การแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับปฏิบัติการของรหัสคำสั่งภาษาจาวา

นายชัชวีร์ ตั้งสายัณห์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2546
ISBN 974-17-4517-6

TRANSFORMATION OF MULTIPLE SEQUENCE DIAGRAMS INTO OPERATION-LEVEL BEHAVIOR OF JAVA CODE

Mr.Chatchawee Tangsayan

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2003
ISBN 974-17-4517-6

หัวข้อวิทยานิพนธ์	การแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับปฏิบัติการของรหัสคำสั่งภาษาจาวา
โดย	นายชัชวีร์ ตั้งสายัณห์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

.....กรรมการ
(อาจารย์นครทิพย์ พร้อมพูล)

ชัชวีร์ ตั้งสายัณห์ : การแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับปฏิบัติการของรหัสคำสั่งภาษาจาวา. (Transformation of Multiple Sequence Diagrams into Operation-Level Behavior of Java Code) อ. ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี, 140 หน้า. ISBN 974-17-4517-6.

วิทยานิพนธ์ฉบับนี้กล่าวถึงการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับปฏิบัติการของรหัสคำสั่งภาษาจาวา ซึ่งเป็นการสนับสนุนการทำการแปลงข้อมูลในระยะของการออกแบบซอฟต์แวร์ไปเป็นข้อมูลในระยะของการอิมพลีเมนต์โดยอัตโนมัติ อันจะทำให้สามารถลดค่าใช้จ่ายและเวลาในกระบวนการพัฒนาซอฟต์แวร์ อีกทั้งยังช่วยเพิ่มความถูกต้องให้กับซอฟต์แวร์ได้เป็นอย่างดี

ในการวิจัยได้มีการออกแบบขั้นตอนและกฎในการทำการแปลงดังกล่าว โดยผู้วิจัยให้ความสำคัญในการขจัดข้อจำกัดของงานวิจัยที่มีอยู่เดิมอันเป็นอุปสรรคต่อการนำไปใช้งานจริง อันได้แก่ การไม่สามารถทำการแปลงแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันหลายระดับและไม่สามารถรวมพฤติกรรมของแต่ละโอเปอร์เรชันซึ่งอาจปรากฏอยู่ได้ในแผนภาพซีเควนซ์หลายแผนภาพตามแต่ละสถานการณ์ที่แตกต่างกัน จากนั้นจึงได้ทำการอิมพลีเมนต์ขั้นตอนและกฎดังกล่าวด้วยภาษาเอ็กซ์เอสแอลที ซึ่งทำให้ได้เครื่องมือที่สามารถทำงานได้บนหลายแพลตฟอร์ม และได้ทำการทดสอบเครื่องมือดังกล่าวกับกรณีศึกษาสองกรณีด้วยการเปรียบเทียบรหัสคำสั่งที่ได้จากเครื่องมือกับรหัสคำสั่งที่ได้จากการทำการแปลงด้วยตนเอง พบว่าเครื่องมือสามารถให้ผลการทำการแปลง ฯ ที่ถูกต้อง ซึ่งเมื่อคิดเป็นร้อยละของขนาดของรหัสคำสั่งเชิงพฤติกรรมที่เครื่องมือสร้างได้จากกรณีศึกษาทั้งสองแล้ว จะคิดได้เป็นร้อยละ 100 สำหรับโอเปอร์เรชันที่ไม่มีพฤติกรรมที่แตกต่างกันตามแต่ละสถานการณ์ และคิดได้เป็นร้อยละตั้งแต่ 72.73 ถึง 82.61 สำหรับโอเปอร์เรชันที่มีพฤติกรรมแตกต่างกันตามแต่ละสถานการณ์การณ์ซึ่งผู้ใช้จะต้องทำการเพิ่มเติมรหัสคำสั่งส่วนของการเลือกสถานการณ์ด้วยตนเอง

ภาควิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....
สาขาวิชา.....วิศวกรรมซอฟต์แวร์..... ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา...2546...

4570676321 : MAJOR SOFTWARE ENGINEERING

KEY WORD: TRANSFORMATION / SEQUENCE DIAGRAM / UML / CODE GENERATION / JAVA / XSLT

CHATCHAWEE TANGSAYAN : TRANSFORMATION OF MULTIPLE SEQUENCE DIAGRAMS INTO OPERATION-LEVEL BEHAVIOR OF JAVA CODE. THESIS ADVISOR : ASSISTANT PROFESSOR PORNSIRI MUENCHAISRI, Ph.D., 140 pp. ISBN 974-17-4517-6.

This thesis describes a transformation of multiple sequence diagrams into operation-level behavior of Java code, which promotes the automatic transformation of information in software design phase into information in implementation phase. As a result, it significantly reduces time and cost in the software development process, and also improves correctness of software.

In this research, steps and rules for the transformation are designed to overcome the major limitations of the previous research, which are obstacles for real situation usage. These limitations include the lack of ability to transform sequence diagrams that have multiple levels of call nesting, and to merge behavior from different scenarios. Then these steps and rules are implemented as a tool using XSLT language, which enables the tool to be applied in multiple platforms. The tool is tested with two case studies by comparing its transformation results to the manual transformation results. The comparison shows that the tool can give the correct results. 100 percent of code is generated for operations that do not have different behavior for each scenario. 72.73 to 82.61 percent of code is generated for operations that have different behavior for each scenario in which users have to add the code for selecting scenarios by themselves.

Department....Computer Engineering..... Student's signature.....
Field of study....Software Engineering..... Advisor's signature.....
Academic year ...2003.....

กิตติกรรมประกาศ

ข้าพเจ้าใคร่ขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า ที่กรุณาแนะนำให้ความรู้ คำปรึกษา ความช่วยเหลือต่าง ๆ ตลอดจนคอยดูแลการทำวิทยานิพนธ์ของข้าพเจ้าจนสำเร็จลุล่วงลงได้ด้วยดี

ขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ ซึ่งเป็นประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา และ อาจารย์นครทิพย์ พร้อมพูล ซึ่งเป็นกรรมการสอบวิทยานิพนธ์ ซึ่งได้สละเวลาและให้คำแนะนำต่าง ๆ ที่เป็นประโยชน์อย่างยิ่งต่อการจัดทำวิทยานิพนธ์ฉบับนี้

ขอขอบคุณอาจารย์ทุกท่าน ที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า รวมถึงชี้แนะสิ่งดี ๆ ตลอดเวลาที่ข้าพเจ้าได้ศึกษาเล่าเรียนในระดับมหาบัณฑิต ณ สถาบันแห่งนี้

ขอขอบคุณศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ ที่ได้จัดการแข่งขันการพัฒนาโปรแกรมคอมพิวเตอร์แห่งประเทศไทย ซึ่งสามารถช่วยกระตุ้นให้ข้าพเจ้ามีความกระตือรือร้นในการทำวิทยานิพนธ์ได้เป็นอย่างดี

ท้ายที่สุด ข้าพเจ้าใคร่ขอกราบขอบพระคุณบิดา มารดา และพี่ชายของข้าพเจ้า ที่คอยให้กำลังใจ และสนับสนุนด้านการเงินแก่ข้าพเจ้าเสมอมา

สารบัญ

หน้า

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

ตาราง หน้า

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เป็นที่ทราบกันดีว่าปัญหาที่สำคัญของศาสตร์ทางด้านวิศวกรรมซอฟต์แวร์ คือ การคิดค้นวิธีที่จะช่วยลดเวลาและค่าใช้จ่ายในกระบวนการพัฒนาซอฟต์แวร์ รวมถึงเพิ่มความถูกต้องให้กับซอฟต์แวร์ที่พัฒนาขึ้นมา ซึ่งผู้วิจัยมีความเห็นว่า วิธีหนึ่งที่จะสามารถช่วยเอาชนะปัญหาดังกล่าวได้เป็นอย่างดีก็คือ การทำให้กระบวนการพัฒนาซอฟต์แวร์เป็นไปโดยอัตโนมัติให้มากขึ้น

โดยในกระบวนการพัฒนาซอฟต์แวร์ในปัจจุบัน การออกแบบซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล (UML : Unified Modeling Language) [5] กำลังเป็นที่นิยมอย่างแพร่หลาย หากสามารถทำการแปลงแผนภาพยูเอ็มแอลไปเป็นรหัสคำสั่งได้โดยตรงก็จะสามารถช่วยลดเวลาและค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ในขั้นตอนการอิมพลีเมนต์ (Implementation) ลงได้เป็นอย่างมาก นอกจากนั้นยังจะทำให้ลดความไม่ถูกต้องของรหัสคำสั่งที่ได้เมื่อเทียบกับที่ได้ออกแบบไว้ ซึ่งเป็นปัจจัยหนึ่งที่จะทำให้คุณภาพของซอฟต์แวร์ดีขึ้นได้

จากเหตุผลดังกล่าว ผู้วิจัยจึงมีความสนใจที่จะศึกษาการแปลงแผนภาพยูเอ็มแอลไปเป็นรหัสคำสั่ง โดยในบรรดาแผนภาพยูเอ็มแอลชนิดต่าง ๆ นั้น แผนภาพซีเควนซ์ (Sequence diagram) ถือเป็นแผนภาพยูเอ็มเอลชนิดหนึ่งที่มีความสำคัญและมักถูกใช้ในการออกแบบซอฟต์แวร์อยู่เสมอ เนื่องจากความสามารถในการใช้แสดงลำดับการส่งเมสเสจ (Message) ระหว่างวัตถุต่าง ๆ ในแต่ละยูสเคส[1] (Use case) จากแผนภาพยูสเคส (Use case diagram) ได้เป็นอย่างดี

ถึงแม้ว่าจะมีความนิยมในการใช้แผนภาพซีเควนซ์ในการออกแบบซอฟต์แวร์อยู่สูง รวมถึงการที่แผนภาพชนิดนี้สามารถให้ข้อมูลลำดับการส่งเมสเสจระหว่างวัตถุต่าง ๆ ได้เป็นอย่างดี แต่การทำการแปลงแผนภาพชนิดนี้ไปเป็นรหัสคำสั่งโดยอัตโนมัติกลับยังไม่เป็นที่แพร่หลายเท่าที่ควร ผู้วิจัยจึงมีความสนใจที่จะศึกษาการแปลงแผนภาพชนิดนี้ เนื่องจากเห็นว่าจะก่อให้เกิดประโยชน์แก่อุตสาหกรรมการพัฒนาซอฟต์แวร์ได้เป็นอย่างมากในทันที

---

[1] ยูสเคส คือ รายละเอียดของชุดลำดับของการกระทำต่าง ๆ ที่ระบบทำเพื่อให้ได้ผลลัพธ์ที่ผู้ใช้ระบบต้องการ [8]

จากการสำรวจงานวิจัยต่าง ๆ พบว่ามีงานวิจัยเกี่ยวกับการแปลงแผนภาพซีเควนซ์ไปเป็นรหัสคำสั่งอยู่บ้างแล้ว แต่ก็ยังไม่สมบูรณ์และมีข้อจำกัดในการใช้งานจริงอยู่ค่อนข้างมาก เช่น ในงานวิจัยการออกแบบกฎในการแปลงแผนภาพซีเควนซ์ไปเป็นรหัสคำสั่งภาษาจาวา (Design of Rules for Transforming UML Sequence Diagrams into Java Code) โดย มฑุปายาส ทองมาก และ พรศิริ หมื่นไชยศรี [1] ยังไม่สามารถทำการแปลงแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันหลายระดับ[2] และไม่สามารถรวมพฤติกรรมของแต่ละโอเปอร์เรชัน (Operation) ซึ่งอาจปรากฏอยู่ได้ในแผนภาพซีเควนซ์หลายแผนภาพตามแต่ละสถานการณ์[3] ที่แตกต่างกัน ส่วนงานวิจัยเรื่องสคีมาตาของอินเทอร์แรคชัน: การคอมไพล์อินเทอร์แรคชันไปเป็นรหัสคำสั่ง (Interaction Schemata: Compiling Interactions to Code) [14] โดย Neeraj Sangal, Edward Farrell, Karl Lieberherr, and David Lorenz ผู้ใช้ต้องมีการเรียนรู้และเสียเวลาในการสร้างสคีมาของอินเทอร์แรคชันจากแผนภาพซีเควนซ์ก่อน เพื่อนำสคีมานี้ไปแปลงไปเป็นรหัสคำสั่ง อีกทั้งยังต้องทำการรวมพฤติกรรมจากสถานการณ์ต่าง ๆ ด้วยตนเอง

จากข้อจำกัดต่าง ๆ ของงานวิจัยการแปลงแผนภาพซีเควนซ์ที่มีอยู่ในปัจจุบัน ส่งผลให้การแปลงแผนภาพชนิดนี้ไม่แพร่หลายเท่าที่ควรทั้ง ๆ ที่มีการใช้แผนภาพชนิดนี้ในการออกแบบซอฟต์แวร์กันอย่างแพร่หลาย ผู้วิจัยจึงขอเสนอที่จะศึกษาและออกแบบขั้นตอนและกฎในการแปลงแผนภาพซีเควนซ์ไปเป็นรหัสคำสั่ง โดยมุ่งเน้นที่จะแก้ไขในข้อจำกัดของงานวิจัยที่มีอยู่เดิม นั่นคือต้องสามารถแปลงแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันหลายระดับ และสามารถทำการรวมพฤติกรรมของแต่ละโอเปอร์เรชันจากหลายสถานการณ์ได้ โดยมีเป้าหมายที่จะให้สามารถนำผลที่ได้จากการวิจัยนี้ไปประยุกต์ใช้ได้จริงในทันที อันจะก่อให้เกิดประโยชน์แก่อุตสาหกรรมการพัฒนาซอฟต์แวร์อย่างกว้างขวางและเป็นรูปธรรม

1.2 คำจำกัดความที่ใช้ในการวิจัย

1.2.1 คำว่า “ปฏิบัติการ” และ “โอเปอร์เรชัน” มีความหมายเหมือนกัน และอาจถูกใช้แทนกันได้

---

[2] การส่งเมสเสจซ้อนกันหลายระดับ คือ วัตถุที่รับเมสเสจทุกตัว สามารถที่จะทำการส่งเมสเสจไปยังวัตถุอื่นได้

[3] สถานการณ์ คือ ลำดับของการกระทำโดยเฉพาะเจาะจงที่แสดงถึงพฤติกรรมของระบบ [8] โดยแต่ละสถานการณ์คือแต่ละเส้นทางการไหลของเหตุการณ์ (flow of events) ในยูสเคส [16]

1.2.2 “การแปลง ฯ” ในวิทยานิพนธ์ฉบับนี้ หมายความถึง “การแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับปฏิบัติการของรหัสคำสั่งภาษาจาวา”

1.3 วัตถุประสงค์ของการวิจัย

1.3.1 เพื่อสร้างวิธีการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา

1.3.2 เพื่อพัฒนาเครื่องมือสนับสนุนการแปลง ฯ ดังกล่าว

1.4 ขอบเขตของการวิจัย

1.4.1 ออกแบบขั้นตอนและกฎในการทำการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา โดยจะต้องได้ขั้นตอนและกฎที่มีความสามารถดังต่อไปนี้

1.4.1.1 สามารถแปลงแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันหลายระดับได้

1.4.1.2 สามารถทำการรวมพฤติกรรมของแต่ละโอเปอร์เรชันที่ปรากฏอยู่ในสถานการณ์ต่าง ๆ ได้

1.4.1.3 สามารถให้ผลการแปลง ฯ เป็นรหัสคำสั่งภาษาจาวาตามข้อกำหนดของภาษาจาวารุ่นที่ 2 (The Java Language Specification, Second Edition) [6] ได้

1.4.2 ออกแบบโครงสร้างพื้นฐานของเครื่องมือและเทคโนโลยีที่ใช้ในการทำการแปลง ฯ โดยการออกแบบนี้จะต้องทำให้ได้เครื่องมือที่มีความสามารถดังต่อไปนี้

1.4.2.1 สามารถรับข้อมูลแผนภาพคลาสและแผนภาพซีเควนซ์ที่ทำตามข้อกำหนดของยูเอ็มแอลรุ่น 1.3 ในรูปของแฟ้มข้อมูลเอ็กซ์เอ็มไอรุ่น 1.1 ที่ถูกส่งออกจากเรชันแนล โรส รุ่น 2002 ด้วยเครื่องมือเสริม ยูนิซิส โรส เอ็กซ์เอ็มแอล ทูล รุ่น 1.3.6 ได้

1.4.2.2 สามารถทำการแปลง ฯ ตามกฎการแปลง ฯ ที่ระบุอยู่ในรูปแบบของภาษาเอ็กซ์เอสแอลที รุ่น 1.0 ได้

1.4.2.3 สามารถให้ผลการแปลง ฯ เป็นแฟ้มข้อมูลรหัสคำสั่งภาษาจาวาได้

1.4.3 พัฒนาเครื่องมือสนับสนุนการทำการแปลง ฯ

1.4.4 ทดสอบการใช้งานของเครื่องมือที่พัฒนาขึ้น เพื่อตรวจสอบว่าเครื่องมือดังกล่าวสามารถทำการแปลง ฯ ได้อย่างถูกต้องหรือไม่ โดยมีรายละเอียดของการทดสอบดังต่อไปนี้

1.4.4.1 มีการทดสอบกับกรณีศึกษาอย่างน้อย 2 กรณี

1.4.4.2 มีการทดสอบการรวมพฤติกรรมของโอเปอร์เรชันที่ถูกระบุอยู่ในสถานการณ์ที่แตกต่างกันอย่างน้อย 2 สถานการณ์

1.4.4.3 มีการทดสอบกับแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันอย่างน้อย 3 ระดับ

1.4.5 ข้อจำกัดของขั้นตอนและกฎในการทำการแปลง ฯ มีดังนี้

1.4.5.1 ไม่มีการพิจารณาการส่งเมสเสจแบบอะซิงโครนัส (Asynchronous)

1.4.5.2 ไม่มีการพิจารณากรณีที่มีการส่งเมสเสจเกิดขึ้นพร้อมกัน (Concurrent)

1.5 ขั้นตอนในการวิจัย

1.5.1 ศึกษาและทำความเข้าใจการออกแบบซอฟต์แวร์ด้วยแผนภาพซีเควนซ์ รวมถึงการดูตัวอย่างการเขียนแผนภาพซีเควนซ์ เพื่อศึกษารูปแบบการเขียนแผนภาพซีเควนซ์ที่เป็นที่นิยม

1.5.2 ศึกษางานวิจัยที่เกี่ยวข้องกับการแปลงแผนภาพซีเควนซ์ไปเป็นรหัสคำสั่ง

1.5.3 ศึกษาเครื่องมือสร้างแผนภาพยูเอ็มแอลที่เป็นที่นิยม โดยพิจารณาถึงความสามารถ ข้อจำกัด และรูปแบบของแฟ้มข้อมูลที่เครื่องมือนั้นจัดเก็บหรือส่งออก

1.5.4 ออกแบบขั้นตอนและกฎในการทำการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา

1.5.5 ศึกษาเทคโนโลยีที่มีความเหมาะสมกับการแปลงข้อมูลที่ได้จากเครื่องมือสร้างแผนภาพยูเอ็มแอลไปเป็นรหัสคำสั่งในภาษาจาวา

1.5.6 ทำการอิมพลีเมนต์เครื่องมือสนับสนุนการทำการแปลง ฯ

1.5.7 ทดสอบการทำงานของเครื่องมือที่พัฒนาขึ้นและปรับปรุงขั้นตอนและกฎในการทำการแปลง ฯ ตามความเหมาะสม

1.5.8 สรุปผลการวิจัย และจัดทำรายงานวิทยานิพนธ์

1.6 ประโยชน์ที่คาดว่าจะได้รับ

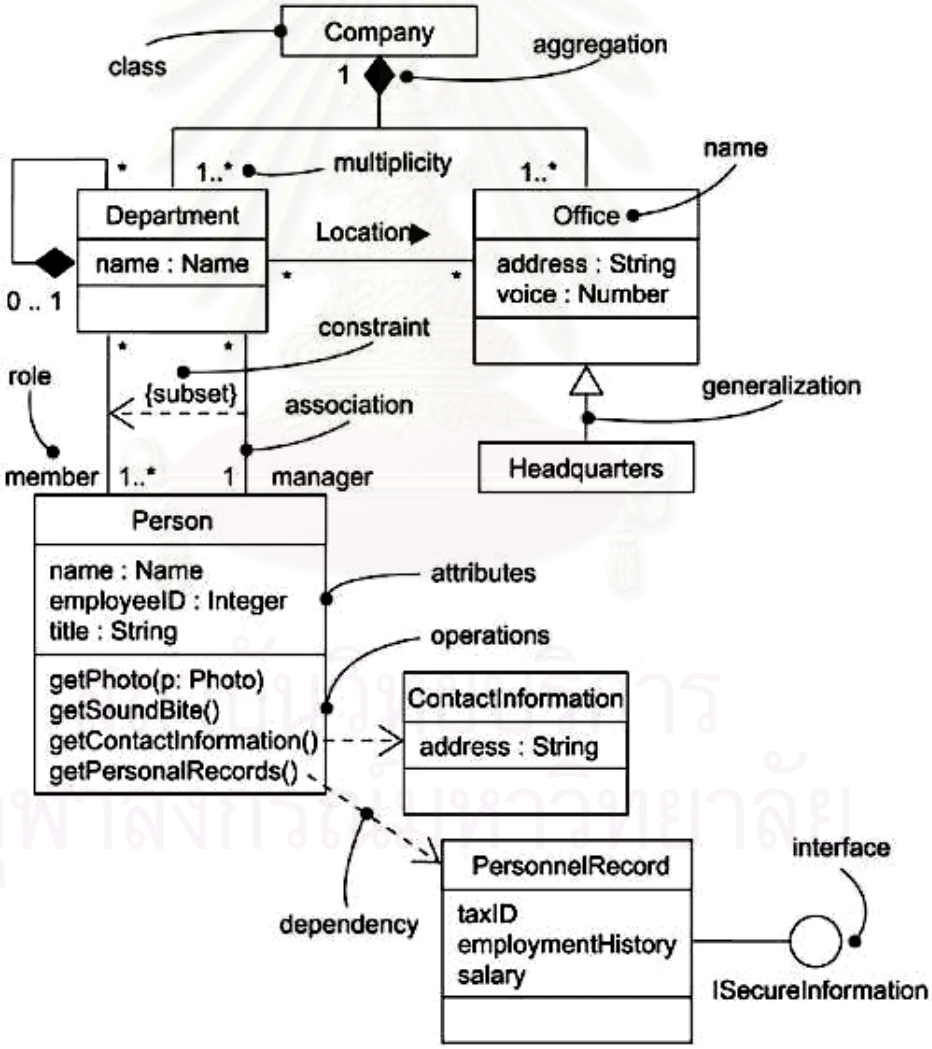
ผลของงานวิจัยนี้ จะทำให้ได้ขั้นตอนและกฎในการทำการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา รวมถึงได้เครื่องมือที่สามารถช่วยนักพัฒนาซอฟต์แวร์ทำการแปลง ฯ ได้โดยอัตโนมัติ ซึ่งจะทำให้สามารถลดค่าใช้จ่ายและเวลาในการพัฒนาซอฟต์แวร์ลงได้ และยังเป็นการเพิ่มความถูกต้องของซอฟต์แวร์ที่ได้ให้เป็นไปตามที่ได้ออกแบบไว้ในแผนภาพซีเควนซ์อีกด้วย

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 แผนภาพคลาส (Class Diagram) [7, 8, 15, 16]

แผนภาพคลาสเป็นแผนภาพยูเอ็มแอลที่พบเห็นได้เสมอในการทำแบบจำลองของระบบเชิงวัตถุและยังมีความสำคัญในการอิมพลีเมนต์ระบบเป็นอย่างมากเนื่องจากสามารถให้ข้อมูลเชิงโครงสร้างของระบบได้เป็นอย่างดี



รูปที่ 2.1 ตัวอย่างของแผนภาพคลาส [8]

รูปที่ 2.1 แสดงถึงตัวอย่างของแผนภาพคลาส โดยแผนภาพคลาสจะประกอบไปด้วย คลาส อินเทอร์เฟส (Interface) และความสัมพันธ์แบบต่าง ๆ ซึ่งมีรายละเอียดดังต่อไปนี้

คลาส คือแบบจำลองของวัตถุแต่ละชนิดในระบบ โดยในแต่ละคลาสจะประกอบไปด้วย 3 ส่วน ได้แก่ ชื่อ แอทริบิวต์ (Attribute) และโอเปอร์เรชัน โดยแอทริบิวต์จะใช้เก็บคุณลักษณะของวัตถุที่ถูกสร้างขึ้นแต่ละตัว ส่วนโอเปอร์เรชันคือการกระทำที่วัตถุสามารถทำได้

อินเทอร์เฟส เป็นการอธิบายถึงกลุ่มของโอเปอร์เรชันที่ใช้ระบุถึงบริการที่เสนอโดยวัตถุของคลาสหนึ่งคลาสใด

ความสัมพันธ์ เป็นการอธิบายถึงความสัมพันธ์ระหว่างสิ่งของต่าง ๆ ซึ่งแบ่งออกเป็นความสัมพันธ์แบบต่าง ๆ ได้ดังนี้

1. ความสัมพันธ์แบบเจเนอรัลไลเซชัน (Generalization) เป็นความสัมพันธ์ระหว่างสิ่งของโดยทั่วไป ซึ่งจะเรียกว่าซูเปอร์คลาสหรือคลาสแม่ กับสิ่งของที่มีความเฉพาะเจาะจงมากกว่า ซึ่งจะเรียกว่าซับคลาสหรือคลาสลูก หรืออาจมองว่าเป็นความสัมพันธ์แบบ “เป็นชนิดหนึ่งของ (Is-a-kind-of)” ก็ได้ โดยคลาสลูกจะรับทอดคุณสมบัติโดยเฉพาะแอทริบิวต์และโอเปอร์เรชันจากคลาสแม่ และสามารถมีแอทริบิวต์และโอเปอร์เรชันเพิ่มเติมจากคลาสแม่ได้ โดยโอเปอร์เรชันของคลาสลูกที่มีลายเซ็น (Signature) เหมือนกับคลาสแม่จะโอเวอร์ไรด์ (Override) โอเปอร์เรชันนั้นของคลาสแม่ ความสัมพันธ์ชนิดนี้แสดงได้ด้วยเส้นทึบที่มีด้านหนึ่งเป็นหัวลูกศรขนาดใหญ่ชี้ไปยังคลาสแม่ โดยจะใช้ความสัมพันธ์ชนิดนี้ในกรณีที่ต้องการแสดงความสัมพันธ์แบบแม่กับลูก
2. ความสัมพันธ์แบบดีเพนเดนซี (Dependency) เป็นความสัมพันธ์แบบ “มีการใช้ (Using)” โดยที่การเปลี่ยนแปลงของข้อกำหนดของสิ่งของหนึ่งอาจจะส่งผลกระทบต่อสิ่งของอื่นที่ใช้สิ่งของนี้ แต่ไม่จำเป็นต้องเกิดผลกระทบในทางกลับกัน ความสัมพันธ์ชนิดนี้จะแสดงได้ด้วยเส้นประชี้ตรงไปยังสิ่งของที่ถูกใช้ โดยจะใช้ความสัมพันธ์แบบนี้ในกรณีที่ต้องการแสดงว่าสิ่งของหนึ่งมีการใช้สิ่งของอื่น
3. ความสัมพันธ์แบบเรียลไลเซชัน (Realization) เป็นความสัมพันธ์ที่แสดงว่าคลาสซิไฟเออร์ (Classifier) หนึ่งจะทำหน้าที่ระบุข้อตกลง โดยที่จะมีคลาสซิไฟเออร์อื่นทำตามข้อตกลงนี้ให้ โดยในแผนภาพคลาสจะใช้ความสัมพันธ์

ชนิดนี้ในการระบุความสัมพันธ์ระหว่างอินเทอร์เฟสกับคลาส โดยอินเทอร์เฟสจะระบุถึงบริการที่เสนอ และคลาสจะให้บริการตามที่ระบุไว้ในอินเทอร์เฟสนั้น ความสัมพันธ์แบบนี้แสดงด้วยเส้นประที่มีหัวลูกศรโปร่งชี้ตรงไปยังคลาสซิไฟเออร์ที่เป็นตัวระบุข้อตกลง แต่ในกรณีที่คลาสซิไฟเออร์เป็นอินเทอร์เฟส อาจแสดงความสัมพันธ์นี้ได้อีกแบบหนึ่งด้วยเส้นทึบและใช้วงกลมโปร่งในการแสดงถึงอินเทอร์เฟส

4. ความสัมพันธ์แบบแอสโซซิเอชัน (Association) เป็นความสัมพันธ์เชิงโครงสร้างที่แสดงว่าวัตถุชนิดหนึ่งถูกเชื่อมต่อกับวัตถุอีกชนิดหนึ่ง โดยถ้ามีการใช้ความสัมพันธ์แบบนี้เชื่อมต่อระหว่างสองคลาส แสดงว่าสามารถที่จะนำทาง (Navigate) จากวัตถุของคลาสหนึ่งไปยังวัตถุของอีกคลาสหนึ่งหรือในทางกลับกันได้ นอกจากนี้ยังสามารถใช้ปลายทั้งสองของความสัมพันธ์ชนิดนี้เชื่อมวนกลับมาหาคลาสเดียวกันซึ่งจะแสดงถึงการเชื่อมต่อกันระหว่างวัตถุของคลาสเดียวกัน ความสัมพันธ์ชนิดนี้แสดงได้ด้วยเส้นทึบเชื่อมระหว่างคลาส โดยจะใช้ความสัมพันธ์ชนิดนี้เมื่อต้องการแสดงความสัมพันธ์เชิงโครงสร้าง

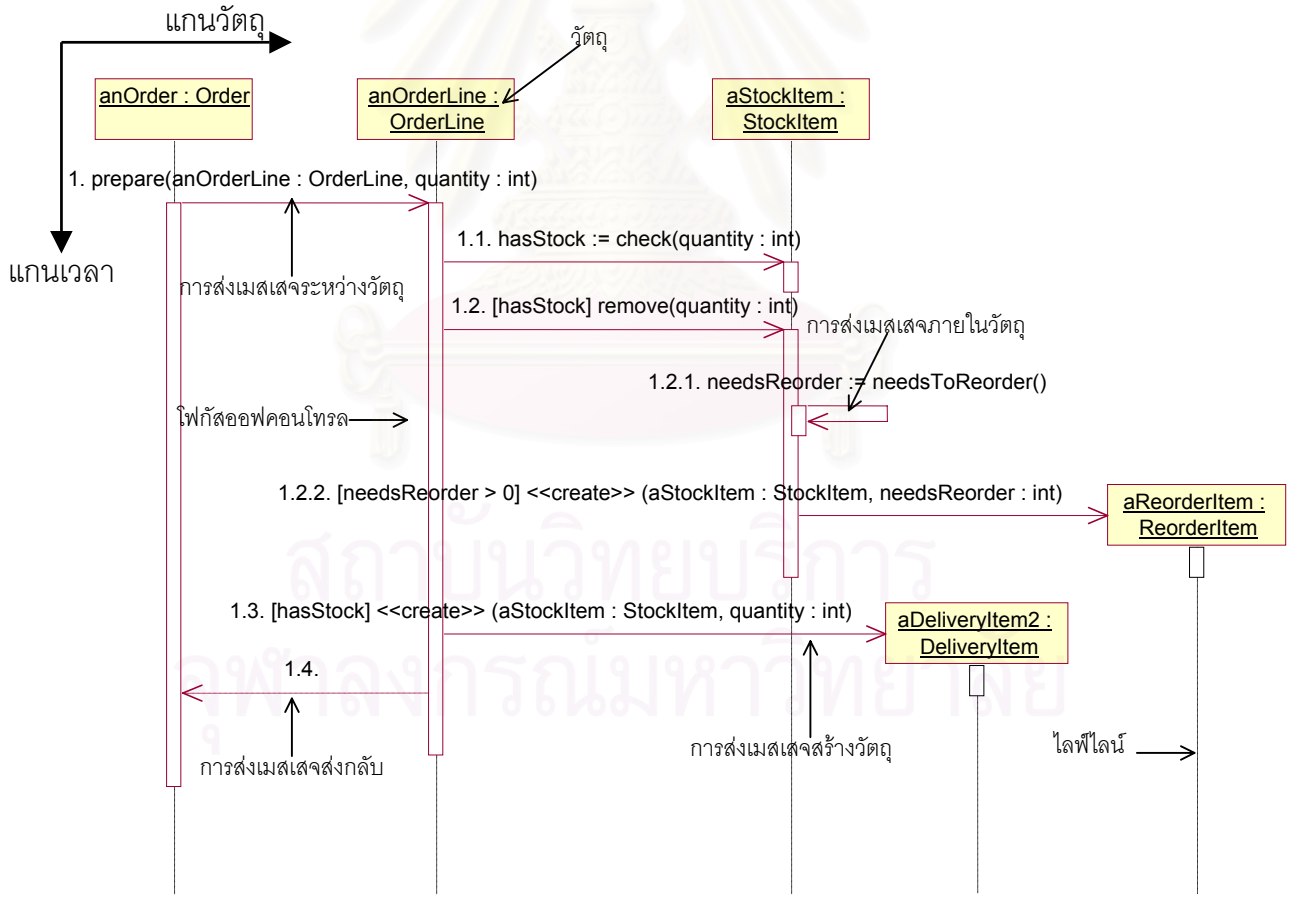
5. ความสัมพันธ์แบบแอกกริเกชัน (Aggregation) เป็นความสัมพันธ์แบบ “ประกอบด้วย (Has-a)” โดยจะมีคลาสซึ่งแสดงถึงสิ่งของที่ใหญ่กว่าที่ประกอบไปด้วยสิ่งของที่เล็กกว่า ความสัมพันธ์ชนิดนี้เป็นกรณีพิเศษของความสัมพันธ์แบบแอสโซซิเอชัน โดยสามารถแสดงได้ด้วยความสัมพันธ์แบบแอสโซซิเอชันที่มีปลายด้านที่เป็นสิ่งของที่ใหญ่กว่าเป็นสี่เหลี่ยมขนมเปียกปูนโปร่ง

6. ความสัมพันธ์แบบคอมโพซิชัน (Composition) เป็นความสัมพันธ์รูปหนึ่งของความสัมพันธ์แบบแอกกริเกชัน แต่จะแสดงถึงความเป็นเจ้าของที่ชัดเจนยิ่งขึ้น และช่วงชีวิตของวัตถุที่เล็กกว่าที่จะขึ้นกับวัตถุที่ใหญ่กว่า โดยในช่วงเวลาใดเวลาหนึ่ง วัตถุที่เล็กกว่าตัวหนึ่งสามารถเป็นส่วนประกอบให้กับวัตถุที่ใหญ่กว่าได้เพียงวัตถุเดียวเท่านั้น ซึ่งแตกต่างไปกับความสัมพันธ์แบบแอกกริเกชันที่วัตถุที่เล็กกว่าตัวหนึ่งสามารถเป็นส่วนประกอบให้กับวัตถุที่ใหญ่กว่าได้หลายวัตถุในเวลาเดียวกัน นอกจากนี้ในความสัมพันธ์แบบคอมโพซิชัน วัตถุที่ใหญ่กว่าจะมีหน้าที่จัดการเกี่ยวกับการสร้างและการทำลายวัตถุที่เล็กกว่า ดังนั้นเมื่อมีการทำลายวัตถุที่ใหญ่กว่า วัตถุที่ใหญ่กว่านั้นจะไปทำลายวัตถุที่เป็นองค์ประกอบของมันด้วย ความสัมพันธ์ชนิดนี้ถือเป็นกรณีพิเศษของความสัมพันธ์แบบ

แอสโซซิเอชัน โดยสามารถแสดงได้ด้วยความสัมพันธ์แบบแอสโซซิเอชันที่มีปลายด้านที่เป็นสิ่งของที่ใหญ่กว่าเป็นสี่เหลี่ยมขนมเปียกปูนทึบ

นอกจากนี้ ในความสัมพันธ์ตระกูลแอสโซซิเอชัน (ความสัมพันธ์ 3 แบบหลัง) ยังอาจมีมัลติพลิซิตี (Multiplicity) ที่แต่ละด้านของความสัมพันธ์ เพื่อแสดงถึงจำนวนที่เป็นไปได้ของวัตถุในด้านนั้นต่อวัตถุแต่ละตัวในอีกด้านหนึ่ง ตัวอย่างเช่น “1” หมายความถึงมีได้ 1 ตัวเท่านั้น “0..1” หมายความถึงมีได้ 0 หรือ 1 ตัว “*” หรือ “0..*” หมายความถึงมีได้หลายตัว “1..*” หมายความถึงมีได้ตั้งแต่ 1 ตัวขึ้นไป หรืออาจระบุจำนวนอย่างเจาะจงเช่น “3” หมายความถึงมีได้ 3 ตัวเท่านั้น เป็นต้น

2.1.2 แผนภาพซีเควนซ์ (Sequence Diagram) [7, 8, 15, 16]



รูปที่ 2.2 ตัวอย่างของแผนภาพซีเควนซ์

แผนภาพซีเควนซ์เป็นแผนภาพอินเทอร์แรคชัน (Interaction diagram) ชนิดหนึ่ง โดยมีจุดเด่นตรงที่สามารถแสดงให้เห็นถึงลำดับการส่งเมสเสจระหว่างวัตถุต่าง ๆ ได้เป็นอย่างดี จึงทำให้มีความนิยมในการใช้แผนภาพนี้ในการแสดงถึงรายละเอียดของกิจกรรมต่าง ๆ ของแต่ละยูสเคสในระยะของการออกแบบระบบ โดยอาจใช้แผนภาพซีเควนซ์เพียงแผนภาพเดียวในการอธิบายถึงสถานการณ์ (Scenario) ทั้งหมดที่เป็นไปได้ใน 1 ยูสเคส แต่โดยทั่วไปแล้วมักมีการแนะนำให้ใช้ 1 แผนภาพซีเควนซ์ในการอธิบายถึง 1 สถานการณ์ในแต่ละยูสเคสมากกว่า

รูปที่ 2.2 เป็นการแสดงตัวอย่างของแผนภาพซีเควนซ์ซึ่งได้ดัดแปลงมาจากหนังสือ UML Distilled [22] วัตถุจะถูกวางอยู่ในแนวของแกนเอ็กซ์ (X axis) ส่วนเมสเสจจะถูกวางตามลำดับของเวลาในแนวแกนวาย (Y axis) โดยมีไลฟ์ไลน์ (Lifeline) เพื่อใช้แสดงถึงช่วงเวลาในการคงอยู่ของวัตถุ และมีโฟกัสออฟคอนโทรล (Focus of control) เพื่อใช้แสดงถึงช่วงเวลาที่วัตถุนั้นแสดงการกระทำ โดยอาจเกิดการซ้อนกันของโฟกัสออฟคอนโทรลขึ้นได้ในกรณีที่เป็นการส่งเมสเสจภายในวัตถุนั้นเอง หรือได้รับการส่งเมสเสจเรียกกลับ (Callback) มาจากวัตถุอื่น

การส่งเมสเสจที่เกี่ยวข้องในงานวิจัยนี้แบ่งออกเป็น 4 ประเภท ได้แก่การส่งเมสเสจระหว่างวัตถุ การส่งเมสเสจภายในวัตถุ การส่งเมสเสจสร้างวัตถุ และการส่งเมสเสจส่งกลับดังที่ได้แสดงในรูปที่ 2.2 โดยในแต่ละการส่งเมสเสจอาจมีการระบุเงื่อนไขไว้ด้วยเช่น [needsReorder > 0] จะหมายความว่าจะเกิดการส่งเมสเสจนั้นก็ต่อเมื่อ needsReorder มีค่ามากกว่า 0 ส่วนในกรณีที่เป็นเงื่อนไขแบบทำซ้ำ จะใส่เครื่องหมายดอกจันหน้าการระบุเงื่อนไขนั้น เช่น *[while(i < 10)] จะหมายความว่าการส่งเมสเสจนี้จะถูกส่งซ้ำไปเรื่อย ๆ ตราบใดที่ i มีค่าน้อยกว่า 10

ในแผนภาพซีเควนซ์ การส่งเมสเสจสามารถเกิดซ้อนกันได้หลายระดับ ตัวอย่างเช่นในรูปที่ 2.2 การส่งเมสเสจในระดับแรกคือการส่งเมสเสจที่ 1 จะไปกระตุ้นให้เกิดการส่งเมสเสจในระดับที่สอง ได้แก่การส่งเมสเสจที่ 1.1 1.2 1.3 และ 1.4 ซึ่งการส่งเมสเสจในระดับที่สองนี้ก็ยังสามารถไปกระตุ้นให้เกิดการส่งเมสเสจในระดับที่สาม ได้แก่การส่งเมสเสจที่ 1.2.1 และ 1.2.2

2.1.3 เอ็กซ์เอ็มไอ (XMI : XML Metadata Interchange) [3, 17]

เอ็กซ์เอ็มไอ เป็นมาตรฐานที่ออกแบบมาเพื่ออำนวยความสะดวกในการแลกเปลี่ยนเมตาเดตา (Metadata) ระหว่างเครื่องมือที่เกี่ยวข้องกับการทำโมเดล (Modeling) ที่มีพื้นฐานมาจากยูเอ็มแอล กับที่เก็บเมตาเดตา (Metadata repositories) ที่มีพื้นฐานมาจากเอ็มโอเอฟ (MOF : Meta Object Facility) ในสภาพแวดล้อมแบบกระจายและมีความหลากหลาย โดยเอ็กซ์เอ็มไอเป็นการรวมเอามาตรฐาน 3 มาตรฐานดังต่อไปนี้เข้าไว้ด้วยกัน

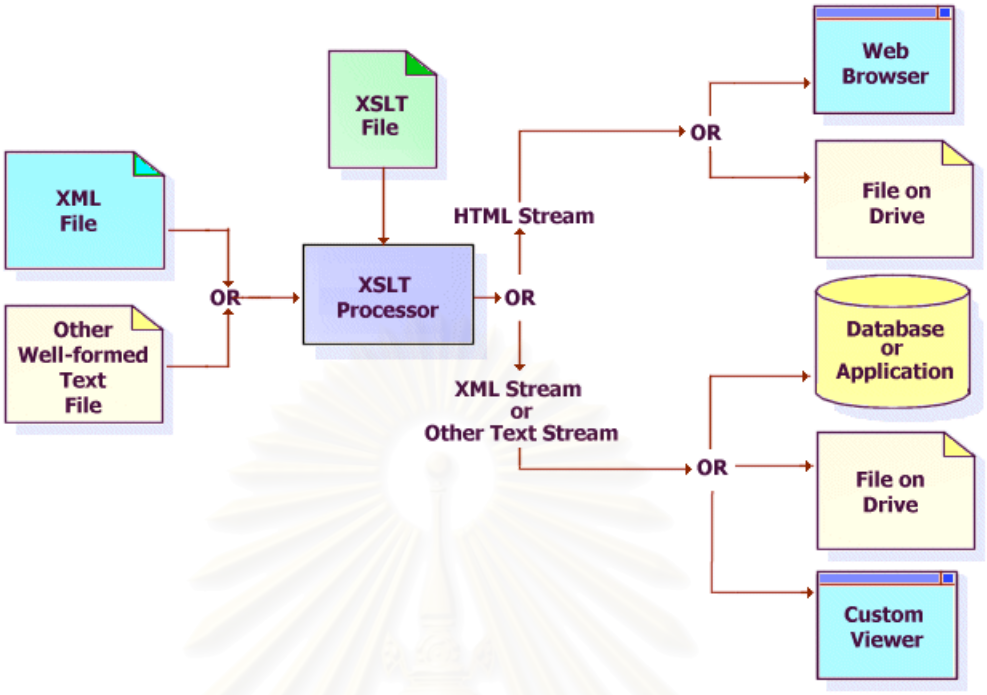
1. เอ็กซ์เอ็มแอล (XML : Extensible Markup Language) [2] เป็นมาตรฐานของดับเบิลยูทรีซี (W3C : World Wide Web Consortium)
2. ยูเอ็มแอล เป็นมาตรฐานที่กำหนดโดยโอเอ็มจี (OMG : Object Management Group) เพื่อใช้ในการทำโมเดล
3. เอ็มโอเอฟ เป็นมาตรฐานที่กำหนดโดยโอเอ็มจี เพื่อใช้ในการทำเมตาโมเดลและที่เก็บเมตาเดตา

การรวมกันของมาตรฐานทั้ง 3 นี้ จึงถือเป็นการรวมเอาเทคโนโลยีเกี่ยวกับเมตาเดตาและการทำโมเดลที่ดีที่สุดของโอเอ็มจีและดับเบิลยูทรีซีเข้าไว้ด้วยกัน ซึ่งทำให้นักพัฒนาสามารถทำการแลกเปลี่ยนโมเดลของวัตถุหรือเมตาเดตาชนิดอื่น ๆ ได้สะดวก โดยเฉพาะอย่างยิ่งในการแลกเปลี่ยนกันทางอินเทอร์เน็ต (Internet)

ในงานวิจัยนี้ จะได้ใช้แฟ้มข้อมูลตามมาตรฐานเอ็กซ์เอ็มไอ เพื่อแลกเปลี่ยนข้อมูลระหว่างเครื่องมือที่จะสร้างขึ้นกับเครื่องมือสร้างภาพยูเอ็มแอล

2.1.4 เอ็กซ์เอสแอลที (XSLT : XSL Transformations) [4, 9, 10, 18, 20, 21]

เอ็กซ์เอสแอลทีเป็นมาตรฐานของดับเบิลยูทรีซี เพื่อใช้ในการอธิบายการแปลงเอกสารเอ็กซ์เอ็มแอลแบบหนึ่งไปเป็นเอกสารเอ็กซ์เอ็มแอลอีกแบบหนึ่งที่มีโครงสร้างแตกต่างกัน หรืออาจใช้อธิบายการแปลงเอกสารเอ็กซ์เอ็มแอลไปเป็นเอกสารประเภทอื่น เช่นเอกสารเอชทีเอ็มแอล (HTML) เอกสารข้อความ (Text) เอกสารพีดีเอฟ (PDF) ฯลฯ ก็ได้เช่นเดียวกัน โดยภาษาเอ็กซ์เอสแอลทีจะถูกระบุอยู่ในรูปแบบของเอกสารเอ็กซ์เอ็มแอล ซึ่งจะได้รับการแปลโดยเอ็กซ์เอสแอลทีโปรเซสเซอร์ (XSLT Processor) ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 การแปลงเอกสารด้วยภาษาเอ็กซ์เอสแอลที [18]

โครงสร้างของภาษาเอ็กซ์เอสแอลทีจะประกอบไปด้วยเทมเพลต (Template) ต่าง ๆ โดยแต่ละเทมเพลตจะระบุกฎในการทำการแปลงเอกสารเอ็กซ์เอ็มแอล โดยวิธีที่จะให้เทมเพลตแต่ละตัวทำการแปลงสามารถแบ่งออกได้เป็น 2 วิธี ดังนี้

1. ระบุแอทริบิวต์ “match” ให้กับเทมเพลต ซึ่งเป็นการบ่งบอกว่าเทมเพลตนี้จะทำการแปลงส่วนย่อย (Element) ใดในเอกสารเอ็กซ์เอ็มแอลที่เป็นข้อมูลเข้า
2. ระบุแอทริบิวต์ “name” ให้กับเทมเพลต แล้วทำการเรียกเทมเพลตนี้โดยตรงด้วยส่วนย่อย “xsl:call-template”

รูปที่ 2.4 และ 2.5 เป็นตัวอย่างของภาษาเอ็กซ์เอสแอลที โดยในรูปที่ 2.4 เป็นตัวอย่างในการใช้ภาษาเอ็กซ์เอสแอลทีทำการแปลงเอกสารเอ็กซ์เอ็มแอลไปเป็นเอกสารเอชทีเอ็มแอล โดยมีการสร้างเทมเพลตต่าง ๆ ที่มีแอทริบิวต์ “match” สอดคล้องกับส่วนย่อยแต่ละชนิดในเอกสารเอ็กซ์เอ็มแอล และมีการระบุกฎการแปลงในแต่ละเทมเพลตเพื่อแปลงส่วนย่อยของเอกสารเอ็กซ์เอ็มแอลแต่ละชนิดไปเป็นภาษาเอชทีเอ็มแอลที่สอดคล้องกัน

XSLT Stylesheet

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

<xsl:template match="bold">
    <p>
        <b>
            <xsl:value-of select="."/>
        </b>
    </p>
</xsl:template>

<xsl:template match="red">
    <p style="color:red">
        <xsl:value-of select="."/>
    </p>
</xsl:template>

<xsl:template match="italic">
    <p>
        <i>
            <xsl:value-of select="."/>
        </i>
    </p>
</xsl:template>


</xsl:stylesheet>
```

XML Source

```
<source>

    <bold>Hello, world.</bold>
    <red>I am </red>
    <italic>fine.</italic>

</source>
```

Output

```
<p>
  <b>Hello, world.</b>
</p>

<p style="color:red">I am </p>

<p>
  <i>fine.</i>
</p>
```

HTML View

Hello, world.

I am

Fine.

รูปที่ 2.4 ตัวอย่างการแปลงเอกสารเอ็กซ์เอ็มแอลไปเป็นเอกสารเอชทีเอ็มแอล ด้วยภาษาเอ็กซ์เอสแอลที [20]

รูปที่ 2.5 เป็นตัวอย่างการหาค่าแฟกทอเรียลด้วยภาษาเอ็กซ์เอสแอลที โดยเทมเพลตแรกมีแอทริบิวต์ “match” เป็น “Number” เพื่อทำการแปลงแต่ละส่วนย่อย “Number” โดยในเทมเพลตแรกนี้จะมีการเรียกใช้เทมเพลตชื่อ “findFactorial” เพื่อหาค่าแฟกทอเรียล ซึ่งในเทมเพลต “findFactorial” นี้ก็ยังมีการเรียกเทมเพลตตัวเองซ้ำเพื่อหาค่าแฟกทอเรียลของตัวเลขที่น้อยกว่าตัวเลขที่มันกำลังหาค่าแฟกทอเรียลอยู่ 1 อีกด้วย

XSLT Stylesheet

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

    <xsl:template match="Number">

        <!--Output-->
        <xsl:text>The factorial of </xsl:text>
        <xsl:value-of select="."/>
        <xsl:text> is </xsl:text>
        <xsl:call-template name="findFactorial">
            <xsl:with-param name="number" select="."/>
        </xsl:call-template>
        <xsl:text>.&#10;</xsl:text>

    </xsl:template>

    <xsl:template name="findFactorial">
        <xsl:param name="number"/>

        <xsl:choose>
            <xsl:when test="$number &gt; 1">

                <xsl:variable name="factorialOf_NumberMinusOne">
                    <!--Recursive Call-->
                    <xsl:call-template name="findFactorial">
                        <xsl:with-param name="number" select="number($number)-1"/>
                    </xsl:call-template>
                </xsl:variable>

                <xsl:value-of select="$number*$factorialOf_NumberMinusOne"/>
            </xsl:when>

            <xsl:when test="$number = 1"><xsl:value-of select="'1'"/></xsl:when>
            <xsl:when test="$number = 0"><xsl:value-of select="'1'"/></xsl:when>
            <xsl:otherwise><xsl:value-of select="'Error'"/></xsl:otherwise>
        </xsl:choose>

    </xsl:template>

</xsl:stylesheet>
```

XML Source	Output

`<Source>`
`<Number>-3</Number>`
`<Number>0</Number>`
`<Number>1</Number>`
`<Number>2</Number>`
`<Number>3</Number>`
`<Number>5</Number>`
`<Number>abc</Number>`
`<Number></Number>`
`</Source>`
	The factorial of -3 is Error.
	The factorial of 0 is 1.
	The factorial of 1 is 1.
	The factorial of 2 is 2.
	The factorial of 3 is 6.
	The factorial of 5 is 120.
	The factorial of abc is Error.
	The factorial of is Error.

รูปที่ 2.5 ตัวอย่างการหาค่าแฟกทอเรียลด้วยภาษาเอ็กซ์เอสแอลที

การใช้ตัวแปรและพารามิเตอร์ในภาษาเอ็กซ์เอสแอลทีมีสิ่งที่ต้องพึงระวัง คือจะทำการกำหนดค่าได้เพียงครั้งเดียวเท่านั้น ดังนั้นในการหาค่าแฟกทอเรียลในรูปที่ 2.5 จะใช้วิธีวนลูป (Loop) เพื่อหาผลคูณตั้งแต่ 1 จนถึงเลขที่ต้องการหาค่าแฟกทอเรียลไม่ได้

ในงานวิจัยนี้จะได้นำภาษาเอ็กซ์เอสแอลทีมาใช้ในการอิมพลีเมนต์กระบวนการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 การออกแบบกฎในการแปลงแผนภาพซีเควนซ์ไปเป็นรหัสคำสั่งภาษาจาวา (Design of Rules for Transforming UML Sequence Diagrams into Java Code) [1] โดย มฑุปายาส ทองมาก และ พรศิริ หมื่นไชยศรี

ในงานวิจัยนี้ได้นำเสนอวิธีในการแปลงแผนภาพซีเควนซ์ไปเป็นรหัสคำสั่งภาษาจาวา โดยแบ่งกระบวนการแปลงออกเป็น 2 ขั้นตอนคือ

1. แปลงแผนภาพซีเควนซ์และคลาสไปเป็นโมเดล (Model) ตามเมตาโมเดล (Meta model) ของแผนภาพซีเควนซ์ ซึ่งเป็นส่วนหนึ่งในเมตาโมเดลยูเอ็มแอล
2. แปลงโมเดลของแผนภาพซีเควนซ์ไปเป็นกฎการแปลง ตามสคีมาของกฎการแปลง (Rule schema) ที่งานวิจัยนี้ได้สร้างขึ้นมาทั้งสิ้น 8 กฎ โดยกฎเหล่านี้อยู่ในรูปแบบของไวยากรณ์ 2 ชั้น (Two-level grammars)

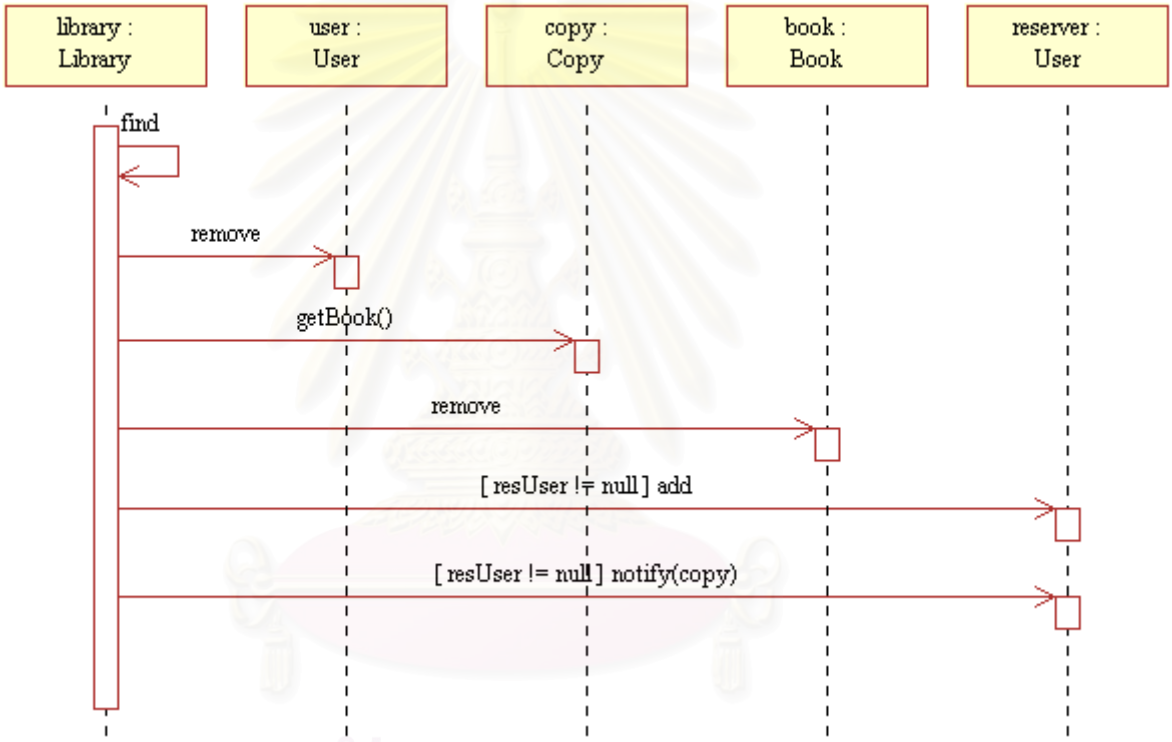
โดยผู้ทำวิทยานิพนธ์จะให้ความสนใจแต่เฉพาะขั้นตอนที่ 2 เนื่องจากข้อมูลเข้าของเครื่องมือที่จะทำการพัฒนาขึ้นมานั้นอยู่ในรูปของโมเดลยูเอ็มแอลอยู่แล้ว

เมื่อศึกษาสคีมาของกฎการแปลง พบว่างานวิจัยนี้มีข้อจำกัดดังนี้

1. ไม่สามารถทำการรวมพฤติกรรมของแต่ละโอเปอร์เรชันที่ปรากฏอยู่ในหลายสถานการณ์ได้
2. จำกัดระดับการซ้อนกันของการส่งเมสเสจไว้ไม่เกินระดับที่สอง
3. การอ้างถึงตัวแปรจะใช้ชื่อตัวแปรจากแผนภาพซีเควนซ์เสมอ ซึ่งจะทำให้เกิดปัญหาในกรณีที่ชื่อของพารามิเตอร์ของการส่งเมสเสจที่เป็นตัวกระตุ้นในแผนภาพซีเควนซ์ ไม่ตรงกันกับชื่อพารามิเตอร์ของโอเปอร์เรชันจากแผนภาพคลาสที่มีความสัมพันธ์กับการส่งเมสเสจนั้นอยู่

ข้อจำกัดดังกล่าวของงานวิจัยนี้ ทำให้ยังไม่สามารถนำผลจากงานวิจัยนี้ไปประยุกต์ใช้งานจริงได้อย่างสมบูรณ์นัก อย่างไรก็ดี ผู้ทำวิทยานิพนธ์มีความเห็นว่าสามารถนำกฎทั้ง 8 ข้อนี้ไปเป็นพื้นฐานของแนวคิดในการออกแบบกฎการแปลง ฯ ได้เป็นอย่างดี

2.2.2 สคีมาตาของอินเทอร์แรคชัน : การคอมไพล์อินเทอร์แรคชันไปเป็นรหัสคำสั่ง (Interaction Schemata: Compiling Interactions to Code) [14] โดย Neeraj Sangal, Edward Farrell, Karl Lieberherr, and David Lorenz



รูปที่ 2.6 แผนภาพซีเควนซ์ที่งานวิจัยในหัวข้อที่ 2.2.2 นำมาเป็นตัวอย่าง

ในงานวิจัยนี้ได้กล่าวถึงความไม่เหมาะสมที่จะทำการแปลงแผนภาพอินเทอร์แรคชันไปเป็นรหัสคำสั่งโดยตรง โดยได้ยกตัวอย่างถึงความไม่สมบูรณ์ของข้อมูลที่ได้จากแผนภาพซีเควนซ์ตามรูปที่ 2.6 ดังนี้

1. ไม่สามารถระบุที่มาของวัตถุที่ปรากฏในแผนภาพซีเควนซ์ได้
2. ไม่สามารถให้ข้อมูลการส่งผ่านวัตถุระหว่างโอเปอร์เรชันต่าง ๆ ได้ดีนัก ทำให้ผู้อ่านแผนภาพต้องอาศัยการคาดคะเนรายละเอียดในการส่งผ่านวัตถุระหว่างโอเปอร์เรชันด้วยตนเอง

3. ขาดรายละเอียดในส่วนของการทำซ้ำ พารามิเตอร์และชนิดของข้อมูลส่งกลับ การแสดงตรรกะ (Boolean expression) ของเงื่อนไข และ รหัสคำสั่งอื่น ๆ เพิ่มเติมที่ไม่เกี่ยวข้องกับการส่งเมสเสจ

โดยงานวิจัยนี้ได้นำเสนอการสร้างและทำการแปลงสคีมาของอินเทอร์แรคชันแทนแผนภาพอินเทอร์แรคชัน ซึ่งสคีมาของอินเทอร์แรคชันนี้มีลักษณะเป็นข้อความที่อธิบายถึงอินเทอร์แรคชันของวัตถุ และสามารถถูกแปลงไปเป็นรหัสคำสั่งได้อย่างสมบูรณ์ รวมถึงมีการเพิ่มการระบุการกระทำโดยทั่ว ๆ ไปกับโครงสร้างข้อมูล ได้แก่ การวนซ้ำข้อมูล การค้นหาข้อมูล การเพิ่มข้อมูล และการเอาข้อมูลออก ไว้ได้ในสคีมานี้ ซึ่งจะสามารถถูกนำไปสร้างเป็นรหัสคำสั่งได้อย่างเหมาะสม

อย่างไรก็ตาม ในความเห็นของผู้ทำวิทยานิพนธ์ หากนักออกแบบซอฟต์แวร์เขียนแผนภาพซีเควนซ์ให้ละเอียดพอ แผนภาพซีเควนซ์นั้นก็จะสามารถให้ข้อมูลได้ค่อนข้างที่จะครบถ้วนได้ ดังนี้

1. ที่มาของวัตถุที่ปรากฏบนแผนภาพซีเควนซ์สามารถดูได้จากตัวแปรที่เป็นพารามิเตอร์ ตัวแปรรับค่าส่งกลับของการส่งเมสเสจ การส่งเมสเสจแบบสร้างวัตถุ และ แอทริบิวต์ของคลาสจากแผนภาพคลาส
2. ข้อมูลการส่งผ่านวัตถุระหว่างโอเปอร์เรชันต่าง ๆ สามารถแสดงได้โดยให้วัตถุที่จะส่งผ่านเป็นพารามิเตอร์ของการส่งเมสเสจ
3. รายละเอียดในส่วนของการทำซ้ำ การแสดงตรรกะของเงื่อนไข สามารถระบุลงไปในแผนภาพซีเควนซ์ตามภาษาที่ใช้อิมพลีเมนต์ได้
4. ชนิดของข้อมูลส่งกลับ สามารถดูได้จากแผนภาพคลาส
5. การระบุการกระทำโดยทั่ว ๆ ไปกับโครงสร้างข้อมูล สามารถแสดงได้โดยการส่งเมสเสจไปยังโอเปอร์เรชันต่าง ๆ ของวัตถุที่ใช้เป็นโครงสร้างข้อมูลนั้นได้โดยตรง

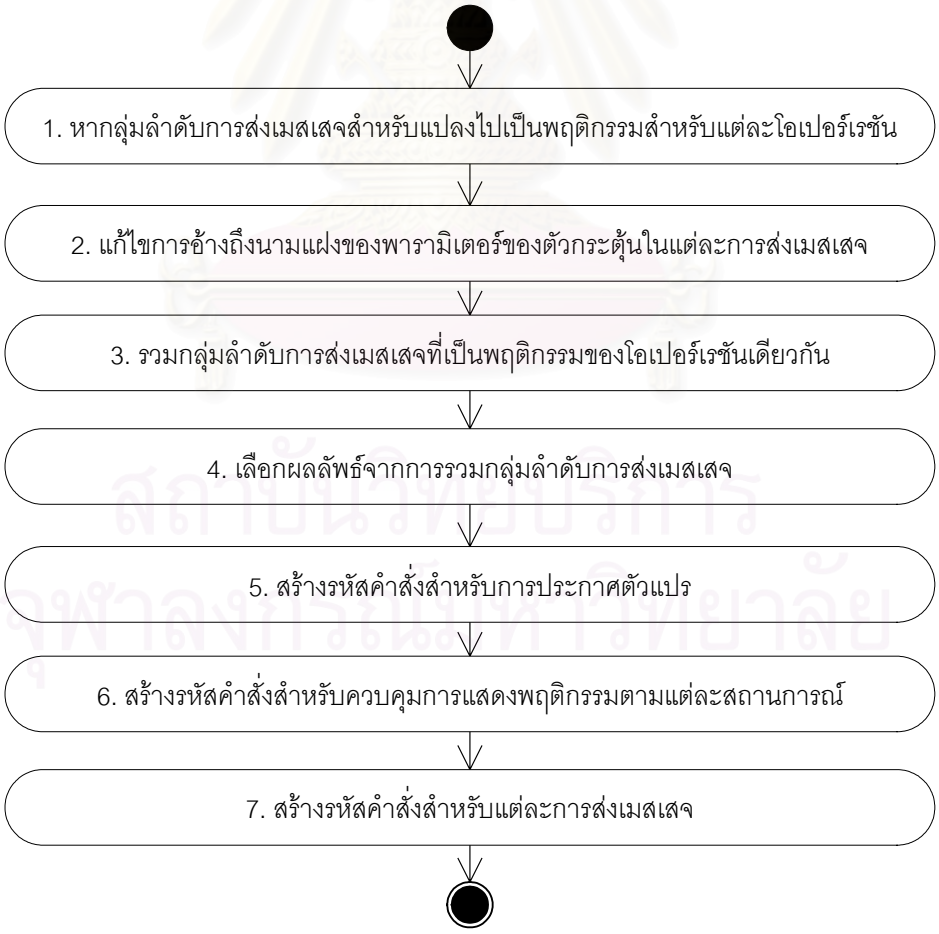
โดยจะขาดก็เพียงแต่การระบุรหัสคำสั่งอื่น ๆ เพิ่มเติมที่ไม่เกี่ยวกับการส่งเมสเสจที่ผู้พัฒนาจะต้องไปเขียนเพิ่มเติมในรหัสคำสั่งเอง ซึ่งเมื่อพิจารณาถึงการที่ผู้พัฒนาจะต้องเรียนรู้และเสียเวลาในการสร้างสคีมาของอินเทอร์แรคชันจากแผนภาพซีเควนซ์ก่อนที่จะทำการแปลงไปเป็นรหัสคำสั่งแล้ว อาจไม่มีความจำเป็นมากพอที่จะต้องใช้สคีมาของอินเทอร์แรคชันในการแปลงไปเป็นรหัสคำสั่งตามงานวิจัยนี้สำหรับการพัฒนาซอฟต์แวร์ในกรณีทั่ว ๆ ไป นอกจากนั้น ในงาน

วิจัยนี้ผู้ใช้จะต้องทำการรวมพฤติกรรมของแต่ละโอเปอร์เรชันที่ปรากฏอยู่ในสถานการณ์ต่าง ๆ จากแผนภาพอินเทอร์แรคชันไปเป็นสคีมาของอินเทอร์แรคชันด้วยตนเอง

บทที่ 3
การออกแบบขั้นตอนและกฎการแปลง ฯ

ในบทนี้จะเป็นการกล่าวถึงการออกแบบขั้นตอนและกฎการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา โดยขั้นตอนและกฎการแปลง ฯ ที่ออกแบบขึ้นมานี้ จะต้องสามารถแปลงแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันหลายระดับ และสามารถทำการรวมพฤติกรรมของแต่ละโอเปอร์เรชันจากสถานการณ์ต่าง ๆ ได้

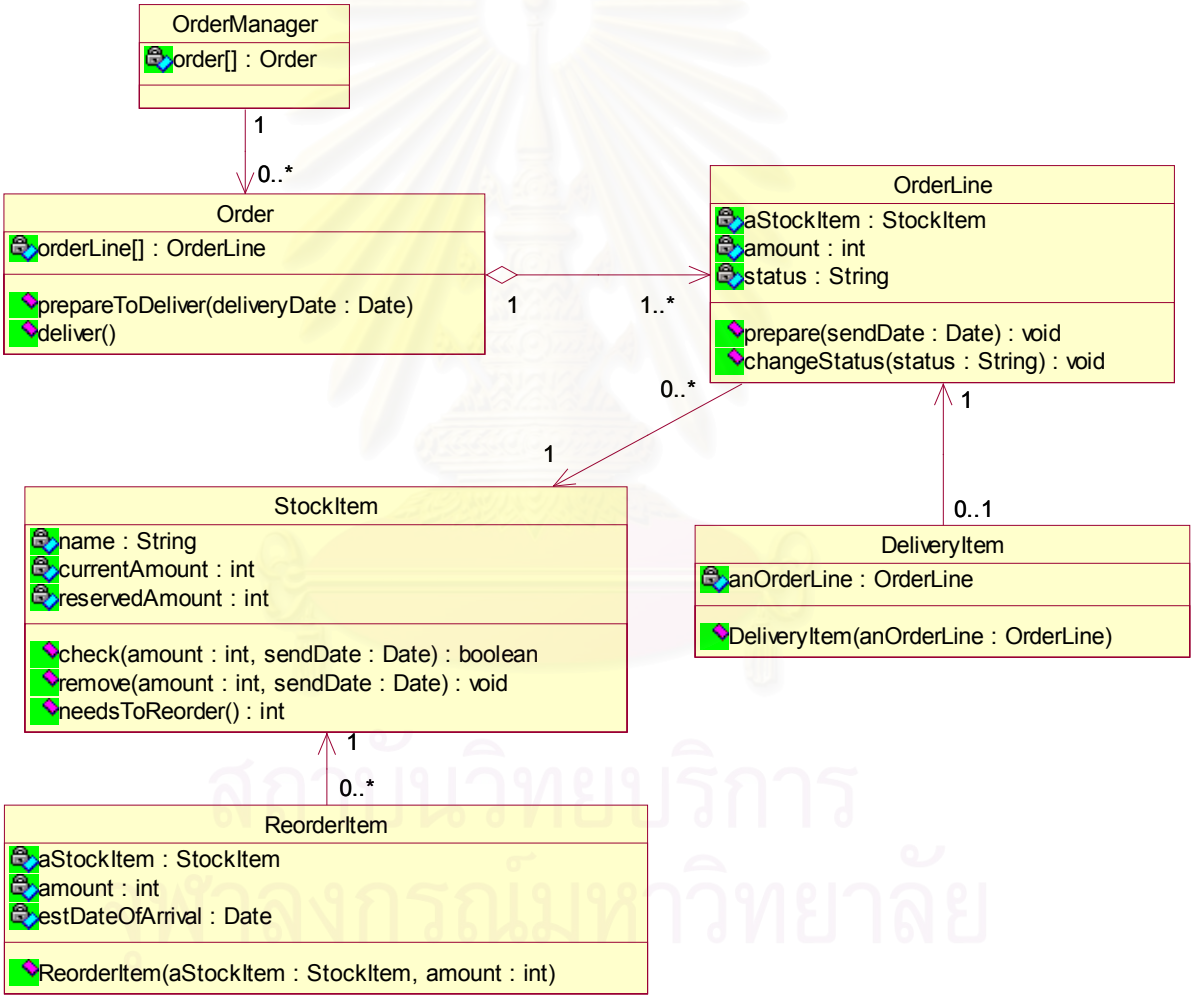
ในการทำการแปลงดังกล่าวนั้น นอกจากจะต้องใช้ข้อมูลจากแผนภาพซีเควนซ์แล้ว ยังต้องอาศัยข้อมูลเชิงโครงสร้างจากแผนภาพคลาสมาประกอบด้วย และเนื่องจากกระบวนการแปลง ฯ ทั้งหมดมีความซับซ้อนสูงเกินกว่าที่จะทำการแปลง ฯ ได้ในขั้นตอนเดียว จึงจะต้องทำการกระจายความซับซ้อนของการแปลง ฯ ออกไปด้วยการออกแบบให้ขั้นตอนการแปลง ฯ มีหลายขั้นตอนดังแสดงในรูปที่ 3.1



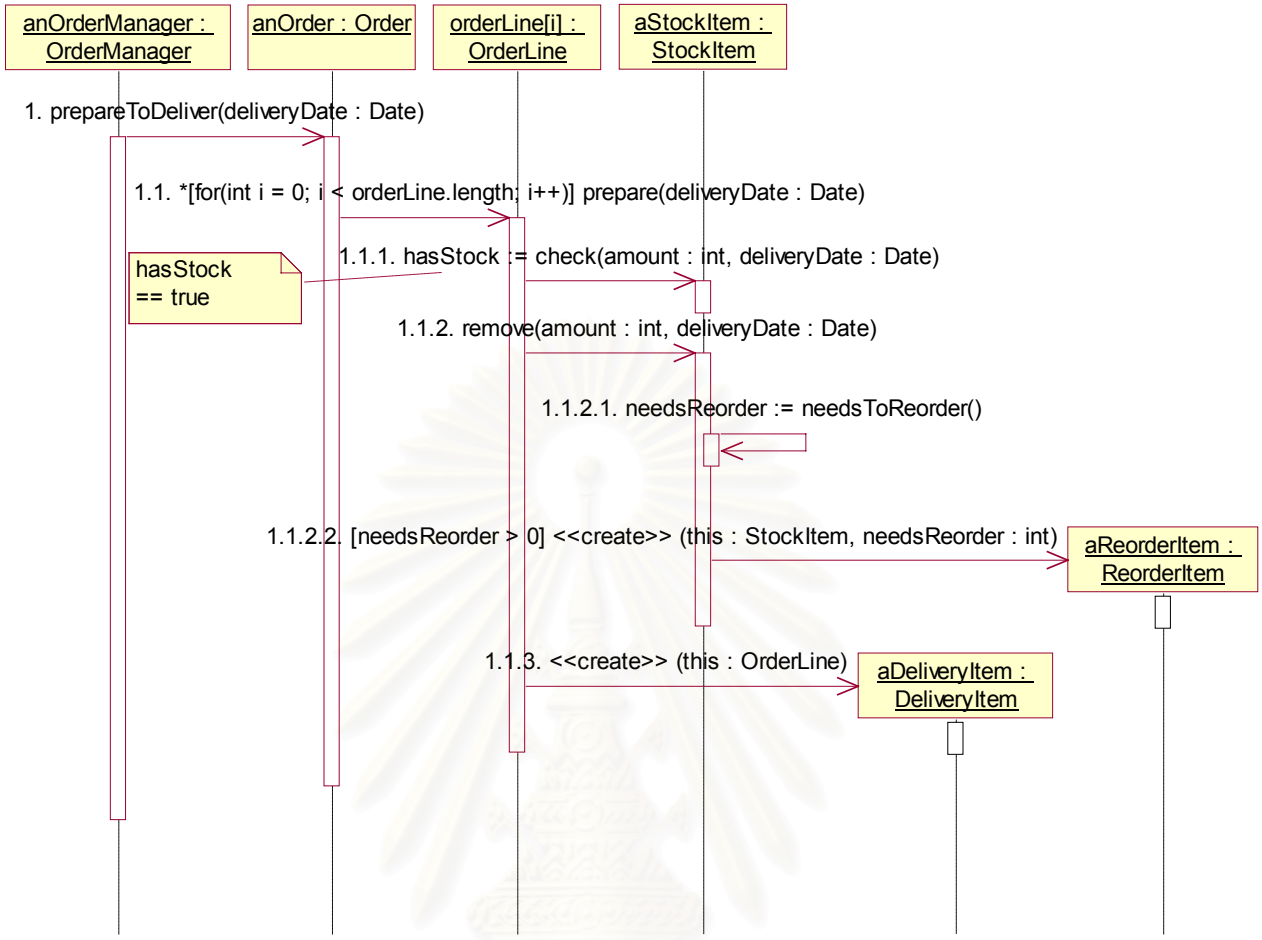
รูปที่ 3.1 ขั้นตอนต่าง ๆ ในการทำการแปลง ฯ

รูปที่ 3.2 ถึง รูปที่ 3.4 เป็นแผนภาพคลาสและแผนภาพซีเควนซ์ของส่วนเตรียมการจัดส่งสินค้าของระบบสั่งซื้อสินค้าซึ่งได้ดัดแปลงมาจากหนังสือ UML Distilled [22] ซึ่งจะใช้แผนภาพดังกล่าวเป็นตัวอย่างประกอบการอธิบายขั้นตอนต่าง ๆ ในการแปลง ฯ โดยตัวอย่างนี้ประกอบไปด้วยสถานการณ์ 2 สถานการณ์ ได้แก่

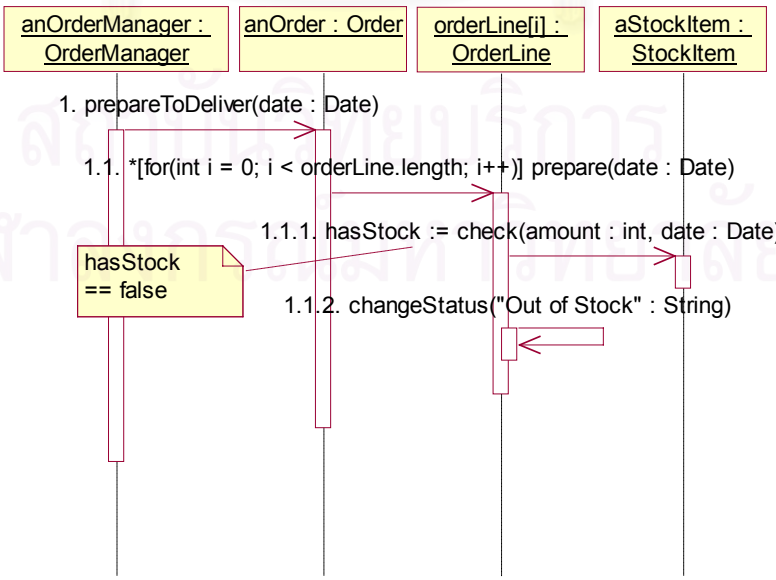
1. เตรียมการจัดส่งสินค้าได้สำเร็จ (สถานการณ์ปกติ) แสดงด้วยรูปที่ 3.3
2. ไม่มีสินค้าเหลือเพียงพอที่จะจัดส่ง แสดงด้วยรูปที่ 3.4



รูปที่ 3.2 แผนภาพคลาสของระบบสั่งซื้อสินค้า (ส่วนเตรียมการจัดส่งสินค้า)



รูปที่ 3.3 แผนภาพซีเควนซ์แสดงสถานการณ์ที่เตรียมการจัดส่งสินค้าได้สำเร็จ (สถานการณ์ปกติ)



รูปที่ 3.4 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ไม่มีสินค้าเหลือเพียงพอที่จะจัดส่ง

3.1 การหากลุ่มลำดับการส่งเมสเสจสำหรับแปลงไปเป็นพฤติกรรมสำหรับแต่ละโอเปอร์เรชัน

ก่อนที่จะหาได้ว่าพฤติกรรมในแต่ละโอเปอร์เรชันจะได้มาจากข้อมูลการส่งเมสเสจใดบ้าง จะต้องรู้จักว่าตัวกระตุ้น (Activator) คืออะไรเสียก่อนตามนิยามดังนี้

นิยามที่ 1 ตัวกระตุ้นของการส่งเมสเสจใด ๆ คือการส่งเมสเสจตัวที่ก่อให้เกิดการส่งเมสเสจนั้น [15]

ในแผนภาพซีเควนซ์ จะสามารถหาตัวกระตุ้นของแต่ละการส่งเมสเสจได้โดยง่ายจากการสังเกตว่าเมสเสจนั้นถูกส่งออกมาจากโฟกัสออฟคอนโทรลตัวใด โดยตัวกระตุ้นของการส่งเมสเสจใด ๆ ก็คือการส่งเมสเสจตัวที่ก่อให้เกิดโฟกัสออฟคอนโทรลที่เมสเสจตัวนั้นถูกส่งออกนั่นเอง

โดยกลุ่มลำดับของการส่งเมสเสจที่มีตัวกระตุ้นเป็นตัวเดียวกัน (ถูกส่งออกมาจากโฟกัสออฟคอนโทรลตัวเดียวกัน) จะเป็นพฤติกรรมของตัวกระตุ้นนั้น ซึ่งสามารถนำไปแปลงเป็นพฤติกรรมของโอเปอร์เรชันที่มีความสัมพันธ์กับตัวกระตุ้นนั้นได้ ทำให้ได้กฎในหากลุ่มลำดับการส่งเมสเสจสำหรับแปลงไปเป็นพฤติกรรมสำหรับแต่ละโอเปอร์เรชันดังนี้

กฎที่ 1 พฤติกรรมของโอเปอร์เรชันใด ๆ จะประกอบไปด้วยกลุ่มลำดับการส่งเมสเสจที่มีตัวกระตุ้นที่มีความสัมพันธ์อยู่กับโอเปอร์เรชันนั้น

เมื่อประยุกต์ใช้กฎที่ 1 กับแผนภาพซีเควนซ์ในรูปที่ 3.3 และ 3.4 พฤติกรรมของโอเปอร์เรชัน “prepareToDeliver” ในคลาส “Order” จะประกอบไปด้วยการส่งเมสเสจที่ 1.1 จากทั้งสองแผนภาพ เนื่องจากการส่งเมสเสจดังกล่าวมีตัวกระตุ้นเป็นการส่งเมสเสจที่ 1 ซึ่งมีความสัมพันธ์อยู่กับโอเปอร์เรชัน “prepareToDeliver” ของคลาส “Order” นั่นเอง ในทำนองเดียวกัน พฤติกรรมของโอเปอร์เรชัน “prepare” ของคลาส “OrderLine” จะประกอบไปด้วยกลุ่มลำดับการส่งเมสเสจที่ 1.1.1 1.1.2 และ 1.1.3 จากรูปที่ 3.3 และกลุ่มลำดับการส่งเมสเสจที่ 1.1.1 และ 1.1.2 จากรูปที่ 3.4

3.2 การแก้ไขการอ้างถึงนามแฝงของพารามิเตอร์ของตัวกระตุ้นในแต่ละการส่งเมสเสจ

เมื่อเปรียบเทียบแผนภาพซีเควนซ์ในรูปที่ 3.3 กับรูปที่ 3.4 จะพบว่า การส่งเมสเสจที่ 1.1 ของทั้งสองแผนภาพนั้นดูเหมือนว่ามีความแตกต่างกันและไม่สามารถนำมารวมกันเป็นการส่งเมสเสจเดียวกันได้เนื่องจากมีชื่อพารามิเตอร์ (Parameter) แตกต่างกัน แต่ความจริงแล้วการส่งเมสเสจตัวที่ 1.1 ของทั้งสองแผนภาพนี้ถือว่าเหมือนกันและสามารถรวมกันได้ ทั้งนี้เป็นเพราะชื่อพารามิเตอร์ที่ใช้ในการส่งเมสเสจที่ 1 ในรูปที่ 3.4 ไม่ตรงกันกับชื่อของพารามิเตอร์ตามที่ปรากฏในแผนภาพคลาสตามรูปที่ 3.2 จึงยังผลให้การอ้างถึงพารามิเตอร์ภายใต้การส่งเมสเสจที่ 1.1 ที่มีการส่งเมสเสจที่ 1 เป็นตัวกระตุ้นในรูปที่ 3.4 ไม่ตรงกับชื่อของพารามิเตอร์นั้นตามที่ปรากฏในแผนภาพคลาสไปด้วย ดังนั้นก่อนที่จะทำการรวมการส่งเมสเสจเพื่อเตรียมแปลงไปเป็นรหัสคำสั่งจะต้องทำการแก้ไขชื่อพารามิเตอร์ที่ถูกอ้างถึงเหล่านี้ให้ตรงกับชื่อตามแผนภาพคลาสเสียก่อน เพื่อให้สามารถทำการรวมการส่งเมสเสจได้อย่างถูกต้อง และทำให้รหัสคำสั่งที่ได้จากการแปลงการส่งเมสเสจมีความสอดคล้องกันกับรหัสคำสั่งเชิงโครงสร้างที่ได้จากแผนภาพคลาส

ในงานวิจัยนี้จะเรียกชื่อพารามิเตอร์ตามที่ปรากฏในแผนภาพซีเควนซ์ว่า “นามแฝง” ส่วนชื่อพารามิเตอร์ที่แท้จริงนั้น จะต้องดูจากแผนภาพคลาส ตามนิยามดังต่อไปนี้

นิยามที่ 2.1 ชื่อของพารามิเตอร์ใด ๆ คือชื่อของพารามิเตอร์นั้นตามที่ปรากฏในแผนภาพคลาส

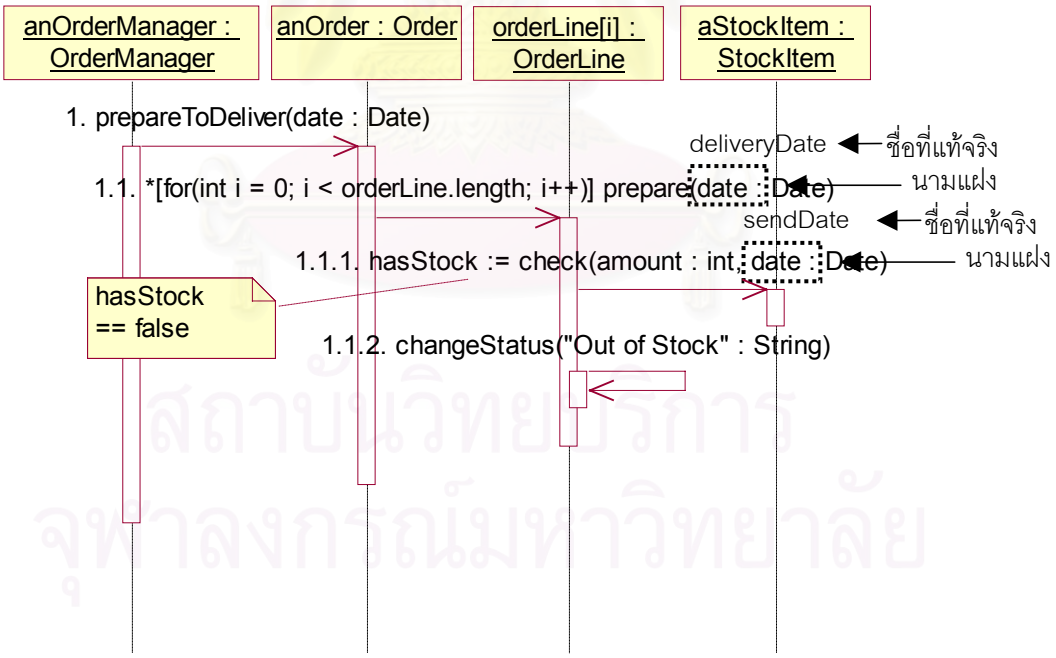
นิยามที่ 2.2 นามแฝงของพารามิเตอร์ใด ๆ คือชื่อของพารามิเตอร์นั้นตามที่ปรากฏในแผนภาพซีเควนซ์ที่กำลังอ้างถึง

จากแผนภาพคลาสในรูปที่ 3.2 โอเปอร์เรชัน “prepareToDeliver” ของคลาส “Order” ในแผนภาพคลาสระบุว่ามีพารามิเตอร์ที่ชื่อ “deliveryDate” ดังนั้นการส่งเมสเสจตัวที่ 1 ในแผนภาพซีเควนซ์ตามรูปที่ 3.4 จะมีชื่อของพารามิเตอร์เป็น “deliveryDate” โดยมีนามแฝงของพารามิเตอร์นี้เป็น “date” ส่วนโอเปอร์เรชัน “prepare” ของคลาส “OrderLine” ในแผนภาพคลาสระบุว่ามีพารามิเตอร์ที่ชื่อ “sendDate” ดังนั้นการส่งเมสเสจตัวที่ 1.1 ในแผนภาพซีเควนซ์ตามรูปที่ 3.4 จะมีชื่อของพารามิเตอร์เป็น “sendDate” โดยมีนามแฝงของพารามิเตอร์นี้เป็น “date”

หลังจากที่มีนิยามชื่อและนามแฝงของพารามิเตอร์แล้ว ต่อไปจะต้องทำการแก้ไขการอ้างถึงพารามิเตอร์ต่าง ๆ ให้ตรงกับชื่อที่แท้จริงของพารามิเตอร์นั้น ตามกฎดังต่อไปนี้

> กฎที่ 2 การส่งเมสเสจใด ๆ หากมีการอ้างถึงพารามิเตอร์ของตัวกระตุ้นด้วยนามแฝงในการส่งเมสเสจนั้นหรือในวัตถุที่เมสเสจนั้นส่งถึง จะต้องแทนที่นามแฝงนั้นด้วยชื่อที่แท้จริงของพารามิเตอร์ที่กำลังอ้างถึง

รูปที่ 3.5 เป็นการแสดงการอ้างถึงพารามิเตอร์ของตัวกระตุ้นด้วยนามแฝงในแผนภาพซีเควนซ์ตามรูปที่ 3.4 ซึ่งจากรูปดังกล่าวจะสังเกตเห็นได้ว่า การส่งเมสเสจที่ 1.1 มีการอ้างถึงพารามิเตอร์ของตัวกระตุ้น (การส่งเมสเสจที่ 1) ด้วยนามแฝงคือ “date” ซึ่งจะต้องแทนที่ด้วยชื่อที่แท้จริงคือ “deliveryDate” ส่วนการส่งเมสเสจที่ 1.1.1 มีการอ้างถึงพารามิเตอร์ของตัวกระตุ้น (การส่งเมสเสจที่ 1.1) ด้วยนามแฝงคือ “date” ซึ่งจะต้องแทนที่ด้วยชื่อที่แท้จริงคือ “sendDate”



รูปที่ 3.5 การอ้างถึงพารามิเตอร์ของตัวกระตุ้นด้วยนามแฝงในแผนภาพซีเควนซ์ตามรูปที่ 3.4

3.3 การรวมกลุ่มลำดับการส่งเมสเสจที่เป็นพฤติกรรมของโอเปอร์เรชันเดียวกัน

จากกฎที่ 1 "พฤติกรรมของโอเปอร์เรชันใด ๆ จะประกอบไปด้วยกลุ่มลำดับการส่งเมสเสจที่มีตัวกระตุ้นที่มีความสัมพันธ์อยู่กับโอเปอร์เรชันนั้น" ซึ่งความสัมพันธ์ระหว่างโอเปอร์เรชันกับตัวกระตุ้นดังกล่าวสามารถเป็นแบบหนึ่งโอเปอร์เรชันต่อหลายตัวกระตุ้นได้ (1 to many) เช่นในกรณีที่มีหลายสถานการณ์ ดังนั้นในการทำการแปลง ฯ จะต้องมีการรวมกลุ่มลำดับการส่งเมสเสจที่มีตัวกระตุ้นแตกต่างกันแต่มีความสัมพันธ์กับโอเปอร์เรชันเดียวกันตามกฎในการรวมดังต่อไปนี้

กฎที่ 3 สำหรับโอเปอร์เรชันใด ๆ ถ้ามีกลุ่มลำดับการส่งเมสเสจที่มีตัวกระตุ้นที่มีความสัมพันธ์อยู่กับโอเปอร์เรชันนั้นมากกว่า 1 กลุ่มลำดับ จะต้องทำการรวมกลุ่มลำดับการส่งเมสเสจเหล่านั้นเข้าด้วยกัน โดยที่จะต้องรักษาลำดับของแต่ละการส่งเมสเสจของผลการรวมให้ถูกต้องเมื่อเทียบกับกลุ่มลำดับการส่งเมสเสจดั้งเดิมแต่ละกลุ่มลำดับ และอาจทำการรวมการส่งเมสเสจที่เหมือนกันจากคนละกลุ่มลำดับการส่งเมสเสจเข้าเป็นการส่งเมสเสจเดียวกันได้

นิยามที่ 3 การส่งเมสเสจที่เหมือนกัน คือการส่งเมสเสจที่มีความสัมพันธ์กับโอเปอร์เรชันเดียวกันและมีเงื่อนไขการส่งเมสเสจ ลายเซ็นของเมสเสจที่ส่ง วัตถุที่เมสเสจนั้นส่งถึง ตัวแปรรับค่าส่งกลับเหมือนกัน

3.4 การเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ

ผลลัพธ์ที่เกิดจากการรวมกลุ่มลำดับการส่งเมสเสจสามารถมีได้หลายผลลัพธ์ ตัวอย่างเช่นในรูปที่ 3.6 เป็นการแสดงตัวอย่างการเกิดผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ 2 กลุ่มลำดับทั้งหมด 3 แบบ โดยในรูปนี้จะใช้ลูกบอลสีดำแทนการส่งเมสเสจในกลุ่มลำดับการส่งเมสเสจที่ 1 และใช้ลูกบอลสีขาวแทนการส่งเมสเสจในกลุ่มลำดับการส่งเมสเสจที่ 2 ส่วนการส่งเมสเสจที่สามารถรวมกันได้ตามกฎที่ 3 คือลูกบอลที่มีสีต่างกันที่มีตัวอักษรกำกับเหมือนกัน

จากรูปดังกล่าว จะสังเกตเห็นได้ว่าการเกิดผลลัพธ์ได้หลายผลลัพธ์ของรูปนี้ เป็นผลเนื่องมาจากการส่งเมสเสจ a ของกลุ่มลำดับการส่งเมสเสจที่ 2 สามารถเลือกที่จะรวมกับการส่งเมสเสจ a ของกลุ่มลำดับการส่งเมสเสจที่ 1 ตัวแรก (ได้เป็นผลการรวมแบบที่ 1) หรือเลือกที่จะรวมกับการส่งเมสเสจ a ของกลุ่มลำดับการส่งเมสเสจที่ 1 ตัวหลัง (ได้เป็นผลการรวมแบบที่ 3)

หรืออาจเลือกที่จะไม่รวมกับการส่งเมสเสจใดเลย (ได้เป็นผลการรวมแบบที่ 2) ก็ได้เช่นเดียวกัน ดังนั้นจะต้องมีวิธีในการเลือกผลลัพธ์ที่เหมาะสมที่สุด

รูปที่ 3.6 ตัวอย่างการเกิดผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจได้หลายผลลัพธ์

ในงานวิจัยนี้ได้มีการคิดค้นมาตรวัดประกอบการเลือกผลลัพธ์ขึ้นมา 2 มาตรวัดดังรายละเอียดในตารางที่ 3.1 และ 3.2 โดยมาตรวัดทั้งสองดังกล่าวจะบ่งบอกถึง ขนาด ความซ้ำซ้อน และความซับซ้อนของรหัสคำสั่งที่จะได้ โดยรหัสคำสั่งที่มีขนาด ความซ้ำซ้อน และความซับซ้อนน้อย ๆ จะส่งผลให้นักพัฒนาทำความเข้าใจและทำการแก้ไขหรือเพิ่มเติมรหัสคำสั่งได้ง่าย

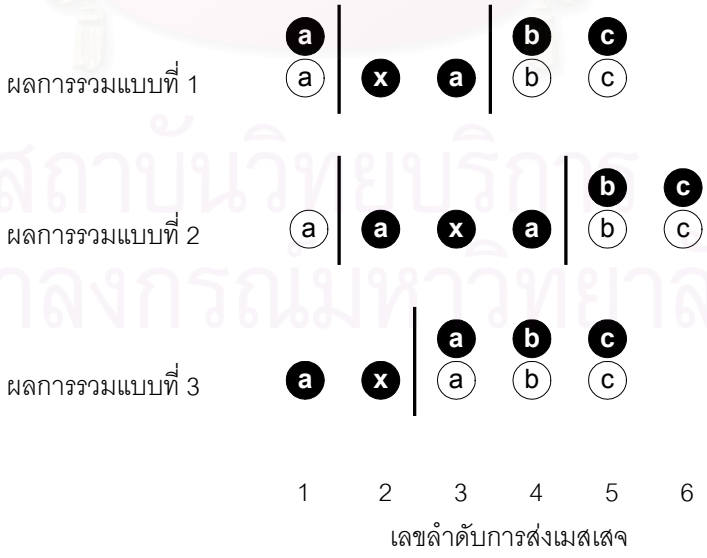
ตารางที่ 3.1 มาตรวัดจำนวนการส่งเมสเสจ

ชื่อมาตรวัด	จำนวนการส่งเมสเสจ
วิธีวัด	นับจำนวนการส่งเมสเสจในผลการรวมกลุ่มลำดับการส่งเมสเสจ
คำอธิบาย	ผลลัพธ์ที่มีจำนวนการส่งเมสเสจน้อยกว่า จะมีขนาดของรหัสคำสั่งและความซ้ำซ้อนของรหัสคำสั่งน้อยกว่าด้วย

ตารางที่ 3.2 มาตรวัดความซับซ้อนของสถานการณ์

ชื่อมาตรวัด	ความซับซ้อนของสถานการณ์
วิธีวัด	นับจำนวนคู่การส่งเมสเสจติดกันที่มีที่มาจากกลุ่มของตัวกระตุ้นที่แตกต่างกัน
คำอธิบาย	ผลลัพธ์ที่มีความซับซ้อนของสถานการณ์น้อยกว่า จะมีความซับซ้อนของรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์น้อยกว่าด้วย เนื่องจากเมื่อมีการเปลี่ยนแปลงกลุ่มของตัวกระตุ้นที่เป็นที่มาของการส่งเมสเสจ จะต้องมีการสร้างรหัสคำสั่งขึ้นมาเพื่อควบคุมการแสดงพฤติกรรมตามสถานการณ์ที่สอดคล้องกันกับกลุ่มของตัวกระตุ้นที่มีการเปลี่ยนแปลงไปดังกล่าว โดยรายละเอียดของการสร้างรหัสคำสั่งดังกล่าวจะอยู่ในหัวข้อที่ 3.6

จากตัวอย่างผลการรวมกลุ่มลำดับการส่งเมสเสจในรูปที่ 3.6 จะสามารถแสดงการหาค่าของมาตรวัดทั้งสองได้ด้วยรูปที่ 3.7 โดยเลขลำดับการส่งเมสเสจด้านล่างของรูปเป็นการแสดงการนับจำนวนการส่งเมสเสจ ดังนั้นมาตรวัดจำนวนการส่งเมสเสจจะสามารถหาได้จากเลขลำดับการส่งเมสเสจที่มากที่สุดในผลการรวมนั้น และเส้นกั้นระหว่างคู่ของการส่งเมสเสจจะเป็นการแสดงคู่ของการส่งเมสเสจติดกันที่มีที่มามาจากกลุ่มของตัวกระตุ้นที่แตกต่างกัน ดังนั้นมาตรวัดความซับซ้อนของสถานการณ์จะสามารถหาได้จากการนับจำนวนเส้นกั้นในผลการรวมนั้นนั่นเอง ซึ่งผลที่ได้จากการหาค่าของมาตรวัดย่อยทั้ง 2 ของตัวอย่างนี้แสดงได้ดังตารางที่ 3.3



รูปที่ 3.7 ตัวอย่างหาค่าของมาตรวัดจากผลการรวมกลุ่มลำดับการส่งเมสเสจในรูปที่ 3.6

ตารางที่ 3.3 ผลการหาค่าของมาตรวัดจากผลการรวมกลุ่มลำดับการส่งเมสเสจในรูปที่ 3.6

ผลการรวม	มาตรวัด	
	จำนวนการส่งเมสเสจ	ความซับซ้อนของสถานการณ์
แบบที่ 1	5	2
แบบที่ 2	6	2
แบบที่ 3	5	1

โดยที่ผู้วิจัยให้ความสำคัญของคุณลักษณะของรหัสคำสั่งที่จะสร้างได้ไปที่ขนาดและความซ้ำซ้อนของรหัสคำสั่งมากกว่าความซับซ้อนของรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ จะได้กฎในการเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจดังต่อไปนี้

กฎที่ 4 ในกรณีที่มีผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจที่เป็นพฤติกรรมของโอเปอร์เรชันใด ๆ มีหลายผลลัพธ์ จะเลือกผลลัพธ์ที่มีจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์น้อยที่สุดโดยพิจารณาที่จำนวนการส่งเมสเสจก่อน ไปใช้ในขั้นตอนการแปลง ฯ ขั้นต่อไป

เมื่อพิจารณาค่าของจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์ในตารางที่ 3.3 ตามกฎดังกล่าว ผลลัพธ์ที่จะถูกเลือกไปใช้ในขั้นตอนการแปลง ฯ ขั้นต่อไป คือผลการรวมแบบที่ 3

3.5 การสร้างรหัสคำสั่งสำหรับการประกาศตัวแปร

การสร้างรหัสคำสั่งเพื่อประกาศตัวแปรที่ใช้ภายในแต่ละโอเปอร์เรชันโดยอัตโนมัติ จะเป็นไปตามกฎดังต่อไปนี้

กฎที่ 5 ในส่วนต้นของรหัสคำสั่งภายในแต่ละโอเปอร์เรชัน จะต้องทำการสร้างรหัสคำสั่งเพื่อประกาศตัวแปรที่ใช้ภายในโอเปอร์เรชันนั้นตามขั้นตอนดังต่อไปนี้

1. เลือกตัวแปรที่ทำหน้าที่รับค่า (ตัวแปรที่อยู่ทางซ้ายมือของเครื่องหมายมอบหมายค่า (=)) ทั้งหมดที่ไม่ซ้ำกันจากผลการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันนั้น โดยที่ตัวแปรนั้นจะต้องไม่เป็นพารามิเตอร์ของโอเปอร์เรชันดังกล่าว และชื่อของตัวแปรนั้นไม่มีเครื่องหมายมหัพภาค (.) ปรากฏอยู่
2. สร้างรหัสคำสั่งเพื่อประกาศตัวแปรสำหรับตัวแปรแต่ละตัวที่ได้จากขั้นตอนที่ 1 ตามรูปแบบดังนี้

```
ชนิดของตัวแปร  ชื่อของตัวแปร;
```

3.6 การสร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์

เนื่องจากการส่งเมสเสจแต่ละตัวในผลการรวมกลุ่มลำดับการส่งเมสเสจ สามารถมาจากกลุ่มของตัวกระตุ้นที่ไม่เหมือน จึงจะต้องมีการสร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมให้สอดคล้องกับแต่ละสถานการณ์ตามกฎที่ 6 โดยสถานการณ์แต่ละสถานการณ์จะบ่งบอกถึงตัวกระตุ้นแต่ละตัวของการส่งเมสเสจในผลการรวมกลุ่มลำดับการส่งเมสเสจ ในงานวิจัยนี้ได้มีการใช้ตัวแปรชื่อ $_scenario ในการระบุสถานการณ์ และมีการทดสอบค่าของตัวแปรนี้เพื่อคอยควบคุมพฤติกรรมให้เป็นไปตามแต่ละสถานการณ์อย่างถูกต้อง อย่างไรก็ตาม ผู้พัฒนาจะต้องระบุรหัสคำสั่งในการกำหนดค่าให้กับตัวแปร $_scenario ตามสถานการณ์ต่าง ๆ ด้วยตัวเอง

กฎที่ 6 ทำการสร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ไว้ภายในแต่ละโอเปอร์เรชัน โดยให้อยู่ต่อจากรหัสคำสั่งสำหรับการประกาศตัวแปร ตามขั้นตอนดังต่อไปนี้

1. ประกาศตัวแปร “$_scenario” เพื่อใช้ระบุสถานการณ์ โดยมีชนิดของข้อมูลเป็น “int” และมีค่าเริ่มต้นเป็น 0 ดังนี้

```
int $_scenario = 0;
```

2. สร้างคอมเมนต์ (Comment) บอกตัวเลขที่ใช้แทนสถานการณ์ต่าง ๆ ในผลการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันนั้น โดยตัวเลขที่ใช้แทนแต่ละสถานการณ์จะต้องมีค่าไม่ซ้ำกัน และอธิบายตัวเลขแต่ละตัวด้วยชื่อของแผนภาพซีเควนซ์ที่สอดคล้องกับสถานการณ์นั้น ตัวอย่างของคอมเมนต์สำหรับผลการรวมกลุ่มลำดับการส่งเมสเสจที่ประกอบด้วย 2 สถานการณ์จากแผนภาพซีเควนซ์ที่ชื่อ “Normal flow” และ “Exceptional flow” เป็นดังนี้

```
/**************************************
  Scenario numbers :
        # 0 = Normal flow
        # 1 = Exceptional flow
 **************************************/
```

3. สร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ โดยแต่ละการส่งเมสเสจในผลการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันนั้นที่ไม่ได้สอดคล้องกับทุกสถานการณ์ จะต้องมีสเตทเมนต์ (Statement) “if” ทำการทดสอบค่าของตัวแปร “$_scenario” ว่าตรงกับสถานการณ์ที่สอดคล้องกับการส่งเมสเสจนั้นหรือไม่ ถ้าตรงกันจึงจะทำรหัสคำสั่งที่สอดคล้องกับการส่งเมสเสจนั้น ตัวอย่างเช่น ถ้าการส่งเมสเสจ “a” มีที่มาจากสถานการณ์ที่แทนด้วยเลข 1 การส่งเมสเสจ “b” มีที่มาจากสถานการณ์ที่แทนด้วยเลข 0 และ 1 การส่งเมสเสจ “c” มีที่มาจากทุกสถานการณ์ จะสร้างรหัสคำสั่งได้ดังนี้

```
if ($_scenario == 1) (transformation of message “a”)
if ($_scenario == 0 || ($_scenario ==1) (transformation of message “a”)
(transformation of message “c”)
```

ตัวอย่างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ของผลการรวมกลุ่มลำดับการส่งเมสเสจแบบที่ 3 จากรูปที่ 3.6 แสดงได้ดังรูปที่ 3.8

```
int $_scenario = 0;

/****************************************
  Scenario numbers :
        # 0 = scenario1
        # 1 = scenario2
****************************************/

//Generated behavior from sequence diagrams
If ($_scenario == 0) {
        (transformation of message “a”)
        (transformation of message “x”)
}

(transformation of message “a”)
(transformation of message “b”)
(transformation of message “c”)
```

รูปที่ 3.8 ตัวอย่างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ของผลการรวมกลุ่มลำดับการส่งเมสเสจแบบที่ 3 จากรูปที่ 3.6

3.7 การสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจ

ในขั้นตอนนี้จะทำการแปลงแต่ละการส่งเมสเสจไปเป็นรหัสคำสั่ง โดยมีกฎในการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจดังนี้

กฎที่ 7 ทำการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจให้สอดคล้องกับประเภทการส่งเมสเสจ โดยมีรูปแบบของรหัสคำสั่งสำหรับการส่งเมสเสจประเภทต่าง ๆ ในกรณีทั่วไปดังนี้

สำหรับการส่งเมสเสจระหว่างวัตถุ

```
if (เงื่อนไข) ตัวแปรรับค่าส่งกลับ = ชื่อวัตถุตัวรับเมสเสจ.ชื่อโอเปอร์เรชัน(รายการชื่อของพารามิเตอร์);
```

สำหรับการส่งเมสเสจภายในวัตถุ

```
if (เงื่อนไข) ตัวแปรรับค่าส่งกลับ =  ชื่อโอเปอร์เรชัน(รายการชื่อของพารามิเตอร์);
```

สำหรับการส่งเมสเสจสร้างวัตถุ

```
if (เงื่อนไข) ชื่อวัตถุที่ถูกสร้าง =  new ชนิดของวัตถุที่ถูกสร้าง(รายการชื่อของพารามิเตอร์);
```

สำหรับการส่งเมสเสจส่งกลับ

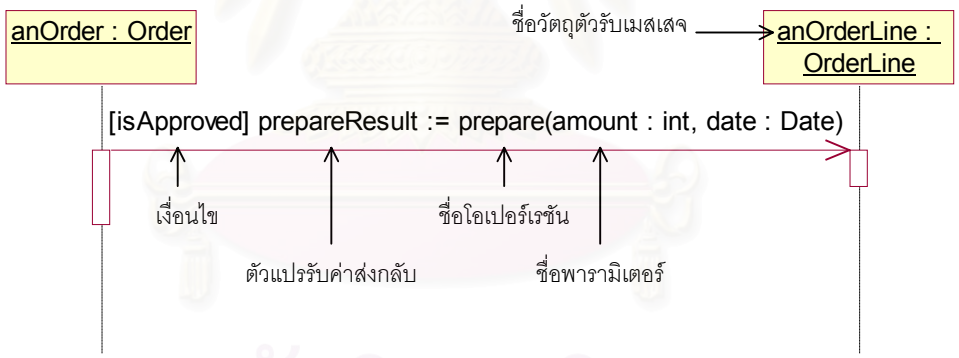
if (เงื่อนไข) return ค่าส่งกลับ;

ในกรณีที่การส่งเมสเสจใด ๆ ไม่มีการระบุเงื่อนไข จะต้องตัดส่วน “if (เงื่อนไข)” ออกไป

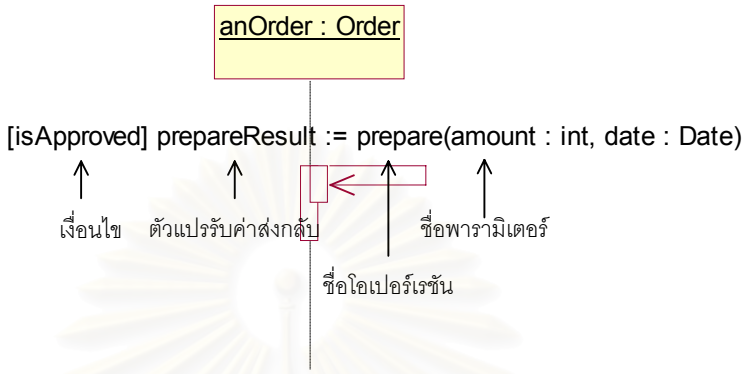
ในกรณีที่เงื่อนไขของการส่งเมสเสจเป็นแบบทำซ้ำ จะต้องแทนที่ “if (เงื่อนไข)” ด้วย “เงื่อนไข”

ในกรณีที่การส่งเมสเสจระหว่างวัตถุหรือภายในวัตถุใด ๆ ไม่มีตัวแปรรับค่าส่งกลับ จะต้องตัดส่วน “ตัวแปรรับค่าส่งกลับ =” ออกไป

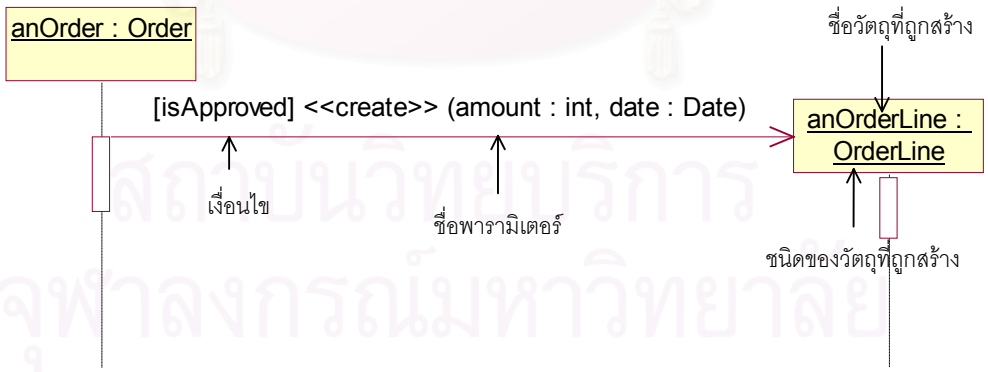
โดยตัวอย่างของการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจตามกฎดังกล่าวสำหรับกรณีทั่วไป แสดงได้ด้วยรูปที่ 3.9 ถึง 3.12

ประเภทการส่งเมสเสจ	ระหว่างวัตถุ
	
รูปแบบของรหัสคำสั่งที่แปลงได้	if (เงื่อนไข) ตัวแปรรับค่าส่งกลับ = ชื่อวัตถุตัวรับเมสเสจ.ชื่อโอเปอร์เรชัน(รายการชื่อของพารามิเตอร์);
ตัวอย่างรหัสคำสั่ง	if (isApproved) prepareResult = anOrderLine.prepare(amount, date);

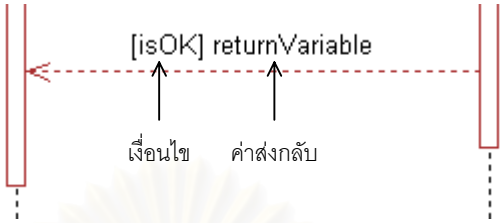
รูปที่ 3.9 รูปแบบและตัวอย่างการสร้างรหัสคำสั่งสำหรับการส่งเมสเสจระหว่างวัตถุ

ประเภทการส่งเมสเสจ	ภายในวัตถุ
	
รูปแบบของรหัสคำสั่งที่แปลงได้	if (เงื่อนไข) ตัวแปรรับค่าส่งกลับ = ชื่อโอเปอร์เรชัน(รายการชื่อของพารามิเตอร์);
ตัวอย่างรหัสคำสั่ง	if (isApproved) prepareResult = prepare(amount, date);

รูปที่ 3.10 รูปแบบและตัวอย่างการสร้างรหัสคำสั่งสำหรับการส่งเมสเสจภายในวัตถุ

ประเภทการส่งเมสเสจ	สร้างวัตถุ
	
รูปแบบของรหัสคำสั่งที่แปลงได้	if (เงื่อนไข) ชื่อวัตถุที่ถูกสร้าง = new ชนิดของวัตถุที่ถูกสร้าง(รายการชื่อของพารามิเตอร์);
ตัวอย่างรหัสคำสั่ง	if (isApproved) anOrderLine = new OrderLine(amount, date);

รูปที่ 3.11 รูปแบบและตัวอย่างการสร้างรหัสคำสั่งสำหรับการส่งเมสเสจสร้างวัตถุ

ประเภทการส่งเมสเสจ	ส่งกลับ
	
รูปแบบของรหัสคำสั่งที่แปลงได้	if (เงื่อนไข) return ค่าส่งกลับ;
ตัวอย่างรหัสคำสั่ง	if (isOK) return returnVariable;

รูปที่ 3.12 รูปแบบและตัวอย่างการสร้างรหัสคำสั่งสำหรับการส่งเมสเสจส่งกลับ

นอกจากนี้ เพื่อให้รหัสคำสั่งที่ได้อ่านง่ายขึ้น จะต้องมีการรวมเงื่อนไขที่เหมือนกันของการส่งเมสเสจที่อยู่ติดกัน ในกรณีที่เงื่อนไขนั้นไม่ได้เป็นเงื่อนไขแบบทำซ้ำ เช่น

```
if (isApproved) anOrderLine = new OrderLine(amount, date);
if (isApproved) prepareResult = anOrderLine.prepare(amount, date);
```

จะถูกรวมเงื่อนไขได้เป็น

```
if (isApproved) {
        anOrderLine = new OrderLine(amount, date);
        prepareResult = anOrderLine.prepare(amount, date);
}
```

3.8 ตัวอย่างผลการแปลง ฯ

ตัวอย่างของรหัสคำสั่งของโอเปอร์เรชัน “prepare” ของคลาส “OrderLine” ที่ได้จากการทำการแปลงตัวอย่างส่วนเตรียมการจัดส่งสินค้าของระบบสั่งซื้อสินค้าตามแผนภาพในรูปที่ 3.2 ถึง 3.4 ตามขั้นตอนและกฎการแปลง ฯ ที่ได้ออกแบบไว้ทั้งหมดในบทนี้ แสดงได้ดังรูปที่ 3.13

```
public void prepare(Date sendDate) {

        boolean hasStock;
        DeliveryItem aDeliveryItem;

        int $_scenario = 0;

        /**************************************
          Scenario numbers :
                # 0 = Normal flow
                # 1 = Exceptional flow (out of stock)
         **************************************/


        // Generated behaviour from sequence diagrams :

        hasStock = aStockItem.check(amount, sendDate);

        if ($_scenario == 1) {
                changeStatus("Out of Stock");
        }

        if ($_scenario == 0) {
                aStockItem.remove(amount, sendDate);
                aDeliveryItem = new DeliveryItem(this);
        }
}
```

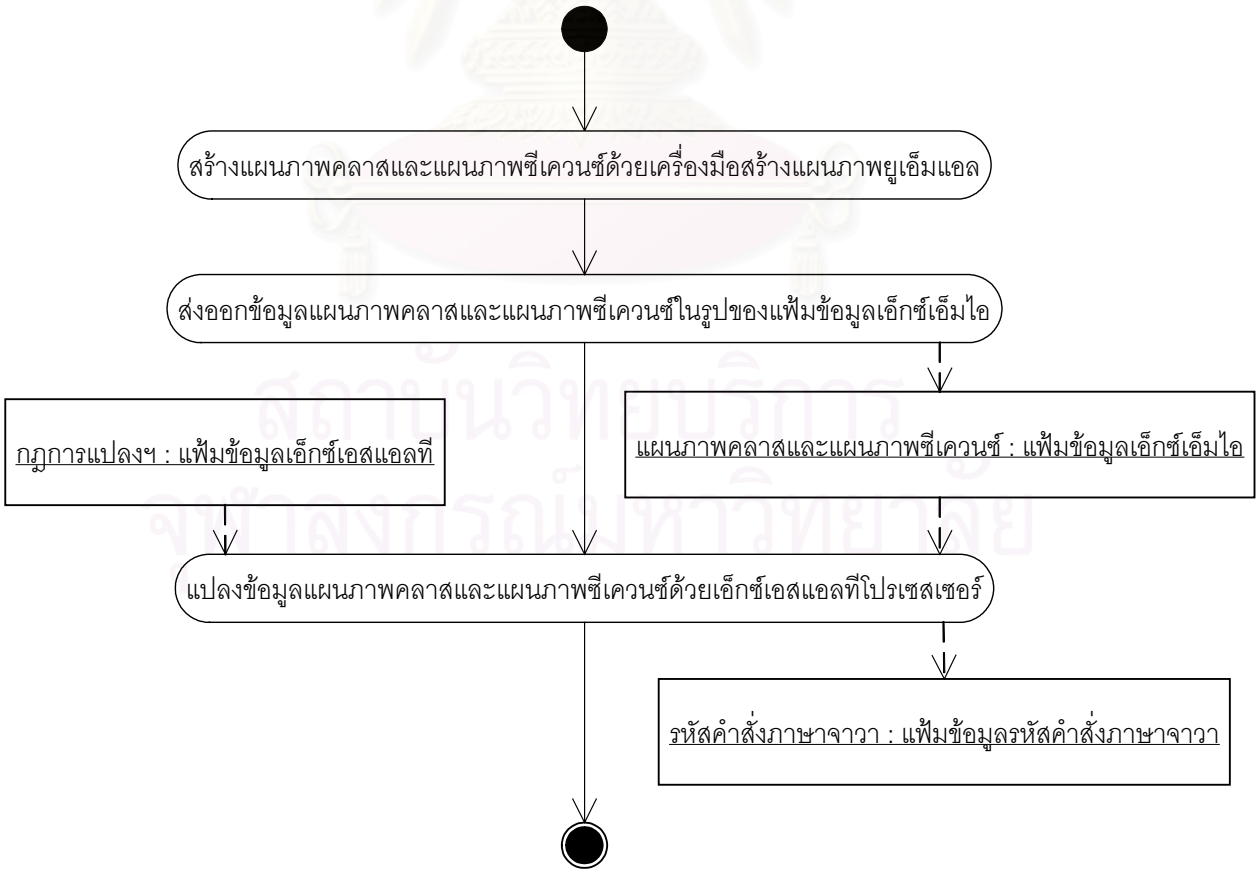
รูปที่ 3.13 ตัวอย่างของรหัสคำสั่งของโอเปอร์เรชัน “prepare” ของคลาส “OrderLine” ที่ได้จากการทำการแปลงตัวอย่างส่วนเตรียมการจัดส่งสินค้าของระบบสั่งซื้อสินค้า

บทที่ 4
การออกแบบเครื่องมือสนับสนุนการทำการแปลง ฯ

ในบทนี้จะเป็นการกล่าวถึงการออกแบบเครื่องมือสนับสนุนการทำการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา โดยเครื่องมือที่ออกแบบขึ้นมานี้จะต้องสามารถทำการแปลง ฯ ได้ตามขั้นตอนและกฎการแปลง ฯ ที่ได้ออกแบบไว้ในบทที่ 3

4.1 การออกแบบโครงสร้างพื้นฐานของเครื่องมือและเทคโนโลยีที่ใช้ในการทำการแปลง ฯ

รูปที่ 4.1 เป็นภาพรวมของกระบวนการทำการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา ซึ่งได้ออกแบบให้ทำงานร่วมกับเครื่องมือสร้างแผนภาพยูเอ็มแอลที่มีอยู่ในปัจจุบัน โดยได้มีการเลือกใช้เทคโนโลยีในการแลกเปลี่ยนข้อมูลกับเครื่องมือสร้างแผนภาพยูเอ็มแอลและเทคโนโลยีที่นำมาใช้อิมพลีเมนต์การแปลง ฯ ที่มีความเหมาะสมกับงานวิจัยนี้ ดังในรายละเอียดที่จะได้กล่าวถึงต่อไปนี้



รูปที่ 4.1 ภาพรวมของกระบวนการทำการแปลง ฯ

4.1.1 การแลกเปลี่ยนข้อมูลกับเครื่องมือสร้างแผนภาพยูเอ็มแอล

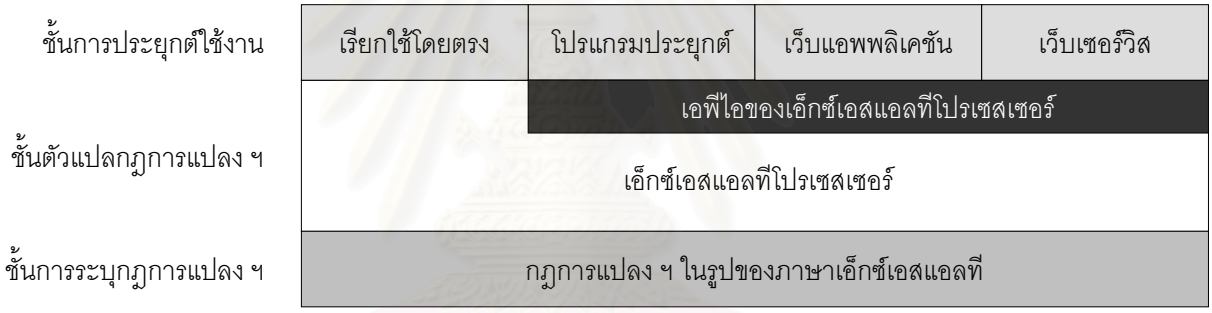
จากการสำรวจเครื่องมือสร้างแผนภาพยูเอ็มแอลต่าง ๆ พบว่าส่วนใหญ่สามารถส่งออกหรือมีการจัดเก็บแบบจำลองยูเอ็มแอลในรูปของแฟ้มข้อมูลชนิดเอ็กซ์เอ็มแอลตามมาตรฐานเอ็กซ์เอ็มไอซึ่งเป็นมาตรฐานที่สามารถทำให้การแลกเปลี่ยนข้อมูลแบบจำลองยูเอ็มแอลระหว่างเครื่องมือต่าง ๆ เป็นไปได้โดยสะดวก ดังนั้นผู้วิจัยจึงได้เลือกใช้แฟ้มข้อมูลชนิดนี้เป็นข้อมูลเข้าของเครื่องมือที่จะอิมพลีเมนต์ขึ้นมา อย่างไรก็ตามโครงสร้างแฟ้มข้อมูลดังกล่าวจะมีรายละเอียดที่แตกต่างกันไปบ้างตามแต่ผู้ผลิต โดยในเบื้องต้น ได้เลือกที่จะให้เครื่องมือสามารถทำงานกับแฟ้มข้อมูลที่ส่งออกจาก เรชันแนล โรส รุ่น 2002 (Rational Rose 2002) [11] ด้วยเครื่องมือเสริม ยูนิซิส โรส เอ็กซ์เอ็มแอล ทูล รุ่น 1.3.6 (Unisys Rose XML Tools 1.3.6) [12] เนื่องจากเป็นเครื่องมือสร้างแผนภาพยูเอ็มแอลที่นิยมใช้กันอย่างแพร่หลาย

4.1.2 ภาษาที่ใช้ในการอิมพลีเมนต์การแปลง ฯ

ในส่วนของการอิมพลีเมนต์การแปลง ฯ ผู้วิจัยเลือกใช้ภาษาเอ็กซ์เอสแอลที เนื่องจากเป็นภาษาที่ออกแบบมาเพื่อให้มีความเหมาะสมกับการแปลงเอกสารเอ็กซ์เอ็มแอลโดยเฉพาะ โดยเอ็กซ์เอสแอลทีโปรเซสเซอร์ที่ใช้เป็นตัวแปลภาษานี้มีให้เลือกใช้อยู่หลายตัวบนหลายแพลตฟอร์ม (Platform) ซึ่งจะทำให้เครื่องมือที่จะพัฒนาขึ้นมามีความสามารถในการปรับเปลี่ยนให้ทำงานบนแพลตฟอร์มต่าง ๆ (Portability) สูงตามไปด้วย โดยผู้วิจัยได้ทำการทดสอบเอ็กซ์เอสแอลทีโปรเซสเซอร์ที่ใช้กันอย่างแพร่หลายในปัจจุบันที่ทำตามเอกสารแนะนำเอ็กซ์เอสแอลที รุ่น 1.0 (XSLT 1.0 recommendation) [10] และมีส่วนขยายสำหรับทำการแปลงส่วนของต้นไม้ผลลัพธ์ (Result tree fragment) ไปเป็นเซตของโหนด (Node-set) ซึ่งจากผลการทดสอบ ผู้วิจัยได้เลือกใช้แซกเซิน (Saxon) รุ่น 7.7 [13] เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์สำหรับเครื่องมือสำหรับแพลตฟอร์มจาวา เนื่องจากเป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์ที่มีความเร็วในการทำงานสูงกว่าเอ็กซ์เอสแอลทีโปรเซสเซอร์ตัวอื่นที่ทำงานบนแพลตฟอร์มจาวาเช่นเดียวกันอย่างเด่นชัด และเลือกใช้ ไมโครซอฟท์ เอ็กซ์เอ็มแอล คอร์ เซอร์วิส รุ่น 4.0 (Microsoft XML Core Services 4.0) [18] เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์สำหรับเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต

4.1.3 สถาปัตยกรรมของการทำการแปลง ฯ

รูปที่ 4.2 แสดงให้เห็นถึงสถาปัตยกรรมของการทำการแปลง ฯ ซึ่งมีลักษณะเป็นชั้น ๆ โดยที่แต่ละชั้นจะมีหน้าที่เฉพาะอย่าง ซึ่งทำให้เกิดความสะดวกในการพัฒนาและการดูแลรักษาเป็นอย่างมาก โดยชั้นล่างสุดจะเป็นชั้นการระบุกฎการแปลง ฯ ซึ่งจะได้นำเอาขั้นตอนและกฎการแปลง ฯ ที่ได้ออกแบบขึ้นมาในงานวิจัยนี้มาระบุด้วยภาษาเอ็กซ์เอสแอลที ชั้นกลางจะเป็นชั้นตัวแปลกฎการแปลง ซึ่งจะได้ใช้เอ็กซ์เอสแอลทีโปรเซสเซอร์เป็นตัวแปลกฎการแปลง ฯ ที่ได้ระบุไว้ ส่วนชั้นบนสุดจะเป็นชั้นการประยุกต์ใช้งาน ซึ่งผู้ใช้อาจสั่งให้เอ็กซ์เอสแอลทีโปรเซสเซอร์ทำการแปลง ฯ โดยตรง หรืออาจให้โปรแกรมประยุกต์ในรูปแบบต่าง ๆ สั่งให้เอ็กซ์เอสแอลทีโปรเซสเซอร์ทำการแปลง ฯ ผ่านทางเอพีไอของเอ็กซ์เอสแอลทีโปรเซสเซอร์ก็ได้

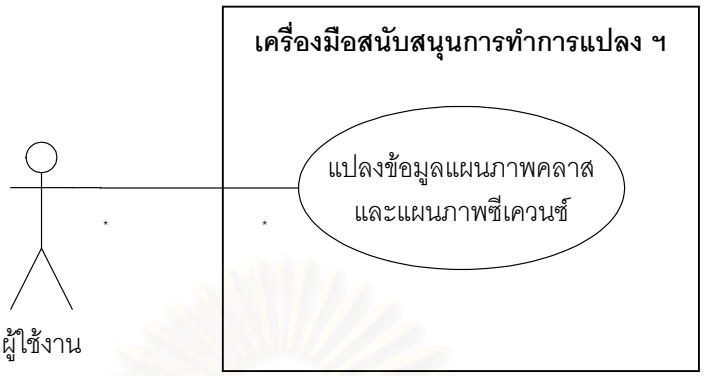


รูปที่ 4.2 สถาปัตยกรรมของการทำการแปลง ฯ

4.2 การออกแบบเครื่องมือ

4.2.1 การออกแบบการใช้งานของเครื่องมือ

รูปที่ 4.3 เป็นการแสดงแผนภาพยูสเคสของเครื่องมือสนับสนุนการทำการแปลง ฯ ที่จะทำการพัฒนาขึ้นมา โดยผู้ใช้งานสามารถให้เครื่องมือแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ที่อยู่ในรูปของแฟ้มข้อมูลเอ็กซ์เอ็มไอ โดยรายละเอียดของกิจกรรมต่าง ๆ ที่เครื่องมือดังกล่าวต้องทำเพื่อให้ได้ผลการแปลง ฯ ที่ผู้ใช้ต้องการ จะได้กล่าวถึงในหัวข้อที่ 4.2.2



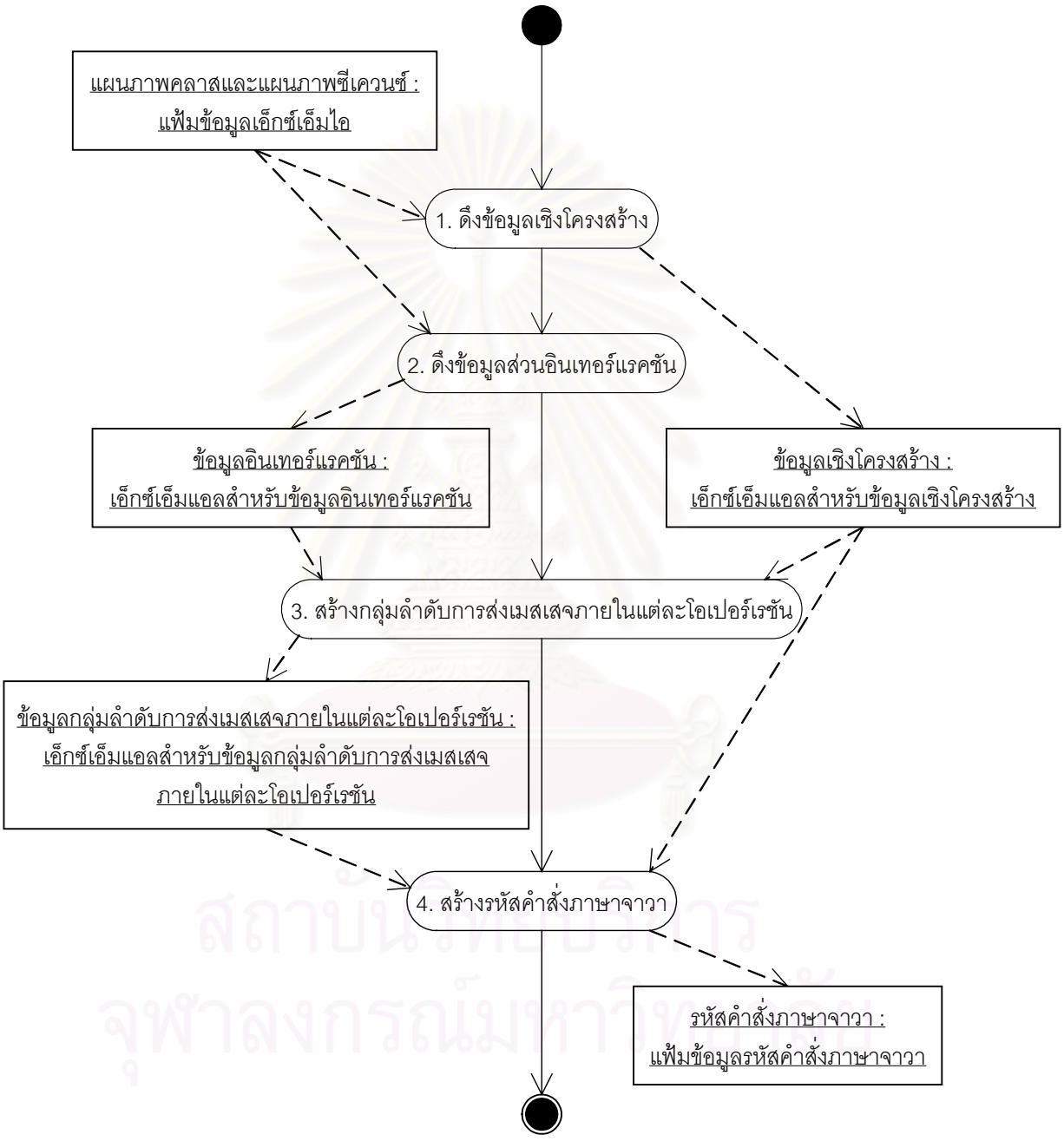
รูปที่ 4.3 แผนภาพยูสเคสของเครื่องมือสนับสนุนการทำการแปลง ฯ

4.2.2 การออกแบบกิจกรรมต่าง ๆ ในการทำการแปลง ฯ ของเครื่องมือ

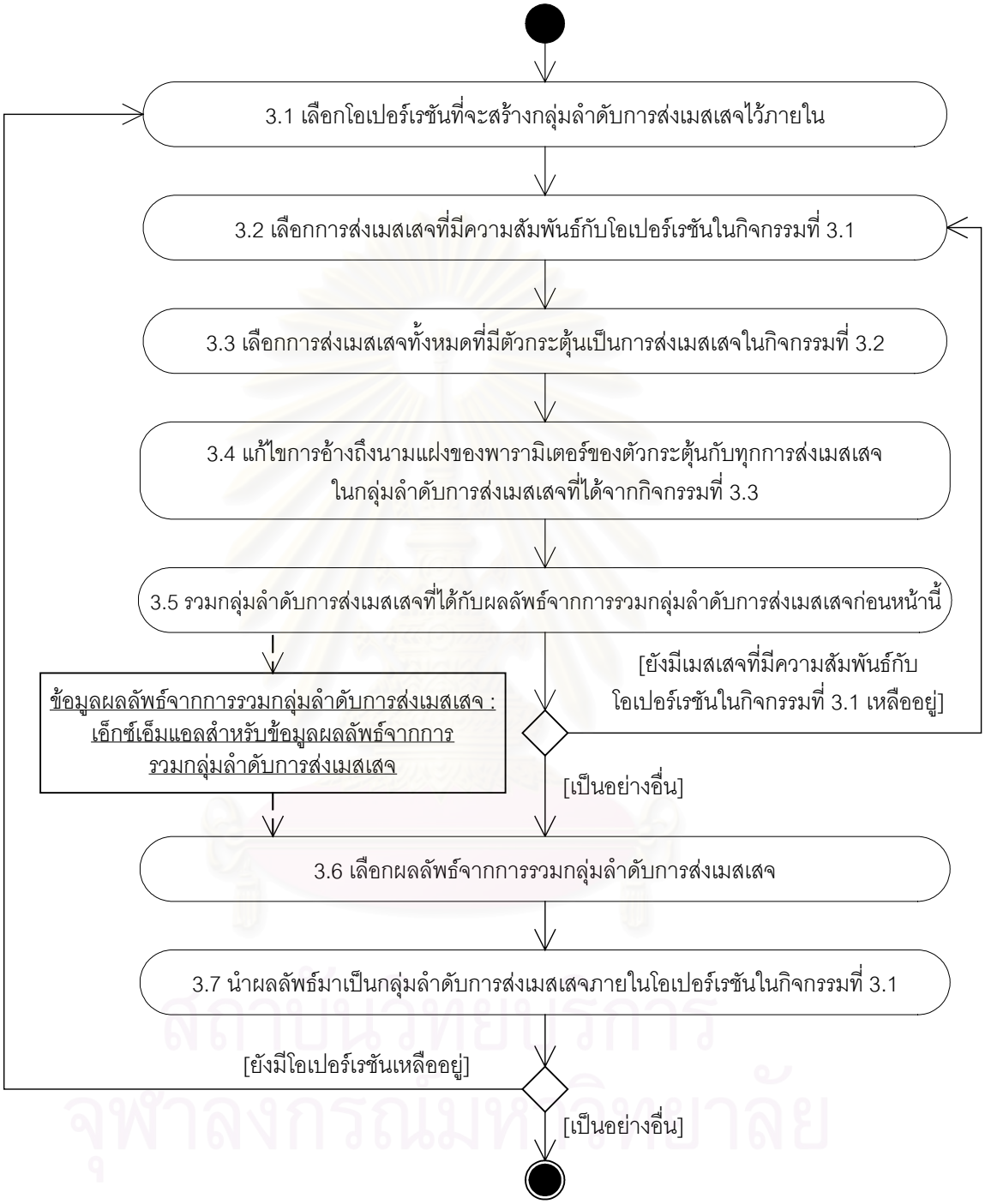
รูปที่ 4.4 เป็นการออกแบบกิจกรรมต่าง ๆ ในการทำการแปลง ฯ ของเครื่องมือ โดยสองกิจกรรมแรกในกระบวนการแปลง ฯ จะเป็นการสกัดเอาเฉพาะข้อมูลที่จะต้องใช้ในการทำการแปลง ฯ จากแฟ้มข้อมูลเอ็กซ์เอ็มไอที่ส่งออกมาจากเครื่องมือสร้างแผนภาพยูเอ็มแอล มาทำการจัดเตรียมให้อยู่ในรูปของข้อมูลเอ็กซ์เอ็มแอลที่เหมาะสมกับกิจกรรมในการทำการแปลง ฯ ในขั้นถัดไป โดยที่ข้อมูลเชิงโครงสร้างจะได้มาจากแผนภาพคลาส ส่วนข้อมูลอินเทอร์แรคชันจะได้มาจากแผนภาพซีเควนซ์

กิจกรรมที่ 3 เป็นการสร้างกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน โดยอาศัยข้อมูลเชิงโครงสร้างที่ได้จากกิจกรรมที่ 1 และข้อมูลอินเทอร์แรคชันในกิจกรรมที่ 2 โดยที่กิจกรรมที่ 3 นี้จะประกอบไปด้วยกิจกรรมย่อยต่าง ๆ แสดงดังรูปที่ 4.5 ซึ่งเป็นการประยุกต์ใช้กฎการแปลง ฯ ที่ได้ออกแบบไว้ในหัวข้อที่ 3.1 ถึง 3.4

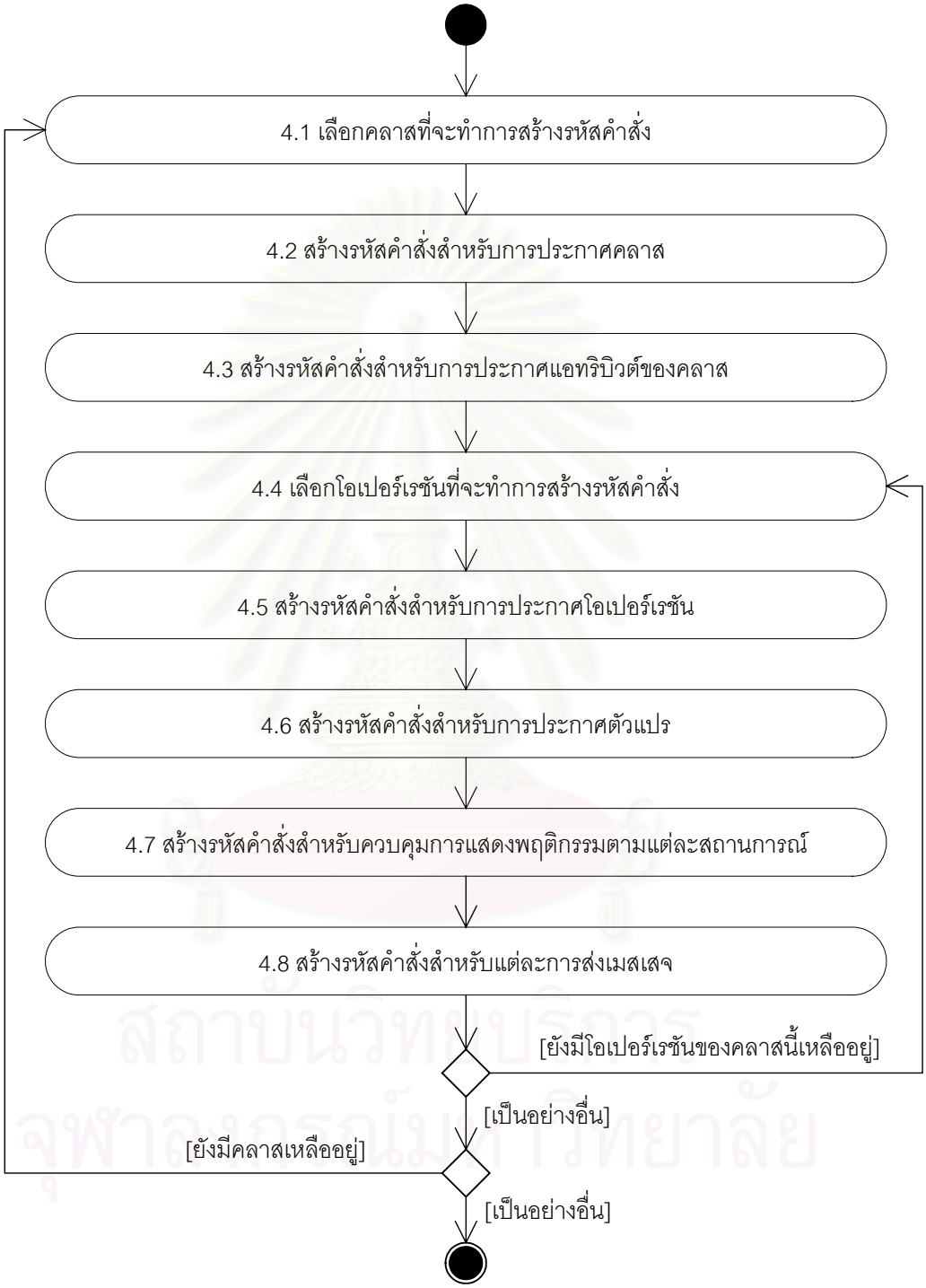
กิจกรรมที่ 4 เป็นการสร้างรหัสคำสั่งภาษาจาวา โดยอาศัยข้อมูลเชิงโครงสร้างจากกิจกรรมที่ 1 และข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชันจากกิจกรรมที่ 3 โดยที่กิจกรรมที่ 4 นี้จะประกอบด้วยกิจกรรมย่อยต่าง ๆ แสดงดังรูปที่ 4.6 ซึ่งเป็นการประยุกต์ใช้กฎการแปลง ฯ ในหัวข้อที่ 3.5 ถึง 3.7



รูปที่ 4.4 กิจกรรมต่าง ๆ ในการทำการแปลง ฯ ของเครื่องมือ



รูปที่ 4.5 กิจกรรมย่อยของกิจกรรม
“สร้างกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน” ในรูปที่ 4.4



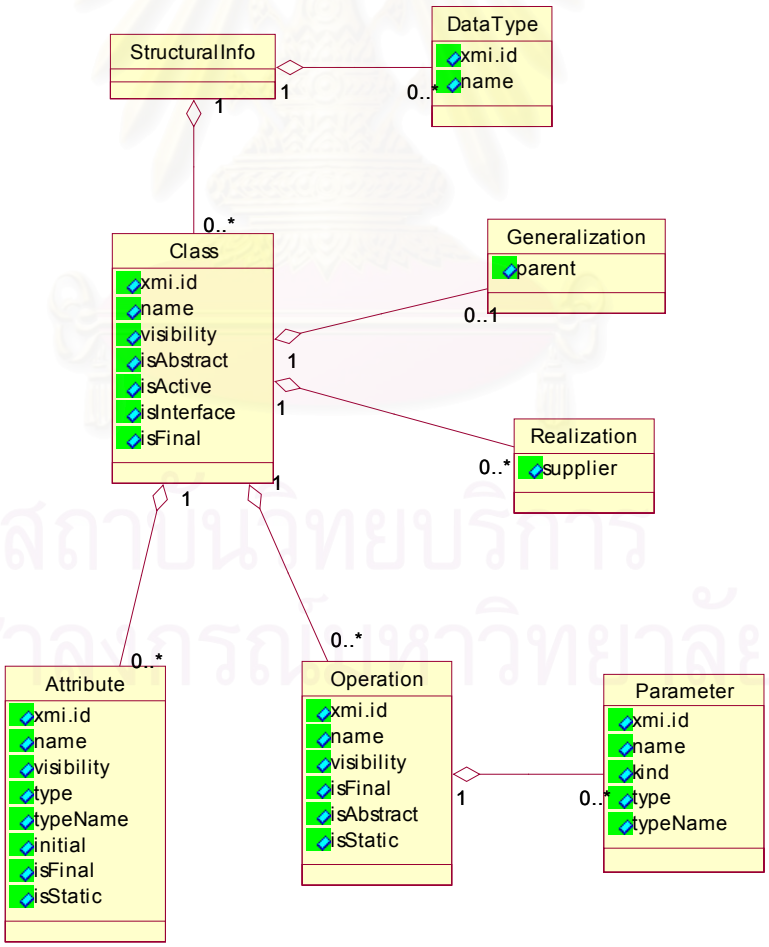
รูปที่ 4.6 กิจกรรมย่อยของกิจกรรม “สร้างรหัสคำสั่งภาษาจาวา” ในรูปที่ 4.4

4.2.3 การออกแบบโครงสร้างข้อมูลเอ็กซ์เอ็มแอลที่ใช้ในเครื่องมือ

จากรูปที่ 4.4 จะสังเกตเห็นว่ามีผลลัพธ์ในระหว่างการทำการแปลง ฯ เกิดขึ้น ซึ่งได้แก่ ข้อมูลเชิงโครงสร้าง ข้อมูลอินเทอร์แรคชัน และข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน และในรูปที่ 4.5 ได้มีข้อมูลผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจเกิดขึ้น ในหัวข้อนี้จะได้กล่าวถึงการออกแบบโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลดังกล่าว

4.2.3.1 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลเชิงโครงสร้าง

ข้อมูลเชิงโครงสร้าง เป็นข้อมูลที่เกี่ยวกับโครงสร้างของคลาส โดยจะถูกใช้เป็นหลักในการสร้างรหัสคำสั่งเพื่อใช้ในการประกาศคลาส แอทริบิวต์ของคลาส และโอเปอร์เรชันของคลาส ซึ่งข้อมูลเหล่านี้จะได้มาจากแผนภาพคลาส รูปที่ 4.7 เป็นการแสดงโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลเชิงโครงสร้างที่ใช้ในเครื่องมือนี้



รูปที่ 4.7 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลเชิงโครงสร้าง

ตารางที่ 4.1 ถึง 4.8 เป็นการแสดงรายละเอียดของแต่ละส่วนย่อยในโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลเชิงโครงสร้าง

ตารางที่ 4.1 รายละเอียดของส่วนย่อย “StructuralInfo”

ชื่อส่วนย่อย	StructuralInfo
คำอธิบาย	ทำหน้าที่เป็นราก (Root) ของข้อมูลเชิงโครงสร้าง

ตารางที่ 4.2 รายละเอียดของส่วนย่อย “DataType”

ชื่อส่วนย่อย	DataType
คำอธิบาย	เก็บข้อมูลชนิดของข้อมูลประเภทปฐมฐาน (Primitive)
แอทริบิวต์	
- xmi.id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- name	ชื่อของชนิดข้อมูล

ตารางที่ 4.3 รายละเอียดของส่วนย่อย “Class”

ชื่อส่วนย่อย	Class
คำอธิบาย	เก็บข้อมูลของคลาส
แอทริบิวต์	
- xmi.id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- name	ชื่อของคลาส
- visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ private protected และ public
- isAbstract	บอกว่าเป็นคลาสประเภทแอบสแตรก (Abstract) หรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false
- isInterface	บอกว่าเป็นอินเทอร์เฟสหรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false
- isFinal	บอกว่าเป็นคลาสประเภทไฟนอล (Final) หรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false

ตารางที่ 4.4 รายละเอียดของส่วนย่อย “Generalization”

ชื่อส่วนย่อย	Generalization
คำอธิบาย	เก็บข้อมูลการรับทอด (Inheritance)
แอทริบิวต์	
- parent	รหัสระบุตัวตนของคลาสแม่

ตารางที่ 4.5 รายละเอียดของส่วนย่อย “Realization”

ชื่อส่วนย่อย	Realization
คำอธิบาย	เก็บข้อมูลการอิมพลีเมนต์อินเทอร์เฟส
แอทริบิวต์	
- supplier	รหัสเพื่อใช้ในการระบุตัวตนของอินเทอร์เฟส

ตารางที่ 4.6 รายละเอียดของส่วนย่อย “Attribute”

ชื่อส่วนย่อย	Attribute
คำอธิบาย	เก็บข้อมูลของแอทริบิวต์
แอทริบิวต์	
- xmi.id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- name	ชื่อของแอทริบิวต์
- type	รหัสระบุตัวตนของชนิดข้อมูลของแอทริบิวต์นี้
- typeName	ชื่อของชนิดข้อมูลของแอทริบิวต์นี้
- initial	ค่าเริ่มต้น
- isFinal	บอกว่าเป็นแอทริบิวต์ประเภทไฟนอลหรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false
- isStatic	บอกว่าเป็นแอทริบิวต์ประเภทสแตติกหรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false

ตารางที่ 4.7 รายละเอียดของส่วนย่อย “Operation”

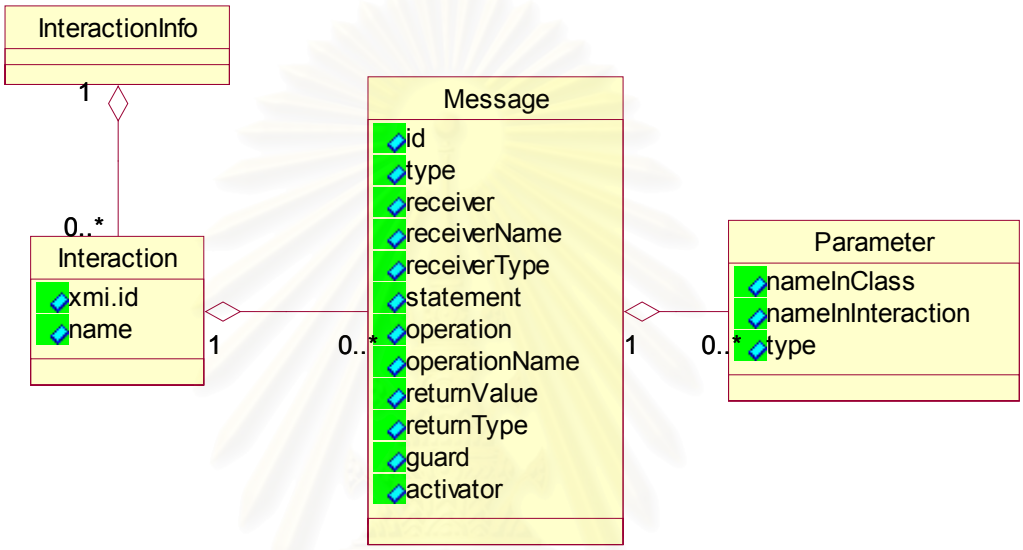
ชื่อส่วนย่อย	Operation
คำอธิบาย	เก็บข้อมูลของโอเปอร์เรชัน
แอทริบิวต์	
- xmi.id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- name	ชื่อของโอเปอร์เรชัน
- visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ private protected และ public
- isFinal	บอกว่าเป็นโอเปอร์เรชันประเภทไฟนอลหรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false
- isAbstract	บอกว่าเป็นโอเปอร์เรชันประเภทแอบสแตรกหรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false
- isStatic	บอกว่าเป็นโอเปอร์เรชันประเภทสแตติกหรือไม่ โดยมีค่าได้ 2 แบบ ได้แก่ true และ false

ตารางที่ 4.8 รายละเอียดของส่วนย่อย “Parameter”

ชื่อส่วนย่อย	Parameter
คำอธิบาย	เก็บข้อมูลของพารามิเตอร์
แอทริบิวต์	
- xmi.id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- name	ชื่อของพารามิเตอร์
- kind	ชนิดของพารามิเตอร์ โดยมีค่าได้ 2 แบบ ได้แก่ inout ในกรณีเป็นพารามิเตอร์ประเภทอินพุต/เอาต์พุต และ return ในกรณีเป็นพารามิเตอร์ประเภทส่งกลับ
- type	รหัสระบุตัวตนของชนิดข้อมูลของพารามิเตอร์นี้
- typeName	ชื่อของชนิดข้อมูลของพารามิเตอร์นี้

4.2.3.2 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลอินเทอร์แรคชัน

ข้อมูลอินเทอร์แรคชัน เป็นข้อมูลที่เกี่ยวกับลำดับการส่งเมสเสจระหว่างวัตถุต่าง ๆ ซึ่งข้อมูลเหล่านี้จะได้มาจากแผนภาพซีเควนซ์ รูปที่ 4.8 เป็นการแสดงโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลอินเทอร์แรคชันที่ใช้ในเครื่องมือนี้



รูปที่ 4.8 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลอินเทอร์แรคชัน

ตารางที่ 4.9 ถึง 4.12 เป็นการแสดงรายละเอียดของแต่ละส่วนย่อยในโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลอินเทอร์แรคชัน

ตารางที่ 4.9 รายละเอียดของส่วนย่อย “InteractionInfo”

ชื่อส่วนย่อย	InteractionInfo
คำอธิบาย	ทำหน้าที่เป็นรากของข้อมูลอินเทอร์แรคชัน

ตารางที่ 4.10 รายละเอียดของส่วนย่อย “Interaction”

ชื่อส่วนย่อย	Interaction
คำอธิบาย	เก็บข้อมูลแผนภาพอินเทอร์แรคชัน
แอทริบิวต์	
- xmi.id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- name	ชื่อของแผนภาพ

ตารางที่ 4.11 รายละเอียดของส่วนย่อย “Message”

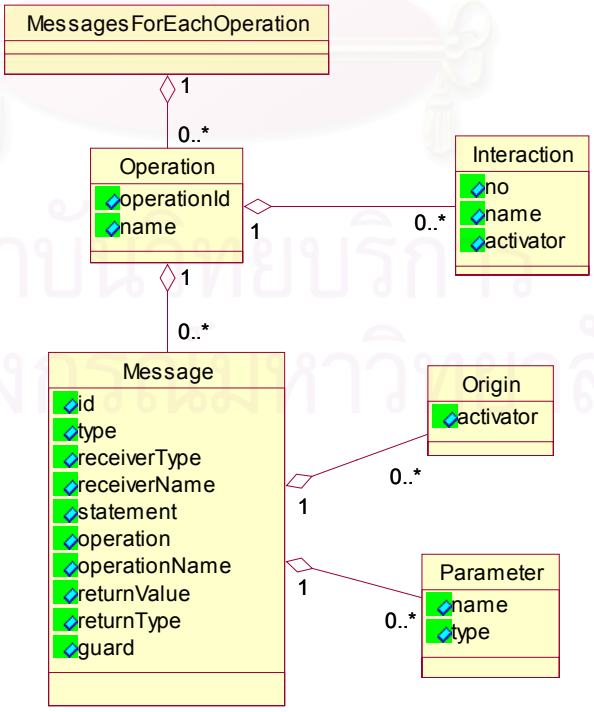
ชื่อส่วนย่อย	Message
คำอธิบาย	เก็บข้อมูลการส่งเมสเสจ
แอทริบิวต์	
- id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- type	ชนิดของการส่งเมสเสจ โดยมีค่าได้ 4 แบบ ได้แก่ - callAction สำหรับการส่งเมสสเสจระหว่างวัตถุ - localInvocation สำหรับการส่งเมสเสจภายในวัตถุ - createAction สำหรับการส่งเมสเสจสร้างวัตถุ - returnAction สำหรับการส่งเมสเสจส่งกลับ
- receiver	รหัสระบุตัวตนของวัตถุที่เป็นตัวรับเมสเสจ
- receiverName	ชื่อของวัตถุที่เป็นตัวรับเมสเสจ
- receiverType	รหัสระบุตัวตนของคลาสของวัตถุที่เป็นตัวรับเมสเสจ
- statement	ข้อความในเมสเสจ
- operation	รหัสระบุตัวตนของโอเปอร์เรชันที่สัมพันธ์กับเมสเสจนี้
- operationName	ชื่อของโอเปอร์เรชันที่สัมพันธ์กับเมสเสจนี้
- returnValue	ชื่อของตัวแปรรับค่าส่งกลับ
- returnType	ชื่อของชนิดข้อมูลของตัวแปรรับค่าส่งกลับ
- guard	เงื่อนไขการส่งเมสเสจ
- activator	รหัสระบุตัวตนของตัวกระตุ้นของการส่งเมสเสจ

ตารางที่ 4.12 รายละเอียดของส่วนย่อย “Parameter”

ชื่อส่วนย่อย	Parameter
คำอธิบาย	เก็บข้อมูลของพารามิเตอร์
แอทริบิวต์	
- nameInClass	ชื่อของพารามิเตอร์ตามแผนภาพคลาส
- nameInInteraction	ชื่อของพารามิเตอร์ตามแผนภาพอินเทอร์แรคชัน
- type	ชื่อของชนิดข้อมูลของพารามิเตอร์

4.2.3.3 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน

ข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน เป็นข้อมูลที่ได้มาจากกิจกรรม “สร้างกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน” ในรูปที่ 4.4 ซึ่งข้อมูลดังกล่าวจะถูกนำไปสร้างเป็นรหัสคำสั่งภายในแต่ละโอเปอร์เรชัน รูปที่ 4.9 เป็นการแสดงโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชันที่ใช้ในเครื่องมือนี้



รูปที่ 4.9 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน

ตารางที่ 4.13 ถึง 4.18 เป็นการแสดงรายละเอียดของแต่ละส่วนย่อยในโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลอินเทอร์แรคชัน

ตารางที่ 4.13 รายละเอียดของส่วนย่อย “MessagesForEachOperation”

ชื่อส่วนย่อย	MessagesForEachOperation
คำอธิบาย	ทำหน้าที่เป็นรากของข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชัน

ตารางที่ 4.14 รายละเอียดของส่วนย่อย “Operation”

ชื่อส่วนย่อย	Operation
คำอธิบาย	เก็บข้อมูลของโอเปอร์เรชัน
แอทริบิวต์	
- operationId	รหัสระบุตัวตนของส่วนย่อย “Operation” ในข้อมูลเชิงโครงสร้าง
- name	ชื่อของโอเปอร์เรชัน

ตารางที่ 4.15 รายละเอียดของส่วนย่อย “Interaction”

ชื่อส่วนย่อย	Interaction
คำอธิบาย	เก็บข้อมูลแผนภาพอินเทอร์แรคชันที่เป็นพฤติกรรมของโอเปอร์เรชันนี้
แอทริบิวต์	
- no	หมายเลขลำดับของแผนภาพอินเทอร์แรคชันที่ใช้ในโอเปอร์เรชันนี้
- name	ชื่อของแผนภาพอินเทอร์แรคชัน
- activator	รหัสระบุตัวตนของส่วนย่อย “Message” ที่ทำหน้าที่เป็นตัวกระตุ้นให้เกิดพฤติกรรมของโอเปอร์เรชันนี้

ตารางที่ 4.16 รายละเอียดของส่วนย่อย “Message”

ชื่อส่วนย่อย	Message
คำอธิบาย	เก็บข้อมูลการส่งเมสเสจ
แอทริบิวต์	
- id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- type	ชนิดของการส่งเมสเสจ โดยมีค่าได้ 4 แบบ ได้แก่ - callAction สำหรับการส่งเมสเสจระหว่างวัตถุ - localInvocation สำหรับการส่งเมสเสจภายในวัตถุ - createAction สำหรับการส่งเมสเสจสร้างวัตถุ - returnAction สำหรับการส่งเมสเสจส่งกลับ
- receiverType	รหัสระบุตัวตนของคลาสของวัตถุที่เป็นตัวรับเมสเสจ
- receiverName	ชื่อของวัตถุที่เป็นตัวรับเมสเสจ
- statement	ข้อความในเมสเสจ
- operation	รหัสระบุตัวตนของโอเปอร์เรชันที่สัมพันธ์กับเมสเสจนี้
- operationName	ชื่อของโอเปอร์เรชันที่สัมพันธ์กับเมสเสจนี้
- returnValue	ชื่อของตัวแปรรับค่าส่งกลับ
- returnType	ชื่อของชนิดข้อมูลของตัวแปรรับค่าส่งกลับ
- guard	เงื่อนไขการส่งเมสเสจ

ตารางที่ 4.17 รายละเอียดของส่วนย่อย “Origin”

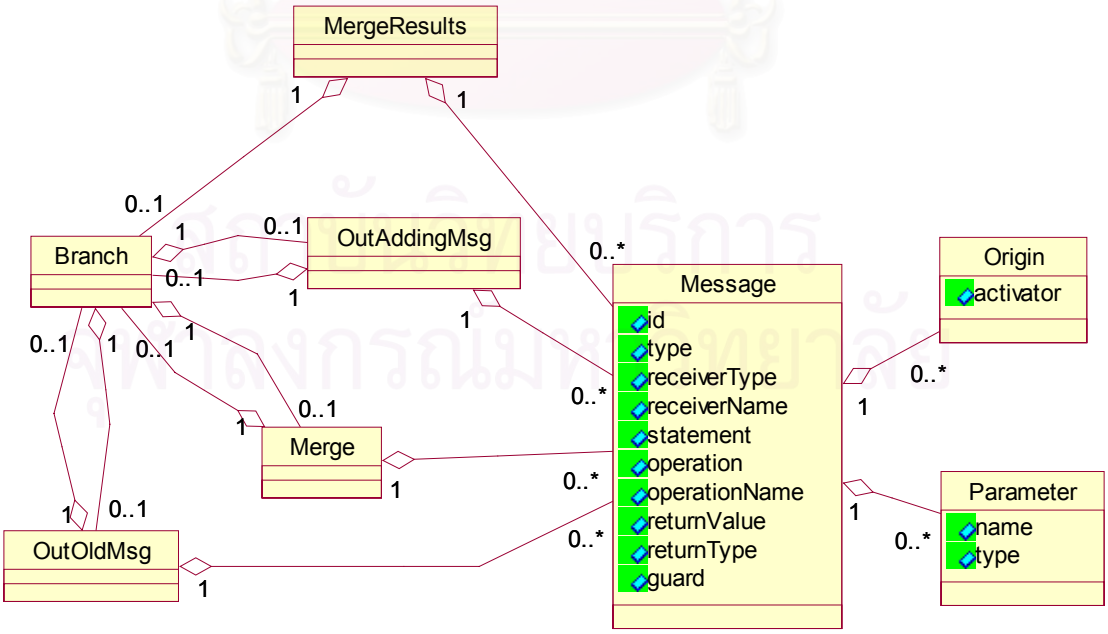
ชื่อส่วนย่อย	Origin
คำอธิบาย	เก็บข้อมูลที่มาของการส่งเมสเสจ
แอทริบิวต์	
- activator	รหัสระบุตัวตนของตัวกระตุ้นของการส่งเมสเสจ

ตารางที่ 4.18 รายละเอียดของส่วนย่อย “Parameter”

ชื่อส่วนย่อย	Parameter
คำอธิบาย	เก็บข้อมูลของพารามิเตอร์
แอทริบิวต์	
- name	ชื่อของพารามิเตอร์
- type	ชื่อของชนิดข้อมูลของพารามิเตอร์

4.2.3.4 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ

ข้อมูลผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ เป็นข้อมูลที่ได้มาจากกิจกรรมที่ 3.5 “รวมกลุ่มลำดับการส่งเมสเสจที่ได้กับผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจก่อนหน้านี้” ในรูปที่ 4.5 ซึ่งข้อมูลผลลัพธ์ดังกล่าวจะถูกนำไปเป็นข้อมูลเข้าในกิจกรรมที่ 3.6 “เลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ” รูปที่ 4.10 เป็นการแสดงโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชันที่ใช้ในเครื่องมือนี้



รูปที่ 4.10 โครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ

ตารางที่ 4.19 ถึง 4.26 เป็นการแสดงรายละเอียดของแต่ละส่วนย่อยในโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ

ตารางที่ 4.19 รายละเอียดของส่วนย่อย “MergeResults”

ชื่อส่วนย่อย	MergeResults
คำอธิบาย	ทำหน้าที่เป็นรากของข้อมูลผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ

ตารางที่ 4.20 รายละเอียดของส่วนย่อย “Branch”

ชื่อส่วนย่อย	Branch
คำอธิบาย	เป็นจุดที่แสดงทางเลือกในการรวมการส่งเมสเสจคู่ใด ๆ

ตารางที่ 4.21 รายละเอียดของส่วนย่อย “Merge”

ชื่อส่วนย่อย	Merge
คำอธิบาย	เก็บผลลัพธ์ที่เกิดจากการเลือกที่จะรวมการส่งเมสเสจคู่ใด ๆ

ตารางที่ 4.22 รายละเอียดของส่วนย่อย “OutOldMsg”

ชื่อส่วนย่อย	OutOldMsg
คำอธิบาย	เก็บผลลัพธ์ที่เกิดจากการเลือกที่จะไม่รวมการส่งเมสเสจคู่ใด ๆ และทำการเลื่อนการส่งเมสเสจตัวต่อไปของกลุ่มลำดับการส่งเมสเสจที่เป็นตัวตั้งต้นขึ้นมาทำการพิจารณา

ตารางที่ 4.23 รายละเอียดของส่วนย่อย “OutAddingMsg”

ชื่อส่วนย่อย	OutAddingMsg
คำอธิบาย	เก็บผลลัพธ์ที่เกิดจากการเลือกที่จะไม่รวมการส่งเมสเสจคู่ใด ๆ และทำการเลื่อนการส่งเมสเสจตัวต่อไปของกลุ่มลำดับการส่งเมสเสจที่เป็นตัวที่เข้าไปทำการรวมขึ้นมาทำการพิจารณา

ตารางที่ 4.24 รายละเอียดของส่วนย่อย “Message”

ชื่อส่วนย่อย	Message
คำอธิบาย	เก็บข้อมูลการส่งเมสเสจ
แอทริบิวต์	
- id	รหัสเพื่อใช้ในการระบุตัวตนของส่วนย่อย
- type	ชนิดของการส่งเมสเสจ โดยมีค่าได้ 4 แบบ ได้แก่ - callAction สำหรับการส่งเมสเสจระหว่างวัตถุ - localInvocation สำหรับการส่งเมสเสจภายในวัตถุ - createAction สำหรับการส่งเมสเสจสร้างวัตถุ - returnAction สำหรับการส่งเมสเสจส่งกลับ
- receiverType	รหัสระบุตัวตนของคลาสของวัตถุที่เป็นตัวรับเมสเสจ
- receiverName	ชื่อของวัตถุที่เป็นตัวรับเมสเสจ
- statement	ข้อความในเมสเสจ
- operation	รหัสระบุตัวตนของโอเปอร์เรชันที่สัมพันธ์กับเมสเสจนี้
- operationName	ชื่อของโอเปอร์เรชันที่สัมพันธ์กับเมสเสจนี้
- returnValue	ชื่อของตัวแปรรับค่าส่งกลับ
- returnType	ชื่อของชนิดข้อมูลของตัวแปรรับค่าส่งกลับ
- guard	เงื่อนไขการส่งเมสเสจ

ตารางที่ 4.25 รายละเอียดของส่วนย่อย “Origin”

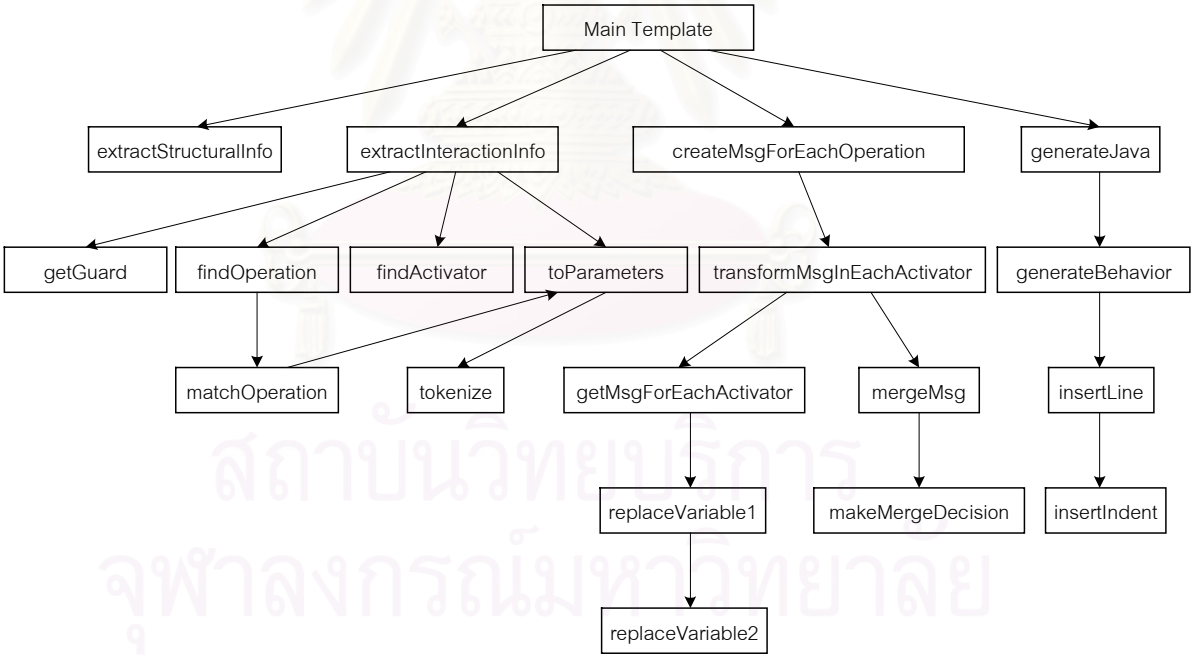
ชื่อส่วนย่อย	Origin
คำอธิบาย	เก็บข้อมูลที่มาของการส่งเมสเสจ
แอทริบิวต์	
- activator	รหัสระบุตัวตนของตัวกระตุ้นของการส่งเมสเสจ

ตารางที่ 4.26 รายละเอียดของส่วนย่อย “Parameter”

ชื่อส่วนย่อย	Parameter
คำอธิบาย	เก็บข้อมูลของพารามิเตอร์
แอทริบิวต์	
- name	ชื่อของพารามิเตอร์
- type	ชื่อของชนิดข้อมูลของพารามิเตอร์

4.2.4 การออกแบบโครงสร้างการเรียกเทมเพลตของเครื่องมือ

รูปที่ 4.11 เป็นการแสดงโครงสร้างการเรียกเทมเพลตของเครื่องมือ โดยหน้าที่ของแต่ละเทมเพลตแสดงอยู่ในตารางที่ 4.27



รูปที่ 4.11 โครงสร้างการเรียกเทมเพลตของเครื่องมือ

ตารางที่ 4.27 หน้าที่ของแต่ละเทมเพลต

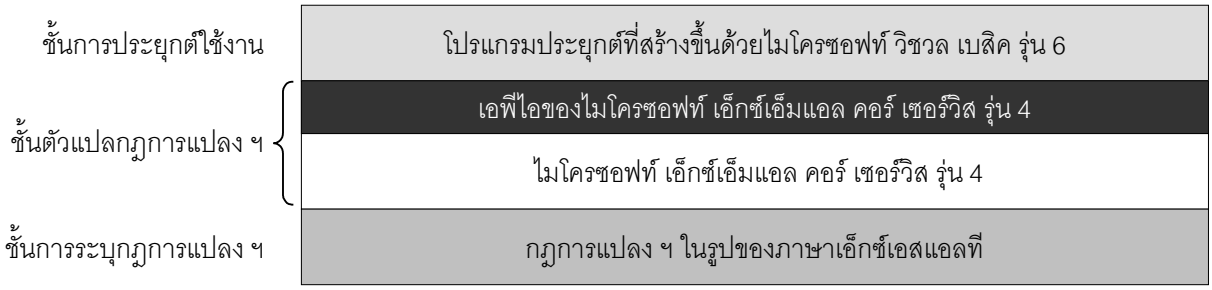
เทมเพลต	หน้าที่
Main Template	ทำหน้าที่สั่งการให้เกิดกระบวนการแปลง ฯ ทั้งหมด
extractStructuralInfo	ทำหน้าที่ดึงข้อมูลเชิงโครงสร้างจากแผนภาพคลาสมาสร้างเป็นข้อมูลตามโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลเชิงโครงสร้างในรูปที่ 4.7
extractInteractionInfo	ทำหน้าที่ดึงข้อมูลส่วนอินเทอร์แรคชันจากแผนภาพซีเควนซ์มาสร้างเป็นข้อมูลตามโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลอินเทอร์แรคชันในรูปที่ 4.8
getGuard	ทำหน้าที่ดึงข้อมูลส่วนเงื่อนไขการส่งเมสเสจจากข้อความที่ปรากฏในการส่งเมสเสจ
findOperation	ทำหน้าที่หาโอเปอร์เรชันที่มีความสัมพันธ์กับการส่งเมสเสจ โดยในเทมเพลตนี้จะนำ ชื่อคลาส ชื่อโอเปอร์เรชัน รายการพารามิเตอร์ และตัวบ่งชี้ว่าเป็นคอนสตรัคเตอร์หรือไม่ จากการส่งเมสเสจมาสร้างเป็นพารามิเตอร์ในการเรียกเทมเพลต “matchOperation” ให้หาโอเปอร์เรชันที่ต้องการต่อไป
matchOperation	ทำหน้าที่ค้นหาโอเปอร์เรชันในแผนภาพคลาส ด้วยข้อมูล ชื่อคลาส ชื่อโอเปอร์เรชัน รายการพารามิเตอร์ และตัวบ่งชี้ว่าเป็นคอนสตรัคเตอร์หรือไม่ ที่ได้รับจากเทมเพลตที่เป็นผู้เรียกเทมเพลตนี้
findActivator	ทำหน้าที่หาตัวกระตุ้นของการส่งเมสเสจ
toParameters	ทำหน้าที่แปลงรายการของพารามิเตอร์ที่อยู่ในรูปของสายอักขระให้อยู่ในรูปของส่วนของต้นไม้ผลลัพธ์
tokenize	ทำหน้าที่แบ่งข้อมูลในสายอักขระ และสร้างเป็นโทเค็นในรูปของส่วนของต้นไม้ผลลัพธ์
createMsgForEach-Operation	ทำหน้าที่สร้างกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชันตามโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลกลุ่มลำดับการส่งเมสเสจภายในแต่ละโอเปอร์เรชันในรูปที่ 4.9
transformMsgInEach-Activator	ทำหน้าที่สั่งการให้หากลุ่มลำดับการส่งเมสเสจที่อยู่ภายใต้แต่ละตัวกระตุ้นที่มีความสัมพันธ์กับโอเปอร์เรชันที่กำหนด แล้วนำมาทำการรวมเข้าด้วยกัน

ตารางที่ 4.19 หน้าที่ของแต่ละเทมเพลต (ต่อ)

เทมเพลต	หน้าที่
getMsgFromEach-Activator	ทำหน้าที่หากลุ่มลำดับการส่งเมสเสจที่อยู่ภายใต้ตัวกระตุ้นที่กำหนด โดยมีการเรียกเทมเพลต “replaceVariable1” เพื่อทำการแก้ไขการอ้างถึงนามแฝงของพารามิเตอร์ของตัวกระตุ้นของแต่ละการส่งเมสเสจ
replaceVariable1	ทำหน้าที่แทนที่ตัวแปรที่ปรากฏในอักขระที่เป็นนามแฝงของพารามิเตอร์ด้วยชื่อที่แท้จริง ตามรายการคู่ของชื่อที่แท้จริงและนามแฝงของพารามิเตอร์ที่ให้
replaceVariable2	ทำหน้าที่แทนที่ตัวแปรที่ปรากฏในอักขระที่เป็นนามแฝงของพารามิเตอร์ด้วยชื่อที่แท้จริง ตามคู่ของชื่อที่แท้จริงและนามแฝงของพารามิเตอร์ที่ให้
mergeMsg	ทำหน้าที่รวมกลุ่มลำดับการส่งเมสเสจ
makeMergeDecision	ทำหน้าที่เลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ
generateJava	ทำหน้าที่สร้างรหัสคำสั่งภาษาจาวา
generateBehavior	ทำหน้าที่สร้างรหัสคำสั่งเชิงพฤติกรรม
insertLine	ทำหน้าที่แทรกบรรทัดและย่อหน้าตามจำนวนที่กำหนด
insertIndent	ทำหน้าที่แทรกย่อหน้าตามจำนวนที่กำหนด

4.2.5 การออกแบบสถาปัตยกรรมของเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต

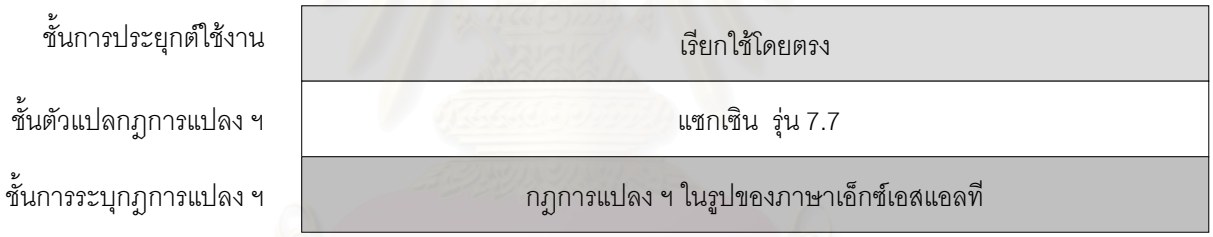
รูปที่ 4.12 เป็นการแสดงสถาปัตยกรรมของเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต ซึ่งมีการใช้ ไมโครซอฟท์ เอ็กซ์เอ็มแอล คอร์ เซอร์วิส รุ่น 4.0 ให้ทำหน้าที่เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์ และใช้ ไมโครซอฟท์ วิชวล เบสิค รุ่น 6.0 (Microsoft Visual Basic 6.0) ในการสร้างส่วนต่อประสานกับผู้ใช้



รูปที่ 4.12 สถาปัตยกรรมของเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต

4.2.6 การออกแบบสถาปัตยกรรมของเครื่องมือสำหรับแพลตฟอร์มจาวา

รูปที่ 4.13 เป็นการแสดงสถาปัตยกรรมของเครื่องมือสำหรับแพลตฟอร์มจาวา ซึ่งมีการใช้ แซกเซิน รุ่น 7.7 ให้ทำหน้าที่เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์ โดยจะให้ผู้ใช้ทำการส่งคำสั่งให้แซกเซินทำการแปลงโดยตรง



รูปที่ 4.13 สถาปัตยกรรมของเครื่องมือสำหรับแพลตฟอร์มจาวา

บทที่ 5
การพัฒนาเครื่องมือสนับสนุนการทำการแปลง ฯ

ในบทนี้จะเป็นการกล่าวถึงการพัฒนาเครื่องมือสนับสนุนการทำการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา ซึ่งเป็นการพัฒนาเครื่องมือตามที่ได้ออกแบบไว้ในบทที่ 4

5.1 เครื่องมือที่ใช้ในการพัฒนา

ในการพัฒนาเครื่องมือสนับสนุนการทำการแปลง ฯ ผู้วิจัยได้ใช้เครื่องมือดังต่อไปนี้

1. เอ็กซ์เอ็มแอลสปาย (XMLSPY) รุ่น 5 สำหรับการเขียนภาษาเอ็กซ์เอสแอลที ดูโครงสร้างข้อมูลเอ็กซ์เอ็มแอล และดูโครงสร้างข้อมูลเอ็กซ์เอ็มไอ
2. ไมโครซอฟท์ เอ็กซ์เอ็มแอล คอร์ เซอร์วิส รุ่น 4.0 สำหรับทำหน้าที่เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์ของเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต
3. แซกเซิน รุ่น 7.7 สำหรับทำหน้าที่เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์ของเครื่องมือสำหรับแพลตฟอร์มจาวา
4. ไมโครซอฟท์ วิชวล เบสิค รุ่น 6.0 สำหรับการสร้างส่วนต่อประสานกับผู้ใช้ของเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต
5. ไมโครซอฟท์ วิชวล อินสทอลเลอร์ (Microsoft Visual Installer) รุ่น 1.1 สำหรับการสร้างตัวติดตั้งของเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต

5.2 การแก้ไขข้อจำกัดของภาษาเอ็กซ์เอสแอลทีที่เป็นอุปสรรคต่อการทำการแปลง ฯ

ในการพัฒนาเครื่องมือนี้ มีการใช้ภาษาเอ็กซ์เอสแอลที รุ่น 1.0 ในการอิมพลีเมนต์การแปลง ฯ ซึ่งภาษาเอ็กซ์เอสแอลทีรุ่นนี้มีข้อจำกัดที่เป็นอุปสรรคต่อการทำการแปลง ฯ อยู่ 2 ประการดังนี้

1. จำกัดการกระทำต่อส่วนของต้นไม้ผลลัพธ์ไว้เพียง 2 การกระทำ คือ การทำสำเนาและการแปลงส่วนของต้นไม้ผลลัพธ์ไปเป็นสายอักขระเท่านั้น ซึ่งไม่เพียงพอต่อกระบวนการทำการแปลง ฯ ซึ่งมีการใช้ส่วนของต้นไม้ผลลัพธ์เป็นผลิตผลระหว่างขั้นตอนต่าง ๆ และต้องการที่จะกระทำกับส่วนของต้นไม้ผลลัพธ์นั้นเช่นเดียวกับที่กระทำได้กับเซตของโหนด

2. ไม่สนับสนุนการมีแฟ้มข้อมูลผลลัพธ์ได้หลายแฟ้ม ซึ่งงานวิจัยนี้จะต้องมีการสร้างแฟ้มข้อมูลจาวาที่เป็นผลลัพธ์จากการแปลง ฯ เท่ากับจำนวนคลาสในแผนภาพคลาส

สำหรับในส่วนของการพัฒนาเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต ซึ่งมีการใช้ไมโครซอฟท์ เอ็กซ์เอ็มแอล คอร์ เซอร์วิส รุ่น 4.0 เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์นั้น เนื่องจากเอ็กซ์เอสแอลทีโปรเซสเซอร์ดังกล่าวถูกพัฒนาตามเอกสารแนะนำเอ็กซ์เอสแอลที รุ่น 1.0 ทำให้มีข้อจำกัดในการทำการแปลง ฯ ดังกล่าว ซึ่งผู้วิจัยได้ทำการแก้ไขข้อจำกัดทั้งสองดังนี้

1. สำหรับการแก้ไขการจำกัดการกระทำต่อส่วนของต้นไม้ผลลัพธ์ไว้เพียง 2 การกระทำนั้น ผู้วิจัยพบว่าทางไมโครซอฟท์ได้ออกแบบฟังก์ชันพิเศษเพิ่มเติมจากมาตรฐานของเอ็กซ์เอสแอลที ชื่อว่า node-set เพื่อทำการแปลงส่วนของต้นไม้ผลลัพธ์ไปเป็นเซตของโหนดได้ ผู้วิจัยจึงจำเป็นจะต้องใช้ฟังก์ชันดังกล่าวทำการแปลงส่วนต้นไม้ผลลัพธ์ให้เป็นเซตของโหนดก่อนที่จะกระทำการกับส่วนของต้นไม้ผลลัพธ์นั้น
2. สำหรับการแก้ไขการไม่สนับสนุนการมีแฟ้มข้อมูลผลลัพธ์ได้หลายแฟ้ม ผู้วิจัยได้ให้ชั้นประยุกต์ใช้งานซึ่งในที่นี้เขียนขึ้นด้วยภาษาวิชวล เบสิค ทำการแบ่งแฟ้มข้อมูลผลลัพธ์ให้

สำหรับในส่วนของการพัฒนาเครื่องมือสำหรับแพลตฟอร์มจาวา ซึ่งมีการใช้ แซกเซิน รุ่น 7.7 เป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์นั้น ถึงแม้ว่าแซกเซินรุ่นนี้จะถูกพัฒนาตามเอกสารแนะนำเอ็กซ์เอสแอลที รุ่น 1.0 เช่นเดียวกับไมโครซอฟท์ เอ็กซ์เอ็มแอล คอร์ เซอร์วิส ก็ตาม แต่ก็ได้เพิ่มคุณสมบัติหลายประการตามที่กำหนดในร่างของเอ็กซ์เอสแอลที รุ่น 2.0 เข้าไว้ด้วย ซึ่งรวมถึงอนุญาตให้กระทำกับส่วนของต้นไม้ผลลัพธ์ได้อย่างเต็มที่ โดยมีการกำหนดชื่อเรียกส่วนของต้นไม้ผลลัพธ์เสียใหม่ว่าต้นไม้ชั่วคราว (Temporary tree) ทำให้ไม่จำเป็นต้องทำการแปลงต้นไม้ชั่วคราวนี้ก่อนที่จะกระทำการใด ๆ กับต้นไม้ชั่วคราวนี้เหมือนกับกรณีของไมโครซอฟท์ เอ็กซ์เอ็มแอล คอร์ เซอร์วิส นอกจากนี้ยังสนับสนุนการมีแฟ้มข้อมูลผลลัพธ์ได้หลายแฟ้มด้วยส่วนย่อยที่ชื่อ xsl:result-document ทำให้สามารถระบุให้สร้างแฟ้มข้อมูลผลลัพธ์หลายแฟ้มในตัวภาษาเอ็กซ์เอสแอลทีได้เลย

5.3 ตัวอย่างการอิมพลีเมนต์การแปลง ฯ

รูปที่ 5.1 เป็นการแสดงรหัสคำสั่งของเทมเพลต “makeMergingDecision” ซึ่งเทมเพลตนี้เป็นการอิมพลีเมนต์กิจกรรมที่ 3.6 ในรูปที่ 4.5 คือ “เลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ” โดยรหัสคำสั่งดังกล่าวเป็นรหัสคำสั่งสำหรับ ไมโครซอฟท์ เอ็กซ์เอ็มแอล คอร์เซอร์วิส ส่วนรหัสคำสั่งสำหรับแซกเซินจะสามารถสร้างได้จากการลบ msxsl:node-set ออกจากรหัสคำสั่ง

เทมเพลต “makeMergeDecision” นี้ จะรับผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจที่มีโครงสร้างดังที่อธิบายในหัวข้อที่ 4.2.3.4 มาทำการเลือกตามกฎที่ 4 ในหัวข้อที่ 3.4 และนำผลลัพธ์ที่ได้รับเลือกมาเป็นผลลัพธ์ของการเรียกเทมเพลตนี้

สำหรับการทำงานของเทมเพลตนี้ จะเริ่มจากตรวจสอบว่าส่วนย่อยแรกสุดเป็นส่วนย่อยชนิดใด ในกรณีที่เป็นส่วนย่อย “Message” ก็จะทำการเรียกตัวเองซ้ำเพื่อหาผลลัพธ์ของส่วนย่อยที่อยู่ถัดจากส่วนย่อยดังกล่าว จากนั้นจะทำการสร้างผลลัพธ์ของเทมเพลตนี้ด้วยการคัดลอกค่าในส่วนย่อยนี้ ตามด้วยส่วนย่อย “Message” ที่อยู่ในผลลัพธ์จากการเรียกตัวเองซ้ำ และต่อด้วยการสร้างส่วนย่อย “Metrics” ที่บ่งบอกค่าของมาตรวัด “จำนวนการส่งเมสเสจ” และ “ความซับซ้อนของสถานการณ์” ที่วัดได้จากส่วนย่อยนี้เป็นต้นไป

ในกรณีที่ส่วนย่อยแรกสุดเป็นส่วนย่อย “Branch” จะทำการเรียกตัวเองซ้ำเพื่อหาค่าของมาตรวัดทั้งสองจากทางเลือกในการรวมคู่การส่งเมสเสจแบบต่าง ๆ ซึ่งจะอยู่ภายใต้ส่วนย่อย “Merge” “OutOldMsg” และ “OutAddingMsg” จากนั้นจะนำผลลัพธ์ที่ได้จากทางเลือกแบบต่าง ๆ มาทำการเรียงลำดับตามค่าของมาตรวัดทั้งสองจากน้อยไปหามากโดยจะให้มาตรวัด “จำนวนการส่งเมสเสจ” มีลำดับความสำคัญสูงกว่ามาตรวัด “ความซับซ้อนของสถานการณ์” แล้วทำการคัดลอกผลลัพธ์ที่อยู่ลำดับแรกจากการเรียงลำดับไปเป็นผลลัพธ์ของเทมเพลตนี้

ในกรณีที่ไม่พบส่วนย่อยเหลืออยู่ก็จะให้ค่าของมาตรวัด “จำนวนการส่งเมสเสจ” เป็น 0 และ “ความซับซ้อนของสถานการณ์” เป็น –1

```
    <xsl:template name="makeMergeDecision">
        <xsl:param name="mergeResults"/>
        <xsl:param name="lastOrigins"/>

        <xsl:variable name="resultsTree" select="msxsl:node-set($mergeResults)"/>
        <xsl:variable name="lastOriginsTree" select="msxsl:node-set($lastOrigins)"/>

        <xsl:choose>


            <!--*************************Found "Message"*********************************-->
            <xsl:when test="(name($resultsTree/*[1]) = 'Message')">

                <xsl:variable name="originsMatchingTest">
                    <xsl:choose>
                        <xsl:when test="count($lastOriginsTree//Origin) = count($resultsTree/*[1]/Origin)">


                            <xsl:for-each select="$lastOriginsTree//Origin">
                                <xsl:variable name="pos" select="position()"/>
                                <xsl:variable name="currentActivator" select="$resultsTree/*[1]//Origin
[$pos]/@activator"/>

                                <xsl:if test="@activator!=$currentActivator">
                                    <xsl:element name="NOTMATCH"/>
                                </xsl:if>
                            </xsl:for-each>

                        </xsl:when>

                        <xsl:otherwise>
                            <xsl:element name="NOTMATCH"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>


                <xsl:variable name="resultFromNext">
                    <!--Recursive Call for Next Element-->
                    <xsl:call-template name="makeMergeDecision">
                        <xsl:with-param name="mergeResults">
                            <xsl:copy-of select="$resultsTree/*[1]/following-sibling::*"/>
                        </xsl:with-param>
                        <xsl:with-param name="lastOrigins">
                            <xsl:copy-of select="$resultsTree/*[1]/Origin"/>
                        </xsl:with-param>
                    </xsl:call-template>
                </xsl:variable>

                <xsl:variable name="resultFromNextTree" select="msxsl:node-set($resultFromNext)"/>


                <!--Out this message-->
                <xsl:copy-of select="$resultsTree/*[1]"/>

                <!--Out messages in next result-->
                <xsl:copy-of select="$resultFromNextTree/Message"/>

                <!--Out the metrics-->
                <xsl:element name="Metrics">
                    <xsl:attribute name="length"><xsl:value-of
select="number($resultFromNextTree/Metrics/@length)+1"/></xsl:attribute>
```

รูปที่ 5.1 รหัสคำสั่งของเทมเพลต “makeMergeDecision”

```
                    <xsl:attribute name="scenarioComplexity">
                        <xsl:choose>
                            <xsl:when test="count(msxsl:node-set($originsMatchingTest)//NOTMATCH)=0">
                                <xsl:value-of
select="number($resultFromNextTree/Metrics/@scenarioComplexity)"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of
select="number($resultFromNextTree/Metrics/@scenarioComplexity)+1"/>
                            </xsl:otherwise>
                        </xsl:choose>

                    </xsl:attribute>
                </xsl:element>

            </xsl:when>


            <!--*************************Found "Branch"*********************************-->
            <xsl:when test="(name($resultsTree/*[1]) = 'Branch') ">

                <xsl:variable name="nextResults">
                    <xsl:for-each select="$resultsTree/*[1]/*"> <!--Merge,OutOldMsg,OutAddingMsg-->
                        <xsl:element name="Result">
                            <!--Recursive Call for Next Element-->
                            <xsl:call-template name="makeMergeDecision">
                                <xsl:with-param name="mergeResults">
                                    <xsl:copy-of select="./*"/>
                                </xsl:with-param>
                                <xsl:with-param name="lastOrigins">
                                    <xsl:copy-of select="$lastOriginsTree"/>
                                </xsl:with-param>
                            </xsl:call-template>
                        </xsl:element>
                    </xsl:for-each>
                </xsl:variable>

                <xsl:variable name="nextResultsTree" select="msxsl:node-set($nextResults)"/>

                <xsl:for-each select="$nextResultsTree/*">
                    <xsl:sort select="./Metrics/@length" data-type="number" order="ascending"/>
                    <xsl:sort select="./Metrics/@scenarioComplexity" data-type="number"
order="ascending"/>
                    <xsl:if test="position()=1">
                        <xsl:copy-of select="./*"/>  <!--Out the best result-->
                    </xsl:if>
                </xsl:for-each>

            </xsl:when>


            <!--*****************************No Element Left*****************************-->

            <xsl:when test="count($resultsTree/*) = 0">
                <xsl:element name="Metrics">
                    <xsl:attribute name="length"><xsl:value-of select="'0'"/></xsl:attribute>
                    <xsl:attribute name="scenarioComplexity"><xsl:value-of select="'-1'"/></xsl:attribute>

                </xsl:element>
            </xsl:when>

        </xsl:choose>

    </xsl:template>
```

รูปที่ 5.1 รหัสคำสั่งของเทมเพลต “makeMergeDecision” (ต่อ)

บทที่ 6
การทดสอบเครื่องมือตามการออกแบบขั้นตอนและกฎการแปลง ฯ

ในบทนี้จะเป็นการทดสอบเครื่องมือที่พัฒนาขึ้น โดยมีวัตถุประสงค์ที่จะตรวจสอบว่าเครื่องมือดังกล่าวสามารถทำงานได้ถูกต้องตามขั้นตอนและกฎการแปลง ฯ ที่ได้ออกแบบไว้ในบทที่ 3 หรือไม่ โดยการทดสอบจะทำโดยให้เครื่องมือดังกล่าวทำการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ตามตัวอย่างที่ใช้ประกอบการอธิบายการออกแบบขั้นตอนและกฎการแปลง ฯ ในบทที่ 3 แล้วทำการเปรียบเทียบผลการแปลงที่ได้กับการอธิบายดังกล่าว โดยตัวอย่างจากบทที่ 3 มี 3 ตัวอย่าง ได้แก่

1. ระบบสั่งซื้อสินค้า (ส่วนเตรียมการจัดส่งสินค้า)
2. ตัวอย่างในขั้นตอนการเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ
3. ตัวอย่างในขั้นตอนการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจ

โดยนอกจากจะทำการทดสอบตามตัวอย่างในบทที่ 3 ดังกล่าวแล้ว ผู้วิจัยยังได้ทำการทดสอบเพิ่มเติมในกรณีที่ผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจมีจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์เท่ากันอีกด้วย

6.1 ระบบสั่งซื้อสินค้า (ส่วนเตรียมการจัดส่งสินค้า)

ในหัวข้อนี้จะเป็นการทดสอบการแปลง ฯ กับตัวอย่างระบบสั่งซื้อสินค้า (ส่วนเตรียมการจัดส่งสินค้า) ตามแผนภาพในรูปที่ 3.2 ถึง 3.4 เพื่อทดสอบขั้นตอนและกฎการแปลง ฯ ทั้งหมดที่ได้ออกแบบไว้

6.1.1 แผนภาพคลาส

แผนภาพคลาสของตัวอย่างนี้ เป็นไปตามรูปที่ 3.2

6.1.2 แผนภาพซีเควนซ์

แผนภาพซีเควนซ์ของตัวอย่างนี้ เป็นไปตามรูปที่ 3.3 และ 3.4

6.1.3 รหัสคำสั่งที่แปลงได้

ผลที่ได้จากการทำการแปลง ฯ ด้วยเครื่องมือ ประกอบด้วยแฟ้มข้อมูล 6 แฟ้ม ได้แก่ DeliveryItem.java Order.java OrderLine.java OrderManager.java ReorderItem.java StockItem.java ดังแสดงด้วยรูปที่ 6.1 ถึง 6.6

```
public class DeliveryItem {
        private OrderLine anOrderLine;
        public  DeliveryItem(OrderLine anOrderLine) {
        }
}
```

รูปที่ 6.1 รหัสคำสั่งของแฟ้มข้อมูล “DeliveryItem.java”

```
public class Order {
        private OrderLine orderLine[];
        public void prepareToDeliver(Date deliveryDate) {
                int $_scenario = 0;
                /**************************************
                  Scenario numbers :
                        # 0 = {Logical View}Normal flow
                        # 1 = {Logical View}Exceptional flow (out of stock)
                 **************************************/

                // Generated behaviour from sequence diagrams :
                for(int i = 0; i < orderLine.length; i++) orderLine[i].prepare(deliveryDate);
        }
        public void deliver() {
        }
}
```

รูปที่ 6.2 รหัสคำสั่งของแฟ้มข้อมูล “Order.java”

```
public class OrderLine {

	private StockItem aStockItem;
	private int amount;
	private String status;

	public void prepare(Date sendDate) {

		boolean hasStock;
		DeliveryItem aDeliveryItem;

		int $_scenario = 0;

		/****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 1 = {Logical View}Exceptional flow (out of stock)
		 ****************************************/


		// Generated behaviour from sequence diagrams :

		hasStock = aStockItem.check(amount, sendDate);

		if ($_scenario == 1) {
			changeStatus("Out of Stock");
		}

		if ($_scenario == 0) {
			aStockItem.remove(amount, sendDate);
			aDeliveryItem = new DeliveryItem(this);
		}
	}

	public void changeStatus(String status) {

	}
}
```

รูปที่ 6.3 รหัสคำสั่งของแฟ้มข้อมูล “OrderLine.java”

```
public class OrderManager {

	private Order order[];
}
```

รูปที่ 6.4 รหัสคำสั่งของแฟ้มข้อมูล “OrderManager.java”

```
public class ReorderItem {

	private StockItem aStockItem;
	private int amount;
	private Date estDateOfArrival;

	public  ReorderItem(StockItem aStockItem, int amount) {

	}
}
```

รูปที่ 6.5 รหัสคำสั่งของแฟ้มข้อมูล “ReorderItem.java”

```
public class StockItem {

	private String name;
	private int currentAmount;
	private int reservedAmount;

	public boolean check(int amount, Date sendDate) {

		return false;

	}

	public void remove(int amount, Date sendDate) {

		int needsReorder;
		ReorderItem aReorderItem;

		// Generated behaviour from sequence diagrams :

		needsReorder = needsToReorder();
		if (needsReorder > 0) aReorderItem = new ReorderItem(this, needsReorder);
	}

	public int needsToReorder() {

		return 0;

	}

}
```

รูปที่ 6.6 รหัสคำสั่งของแฟ้มข้อมูล “StockItem.java”

6.1.4 วิเคราะห์รหัสคำสั่งที่แปลงได้

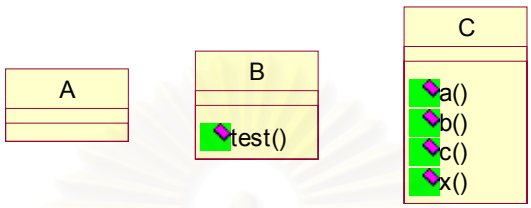
เมื่อทำการตรวจสอบรหัสคำสั่งที่เครื่องมือแปลงได้ พบว่ารหัสคำสั่งดังกล่าวมีความสอดคล้องกันกับคำอธิบายในการออกแบบขั้นตอนและกฎการแปลง ฯ ทั้งหมดที่ได้ออกแบบไว้

6.2 ตัวอย่างในขั้นตอนการเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ

ในหัวข้อนี้จะเป็นการทดสอบการแปลง ฯ กับตัวอย่างในขั้นตอนการเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ เพื่อเปรียบเทียบผลการแปลง ฯ กับคำอธิบายในขั้นตอนการเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ และการสร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ โดยในการทดสอบตัวอย่างนี้ จะต้องมีการสร้างแผนภาพคลาสและแผนภาพซีเควนซ์ให้สอดคล้องกับรูปที่ 3.6 เพื่อให้เครื่องมือทำการแปลง ฯ

6.2.1 แผนภาพคลาส

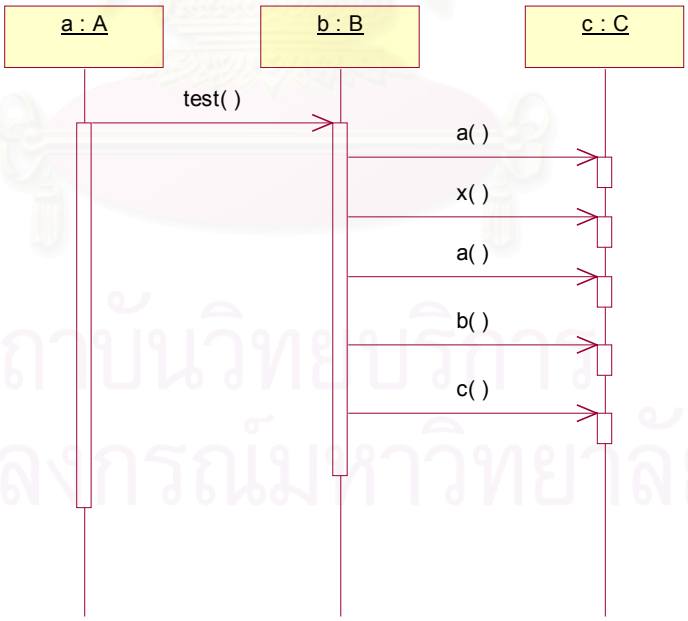
แผนภาพคลาสของตัวอย่างนี้ แสดงได้ดังรูปที่ 6.7



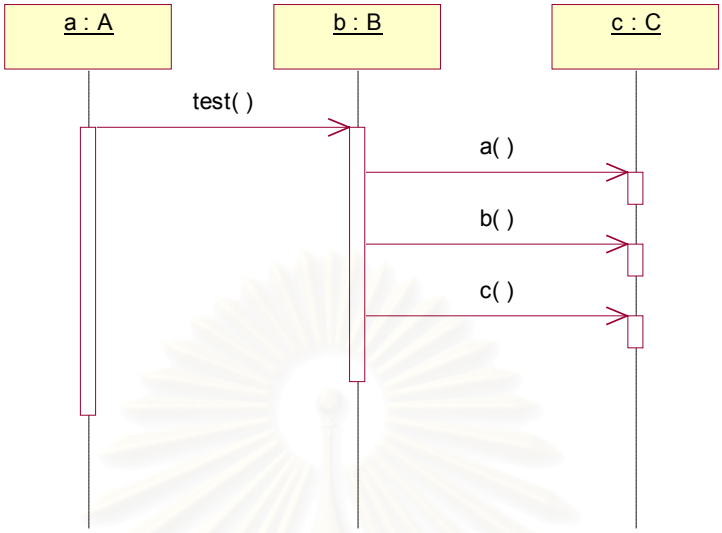
รูปที่ 6.7 แผนภาพคลาสของตัวอย่าง

6.2.2 แผนภาพซีเควนซ์

แผนภาพซีเควนซ์ของตัวอย่างนี้ แสดงได้ดังรูปที่ 6.8 และ 6.9



รูปที่ 6.8 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ 1



รูปที่ 6.9 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ 2

6.2.3 รหัสคำสั่งที่แปลงได้

ในตัวอย่างนี้จะขอแสดงเฉพาะรหัสคำสั่งที่แปลงได้เฉพาะในส่วนของคลาส B ดังรูปที่ 6.10

```
public class B {

        public void test() {

                int $_scenario = 0;

                /*****************************************
                  Scenario numbers :
                        # 0 = {Logical View}Scenario1
                        # 1 = {Logical View}Scenario2
                 *****************************************/


                // Generated behaviour from sequence diagrams :


                if ($_scenario == 0) {
                        c.a();
                        c.x();
                }
                c.a();
                c.b();
                c.c();
        }
}
```

รูปที่ 6.10 รหัสคำสั่งของคลาส B

6.2.4 วิเคราะห์รหัสคำสั่งที่แปลงได้

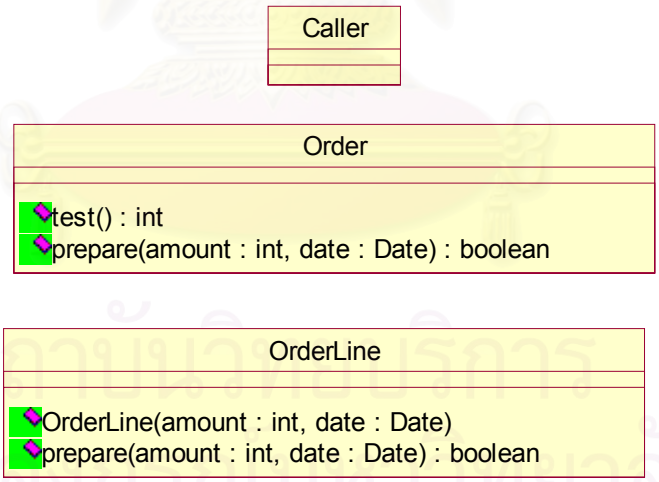
เมื่อทำการตรวจสอบรหัสคำสั่งที่เครื่องมือแปลงได้ พบว่ารหัสคำสั่งดังกล่าวมีความสอดคล้องกันกับคำอธิบายในขั้นตอนการเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ และการสร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์

6.3 ตัวอย่างในขั้นตอนการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจ

ในหัวข้อนี้จะเป็นการทดสอบการแปลง ฯ กับตัวอย่างในขั้นตอนการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจ เพื่อเปรียบเทียบผลการแปลง ฯ กับคำอธิบายดังกล่าว โดยในการทดสอบตัวอย่างนี้ จะต้องมีการสร้างแผนภาพคลาสและแผนภาพซีเควนซ์ให้สอดคล้องกับตัวอย่างการส่งเมสเสจแบบต่าง ๆ ในรูปที่ 3.9 ถึง 3.12 เพื่อให้เครื่องมือทำการแปลง ฯ

6.3.1 แผนภาพคลาส

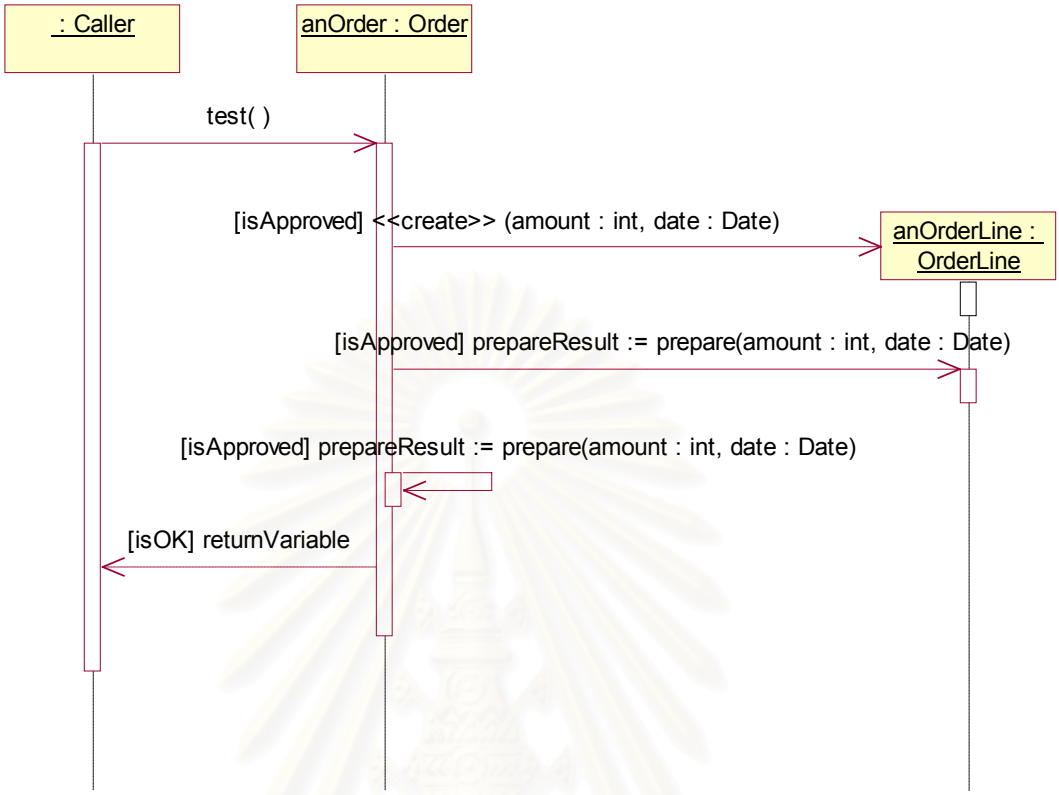
แผนภาพคลาสของตัวอย่างนี้ แสดงได้ดังรูปที่ 6.11



รูปที่ 6.11 แผนภาพคลาสของตัวอย่าง

6.3.2 แผนภาพซีเควนซ์

แผนภาพซีเควนซ์ของตัวอย่างนี้ แสดงได้ดังรูปที่ 6.12



รูปที่ 6.12 แผนภาพซีเควนซ์ของตัวอย่าง

6.3.3 รหัสคำสั่งที่แปลงได้

ในตัวอย่างนี้จะขอแสดงรหัสคำสั่งเฉพาะส่วนที่แสดงผลการแปลงการส่งเมสเสจ ดังรูปที่ 6.13

```
if (isApproved) {
        anOrderLine = new OrderLine(amount, date);
        prepareResult = anOrderLine.prepare(amount, date);
        prepareResult = prepare(amount, date);
}

if (isOK) return returnVariable;
```

รูปที่ 6.13 รหัสคำสั่งแสดงผลการแปลงการส่งเมสเสจ

6.3.4 วิเคราะห์รหัสคำสั่งที่แปลงได้

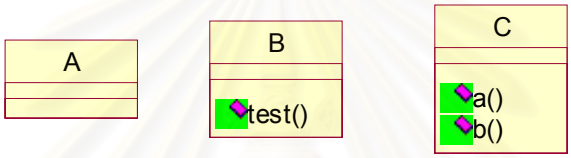
เมื่อทำการตรวจสอบรหัสคำสั่งที่เครื่องมือแปลงได้ พบว่ารหัสคำสั่งดังกล่าวมีความสอดคล้องกันกับคำอธิบายในขั้นตอนการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจ

6.4 การทดสอบกรณีที่ผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจมีจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์เท่ากัน

ในหัวข้อนี้จะเป็นการทดสอบการแปลง ฯ กับตัวอย่างที่ทำให้เกิดผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันเดียวกันมีจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์เท่ากัน

6.4.1 แผนภาพคลาส

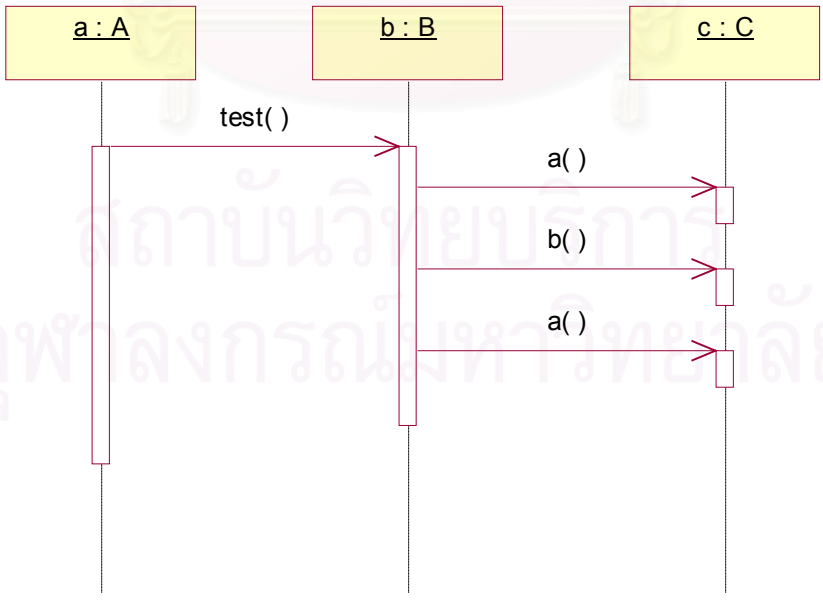
แผนภาพคลาสของตัวอย่างนี้ แสดงได้ดังรูปที่ 6.14



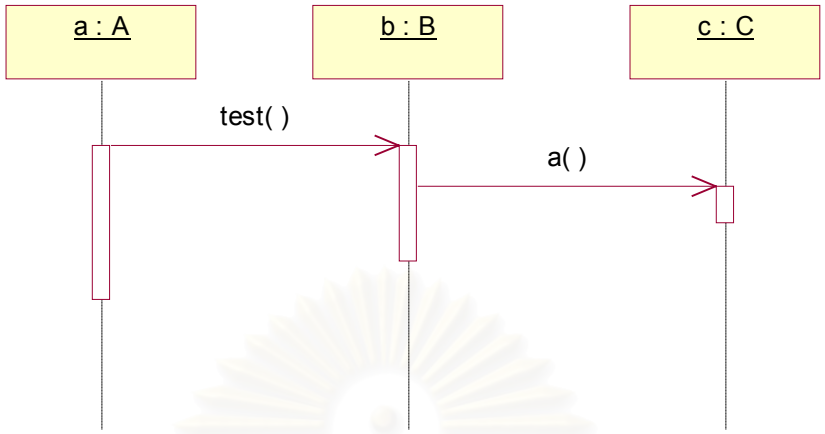
รูปที่ 6.14 แผนภาพคลาส

6.4.2 แผนภาพซีเควนซ์

แผนภาพซีเควนซ์ของตัวอย่างนี้ แสดงได้ดังรูปที่ 6.15 ถึง 6.16



รูปที่ 6.15 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ 1



รูปที่ 6.16 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ 2

6.4.3 รหัสคำสั่งที่แปลงได้

ในตัวอย่างนี้จะขอแสดงเฉพาะรหัสคำสั่งที่แปลงได้เฉพาะในส่วนของคลาส B ดังรูปที่ 6.17

```
public class B {

        public void test() {

                int $_scenario = 0;

                /*****************************************
                  Scenario numbers :
                        # 0 = {Logical View}Scenario1
                        # 1 = {Logical View}Scenario2
                 *****************************************/


                // Generated behaviour from sequence diagrams :

                c.a();

                if ($_scenario == 0) {
                        c.b();
                        c.a();
                }
        }

}
```

รูปที่ 6.17 รหัสคำสั่งของคลาส B

6.4.4 วิเคราะห์รหัสคำสั่งที่แปลงได้

เมื่อพิจารณากฎที่ 4 ในหัวข้อที่ 3.4 “การเลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ” ที่ให้เลือกผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจที่มีจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์น้อยที่สุด จะพบว่าตัวอย่างนี้สามารถถูกแปลงได้เป็นรหัสคำสั่งอีกรูปแบบหนึ่งดังแสดงในรูปที่ 6.18 โดยที่ทั้งรหัสคำสั่งในรูปที่ 6.17 และ 6.18 ต่างก็มีจำนวนการส่งเมสเสจเท่ากับ 3 และความซับซ้อนของสถานการณ์เท่ากับ 1 เช่นเดียวกัน ซึ่งไม่ว่าเครื่องมือจะให้รหัสคำสั่งตามรูปใดใน 2 รูปนี้ก็ถือได้ว่าถูกต้องทั้งสิ้น

```
public class B {

        public void test() {

                int $_scenario = 0;

                /*****************************************
                  Scenario numbers :
                        # 0 = {Logical View}Scenario1
                        # 1 = {Logical View}Scenario2
                 *****************************************/


                // Generated behaviour from sequence diagrams :


                if ($_scenario == 0) {
                        c.a();
                        c.b();
                }

                c.a();
        }

}
```

รูปที่ 6.18 รหัสคำสั่งของคลาส B ที่เป็นไปได้รูปแบบที่ 2

ส่วนการที่เครื่องมือให้รหัสคำสั่งตามรูปที่ 6.17 นั้น สามารถวิเคราะห์ได้จากผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจของตัวอย่างนี้ซึ่งแสดงได้ดังในรูปที่ 6.19 กับรหัสคำสั่งของเครื่องมือในส่วนที่ทำการเลือกผลลัพธ์ในรูปที่ 6.20 ซึ่งเป็นส่วนหนึ่งของเทมเพลต “makeMergeDecision” ที่แสดงในรูปที่ 5.1 โดยเมื่อเครื่องมือพบกับส่วนย่อย “Branch” ในรูปที่ 6.19 ลำดับแรกที่แสดงด้วยตัวอักษรแบบหนาแล้ว (ส่วนย่อยนี้จะเป็นการบอกถึงการเกิดทางเลือกในการรวมคู่การส่งเมสเสจแรก ซึ่งผลจากทางเลือกในการรวมแบบต่าง ๆ จะอยู่ภายใต้ส่วนย่อย “Merge” “OutOldMsg” และ “OutAddingMsg” ที่แสดงด้วยตัวอักษรแบบหนา) ก็จะทำการสร้างผลการรวมแบบต่าง ๆ ด้วยการเรียกเทมเพลตตัวเองซ้ำกับส่วนย่อย “Merge” “OutOldMsg”

และ “OutAddingMsg” ที่แสดงด้วยตัวอักษรแบบหนา หลังจากนั้นจะทำการเรียงลำดับผลการรวมแบบต่าง ๆ ตามจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์ของผลการรวมแต่ละแบบจากน้อยไปมาก ซึ่งผลการรวมเหล่านี้สามารถหาได้โดยรหัสคำสั่งที่ทำหน้าที่เรียงลำดับดังกล่าวได้ถูกเน้นด้วยตัวอักษรแบบหนาในรูปที่ 6.20 ซึ่งในกรณีนี้ผลการรวมแบบที่ 1 (ซึ่งสอดคล้องกับรหัสคำสั่งในรูปที่ 6.17) อยู่ในส่วนย่อย “Merge” และผลการรวมแบบที่ 2 (ซึ่งสอดคล้องกับรหัสคำสั่งในรูปที่ 6.18) อยู่ในส่วนย่อย “OutOldMsg” มีค่าของมาตรวัดทั้งสองน้อยที่สุดเท่ากัน โดยจากการทดสอบการเรียงลำดับของเอ็กซ์เอสแอลทีโปรเซสเซอร์ทั้งสองที่ใช้ในงานวิจัยนี้ พบว่าเมื่อส่วนย่อยที่นำมาทำการเรียงลำดับมีค่าที่ใช้ในการเรียงลำดับเท่ากัน ผลการเรียงที่ได้จะยังคงรักษาลำดับเดิมไว้ ซึ่งในกรณีนี้ส่วนย่อย “Merge” จะอยู่ก่อน “OutOldMsg” จึงทำให้ผลการรวมแบบที่ 1 ถูกเรียงไว้เป็นลำดับแรก ต่อด้วยผลการรวมแบบที่ 2 ซึ่งเมื่อพิจารณารหัสคำสั่งแล้วจะพบว่า ผลการรวมที่ถูกเรียงไว้เป็นลำดับแรกจะถูกเลือกไปเป็นผลลัพธ์ เครื่องมือจึงได้เลือกผลลัพธ์แบบที่ 1 ไปทำการแปลงจนได้เป็นรหัสคำสั่งตามรูปที่ 6.17 นั่นเอง

```
<mergeResults>
    <Branch>
        <Merge>
            <Message id="G.7" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                <Origin activator="G.6"/>
                <Origin activator="G.16"/>
            </Message>
            <Message id="G.8" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="b( )" operation="S.108.2106.40.4" operationName="b" returnValue="" returnType="" guard="">
                <Origin activator="G.6"/>
            </Message>
            <Message id="G.9" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                <Origin activator="G.6"/>
            </Message>
        </Merge>
        <OutOldMsg>
            <Message id="G.7" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                <Origin activator="G.6"/>
            </Message>
            <Branch>
                <OutOldMsg>
                    <Message id="G.8" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="b( )" operation="S.108.2106.40.4" operationName="b" returnValue="" returnType="" guard="">
                        <Origin activator="G.6"/>
                    </Message>
                    <Branch>
                        <Merge>
                            <Message id="G.9" type="callAction" receiverType="S.108.2106.40.2"
receiverName="c" statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue=""
returnType="" guard="">
                                <Origin activator="G.6"/>
                                <Origin activator="G.16"/>
                            </Message>
                        </Merge>
```

การส่งเมสเสจที่ 1 ของผลการรวมแบบที่ 1

การส่งเมสเสจที่ 2 ของผลการรวมแบบที่ 1

การส่งเมสเสจที่ 3 ของผลการรวมแบบที่ 1

การส่งเมสเสจที่ 1 ของผลการรวมแบบที่ 2

การส่งเมสเสจที่ 2 ของผลการรวมแบบที่ 2

การส่งเมสเสจที่ 3 ของผลการรวมแบบที่ 2

รูปที่ 6.19 ผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ

```
                        <OutOldMsg>
                            <Message id="G.9" type="callAction" receiverType="S.108.2106.40.2"
receiverName="c" statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue=""
returnType="" guard="">
                                <Origin activator="G.6"/>
                            </Message>
                            <Message id="G.17" type="callAction" receiverType="S.108.2106.40.2"
receiverName="c" statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue=""
returnType="" guard="">
                                <Origin activator="G.16"/>
                            </Message>
                        </OutOldMsg>
                        <OutAddingMsg>
                            <Message id="G.17" type="callAction" receiverType="S.108.2106.40.2"
receiverName="c" statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue=""
returnType="" guard="">
                                <Origin activator="G.16"/>
                            </Message>
                            <Message id="G.9" type="callAction" receiverType="S.108.2106.40.2"
receiverName="c" statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue=""
returnType="" guard="">
                                <Origin activator="G.6"/>
                            </Message>
                        </OutAddingMsg>
                    </Branch>
                </OutOldMsg>
                <OutAddingMsg>
                    <Message id="G.17" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                        <Origin activator="G.16"/>
                    </Message>
                    <Message id="G.8" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="b( )" operation="S.108.2106.40.4" operationName="b" returnValue="" returnType="" guard="">
                        <Origin activator="G.6"/>
                    </Message>
                    <Message id="G.9" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                        <Origin activator="G.6"/>
                    </Message>
                </OutAddingMsg>
            </Branch>
        </OutOldMsg>
        <OutAddingMsg>
            <Message id="G.17" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                <Origin activator="G.16"/>
            </Message>
            <Message id="G.7" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                <Origin activator="G.6"/>
            </Message>
            <Message id="G.8" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="b( )" operation="S.108.2106.40.4" operationName="b" returnValue="" returnType="" guard="">
                <Origin activator="G.6"/>
            </Message>
            <Message id="G.9" type="callAction" receiverType="S.108.2106.40.2" receiverName="c"
statement="a( )" operation="S.108.2106.40.3" operationName="a" returnValue="" returnType="" guard="">
                <Origin activator="G.6"/>
            </Message>
        </OutAddingMsg>
    </Branch>
</mergeResults>
```

รูปที่ 6.19 ผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจ (ต่อ)

```
<!--*************************Found "Branch"*********************************-->
<xsl:when test="(name($resultsTree/*[1]) = 'Branch') ">

    <xsl:variable name="nextResults">
        <xsl:for-each select="$resultsTree/*[1]/*"> <!--Merge,OutOldMsg,OutAddingMsg-->
            <xsl:element name="Result">
                <!--Recursive Call for Next Element-->
                <xsl:call-template name="makeMergeDecision">
                    <xsl:with-param name="mergeResults">
                        <xsl:copy-of select="./*"/>
                    </xsl:with-param>
                    <xsl:with-param name="lastOrigins">
                        <xsl:copy-of select="$lastOriginsTree"/>
                    </xsl:with-param>
                </xsl:call-template>
            </xsl:element>
        </xsl:for-each>
    </xsl:variable>

    <xsl:variable name="nextResultsTree" select="msxsl:node-set($nextResults)"/>

    <xsl:for-each select="$nextResultsTree/*">
        <xsl:sort select="./Metrics/@length" data-type="number" order="ascending"/>
        <xsl:sort select="./Metrics/@scenarioComplexity" data-type="number"
order="ascending"/>
        <xsl:if test="position()=1">
            <xsl:copy-of select="./*"/>  <!--Out the best result-->
        </xsl:if>
    </xsl:for-each>
```

รูปที่ 6.20 รหัสคำสั่งของเครื่องมือในส่วนที่ทำการเลือกผลลัพธ์

6.5 สรุปผลการทดสอบ

จากผลการทดสอบด้วยตัวอย่างที่ใช้ประกอบการอธิบายขั้นตอนและกฎการแปลง ฯ ที่ได้ออกแบบขึ้นมาในบทที่ 3 สามารถแสดงได้ว่าเครื่องมือที่พัฒนาขึ้นมาสามารถทำการแปลง ฯ ตามขั้นตอนและกฎการแปลง ฯ ดังกล่าวได้อย่างถูกต้อง และยังสามารถทำการแปลงตัวอย่างที่ทำให้เกิดผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันเดียวกันมีจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์เท่ากันได้อย่างถูกต้องอีกด้วย

บทที่ 7
การทดสอบเครื่องมือด้วยกรณีศึกษา

ในบทนี้จะเป็นการทดสอบเครื่องมือที่พัฒนาขึ้นกับกรณีศึกษา 2 กรณี โดยมีวัตถุประสงค์ที่จะตรวจสอบว่าเครื่องมือดังกล่าวสามารถทำการแปลง ฯ ได้ถูกต้องและครบถ้วนตามที่ได้ระบุไว้ในขอบเขตของการวิจัยหรือไม่

7.1 วิธีการทดสอบ

การทดสอบเครื่องมือ จะทำโดยให้เครื่องมือดังกล่าวทำการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ของกรณีศึกษา 2 กรณี โดยในแต่ละกรณีศึกษา จะต้องสามารถแสดงการทดสอบในประเด็นดังต่อไปนี้ได้

1. การรวมพฤติกรรมของโอเปอร์เรชันที่ถูกระบุอยู่ในสถานการณ์ที่แตกต่างกันอย่างน้อย 2 สถานการณ์
2. การแปลงแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันอย่างน้อย 3 ระดับ

ซึ่งกรณีศึกษาทั้งสองจะอยู่ในรูปของแฟ้มข้อมูลเอ็กซ์เอ็มไอรุ่น 1.1 ของยูเอ็มแอลรุ่น 1.3 ที่ถูกสร้างขึ้นด้วย เรชันแนล โรส รุ่น 2002 ด้วยเครื่องมือเสริมยูนิซิส โรส เอ็กซ์เอ็มแอล ทูล รุ่น 1.3.6

โดยจะถือว่าเครื่องมือสามารถให้ผลการแปลงที่ถูกต้องก็ต่อเมื่อรหัสคำสั่งที่ได้เป็นไปตามเงื่อนไขดังต่อไปนี้ทั้งหมด

1. มีความสอดคล้องกับรหัสคำสั่งที่ทำการแปลงด้วยตนเองตามขั้นตอนและกฎการแปลง ฯ ของงานวิจัยนี้
2. สามารถผ่านการคอมไพล์ด้วยคอมไพเลอร์ที่ทำตามข้อกำหนดของภาษาจาวารุ่นที่ 2

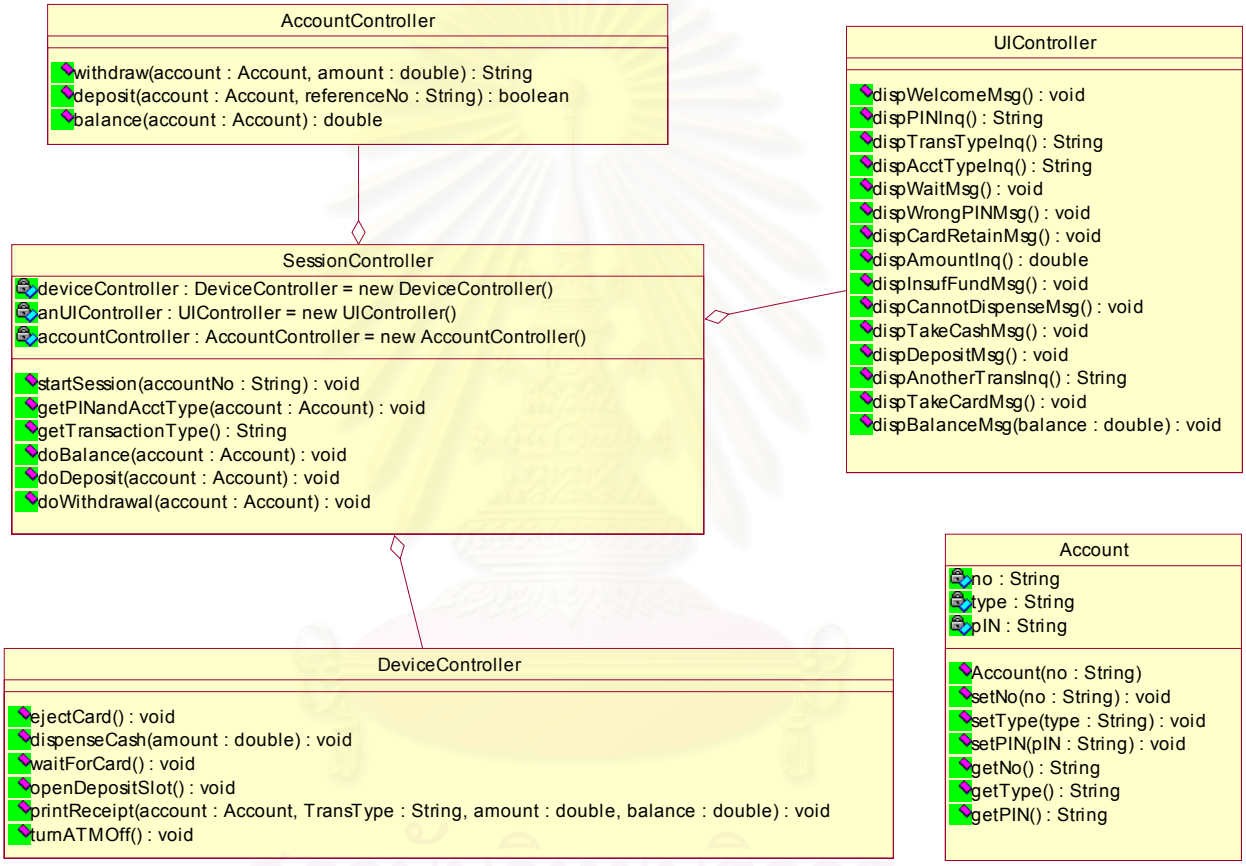
7.2 กรณีศึกษาที่ 1 ระบบเอทีเอ็มอย่างง่าย

กรณีศึกษาที่ 1 ที่ขอนำมาเป็นระบบทดสอบคือระบบเอทีเอ็มอย่างง่าย ซึ่งประกอบไปด้วยสถานการณ์ทั้งหมด 4 สถานการณ์ ได้แก่

1. ผู้ใช้ทำรายการถอนเงินได้สำเร็จ
2. ผู้ใช้ทำรายการถอนเงินไม่สำเร็จ
3. ผู้ใช้ทำรายการสอบถามยอดเงินคงเหลือ
4. ผู้ใช้ทำรายการฝากเงิน

7.2.1 แผนภาพคลาสของระบบ

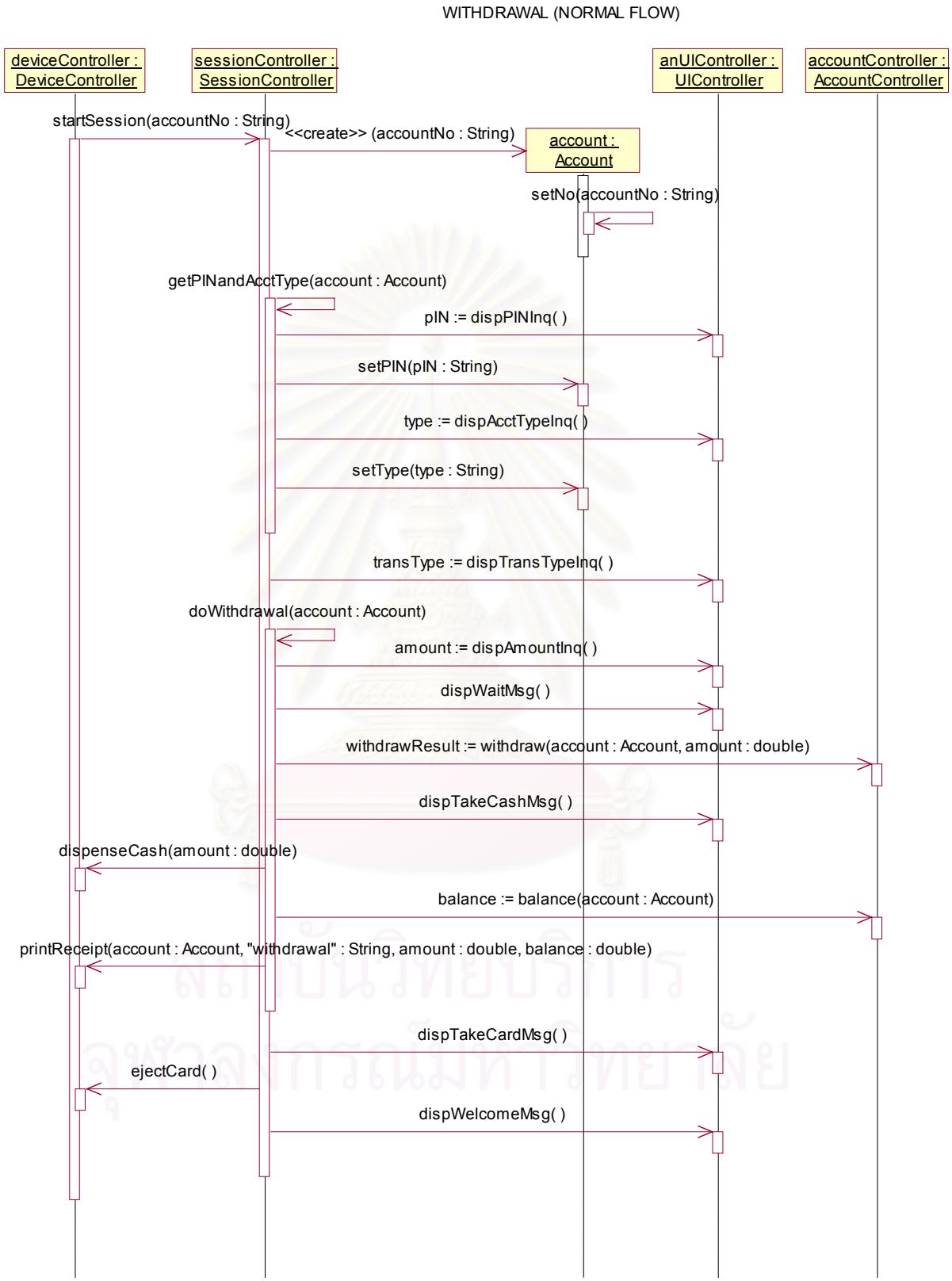
แผนภาพคลาสของระบบเอทีเอ็มอย่างง่าย ซึ่งประกอบไปด้วยคลาส Account AccountController DeviceController SessionController และ UIController สามารถแสดงได้ดังรูปที่ 7.1



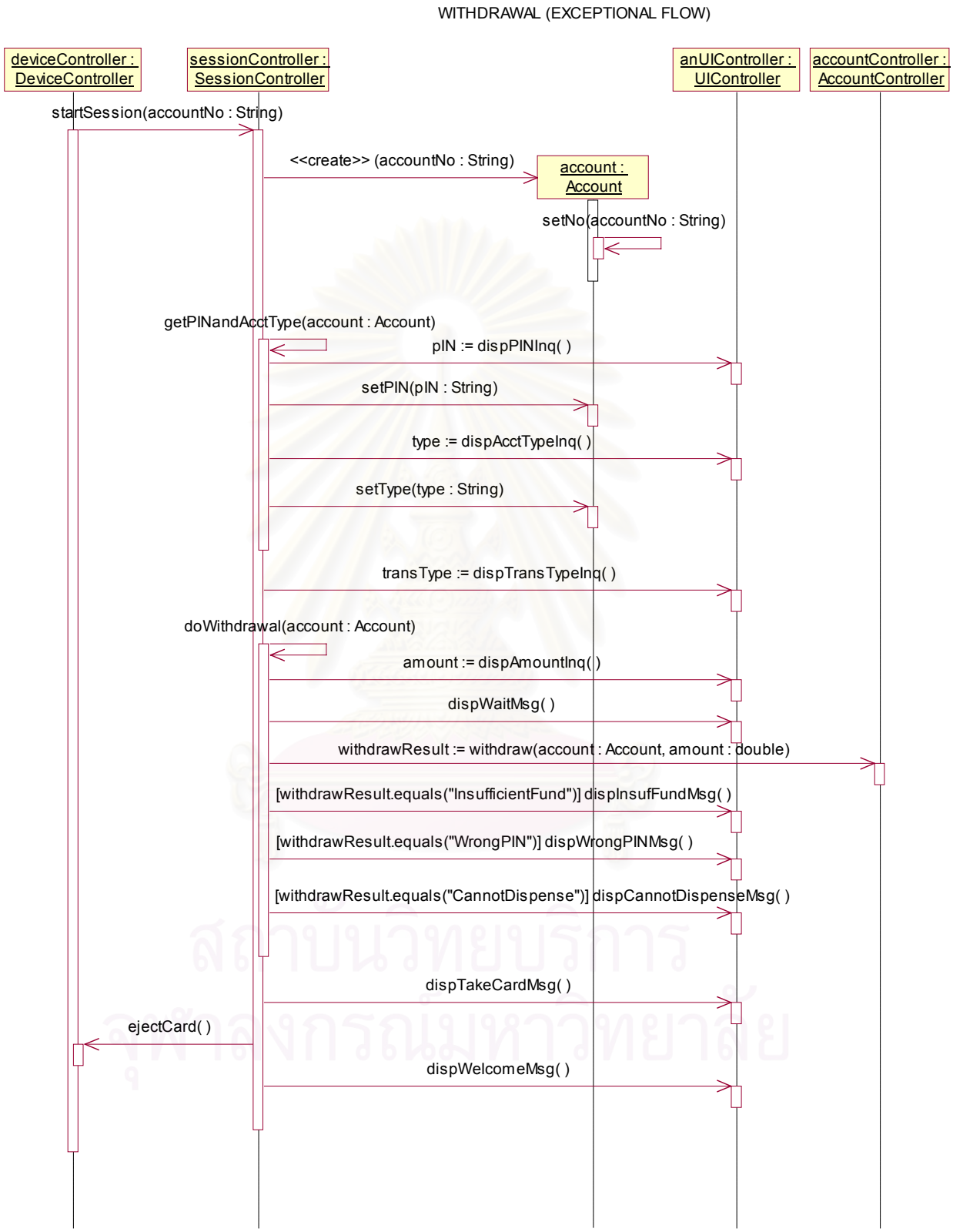
รูปที่ 7.1 แผนภาพคลาสของระบบเอทีเอ็มอย่างง่าย

7.2.2 แผนภาพซีเควนซ์ของระบบ

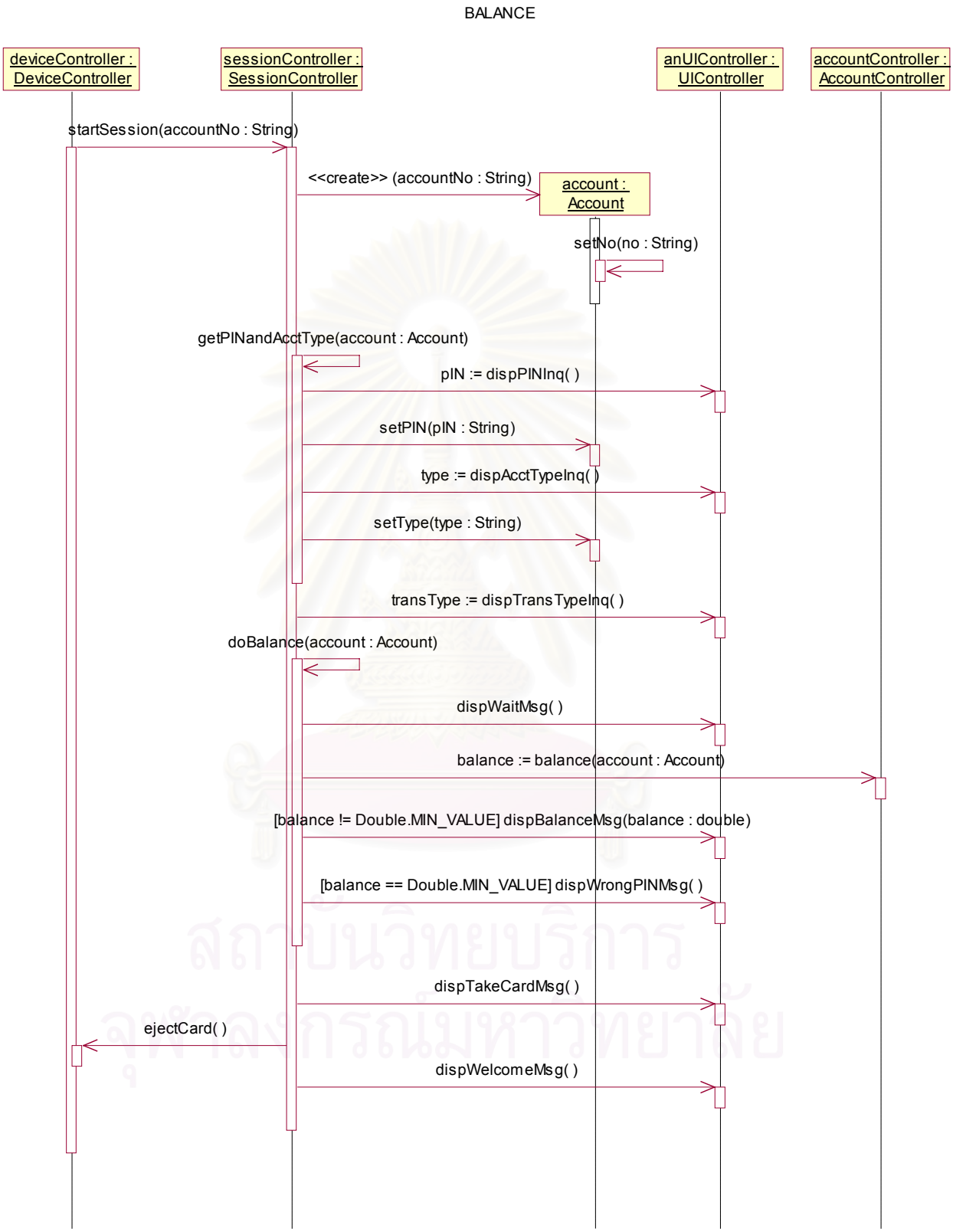
แผนภาพซีเควนซ์สำหรับแต่ละสถานการณ์ ได้แก่ ผู้ใช้ทำรายการถอนเงินได้สำเร็จ ผู้ใช้ทำรายการถอนเงินไม่สำเร็จ ผู้ใช้ทำรายการสอบถามยอดเงินคงเหลือ และ ผู้ใช้ทำรายการฝากเงิน แสดงได้ดังรูปที่ 7.2 ถึงรูปที่ 7.5 ตามลำดับ



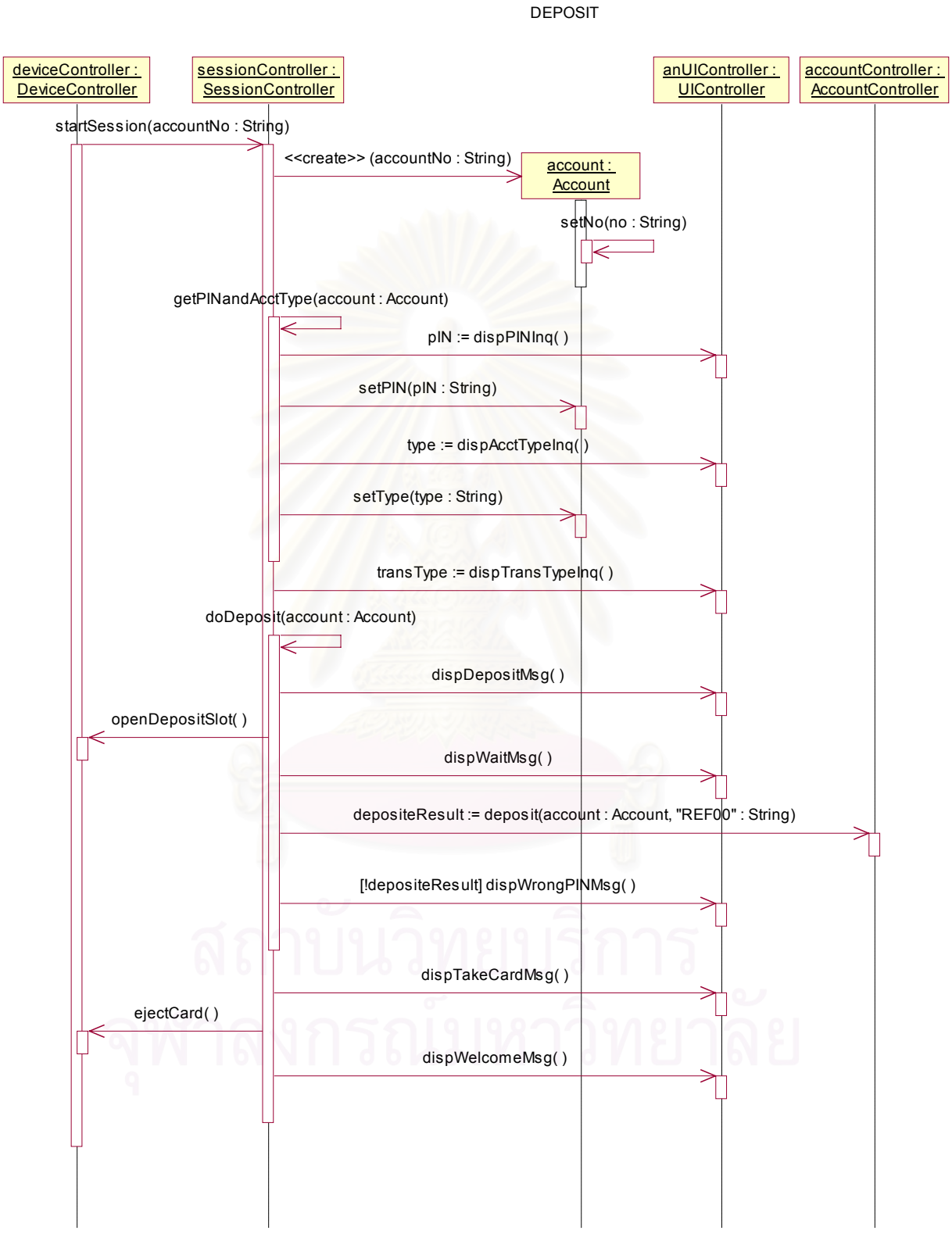
รูปที่ 7.2 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ผู้ใช้ทำรายการถอนเงินได้สำเร็จ



รูปที่ 7.3 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ผู้ใช้ทำรายการถอนเงินไม่สำเร็จ



รูปที่ 7.4 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ผู้ใช้ทำรายการสอบถามยอดเงินคงเหลือ



รูปที่ 7.5 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ผู้ใช้ทำรายการฝากเงิน

7.2.3 รหัสคำสั่งที่แปลงได้

ผลที่ได้จากการทำการแปลง ฯ ด้วยเครื่องมือ ประกอบด้วยแฟ้มข้อมูล 5 แฟ้ม ได้แก่ Account.java AccountController.java DeviceController.java SessionController.java และ UIController.java โดยมีรหัสคำสั่งของแต่ละแฟ้มข้อมูลแสดงดังรูปที่ 7.6 ถึง 7.10

```
public class Account {

	private String no;
	private String type;
	private String pIN;

	public  Account(String no) {

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Withdrawal (normal flow)
			# 1 = {Logical View}Withdrawal (exceptional flow)
			# 2 = {Logical View}Balance
			# 3 = {Logical View}Deposit
		 *****************************************/


		// Generated behaviour from sequence diagrams :

		setNo(no);
	}

	public void setNo(String no) {
	}

	public void setType(String type) {
	}

	public void setPIN(String pIN) {
	}

	public String getNo() {

		return null;
	}

	public String getType() {

		return null;
	}

	public String getPIN() {

		return null;
	}
}
```

รูปที่ 7.6 รหัสคำสั่งของแฟ้มข้อมูล “Account.java”

```
public class AccountController {
	static public String withdraw(Account account, double amount) {
		return null;
	}
	public boolean deposit(Account account, String referenceNo) {
		return false;
	}
	static public double balance(Account account) {
		return 0.0;
	}
}
```

รูปที่ 7.7 รหัสคำสั่งของแฟ้มข้อมูล “AccountController.java”

```
public class DeviceController {
	public void ejectCard() {
	}
	public void dispenseCash(double amount) {
	}
	public void waitForCard() {
	}
	public void openDepositSlot() {
	}
	public void printReceipt(Account account, String TransType, double amount, double balance) {
	}
	public void turnATMOff() {
	}
}
```

รูปที่ 7.8 รหัสคำสั่งของแฟ้มข้อมูล “DeviceController.java”

```
public class SessionController {

	private DeviceController deviceController = new DeviceController();
	private UIController anUIController = new UIController();
	private AccountController accountController = new AccountController();

	public void startSession(String accountNo) {

		Account account;
		String transType;

		int $_scenario = 0;

		/****************************************
		  Scenario numbers :
			# 0 = {Logical View}Withdrawal (normal flow)
			# 1 = {Logical View}Withdrawal (exceptional flow)
			# 2 = {Logical View}Balance
			# 3 = {Logical View}Deposit
		 ****************************************/


		// Generated behaviour from sequence diagrams :

		account = new Account(accountNo);
		getPINandAcctType(account);
		transType = anUIController.dispTransTypeInq();

		if ($_scenario == 3) {
			doDeposit(account);
		}

		if ($_scenario == 2) {
			doBalance(account);
		}

		if ($_scenario == 0 || $_scenario == 1) {
			doWithdrawal(account);
		}
		anUIController.dispTakeCardMsg();
		deviceController.ejectCard();
		anUIController.dispWelcomeMsg();
	}

	public void getPINandAcctType(Account account) {

		String pIN;
		String type;

		int $_scenario = 0;

		/****************************************
		  Scenario numbers :
			# 0 = {Logical View}Withdrawal (normal flow)
			# 1 = {Logical View}Withdrawal (exceptional flow)
			# 2 = {Logical View}Balance
			# 3 = {Logical View}Deposit
		 ****************************************/
```

รูปที่ 7.9 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java”

```
		// Generated behaviour from sequence diagrams :

		pIN = anUIController.dispPINInq();
		account.setPIN(pIN);
		type = anUIController.dispAcctTypeInq();
		account.setType(type);
	}

	public String getTransactionType() {

		return null;

	}

	public void doBalance(Account account) {

		double balance;

		// Generated behaviour from sequence diagrams :

		anUIController.dispWaitMsg();
		balance = accountController.balance(account);
		if (balance != Double.MIN_VALUE) anUIController.dispBalanceMsg(balance);
		if (balance == Double.MIN_VALUE) anUIController.dispWrongPINMsg();
	}

	public void doDeposit(Account account) {

		boolean depositeResult;

		// Generated behaviour from sequence diagrams :

		anUIController.dispDepositMsg();
		deviceController.openDepositSlot();
		anUIController.dispWaitMsg();
		depositeResult = accountController.deposit(account, "REF00");
		if (!depositeResult) anUIController.dispWrongPINMsg();
	}

	public void doWithdrawal(Account account) {

		double amount;
		String withdrawResult;
		double balance;

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Withdrawal (normal flow)
			# 1 = {Logical View}Withdrawal (exceptional flow)
		 *****************************************/


		// Generated behaviour from sequence diagrams :

		amount = anUIController.dispAmountInq();
		anUIController.dispWaitMsg();
		withdrawResult = accountController.withdraw(account, amount);
```

รูปที่ 7.9 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” (ต่อ)

```
                    if ($_scenario == 1) {
                            if (withdrawResult.equals("InsufficientFund"))
                                    anUIController.dispInsufFundMsg();
                            if (withdrawResult.equals("WrongPIN"))
                                    anUIController.dispWrongPINMsg();
                            if (withdrawResult.equals("CannotDispense"))
                                    anUIController.dispCannotDispenseMsg();
                    }


                    if ($_scenario == 0) {
                            anUIController.dispTakeCashMsg();
                            deviceController.dispenseCash(amount);
                            balance = accountController.balance(account);
                            deviceController.printReceipt(account, "withdrawal", amount, balance);
                    }
            }
}
```

รูปที่ 7.9 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” (ต่อ)

```
public class UIController {

        public void dispWelcomeMsg() {

        }

        public String dispPINInq() {

                return null;

        }

        public String dispTransTypeInq() {

                return null;

        }

        public String dispAcctTypeInq() {

                return null;

        }

        public void dispWaitMsg() {

        }

        public void dispWrongPINMsg() {

        }
```

รูปที่ 7.10 รหัสคำสั่งของแฟ้มข้อมูล “UIController.java”

```
	public void dispCardRetainMsg() {
	}
	public double dispAmountInq() {
		return 0.0;
	}
	public void dispInsufFundMsg() {
	}
	public void dispCannotDispenseMsg() {
	}
	public void dispTakeCashMsg() {
	}
	public void dispDepositMsg() {
	}
	public String dispAnotherTransInq() {
		return null;
	}
	public void dispTakeCardMsg() {
	}
	public void dispBalanceMsg(double balance) {
	}
}
```

รูปที่ 7.10 รหัสคำสั่งของแฟ้มข้อมูล “UIController.java” (ต่อ)

7.2.4 ผลการทดสอบ

เครื่องมือสามารถทำการแปลงแฟ้มข้อมูลของกรณีศึกษาที่ 1 และให้ผลการแปลงเป็นแฟ้มข้อมูลรหัสคำสั่งภาษาจาวาได้ดังที่แสดงไว้หัวข้อที่ 7.2.3 ซึ่งเมื่อเปรียบเทียบรหัสคำสั่งที่แปลงได้จากเครื่องมือกับรหัสคำสั่งที่ทำการแปลงด้วยตนเองแล้วพบว่ามีความสอดคล้องกัน และเมื่อนำไปคอมไพล์ด้วยคอมไพเลอร์ที่ทำตามข้อกำหนดของภาษาจาวารุ่นที่ 2 พบว่าสามารถทำการคอมไพล์ได้สำเร็จ

7.2.5 วิเคราะห์ผลการทดสอบ

จากผลการทดสอบ พบว่าเครื่องมือสามารถทำการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ของกรณีศึกษาที่ 1 ได้อย่างถูกต้อง อย่างไรก็ตามก่อนที่จะนำรหัสคำสั่งนี้ไปใช้จริง ผู้ใช้จะต้องเพิ่มรหัสคำสั่งส่วนของการเลือกสถานการณ์ด้วยตนเองลงในคลาส SessionController อยู่สองโอเปอร์เรชันดังแสดงในรูปที่ 7.11 โดยรหัสคำสั่งส่วนที่ผู้ใช้ต้องทำเพิ่มได้ถูกเน้นด้วยตัวอักษรแบบหนา

```
public class SessionController {

        private DeviceController deviceController = new DeviceController();
        private UIController anUIController = new UIController();
        private AccountController accountController = new AccountController();

        public void startSession(String accountNo) {

                Account account;
                String transType;

                int $_scenario = 0;

                /****************************************
                  Scenario numbers :
                          # 0 = {Logical View}Withdrawal (normal flow)
                          # 1 = {Logical View}Withdrawal (exceptional flow)
                          # 2 = {Logical View}Balance
                          # 3 = {Logical View}Deposit
                 ****************************************/

                // Generated behaviour from sequence diagrams :

                account = new Account(accountNo);
                getPINandAcctType(account);
                transType = anUIController.dispTransTypeInq();

                if (transType.equals("withdrawal"))
                        $_scenario = 0;
                else if (transType.equals("balance"))
                        $_scenario = 2;
                else if (transType.equals("deposit"))
                        $_scenario = 3;

                if ($_scenario == 3) {
                        doDeposit(account);
                }

                if ($_scenario == 2) {
                        doBalance(account);
                }

                if ($_scenario == 0 || $_scenario == 1) {
                        doWithdrawal(account);
                }
```

รูปที่ 7.11 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” หลังทำการเพิ่มด้วยตนเอง

```
		anUIController.dispTakeCardMsg();
		deviceController.ejectCard();
		anUIController.dispWelcomeMsg();
	}

	public void getPINandAcctType(Account account) {

		String pIN;
		String type;

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Withdrawal (normal flow)
			# 1 = {Logical View}Withdrawal (exceptional flow)
			# 2 = {Logical View}Balance
			# 3 = {Logical View}Deposit
		 *****************************************/


		// Generated behaviour from sequence diagrams :

		pIN = anUIController.dispPINInq();
		account.setPIN(pIN);
		type = anUIController.dispAcctTypeInq();
		account.setType(type);
	}

	public String getTransactionType() {

		return null;

	}

	public void doBalance(Account account) {

		double balance;

		// Generated behaviour from sequence diagrams :

		anUIController.dispWaitMsg();
		balance = accountController.balance(account);
		if (balance != Double.MIN_VALUE) anUIController.dispBalanceMsg(balance);
		if (balance == Double.MIN_VALUE) anUIController.dispWrongPINMsg();
	}

	public void doDeposit(Account account) {

		boolean depositeResult;

		// Generated behaviour from sequence diagrams :

		anUIController.dispDepositMsg();
		deviceController.openDepositSlot();
		anUIController.dispWaitMsg();
		depositeResult = accountController.deposit(account, "REF00");
		if (!depositeResult) anUIController.dispWrongPINMsg();
	}
```

รูปที่ 7.11 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” หลังทำการเพิ่มด้วยตนเอง (ต่อ)

```
        public void doWithdrawal(Account account) {

                double amount;
                String withdrawResult;
                double balance;

                int $_scenario = 0;

                /****************************************
                  Scenario numbers :
                        # 0 = {Logical View}Withdrawal (normal flow)
                        # 1 = {Logical View}Withdrawal (exceptional flow)
                 ****************************************/


                // Generated behaviour from sequence diagrams :

                amount = anUIController.dispAmountInq();
                anUIController.dispWaitMsg();
                withdrawResult = accountController.withdraw(account, amount);

                if (withdrawResult .equals("OK"))
                        $_scenario = 0;
                else
                        $_scenario = 1;

                if ($_scenario == 1) {
                        if (withdrawResult.equals("InsufficientFund"))
                                anUIController.dispInsufFundMsg();
                        if (withdrawResult.equals("WrongPIN"))
                                anUIController.dispWrongPINMsg();
                        if (withdrawResult.equals("CannotDispense"))
                                anUIController.dispCannotDispenseMsg();
                }

                if ($_scenario == 0) {
                        anUIController.dispTakeCashMsg();
                        deviceController.dispenseCash(amount);
                        balance = accountController.balance(account);
                        deviceController.printReceipt(account, "withdrawal", amount, balance);
                }
        }
}
```

รูปที่ 7.11 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” หลังทำการเพิ่มด้วยตนเอง (ต่อ)

โดยขนาดของรหัสคำสั่งเชิงพฤติกรรมก่อนและหลังการทำการเพิ่มด้วยตนเองในโอเปอร์เรชันทั้งสองของคลาสดังกล่าว แสดงได้ดังตารางที่ 7.1 ส่วนโอเปอร์เรชันอื่น ๆ นอกเหนือจากโอเปอร์เรชันทั้งสองนี้ จะไม่มีพฤติกรรมที่แตกต่างกันตามแต่ละสถานการณ์ ผู้ใช้จึงไม่ต้องทำการเพิ่มรหัสคำสั่งส่วนของการเลือกสถานการณ์ด้วยตนเองลงในโอเปอร์เรชันดังกล่าว

ตารางที่ 7.1 ขนาดของรหัสคำสั่งเชิงพฤติกรรมก่อนและหลังการทำการเพิ่มด้วยตนเองของระบบเอทีเอ็มอย่างง่าย[4]

คลาส	โอเปอร์เรชัน	ขนาดของรหัสคำสั่งเชิงพฤติกรรม (สเตทเมนต์)		ร้อยละของรหัสคำสั่งที่เครื่องมือสร้างได้
		ก่อนการเพิ่ม	หลังการเพิ่ม	
SessionController	StartSession	16	22	72.73
SessionController	doWithdrawal	19	23	82.61

7.3 กรณีศึกษาที่ 2 ระบบห้องสมุด (ส่วนบริการให้ยืมหนังสือ)

กรณีศึกษาที่ 2 ที่ขอนำมาเป็นระบบทดสอบคือส่วนบริการให้ยืมหนังสือของระบบห้องสมุด ซึ่งประกอบไปด้วยสถานการณ์ทั้งหมด 3 สถานการณ์ ได้แก่

1. อนุญาตให้ผู้ใช้ทำการยืมหนังสือได้ (สถานการณ์ปกติ)
2. ไม่อนุญาตให้ผู้ใช้ยืมหนังสือได้เลย
3. ไม่อนุญาตให้ผู้ใช้ยืมหนังสือเล่มใดเล่มหนึ่ง

7.3.1 แผนภาพคลาสของระบบ

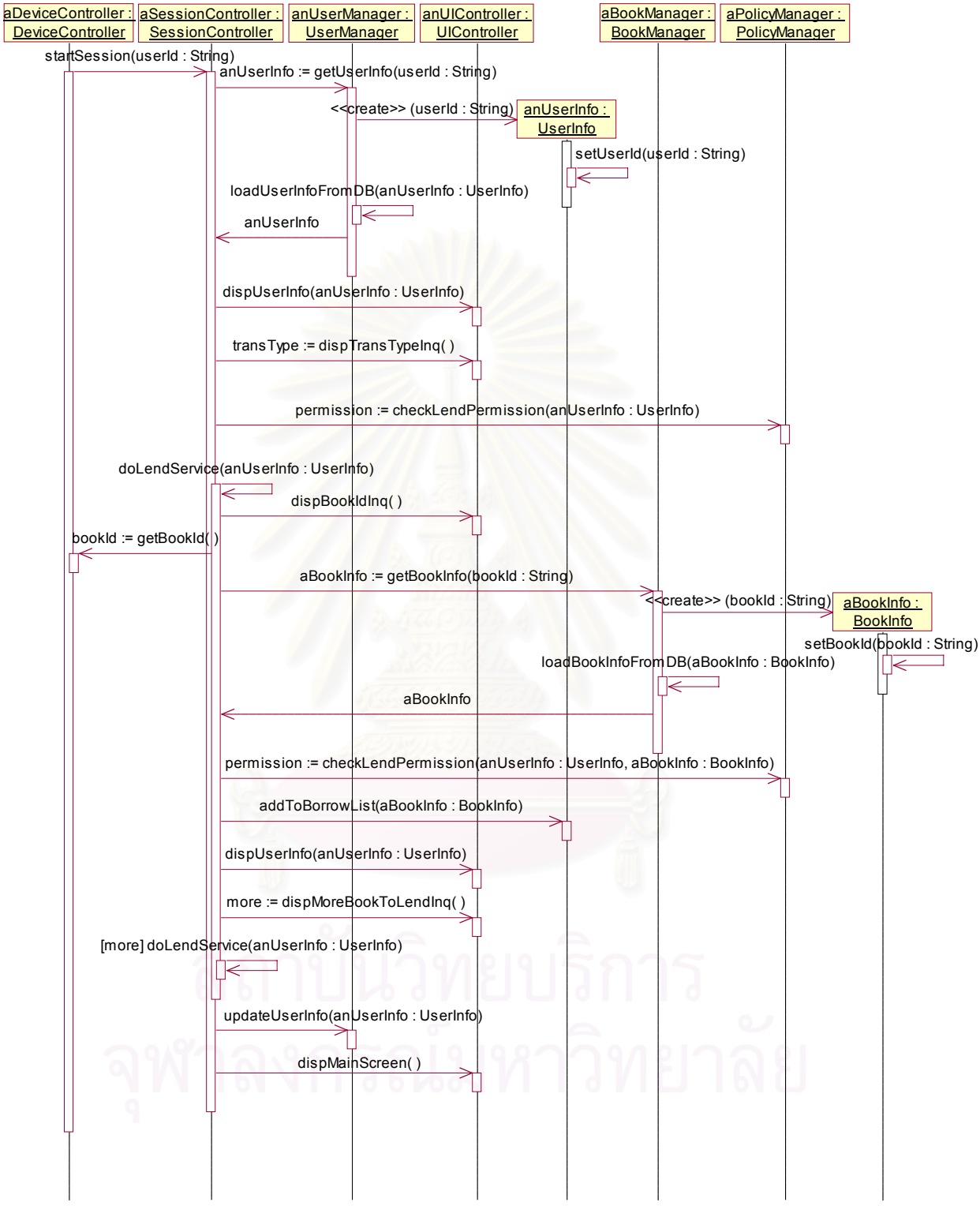
แผนภาพคลาสของระบบห้องสมุด (ส่วนบริการให้ยืมหนังสือ) ซึ่งประกอบไปด้วยคลาส BookInfo BookManager DeviceController PolicyManager SessionController UIController UserInfo และ UserManager สามารถแสดงได้ดังรูปที่ 7.12

7.3.2 แผนภาพซีเควนซ์ของระบบ

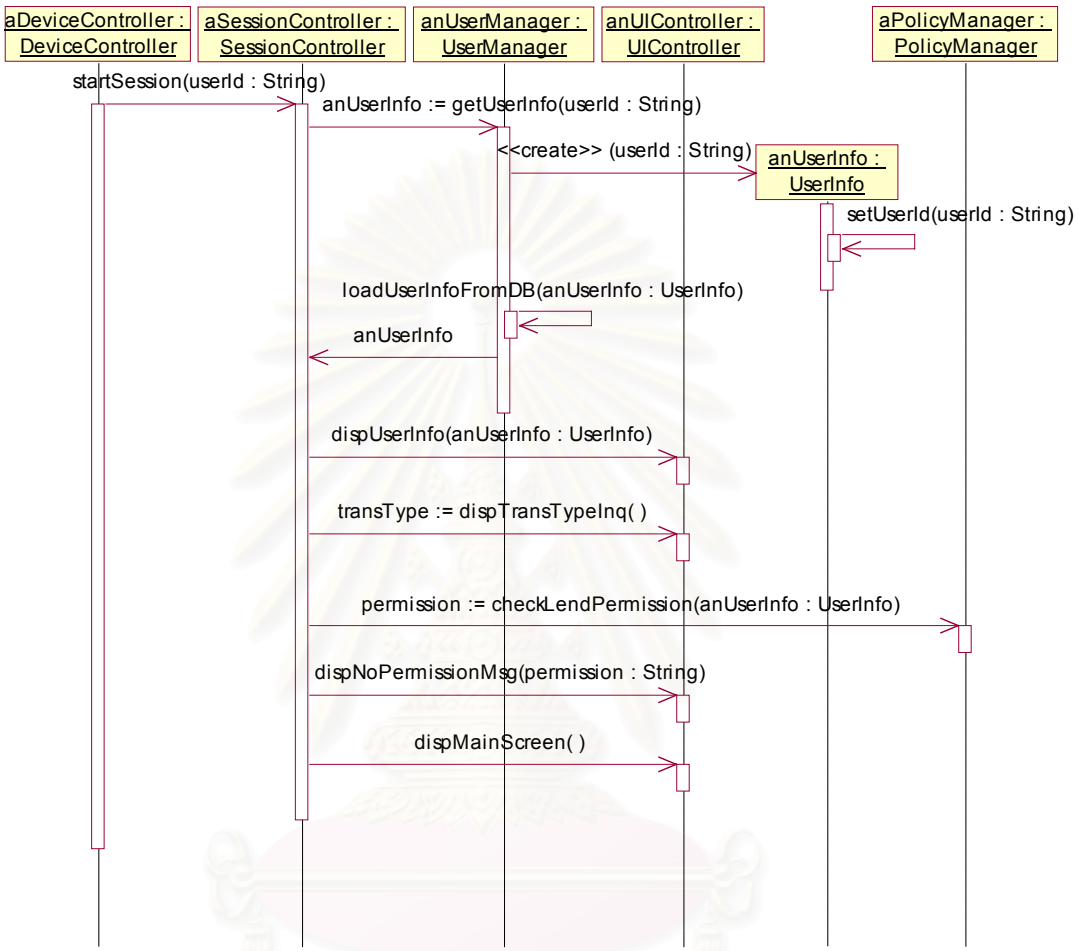
แผนภาพซีเควนซ์สำหรับแต่ละสถานการณ์ ได้แก่ อนุญาตให้ผู้ใช้ทำการยืมหนังสือได้ (สถานการณ์ปกติ) ไม่อนุญาตให้ผู้ใช้ยืมหนังสือได้เลย ไม่อนุญาตให้ผู้ใช้ยืมหนังสือเล่มใดเล่มหนึ่ง แสดงได้ดังรูปที่ 7.13 ถึงรูปที่ 7.15 ตามลำดับ

[4] วิธีการวัดขนาดของรหัสคำสั่งเชิงพฤติกรรม แสดงอยู่ในภาคผนวก ก

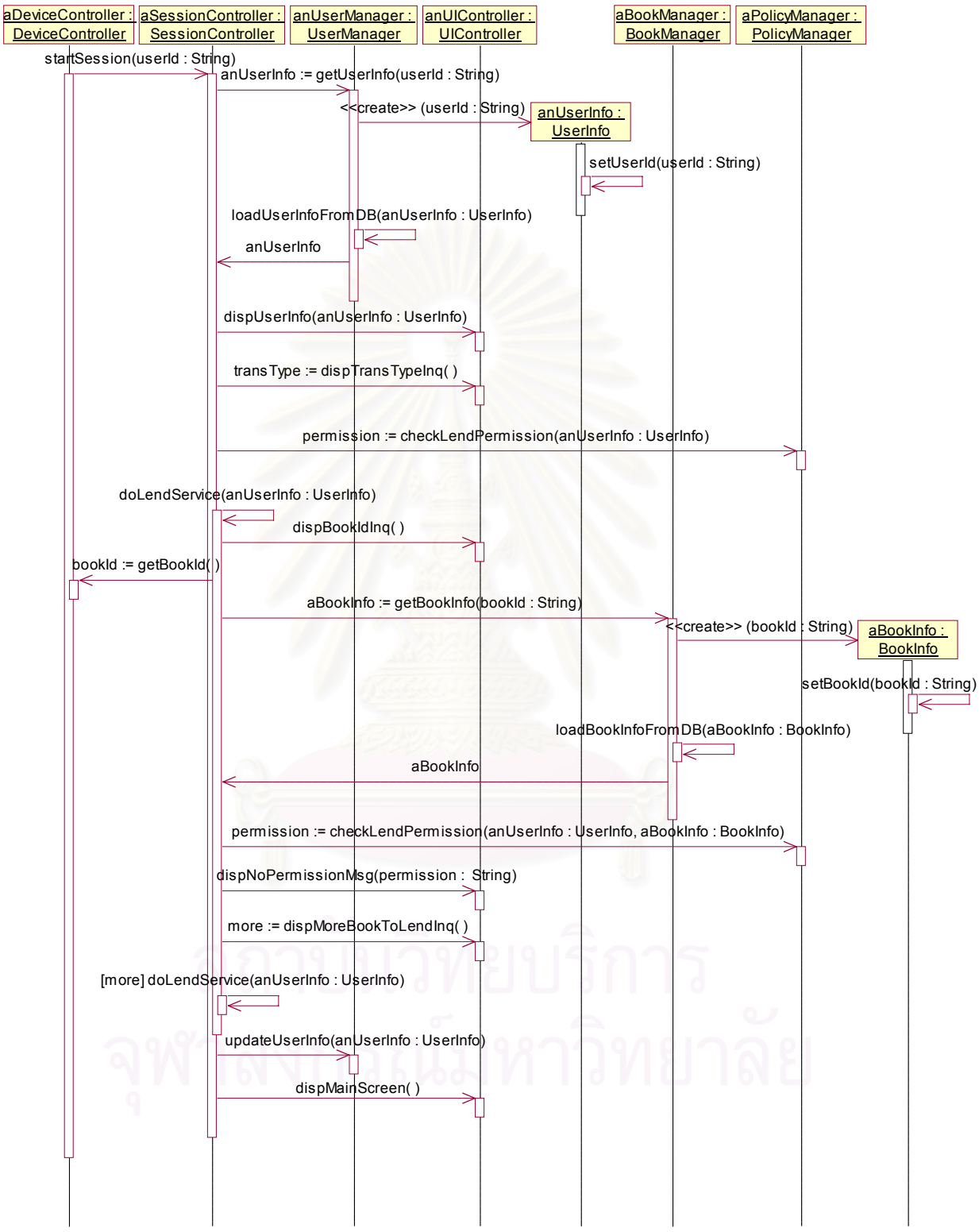
รูปที่ 7.12 แผนภาพคลาสของระบบห้องสมุด (ส่วนบริการให้ยืมหนังสือ)



รูปที่ 7.13 แผนภาพซีเควนซ์แสดงสถานการณ์ที่อนุญาตให้ผู้ใช้ทำการยืมหนังสือได้ (สถานการณ์ปกติ)



รูปที่ 7.14 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ไม่อนุญาตให้ผู้ใช้ยืมหนังสือได้เลย



รูปที่ 7.15 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ไม่อนุญาตให้ผู้ใช้ยืมหนังสือเล่มใดเล่มหนึ่ง

7.3.3 รหัสคำสั่งที่แปลงได้

ผลที่ได้จากการทำการแปลง ฯ ด้วยเครื่องมือ ประกอบด้วยแฟ้มข้อมูล 8 แฟ้ม ได้แก่ BookInfo.java BookManager.java DeviceController.java PolicyManager.java SessionController.java UIController.java UserInfo.java และ UserController.java โดยมีรหัสคำสั่งของแต่ละแฟ้มข้อมูลแสดงดังรูปที่ 7.16 ถึง 7.23

```
public class BookInfo {

        protected String bookId;
        protected String title;
        protected String author;
        protected String type;

        public  BookInfo(String bookId) {

                int $_scenario = 0;

                /*****************************************
                  Scenario numbers :
                        # 0 = {Logical View}Normal flow
                        # 1 = {Logical View}Exceptional flow (don't allow this user to borrow any books)
                 *****************************************/


                // Generated behaviour from sequence diagrams :

                setBookId(bookId);
        }

        public void setBookId(String bookId) {

        }
}
```

รูปที่ 7.16 รหัสคำสั่งของแฟ้มข้อมูล "BookInfo.java"

```
public class BookManager {

        public BookInfo getBookInfo(String bookId) {

                BookInfo aBookInfo;

                int $_scenario = 0;

                /*****************************************
                  Scenario numbers :
                        # 0 = {Logical View}Normal flow
                        # 1 = {Logical View}Exceptional flow (don't allow this user to borrow any books)
                 *****************************************/
```

รูปที่ 7.17 รหัสคำสั่งของแฟ้มข้อมูล "BookManager.java"

```
		// Generated behaviour from sequence diagrams :

			aBookInfo = new BookInfo(bookId);
			loadBookInfoFromDB(aBookInfo);
			return aBookInfo;
		}

		public void loadBookInfoFromDB(BookInfo aBookInfo) {

		}
}
```

รูปที่ 7.17 รหัสคำสั่งของแฟ้มข้อมูล “BookManager.java” (ต่อ)

```
public class DeviceController {

		public String getBookId() {

			return null;

		}
}
```

รูปที่ 7.18 รหัสคำสั่งของแฟ้มข้อมูล “DeviceController.java”

```
public class PolicyManager {

		public String checkLendPermission(UserInfo anUserInfo, BookInfo aBookInfo) {

			return null;

		}

		public String checkLendPermission(UserInfo anUserInfo) {

			return null;

		}
}
```

รูปที่ 7.19 รหัสคำสั่งของแฟ้มข้อมูล “PolicyManager.java”

```
public class SessionController {

		private DeviceController aDeviceController = new DeviceController();
		private UIController anUIController = new UIController();
		private UserManager anUserManager = new UserManager();
		private BookManager aBookManager = new BookManager();
		private PolicyManager aPolicyManager = new PolicyManager();
```

รูปที่ 7.20 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java”

```
	public void startSession(String userId) {

		UserInfo anUserInfo;
		String transType;
		String permission;

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 1 = {Logical View}Exceptional flow (don't allow this user to borrow this
book)
			# 2 = {Logical View}Exceptional flow (don't allow this user to borrow any
books)
		 *****************************************/


		// Generated behaviour from sequence diagrams :

		anUserInfo = anUserManager.getUserInfo(userId);
		anUIController.dispUserInfo(anUserInfo);
		transType = anUIController.dispTransTypeInq();
		permission = aPolicyManager.checkLendPermission(anUserInfo);

		if ($_scenario == 1) {
			anUIController.dispNoPermissionMsg(permission);
		}

		if ($_scenario == 0 || $_scenario == 2) {
			doLendService(anUserInfo);
			anUserManager.updateUserInfo(anUserInfo);
		}
		anUIController.dispMainScreen();
	}

	public void doLendService(UserInfo anUserInfo) {

		String bookId;
		BookInfo aBookInfo;
		String permission;
		boolean more;

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 2 = {Logical View}Exceptional flow (don't allow this user to borrow any
books)
		 *****************************************/


		// Generated behaviour from sequence diagrams :

		anUIController.dispBookIdInq();
		bookId = aDeviceController.getBookId();
		aBookInfo = aBookManager.getBookInfo(bookId);
		permission = aPolicyManager.checkLendPermission(anUserInfo, aBookInfo);
```

รูปที่ 7.20 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” (ต่อ)

```
                if ($_scenario == 2) {
                        anUIController.dispNoPermissionMsg(permission);
                }

                if ($_scenario == 0) {
                        anUserInfo.addToBorrowList(aBookInfo);
                        anUIController.dispUserInfo(anUserInfo);
                }
                more = anUIController.dispMoreBookToLendInq();
                if (more) doLendService(anUserInfo);
        }

}
```

รูปที่ 7.20 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” (ต่อ)

```
public class UIController {

        public void dispUserInfo(UserInfo anUserInfo) {

        }

        public String dispTransTypeInq() {

                return null;

        }

        public String dispBookIdInq() {

                return null;

        }

        public boolean dispMoreBookToLendInq() {

                return false;

        }

        public void dispMainScreen() {

        }

        public void dispNoPermissionMsg(String message) {

        }
}
```

รูปที่ 7.21 รหัสคำสั่งของแฟ้มข้อมูล “UIController.java”

```
public class UserInfo {

	protected String userId;
	protected String name;
	protected boolean valid;
	protected BookInfo borrowList;

	public  UserInfo(String userId) {

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 1 = {Logical View}Exceptional flow (don't allow this user to borrow this
book)
			# 2 = {Logical View}Exceptional flow (don't allow this user to borrow any
books)
		 *****************************************/


		// Generated behaviour from sequence diagrams :

		setUserId(userId);
	}

	public void addToBorrowList(BookInfo aBookInfo) {

	}

	public boolean isValid() {

		return false;
	}

	public void setUserId(String userId) {

	}
}
```

รูปที่ 7.22 รหัสคำสั่งของแฟ้มข้อมูล “UserInfo.java”

```
public class UserManager {

	public UserInfo getUserInfo(String userId) {

		UserInfo anUserInfo;

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 1 = {Logical View}Exceptional flow (don't allow this user to borrow this
book)
			# 2 = {Logical View}Exceptional flow (don't allow this user to borrow any
books)
		 *****************************************/
```

รูปที่ 7.23 รหัสคำสั่งของแฟ้มข้อมูล “UserManager.java”

```
                // Generated behaviour from sequence diagrams :

                anUserInfo = new UserInfo(userId);
                loadUserInfoFromDB(anUserInfo);
                return anUserInfo;
        }

        public void updateUserInfo(UserInfo anUserInfo) {

        }

        public void loadUserInfoFromDB(UserInfo anUserInfo) {

        }
}
```

รูปที่ 7.23 รหัสคำสั่งของแฟ้มข้อมูล “UserManager.java” (ต่อ)

7.3.4 ผลการทดสอบ

เครื่องมือสามารถทำการแปลงแฟ้มข้อมูลของกรณีศึกษาที่ 2 และให้ผลการแปลงเป็นแฟ้มข้อมูลรหัสคำสั่งภาษาจาวาได้ดังที่แสดงไว้หัวข้อที่ 7.3.3 ซึ่งเมื่อเปรียบเทียบรหัสคำสั่งที่แปลงได้จากเครื่องมือกับรหัสคำสั่งที่ทำการแปลงด้วยตนเองแล้วพบว่ามีความสอดคล้องกัน และเมื่อนำไปคอมไพล์ด้วยคอมไพเลอร์ที่ทำตามข้อกำหนดของภาษาจาวารุ่นที่ 2 พบว่าสามารถทำการคอมไพล์ได้สำเร็จ

7.3.5 วิเคราะห์ผลการทดสอบ

จากผลการทดสอบ พบว่าเครื่องมือสามารถทำการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ของกรณีศึกษาที่ 2 ได้อย่างถูกต้อง อย่างไรก็ตามก่อนที่จะนำรหัสคำสั่งนี้ไปใช้จริง ผู้ใช้จะต้องเพิ่มรหัสคำสั่งส่วนของการเลือกสถานการณ์ด้วยตนเองลงในคลาส SessionController อยู่สองโอเปอร์เรชันดังแสดงในรูปที่ 7.24 โดยรหัสคำสั่งส่วนที่ผู้ใช้ต้องทำเพิ่มได้ถูกเน้นด้วยตัวอักษรแบบหนา

```
public class SessionController {

        private DeviceController aDeviceController = new DeviceController();
        private UIController anUIController = new UIController();
        private UserManager anUserManager = new UserManager();
        private BookManager aBookManager = new BookManager();
        private PolicyManager aPolicyManager = new PolicyManager();
```

รูปที่ 7.24 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” หลังทำการเพิ่มด้วยตนเอง

```
	public void startSession(String userId) {

		UserInfo anUserInfo;
		String transType;
		String permission;

		int $_scenario = 0;

		/****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 1 = {Logical View}Exceptional flow (don't allow this user to borrow this
book)
			# 2 = {Logical View}Exceptional flow (don't allow this user to borrow any
books)
		 ****************************************/


		// Generated behaviour from sequence diagrams :

		anUserInfo = anUserManager.getUserInfo(userId);
		anUIController.dispUserInfo(anUserInfo);
		transType = anUIController.dispTransTypeInq();
		permission = aPolicyManager.checkLendPermission(anUserInfo);

		if (permission.equals("ok"))
			$_scenario = 0;
		else
			$_scenario = 1;

		if ($_scenario == 1) {
			anUIController.dispNoPermissionMsg(permission);
		}

		if ($_scenario == 0 || $_scenario == 2) {
			doLendService(anUserInfo);
			anUserManager.updateUserInfo(anUserInfo);
		}
		anUIController.dispMainScreen();
	}

	public void doLendService(UserInfo anUserInfo) {

		String bookId;
		BookInfo aBookInfo;
		String permission;
		boolean more;

		int $_scenario = 0;

		/****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 2 = {Logical View}Exceptional flow (don't allow this user to borrow any
books)
		 ****************************************/
```

รูปที่ 7.24 รหัสคำสั่งของแฟ้มข้อมูล "SessionController.java" หลังทำการเพิ่มด้วยตนเอง (ต่อ)

```
                // Generated behaviour from sequence diagrams :

                anUIController.dispBookIdInq();
                bookId = aDeviceController.getBookId();
                aBookInfo = aBookManager.getBookInfo(bookId);
                permission = aPolicyManager.checkLendPermission(anUserInfo, aBookInfo);

                if (permission.equals("ok"))
                        $_scenario = 0;
                else
                        $_scenario = 2;

                if ($_scenario == 2) {
                        anUIController.dispNoPermissionMsg(permission);
                }

                if ($_scenario == 0) {
                        anUserInfo.addToBorrowList(aBookInfo);
                        anUIController.dispUserInfo(anUserInfo);
                }
                more = anUIController.dispMoreBookToLendInq();
                if (more) doLendService(anUserInfo);
        }
}
```

รูปที่ 7.24 รหัสคำสั่งของแฟ้มข้อมูล “SessionController.java” หลังทำการเพิ่มด้วยตนเอง (ต่อ)

โดยขนาดของรหัสคำสั่งเชิงพฤติกรรมก่อนและหลังการทำการเพิ่มด้วยตนเองในโอเปอร์เรชันทั้งสองของคลาสดังกล่าว แสดงได้ดังตารางที่ 7.2 ส่วนโอเปอร์เรชันอื่น ๆ นอกเหนือจากโอเปอร์เรชันทั้งสองนี้ จะไม่มีพฤติกรรมที่แตกต่างกันตามแต่ละสถานการณ์ ผู้ใช้จึงไม่ต้องทำการเพิ่มรหัสคำสั่งส่วนของการเลือกสถานการณ์ด้วยตนเองลงในโอเปอร์เรชันดังกล่าว

ตารางที่ 7.2 ขนาดของรหัสคำสั่งเชิงพฤติกรรมก่อนและหลังการทำการเพิ่มด้วยตนเองของระบบห้องสมุด (ส่วนบริการให้ยืมหนังสือ)

คลาส	**โอเปอร์เรชัน**	**ขนาดของรหัสคำสั่งเชิงพฤติกรรม (สเตทเมนต์)**		**ร้อยละของรหัสคำสั่งที่เครื่องมือสร้างได้**
		ก่อนการเพิ่ม	**หลังการเพิ่ม**	
SessionController	StartSession	15	19	78.95
SessionController	doLendService	17	21	80.95

7.4 สรุปผลการทดสอบ

จากผลการทดสอบด้วยกรณีศึกษาทั้ง 2 กรณี สามารถแสดงได้ว่า ขั้นตอนและกฎในการทำการแปลงกลุ่มของแผนภาพซีเควนซ์ไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา รวมถึงเครื่องมือสนับสนุนการทำการแปลงดังกล่าว ซึ่งทั้งหมดเป็นผลที่ได้จากงานวิจัยนี้สามารถที่จะใช้ทำการแปลง ฯ ได้อย่างถูกต้องและครบถ้วนตามที่ได้ระบุไว้ในขอบเขตของการวิจัย และให้ผลเป็นรหัสคำสั่งภาษาจาวาที่เกือบจะสมบูรณ์เมื่อเทียบกับข้อมูลในแผนภาพซีเควนซ์ โดยจะขาดก็แต่เพียงรหัสคำสั่งส่วนของการเลือกสถานการณ์ที่ผู้ใช้จะต้องทำการเพิ่มด้วยตนเองอีกเพียงเล็กน้อยในบางโอเปอร์เรชันเท่านั้น ซึ่งเมื่อคิดเป็นร้อยละของขนาดของรหัสคำสั่งเชิงพฤติกรรมที่เครื่องมือสร้างได้จากกรณีศึกษาทั้งสองแล้ว จะคิดได้เป็นร้อยละ 100 สำหรับโอเปอร์เรชันที่ไม่มีพฤติกรรมที่แตกต่างกันตามแต่ละสถานการณ์ และคิดได้เป็นร้อยละตั้งแต่ 72.73 ถึง 82.61 สำหรับโอเปอร์เรชันที่มีพฤติกรรมแตกต่างกันตามแต่ละสถานการณ์

บทที่ 8
สรุปผลการวิจัย

8.1 สรุปผลการวิจัย

ในงานวิจัยนี้ ได้มีการออกแบบขั้นตอนและกฎในการทำการแปลงกลุ่มของแผนภาพซีเควนซ์ไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวา โดยสามารถขจัดข้อจำกัดของงานวิจัยที่มีอยู่เดิม นั่นคือขั้นตอนและกฎดังกล่าวสามารถทำการแปลงแผนภาพซีเควนซ์ที่มีการส่งเมสเสจซ้อนกันหลายระดับและทำการรวมพฤติกรรมของแต่ละโอเปอร์เรชันจากหลายสถานการณ์ได้ ซึ่งทำให้สามารถนำผลของงานวิจัยนี้ไปประยุกต์ใช้งานจริงได้โดยสะดวก

นอกจากนั้น ยังได้มีการออกแบบและพัฒนาเครื่องมือสนับสนุนการแปลงดังกล่าวโดยอัตโนมัติ โดยผู้วิจัยเลือกใช้ภาษาเอ็กซ์เอสแอลทีในการอิมพลีเมนต์ขั้นตอนและกฎการแปลง ฯ เนื่องจากภาษาดังกล่าวได้ถูกออกแบบมาให้มีความเหมาะสมสำหรับการแปลงเอกสารเอ็กซ์เอ็มแอลโดยเฉพาะ อีกทั้งยังมีความยืดหยุ่นในการปรับเปลี่ยนให้ทำงานบนแพลตฟอร์มการพัฒนาต่าง ๆ สูง จากนั้นจึงได้ทำการทดสอบเครื่องมือดังกล่าวกับกรณีศึกษาสองกรณีด้วยการเปรียบเทียบรหัสคำสั่งที่ได้จากเครื่องมือกับรหัสคำสั่งที่ได้จากการทำการแปลงด้วยตนเอง พบว่าเครื่องมือสามารถให้ผลการแปลง ฯ ที่ถูกต้อง ซึ่งเมื่อคิดเป็นร้อยละของขนาดของรหัสคำสั่งเชิงพฤติกรรมที่เครื่องมือสร้างได้จากกรณีศึกษาทั้งสองแล้ว จะคิดได้เป็นร้อยละ 100 สำหรับโอเปอร์เรชันที่ไม่มีพฤติกรรมที่แตกต่างกันตามแต่ละสถานการณ์ และคิดได้เป็นร้อยละตั้งแต่ 72.73 ถึง 82.61 สำหรับโอเปอร์เรชันที่มีพฤติกรรมแตกต่างกันตามแต่ละสถานการณ์ซึ่งผู้ใช้จะต้องทำการเพิ่มเติมรหัสคำสั่งส่วนของการเลือกสถานการณ์ด้วยตนเอง

ผลลัพธ์ที่ได้จากงานวิจัยนี้ ทำให้นักพัฒนาซอฟต์แวร์สามารถทำการแปลงข้อมูลในระยะของการออกแบบซอฟต์แวร์ไปเป็นข้อมูลในระยะของการอิมพลีเมนต์ได้โดยอัตโนมัติ ซึ่งสามารถที่จะช่วยลดเวลา ลดค่าใช้จ่าย และเพิ่มความถูกต้องให้กับกระบวนการพัฒนาซอฟต์แวร์ได้เป็นอย่างดี โดยเมื่อพิจารณาถึงความนิยมในการใช้แผนภาพซีเควนซ์ที่มีอยู่สูงด้วยแล้ว ก็ยิ่งทำให้งานวิจัยนี้สามารถที่จะก่อให้เกิดประโยชน์แก่อุตสาหกรรมซอฟต์แวร์ได้อย่างกว้างขวางและเป็นรูปธรรมได้มากยิ่งขึ้น

8.2 ปัญหาและอุปสรรค

ถึงแม้ว่าจะมีการกำหนดมาตรฐานเอ็กซ์เอ็มไอมาเพื่ออำนวยความสะดวกในการแลกเปลี่ยนข้อมูลโมเดลยูเอ็มแอลระหว่างเครื่องมือต่าง ๆ แต่เครื่องมือสร้างแผนภาพยูเอ็มแอลแต่ละเครื่องมือก็ไม่ได้ทำตามมาตรฐานดังกล่าวอย่างเคร่งครัด ทำให้ต้องมีการพัฒนาส่วนสำหรับอ่านข้อมูลจากเครื่องมือสร้างแผนภาพยูเอ็มแอลแต่ละเครื่องมือโดยเฉพาะ ซึ่งทำให้เครื่องมือที่พัฒนาขึ้นมานี้ยังมีข้อจำกัดให้ใช้ได้กับเครื่องมือสร้างแผนภาพยูเอ็มแอลที่กำหนดเท่านั้น

8.3 แนวทางในการประยุกต์ใช้ร่วมกับงานวิจัยอื่น ๆ

ผลที่ได้จากงานวิจัยนี้สามารถที่จะนำไปประยุกต์ใช้ร่วมกับงานวิจัยที่เกี่ยวกับการแปลงแผนภาพยูเอ็มแอลชนิดอื่น ๆ ซึ่งจะทำให้ได้รหัสคำสั่งที่สมบูรณ์ยิ่งขึ้น เช่นประยุกต์ใช้ร่วมกับงานวิจัยที่เกี่ยวข้องกับการแปลงแผนภาพคลาส ที่มีการศึกษาการแปลงความสัมพันธ์ต่าง ๆ ระหว่างคลาส และคุณสมบัติอื่น ๆ ของแผนภาพคลาสไปเป็นรหัสคำสั่ง ก็จะทำให้ได้รหัสคำสั่งเชิงโครงสร้างที่มีความสมบูรณ์มากยิ่งขึ้น หรือเมื่อประยุกต์ใช้ร่วมกับงานวิจัยที่เกี่ยวข้องกับการแปลงแผนภาพสเตทชาร์ต (Statechart diagram) ที่มีการศึกษาการแปลงพฤติกรรมในระดับวัตถุที่ระบุอยู่ในแผนภาพสเตทชาร์ตไปเป็นรหัสคำสั่ง ก็จะทำให้ได้รหัสคำสั่งเชิงพฤติกรรมทั้งในระดับวัตถุและระดับโอเปอร์เรชัน

8.4 แนวทางในการวิจัยต่อไป

ประเด็นที่งานวิจัยนี้ยังไม่ได้ศึกษา และสามารถทำการวิจัยเพิ่มเติมได้ในอนาคต ได้แก่

1. การส่งเมสเสจแบบอะซิงโครนัส
2. การมีการส่งเมสเสจที่เกิดขึ้นพร้อมกัน
3. การจัดการกับความผิดพลาด (Exception handling)

นอกจากนี้ ข้อกำหนดของยูเอ็มแอลของรุ่นถัดไปอาจมีคุณสมบัติที่แตกต่างกับรุ่นที่งานวิจัยนี้ศึกษา (รุ่น 1.3) ซึ่งมีโอกาสที่จะก่อให้เกิดประเด็นใหม่ ๆ ที่สามารถทำการวิจัยได้ต่อไป

รายการอ้างอิง

[1] Mathupayas Thongmak, and Pornsiri Muenchaisri. "Design of Rules for Transforming UML Sequence Diagrams into Java Code", Proceedings of the Ninth Asia-Pacific Software Engineering Conference (APSEC'02), 2002.

[2] Extensible Markup Language (XML) [Online]. Available from : http://www.w3.org/XML, [2003, November 20].

[3] XML Metadata Interchange (XMI) [Online]. Available from : http://www.omg.org/technology/xml, [2003, November 20].

[4] XSL Transformations (XSLT) [Online]. Available from : http://www.w3.org/TR/xslt, [2003, November 20].

[5] Unified Modeling Language (UML) [Online]. Available from : http://www.omg.org/technology/uml, [2003, November 20].

[6] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. "The Java Language Specification, Second Edition", Massachusetts: Addison-Wesley, 2000.

[7] James Rumbaugh, Ivar Jacobson, and Grady Booch. "The Unified Modeling Language Reference Manual", Massachusetts: Addison-Wesley, 1999.

[8] Grady Booch, James Rumbaugh, and Ivar Jacobson. "The Unified Modeling Language User Guide", Massachusetts: Addison-Wesley, 1999.

[9] Michael Kay. "XSLT Programmer's Reference 2nd Edition", Birmingham: Wrox Press, 2001.

[10] James Clark. "XSLT 1.0 recommendation"[Online]. World Wide Web Consortium (W3C). Available from : http://www.w3.org/TR/1999/REC-xslt-19991116, [2003, November 20].

[11] Rational Rose [Computer software]. Available from : http://www.rational.com, [2003, November 20].

[12] Unisys Rose XML Tools [Computer software]. Available from : http://www.rational.com, [2003, November 20].

[13] Saxon [Computer software]. Available from : http://saxon.sourceforge.net, [2003, November 20].

[14] Neeraj Sangal, Edward Farrell, Karl Lieberherr, and David Lorenz. “Interaction Schemata: Compiling Interactions to Code”, Proceeding of Technology of Object-Oriented Language and Systems (TOOLS USA’ 99), pp. 268-277, 1999.

[15] Object Management Group. “OMG Unified Modeling Language Specification Version 1.5”[Online]. Available from : http://www.omg.org/uml, [2003, November 20].

[16] Alan Dennis, Barbara Haley Wixom, and David Tegarden. “System Analysis and Design, An Object-Oriented Approach with UML”, New York: John Wiley & Sons, 2002.

[17] Object Management Group. “OMG XML Metadata Interchange (XMI) Specification Version 1.2”[Online]. Available from : http://www.omg.org/uml, [2003, July 10].

[18] Microsoft Corporation. “The MSXML 4.0 Software Development Kit”[Computer software]. Available from : http://www.microsoft.com, [2003, November 20].

[19] Watts S. Humphrey. “A Discipline for Software Engineering”, Massachusetts: Addison-Wesley, 1995.

[20] Miloslav Nic. “XSLT Tutorial”[Online]. Available from : http://www.zvon.org, [2003, November 20].

[21] Miloslav Nic. “XSLT Reference”[Online]. Available from : http://www.zvon.org, [2003, November 20].

[22] Martin Fowler, and Kendall Scott. “UML Distilled”, Massachusetts: Addison-Wesley, 1997.

ภาคผนวก

ภาคผนวก ก
การวัดขนาดของรหัสคำสั่งเชิงพฤติกรรม

ในงานวิจัยนี้ ได้มีการใช้ขนาดของรหัสคำสั่งเชิงพฤติกรรมในการเปรียบเทียบรหัสคำสั่งที่สร้างขึ้นได้จากเครื่องมือว่ามีความสมบูรณ์มากน้อยเพียงใด โดยรหัสคำสั่งเชิงพฤติกรรมที่จะนำมาทำการวัดขนาดคือรหัสคำสั่งส่วนที่อยู่ภายในของแต่ละโอเปอร์เรชัน ซึ่งในภาคผนวกนี้ จะได้อธิบายถึงวิธีการวัดขนาดของรหัสคำสั่งเชิงพฤติกรรมดังกล่าว

ผู้วิจัยได้ประยุกต์วิธีการวัดขนาดของรหัสคำสั่งมาจากตัวอย่างมาตรฐานการนับรหัสคำสั่งของภาษาซีพลัสพลัส (C++) ในหนังสือ A Discipline for Software Engineering [19] หน้าที่ 74 ถึง 81 โดยกำหนดให้มาตรวัดขนาดของรหัสคำสั่งเชิงพฤติกรรม คือจำนวนของสเตทเมนต์เชิงตรรกะ (Logical statement) ของรหัสคำสั่งนั้น โดยกำหนดกฎในการนับจำนวนของสเตทเมนต์เชิงตรรกะดังกล่าวดังนี้

1. นับหนึ่งสำหรับทุก ๆ การปรากฏของคำสำคัญต่อไปนี้
   case, catch, do, else, else if, for, if, switch, synchronized, try, while
2. นับหนึ่งสำหรับทุก ๆ การปรากฏของสัญลักษณ์ต่อไปนี้
   ; || &&

ตัวอย่างการนับจำนวนสเตทเมนต์เชิงตรรกะ แสดงได้ดังรูปที่ ก.1 โดยจุดที่ทำการนับแสดงด้วยสัญลักษณ์สี่เหลี่ยม ซึ่งจากรูปดังกล่าว จะนับจำนวนของสเตทเมนต์เชิงตรรกะได้ 10 สเตทเมนต์

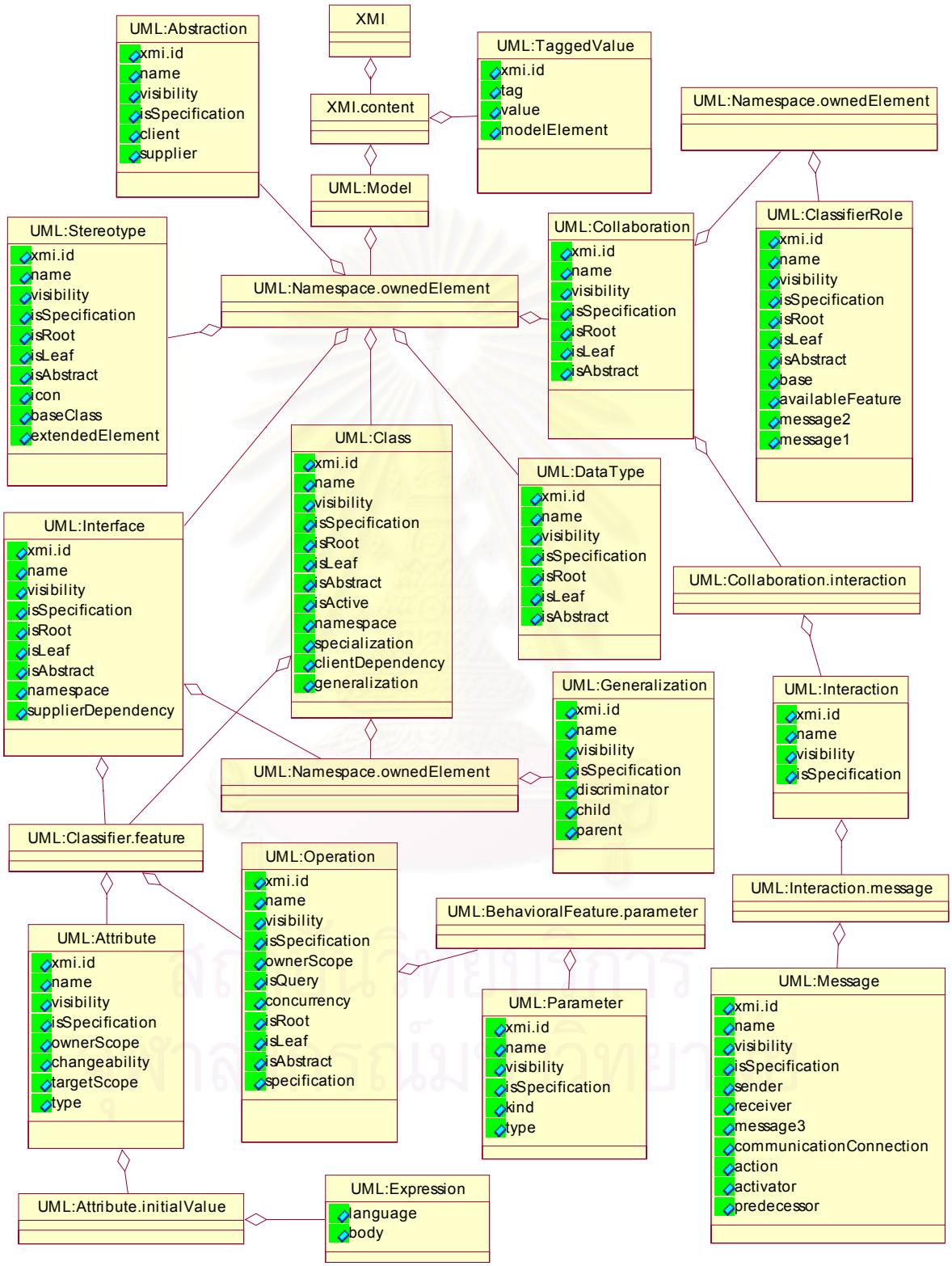
```
if (permission.equals("ok"))
        $_scenario = 0;
else if (count == 0)
        $_scenario = 2;
else
        $_scenario = 1;

if ($_scenario == 0 || $_scenario == 2) {
        doLendService(anUserInfo);
        anUserManager.updateUserInfo(anUserInfo);
}
```

รูปที่ ก.1 ตัวอย่างการนับจำนวนสเตทเมนต์เชิงตรรกะ

ภาคผนวก ข
แฟ้มข้อมูลเอ็กซ์เอ็มไอที่นำมาทำการแปลง ฯ

ในภาคผนวกนี้จะเป็นการแสดงโครงสร้างและตัวอย่างของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่เครื่องมือที่พัฒนาขึ้นมาในงานวิจัยนี้นำมาทำการแปลง ฯ โดยแฟ้มข้อมูลดังกล่าวเป็นแฟ้มข้อมูลเอ็กซ์เอ็มไอรุ่น 1.1 ที่ถูกส่งออกจากเรชันแนล โรส รุ่น 2002 ด้วยเครื่องมือเสริม ยูนิซิส โรส เอ็กซ์เอ็มแอล ทูล รุ่น 1.3.6 และเนื่องจากโครงสร้างเอ็กซ์เอ็มไอมีความซับซ้อนและมีส่วนย่อยอยู่มาก ในที่นี้จึงขอเลือกแสดงเฉพาะส่วนที่เครื่องมือดังกล่าวใช้ในการแปลง ฯ เท่านั้น

รูปที่ ข.1 เป็นการแสดงโครงสร้างของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่นำมาทำการแปลง ฯ ส่วนรูปที่ ข.2 เป็นการแสดงส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพคลาสตามรูปที่ 3.2 และรูปที่ ข.3 เป็นการแสดงส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพซีเควนซ์ตามรูปที่ 3.3 และ 3.4 โดยรูปทั้ง 3 นี้ได้มีการตัดส่วนที่ไม่ได้ใช้ในการทำการแปลง ฯ ออกไปเรียบร้อยแล้ว



รูปที่ ข.1 โครงสร้างของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่นำมาทำการแปลง ฯ

(เฉพาะส่วนที่ใช้ในการแปลง ฯ)

```
<UML:Class xmi.id="S.092.0438.07.1" name="Order" visibility="public" isSpecification="false" isRoot="true"
isLeaf="true" isAbstract="false" isActive="false" namespace="G.0">
    <UML:Classifier.feature>
        <!-- ==================== OrderSystem::Order.orderLine[]    [Attribute] ==================== -->
        <UML:Attribute xmi.id="S.092.0438.07.2" name="orderLine[]" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="S.092.0438.07.5">

            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::Order::prepareToDeliver    [Operation]
==================== -->
        <UML:Operation xmi.id="S.092.0438.07.3" name="prepareToDeliver" visibility="public"
isSpecification="false" ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false"
isLeaf="false" isAbstract="false" specification="">
            <UML:BehavioralFeature.parameter>
                <UML:Parameter xmi.id="XX.2.438.9.3" name="deliveryDate" visibility="public"
isSpecification="false" kind="inout" type="G.59">
                </UML:Parameter>
            </UML:BehavioralFeature.parameter>
        </UML:Operation>
        <!-- ==================== OrderSystem::Order::deliver    [Operation] ==================== -->
        <UML:Operation xmi.id="S.092.0438.07.4" name="deliver" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
specification=""/>
    </UML:Classifier.feature>
</UML:Class>
<!-- ==================== Date    [DataType] ==================== -->
<UML:DataType xmi.id="G.59" name="Date" visibility="public" isSpecification="false" isRoot="false" isLeaf="false"
isAbstract="false"/>
<!-- ==================== OrderSystem::OrderLine    [Class] ==================== -->
<UML:Class xmi.id="S.092.0438.07.5" name="OrderLine" visibility="public" isSpecification="false" isRoot="true"
isLeaf="true" isAbstract="false" isActive="false" namespace="G.0">
    <UML:Classifier.feature>
        <!-- ==================== OrderSystem::OrderLine.aStockItem    [Attribute] ==================== -
->
        <UML:Attribute xmi.id="S.092.0438.07.6" name="aStockItem" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="S.092.0438.07.11">

            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::OrderLine.amount    [Attribute] ==================== -->
        <UML:Attribute xmi.id="S.092.0438.07.7" name="amount" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="G.60">
            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::OrderLine.status    [Attribute] ==================== -->
        <UML:Attribute xmi.id="S.092.0438.07.8" name="status" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="G.61">
            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::OrderLine::prepare    [Operation] ==================== --
>
        <UML:Operation xmi.id="S.092.0438.07.9" name="prepare" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
specification="">
```

รูปที่ ข.2 ส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพคลาสตามรูปที่ 3.2

```
                <UML:BehavioralFeature.parameter>
                    <UML:Parameter xmi.id="XX.2.438.9.5" name="sendDate" visibility="public" isSpecification="false"
kind="inout" type="G.59">
                    </UML:Parameter>
                    <UML:Parameter xmi.id="XX.2.438.9.6" name="prepare.Return" visibility="public"
isSpecification="false" kind="return" type="G.62">
        </UML:Parameter>
                </UML:BehavioralFeature.parameter>
            </UML:Operation>
            <!-- ==================== OrderSystem::OrderLine::changeStatus    [Operation]
==================== -->
            <UML:Operation xmi.id="S.092.0438.07.10" name="changeStatus" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
specification="">
                <UML:BehavioralFeature.parameter>
                    <UML:Parameter xmi.id="XX.2.438.9.7" name="status" visibility="public" isSpecification="false"
kind="inout" type="G.61">
                    </UML:Parameter>
                    <UML:Parameter xmi.id="XX.2.438.9.8" name="changeStatus.Return" visibility="public"
isSpecification="false" kind="return" type="G.62">
        </UML:Parameter>
                </UML:BehavioralFeature.parameter>
            </UML:Operation>
        </UML:Classifier.feature>
</UML:Class>
<!-- ==================== int    [DataType] ==================== -->
<UML:DataType xmi.id="G.60" name="int" visibility="public" isSpecification="false" isRoot="false" isLeaf="false"
isAbstract="false"/>
<!-- ==================== String    [DataType] ==================== -->
<UML:DataType xmi.id="G.61" name="String" visibility="public" isSpecification="false" isRoot="false" isLeaf="false"
isAbstract="false"/>
<!-- ==================== void    [DataType] ==================== -->
<UML:DataType xmi.id="G.62" name="void" visibility="public" isSpecification="false" isRoot="false" isLeaf="false"
isAbstract="false"/>
<!-- ==================== OrderSystem::StockItem    [Class] ==================== -->
<UML:Class xmi.id="S.092.0438.07.11" name="StockItem" visibility="public" isSpecification="false" isRoot="true"
isLeaf="true" isAbstract="false" isActive="false" namespace="G.0">
    <UML:Classifier.feature>
        <!-- ==================== OrderSystem::StockItem.name    [Attribute] ==================== -->
        <UML:Attribute xmi.id="S.092.0438.07.12" name="name" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="G.61">
            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::StockItem.currentAmount    [Attribute]
==================== -->
        <UML:Attribute xmi.id="S.092.0438.07.13" name="currentAmount" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="G.60">
            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::StockItem.reservedAmount    [Attribute]
==================== -->
        <UML:Attribute xmi.id="S.092.0438.07.14" name="reservedAmount" visibility="private"
isSpecification="false" ownerScope="instance" changeability="changeable" targetScope="instance" type="G.60">
            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::StockItem::check    [Operation] ==================== -->
        <UML:Operation xmi.id="S.092.0438.07.15" name="check" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
specification="">
```

รูปที่ ข.2 ส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพคลาสตามรูปที่ 3.2 (ต่อ)

```
            <UML:BehavioralFeature.parameter>
                <UML:Parameter xmi.id="XX.2.438.9.10" name="amount" visibility="public" isSpecification="false"
kind="inout" type="G.60">
                </UML:Parameter>
                <UML:Parameter xmi.id="XX.2.438.9.11" name="sendDate" visibility="public" isSpecification="false"
kind="inout" type="G.59">
                </UML:Parameter>
                <UML:Parameter xmi.id="XX.2.438.9.12" name="check.Return" visibility="public"
isSpecification="false" kind="return" type="G.63">
      </UML:Parameter>
            </UML:BehavioralFeature.parameter>
        </UML:Operation>
        <!-- ==================== OrderSystem::StockItem::remove    [Operation] ==================== -->
        <UML:Operation xmi.id="S.092.0438.07.16" name="remove" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
specification="">
            <UML:BehavioralFeature.parameter>
                <UML:Parameter xmi.id="XX.2.438.9.13" name="amount" visibility="public" isSpecification="false"
kind="inout" type="G.60">
                </UML:Parameter>
                <UML:Parameter xmi.id="XX.2.438.9.14" name="sendDate" visibility="public" isSpecification="false"
kind="inout" type="G.59">
                </UML:Parameter>
                <UML:Parameter xmi.id="XX.2.438.9.15" name="remove.Return" visibility="public"
isSpecification="false" kind="return" type="G.62">
      </UML:Parameter>
            </UML:BehavioralFeature.parameter>
        </UML:Operation>
        <!-- ==================== OrderSystem::StockItem::needsToReorder    [Operation]
==================== -->
        <UML:Operation xmi.id="S.092.0438.07.17" name="needsToReorder" visibility="public"
isSpecification="false" ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false"
isLeaf="false" isAbstract="false" specification="">
            <UML:BehavioralFeature.parameter>
                <UML:Parameter xmi.id="XX.2.438.9.16" name="needsToReorder.Return" visibility="public"
isSpecification="false" kind="return" type="G.60">
      </UML:Parameter>
            </UML:BehavioralFeature.parameter>
        </UML:Operation>
    </UML:Classifier.feature>
</UML:Class>
<!-- ==================== boolean    [DataType] ==================== -->
<UML:DataType xmi.id="G.63" name="boolean" visibility="public" isSpecification="false" isRoot="false" isLeaf="false"
isAbstract="false"/>
<!-- ==================== OrderSystem::ReorderItem    [Class] ==================== -->
<UML:Class xmi.id="S.092.0438.07.18" name="ReorderItem" visibility="public" isSpecification="false" isRoot="true"
isLeaf="true" isAbstract="false" isActive="false" namespace="G.0">
    <UML:Classifier.feature>
        <!-- ==================== OrderSystem::ReorderItem.aStockItem    [Attribute]
==================== -->
        <UML:Attribute xmi.id="S.092.0438.07.19" name="aStockItem" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="S.092.0438.07.11">
            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::ReorderItem.amount    [Attribute] ==================== --
>
        <UML:Attribute xmi.id="S.092.0438.07.20" name="amount" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="G.60">
            <UML:Attribute.initialValue>
                <UML:Expression language="" body=""/>
            </UML:Attribute.initialValue>
        </UML:Attribute>
        <!-- ==================== OrderSystem::ReorderItem.estDateOfArrival    [Attribute]
==================== -->
```

รูปที่ ข.2 ส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพคลาสตามรูปที่ 3.2 (ต่อ)

```
            <UML:Attribute xmi.id="S.092.0438.07.21" name="estDateOfArrival" visibility="private"
isSpecification="false" ownerScope="instance" changeability="changeable" targetScope="instance" type="G.59">
                <UML:Attribute.initialValue>
                    <UML:Expression language="" body=""/>
                </UML:Attribute.initialValue>
            </UML:Attribute>
            <!-- ==================== OrderSystem::ReorderItem::ReorderItem    [Operation]
==================== -->
            <UML:Operation xmi.id="S.092.0438.07.22" name="ReorderItem" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
specification="">
                <UML:BehavioralFeature.parameter>
                    <UML:Parameter xmi.id="XX.2.438.9.18" name="aStockItem" visibility="public"
isSpecification="false" kind="inout" type="S.092.0438.07.11">
                    </UML:Parameter>
                    <UML:Parameter xmi.id="XX.2.438.9.19" name="amount" visibility="public" isSpecification="false"
kind="inout" type="G.60">
                    </UML:Parameter>
                </UML:BehavioralFeature.parameter>
            </UML:Operation>
        </UML:Classifier.feature>
</UML:Class>
<!-- ==================== OrderSystem::DeliveryItem    [Class] ==================== -->
<UML:Class xmi.id="S.092.0438.07.23" name="DeliveryItem" visibility="public" isSpecification="false" isRoot="true"
isLeaf="true" isAbstract="false" isActive="false" namespace="G.0">
        <UML:Classifier.feature>
            <!-- ==================== OrderSystem::DeliveryItem.anOrderLine    [Attribute]
==================== -->
            <UML:Attribute xmi.id="S.092.0438.07.24" name="anOrderLine" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="S.092.0438.07.5">
                <UML:Attribute.initialValue>
                    <UML:Expression language="" body=""/>
                </UML:Attribute.initialValue>
            </UML:Attribute>
            <!-- ==================== OrderSystem::DeliveryItem::DeliveryItem    [Operation]
==================== -->
            <UML:Operation xmi.id="S.092.0438.07.25" name="DeliveryItem" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
specification="">
                <UML:BehavioralFeature.parameter>
                    <UML:Parameter xmi.id="XX.2.438.9.21" name="anOrderLine" visibility="public"
isSpecification="false" kind="inout" type="S.092.0438.07.5">
                    </UML:Parameter>
                </UML:BehavioralFeature.parameter>
            </UML:Operation>
        </UML:Classifier.feature>
</UML:Class>
<!-- ==================== OrderSystem::OrderManager    [Class] ==================== -->
<UML:Class xmi.id="S.092.0438.07.26" name="OrderManager" visibility="public" isSpecification="false"
isRoot="true" isLeaf="true" isAbstract="false" isActive="false" namespace="G.0">
        <UML:Classifier.feature>
            <!-- ==================== OrderSystem::OrderManager.order[]    [Attribute] ==================== -
->
            <UML:Attribute xmi.id="S.092.0438.07.27" name="order[]" visibility="private" isSpecification="false"
ownerScope="instance" changeability="changeable" targetScope="instance" type="S.092.0438.07.1">
                <UML:Attribute.initialValue>
                    <UML:Expression language="" body=""/>
                </UML:Attribute.initialValue>
            </UML:Attribute>
        </UML:Classifier.feature>
</UML:Class>
```

รูปที่ ข.2 ส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพคลาสตามรูปที่ 3.2 (ต่อ)

```
<UML:Collaboration xmi.id="S.092.0438.07.28" name="Logical View-Collaboration" visibility="public"
isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false">
    <UML:Namespace.ownedElement>
        <UML:ClassifierRole xmi.id="G.25" name="anOrder" visibility="public" isSpecification="false" isRoot="false"
isLeaf="false" isAbstract="false" base="S.092.0438.07.1" availableFeature="S.092.0438.07.3" message2="G.38"
message1="G.37">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.27" name="orderLine[i]" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.5" message2="G.39 G.40 G.43"
message1="G.38">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.30" name="aStockItem" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.11" message2="G.41 G.42" message1="G.39
G.40 G.41">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.33" name="aDeliveryItem" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.23" message1="G.43">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.34" name="aReorderItem" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.18" message1="G.42">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.35" name="anOrderManager" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.26" message2="G.37">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.45" name="anOrder" visibility="public" isSpecification="false" isRoot="false"
isLeaf="false" isAbstract="false" base="S.092.0438.07.1" availableFeature="S.092.0438.07.3" message2="G.55"
message1="G.54">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.47" name="orderLine[i]" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.5" message2="G.56 G.57" message1="G.55
G.57">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.50" name="aStockItem" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.11" message1="G.56">
        </UML:ClassifierRole>
        <UML:ClassifierRole xmi.id="G.52" name="anOrderManager" visibility="public" isSpecification="false"
isRoot="false" isLeaf="false" isAbstract="false" base="S.092.0438.07.26" message2="G.54">
        </UML:ClassifierRole>
    </UML:Namespace.ownedElement>
    <UML:Collaboration.interaction>
        <UML:Interaction xmi.id="G.44" name="{Logical View}Normal flow" visibility="public" isSpecification="false">
            <UML:Interaction.message>
                <UML:Message xmi.id="G.37" name="prepareToDeliver(deliveryDate : Date)" visibility="public"
isSpecification="false" sender="G.35" receiver="G.25" message3="G.38" communicationConnection="G.36"
action="XX.2.438.9.41"/>
                <UML:Message xmi.id="G.38" name="*[for(int i = 0; i &lt; orderLine.length; i++)] prepare
(deliveryDate : Date)" visibility="public" isSpecification="false" activator="G.37" sender="G.25" receiver="G.27"
message3="G.39" predecessor="G.37" communicationConnection="G.26" action="XX.2.438.9.42"/>
                <UML:Message xmi.id="G.39" name="hasStock := check(amount : int, deliveryDate : Date)"
visibility="public" isSpecification="false" activator="G.38" sender="G.27" receiver="G.30" message3="G.40"
predecessor="G.38" communicationConnection="G.28" action="XX.2.438.9.43"/>
                <UML:Message xmi.id="G.40" name="remove(amount : int, deliveryDate : Date)" visibility="public"
isSpecification="false" activator="G.38" sender="G.27" receiver="G.30" message3="G.41" predecessor="G.39"
communicationConnection="G.28" action="XX.2.438.9.44"/>
                <UML:Message xmi.id="G.41" name="needsReorder := needsToReorder()" visibility="public"
isSpecification="false" activator="G.40" sender="G.30" receiver="G.30" message3="G.42" predecessor="G.40"
communicationConnection="G.31" action="XX.2.438.9.45"/>
                <UML:Message xmi.id="G.42" name="[needsReorder &gt; 0] &lt;&lt;create&gt;&gt; (this : StockItem,
needsReorder : int)" visibility="public" isSpecification="false" activator="G.40" sender="G.30" receiver="G.34"
message3="G.43" predecessor="G.41" communicationConnection="G.32" action="XX.2.438.9.46"/>
                <UML:Message xmi.id="G.43" name="&lt;&lt;create&gt;&gt; (this : OrderLine)" visibility="public"
isSpecification="false" activator="G.38" sender="G.27" receiver="G.33" predecessor="G.42"
communicationConnection="G.29" action="XX.2.438.9.47"/>
            </UML:Interaction.message>
        </UML:Interaction>
        <UML:Interaction xmi.id="G.58" name="{Logical View}Exceptional flow (out of stock)" visibility="public"
isSpecification="false">
```

รูปที่ ข.3 ส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพซีเควนซ์ตามรูปที่ 3.3 และ 3.4

```
<UML:Interaction.message>
                    <UML:Message xmi.id="G.54" name="prepareToDeliver(date : Date)" visibility="public"
isSpecification="false" sender="G.52" receiver="G.45" message3="G.55" communicationConnection="G.53"
action="XX.2.438.10.62"/>
                    <UML:Message xmi.id="G.55" name="*[for(int i = 0; i &lt; orderLine.length; i++)] prepare(date :
Date)" visibility="public" isSpecification="false" activator="G.54" sender="G.45" receiver="G.47" message3="G.56"
predecessor="G.54" communicationConnection="G.46" action="XX.2.438.10.63"/>
                    <UML:Message xmi.id="G.56" name="hasStock := check(amount : int, date : Date)"
visibility="public" isSpecification="false" activator="G.55" sender="G.47" receiver="G.50" message3="G.57"
predecessor="G.55" communicationConnection="G.48" action="XX.2.438.10.64"/>
                    <UML:Message xmi.id="G.57" name='changeStatus("Out of Stock" : String)' visibility="public"
isSpecification="false" activator="G.55" sender="G.47" receiver="G.47" predecessor="G.56"
communicationConnection="G.49" action="XX.2.438.10.65"/>
                </UML:Interaction.message>
            </UML:Interaction>
        </UML:Collaboration.interaction>
</UML:Collaboration>
```

รูปที่ ข.3 ส่วนของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่สอดคล้องกับแผนภาพซีเควนซ์ตามรูปที่ 3.3 และ 3.4 (ต่อ)

ภาคผนวก ค
การใช้งานเครื่องมือ

ในภาคผนวกนี้จะเป็นการกล่าวถึงการใช้งานเครื่องมือสนับสนุนการทำการแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวาที่พัฒนาขึ้นมาในงานวิจัยนี้

ค.1 เครื่องมืออื่น ๆ ที่ใช้ร่วมกับเครื่องมือที่พัฒนาขึ้น

ในการใช้งานเครื่องมือนี้ ผู้ใช้จะต้องสร้างแผนภาพคลาสและแผนภาพซีเควนซ์ด้วย เรชันแนล โรส รุ่น 2002 และทำการส่งออกข้อมูลแผนภาพดังกล่าวในรูปของแฟ้มข้อมูลเอ็กซ์เอ็มไอด้วยเครื่องมือเสริม ยูนิซิส โรส เอ็กซ์เอ็มแอล ทูล รุ่น 1.3.6

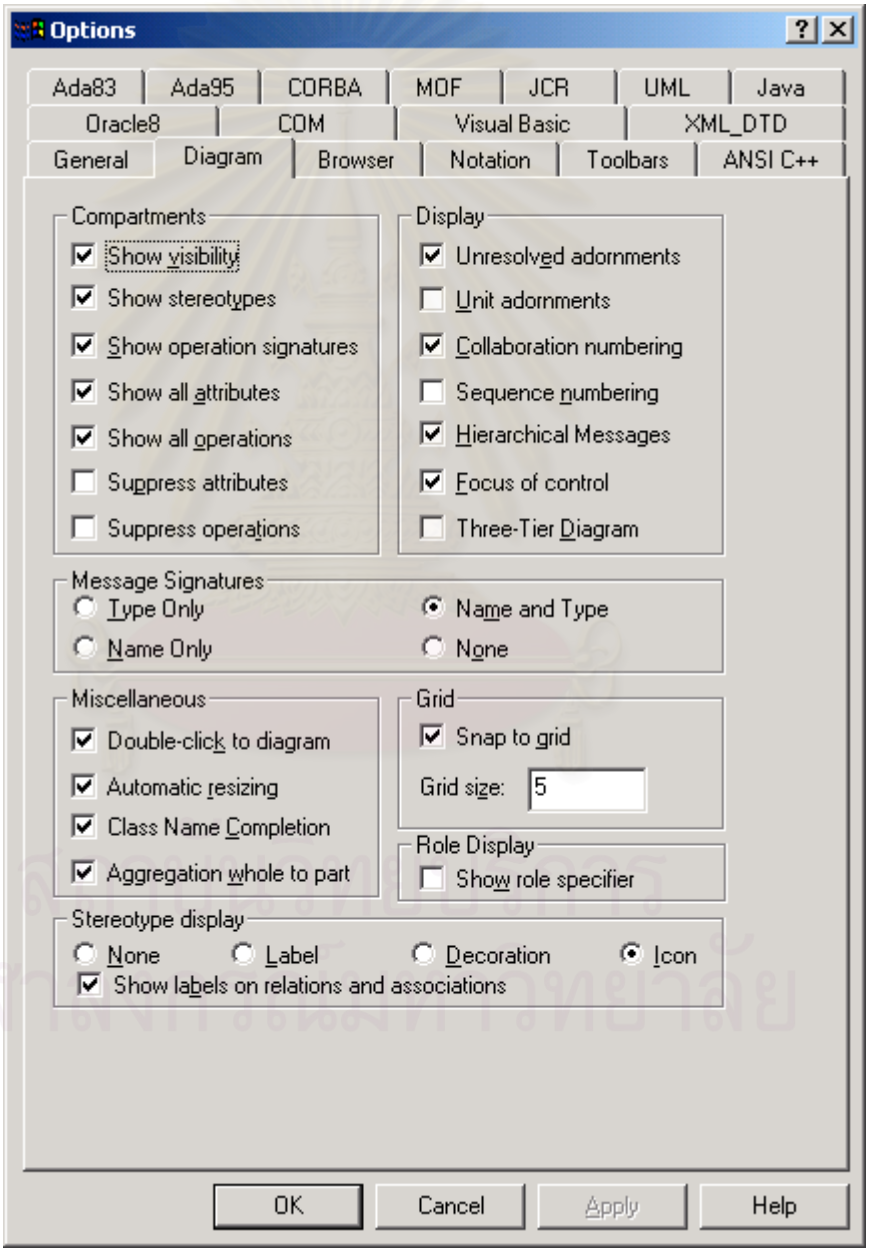
ส่วนเครื่องมือที่จะใช้ทำการคอมไพล์รหัสคำสั่งที่สร้างขึ้นจากเครื่องมือนี้ สามารถใช้คอมไพเลอร์ที่ทำตามข้อกำหนดของภาษาจาวารุ่นที่ 2 ตัวใดก็ได้

ค.2 การปรับตั้งค่า เรชันแนล โรส

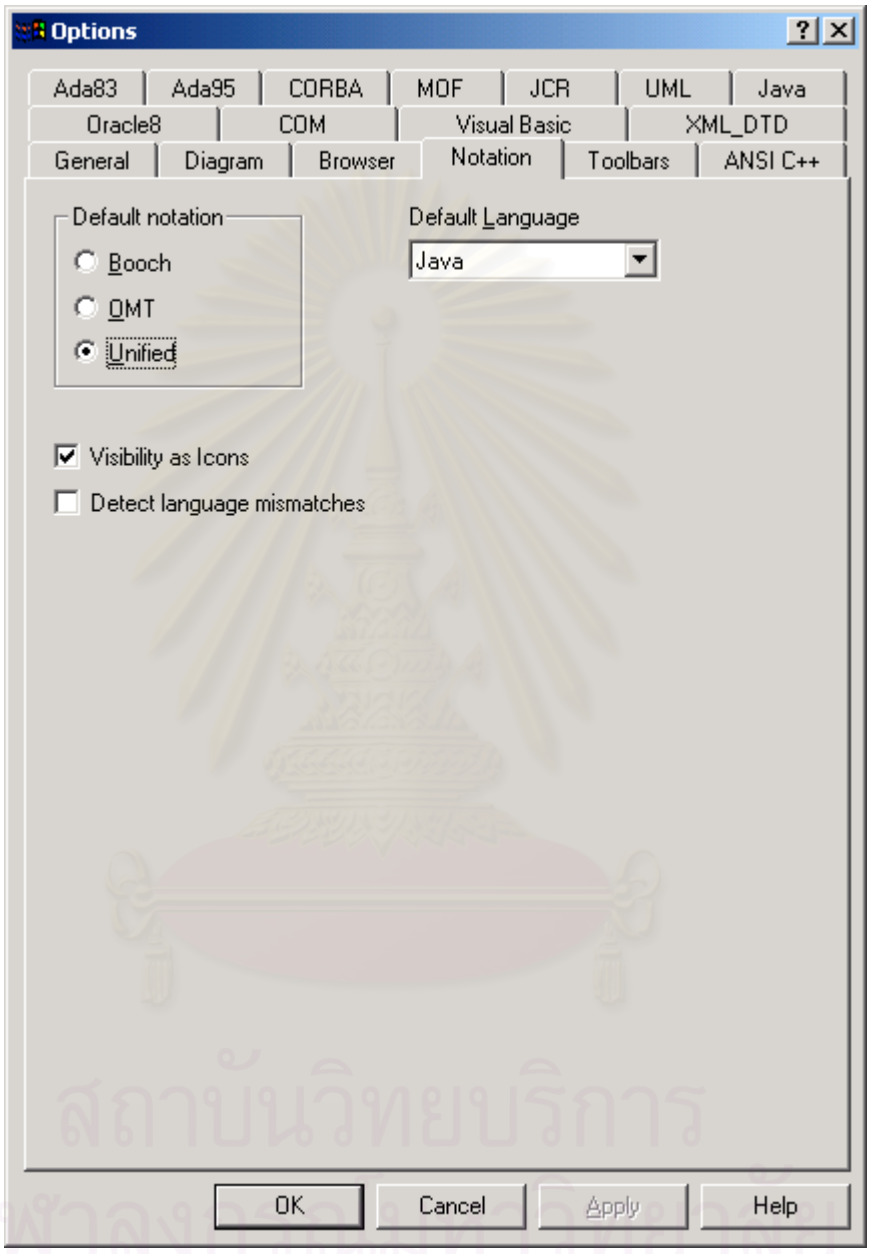
เพื่อให้การสร้างแผนภาพมีความถูกต้อง และเหมาะสมที่จะให้เครื่องมือที่พัฒนาขึ้นมาในงานวิจัยนี้ทำการแปลง จะต้องแก้ไขทางเลือก (Options) ของ เรชันแนล โรส จากค่าเริ่มต้น ตามขั้นตอนต่อไปนี้

1. ไปที่เมนู “Tools” แล้วเลือกเมนูย่อย “Options” จะปรากฏหน้าต่าง “Options” ขึ้นมา
2. ทำการเลือกแท็บ (Tab) “Diagram” จะปรากฏหน้าจอแสดงดังรูปที่ ค.1
3. ในกรอบ “Compartments” ทำการเลือก “Show operation signatures” เพื่อให้แสดงรายละเอียดของรายการพารามิเตอร์ของแต่ละโอเปอร์เรชันในแผนภาพคลาส
4. ในกรอบ “Display” ทำการเลือก “Hierarchical Messages” เพื่ออนุญาตให้เกิดการซ้อนกันของโฟกัสออฟคอนโทรลขึ้นได้
5. ในกรอบ “Message Signatures” ทำการเลือก “Name and Type” เพื่อให้แสดงรายละเอียดของรายการพารามิเตอร์ของแต่ละการส่งเมสเสจในแผนภาพซีเควนซ์

6. ทำการเลือกแท็บ “Notation” จะปรากฏหน้าจอแสดงดังรูปที่ ค.2
7. ในกล่อง “Default Language” ทำการเลือกให้เป็น “Java” เพื่อให้สามารถกำหนดมอดดิฟายเออร์ (Modifier) ของคลาส แอทริบิวต์ และ โอเปอร์เรชัน ในแผนภาพคลาส ให้เป็นไปตามข้อกำหนดของภาษาจาวาได้
8. กดปุ่ม “OK”



รูปที่ ค.1 หน้าจอเมื่อทำการเลือกแท็บ “Diagram” ในหน้าต่าง “Options”



รูปที่ ค.2 หน้าจอเมื่อทำการเลือกแท็บ “Notation” ในหน้าต่าง “Options”

ค.3 การสร้างแผนภาพคลาสและแผนภาพซีเควนซ์

เครื่องมือในงานวิจัยนี้จะทำการแปลงข้อมูลที่ได้จากแผนภาพยูเอ็มแอลสองชนิด ได้แก่ แผนภาพคลาสและแผนภาพซีเควนซ์ โดยข้อมูลในแผนภาพคลาสจะถูกนำไปสร้างเป็นรหัสคำสั่งเชิงโครงสร้าง ซึ่งก็คือการประกาศคลาส แอทริบิวต์ของคลาส และโอเปอร์เรชันของคลาส ส่วนข้อมูลในแผนภาพซีเควนซ์จะถูกนำไปสร้างเป็นรหัสคำสั่งเชิงพฤติกรรมในระดับโอเปอร์เรชัน ซึ่งก็คือรหัสคำสั่งภายในแต่ละโอเปอร์เรชันนั่นเอง

โดยแผนภาพที่นำมาทำการแปลงดังกล่าวจะต้องได้รับออกแบบอย่างละเอียดและสร้างขึ้นตามหัวข้อที่ ค.3.1 และ ค.3.2 ที่จะได้กล่าวถึงต่อไป เพื่อให้ได้ผลการแปลงที่สมบูรณ์

ค.3.1 การสร้างแผนภาพคลาส

การสร้างแผนภาพคลาสซึ่งประกอบไปด้วย คลาส อินเทอร์เฟส และความสัมพันธ์ มีรายละเอียดในการสร้างดังต่อไปนี้

ค.3.1.1 การระบุรายละเอียดในคลาสและอินเทอร์เฟส

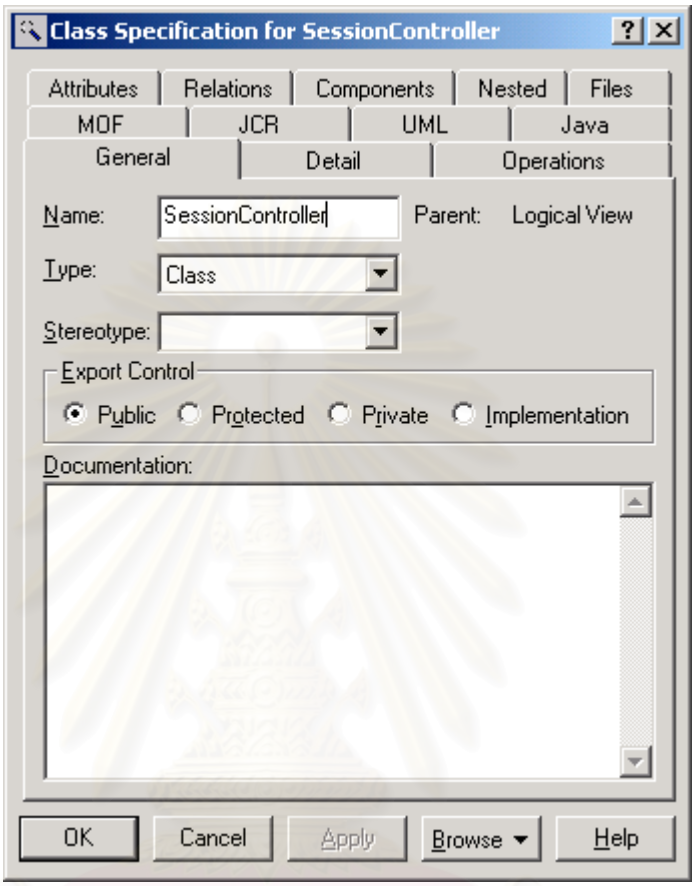
เนื่องจากรายละเอียดของแต่ละคลาสและอินเทอร์เฟสในเรชันแนล โรส มีค่อนข้างมาก และในบางรายการอาจพบตำแหน่งที่สามารถระบุได้หลายตำแหน่ง โดยที่แต่ละตำแหน่งอาจให้ผลเป็นเอ็กซ์เอ็มไอที่แตกต่างกันออกไป ดังนั้นเพื่อจะให้เครื่องมือทำการแปลง ฯ ได้อย่างถูกต้อง ผู้ใช้จะต้องระบุรายละเอียดของแต่ละคลาสและอินเทอร์เฟสตามวิธีที่อธิบายในหัวข้อนี้

การระบุรายละเอียดดังกล่าวเริ่มต้นจากทำการคลิกขวาบนคลาสหรืออินเทอร์เฟสที่ต้องการแล้วเลือกเมนู “Open Standard Specification...” โดยจะปรากฏหน้าต่างดังรูปที่ ค.3

ในแท็บ “General” ให้ระบุชื่อในช่อง “Name” และเลือกค่าในกรอบ “Export Control” เฉพาะ “Public” “Protected” หรือ “Private” เท่านั้น ในกรณีที่เป็นอินเทอร์เฟสให้เลือกค่าในช่อง “Stereotype” เป็น “Interface”

ในแท็บ “Detail” ให้ทำการเลือกหรือไม่เลือกกล่อง “Abstract” เพื่อระบุว่าเป็นแอบสแตรกคลาสหรือไม่

ในแท็บ “Java” ให้ทำการเลือกว่าคลาสหรืออินเทอร์เฟสนี้เป็นแบบไฟนอลหรือไม่



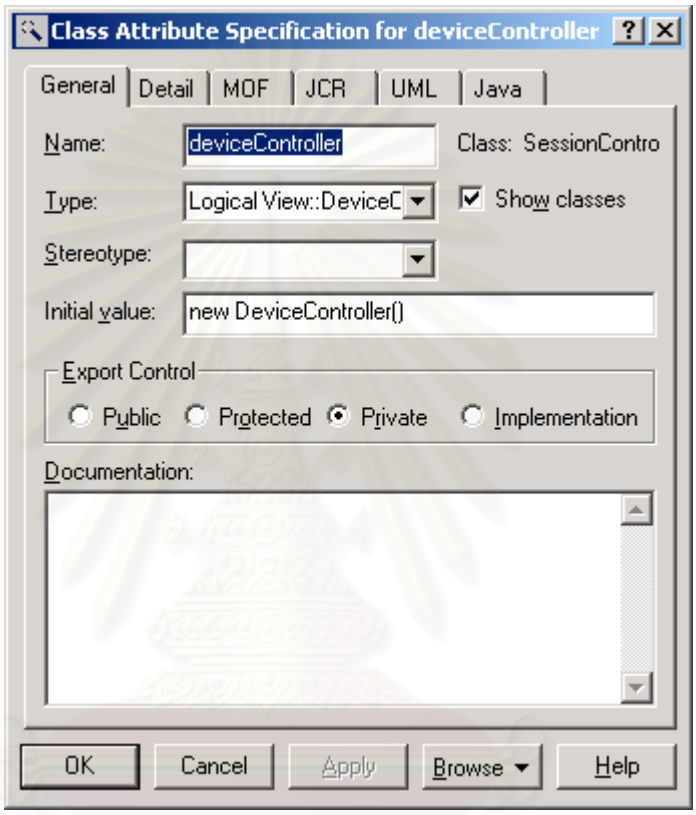
รูปที่ ค.3 หน้าต่าง “Class Specification”

ในส่วนของแอทริบิวต์ เครื่องมือจะไม่ทำการแปลงความสัมพันธ์ประเภทต่าง ๆ ไปเป็นแอทริบิวต์ให้โดยอัตโนมัติ ผู้ใช้จะต้องทำการสร้างแอทริบิวต์ทั้งหมดในแต่ละคลาสด้วยตนเอง และในการระบุชนิดของข้อมูลประเภทปฐมฐาน ผู้ใช้จะต้องใช้ชนิดของข้อมูลประเภทปฐมฐานที่มีในข้อกำหนดของภาษาจาวา ส่วนในการระบุรายละเอียดต่าง ๆ ของแต่ละแอทริบิวต์จะต้องทำโดยการคลิกขวาที่แอทริบิวต์ที่ต้องการในหน้าต่าง “Browser” แล้วเลือกเมนู “Open Standard Specification…” โดยจะปรากฏหน้าต่างดังรูปที่ ค.4

ในแท็บ “General” ให้ระบุชื่อในช่อง “Name” ชนิดของข้อมูลในช่อง “Type” ค่าเริ่มต้นในช่อง “Initial value” และเลือกค่าในกรอบ “Export Control” เฉพาะ “Public” “Protected” หรือ “Private” เท่านั้น

ในแท็บ “Detail” ให้ทำการเลือกหรือไม่เลือกกล่อง “Static” เพื่อระบุว่าเป็นชนิดสแตติกหรือไม่

ในแท็บ “Java” ให้ทำการเลือกว่าแอทริบิวต์นี้เป็นแบบไฟนอลหรือไม่



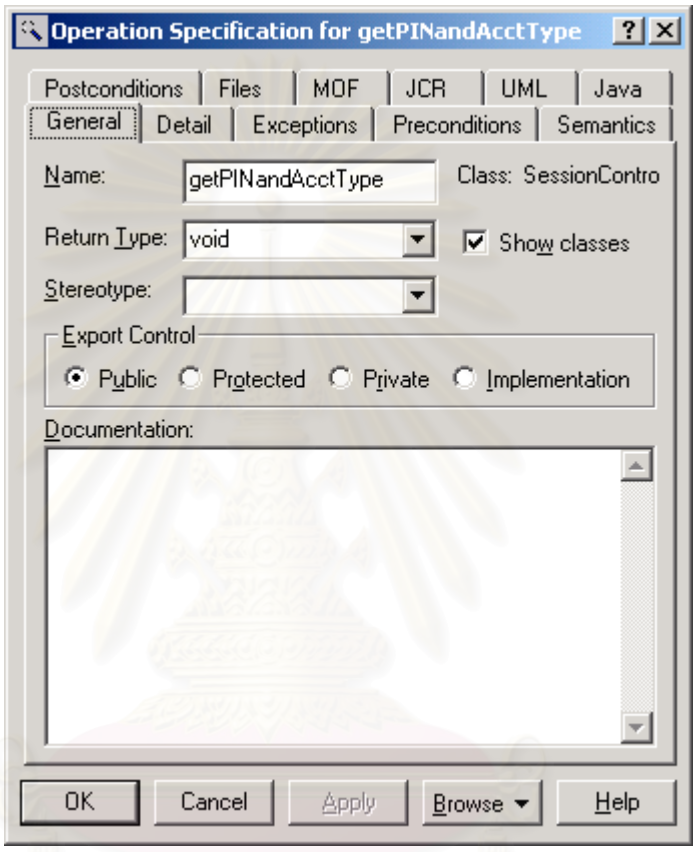
รูปที่ ค.4 หน้าต่าง “Class Attribute Specification”

ในการระบุรายละเอียดต่าง ๆ ของแต่ละโอเปอร์เรชันจะต้องทำโดยการคลิกขวาที่โอเปอร์เรชันที่ต้องการในหน้าต่าง “Browser” แล้วเลือกเมนู “Open Standard Specification…” โดยจะปรากฏหน้าต่างดังรูปที่ ค.5

ในแท็บ “General” ให้ระบุชื่อในช่อง “Name” ชนิดของข้อมูลส่งกลับในช่อง “Return Type” และเลือกค่าในกรอบ “Export Control” เฉพาะ “Public” “Protected” หรือ “Private” เท่านั้น

ในแท็บ “Detail” ให้ทำการระบุพารามิเตอร์ลงในส่วนของ “Arguments” โดยในแต่ละพารามิเตอร์ให้ระบุชื่อในคอลัมน์ “Name” และชนิดข้อมูลในคอลัมน์ “Type”

ในแท็บ “Java” ให้ทำการเลือกว่าแอทริบิวต์นี้เป็นแบบแอบสแตรก สแตติก และไฟนอลหรือไม่



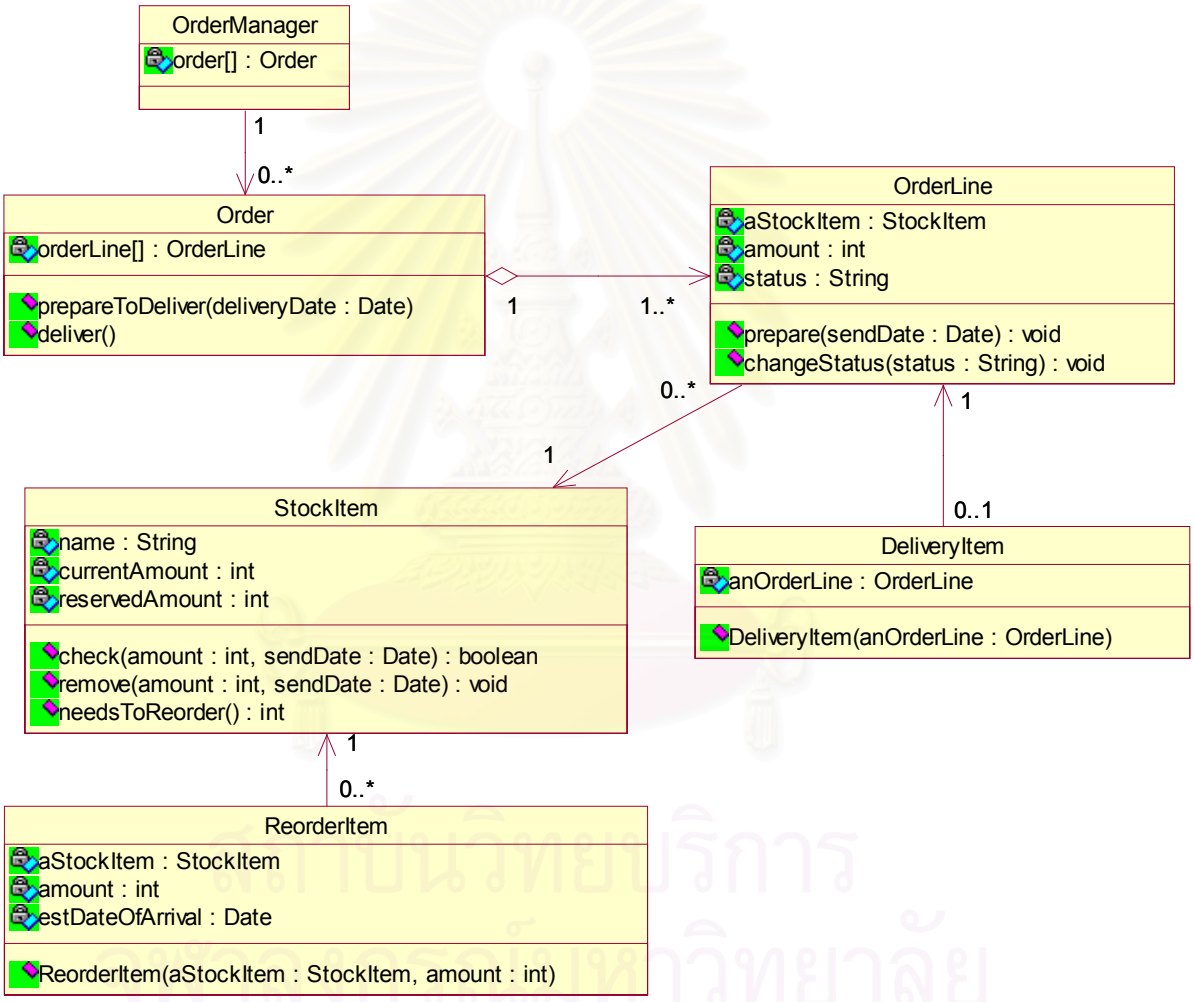
รูปที่ ค.5 หน้าต่าง “Operation Specification”

ค.3.1.2 การสร้างความสัมพันธ์

ความสัมพันธ์ที่เครื่องมือนี้จะทำการแปลงไปเป็นรหัสคำสั่งมีอยู่ 2 ชนิด คือความสัมพันธ์แบบเจเนอรัลไลเซชันและความสัมพันธ์แบบแบบเรียลไลเซชันซึ่งจะถูกแปลงไปเป็นสเตทเมนต์ “extends” และ “implements” ตามลำดับ ส่วนความสัมพันธ์แบบอื่น ๆ นั้นสามารถใส่ลงในแผนภาพเพื่อช่วยในการอธิบายได้แต่จะไม่มีส่วนเกี่ยวข้องกับการทำการแปลงไปเป็นรหัสคำสั่งของเครื่องมือ

ค.3.1.3 ตัวอย่างของแผนภาพคลาส

รูปที่ ค.6 เป็นตัวอย่างของแผนภาพคลาสของระบบสั่งซื้อสินค้า (ส่วนเตรียมการจัดส่งสินค้า) โดยเครื่องมือจะนำรายละเอียดในแต่ละคลาสไปสร้างเป็นรหัสคำสั่งเชิงโครงสร้างของคลาสนั้น



รูปที่ ค.6 แผนภาพคลาสของระบบสั่งซื้อสินค้า (ส่วนเตรียมการจัดส่งสินค้า)

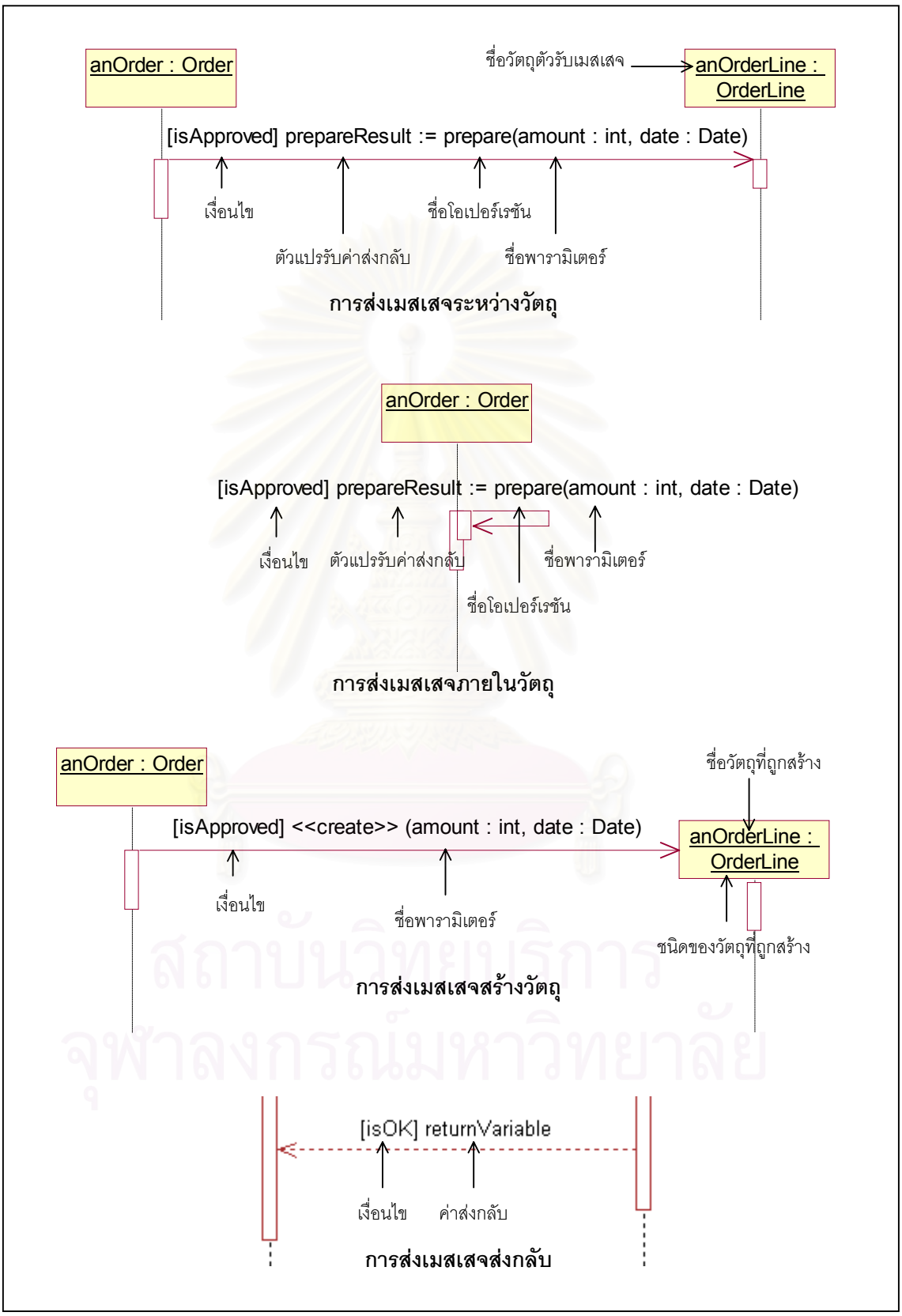
ค.3.2 การสร้างแผนภาพซีเควนซ์

การสร้างแผนภาพซีเควนซ์ซึ่งเป็นแผนภาพที่ใช้แสดงลำดับการส่งเมสเสจระหว่างวัตถุต่าง ๆ มีรายละเอียดดังต่อไปนี้

ค.3.2.1 ข้อกำหนดในการสร้างแผนภาพซีเควนซ์

เพื่อให้เครื่องมือสามารถทำการแปลงแผนภาพซีเควนซ์ได้ถูกต้อง ผู้ใช้จะต้องทำตามข้อกำหนดในการสร้างแผนภาพซีเควนซ์ดังต่อไปนี้

1. ต้องระบุชื่อของวัตถุทุกตัว
2. ใช้โฟกัสออฟคอนโทรลในการบอกขอบเขตพฤติกรรมของแต่ละโอเปอร์เรชัน โดยลำดับการส่งเมสเสจที่ถูกส่งออกจากโฟกัสออฟคอนโทรลตัวใด จะถูกแปลงไปเป็นพฤติกรรมของโอเปอร์เรชันที่มีความสัมพันธ์กับการส่งเมสเสจที่เป็นตัวกระตุ้นให้เกิดโฟกัสออฟคอนโทรลตัวนั้น
3. การระบุพฤติกรรมของคอนสตรัคเตอร์ให้กับวัตถุที่ถูกสร้างด้วยการส่งเมสเสจแบบสร้างวัตถุ สามารถทำได้โดยให้กลุ่มลำดับการส่งเมสเสจที่เป็นพฤติกรรมของคอนสตรัคเตอร์ถูกส่งออกจากโฟกัสออฟคอนโทรลตัวที่อยู่บนสุดของวัตถุนั้น
4. เมื่อมีการส่งเมสเสจที่ไม่ใช่แบบส่งกลับส่งเข้าหาโฟกัสออฟคอนโทรลที่มีอยู่เดิมจะต้องเกิดการโฟกัสออฟคอนโทรลตัวใหม่ซ้อนขึ้นบนโฟกัสออฟคอนโทรลตัวที่มีอยู่เดิม
5. การส่งเมสเสจประเภทต่าง ๆ ในกรณีทั่วไปจะมีรูปแบบดังรูปที่ ค.7
6. การระบุเงื่อนไขของการส่งเมสเสจเป็นแบบทำซ้ำ ทำได้โดยระบุเครื่องหมายดอกจัน (*) ไว้ข้างหน้าการระบุเงื่อนไข และจะต้องระบุสเตทเมนต์ในการทำซ้ำด้วย เช่น “*[for(int i = 0; i < 10; i++)]” หรือ “*[while(i++ < n)]” เป็นต้น
7. ไม่มีการส่งเมสเสจแบบอะซิงโครนัส
8. ไม่มีการส่งเมสเสจเกิดขึ้นพร้อมกันหลายการส่งเมสเสจ



รูปที่ ค.7 รูปแบบของการส่งเมสเสจประเภทต่าง ๆ

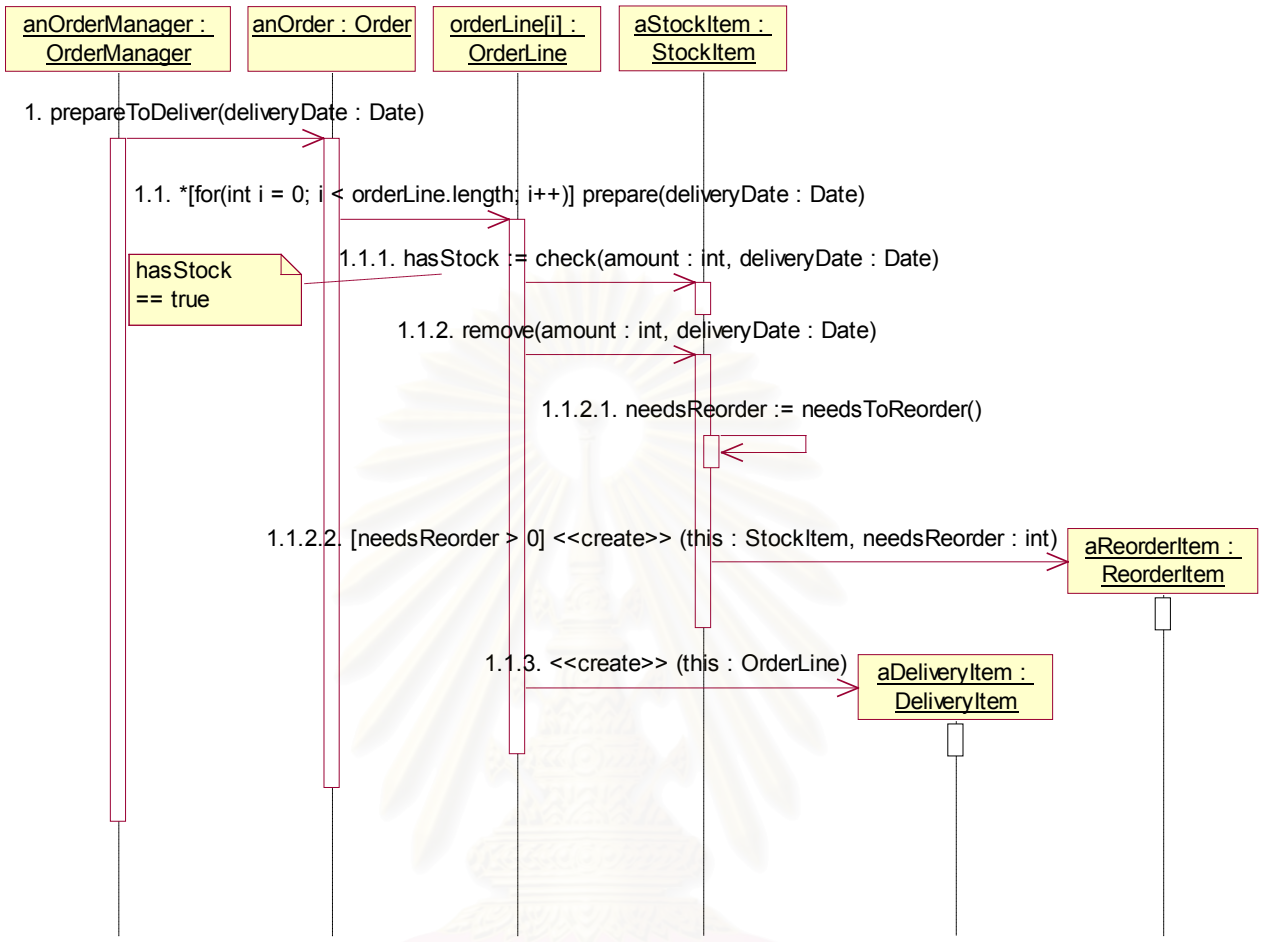
ค.3.2.2 ข้อแนะนำในการสร้างแผนภาพที่มีหลายสถานการณ์

ข้อแนะนำในการสร้างแผนภาพที่มีหลายสถานการณ์มีดังต่อไปนี้

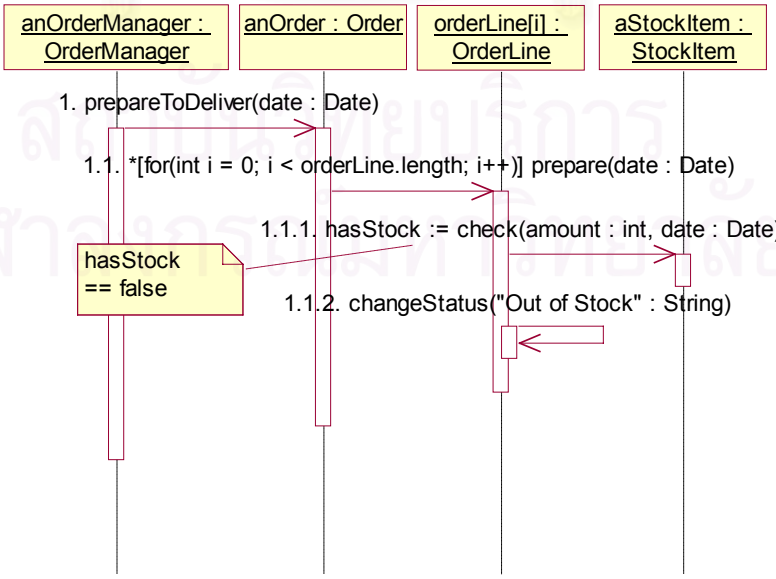
1. ควรจะตั้งชื่อของแผนภาพซีเควนซ์ให้สื่อถึงสถานการณ์ของแผนภาพนั้น เนื่องจากชื่อเหล่านี้จะถูกนำไปอธิบายตัวเลขที่ใช้แทนแต่ละสถานการณ์ในรหัสคำสั่งในส่วนของคอมเมนต์
2. ไม่จำเป็นต้องระบุเงื่อนไขที่ก่อให้เกิดสถานการณ์ของแผนภาพนั้นลงในแต่ละการส่งเมสเสจที่เป็นพฤติกรรมของสถานการณ์ดังกล่าว แต่อาจทำการบันทึกเงื่อนไขที่ก่อให้เกิดสถานการณ์ของแต่ละแผนภาพด้วยเครื่องมือ “โน้ต” (Note) เพื่อเป็นข้อมูลในขั้นตอนที่ผู้ใช้ต้องทำการเพิ่มรหัสคำสั่งส่วนของการเลือกสถานการณ์ลงในรหัสคำสั่งที่ได้จากการทำการแปลง ฯ ด้วยตนเองในภายหลัง

ค.3.2.3 ตัวอย่างของแผนภาพซีเควนซ์

รูปที่ ค.8 และ ค.9 เป็นตัวอย่างของแผนภาพซีเควนซ์ของระบบสั่งซื้อสินค้า (ส่วนเตรียมการจัดส่งสินค้า) โดยรูปที่ ค.8 แสดงสถานการณ์ที่เตรียมการจัดส่งสินค้าได้สำเร็จ (สถานการณ์ปกติ) ส่วนรูปที่ ค.9 แสดงสถานการณ์ที่ไม่มีสินค้าเหลือเพียงพอที่จะจัดส่ง โดยเครื่องมือจะนำข้อมูลที่ปรากฏในแผนภาพซีเควนซ์ทั้งสองนี้ไปสร้างเป็นรหัสคำสั่งภายในแต่ละโอเปอร์เรชัน



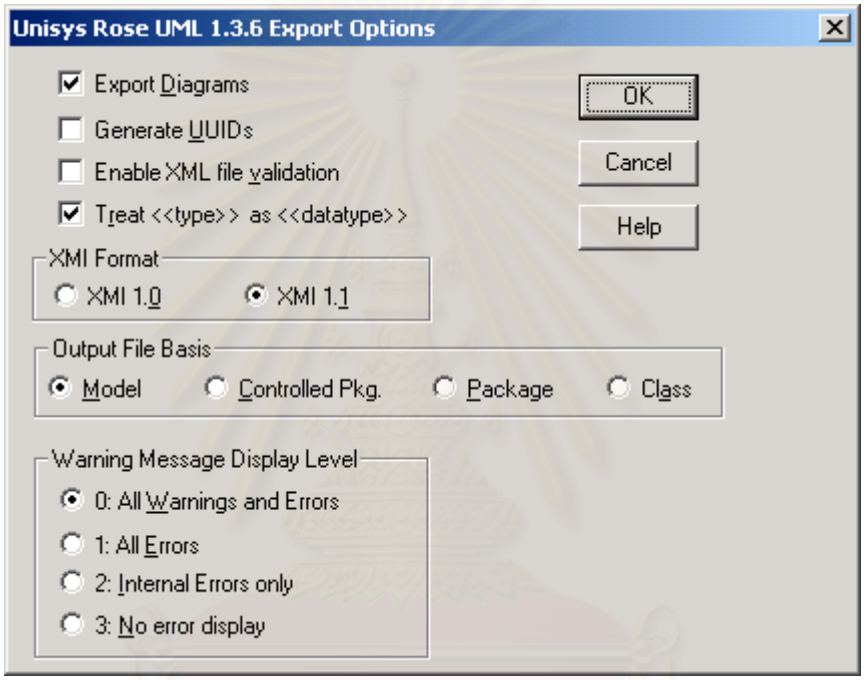
รูปที่ ค.8 แผนภาพซีเควนซ์แสดงสถานการณ์ที่เตรียมการจัดส่งสินค้าได้สำเร็จ (สถานการณ์ปกติ)



รูปที่ ค.9 แผนภาพซีเควนซ์แสดงสถานการณ์ที่ไม่มีสินค้าเหลือเพียงพอที่จะจัดส่ง

ค.4 การส่งออกข้อมูลจาก เรชันแนล โรส

การส่งออกข้อมูลจาก เรชันแนล โรส ทำได้โดยเลือกเมนู “Tools” และเลือกเมนูย่อย “UML 1.3 XMI Addin” และ “UML 1.3 XMI Export” ตามลำดับ จะปรากฏหน้าต่างดังรูปที่ ค.10 ซึ่งผู้ใช้ไม่ต้องแก้ไขค่าใด ๆ จากค่าเริ่มต้น โดยสามารถกดปุ่ม “OK” เพื่อทำการส่งออกข้อมูลได้ทันที



รูปที่ ค.10 เครื่องมือส่งออกข้อมูลจาก เรชันแนล โรส

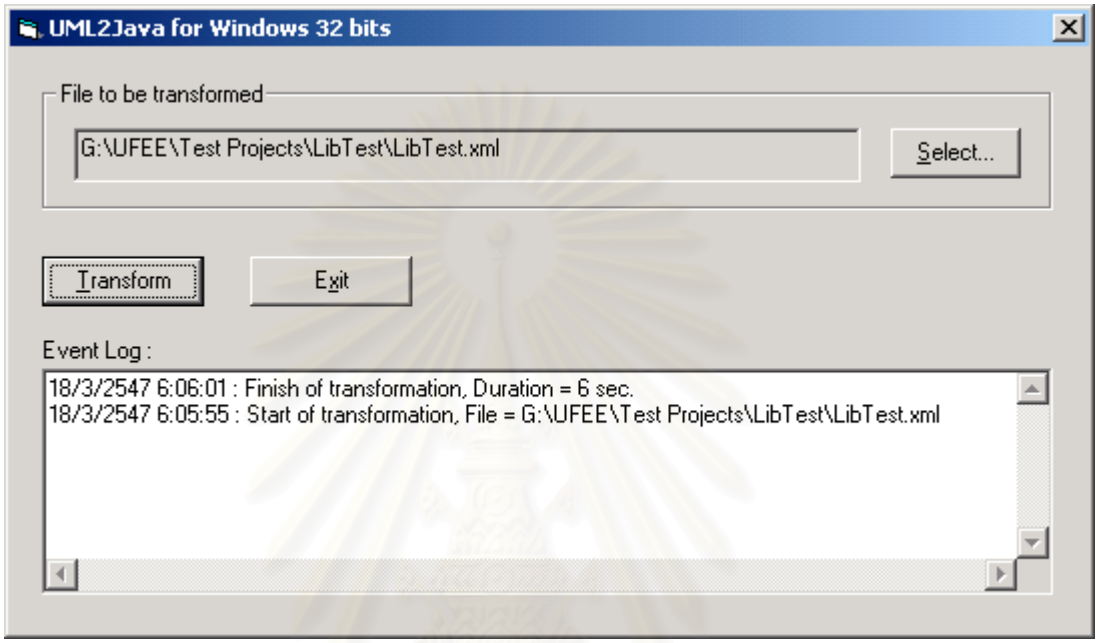
ค.5 การทำการแปลง ฯ

ในงานวิจัยนี้ได้มีการพัฒนาเครื่องมือขึ้นมาสำหรับแพลตฟอร์มวินโดวส์ 32 บิตและแพลตฟอร์มจาวา ซึ่งจะขอแยกอธิบายการทำการแปลง ฯ ด้วยเครื่องมือสำหรับแต่ละแพลตฟอร์มดังต่อไปนี้

ค.5.1 การทำการแปลง ฯ ด้วยเครื่องมือสำหรับแพลตฟอร์มวินโดวส์ 32 บิต

เมื่อเรียกใช้เครื่องมือ จะปรากฏหน้าต่างดังแสดงในรูปที่ ค.11 ผู้ใช้สามารถทำการเลือกแฟ้มข้อมูลที่ส่งออกมาจาก เรชันแนล โรส ได้โดยกดปุ่ม “Select” และสามารถทำการแปลงข้อมูลในแฟ้มข้อมูลดังกล่าวได้โดยการกดปุ่ม “Transform” เมื่อเครื่องมือทำการแปลงเสร็จเรียบร้อย

แล้ว จะให้ผลการแปลงเป็นแฟ้มข้อมูลที่มีนามสกุล “java” ในไดเร็คทอรี (Directory) เดียวกันกับแฟ้มข้อมูลที่ทำการแปลง



รูปที่ ค.11 เครื่องมือทำการแปลง ฯ สำหรับแพลตฟอร์มวินโดวส์ 32 บิต

ค.5.2 การทำการแปลง ฯ ด้วยเครื่องมือสำหรับแพลตฟอร์มจาวา

คำสั่งเพื่อให้แซกเซินซึ่งเป็นเอ็กซ์เอสแอลทีโปรเซสเซอร์ของเครื่องมือสำหรับแพลตฟอร์มจาวาทำการแปลง ฯ เป็นดังนี้

```
java -jar dir/saxon/saxon7.jar -o uml2java.log input-file dir/Seq2Java_Saxon.xslt
```

โดยผู้ใช้จะต้องทำการแทนที่ “dir” ด้วยไดเร็คทอรีของเครื่องมือ และแทนที่ “input-file” ด้วยแฟ้มข้อมูลที่ต้องการทำการแปลง เมื่อเครื่องมือทำการแปลงเสร็จเรียบร้อยแล้ว จะให้ผลการแปลงเป็นแฟ้มข้อมูลที่มีนามสกุล “java” ในไดเร็คทอรีเดียวกันกับแฟ้มข้อมูลที่ทำการแปลง

ค.6 การเพิ่มรหัสคำสั่งลงในรหัสคำสั่งที่ได้จากการทำการแปลง ฯ

หลังจากทำการแปลง ฯ ด้วยเครื่องมือแล้ว ผู้ใช้จะต้องเพิ่มรหัสคำสั่งส่วนของการเลือกสถานการณ์ด้วยตนเองลงในโอเปอร์เรชันที่มีพฤติกรรมแตกต่างกันตามแต่ละสถานการณ์ โดยจะ

ต้องระบุเงื่อนไขที่ก่อให้เกิดสถานการณ์ต่าง ๆ และมอบหมายค่าให้กับตัวแปร “$_scenario” ให้สอดคล้องกับสถานการณ์เหล่านั้น นอกจากนั้นถ้ามีการอ้างอิงถึงคลาสของแพคคิจ (Package) อื่น ก็จะต้องทำการนำเข้าคลาสนั้นด้วยสเตทเมนต์ “import” ด้วยตนเอง

รูปที่ ค.12 เป็นการแสดงรหัสคำสั่งของแฟ้มข้อมูล “OrderLine.java” ที่ได้จากการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ตามรูปที่ ค.6 ค.8 และ ค.9 และได้ทำการเพิ่มรหัสคำสั่งด้วยตนเองแล้ว โดยรหัสคำสั่งส่วนที่ทำการเพิ่มด้วยตนเองได้ถูกเน้นด้วยตัวอักษรแบบหนา

```
import java.util.*;
public class OrderLine {

	private StockItem aStockItem;
	private int amount;
	private String status;

	public void prepare(Date sendDate) {

		boolean hasStock;
		DeliveryItem aDeliveryItem;

		int $_scenario = 0;

		/*****************************************
		  Scenario numbers :
			# 0 = {Logical View}Normal flow
			# 1 = {Logical View}Exceptional flow (out of stock)
		 *****************************************/


		// Generated behaviour from sequence diagrams :

		hasStock = aStockItem.check(amount, sendDate);

		if (hasStock)
			$_scenario = 0;
		else
			$_scenario = 1;

		if ($_scenario == 1) {
			changeStatus("Out of Stock");
		}

		if ($_scenario == 0) {
			aStockItem.remove(amount, sendDate);
			aDeliveryItem = new DeliveryItem(this);
		}
	}

	public void changeStatus(String status) {

	}
}
```

รูปที่ ค.12 รหัสคำสั่งของแฟ้มข้อมูล “OrderLine.java” ที่ได้เพิ่มรหัสคำสั่งด้วยตนเองแล้ว

ภาคผนวก ง
สรุปขั้นตอนและกฎการแปลง ฯ

ในภาคผนวกนี้จะเป็นการสรุปขั้นตอนและกฎการแปลงแปลงแผนภาพซีเควนซ์หลายแผนภาพไปเป็นพฤติกรรมในระดับโอเปอร์เรชันของรหัสคำสั่งภาษาจาวาในบทที่ 3 โดยขั้นตอนการแปลง ฯ แสดงได้ดังรูปที่ ง.1

รูปที่ ง.1 ขั้นตอนต่าง ๆ ในการทำการแปลง ฯ

ในขั้นตอนการแปลง ฯ แต่ละขั้นตอนดังกล่าว จะทำการแปลงตามกฎการแปลง ฯ ดังต่อไปนี้ โดยเลขที่ของแต่ละกฎจะสอดคล้องกันกับหมายเลขลำดับของแต่ละขั้นตอนในรูปที่ ง.1

กฎที่ 1 พฤติกรรมของโอเปอร์เรชันใด ๆ จะประกอบไปด้วยกลุ่มลำดับการส่งเมสเสจที่มีตัวกระตุ้นที่มีความสัมพันธ์อยู่กับโอเปอร์เรชันนั้น

กฎที่ 2 การส่งเมสเสจใด ๆ หากมีการอ้างถึงพารามิเตอร์ของตัวกระตุ้นด้วยนามแฝงในการส่งเมสเสจนั้นหรือในวัตถุที่เมสเสจนั้นส่งถึง จะต้องแทนที่นามแฝงนั้นด้วยชื่อที่แท้จริงของพารามิเตอร์ที่กำลังอ้างถึง

กฎที่ 3 สำหรับโอเปอร์เรชันใด ๆ ถ้ามีกลุ่มลำดับการส่งเมสเสจที่มีตัวกระตุ้นที่มีความสัมพันธ์อยู่กับโอเปอร์เรชันนั้นมากกว่า 1 กลุ่มลำดับ จะต้องทำการรวมกลุ่มลำดับการส่งเมสเสจเหล่านั้นเข้าด้วยกัน โดยที่จะต้องรักษาลำดับของแต่ละการส่งเมสเสจของผลการรวมให้ถูกต้องเมื่อเทียบกับกลุ่มลำดับการส่งเมสเสจดั้งเดิมแต่ละกลุ่มลำดับ และอาจทำการรวมการส่งเมสเสจที่เหมือนกันจากคนละกลุ่มลำดับการส่งเมสเสจเข้าเป็นการส่งเมสเสจเดียวกันได้

กฎที่ 4 ในกรณีที่มีผลลัพธ์จากการรวมกลุ่มลำดับการส่งเมสเสจที่เป็นพฤติกรรมของโอเปอร์เรชันใด ๆ มีหลายผลลัพธ์ จะเลือกผลลัพธ์ที่มีจำนวนการส่งเมสเสจและความซับซ้อนของสถานการณ์น้อยที่สุดโดยพิจารณาที่จำนวนการส่งเมสเสจก่อน ไปใช้ในขั้นตอนการแปลง ฯ ขั้นต่อไป

กฎที่ 5 ในส่วนต้นของรหัสคำสั่งภายในแต่ละโอเปอร์เรชัน จะต้องทำการสร้างรหัสคำสั่งเพื่อประกาศตัวแปรที่ใช้ภายในโอเปอร์เรชันนั้นตามขั้นตอนดังต่อไปนี้

1. เลือกตัวแปรที่ทำหน้าที่รับค่า (ตัวแปรที่อยู่ทางซ้ายมือของเครื่องหมายมอบหมายค่า (=)) ทั้งหมดที่ไม่ซ้ำกันจากผลการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันนั้น โดยที่ตัวแปรนั้นจะต้องไม่เป็นพารามิเตอร์ของโอเปอร์เรชันดังกล่าว และชื่อของตัวแปรนั้นไม่มีเครื่องหมายมหัพภาค (.) ปรากฏอยู่
2. สร้างรหัสคำสั่งเพื่อประกาศตัวแปรสำหรับตัวแปรแต่ละตัวที่ได้จากขั้นตอนที่ 1 ตามรูปแบบดังนี้

ชนิดของตัวแปร ชื่อของตัวแปร;

กฎที่ 6 ทำการสร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ไว้ภายในแต่ละโอเปอร์เรชัน โดยให้อยู่ต่อจากรหัสคำสั่งสำหรับการประกาศตัวแปร ตามขั้นตอนดังต่อไปนี้

1. ประกาศตัวแปร “$_scenario” เพื่อใช้ระบุสถานการณ์ โดยมีชนิดของข้อมูลเป็น “int” และมีค่าเริ่มต้นเป็น 0 ดังนี้

```
int $_scenario = 0;
```

2. สร้างคอมเมนต์บอกตัวเลขที่ใช้แทนสถานการณ์ต่าง ๆ ในผลการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันนั้น โดยตัวเลขที่ใช้แทนแต่ละสถานการณ์จะต้องมีค่าไม่ซ้ำกัน และอธิบายตัวเลขแต่ละตัวด้วยชื่อของแผนภาพซีเควนซ์ที่สอดคล้องกับสถานการณ์นั้น ตัวอย่างของคอมเมนต์สำหรับผลการรวมกลุ่มลำดับการส่งเมสเสจที่ประกอบด้วย 2 สถานการณ์จากแผนภาพซีเควนซ์ที่ชื่อ “Normal flow” และ “Exceptional flow” เป็นดังนี้

```
/****************************************
  Scenario numbers :
        # 0 = Normal flow
        # 1 = Exceptional flow
****************************************/
```

3. สร้างรหัสคำสั่งสำหรับควบคุมการแสดงพฤติกรรมตามแต่ละสถานการณ์ โดยแต่ละการส่งเมสเสจในผลการรวมกลุ่มลำดับการส่งเมสเสจของโอเปอร์เรชันนั้นที่ไม่ได้สอดคล้องกับทุกสถานการณ์ จะต้องมีสเตทเมนต์ “if” ทำการทดสอบค่าของตัวแปร “$_scenario” ว่าตรงกับสถานการณ์ที่สอดคล้องกับการส่งเมสเสจนั้นหรือไม่ ถ้าตรงกันจึงจะทำรหัสคำสั่งสำหรับการส่งเมสเสจนั้น ตัวอย่างเช่น ถ้าการส่งเมสเสจ “a” มีที่มาจากสถานการณ์ที่แทนด้วยเลข 1 การส่งเมสเสจ “b” มีที่มาจากสถานการณ์ที่แทนด้วยเลข 0

และ 1 การส่งเมสเสจ “c” มีที่มาจากทุกสถานการณ์ จะสร้างรหัสคำสั่งได้ดังนี้

```
if ($_scenario == 1) (transformation of message “a”)
if ($_scenario == 0 || ($_scenario ==1) (transformation of message “a”)
(transformation of message “c”)
```

กฎที่ 7 ทำการสร้างรหัสคำสั่งสำหรับแต่ละการส่งเมสเสจให้สอดคล้องกับประเภทการส่งเมสเสจ โดยมีรูปแบบของรหัสคำสั่งสำหรับการส่งเมสเสจประเภทต่าง ๆ ในกรณีทั่วไปดังนี้

สำหรับการส่งเมสเสจระหว่างวัตถุ

```
if (เงื่อนไข) ตัวแปรรับค่าส่งกลับ = ชื่อวัตถุตัวรับเมสเสจ.ชื่อโอเปอร์เรชัน(รายการชื่อของพารามิเตอร์);
```

สำหรับการส่งเมสเสจภายในวัตถุ

```
if (เงื่อนไข) ตัวแปรรับค่าส่งกลับ = ชื่อโอเปอร์เรชัน(รายการชื่อของพารามิเตอร์);
```

สำหรับการส่งเมสเสจสร้างวัตถุ

```
if (เงื่อนไข) ชื่อวัตถุที่ถูกสร้าง = new ชนิดของวัตถุที่ถูกสร้าง(รายการชื่อของพารามิเตอร์);
```

สำหรับการส่งเมสเสจส่งกลับ

```
if (เงื่อนไข) return ค่าส่งกลับ;
```

ในกรณีที่การส่งเมสเสจใด ๆ ไม่มีการระบุเงื่อนไข จะต้องตัดส่วน “if (เงื่อนไข)” ออกไป

ในกรณีที่เงื่อนไขของการส่งเมสเสจเป็นแบบทำซ้ำ จะต้องแทนที่ “if (เงื่อนไข)” ด้วย “เงื่อนไข”

ในกรณีที่การส่งเมสเสจระหว่างวัตถุหรือภายในวัตถุใด ๆ ไม่มีตัวแปรรับค่าส่งกลับ จะต้องตัดส่วน “ตัวแปรรับค่าส่งกลับ =” ออกไป

ประวัติผู้เขียนวิทยานิพนธ์

นายชัชวีร์ ตั้งสายัณห์ เกิดเมื่อวันที่ 13 พฤศจิกายน พ.ศ. 2522 ที่จังหวัดราชบุรี สำเร็จการศึกษาระดับปริญญาบัณฑิต หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า จากจุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2544 หลังจากสำเร็จการศึกษา ได้ทำงานในตำแหน่งนักวิเคราะห์ซอฟต์แวร์ให้กับบริษัทเบ็กซ์คอม (ประเทศไทย) จำกัด เป็นระยะเวลาประมาณ 1 ปี และได้เข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ ณ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2545