop on software engineering for CSE." *Computing in Science & Engineering*, AIP Publishing, 11(6), 14–19.

Carver, J. C. (2012). "Software engineering for computational science and engineering." *Computing in Science & Engineering*, AIP Publishing, 14(2), 8–11.

Carver, J. C., Kendall, R. P., Squires, S. E., and Post, D. E. (2007). "Software development environments for scientific and engineering software: A series of case studies." *Proceedings of The 29th International Conference on Software Engineering (ICSE 2007)*, Minneapolis, MN, USA: 20-26 May, 2007.

Clerman, N. S., and Spector, W. (2011). *Modern Fortran: style and usage*. Cambridge University Press.

Craig, R., Matthew, S., and Dan, Q. (2018). "Open Fortran Parser." *(Online) Available on http://fortran-parser.sourceforge.net/ (20 Jan 2018)*.

Cray. (2016). "Cray." *(Online) Available on http://www.nersc.gov/users/software/compilers/cray-compilers/ (19 Dec 2017)*.

Decyk, V. K., Norton, C. D., and Gardner, H. J. (2007). "Why fortran?" *Computing in Science and Engineering*, 9(4), 68–71.

Dobing, B., and Parsons, J. (2006). "How UML is used." *Communications of the ACM*, 49(5), 109–113.

Feathers, M. (2004). *Working effectively with legacy code*. Prentice Hall Professional.

Felderer, M., and Herrmann, A. (2018). "Comprehensibility of system models during test design: a controlled experiment comparing UML activity diagrams and state machines." *Software Quality Journal*, 1–23.

ForTrilinos. (2017). "ForTrilinos." *(Online) Available on http://trilinos.sandia.gov/packages/fortrilinos/ (2 Jan 2017)*.

IBM. (2016). "IBM XL Fortran." *(Online) Available on http://www-03.ibm.com/software/products/en/fortcompfami/ (19 Dec 2017)*.

IEEE. (2017). "Top programming languages." *(Online) Available on http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages/ (2 Jan 2017)*.

Intel. (2016). “Intel Fortran.” *(Online) Available on https://software.intel.com/en-us/fortran-compilers/ (19 Dec 2017)*.

Jiang, Y., Duan, Y., Huang, M., Chen, M., Li, J., and Zhou, H. (2017). “Processing Redundancy in UML Diagrams Based on Knowledge Graph.”, *Proceedings of The 8th International Symposium on Parallel Architecture, Algorithm and Programming (PAAP 2017)*, Springer, Singapore: 06 October, 2017.

Korshunova, E., Petkovic, M., Van Den Brand, M. gj, and Mousavi, M. R. (2006). “CPP2XMI: reverse engineering of UML class, sequence, and activity diagrams from C++ source code.”, *Proceedings of The 13th Working Conference on Reverse Engineering (WCRE 2006)*, Benevento, Italy: 23-27 October, 2006.

Lanza, M., and Ducasse, S. (2003). “Polymetric views-a lightweight visual approach to reverse engineering.” *IEEE Transactions on Software Engineering*, IEEE, 29(9), 782–795.

Lethbridge, T. C., Tichelaar, S., and Plödereder, E. (2004). “The dagstuhl middle metamodel: A schema for reverse engineering.” *Electronic Notes in Theoretical Computer Science*, Elsevier, 94, 7–18.

Li, C., Dou, L., and Yang, Z. (2014). “A metamodeling level transformation from UML sequence diagrams to Coq.”, *Proceedings of The 1st International Conference on Information and Communication Technology for Competitive Strategies (ICTCS 2014)*, Perugia, Italy: 17-19 September, 2014.

Li, Y., Gu, P., and Zhang, C. (2014). “Transforming UML class diagrams into HBase based on meta-model.”, *Proceedings of 4th International Conference on Information Science, Electronics and Electrical Engineering (ISEEE 2014)*, Sapporo, Japan: 26-28 April, 2014.

Linzhang, W., Yu, L., Xuandong, L., Chen, Z., and others. (2017). “Activity diagram model-based system behavior simulation method.” *(Online) Available on https://patents.google.com/patent/US9594543B2/en (19 Dec 2017)*.

Medvidovic, N., Rosenblum, D. S., Redmiles, D. F., and Robbins, J. E. (2002). “Modeling software architectures in the Unified Modeling Language.” *ACM Transactions on Software Engineering and Methodology*, 11(1), 2–57.

Merah, E. (2014). “Design of ATL Rules for Transforming UML 2 Sequence Diagrams into Petri Nets.” *International Journal of Computer Science and Business Informatics*, 8(1), 1–21.

Metcalf, M. (2011). “The seven ages of fortran.” *Journal of Computer Science & Technology*, 11.

MLD2P4. (2017). “MLD2P4.” *(Online) Available on http://www.mld2p4.it/ (2 Jan 2017)*.

Modeliosoft. (2017). “Modelio.” *(Online) Available on https://www.modelio.org/ (19 Dec 2017)*.

Morris, K., Rouson, D. W., Lemaster, M. N., and Filippone, S. (2012). “Exploring capabilities within ForTrilinos by solving the 3D Burgers equation.” *Scientific Programming*, Hindawi Publishing Corporation, 20(3), 275–292.

Müller, H. A., Orgun, M. A., Tilley, S. R., and Uhl, J. S. (1993). “A reverse-engineering approach to subsystem structure identification.” *Journal of Software Maintenance: Research and Practice*, Wiley Online Library, 5(4), 181–204.

Mythily, M., Valarmathi, M., and Durai, C. A. D. (2018). “Model transformation using logical prediction from sequence diagram: an experimental approach.” *Cluster Computing*, 1–12.

Nanthaamornphong, A., Carver, J., Morris, K., and Filippone, S. (2015). “Extracting uml class diagrams from object-oriented fortran: Foruml.” *Scientific Programming, Article ID 421816*, Hindawi Publishing Corp., 1–15.

Nikulchev, E., and Deryugina, O. (2016). “Model and Criteria for the Automated Refactoring of the UML Class Diagrams.” *International Journal of Advanced Computer Science and Applications*, 7(12), 76–79.

Ning, J. Q., Engberts, A., and Kozaczynski, W. V. (1994). “Automated support for legacy code understanding.” *Communications of the ACM*, Association for Computing Machinery, Inc., 37(5), 50–58.

Obeo. (2017). “UML Designer.” *(Online) Available on http://www.umldesigner.org/ (19 Dec 2017)*.

OMG. (2017). “Object Constraint Language.” *(Online) Available on http://www.omg.org /spec/OCL/ (19 Dec 2017)*.

OMG. (2017). “UML specification v2.5.” *(Online) Available on http://www.omg.org/spec/UML/2.5/PDF (2 Jan 2017)*.

Parada, A. G., Siegert, E., and Brisolara, L. B. de. (2011). “Generating Java code from UML class and sequence diagrams.”, *Proceedings of The 1st The Brazilian Symposium on Computing System Engineering (SBESC 2011)*, Florianopolis, Brazil: 7-11 November 2011.

Paul, B., Steven, B., and Bud, D. (2016). “GNU.” *(Online) Available on http://gcc.gnu.org/fortran/ (19 Dec 2017)*.

PSBLAS. (2017). “PSBLAS.” *(Online) Available on http://www.ce.uniroma2.it/psblas/ (2 Jan 2017)*.

Reid, J. (2003). “The future of Fortran.” *Computing in Science and Engineering*, 5(4), 59–67.

Reid, J. (2008). “The new features of Fortran 2008.” *ACM SIGPLAN Fortran Forum*, ACM, 8–21.

Rhazali, Y., Hadi, Y., and Mouloudi, A. (2016). “Model Transformation with ATL into MDA from CIM to PIM Structured through MVC.” *Procedia Computer Science*, 83, 1096–1101.

Rouson, D. W., Adalsteinsson, H., and Xia, J. (2010). “Design patterns for multiphysics modeling in Fortran 2003 and C++.” *ACM Transactions on Mathematical Software*, 37(3), 1–30.

Rouson, D. W., Xia, J., and Xu, X. (2010). “Object construction and destruction design patterns in fortran 2003.” *Procedia Computer Science*, Elsevier, 1(1), 1495–1504.

Rukin, A. (2017). “Java decompilers.” *(Online) Available on http://www.javadecompilers.com/ (19 Dec 2017)*.

Salah, R. M., Alves, G. R., Guerreiro, P., and Gustavsson, I. (2016). “Using UML Models to Describe the VISIR System.” *International Journal of Online Engineering (iJOE)*, 12(6), 34–42.

Sawprakhon, P., and Limpiyakorn, Y. (2014). “Sequence Diagram Generation with Model Transformation Technology.”, *Proceedings of The 23rd International*

MultiConference of Engineers and Computer Scientists (IMECS 2014), Kowloon, Hong Kong: 12-14 March, 2014.

Srai, A., Guerouate, F., Berbiche, N., and Drissi, H. (2017). "An MDA approach for the development of data warehouses from Relational Databases Using ATL Transformation Language." *International Journal of Applied Engineering Research*, 12(12), 3532–3538.

Systa, T. (1999). "On the relationships between static and dynamic models in reverse engineering java software.", *Proceedings of The 6th Working Conference on Reverse Engineering (WCRE 1999)*, Atlanta, GA, USA: 8 October, 1999.

Team, U. (2017). "Umbrello." *(Online) Available on https://umbrello.kde.org/ (19 Dec 2017)*.

Terence, P. (2017). "ANother Tool for Language Recognition." *(Online) Available on http://www.antlr.org/ (2 Jan 2017)*.

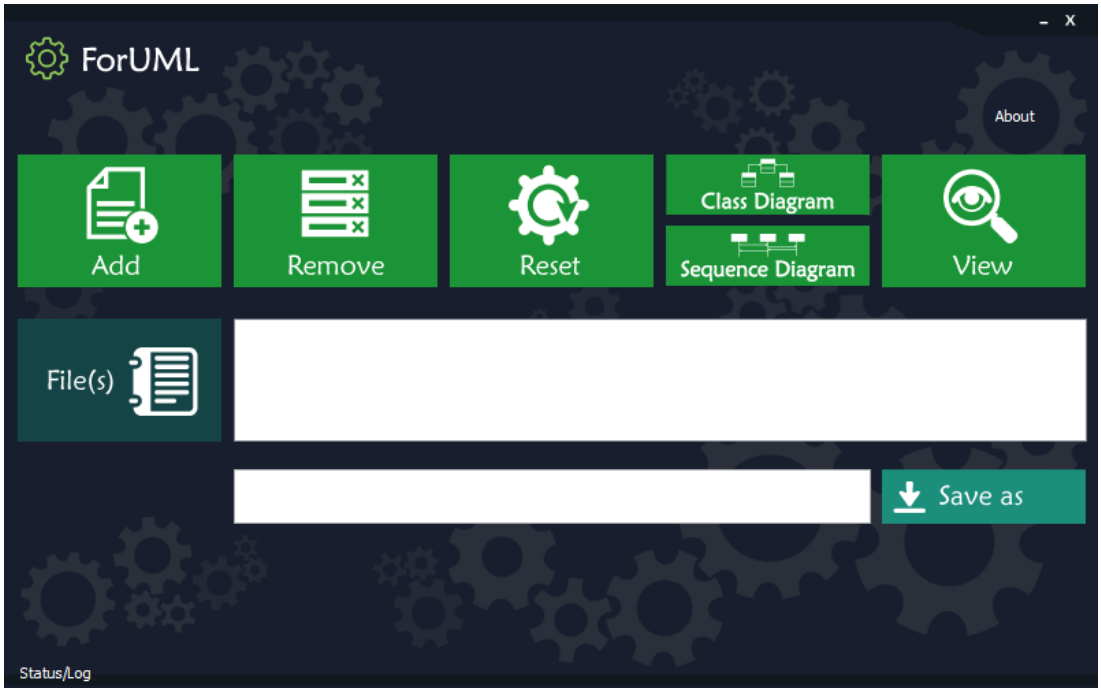
Willems, C., and Freiling, F. C. (2012). "Reverse Code Engineering—State of the Art and Countermeasures." *Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 54(2), 53–63.

ภาคผนวก

ภาคผนวก

รายละเอียดเพิ่มเติมของการใช้งานเครื่องมือในการสร้างแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรม

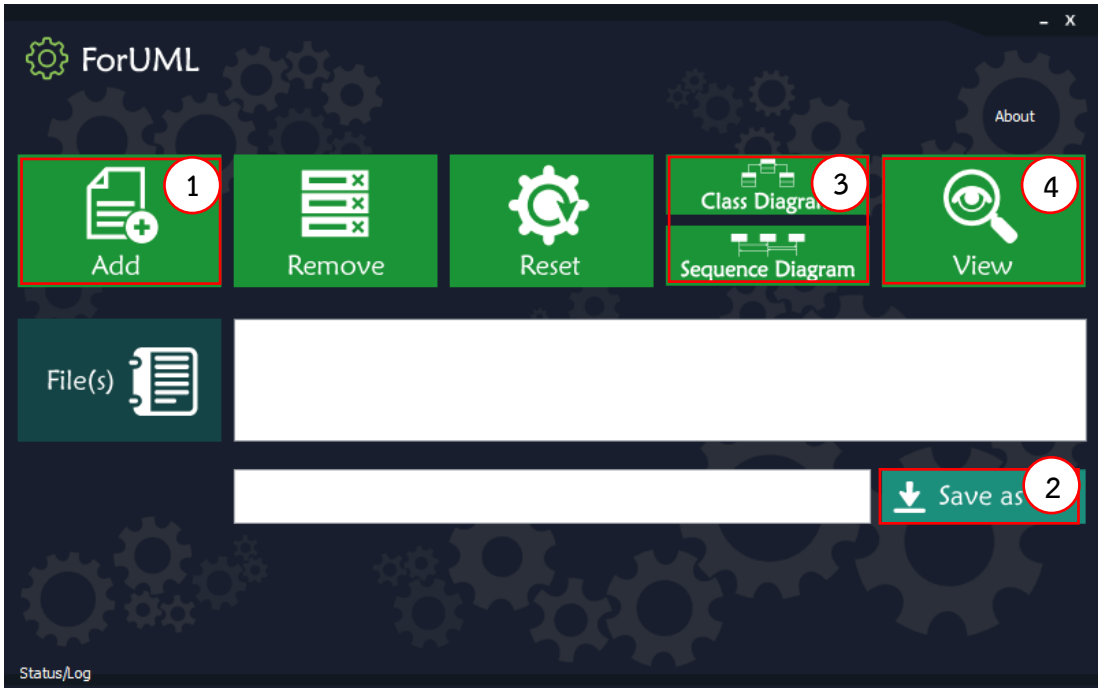
เครื่องมือถูกพัฒนาขึ้นมาเพื่อใช้ในการสร้างแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมขึ้นมาจากซอร์สโค้ดภาษาฟอร์แทรน โดยการใช้งานเครื่องมือนั้นจำเป็นจะต้องมีเอกสารซอร์สโค้ดภาษาฟอร์แทรนที่เป็นภาษาเชิงวัตถุเวอร์ชัน 2003 ขึ้นไป รูปแผนภาพยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้นั้นจะถูกแสดงผ่านซอฟต์แวร์มอเดลลิโอ



รูปที่ ก.1 หน้าจอของเครื่องมือ ForUML

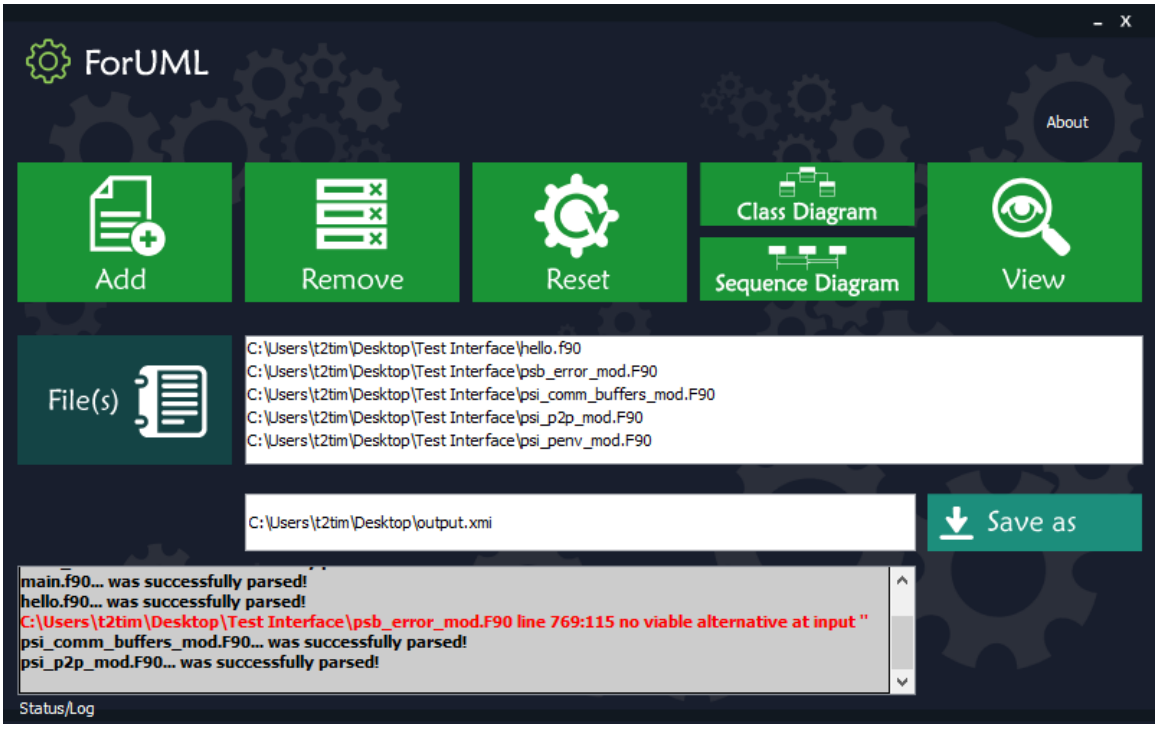
จากรูปที่ ก.1 แสดงหน้าจอของเครื่องมือ ForUML ซึ่งประกอบด้วย 7 ปุ่มการทำงานหลัก คือ 1) ปุ่ม Add ทำการเพิ่มไฟล์ซอร์สโค้ด 2) ปุ่ม Remove ทำการลบไฟล์ซอร์สโค้ดที่เพิ่มเข้ามา 3) ปุ่ม Reset ทำการรีเซตระบบเมื่อเกิดปัญหา 4) ปุ่ม Class Diagram และ Sequence Diagram ทำการสร้างเอกสารเอกซ์เอ็มไอของแผนภาพยูเอ็มแอลไดอะแกรมขึ้นมา 5) ปุ่ม View ทำการแสดงแผนภาพยูเอ็มแอลไดอะแกรมขึ้นมาผ่านซอฟต์แวร์มอเดลลิโอ 6) ปุ่ม Save as ทำการเลือกตำแหน่งบันทึกเอกสาร

เอกซ์เอ็มไอ 7) ปุ่ม Status/Log แสดงข้อความเมื่อระบบทำงานสำเร็จหรือมีข้อผิดพลาดเกิดขึ้น โดยจะต้องมีการเรียงลำดับการใช้งานของระบบ แสดงดังรูปที่ ก.2



รูปที่ ก.2 ลำดับการใช้งานของเครื่องมือ ForUML

จากรูปที่ ก.2 แสดงลำดับการใช้งานของเครื่องมือ โดยเริ่มต้นผู้ใช้งานจะต้องกดปุ่ม Add เพื่อนำเอกสารซอร์สโค้ดภาษาฟอร์แทรนเข้าสู่ระบบ จากนั้นเมื่อได้เอกสารซอร์สโค้ดที่ต้องการแล้ว ผู้ใช้งานจะต้องกดปุ่ม Save as เพื่อทำการเลือกตำแหน่งบันทึกเอกสารเอกซ์เอ็มไอ เมื่อเลือกตำแหน่งเสร็จแล้วผู้ใช้งานสามารถกดปุ่ม Class Diagram หรือ Sequence Diagram เพื่อสร้างเอกสารเอกซ์เอ็มไอขึ้นมา สุดท้ายผู้ใช้งานสามารถกดปุ่ม View เพื่อแสดงแผนภาพยูเอ็มแอลไดอะแกรมขึ้นมาจากเอกสารเอกซ์เอ็มไอ โดยถ้าหากซอร์สโค้ดที่นำเข้ามีการทำงานผิดพลาดเกิดขึ้น เครื่องมือสามารถแจ้งเตือนผู้ใช้งานผ่านหน้าจอ Status/Log แสดงดังรูปที่ ก.3



รูปที่ ก.3 ตัวอย่างของซอร์สโค้ดที่มีข้อผิดพลาดเมื่อนำเข้าระบบ

ประวัติผู้เขียน

ชื่อ-สกุล นายอนวัช เล่ห์ทองคำ

รหัสประจำตัวนักศึกษา 5930223003

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิทยาศาสตรบัณฑิต (วิศวกรรมซอฟต์แวร์)	มหาวิทยาลัยสงขลานครินทร์	2559

ทุนการศึกษา

ทุนอุดหนุนการวิจัยจากกองทุนวิจัยวิทยาลัยการคอมพิวเตอร์

การตีพิมพ์เผยแพร่ผลงาน

Nanthaamornphong, A., Leatongkam, A., Kitpanich, T., and Thongnuan, P. (2015). "Bytecode-based class dependency extraction tool: Bytecode-CDET.", *Proceedings of The 7th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Chiang Mai, Thailand: 29-30 October, 2015.

Leatongkam, A., Nanthaamornphong, A., and Rouson, D. (2017). " WIP: Generating Sequence Diagrams for Modern Fortran", *Proceedings of The 12th International Workshop on Software Engineering for Science co-located with ICSE 2017*, Buenos Aires, Argentina: 22-22 May, 2017.

อนวัช เล่ห์ทองคำ, และ อชีส นันทอมรพงศ์. (2560). “กฎการแปลงโมเดิร์นฟอร์แทรนสำหรับยูเอ็มแอลซีเควนซ์ไดอะแกรม.”, *งานประชุมวิชาการระดับชาติด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ครั้งที่ 13*, โรงแรมอโนมา แกรนด์ กรุงเทพมหานคร: 6-7 กรกฎาคม 2560.

Nanthaamornphong, A., and Leatongkam, A. (2017) "Modern Fortran Transformation Rules for UML Sequence Diagrams" *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(3-4), 123-129.